PROVIDE QUALITY OF SERVICE FOR DIFFERENTIATED SERVICES NETWORKS BY POLICY-BASED NETWORKING

ZHANG YI

(*B.Eng. SJTU*)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2002

Acknowledgements

First of all, I would express my gratitude to my supervisor, Dr. Tham Chen Khong for his valuable and continuous guidance and encouragement in my research work.

I also would take the chance to express deep gratitude to my parents for their endless support and love.

I am immensely grateful to National University of Singapore for the award of a research scholarship to support my research.

Last but not least I would express my thanks Mr. Liu Yong, Mr. Michael Yuan, Mr. Gan Yung Sze and Mr. Jiang Yuming, for their help and suggestions.

Contents

List of Figures

List of Tables

List of Abbreviations

AF: Assured Forwarding
BA: Behavior Aggregate
BGP: Border Gateway Protocol
CBQ: Class Based Queuing
CBWFQ: Class Based Weighted Fair Queuing
CIM: Common Information Model
CLI: Command Line Interface
CORBA: Common Object Request Broker Architecture
COPS: Common Open Policy Service
COPS-PR: COPS for Policy Provisioning
COPS-RSVP: COPS usage for RSVP
DDNS: Dynamic Domain Name Service
DEN: Directory Enabled Networks
DHCP: Dynamic Host Configuration Protocol
DiffServ: Differentiated Services
DMTF: Distributed Management Task Force
DSCP: DiffServ CodePoint
EF: Expedited Forwarding
FTP: File Transfer Protocol
GUI: Graphical User Interface
HTTP: HyperText Transfer Protocol
IETF: Internet Engineering Task Force
IntServ: Integrated Services
IP: Internet Protocol
ISO: International Organization for Standardization
ISP: Internet Services Provider
LDAP: Lightweight Directory Access Protocol
MIB: Management Information Base
MLTTM: MultiLateral Two-Tier Mode
PBN: Policy Based Networking
PDP: Policy Decision Point
PDU: Protocol Data Unit
PEP: Policy Enforcement Point
PHB: Per Hop Behavior
PIB: Policy Information Base
PDB: Per Domain Behavior
PRC: PRvisioning Class
PRI: PRovisioning Instance
QoS: Quality of Service

QPM:	QoS Policy Manager
RAR:	Resource Allocation Request
RAS:	Remote Access Server
RPC:	Remote Procedure Call
RSVP:	Resource ReSerVation Protocol
SIBBS:	Simple Inter-domain Bandwidth Broker Specification
SLA:	Service Level Agreement
SLO:	Service Level Objective
SLS:	Service Level Specification
SNMP:	Simple Network Management Protocol
SMI:	Structure of Management Information
SPPI:	Structure of Policy Provisioning Information
TCB:	Traffic Conditioner Block
TOS:	Type Of Service
VPN:	Virtual Private Networks
WAN:	Wide Area Network

Summary

Policy-Based Networking (PBN) represents a revolution in network management, transforming it from passive network monitoring to active QoS provisioning, and from managing network elements to provide network services. It adds intelligence to networks and takes away the tedious configuration burden from network managers. Policy-based networking for providing QoS in DiffServ networks is receiving intensive research attention.

There are two models to enforce policy: policy-outsourcing model and policy-provisioning model. In this thesis, policy-based admission control and policy-based provisioning are implemented on a DiffServ testbed, which mainly provides intra-domain policy enforcement. There are three features in the implementations. Firstly, policy-based admission control is based on COPS-RSVP concepts with the Bandwidth Broker playing the role of policy client. Secondly, the exchange of policies between policy server and client is through COPS, a protocol tailored for policy use. Finally, the policies enforced on DiffServ devices are defined using DiffServ Policy Information Base.

For inter-domain policies, the two-tier model and bilateral agreements are now pervasive as the proposed building blocks for a policy framework. This thesis proposes a novel model MLTTM based on the two-tier model but taking a different approach to support multi-lateral agreements for end-to-end QoS in DiffServ networks. Comparisons between these two models are made and simulation results, which show how the proposed model improves load balancing and network efficiency, and how the information update interval and pricing scheme affect the reservation efficiency, are presented.

Chapter 1

Introduction

There is no authoritative definition in standard bodies such as IETF (Internet Engineering Task Force) or ISO (International Organization for Standardization) for policy-based networking currently since policy-based networking is still in its infancy. But there are lots of documents describing what policy-based networking can do and what benefits it can bring to network management. Based on these documents policy-based networking here is defined as follows:
Policy-based networking is the management of the networks by expressing the business goals as a set of rules and then enforcing the rules throughout the networks in a consistent way to make the network performance best meets the business goals.

The thesis will focus on policy-based networking providing QoS (Quality of Service) for DiffServ Networks, both in intra-domain and inter-domain area.

This chapter will briefly introduce the reasons for implementing policy-based networking, the benefits of the policy-based networking, the definition of policies and the requirements of a policy-based networking system.

1.1 Why Policy-based Networking?

The reasons for policy-based networking stem from the increasing complexities, diversities and enormities of the Internet. Although implementing policy-based networking is not limited to Internet and both the enterprise networks and ISPs can use policy-based networking to manage their networks efficiently, policy-based networking is mostly used to deal with the challenges brought by the development of the Internet.

The Internet has changed dramatically during the last decade, not only on the scale and the amounts of the users and hosts but also on the types of services it provides to the users. Apart from the traditional services such as FTP, E-mail, Telnet, World Wide Web and so on, new multimedia applications such as Voice over IP, Video Conferencing, Video and Audio Streaming evolve as the new services provided by ISPs.

Traditional Internet is a best-effort global informational system, which only provides non-differentiated services. For new emerging services and applications to function properly, some requirements such as bandwidth, delay, delay jitter and packet loss must be satisfied to meet the customers' satisfaction. But since the Internet is free and all kinds of the traffic are converged and compete for the resources on the same network, the mission-critical applications may not have enough resources to meet their requirements. Since over-provisioning of the network resources is not economically viable especially for WAN resources, Quality of Service is issued to address such problems. Quality of Service refers to the collective effect of service performances, which determines the degree of satisfaction of a user to the service. Also, since most of the private networks are connected to the Internet in a variety of ways, security is also an important issue to network managers. To both provide the QoS and security to the users and applications, network managers need a new paradigm to manage the network. This is where the policy-based networking can play its role.

In the past, network managers manage networks by manually configuring devices or interfaces one by one, which is tedious and error prone. Now network management has been shifted from point-to-point management to management of users and applications. To provide end-to-end QoS and consistent security policies under dynamic network traffic conditions, network managers need to manage the network from a holistic and dynamic view. With the numerous devices from different vendors in networks, performing one-by-one configuration of devices or interfaces to enforce the QoS or security mechanisms consistently end-to-end is error-prone or even impossible for network managers. The new network devices and technologies make even difficult to find qualified and experienced personnel to deploy the complicated mechanisms through manual device configuration. Also, because customers are concerned with the services from the business view, network management should focus on users and applications, which needs to define rules from a high level, shielding the network details from network managers, thus easing the management of the network.

1.2 Benefits of Policy-based Networking

The design philosophy of the Internet is decentralized control combined with intelligent endpoint control, which is exactly the opposite of that of the telephone network, where intermediary equipment provides all the intelligence. While networks based on such a design have been proven relatively simple and robust, they are expected to be more complicated and fragile when supporting services customized for different users and applications. The trend is to blend the two features of Internet and Telephone Network by providing appropriate simplicity and robustness for a network's core while adding some complexities to the edge. Policy-based networking is an efficient way to control the added complexities.

One of the benefits of policy-based networking is that it allows network managers to define the policies according to business goals, which is then enforced across an entire network. The policies are defined centrally through a console but enforced in a distributed fashion through the automatic configuration of network devices. Such architecture is suitable for applying policies either enterprise wide or within a single domain. For example, the traffic from and to the CEO's office should have higher priority than that of others since he/she pays a more important rule in the company. The traffic of account department should get a better treatment at the end of every month and the sensitive data may be prohibited from accessing by the applications from other departments. All these business goals can be achieved by policy-based networking.

Another benefit of policy-based networking lies in that it can automate many configuration tasks of the network devices such as routers and switches, which in the past had to be performed by network managers manually. This is especially useful for implementing QoS mechanisms. By centrally defining the policies and applying them according to the roles that the network devices play in the network, the network devices can be configured consistently without efforts. By using the directory as policy store or link the policy store with the stores of other information such as users, applications, DHCP etc., policy-based networking can be integrated with other management functions such as inventory control, accounting, monitoring etc.

1.3 What is a Policy?

According to [1], from a broad view, policies can be defined as a definite goal, course or method of action to guide and determine present and future decisions. Within the context of the networking, policies can be defined as a set of rules to administer, manage and control access to network resources [2]. This also implies the discriminated treatments of different users or applications, which are the reason to implement QoS or VPN.

A rule is a basic building block of a policy-based system. It is the binding of a set of actions to a set of conditions, where the conditions are evaluated to determine whether the actions are performed [2]. Policy actions define what is to be done to enforce a policy rule, when the conditions of the rule are met. Policy Actions may result in the execution of one or more operations to affect and/or configure network traffic and network resources. Policy conditions define the necessary states and/or prerequisites in order to apply policy actions. When the policy conditions are evaluated to be true, then the rule should be enforced. A policy is often described as: *IF condition THEN action.* A rule's condition can be expressed as either an ORed set of ANDed sets of statements (disjunctive normal form) or an ANDed set of ORed sets of statements (conjunctive normal form)[2].

When implementing a policy-based networking system, network managers must consider the different abstraction levels at which policies should be expressed. The different policy levels are shown in Figure 1.1 according to [3].

At the highest level, policies are defined from a user-friendly English-like syntax to hide the details of device configurations from network managers, allowing the network managers to create network-wide policies that define and control services. In QoS context, high-level business policies are used to express the requirements of the different applications, and prioritize which applications get better services when the network is under congestion.

At the middle level, device-independent policies translate business policies into a set of generalized operational and configuration policies that are independent of specific devices but dependent on the particular set of QoS or security mechanisms. Device-independent policies provide a common layer of abstraction for managing multiple devices with different capabilities from different vendors.

At the lowest level, device-dependent policies translate device-independent policies into ones that are specific for a particular device. It enables the devices from different vendors with different capabilities to understand and enforce the policies received.

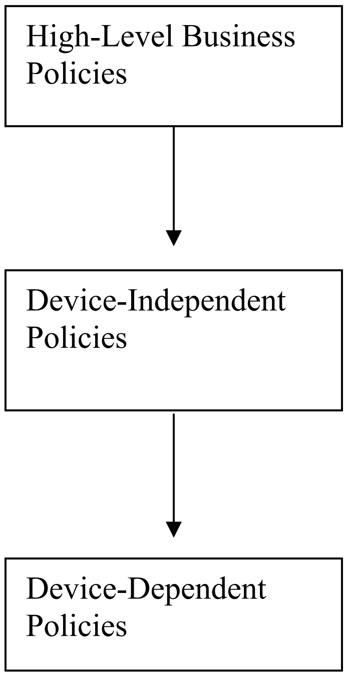

High-Level Business Policies
Device-Independent Policies
Device-Dependent Policies

Figure 1.1: The Policy Abstraction Level

For example, according to business needs, the traffic of accounting department should receive better services than others at the end of every month. The network manager can define a high level business policy through a console centrally such as "The traffic of accounting department should receive highest priority at the end of every month". This policy can then in turn be mapped to a device-independent policy "The traffic of network

137.132.12.0 should receive EF services for last three days of every month”. At the lowest level the device-independent policies will be translated into device-dependent configuration commands to automatically configure the ingress and egress routers and other related devices across the networks.

1.4 Requirements of Policy-Based Networking

For a policy-based networking system to properly and efficiently control networks and applications, it must meet some requirements [4].

First, there must be an information model for network elements, network services, networks and clients of the network. An information model is a representation of the entities that make up the managed environment and the way they interact with each other. An information model describes the types of devices and services that make up the network, what components each service or device includes, how the devices and services relate to each other. The DEN (Directory Enabled Networks) model, which was first initiated by Microsoft and Cisco and then was incorporated with CIM (Common Information Model) by DMTF (Distributed Management Task Force) [59], is the best-known information model for policy-based networking.

Also, there must be a policy specification language that can represent business requirements and functions in a vendor and device-independent manner. An information model is used to define objects but a policy language is used as a common language to represent these objects so these objects can be shared among applications. Take an analogy with daily life, an information model describes an object type “House”, but “House” can be expressed in different languages such as English, Chinese or French and so on. People from different country cannot understand the meaning the “House” expressed by a foreign language. To make the “House” understood by all of the people, it is better the “House” is spelt by one common language understood by all of the people.

Additionally, policy-based networking system must be scalable for policy administration, management, conflict resolution and distribution. With the rapid development of the Internet, network managers must manage ever-increasing numbers of network devices with ever-increasing numbers of configurations. The scalability issues thus include not only the framework of the system but also policy representation and storage as well as policy retrieval methods and policy distribution protocols.

1.5 Contributions of the Thesis

The thesis makes contributions in the areas of (1) policy-based admission control and policy provisioning in the intra-domain area to provide DiffServ QoS, and (2) on MLTTM (MultiLateral Two-Tier Model) which addresses inter-domain resource management for QoS.

Implementation of Policy-based Admission Control

In order to provide QoS for certain applications, it must be guaranteed that the network will not be congested during the life of applications. One method is to reject some flows in order that the resources are available and enough to accepted flows. Policy-based admission control is used to perform this admission function in terms of the requirements of users, applications, time etc., and the availability of the network resources. Currently, there are two types of policy-based admission controls. In IntServ architecture RSVP reserves resources through router while policies in RSVP message outsource to a Policy Decision Point (PDP). In DiffServ architecture Bandwidth Broker plays the role of Policy Decision Point to perform resource allocation for the aggregates of its own domain.
The first type is used for per-flow based admission control, which can provide guaranteed QoS but can only be used in a small network since it has scalability problems and demands all the devices along the route should be RSVP aware. The second type solves the scalability problems since bandwidth brokers can be distributed in each domain and the number of their managed clients is limited. But bandwidth brokers are designed mainly for QoS purposes, business or organization related policies are better left out of bandwidth broker.
The uniqueness of the implementation in this thesis is:

- A bandwidth broker plays the role of policy enforcement point (PEP) and a policy server provides the policy decision beyond the capacities of the BB, so the BB can concentrate on its main duty.
- The concepts of the RSVP (Resource Reservation Protocol) especially COPS (Common Open Policy Service) for RSVP are borrowed for policy-based admission control.
- COPS protocol is implemented based on Vovida COPS stack [40] for policy exchanging between BB and policy server.

Implementation of Policy Provisioning

Traditionally, devices are manually configured, which is tedious and error-prone. Policy provisioning provides a method to automatically and consistently configure devices. Policy provisioning is suitable for configuring the devices whose configuration is relatively steady while the dynamic QoS requests are left to edge devices that provide policy-based admission control capability. Devices used in DiffServ core networks have this steady feature since the core devices only perform forwarding behaviors. When the access lists in edge devices are relatively steady (for example not change for several days), policy provisioning is also suitable for edge devices. Compared to other implementations, the implementation in this thesis has following unique features:

- COPS for Policy Provisioning (COPS-PR) is implemented as the policy exchange protocol between policy server and policy client.
- The provisioning policy data takes the data format in DiffServ QoS Policy Information Base, which aims to be the standard format for QoS policy data in DiffServ networks.
- Policy server also plays the role of policy proxy to map the DiffServ QoS PIB to Cisco router configuration commands since currently there are no PIB-aware devices.
- *Expect* [49] is used to perform the auto-telnet to automatically configure the Cisco routers

MultiLateral Two-Tier Model (MLTTM)

Service Level Agreement (SLA) plays an important role in achieving end-to-end QoS for DiffServ networks. A dynamic SLA is achieved by inter-domain policy negotiation between two BBs (Policy Server) in different domains. The prevalent model is Two-tier model proposed in [23]. This model is featured by bilateral SLAs between neighboring domains, which implies the BB in one domain has no knowledge of non-neighboring domains. But since in the Two-tier model the route is determined without considering the network status along the route, the load is not balanced and resource usage is not efficient. MLTTM is proposed in this thesis based on the two-tier model but have the following unique features:

- QoS based source routing is used by the BB to determine the passed domains.
- BGP table in edge routers are dynamic and updated by the request from other BBs.
- A distributed common data store for resource availability and price information.
- Pricing information is determined by local BB in terms of resource availability.
- The contract is in nature multilateral and consistent.
- Load is balanced across the network and network resource utilization is efficient.

1.6 Organization of the Thesis

This thesis is organized into five chapters.

In this chapter some basic background knowledge of policy-based networking is introduced.

Chapter 2 will discuss the QoS architecture including IntServ and DiffServ but with an emphasis on DiffServ, the QoS mechanisms of the DiffServ elements, the architecture of policy-based networking (PBN), the components of the PBN and the protocols in the PBN.

Chapter 3 will discuss COPS for RSVP (Resource Reservation Protocol) and policy-based admission control implementation based on COPS-RSVP concepts.

Chapter 4 will discuss COPS for Policy Provisioning, the DiffServ QoS PIB (Differentiated Services Quality of Service Policy Information Base) and also the implementation of the COPS-PR.

Chapter 5 will introduce a novel resource reservation model called MLTTM based on policy-based networking with an emphasis on inter-domain policy exchanges since the previous two chapters focus on intra-domain and also present some simulation results showing the load-balancing advantages and factors affecting the reservation performance based on this model.

Conclusion and future research issues will be addressed in chapter 6.

Chapter 2

Architecture of QoS Policy-based Networking

There are the two kinds of Internet architecture proposed by IETF: IntServ and DiffServ, which provide the foundation for a discussion of architecture of PBN for QoS.

2.1 Integrated Services Architecture

The Integrated Services Architecture (IntServ) is defined by the Internet Engineering Task Force (IETF) to transit the Internet into a robust-service communication infrastructure for supporting the transport of audio, video, real-time and elastic data traffic [5]. It is a reservation-based architecture in that network resources are allocated according to an application's request.

The IntServ provides the ability for the applications to choose among multiple, controlled levels of delivery services for their data packets [6]. To support such a capability, two conditions must be satisfied:

- The individual network elements along the path of the application flow must support the QoS mechanisms for the reservation of the resources.
- There must be a protocol to communicate the application's requirements to network elements along the path and convey QoS management information. Besides the protocol must keep the flow status during the packets transmission period.

To meet the first condition, IETF defines two QoS controlled services, namely Guaranteed Quality of Service [7] and Controlled Load Network Element Service [8]. To meet the second requirement, IETF defines Resource ReSerVation Protocol to set up resource reservation along the path although manual configuration of the devices through SNMP also is feasible.

The Resource ReSerVation Protocol is a signaling protocol defined by IETF to set up resource reservations along the path of flows in IntServ network. RSVP is a reservation setup and control protocol that provides a type of circuit-emulation on IP networks [9]. The concepts of its use with COPS in COPS-RSVP are borrowed in the implementation of Policy-based Admission control in Chapter 3.

Currently many network device companies such as Cisco, Fore Systems, and Sun Microsystems have RSVP implementations on their routers [10]. But since RSVP must keep per-flow states for each reservation, its implementation faces serious scalability problems in large networks. Differentiated Services as an alternative to IntServ was proposed by IETF as the standard of next generation Internet architecture.

2.2 Differentiated Services Architecture

The Differentiated Services architecture (DiffServ) is based on a simple model [11] where traffic entering a network is classified and possibly conditioned at the boundaries of the network and assigned to different behavior aggregates, which is identified by a single DiffServ codepoint (DSCP). Within the core of the network, packets are forwarded according to the per-hop behavior (PHB) associated with the DSCP. By implementing complex classification and conditioning functions only at boundary nodes and by applying PHB to aggregates of traffic that have been appropriately marked using the DS field in the IP headers, DiffServ has successfully resolved scalability problems, which limit the IntServ implementation in large networks. In this way, service provisioning and traffic conditioning policies are sufficiently decoupled from the forwarding behavior within the core network, which allows the core network elements only to perform forwarding service and thus assure the performance of the core devices are not affected by implementing the QoS mechanisms.

2.2.1 DiffServ Domain

A DS domain is a DS-capable domain, a contiguous set of network elements which operate with a common set of service provisioning policies and PHB definitions [11]. A DS domain has a well-defined boundary consisting of DS boundary elements, which classify and possibly condition ingress traffic to ensure that packets which transit the domain are appropriately marked to select a PHB from one of the PHB groups supported within the domain. Network elements within the DS domain select the forwarding behavior for packets based on their DSCP in IP header, mapping that value to one of the supported PHBs using either the recommended DSCP to PHB mapping or a locally customized mapping. A DS domain normally consists of one or more networks under the same administration.

2.2.2 Differentiated Services Codepoint

Differentiated services can be constructed by a combination of the following:

- Set bits in an IP header field at network boundaries to mark the packets into one aggregate.
- Use marked bits to determine how to forward packets within the core.
- Condition the marked packets at network boundaries in accordance with the requirements or rules of each service.

The above mentioned bits in IP head is called the DS field, which is intended to supersede the existing definitions of the IPv4 TOS octet [12] and the IPv6 Traffic Class octet [13]. Six bits of the DS field are used as a DSCP while the other two bits currently are unused.

2.2.3 Per Hop Behavior

Per Hop Behavior (PHB) is the externally observable forwarding behavior applied at a DS-compliant node to a DS behavior aggregate [11]. A set of one or more PHBs can only be meaningfully specified and implemented simultaneously, due to a common constraint applying to all PHBs in the set such as a queue servicing or queue management policy. A PHB group provides a service building block that allows a set of related forwarding behavior to be specified together.

Which PHB is chosen for a packet depends on the DSCP in the packet IP header. All DSCPs must be mapped to some PHBs and such mappings may be standardized or locally defined. Currently there are two PHB groups defined by IETF, namely Expedited Forwarding PHB [14] and Assured Forwarding PHB group [15].

The EF PHB forwards a particular DiffServ aggregate where the departure rate of the aggregate's packets from any DiffServ node must equal or exceed a configurable rate. The EF traffic should receive this rate independent of the intensity of any other traffic attempting to transit the node. Such a service appears to the end points as a virtual leased line. Several types of scheduling algorithm can be used to implement the EF PHB such as priority queuing, CBQ (Class Based Queuing) and CBWFQ (Class Based Weighted Fair Queuing). The recommended DSCP for EF is 101110.

The AF PHB group provides delivery of IP packets in four independent forwarded AF classes. For each AF class, an IP packet can be assigned one of three different levels of drop precedence. AF provides an assurance that IP packets with rate not exceeding the profile have a high probability to be forwarded. AF PHB group is a means for a provider DS domain to offer different forwarding assurances for the packets from the customer domains. There are four AF classes defined with each AF class in each DS node allocated a certain amount of resources. Within each AF class IP packets can be marked with one of three-drop precedence. When congestion occurs, a DS node tries to protect packets with lower drop precedence by dropping the packets with high drop precedence first. The recommended corresponding DSCPs for the AF PHB group are shown in Figure 2.1:

	AF1X	AF2X	AF3X	AF4X
Low Drop Prec (1)	001010	010010	011010	100010
Medium Drop Prec (2)	001100	010100	011100	100100
High Drop Prec (3)	001110	010110	011110	100110

Figure 2.1: AF PHBs and Corresponding DSCPs

2.2.4 Quality of Service Mechanisms in a DiffServ Node

Figure 2.2 shows the QoS mechanisms in a DiffServ-capable node (router) according to [16].

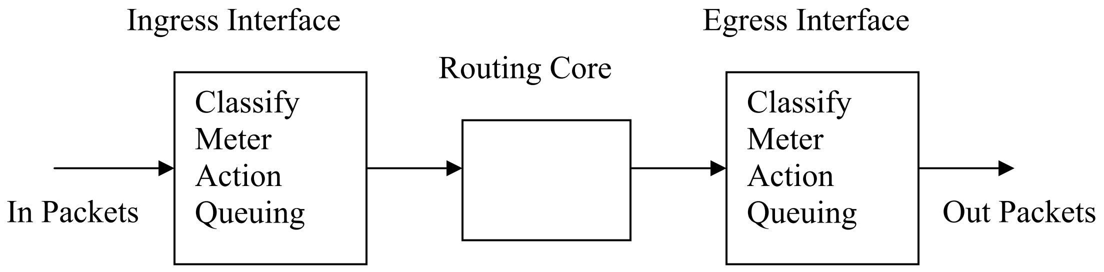


Figure 2.2: DiffServ Router Major Functional Block

The QoS mechanisms in a DS router includes: traffic classification elements; metering elements; action element which includes marking, absolute dropping, counting and multiplexing; queuing elements which includes the capabilities of algorithm dropping and scheduling; and traffic conditional blocks combined by the above functional elements.
An ingress router often performs the packets classification and conditioning functions.

The combination of meter, marker, shaper and dropper is called a Traffic Conditioner, which performs the major function of an ingress edge router. Figure 2.3 is a logical view of the classifier and traffic conditioner block.

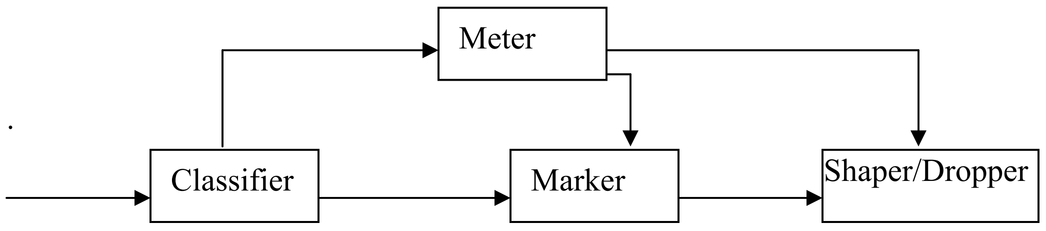


Figure 2.3: Logical Views of a Packet Classifier and Traffic Conditioner

The packet classifiers classify the packets in a traffic stream into an aggregate based on content of some portion of the IP header. There are two types of classifiers: Behavior Aggregate (BA) classifier based only on DSCP; Multi-Field (MF) classifier based on more header fields such as source address, destination address, DS field, protocol ID, source and destination port numbers or other information such as incoming interface.

A traffic meter measures the temporal properties of the traffic selected by a classifier against a traffic profile. The traffic in a profile may be marked a DSCP by a marker to indicate the service it should receive and the traffic out of a profile may be remarked, dropped or reshaped.

2.3 Architecture of Policy-based Networking

A policy-based networking system typically includes these components: Console, Management Tools, Repository, Policy Decision Point (PDP) and Policy Enforcement Point (PEP). Together these components are organized into a system to perform a large number of tasks in order to manage today's complex networks. These tasks consist of policy editing, policy storage, policy distribution, policy enforcement and so on. Because these components may reside in different network devices, protocols are also needed to be defined to facilitate the data transfer.

The policy console serves as an interface between the system and network manager. A network manager can use the policy console to edit policies and monitor the status of the network performance to ascertain the enforcement of the policies.

The policy management tool works in conjunction with and behind the policy console to validate the rules in the first step, checking semantics and data types and check the conflicts between the rules when they are combined into policies.

The policy repository, which can be a directory and a database, is used to store the rules and policies required by the system.

The policy decision point (PDP) is responsible for accessing the policies in the repository and making decisions based on those policies.

The policy enforcement point (PEP) implements the decisions from PDP to process traffic. Policy enforcement points include network devices such as routers, VPN security gateways, firewalls and even servers, hosts etc.

There are two types of architectures of policy-based networking: two-tiered policy management architecture and three-tiered policy management architecture. These two types of architectures are shown in Figure 2.4 and Figure 2.5 respectively. Both architectures have pros and cons and are used in some real-life implementations. For example, Alcatel, Fore Systems and IBM released their two-tiered products while 3Com, Cisco, IP Highway and Lucent have implemented their three-tiered systems.

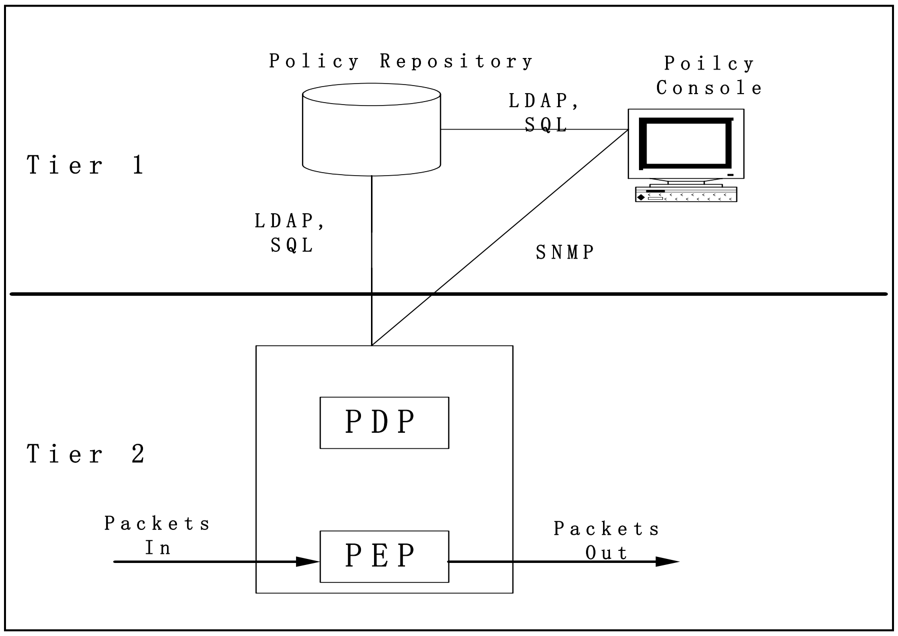


Figure 2.4: Two-Tier Policy Architecture

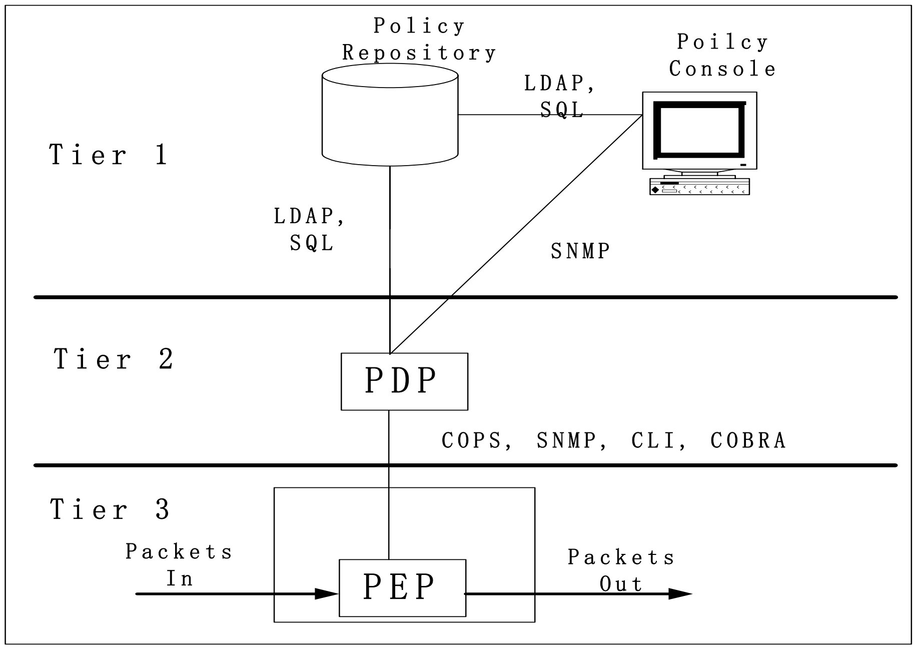


Figure 2.5: Three-Tier Policy Architecture

In a two-tiered architecture, a PDP resides physically on the same device as the PEP, eliminating the need for an intermediary communication protocol such as COPS. In a three-tiered architecture, PDPs and PEPs are physically separate devices, which require some protocols such as COPS, SNMP, CLI or CORBA to exchange policy data between them. The main advantage of the two-tiered architecture is the elimination of the communication protocol between PDP and PEP. The link between PDP and PEP is entirely proprietary and custom-written for that particular device.

One of the strengths of the three-tiered architecture lies in that it simplifies the management of legacy system that is not policy-aware. In this architecture, a policy-translator or proxy can be used with the PDP to convert policy decisions into information such as SNMP or CLI command that a legacy device can understand. Another strength of the three-tiered architecture is that it provides a more scalable solution than the two-tiered model since a middle layer is provided between the PEP and policy repository.

2.3.1 Policy Console and Policy Management Tool

Since policy console and policy management tools often coexist on the same host, they are discussed together in this section.

Policy console is the interface between the policy-based networking system and network managers. Usually the policy console is a Graphical User Interface, which provides network manager the functions of policy editing, policy presentation and feedback.

Policy management tools work together with and behind the console. They provide the functions of rule translation, rule validation, global conflicts resolution [17], notifying PDPs, storing policies in repository, and retrieving policies from repository. Many of these functions are driven by policy console. For example, when policies are created, the management tools translate policies into a form compatible with the schema and storage requirements of the policy repository and handle the communications between policy console and policy repository. Also network managers can retrieve existing policies or policy rules to edit them or store the newly created policies in the policy repository through the policy management tools.

There are two types of policy conflicts [18]. Global conflicts are those based on the properties of the policy and not the specific devices to which the policy may apply. Local policy conflicts are those based on specific devices. Normally the local policy conflicts are checked by PDP or the Policy Proxy relegated to the PDP while the policy console and policy management tools are responsible for checking global conflicts.

A global conflict occurs when all the conditions of the policies are satisfied but one or more of the actions of one policy conflicts with one or more of the actions of another policy. The actions of two policies conflict when they cause different operations to be applied to the same resource. For example, if policy A specifies that all traffic from accounting department should receive EF service and policy B specifies that all ftp traffic should receive best-effort service. These two policies conflict to each other since according to policy A, ftp from accounting department should receive EF while according to B it should receive best-effort treatment.

2.3.2 Policy Repository

The basic function of a policy repository is to store policies so that whoever or whatever needs to use these policies can retrieve them from it. The retrieval may be from a network manager using a policy console or from a PDP on behalf of the PEPs it controls.

In order to accomplish the above-mentioned tasks, firstly a policy repository must maintain the integrity of the data it receives and stores, which means that any transaction that changes data must be complete and validated or else the affected data is restored, or rolled back to its previous state. Also, usually a policy repository that appears as a single logical entity within the system may be distributed across multiple locations to deal with scalability or reliability problems. In such a situation, the policy repository should include methods for replicating data among different locations and ensure that data replication does not affect the distribution of the policies. Additionally, the policy repository should include security mechanisms to prevent data from unauthorized access and changes.

The policy repository may also include some extended functions such as storing other data that the policy-based networking system requires or include pointers to other data stores that store such data.

There are two types of policy repositories for storing policies: directory and relational database. Both have pros and cons, which are described in details in [18].

2.3.3 Policy Decision Point

A Policy Decision Point (PDP), also called policy server, is a logical entity that makes policy decisions for itself or for other network elements that request such decisions [19]. As a newly introduced component of network management, it not only performs much of the work of translating policies into information that the network devices can understand to enforce the policies, but also serves as the central point for distributing policies to policy enforcement points. To fulfill such functions as mentioned above, a PDP must access the policies stored in a policy repository, but also be aware of the capabilities of the PEPs it controls. There are a number of protocols used between the PDP and policy repository, and between the PDP and PEPs it controls. A PDP can use either LDAP or SQL to retrieve policies from repository. SNMP, COPS, telnet/CLI or CORBA can be used for PDP and PEP communications.

The PDP must verify that not only PEPs receive the proper policy information but also the devices actually enforce the policies, which requires the PDP has the knowledge of the PEP’s capabilities.

The extended functions of a PDP include serving as an aggregation point of device events or accounting information before such information makes its way to the policy console, and exchanging policy information with its peer PDPs (Since in a large network, one PDP may be insufficient for controlling a large number of devices).

2.3.4 Policy Enforcement Point

Policy Enforcement Point (PEP) is a logical entity that enforces policy decisions [6], which is the underlying layer of equipment that needs to be managed. The PEP controls the traffic on the networks, helping the network support the business goals and policies set at higher levels in the policy-based networking system. Since QoS is one of the main forces driving the policy-based networking, it is no doubt that routers and switches are two types of most important PEPs, which are also the main PEPs addressed in this thesis. There are also other types of PEPs, either used for QoS or security. The PEPs are divided into classes which include: routers and switches, web switches, traffic shapers, firewalls, Remote Access Servers, Virtual Private Networks, DHCP and DDNS (Dynamic Host Configuration Protocol [20] and Dynamic Domain Name Service [21]), Servers and end-user hosts etc. Most of these PEPs are now policy-unaware.

2.4 Provide End-to-End QoS Using Policy-based Networking for DiffServ Networks

From a customer's view, QoS is observed from the receiver's host, not from the individual elements, which means all the networks along the path must cooperate to achieve the desired goal. This demands QoS to be end-to-end. To provide end-to-end QoS in IntServ architecture, RSVP is proposed as a signaling protocol to reserve the network resources for an application flow [6]. Once the reservation is accepted and the resources are allocated according to the reservation, end-to-end QoS is achieved. Because of the scalability problems in IntServ, DiffServ as an alternate architecture is proposed by IETF. But in DiffServ architecture because the resources are allocated on a coarse-grained per-class basis and the aggregate traffic of a class consists of many flows from various sources to various destinations, end-to-end QoS is difficult to achieve. Many investigators proposed new management scheme to address this problem. For instance, [22] proposed Clearing House architecture which models the Clearing House in banking industry to facilitate resource reservation over multiple network domains and perform local admission control. A two-tier resource management model for DiffServ users is addressed in [23]. Currently the two-tier model is more prevalent and this model will be explained in this section. In chapter 5, MultiLateral Two-Tier Model (MLTTM) model is proposed for inter-domain policy-based networking. But before further explaining about the two-tier model, some terminologies have to be elaborated first.

2.4.1 DS Domains

A DS domain is a contiguous portion of the Internet over which a consistent set of differentiated service policies are administered in a coordinated fashion. A differentiated services domain can represent different administrative domains or autonomous systems, different trust regions, different network technologies (e.g., cell/frame), hosts and routers, etc [24].

The devices in a DS domain are classified into edge devices and core devices. The edge devices mainly perform the classification and traffic conditioning function while the core devices only perform the packets forwarding PHB according to DSCP in the packet headers. In this way, the complexities are pushed to the edge while the core is kept simple, which is also compatible with the design philosophy of the Internet. The policy enforcement is mainly performed on the edge devices.

2.4.2 Bandwidth Broker

The concept of QoS broker has been addressed in [25], which is an end-point entity, meaning the resource allocation is negotiated between two end-hosts. Different from the QoS broker, the bandwidth broker (BB) is used to negotiate and allocate resources on the network elements along the path. Bandwidth brokers are designed to be configured with organized policies, keep track of the current allocation of the marked traffic and interpret new requests to mark the traffic in terms of the policies and current allocation. They are intended to be used in resource allocation for end-to-end QoS with fewer states since they need not keep per-flow states.

A BB actually is a PDP in one DS domain, which has two responsibilities, intra-domain resource allocation and inter-domain resource negotiation. The intra-domain resource allocation includes partitioning the local marked traffic allocations and setting up the leaf routers within the domain. The inter-domain negotiation includes managing the messages that are sent across boundaries to adjacent DS's BBs and making the service request and reservation with those BBs.

2.4.3 Service Level Agreements

From a general view, a Service Level Agreement (SLA) is a formal definition of the relationship that exists between two organizations, usually a supplier of services and its customers [26].

From the viewpoint of QoS, a Service Level Agreement is a service contract between service providers and their customers in terms of network levels of QoS (throughput, loss, delays and jitter) and times of availability, method of measurement, consequences that service levels are not met or the stated traffic profiles are exceeded by the customers, and all costs involved.

A Service Level Specification (SLS) or Objective (SLO) documents the details of the operational characteristics for an SLA. It partitions an SLA into individual objectives that can be mapped into policies for execution. It may consist of expected throughput, drop probability, latency, traffic profiles, marking and shaping services provided and so on.

2.4.4 A Two-tier Resource Management Model

After the explanation of the terminologies used, the two-tier resource management model is discussed next.

As an example, Figure 2.6 demonstrates a typical scenario in which a host in an enterprise network sends traffic across the ISP backbone to another host in another enterprise network. There are three DS domains of which two are enterprise DS domains and one belongs to the ISP. The routers directly connected to the end-host are called leaf routers. The routers in the enterprise networks at the boundary between the enterprises and ISP networks are called edge routers. The routers in the ISP networks between the enterprises and the ISP networks are called border routers. All these three types of routers are the PEPs which the BB controls.

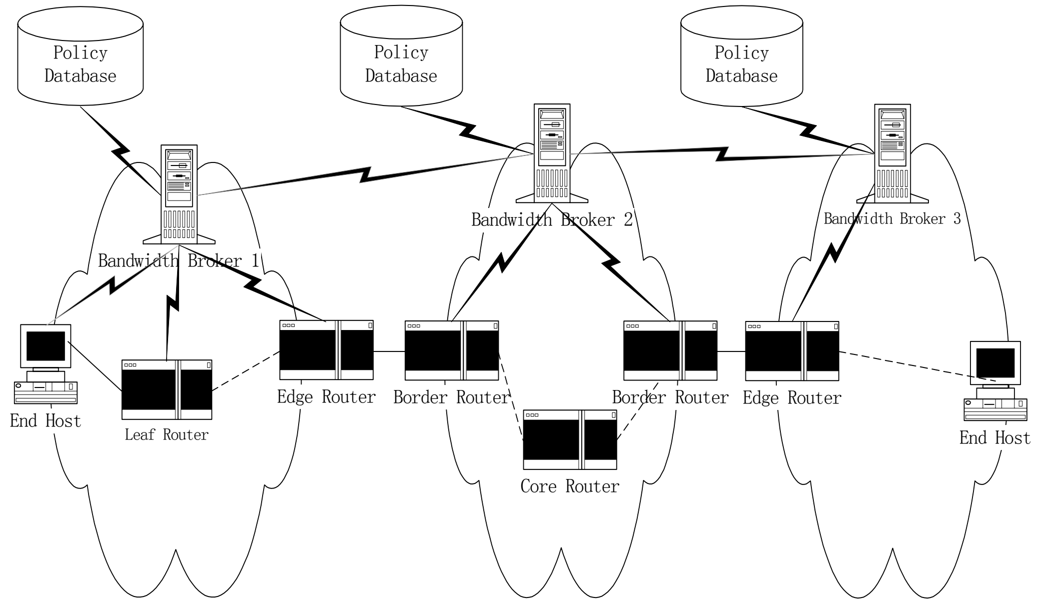


Figure 2.6: Two-tier model for end-to end QoS

Although in Figure 2.6 the source or destination is illustrated as an end-host, bear in mind the same procedure also applicable to the aggregate. The resource reservation procedure is described as follows:

- Source host sends a request message to BB1 in its local domain.
- BB1 makes the admission control decision based on its resource availability, policies and the SLA with BB2. If resources are sufficient, request is accepted, otherwise is rejected.
- If accepted, BB1 transfers this request to BB2; BB2 makes the admission control decision based on policies, the SLA between itself and BB3. If resources are sufficient, request is accepted, otherwise it is rejected.
- If accepted, BB2 transfers this request to BB3; BB3 makes the decision based on policies and the availabilities of its local resources. If accepted, a message will be sent back along the reverse path to source and the resources are allocated.
- If the SLA is dynamic, a BB may reject the request and re-negotiate a new SLA with its adjacent BB, then use this new SLA for making decisions as before.

2.4.5 Current Implementations of Policy-based Networking

During the past several years, many vendors have started to implement policy-based networking (PBN) and currently many products have been offered in the market.

Cisco has released their QoS Policy Manager (QPM) as a solution to the policy-based networking. Since Cisco is the biggest vendor of the WAN device market and from Cisco's IOS 11, many new features to support the QoS have been added to their devices. Their products are especially necessary to keep an eye on.

As a full-featured QoS policy system, QPM enables differentiated and signaled services (i.e. IntServ), ensuring consistent QoS across the network through application and user-aware, centralized policy management.

There are also evaluations of the QPM using real life traffic. [27] has reported an evaluation performed by Netigy Corporation of the effectiveness of Cisco QPS combined with IOS in providing preferred network access for business-critical PeopleSoft financial users. The results show either using Priority Queuing (PQ) or Class Based Weighted Fair Queuing, the PeopleSoft traffic gets large improvements in throughput compared to the throughput under congestion without policies applied.

Many other vendors also have their products released. Among them are: Orchestream Enterprise 2.0, Allot's NetPolicy, ExtremeWare Enterprise Manager, HP's OpenView PolicyXpert, IPHighway's QoSmaster and Open Policy System, Lucent's RealNet Rules, Nortel's Optivity Policy Services 1.0, Spectrum Management Policy Based Network Management Suite and so on.

The Real-World labs at the University of Wisconsin Madison have tested some vendors' products to decide which one is the best bet based on the long-time strategy factors, such as how vendors planned to integrate with legacy tools for network management, what standards-based protocols the vendors claimed to embrace, the multi-vendor support and the vendor's chance of delivering on the strategy [58]. Among them Cisco QPS and Orchestream Enterprise 2.0 play the leading roles. The current implementations have following features:

Almost all the implementations support *layer 3* conditions except that ExtremeWare Enterprise Manager 2.0 does not support *source subnet* and *DiffServ/TOS marking*, and

RealNet Rules does not support *DiffServ/TOS marking*. All the implementations support *layer 4* conditions. These conditions are compatible to the DiffServ PIB in chapter 4. Except Cisco QPM 1.1 and ExtremeWare Enterprise Manager, all other implementations support temporal indicators such as *time of day/date, day of week or month* etc. Although some vendors provide support for other layer conditions, the common support for *layer 3* and *layer 4* is enough for deploying policy in IP networks.

For the actions part, it is found that no condition is supported by all the implementations. All implementations except ExtremeWare Enterprise Manager 2.0 and RealNet Rules support *Color using IPTOS/Differv* and *Committed Access Rate*, which is compatible to DiffServ PIB. But some actions in DiffServ PIB such as *Random Early Detection* and *Weighted Random Early Detection* are not supported by most of the implementations. This is mainly because many of the current network devices do not provide such capabilities and there is no common-accepted standard on the QoS support of the network devices.

All the implementations tested in [58] are based on Three-Tier Policy Architecture in Figure 2.5 since they all claim to support the devices from other vendors. The test results show they all work best on their own devices (except Orchesteam and IPHighway since they have no their own hardware) and most of them also support Cisco routers, the dominant devices in WAN. There is no consensus as to which protocol should be used to exchange policies between multiple servers, policy repositories and the devices being managed. CLI, COPS, LDAP, SNMP are all used in the current implementations. Because of the proprietary in products, for CLI to be used, they must be tested on every version of the switch and router software. If the device vendor changes even one command syntax between software versions, the PBN vendor has to account for those changes. LDAP is used by some vendors such as Lucent mainly because LDAP is the most common way vendors access stored user and resource data and its scalability and fault tolerance. But its flaws such as slow write to the directory have been attacked by other vendors. SNMP is used mainly because of its popularity in network management. Although SNMP provides reasonable device-monitoring capabilities, it has poor device-configuration functionalities. COPS does not have the drawbacks of the above protocols. Because COPS [28] and its extension COPS-PR [29] are now standardized by IETF, although now not implemented by many vendors, it will dominate the market in the long term.

On the PEP part, Cisco's IOS 11 began to introduce new QoS capabilities to manage the traffic on LAN and WAN links. Nortel Networks' BayRS 13.2 introduced many new QoS-specific commands.

Note that all the mentioned products are limited in intra-domain implementations. Policy-based networking is far from full-fledged; so many topics still need thorough investigation.

Chapter 3

Policy-based Admission Control and Implementation

3.1 Common Open Policy Service (COPS)

Since COPS is used as the policy exchange protocol in the implementation, it is described in detail in this section.

The IETF Resource Allocation Protocol WG has defined the Common Open Policy Service as a scalable protocol that can be used to exchange policy information between a policy server (PDP) and its clients (PEP) [28].

3.1.1 Features of COPS

As a simple query and response protocol with extensible capabilities, the COPS protocol includes the following features [28]:

- COPS employs a client/server model where the client PEP sends requests or reports to the remote server PDP and the PDP returns decisions back to the PEP.
- COPS is more reliable than SNMP since it uses TCP as its transport protocol for the exchange of messages between the PDP and the PEP while SNMP uses UDP.
- COPS is extensible since it follows the object-oriented design and objects are self-identifying, eliminating the need for modifications to the COPS protocol itself.

- COPS is secure in nature since it provides message level security for authentication, replay protection and message integrity.
- COPS protocol is stateful, which means its previous request/decision states are remembered by both the PDP and the PEP
- COPS allows the server to push configuration information to the client besides the pull method.

3.1.2 COPS Message Types

In [28] the interactions and exchange contents of COPS are classified into ten messages:

1. Request (REQ) PEP → PDP

   REQ is used to report the current local state of the PEP with the given client type and handle, and to request the PDP to send decisions back based on the state contained in the message objects.
2. Decision (DEC) PDP → PEP

   The DEC message is used to send decisions to the PEP, either as a direct answer to the REQ message or unsolicited. A handle is provided to identify the target and the attached objects specify the actual decision.
3. Report State (RPT) PEP → PDP

   The RPT message is used by the PEP to communicate to the PDP its success or failure in committing the PDP's decision, or to report an accounting related change in the state.
4. Delete Request State (DRQ) PEP → PDP

   Sent from the PEP, this message is used to indicate to the PDP that the state identified by the client handle is no longer available or relevant.
5. Synchronize State Request (SSQ) PDP → PEP

   This message indicates that the PDP wishes the PEP to re-send the state associated with the client handle.
6. Client Open (OPN) PEP → PDP

This message indicates that the PEP will send at least one REQ message associated with the client type. This message can be sent any time during a COPS connection.

7. Client Accept (CAT) PDP → PEP

   This message is used to indicate an OPN message is accepted by the PDP

8. Client Close (CC) PDP ↔ PEP

   This message informs the receiver to close the current connection between a client type and the PDP.

9. Keep Alive (KA) PDP ↔ PEP

   This message is transmitted to verify that the connection is up and working.

10. Synchronize Complete (SSC) PEP → PDP

    This message informs the PDP that the synchronization initialized earlier by the PDP with a SSQ message is complete.

3.1.3 COPS Format

Each COPS message consists of the COPS common header followed by a number of types of objects [28]. The common header is shown in Figure 3.1.

0		1	2	3
Version	Flags	Op Code	Client-type	
Message Length (octets)				

Figure 3.1: COPS Common Header

The fields in the header are explained as follows:

- Version: 4 bits, COPS version number, current version is 1.
- Flags: 4 bits, defined flag values.
- Op Code: 8 bits, the COPS message type. The Op Code and its corresponding format type is show in table 3.1.

Operation Code	1	2	3	4	5	6	7	8	9	10
Message Type	REQ	DEC	RPT	DRQ	SSQ	OPN	CAT	CC	KA	SSC

Table 3.1: The Operation Code and Corresponding Message Type

- Client-type: 16 bits. The Client-type is the type of the policy client. For COPS usage for RSVP the client type is 1. In a KA message, the Client-type must be set to 0. Client-types from 0x8000-0xFFFF are enterprise specific.
- Message Length: 32 bits, the length of the message in octets, which includes the COPS header and all encapsulated objects and must be aligned on 4 octet intervals.

The objects encapsulated in COPS follow the object format in Figure 3.2 [28]:

0	1	2	3
Length (octets)		C-Num	C-Type
Object Contents			

Figure 3.2: COPS Object Format

The fields are explained as follows:

- Length: 16bits, the number of octets of the object (including the object header).
- C-Num: 8 bits, the classes of information contained in the object. There are 16 classes defined in [28], which are shown as follows:

  1 = Handle

  2 = Context

  3 = In Interface

  4 = Out Interface

  5 = Reason code

  6 = Decision

  7 = LPDP Decision

  8 = Error

9 = Client Specific Info

10 = Keep-Alive Timer

11 = PEP Identification

12 = Report Type

13 = PDP Redirect Address

14 = Last PDP Address

15 = Accounting Timer

16 = Message Integrity

- C-type: 8 bits, the subtype or version of the C-Num.
- Object Contents: the information contained in the object.

3.2 Two Models of COPS for Policy Control

As a query/response protocol, COPS supports two common models: Outsourcing and Provisioning [29].

In COPS Outsourcing model, the events occurring at the PEP (for example, a manager initiates a video conferencing with a remote branch office) drive the PEP to issue an instantaneous policy request. In response to the policy request from the PEP, an external PDP makes decisions for the PEP. COPS-RSVP is an example of the Outsourcing model, which will be addressed later in this chapter.

In COPS Provisioning model, there are no one-to-one relationships between PEP events and PDP decisions. The PDP may proactively provision the PEP in response to external events (for example, a network manager clicks a button on the policy console to deploy the defined policies; a router in the domain is out of service etc.), PEP events (for example, the configuration or the capability of a PEP has been changed) and any combination of such events. Provisioning may be performed in bulk, (e.g. entire router QoS configuration) or in portions (e.g., updating a DiffServ marking filter) [29]. When the devices have been configured, they will independently dealing with the flows (no need to request a decision from a PDP). COPS for Policy Provisioning (COPS-PR) is an example of COPS Provisioning model, which will be addressed in the next chapter.

3.3 Policy-based Admission Control and COPS-RSVP

3.3.1 Policy-Based Admission Control

The IETF working groups have made extensions to current IP architecture to support the specific QoS for applications and users. The IntServ, DiffServ and MPLS [30] are three proposed architecture in Internet society.

In the IntServ model, providing the end-to-end QoS requires explicit signaling of the QoS requirements from the end-points and provisioning of admission and traffic control at IntServ routers. RSVP [9] is the standard signaling protocol for supporting IntServ. Network managers need to be able to monitor, control and enforce use of the network resources and services based on policies derived from the criteria such as the identity of users and applications, bandwidth requirements, security considerations and time of day etc. However, the IntServ model does not include an important aspect of admission control-the policy support. The admission control component in IntServ devices such as a router only takes into account the resource reservation request and available capacity to decide whether or not to accept the QoS request. To include the policy elements in admission control process in IntServ architecture, [19] specifies a framework for providing policy-based control over admission control decisions. Although the document focuses on policy-based admission control using RSVP as the QoS signaling mechanism, the framework is general and can be used in other QoS contexts. This is the reason policy-based admission control framework and COPS-RSVP concepts can be used in DiffServ policy-based networking implementation, which will be discussed later in this chapter.

When using COPS as the policy exchange protocol between PDP and PEP, Policy-based admission framework is based on COPS Outsourcing model. Under this model, the PEP uses the following steps to perform admission control [19]:

- When a local event or message invokes PEP for a policy decision, the PEP creates a request including information from the message and appropriate policy elements.

- The PEP may consult a local configuration database to identify a set of policy elements to be evaluated locally and also a set of policy elements to be evaluated remotely. The set of the policy elements to be evaluated locally are passed to a Local PDP (LPDP) and the decisions of the LPDP are collected.
- The PEP then passes the set of policy elements to be evaluated remotely and the local decisions of LPDP to the PDP.
- The PDP returns the final decision to the PEP.

3.3.2 RSVP Extensions for Policy Control

It is reasonable to expect the RSVP be companied by mechanisms for controlling and enforcing access and usage policies. But the current RSVP Functional Specification [9] only addresses the interface to admission controls based on resource availabilities. Since in [9] the RSVP data format leaves a placeholder for policy support in the form of the POLICY_DATA object, as extensions to [9], [31] describes a set of extensions to RSVP using this placeholder for supporting policy-based admission control.
POLICY_DATA objects are carried by RSVP messages and contain policy information. All policy-capable nodes can generate, modify or remove policy objects. Thus the RSVP can be used to exchange policies across the path. The POLICY_DATA format is shown in Figure 3.3.

0	1	2	3
Length		POLICY_DATA	C-Type (1)
Data Offset		0 (reserved)	
Option List			
Policy Element List			

Figure 3.3: POLICY_DATA format in RSVP message

The field meaning is shown as follows:

- Length: 16 bits, the size in octets of the POLICY_DATA object.

- POLICY_DATA=14, 8 bits.
- C-Type=1, 8 bits.
- Data Offset: 16 bits, the offset in bytes of the data portion (from the first bytes of the object header).
- Reserved: 16 bits, always 0.
- Option List: Variable length, the list of options.
- Policy Element List: variable length, the contents of policy elements.

3.3.3 COPS Usage for RSVP

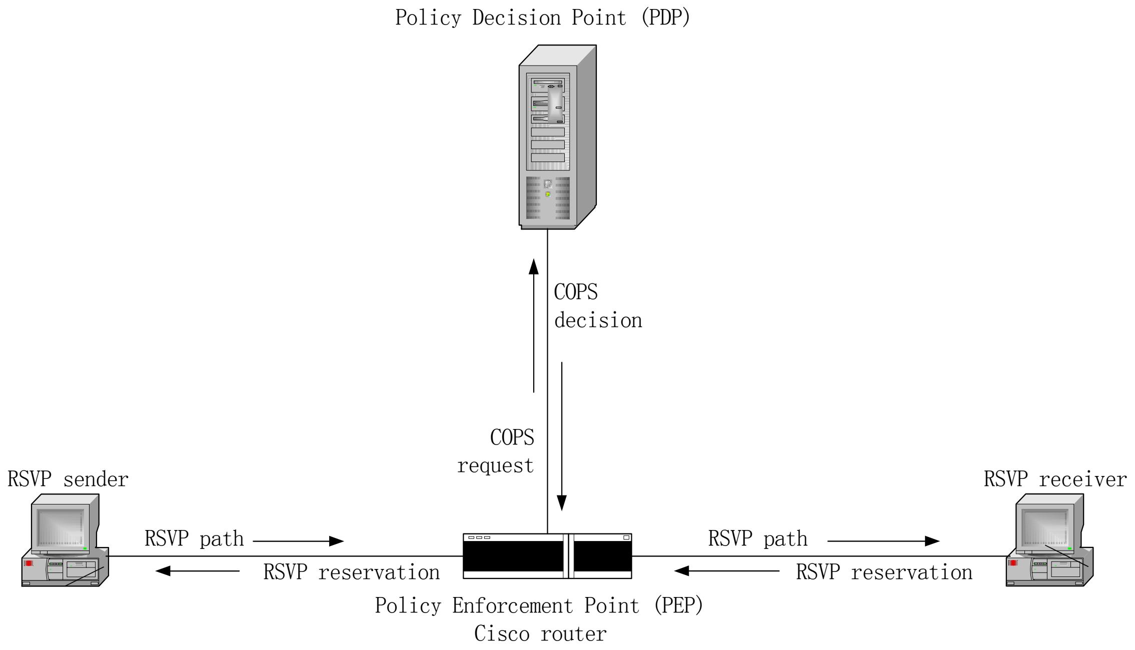


Figure 3.4: Sample Arrangements of COPS-RSVP

After Policy-based Admission Control and RSVP Extensions for Policy Control has been discussed, now COPS usage for RSVP (COPS-RSVP) [32] can be described. Because COPS is a protocol to communicate the traffic policy information with network devices and RSVP is a signaling protocol to reserve network resources, COPS-RSVP gives network manager centralized monitoring and controlling of resource reservation.

Since only the concepts of COPS-RSVP are lent to policy-based networking in DiffServ to perform admission control, it is necessary only to consider how the COPS-RSVP procedure is performed between a PEP and a PDP.

Figure 3.4 illustrates the sample arrangements for COPS-RSVP in Cisco networks [33]. The COPS-RSVP works as follows:

- When RSVP signaling message arrives at a router, the router sends a COPS-RSVP request message to policy server.
- The policy server queries the policy database, makes the decisions and then sends back the decision to the router.
- The router processes the message as instructed.

3.4 Implementation of Policy-based Admission Control

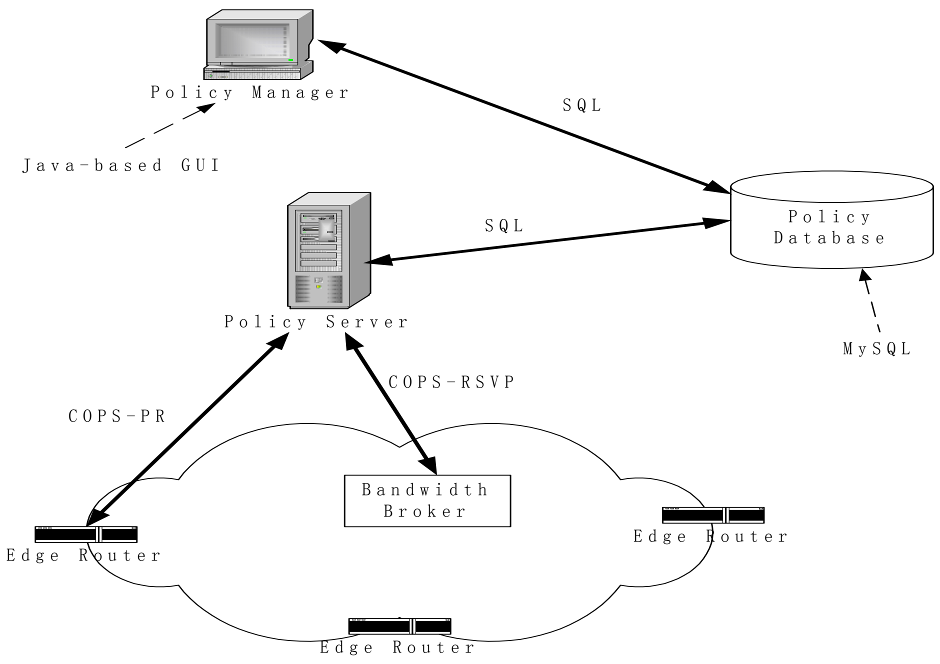


Figure 3.5: The Implementation of Policy-based Networking

Figure 3.5 illustrates the design of the implementation. In this design, the policy-based networking includes four components: a Java-based GUI console and management tools (i.e. Policy Manager) running on Windows 2000, a MySQL [34] database server as the policy repository running on Windows 2000, a policy server running on RedHat Linux

6.2, a policy client as the agent of PEP device running on Linux 6.2. There are two types of PEPs: one is the bandwidth broker, and another is edge router. Normally the bandwidth broker plays the roles of PDP, allocating the resources in response to requests from the customers. But since the functions of bandwidth brokers focus on bandwidth allocation and checking the bandwidth availability, more complex and manual control of the resource allocation tasks should be left to a higher layer above it. In this model the bandwidth broker plays the role of PEP. When there is a request from the customers, the request will be forwarded to the policy server to check if this request can be accepted according to the policies stored in the policy repository.

The protocol used to exchange policy information between bandwidth broker and policy server is the COPS protocol under the policy-outsourcing model, and between the policy server and edge router is the COPS protocol under the policy-provisioning model. The discussion of the COPS-PR will be deferred to next chapter and the implementation of COPS usage for policy admission control will be addressed in this chapter. The exchange of policy information between policy server and policy database is by Structure Query Language (SQL) and the same language is between policy manager and policy database.

In the implementation of policy-based admission control, since the policy rule is simple, the GUI console is not necessary. Instead a command line mode of the MySQL server is used to input the policy into the policy database, so GUI console is not discussed now but left to the next chapter.

3.4.1 Design of the COPS-RSVP Implementation

The topology of the test-bed is shown in Figure 3.6. Figure 3.7 illustrates the program modules of the policy-based admission control implementation. The design includes four modules: *RarClient, RsvpRar, Policy Server* and *CheckRar*. All the four modules reside on the machine Policy Sever & Client (192.168.1.3 as shown in Figure 3.6). *Policy Repository* resides on the machine Policy Manager (192.168.1.2 as shown in Figure 3.6). *RarClient* and *RsvpRar* together play the role of PEP and *Policy Server* and *CheckRar* play the role of PDP. Their functions are summarized as follows:

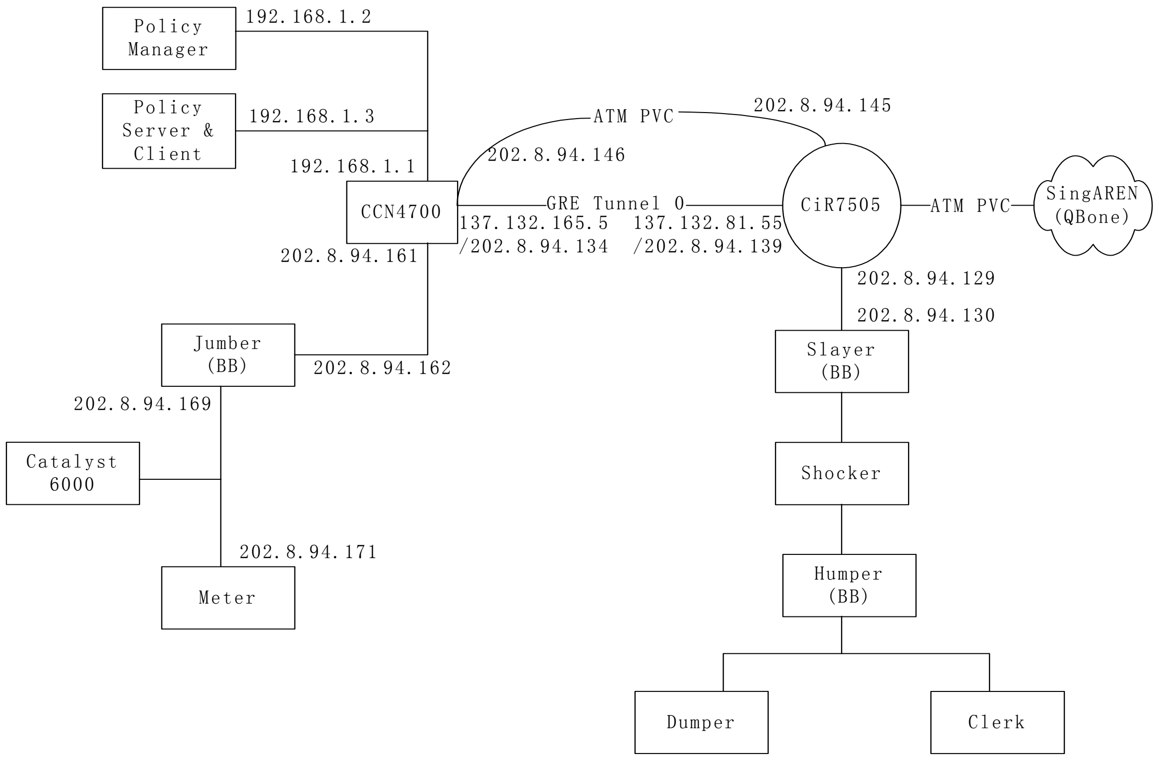


Figure 3.6: The Topology of Testbed

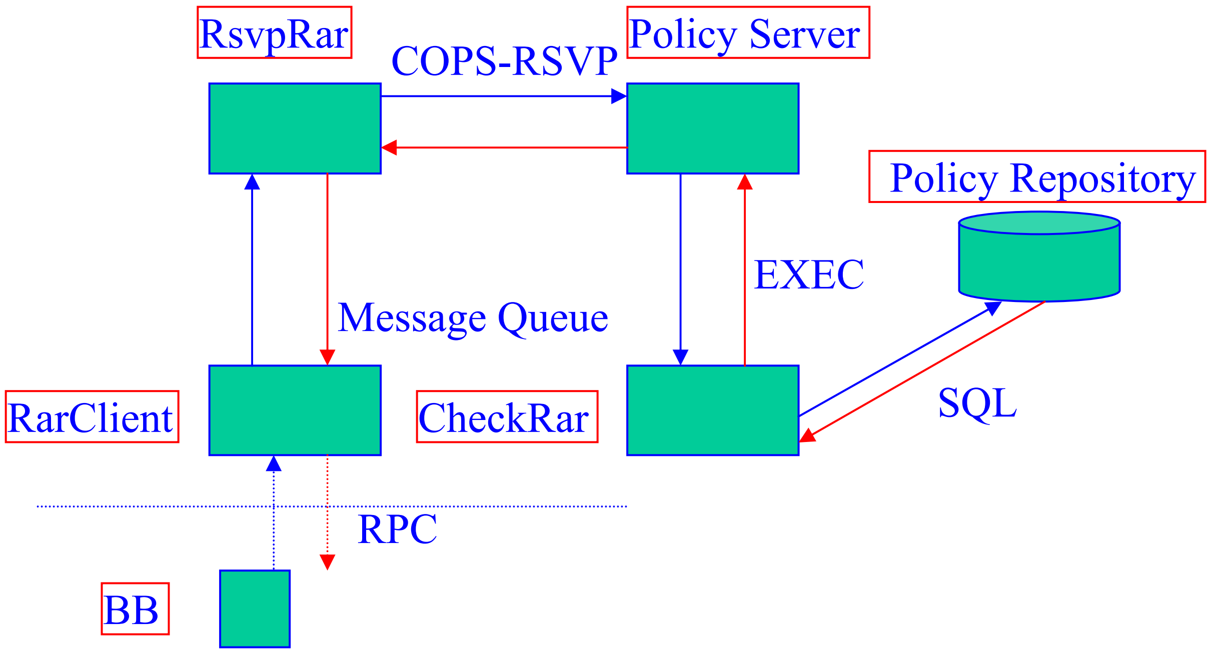


Figure 3.7: Policy-based Admission Control Implementation Diagram

- *RarClient*: plays an agent of bandwidth broker (Humper in Figure 3.6) to receive a request from and relay a reply to BB through *RPC* (Remote Procedure Call); issues a request to *RsvpRar* and receives a decision through a *Message Queue*.
- *RsvpRar*: as a client transfers the request to *Policy Server*, gets the reply from the *Policy Server* and also keeps the client and server connected through Keep Alive message.
- *Policy Server*: accepts the request from the policy clients (*RsvpRar*), forks a process *CheckRar* to query the policy database, gets the decision from *CheckRar* and transfers the decision back to its clients.
- *CheckRar*: queries the policy database, makes the decision and returns the decision to the *Policy Server*.

All the modules are coded using C/C++ on Linux 6.2 except the MySQL server, which is running on Windows 2000. The main coding files in this implementation are *rarclient.c*, *checkrar.c*, *RsvpRar.cxx*, and *COPSProcessor.cxx,* which is the main component of *Policy Server*. All other files in the implementation are borrowed from COPS1.2.0 with minor changes. The detailed implementation of the modules is described in following sections respectively.

3.4.2 Implementation of RarClient Module

The function of the RarCient is to perform the Resource Allocation Request (RAR), acts as an agent of bandwidth brokers. It receives the policy request from a BB through Sun's RPC (Remote Procedure Call).

The Internet2 QoS Working Group Draft has defined the Internet2 (i.e. QBone) as an inter-domain test-bed for DiffServ that seeks to provide the higher-education community with end-to-end services in support of emerging advanced networked applications [35]. In QBone architecture, a bandwidth broker [36] centrally manages the resource allocations. It also negotiates the inter-domain resource allocations with its neighboring domains. [35] has defined a simple inter-domain Bandwidth Broker protocol for resource negotiating between neighboring domains. [37] defines the format of Simple Inter-domain Bandwidth Broker Specification (SIBBS) PDUs. [38] specifies a SIBBS version

1 RAR for providing the Premium service. The same data as used in the [38] are used in this implementation, but not abide by the same Type, Length and Value (TLV) rules for RAR data. Instead the RAR format is defined as the following data structure:

```
struct rar_request
{
        unsigned long int bbaddr;
        unsigned int seqnum;
        unsigned long int souraddr;
        unsigned long int sourmask;
        unsigned long int destaddr;
        unsigned long int destmask;
        unsigned long int iraddr;
        unsigned short int phb;
        unsigned short int dscp;
        unsigned long int qpspr;
        unsigned long int qpsbs;
        unsigned short int mtu;
        unsigned long int starttime;
        unsigned long int stoptime;
        unsigned long int rarflags;
}
struct rar_reply
{
        int result;
        char reason[128]
}
```

Once a request is received from a BB, it is encapsulated and put into a message queue. There is also another queue for placing the reply from *RsvpRar* module.

3.4.3 Implementation of RsvpRar Module

RsvpRar accepts the data in the message queue and encapsulates the data into a COPS request message to *Policy Server*. Since COPS is tailored for policy use, it has efficiency and advantages compared to SNMP, which is evaluated by some researchers such as in [39]. COPS is used to exchange information between *RsvpRar* and *Policy Server*. Different from COPS-PR, COPS using COPS-RSVP concept is a different client type and follows a different policy model. The base COPS protocol stack implementation used is downloaded from Vovida - a communication community dedicated to provide a forum for open source software used in datacom and telecom environments. COPS1.2.0 [40] as the Open Source contribution provides the most of the functionalities of the COPS protocol. Its COPS Stack implementation follows the IETF standards [28]. The Vovida implementation of COPS stack hierarchy of classes is shown as follows:

Hierarchy of classes

- *COPSBadDataException*
- *COPSData*
- *COPSHeader*
- *COPSIpv4Address*
- *COPSIpv6Address*
- *COPSMsg*
    - *COPSSyncStateMsg*
    - *COPSReqMsg*
    - *COPSReportMsg*
    - *COPSKAMsg*
    - *COPSDeleteMsg*
    - *COPSDecisionMsg*
    - *COPSClientOpenMsg*
    - *COPSClientCloseMsg*
    - *COPSClientAcceptMsg*

- *COPSMsgParser*
- *COPSObjBase*
    - *COPSTimer*
        - *COPSKATimer*
        - *COPSAcctTimer*
    - *COPSReportType*
    - *COPSReason*
    - *COPSPepId*
    - *COPSPdpAddress*
        - *COPSIpv6PdpAddress*
            - *COPSIpv6PdpRedirAddr*
            - *COPSIpv6LastPdpAddr*
        - *COPSIpv4PdpAddress*
            - *COPSIpv4PdpRedirAddr*
            - *COPSIpv4LastPdpAddr*
    - *COPSObjHeader*
    - *COPSInterface*
        - *COPSIpv6Interface*
            - *COPSIpv6OutInterface*
            - *COPSIpv6InInterface*
        - *COPSIpv4Interface*
            - *COPSIpv4OutInterface*
            - *COPSIpv4InInterface*
    - *COPSIntegrity*
    - *COPSHandle*
    - *COPSError*
    - *COPSDecision*
        - *COPSLPDPDecision*
    - *COPSContext*
    - *COPSClientSI*
- *COPSPolElementHeader*

- *COPSPolicyElement*
    - *COPSPolicingSuppElement*
    - *COPSMarkingSuppElement*
- *COPSPrObjBase*
    - *COPSPrefixPrID*
    - *COPSPrID*
    - *COPSPrErrorPRID*
    - *COPSPrError*
    - *COPSPrEPD*
- *COPSPrObjParser*
- *COPSRoleCombination*
    - *COPSOutRoleCombination*
    - *COPSInRoleCombination*
- *COPSTransceiver*
- *COPSUtil*

Based on the COPS stack the Vovida provided, policy client *RsvpRar* and *Policy Server* is coded using GNU C++ on Linux 6.2. The *RsvpRa*r works as follows:

- *RsvpRar* opens a connection to *Policy Server* using the *COPSClientOpenMsg*.
- *RsvpRar* periodically sends the *COPSKAMsg* to *Policy Server*.
- After receiving the *COPSClientAcceptMsg* from *Policy Server*, *RsvpRar* creates two message queues using the same keys as in *RarClient*.
- *RsvpRar* takes the data from first queue, encapsulates the data into the *ClientSI* part of *COPSRsqMsg*, sets the *Client Type* as 1 (RSVP) and sends the message to *Policy Server*.
- *RsvpRar* accepts the decisions from the *Policy Server* and transfer the decision to *RarClient*.
- *RsvpRar* sends a *COPSReportMsg* to *Policy Server*.

3.4.4 Implementation of Policy Server Module

The functions of the *Policy Server* are to accept the request from the policy clients (*RsvpRar*), fork a process *CheckRar* to query the policy database, get the decision from *CheckRar* and transfer the decision back to its clients. The codes of *Policy Server* are based on the COPS Stack provided in the COPS1.2.0, but are modified on the functions the server used to perform the request message processing. The *Policy Server* works as following:

- The *Policy Server* opens a thread pool and wait for the connection from the clients.
- If a *COPSOpenMsg* is received and there are available threads in the pool, the *Policy Server* will send a *COPSAcceptMsg* to the client and also create a thread to deal with all the messages it receives from the specific client.
- If the message from the client is a *COPSKAMsg*, the thread will send back a *COPSKAMsg* to the specific client.
- If the message is a *COPSReportMsg*, the thread does nothing except printing the message on screen.
- If the message is a *COPSCloseMsg*, the thread will close the connection with this client and return the thread to the pool.
- If the message is a *COPSReqMSg*, the thread will check the client type first. If the client type is 1 (RSVP), the thread will take out the data from the message, parse the data and fork a child process to execute the process *CheckRar* and then encapsulate the results from the child to a *COPSDecisionMsg*.

3.4.5 Implementation of CheckRar Module

The functions of *CheckRar* are to query the policy database, make the decisions and return the decisions to *Policy Server*. Although the structure *rar_request* includes many fields, *CheckRar* does not check all these fields since they are implementation-dependent.

In implementation, the table *rateLimitTable* stores the policy information for querying, which includes the following fields:

- *filterSourAddr*: a string representing the source address of the traffic.
- *filterDestAddr*: a string representing the destination address of the traffic.
- *dscp*: a string representing the DiffServ Code Point the traffic requests.
- *peakRat*e: a string representing the peak rate of the traffic.
- *burstSize*: a string representing the burst size.
- *MTU*: a string representing the Maximum Transmission Unit.
- *startTime*: a string in the *year:month:day:hour:minute:second* format representing the start time of the traffic.
- *stopTime*: a string in the same format as *startTime* representing the stop time of the traffic.

CheckRar works as follows:

- *CheckRar* parses the data received from *Policy Server*.
- It checks field by field if the value is accepted according to the values in the *rateLimitTable*.
- If all the values are accepted, *CheckRar* will return a "0" to the *Policy Server*, otherwise a negative integer and a string representing the rejection reason will be returned.

3.4.6 An Example of Policy-based Admission Control

An example of policy-based admission control running on test-bed is shown as follows:

The video traffic from CCN video server to the clients in CIR wants to request the EF service across the network from 8:00am, 14th Feb, 2002 to 9:00am, 15th Feb, 2002. The traffic parameters are as follows:
Source Address: 192.168.1.2
Destination Address: 202.8.94.139 (Ingress Router of CIR domain)

Peak Rate: 2 M

Burst Size: 4000 Bytes

Maximum Transmission Unit: 1500 Bytes

The policy made by a policy manager is:

The traffic from CCN video server to the clients in CIR is allowed to send no more than 2.5Mb EF traffic in February 2002.

The policy represented in the *rateLimitTable* is shown as:

filterSourceAddr: 192.168.1.2

filterDestAddr: 202.8.94.139

dscp: 46

peakRate: 2000

burstSize: 4000

MTU: 1500

startTime: 2002:2:1:0:0:0

stopTime: 2002:3:1:0:0:0

After querying the policy data, *CheckRar* decided that the request should be satisfied and a “0” message was sent back to BB.

3.5 Conclusions

In this chapter, COPS as the standard policy exchanging protocol is introduced, mainly on its message types and formats. Then two models of policy control are defined. COPS-RSVP as an example of the Outsourcing Model is described in details. COPS-PR as an example of Provisioning Model is left to the next chapter. A Policy-based Admission Control system based on the concept of COPS-RSVP is implemented.

This system is suitable for processing the policy requests from the bandwidth brokers in the same domain whereas resource allocation is left to bandwidth brokers. Because Policy-based Admission Control is a request-response process, the system could quickly allocate and release the resources as needed. The testing shows that such a request-response process normally needs several seconds. Whether the QoS can be guaranteed depends on how the bandwidth broker works. A benefit of this model is the separation of functionalities between BB and policy server in dealing with the requests. The BB concentrates on the resource allocation purely based on resource availability and resource requests from applications. Policy server only deals with the requests from BB to enquiry the business or organization related policies. Such a separation of the functionalities will eliminate the scalability problems a policy sever may faced in RSVP since in many case there is only one BB in each domain. The BB also reduces the burden of policy server to process requests thus the main functionalities of the policy server are left to COPS-PR, which will be addressed in the next chapter.

Chapter 4

Policy Provisioning and Implementation

It is mentioned in the last chapter there are two models of policy-based networking: outsourcing model and provisioning model. The last chapter focuses on the implementation of outsourcing model, this chapter will address provisioning model.
Firstly the Common Open Policy service (COPS) protocol for support of the policy provisioning (COPS-PR) will be described. Then Differentiated Services Quality of Service Policy Information Base (DiffServ QoS PIB) will be discussed. Finally implementation of policy-based networking based on COPS-PR and DiffServ QoS PIB is presented.

4.1 COPS Usage for Policy Provisioning (COPS-PR)

COPS-PR extends COPS to efficiently support the policy provisioning [29].

4.1.1 Features of COPS-PR

COPS-PR has following features to support policy provisioning [29]:

- COPS-PR can efficiently transport attributes, large atomic transaction data and flexible error reporting.
- Policy configuration can be effectively locked even from local console configuration because there is only one connection between the PEP and PDP via COPS-PR.
- COPS-PR is defined as a real-time event-driven communication protocol and does not need polling mechanism between PEP and PDP.

4.1.2 COPS-PR Procedure

Different from COPS-RSVP, COPS-PR has a "push" nature from PDP to PEP; typically the interaction between a PEP and its PDP follows some basic steps. Since the Vovida implementation of COPS1.2.0 has already been introduced, the terms in COPS Stack will be used to describe the steps of the COPS-PR [29]:

- When a device boots, it sends a *COPSOpenMsg* to its PDP.
- After authentication, the PDP sends back a *COPSAcceptMsg* to PEP.
- PEP sends periodical *COPSKAMsg* to keep the connection between PEP and PDP alive.
- The PEP sends information about itself in a *COPSRsqMsg* to PDP.
- The PDP retrieves all the policies relevant to the device from the policy data store, makes the necessary translation to device-dependent policies and sends the policies in a *COPSDecisionMsg* to PEP.
- The PEP uses the policies received to do the configuration of the devices and thereafter sends a *COPSReportMsg* to its PDP.

4.1.3 COPS-PR Extensions to COPS

In COPS, a request message or report state message from PEP to PDP optionally includes a Client Specific Information (ClientSI) object and a decision message from PDP to PEP includes a Decision(s) object. In case of policy provisioning, the ClientSI in a request or report state message is called Named ClientSI since it includes named information about the module, interface or functionality to be configured [28]. Similarly, the Decision(s) in a decision message is called Named Decision Data. The Request message (REQ), Decision message (DEC) and Report state message (RPT) can be represented in ASN.1 [41] semantics as follows respectively [29]:

- <Request Message> ::= <Common Header>

  <Client Handle>

<Context = config request>
*(<Named ClientSI>)
[<Integrity>]

- <Decision Message> ::= <Common Header>
  <Client Handle>
  *(<Decision>) | <Error>
  [<Integrity]

  <Decision> ::= <Context>
  <Decision: Flags>
  [<Named Decision Data: Provisioning]
- <Report State> ::= <Common Header>
  <Client Handle>
  <Report Type>
  *(<Named ClientSI>)
  [<Integrity>]

COPS-PR has defined several new objects within the COPS Named ClientSI and Decision Data objects. But before these new objects are described, two new terms will be introduced first: PRC and PRI.

PRC represents Provisioning Class, which is a type of policy data, while PRI means Provisioning Instance, which is an instance of PRC. Their relationship is similar to that of class and object in the Object Oriented programming methodology.

4.1.3.1 COPS-PR Protocol Objects

The format for the new object is shown in Figure 4.1 [29]:

0	1	2	3
Length		S-Num	S-Type
Encoded Object			

Figure 4.1: Basic Format for COPS-PR objects

The meanings of the fields are shown below:

- Length: 16 bits, the length of the object in octets.
- S-Num: 8 bits, identify the general purpose of the object.
- S-Type: 8 bits, identify the specific encoding rules used for this object. For Basic Encoding Rules (BER i.e. TLV encoding) now used in COPS-PR, S-Type=1.
- Encoded Object: variable length data padding to 32 bits boundary, representing data encoded according to S-Type.

4.1.3.1.1 Complete Provisioning Instance Identifier (PRID)

The PRID object is used to carry the identifier of a PRI. Its format and field values are shown in Figure 4.2 [29]:

0	1	2	3
Length (variable)		S-Num (PRID=1)	S-Type (BER=1)
Encoded Instance Identifier			

Figure 4.2: Data Format for PRID

4.1.3.1.2 Prefix PRID (PPRID)

The Prefix PRID is used in Remove Decision operation to optimally perform bulk operation with the common prefixes of the PRIDs. Its format and field values are shown in Figure 4.3 [29]:

0	1	2	3
Length (variable)		S-Num (PPRID=2)	S-Type (BER=1)
Encoded Prefix PRID			

Figure 4.3: Data Format for Prefix PRID

4.1.3.1.3 Encoded Provisioning Instance Data (EPD)

The EPD carries the encoded value of a PRI, which includes all the individual values of the attributes that comprise the PRC from which the PRI is instantiated. Its format and field values are shown in Figure 4.4 [29]:

0	1	2	3
Length (variable)		S-Num (EPD=3)	S-Type (BER=1)
Encoded PRI Value			

Figure 4.4: Encoded PRI Object

4.1.3.1.4 Global Provisioning Error Object (GPERR)

The GPERR carries the general errors that do not map to a specific PRC. Its format and field values are shown in Figure 4.5 [29]:

0	1	2	3
Length (variable)		S-Num (GPERR=4)	S-Type (BER=1)
Error-code		Error Sub-code	

Figure 4.5: Global Provisioning Error Object

4.1.3.1.5 PRC Class Provisioning Error Object (CPERR)

The CPERR object carries the errors relating to specific PRCs and must have an associated PRID object. Its format and field values are shown in Figure 4.6 [29]:

0	1	2	3
Length (variable)		S-Num (CPERR=5)	S-Type (BER=1)
Error-code		Error Sub-code	

Figure 4.6: PRC Class Provisioning Error Object

4.1.3.1.6 Error PRID Object (ErrorPRID)

The ErrorPRID object carries the PRID of a PRI that caused an installation error or could not be removed or installed. Its format and field values are shown in Figure 4.7 [29]:

0	1	2	3
Length (variable)		S-Num (CPERR=6)	S-Type (BER=1)
Encoded PRID			

Figure 4.7: Error PRID Object

4.1.3.2 COPS-PR Message Format

Combined field definitions of COPS in [28] and object definitions in COPS-PR [29], the Request message (REQ), Decision message (DEC) and Report state message (RPT) in COPS-PR can be represented in ASN.1 semantics as following respectively:

- <Request Message> ::= <Common Header>
  <Client Handle>
  <Context = config request>
  *(<Named ClientSI: Request>)
  [<Integrity>]

  <Named ClientSI: Request> ::=<*(<PRID> <EPD>)

- <Decision Message> ::= <Common Header>
  <Client Handle>
  *(<Decision>) | <Error>
  [<Integrity]

  <Decision> ::= <Context>
  <Decision: Flags>
  [<Named Decision Data: Provisioning>]

```
<Named Decision Data: Provisioning> ::=<<Install Decision> |
                                         <Remove Decision>>

<Install Decision> ::= *(<PRID> <EPD>)

<Remove Decision> ::= *(<PRID> | <PPRID>)
```

- <Report State> ::= <Common Header>

```
                   <Client Handle>
                   <Report Type>
                   *(<Named ClientSI: Report>)
                   [<Integrity>]

<Named ClientSI: Report> ::=<Named ClientSI: Report: Success | Failure>
                          | <Named ClientSI: Report: Accounting>

<Named ClientSI: Report: Success | Failure> ::= <[<GPERR>] *(<report>)>

<report> ::=<ErrorPRID> <CPERR> *(<PRID><EPD>)

<Named ClientSI: Report: Accounting> ::=<*(<PRID><EPD>)>
```

4.2 Differentiated Services Quality of Service Policy Information Base (DiffServ QoS PIB)

COPS-PR is now the standard protocol for provisioning policies, which supports multiple client types of provisioning policies for a specific domain such as QoS or security domain. Since this thesis focuses on the policy provisioning of DiffServ QoS, the PRCs used in policy communication by COPS-PR will be introduced. These PRCs are defined in [42]. In Chapter 2 how to use QoS elements to build up a Traffic Conditioner Block (TCB) has been described already. The DiffServ QoS PIB models the individual QoS elements that make up the TCBs.

The Policy QoS Informational Model describes how policies are organized in a policy repository [43]. The Structure of Policy Provisioning Information (SPPI) [44] defines the data type and structures used to write Policy Information Base (PIB) modules, including the DiffServ QoS PIB module in this chapter. SPPI and PIB are adapted subset of SNMP's SMI [45] and MIB [46] modules respectively and their relationship is similar to SMI and MIB. There is also a modified version of MIB for SNMP [47], while PIB is tailored for COPS. In [44], *PRC* is used for table and row definitions, *PRI* is used for an instantiation of a row definition, and *attribute* is used for a column object of a table definition.

A PIB is a virtual database of policy information. According to the policies in PIB, an interface may perform some or all of the following functions:

- Packets classification according to the rules set in filters.
- Traffic metering to determine the in-profile packets and out-of profile packets.
- Perform dropping or marking with DSCP actions to packets.
- Enqueue the packets in appropriate queues and forward the packets in queue according to scheduling algorithms.

The DiffServ PIB consists of three types of PRCs: classes that represent QoS function elements in the interfaces such as classifiers, meters, actions and so on; classes that specify parameters applying to a certain type of functional elements such as a token Bucket Meter; and classes that describe the capabilities and limitations of the devices. According to the PIB access rights, the PRCs are classified into two categories: Notify and Install. The first and second types of PRCs belong to the Install category and normally are carried from PDP to PEP. The third type of PRCs belongs to Notify category and normally is carried from PEP to PDP. All these PRCs are defined as separate tables in the DiffServ PIB and the table classification according to PRC category and type is shown as following:

- *Notify*

- *Interface Capabilities Group*
  - *qosIfClassificationCapsTable*
  - *qosIfMeteringCapsTable*
  - *qosIfSchedulingCapsTable*
  - *qosIfElmDepthCapsTable*
  - *qosIfElmLinkCapsTable*

- *Install*
  - *QoS Policy Classes Group*
    - *Functional Element Table*
      - *qosDataPathTable*
      - *qosClfrTable*
      - *qosClfrElementTable*
      - *qosMeterTable*
      - *qosActionTable*
      - *qosAlgDropTable*
      - *qosQTable*
      - *qosSchedulerTable*
    - *Parameter Table*
      - *FrwkIpFilterTable* [48]
      - *qosTBMeterTable*
      - *qosDscpMarkActTable*
      - *qosRandomDropTable*
      - *qosSchdParamTable*

The function of entries in the tables is addressed respectively.

- *qosIfClassificationCapsTable*: specifies the classification capabilities of an interface type.
- *qosIfMeteringCapsTable*: specifies the metering capabilities of an interface type.
- *qosIfSchedulingCapsTable*: specifies the scheduling capabilities of an interface type.

- *qosIfElmDepthCapsTable*: specifies the number of elements of the same type that can be cascaded together in a data path.
- *qosElmLinkCapsTable*: specifies what types of data path functional elements may be used as the next downstream element for a specific type of functional element.
- *qosDataPathTable*: specifies the data path in this device by the interface, role combination and traffic direction.
- *qosClfrTable*: specifies a classifier including multiple classifier elements of same or different types to be used together.
- *qosClfrElementTable*: specifies a classification pattern defined in a filter table entries and subsequent downstream DiffServ data path element when the classification element is satisfied.
- *frwkIpFilterTable*: specifies filters used by classifier elements to perform classification.
- *qosMeterTable*: specifies specific meters that a system may use to police a stream of traffic.
- *qosTBMeterTable*: specifies a specific token-bucket meter used by the entries in *qosMeterTable* to perform metering.
- *qosActionTable*: specifies actions that can be performed to a stream of traffic.
- *qosDscpMarkActTable*: specifies specific parameters to the DSCP mark action used by entries in *qosActionTable* to perform marking.
- *qosAlgDropTable*: specifies an element that drops packets according to some algorithm.
- *qosRandomDropTable*: specifies a process that drops packets randomly used by entries in *qosAlgDropTable* to perform dropping.
- *qosQtable*: specifies a queue, which scheduler will service the queue and how the scheduler will service the queue.
- *qosSchedulerTable*: specifies a packet scheduler to service the queues.
- *qosSchdParamTable*: specifies scheduling parameters used by entries in *qosQTable* and *qosSchdulerTable* to perform scheduling.

4.3 Implementation of DiffServ QoS PIB

The topology of the implementation of DiffServ PIB is the same as shown in Figure 3.5. The main differences from the policy-based admission control lie in that the PEP is a Cisco 4700 Router (192.168.1.1 in Figure 3.6) as the edge router in the topology and the policy provisioning is initiated from a policy console.

4.3.1 Design of the Implementation of COPS-PR

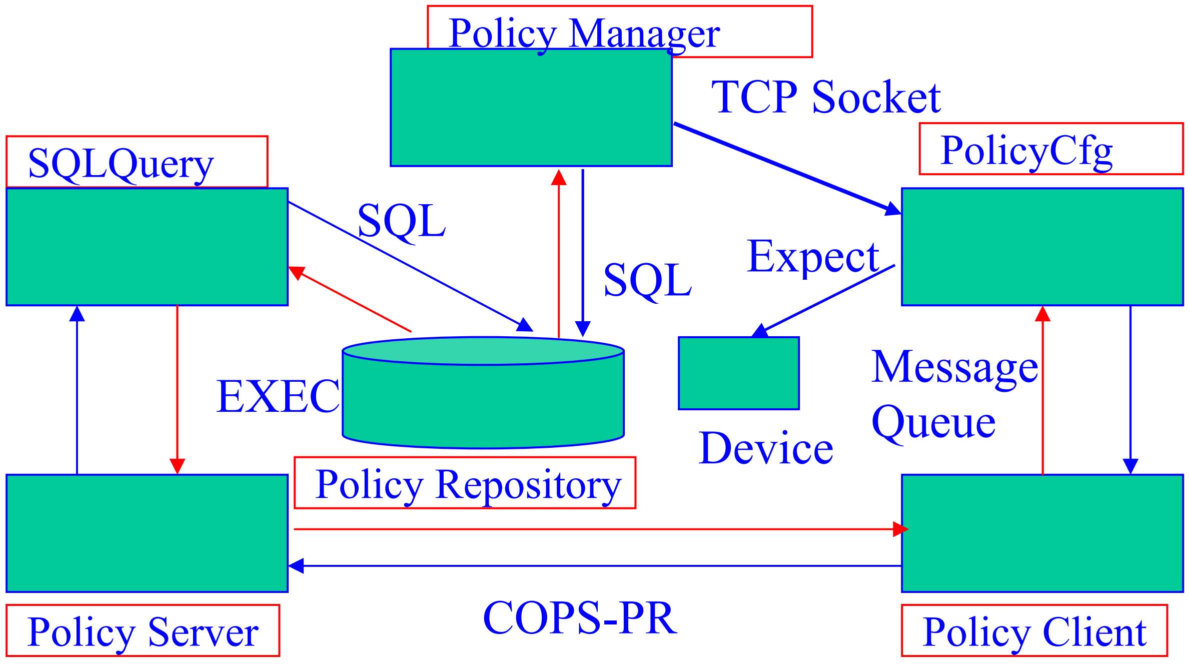


Figure 4.8: Policy Provisioning Implementation Diagram

Figure 4.8 illustrates the program modules of the COPS-PR implementation. The design includes five modules: *Policy Manager, PolicyCfg*, *Policy Client*, *Policy Server* and *SQLQuery*. Among them, *Policy Manager* plays the role of Policy Console and management tools, *PolicyCfg* and *Policy Client* play the role of PEP, and *Policy Server* and *SQLQuery* play the role of PDP. Their functions are summarized as follows:

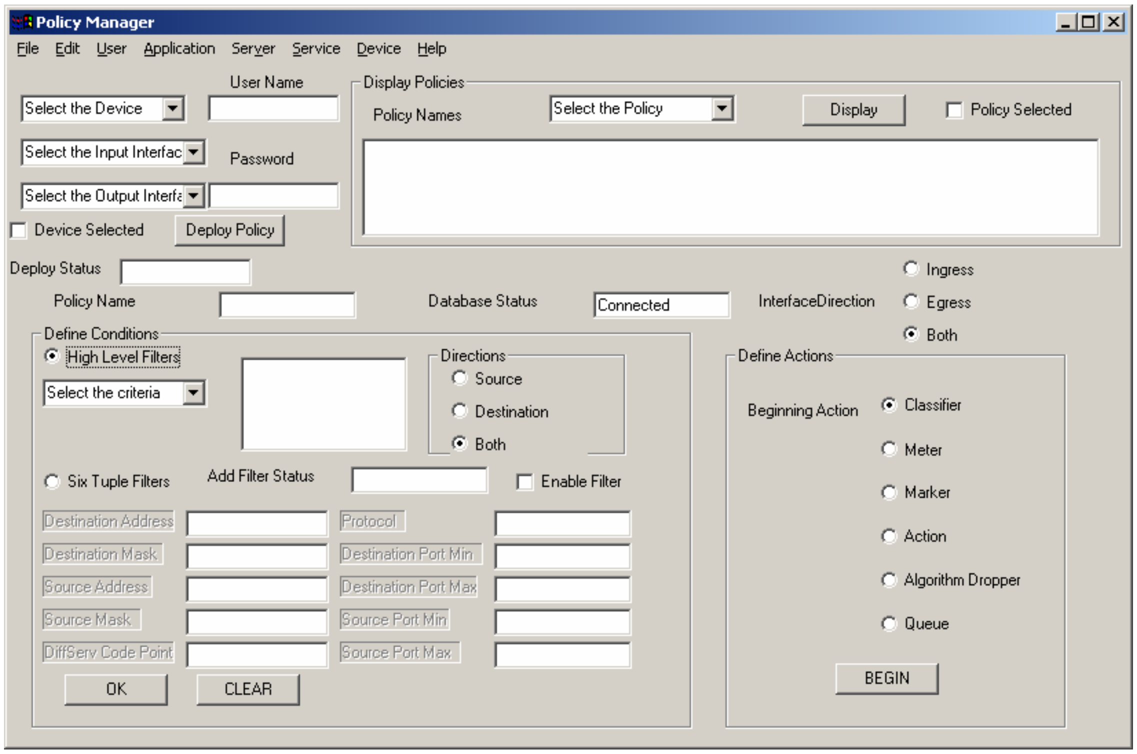
- *Policy Manager*: provides a GUI console for policy editing and reviewing, and management tools for semantics and syntax checking, storage and retrieval of policies in MySQL database.
- *PolicyCfg*: Acts as an agent of Cisco 4700 router to perform the configuration tasks, accepts the configuration commands and parameters through a socket, and executes Expect files to configure the router.
- *Policy Client*: Acts as a COPS-PR client to transfer the request to policy server, gets the reply from the policy server and also keeps the client and server connected through *COPSKAMsg* message.
- *Policy Server*: accepts the request from the policy clients (*Policy Client*), forks a process *SQLQuery* to query the policy database, gets the configuration file from *SQLQuery* and transfers the configuration file back to its clients.
- *SQLQuery*: queries the policy database, generates the configuration file according to the retrieved PIB, and returns the configuration file to policy server.

In reality, there should be a link between the *Policy Manager* and *Policy Server*. In this implementation, the existing TCP socket between *Policy Client* and Policy *Server* is used to transfer configuration commands and parameters. This is just a programming convenience.

The modules *PolicyCfg*, *Policy Client*, *Policy Server* and *SQLQuery* are coded using C/C++ on Linux 6.2 while the *Policy Repository* is a MySQL server running on Windows 2000 platform. The main coding files in this implementation are *PolicyCfg.c*, *PolicyClient.cxx*, *SQLQuery.c*, and *COPSProcessor.cxx,* which is the main component of *Policy Server*. All other files are borrowed directly from COPS1.2.0 with minor changes. *Policy Manager* is coded using Microsoft J++. The detailed implementation of the modules is described in the following sections.

4.3.2 Implementation of Policy Manager Module

The functions of the *Policy Manager* module are to provide a GUI console for policy editing and reviewing, and management tools for semantics and syntax checking, storage and retrieval of policies in MySQL database. The policy console is shown in Figure 4.9.



Database Status: Connected

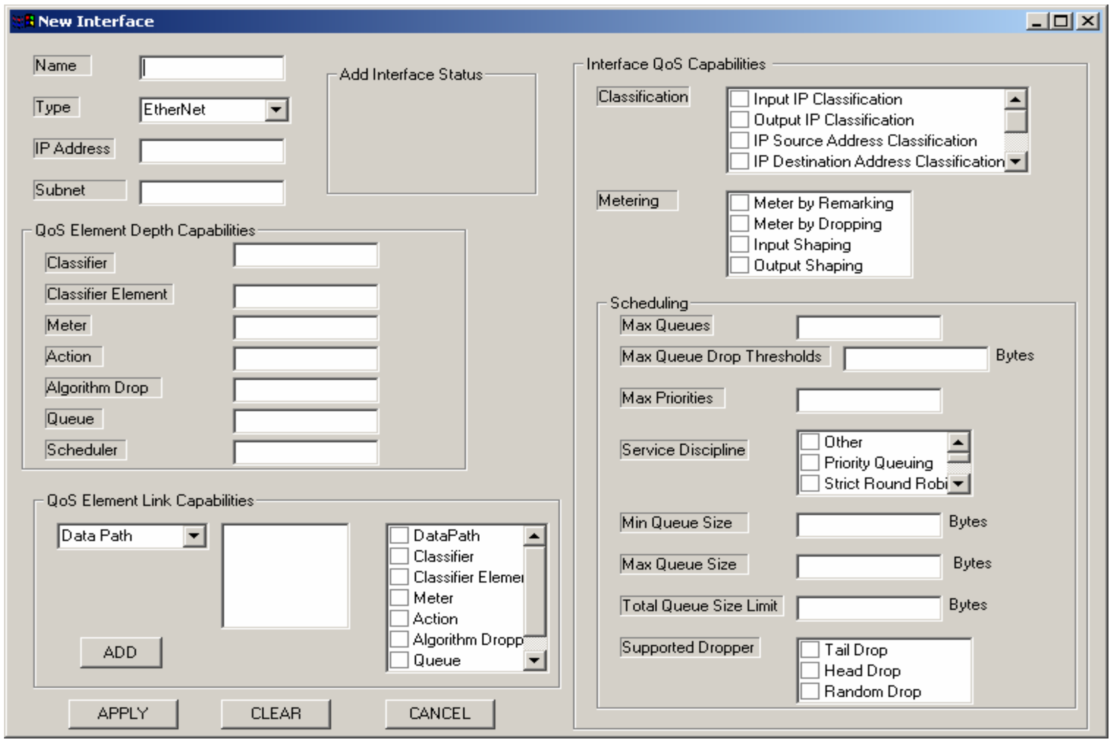
InterfaceDirection: Ingress [✓ Both] Egress

Define Conditions: [✓ High Level Filters] Six Tuple Filters

Directions: Source Destination [✓ Both]

Beginning Action: [✓ Classifier] Meter Marker Action Algorithm Dropper Queue

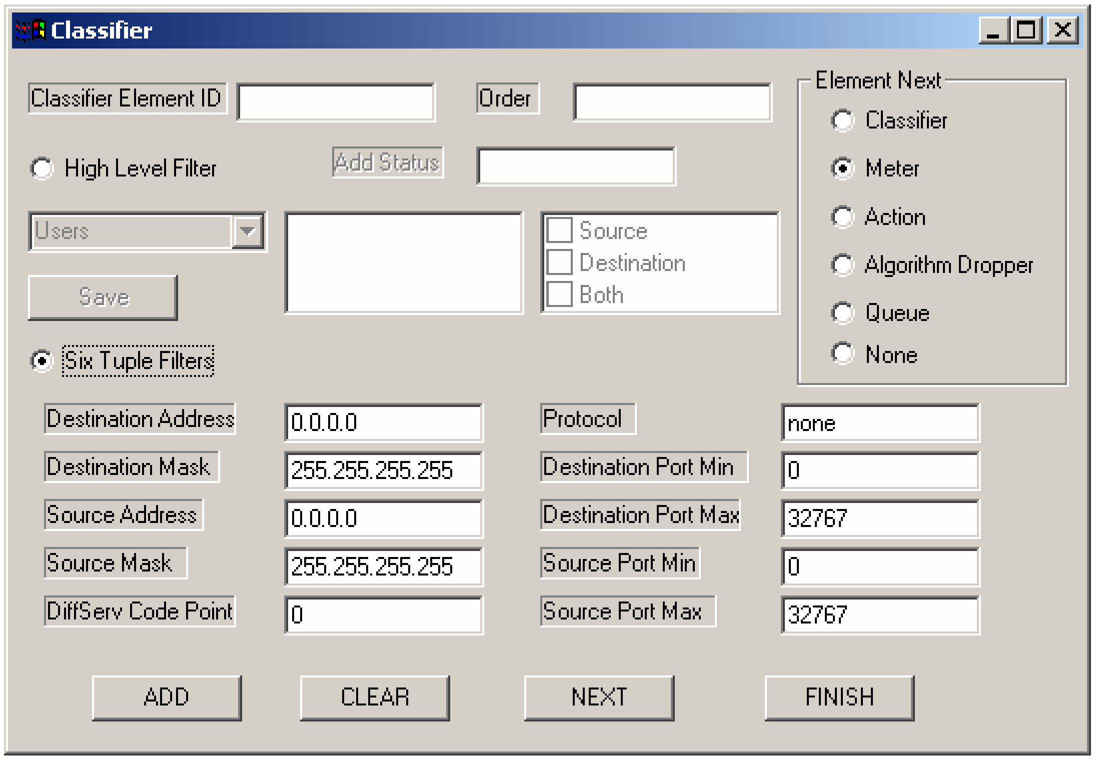
Figure 4.9: Policy Manager GUI



Type: EtherNet

QoS Element Link Capabilities: Data Path

Figure 4.10: Form for Defining Interface Capabilities


Classifier

Classifier Element ID		Order	
High Level Filter		Add Status	
Users		Source / Destination / Both	
[✓ Six Tuple Filters]			
Destination Address	0.0.0.0	Protocol	none
Destination Mask	255.255.255.255	Destination Port Min	0
Source Address	0.0.0.0	Destination Port Max	32767
Source Mask	255.255.255.255	Source Port Min	0
DiffServ Code Point	0	Source Port Max	32767

Element Next: Classifier [✓ Meter] Action Algorithm Dropper Queue None

Figure 4.11: Form for Defining Classifier

The main utilization of the policy console is to define the DiffServ PIB, i.e. the tables defined in section 4.2. The forms for defining the tables of capabilities and limitations of the interface can be open under the submenu *Interface* under the main menu *Device*. The forms for defining the tables of functional elements and parameters can be open at subsequent forms after the *Begin* button has been clicked. Since the PIB starts from the *dataPathTable* and follows the *PridNext* element in the table to open the corresponding tables, it is flexible for describing various policies enforced on the devices if only the devices' capacities and limitations support such policies. A form for defining interface capabilities is shown in Figure 4.10 while a form for defining a Classifier Element and its parameters is shown in Figure 4.11.

Another use of the policy console is to define the profiles for users, applications, servers and services. Since from the high-level view, a policy can be viewed to provide certain services to certain users, applications and servers, this use facilitates the definition of high-level policies and mapping the high-level policies to DiffServ PIBs. A form for defining a user profile is shown in Figure 4.12.

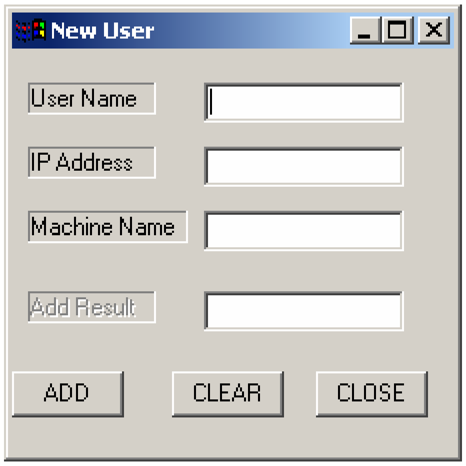


Figure 4.12: Form for Defining User

The management tools for policy validation, conflict solving, semantics and syntax checking are coded together with the policy console. Currently only simple policy conflicts validation is provided to avoid the new policy conflicting with the enforced policies. For example when a new policy is input and will be put in the policy repository, the system will check whether there is a policy with the same filter and different marker action. If this happens, the system will report an error and alert the network manager to check the policies. If the network manager wants to enforce the new policies, he must delete the old policy first. After the policies are input from the form and validated, they are stored in the corresponding PIB tables in the MySQL databases.

4.3.3 Implementation of PolicyCfg Module

The main function of *PolicyCfg* is to act as an agent of Cisco 4700 router to perform the configuration tasks. Because currently the DiffServ PIB is on its draft stage and there are no devices that are PIB-aware, the PIB must be translated into a language the device can understand. Currently, the router configuration methods mainly including: Command Line Interface, Telnet and loading the configuration files when the router is booting up. Since policy-based networking is introduced to take off the burden of manual configuration, the configuration process must be automated.

Since *Expect* [49] can automate interactive programs by intercepting a program's terminal output and sending the program appropriate responses, *Expect* is used to do the auto-telnet to automate configuration tasks. The *Expect* program includes three main building blocks: the *send* command, the *expect* command and the *spawn* command. The *send* command sends text strings to a process, the *expect* command waits for a specified text string in the process's output stream, and the *spawn* command starts a process.

The *PolicyCfg* works as follows:

- Open a server socket, waiting for a connection from *Policy Manager*.
- Upon receiving the messages from *Policy Manager*, the *PolicyCfg* parses the data, encapsulates the data in a message and passes the message to *Policy Client* through a message queue, and waiting for the returning data from *Policy Client*.
- Upon receiving the data from the *Policy Client*, the *PolicyCfg* runs the *Expect* program to execute the *Expect script* to do the configuration of the router.
- After the configuration is finished, the *PolicyCfg* returns the configuration results (success or failure) to the *Policy Manager*.

4.3.4 Implementation of Policy Client Module

Policy Client accepts the data in the message queue and encapsulates the data into a COPS-PR request message to *Policy Server*. Different from the COPS-RSVP client type, COPS-PR uses the classes *COPSPrefixPrID*, *COPSPrID*, *COPSPrErrorPRID*, *COPSPrError* and *COPSPrEPD*, which are subclasses of *COPSPrObjBase*, to encapsulate PIB data. Class *COPSPrObjParser* is used for COPS-PR object parsing.

Based on the COPS stack the Vovida provided, *Policy Client* and *Policy Server* are coded using GNU C++ on Linux 6.2. The *Policy Client* works as follows:

- *Policy Client* opens a connection to *Policy Server* by a *COPSClientOpenMsg*.
- *Policy Client* periodically sends the *COPSKAMsg* to *Policy Server*.
- After receiving the *COPSClientAcceptMsg* from *Policy Server*, *Policy Client* creates two message queues using the same keys as in *PolicyCfg*.
- *Policy Client* takes the data from queue, parses the data, encapsulates the data in respective COPS-PR objects, then encapsulates the COPS-PR object in *ClientSI* part of the *COPSRsqMsg*, sets the *Client Type* as 2(COPS-PR) and sends the message to *Policy Server*.

- *Policy Client* waits for the decisions from the *Policy Server*, parses the COPS-PR objects in the *Named Decision Data*, takes out the configuration data from the *COPSPrEPD* object and transfers the configuration data to *PolicyCfg*.
- *Policy Client* sends a *COPSReportMsg* back to *Policy Server*.

4.3.5 Implementation of Policy Server Module

The functions of the *Policy Server* are to accept the request from policy clients (including *Policy Client*), fork a process *SQLQuery* to query the policy database, get the decision from *SQLQuery* and transfer the decision back to its clients. It is the same policy server as that of COPS-RSVP, since a server should service different clients. It works following the same steps as in COPS-RSVP, except now the child process is *SQLQuery* and the client is provisioning client *Policy Client*.

4.3.6 Implementation of SQLQuery Module

The functions of *SQLQuery* are to query the policy database, get PIB data and configuration data, and return the data to *Policy Server*, which is coded using MySQL C API on Linux 6.2 platform.

SQLQuery works as follows:

- *SQLQuery* parses the data received from *Policy Server*.
- *SQLQuery* uses the policy key from the parsed data to look for the policy it should use.
- *SQLQuery* translates the DiffServ QoS PIB into an *Expect script*.
- *SQLQuery* returns the PIB and *Expect script* data to its parent process *Policy Server.*

4.3.7 An Example of Policy Provisioning

An example of the policy of Policy Provisioning which has been enforced on edge router-Cisco 4700 of the test-bed is shown as follows:

High-level policy for the ingress interface:

If the video traffic from QuickTime Streaming Server (192.168.1.2:554) has an average rate not exceeding 2000 kbits and a burst size not exceeding 2000 bytes, the traffic will receive EF service and the traffic exceeding the profile will be dropped.

High-level policy for the egress interface:

If the EF traffic does not exceed 4000 kbits, it will be guaranteed forwarded. If the EF traffic exceeds the profile, it will be randomly dropped.

After inputting the policies through the policy console, the two policies are stored in the MySQL database in the form of DiffServ PIB. After selecting the interfaces to enforce the policies and click the *Deploy Policy* button, the entire system will run according to the procedures described before. The PIB data in this example is shown as follows:

```
#PrId: policy_24
#PrEPD:
[policy_24][general][in][clfr_24]:
[clfr_24][clfr_24]:
[clfrem124][clfr_24][2][meter_24][filter_24]:
[filter_24][Internet][0.0.0.0][255.255.255.255][Internet][192.168.1.2][0.0.0.0][0][udp]
[0][32727][3288][3288][Enabled]:
[meter_24][action_24s][0.0][tbmeter_24]:
[tbmeter_24][tokenBucket][2000][2000][2000]:
[action_24s][algdrop_24][marker_24][2]:
[marker_24][46]:
[algdrop_24][wred][queue_24][24][0][rddrop_24]:
[queue_24][schdlr_24][schdpm_24]:
[schdpm_24][20][4000][0][8000][0]:
[schdlr_24][][4][0.0]:
```

[rddrop_24][3000][0][4000][0][10][10][10]:

The *Expect script* for ingress interface configuration:

```
#!/usr/bin/expect --
#PrId: policy_24
#PrEPD:
if {[llength $argv]!=0} {
        puts "usage: configfile"
        exit 1
}
set timeout 20
spawn telnet 192.168.1.1
expect "Username:"
send "zhangyi\r"
expect "Password:"
send "zhang\r"
expect "ccn-4700#"
send "configure terminal\r"
expect "ccn-4700(config)#"
send "access-list 124 permit ip udp any 192.168.1.2 0.0.0.0 eq 554\r"
expect "ccn-4700(config)#"
send "class-map clfr_24\r"
expect "ccn-4700(config-cmap)#"
send "match access-group 124\r"
expect "ccn-4700(config-cmap)#"
send "exit\r"
expect "ccn-4700(config)#"
send "policy-map meter_24\r"
expect "ccn-4700(config-pmap)#"
send "class clfr_24\r"
```

```
expect "ccn-4700(config-pmap-c)#"
send "police 2000000 2000 2000 conform-action set-dscp-transmit 46 exceed-action drop
\r"
expect "ccn-4700(config-pmap-c)#"
send "exit\r"
expect "ccn-4700(config-pmap)#"
send "exit\r"
expect "ccn-4700(config)#"
send "interface ethernet2\r"
expect "ccn-4700(config-if)#"
send "service-policy input meter_24\r"
expect "ccn-4700(config-if)#"
send "exit\r"
expect "ccn-4700(config)#"
send "exit\r"
expect "ccn-4700#"
send "exit\r"
expect "Connection closed by foreign host."
puts "Configure input interface successfully"
```

The Expect script for egress interface configuration:

```
#!/usr/bin/expect --
#PrId: policy_24
#PrEPD:
if {[llength $argv]!=0} {
        puts "usage: configfile"
        exit 1
}
set timeout 20
spawn telnet 192.168.1.1
```

```
expect "Username:"
send "zhangyi\r"
expect "Password:"
send "zhang\r"
expect "ccn-4700#"
send "configure terminal\r"
expect "ccn-4700(config)#"
send "class-map action_24s\r"
expect "ccn-4700(config-cmap)#"
send "match ip dscp 46\r"
send "exit\r"
expect "ccn-4700(config)#"
send "policy-map schdlr_24\r"
expect "ccn-4700(config-pmap)#"
send "class action_24s\r"
expect "ccn-4700(config-pmap-c)#"
send "bandwidth 4000\r"
expect "ccn-4700(config-pmap-c)#"
send "random-detect\r"
expect "ccn-4700(config-pmap-c)#"
send "exit\r"
expect "ccn-4700(config-pmap)#"
send "exit\r"
expect "ccn-4700(config)#"
send "interface ethernet1\r"
expect "ccn-4700(config-if)#"
send "service-policy output schdlr_24\r"
expect "ccn-4700(config-if)#"
send "exit\r"
expect "ccn-4700(config)#"
send "exit\r"
```

```
expect "ccn-4700#"
send "exit\r"
expect "Connection closed by foreign host."
puts "Configure output interface successfully"
```

After the enforcement of the policies, the configuration of Cisco 4700:

```
ccn-4700#show access-list
Extended IP access list 124
    permit udp host 192.168.1.2 any eq 554
ccn-4700#show policy-map interface ethernet2
 Ethernet2

  Service-policy input: meter_24

    Class-map: clfr_24 (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: access-group 124
      police:
        200000 bps, 2000 limit, 2000 extended limit
        conformed 0 packets, 0 bytes; action: set-dscp-transmit 46
        exceeded 0 packets, 0 bytes; action: drop
        conformed 0 bps, exceed 0 bps violate 0 bps

    Class-map: class-default (match-any)
      1616 packets, 102227 bytes
      5 minute offered rate 1000 bps, drop rate 0 bps
      Match: any
ccn-4700#show policy-map interface ethernet1
 Ethernet1
```

```
Service-policy output: schdlr_24

  Class-map: action_24s (match-all)
    0 packets, 0 bytes
    5 minute offered rate 0 bps, drop rate 0 bps
    Match: ip dscp 46
    Weighted Fair Queueing
      Output Queue: Conversation 265
      Bandwidth 4000 (kbps)
      (pkts matched/bytes matched) 0/0
      (depth/total drops/no-buffer drops) 0/0/0
       exponential weight: 9
       mean queue depth: 0

  Class     Random drop      Tail drop       Minimum   Maximum     Mark
  (Prec)    pkts/bytes       pkts/bytes    threshold threshold probability
     0      0/0             0/0          20       40       1/10
     1      0/0             0/0          22       40       1/10
     2      0/0             0/0          24       40       1/10
     3      0/0             0/0          26       40       1/10
     4      0/0             0/0          28       40       1/10
     5      0/0             0/0          30       40       1/10
     6      0/0             0/0          32       40       1/10
     7      0/0             0/0          34       40       1/10
   rsvp       0/0             0/0          36       40       1/10

  Class-map: class-default (match-any)
    740 packets, 81059 bytes
    5 minute offered rate 0 bps, drop rate 0 bps
    Match: any
```

4.4 Conclusions

In this chapter, as the policy exchanging protocol of the implementation, COPS Usage for Policy Provisioning (COPS-PR) is described. The virtual database for DiffServ QoS policy, DiffServ QoS PIB used as the foundation for implementing the policy repository is discussed in detail. Finally the implementations of the modules of the PBN system are described and an example is used to describe the usage of the PBN system. The total time from initiating the configuration command to finishing the configuration is about 20 seconds, of which most is used in the configuration of the devices by *Expect.* This system has greatly leveraged the capabilities and eased the tasks of the configuration process.

Different from Policy-based Admission Control, COPS-PR “pushes” the policies from a PDP to the client devices. Policies in COPS-PR are relatively static and they are coarse grained in the core devices since the treatment of the packets is only based on DSCP field.

Chapter 5

Providing End-to-End QoS by PBN for DiffServ Networks

In the two previous chapters, the implementation of policy based networking focuses on the intra-domain QoS. But since network resources must be provided end-to-end for a flow to guarantee QoS, inter-domain resource management plays an important role for negotiating and coordinating the resources between domains, i.e. a policy server in one domain interacts with its peers in other domains through an inter-domain protocol to establish SLAs. In this Chapter, a novel model called MLTTM (MultiLateral Two-Tier Model) for policy-based resource management to provide end-to-end QoS in DiffServ networks is proposed.

5.1 Motivations for a New Model to Provide End-to-End QoS

The new emerging applications such as IP-telephony, video on demand, video conferencing and so on have characteristics different from the traditional best-effort services in that they have some requirements on the network QoS such as bandwidth, delay, delay variations etc. To meet the end-to-end QoS of these applications, these requirements must be satisfied throughout the paths of the flows since the delayed packets may be deemed useless. The IETF have published two architecture standards to address the QoS: the IntServ/RSVP [6] and DiffServ [11]. The DiffServ architecture has successfully addressed the scalability problem based on the coarse grained classification of the flows. If the Internet is assumed to be comprised of many inter-connected autonomous systems (ISPs), which provide transit services for others (ISPs) and these AS are all DiffServ compliant, then according to the definition in [11], Internet can be viewed as inter-connected DS domains. Under this assumption, a flow in Internet traverses all the DS domains along the path from its source DS domain to its destination DS domain. To provide QoS, the ingress nodes of the DS domains perform the classification and traffic conditioning to sort packets into their corresponding classes and control the amount of traffic of each class and the core nodes only perform corresponding Per Hop Behaviors [14] [15] based on the DiffServ Code Point [24] to forward the traffic.

To guarantee the end-to-end QoS, the resources must be reserved end-to-end for each class in some form. In [23] a two-tier resource management model is proposed to address this problem and the end-to-end QoS is achieved by the concatenation of intra-domain resource allocation and bilateral resource agreements between neighboring domains. A Bandwidth Broker in each DS domain is employed to perform both local resource allocation and the reservations of resources with the neighboring DS domains. Since one domain does not know about whether the flows out of it have enough resources in the non-neighboring domains, how to make the resource commitment for each class between neighboring domains consistent along the path is a challenging issue. In [23], the author uses the current load of a class from egress routers to estimate the amount of inter-domain resources needed for the future.

The two-tier model performs the admission control by predicting future available bandwidth based on current load. But predicting the future in terms of now is not feasible unless there are many assumptions or restrictions on characteristics of the traffic.

The typical procedure to negotiate SLAs along the route end-to-end in the two-tier model is described using an example. In Figure 5.1, DS1 wants to transmit some traffic to DS6 with the constraints that end-to-end QoS must be met. According to the BGP table in edge routers the route is pre-determined as DS1-DS2-DS4-DS6. Then the BB in DS1 will first check the SLA with DS2. If the SLA cannot satisfy the request, then the SLA will be renegotiated. If the BB in DS2 accepts the request, it will in turn check its SLA with DS4. In this way until all the BBs along the path assure there are enough resources, the request is accepted, and otherwise it is rejected.

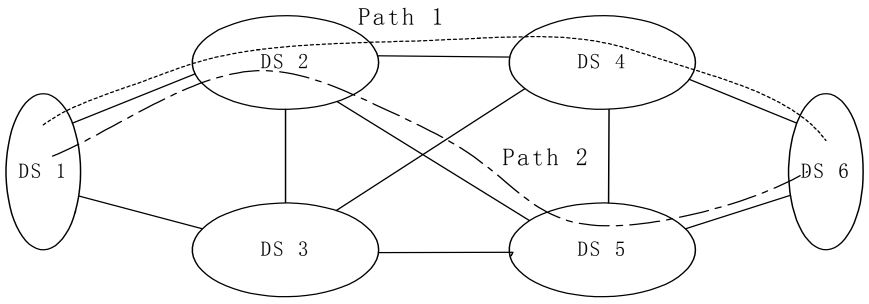


Figure 5.1: Resource Allocation in a Two-Tier Model

There are two assumptions in this model. The first assumption is that the BGP table is static. For instance, an aggregate from DS1 can only choose the route DS1-DS2-DS4-DS6. Once the request is rejected, it cannot choose another route such as DS1-DS2-DS5-DS6 even along this route there are enough resources. It can only wait for some time, then retry the request along the same route.

The second assumption is that a BB in one DS domain has no way to know the resource availabilities of non-neighboring domains. For instance DS1 can only talk to DS2 and DS3, but it has no way to know if there are enough resources in DS4, DS5 and DS6.

These two assumptions are the current situation of Internet. But since a model is designed for next generation Internet, the design of the model should not be restricted by the current situation. Before discussing the proposed model, let's consider a daily example first.

Suppose a traveler wants to fly from Beijing to Chicago and there is no direct flight but there are two alternate ways. One way is to take a plane of China Airlines to Seoul, then transfer to a plane of Korea Airlines to Chicago. Another way is to take a plane of Japan Airlines to Tokyo, then transfer to a plane of American Airlines to Chicago. After she makes a call to a travel agent in Beijing, the travel agent will tell her if there are tickets for these two routes and how much is the price for the tickets. Then the traveler will make the decision which route she would to choose based on the routes and prices, and just book the ticket from the agent. The travel agent need not to call a peer in Seoul or Tokyo to check whether there are tickets left in Korea Airlines or American Airlines and their prices. All the necessary information can be checked just in the agent's office because the updated information from various airline companies can be checked in his local databases.

Compared to the two assumptions in the two-tier model in [23], the example above has following assumptions:

- There are alternate routes from a source to a destination.
- There is a common data store distributed across the world to check the resource (tickets) availability and their prices.
- The sender (traveler) decides which route to choose to the destination in terms of the resource availability and prices.
- The contract is in nature multilateral and consistent, although from appearance it is still a bilateral contract (between traveler and agent, agent in Beijing with agent in Seoul etc.).

Putting above assumptions in the design, in this thesis, a model based on the framework in [23] is proposed, which is called MultiLateral Two-Tier Model (MLTTM) to distinguish from the well-known two-tier model. MLTTM is featured by: distributed global information stores, dynamic BGP tables, source-based QoS routing, threshold-based pricing, competition among alternate peers and explicit signaling for resource reservations.

In an integrated services network, end-to-end QoS of a flow is provided by explicitly signaling the QoS requirements, reserving the resources and keeping the state of the flow along the path [6] [9]. In MLTTM, the concepts of explicit signaling of requirements is borrowed from [6] [9] but these concepts are used in a scalable manner in that per-flow reservation states is not maintained in routers but aggregate reservation states in BBs. The global sharing of the resource availability and pricing information is used by the BBs to do the resource request decision. In MLTTM, the Internet consists of many DS domains, which are administratively independent of other domains. Every domain has a Bandwidth Broker to manage the resources of its own domain. Since a BB manages a transit network, its main tasks are to negotiate the SLAs with neighboring domains and manage the local domain resources to satisfy the SLAs. The BB can also exchange global information of non-neighboring domains through the neighboring BBs. In a word, MLTTM is no different from the two-tier resource model in [23] in appearance. But a different approach is taken to do the resource management. In [23], the authors use a predictive method to do the admission control on demand based on measurement of current load [50] of the links while in this thesis, advanced explicit signaling is used to do the reservation of the resources. What has been done is to change the functionalities and mechanisms of the BB to make the dynamic SLA negotiation and establishment globally consistent and by adding a pricing mechanism reflecting the network resource availability status and performing QoS routing to choose the best route. In this way, although the contracts are still between neighboring domains, because of the consistency among the contracts it can be viewed every domain makes a contract with all other domains. Note in the following sections the term SLS (Service Level Specification) instead of SLA is used, indicating only the technical issues of the SLA is addressed.

MLTTM tries to achieve these goals: guaranteed QoS, fairness, load balancing and efficiency.

Guaranteed QoS is achieved by reservation of the resources along the route. Because in MLTTM, flows with one class of reserved resources are marked different from other classes, the proposed model prevents the resource starvation from malicious flows. Also since the resources cannot be used until they are reserved, the QoS of the previous allocated flows is not demoted because of establishment of the new SLSs.

Fairness is achieved by QoS routing and pricing scheme (here fairness only has its meaning in economics, i.e. free competition in the market). In MLTTM, a seller can adjust resource prices according to resource availability while a buyer can choose the sellers according to its QoS and budget. This also means sellers can not monopoly the prices since buyers can choose alternatives among them.

By QoS routing and pricing scheme, there should be no links where the reserved traffic faces congestion problems since in this scenario the prices of the resources in this link will be prohibitively expensive. So the load balancing can be achieved. Since the load is balanced across the network, and in most cases, there will be less waste of the resources, network efficiency is achieved.

The rest of this chapter is organized as following: Section 2 concentrates on intra-domain QoS management and QoS based routing; Section 3 explains how a BB exchanges information with its neighboring domain BBs, what the information should include, how bilateral agreements are negotiated and established; Section 4 deals with the traffic monitoring, accounting and charging issues; Section 5 presents simulation results and concludes this chapter with some summary discussions.

5.2 Intra-domain Resource Management and QoS Routing

Figure 5.2 is used to elaborate some terminologies used in MLTTM model. All the transit networks are assumed to be DiffServ-compliant, i.e. a DS Domain. The user networks called stub networks, which may or may not be DiffServ capable have DiffServ capable Access Routers connected to the transit network DS. The edge routers in DS domain connected with stub networks are also called Access Routers to distinguish from the edge routers connected only with other transit networks, which are called Border Routers. (Note: A router actually has many interfaces. But when discussing an aggregate edge-to-edge pair on the link basis, the interfaces of the routers are actually referred to). There are many core nodes within the transit network. But for simplicity, they are not illustrated and this will not affect discussion. All flow aggregates illustrated in Figure 5.2 should be considered as virtual links from a macroview.

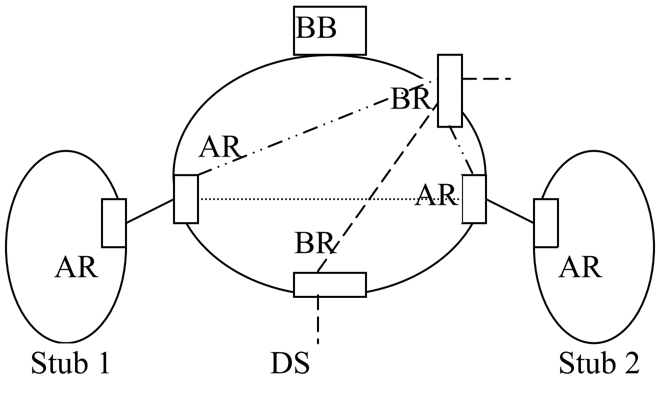


Figure 5.2: Traffic Type Classification in a DS domain

The traffic within the DS is classified into four types: The traffic from one Access Router to another Access Router is called Local Traffic since its origin and destination are all local stub networks. The traffic from an Access Router to a Border Router is called Local-Origin Traffic since its origin is a local stub network. The traffic from a Border Router to an Access Router is called Local-Destination Traffic since it terminates at a local stub network. The traffic between two Border Routers is called Transit Traffic since neither its origin nor its destination is a local stub network.

In MLTTM, the resources for Local Traffic are allocated by the Bandwidth Broker in local DS domains depending on the resource availability in DS. The resources needed for Local-Origin Traffic are divided into two parts along the path: the resources of the local DS part is allocated by local DS BB and the remote part on the path is requested by local BB from other BBs in the traversing transit networks along the path. The resources needed for Local-Destination traffic are allocated by local BBs in response to the requests from the remote BBs of the domains whose stub networks originate these aggregates. The resources needed for the Transit Traffic are also allocated by local BB in response to the request from the remote BBs in the domains whose stub networks originate the traffic.

Figure 5.3 shows a logical view of the BB architecture designed in this framework. Here a Bandwidth broker plays three roles: a seller who sells the local available resources for the Transit Traffic and Local Destination Traffic to other BBs, a buyer who buys resources for its Local-Origin Traffic from other BBs and a manager who manage the Local Traffic.

Recently, the concept of Per Domain Behavior is defined by IETF [51]. In [51], the PDB is defined as "*the expected treatment that an identifiable target group of packets will receive from 'edge-to-edge' of a DS domain. A particular PHB (or, if applicable, list of PHBs) and traffic conditioning requirements are associated with each PDB.*" From this definition, a PDB can be viewed as the treatment of the packets received along a virtue link (edge to edge), a concatenation of the PHBs along the route the packets traverse. Since a PHB is the treatment of the packets in one node and the different edge-to-edge virtual links may include different number of intermediate nodes, the same PHB along the different virtual link possibly results different PDBs. Since in a transit network, only the treatment of an aggregate received from a virtual link (an ingress Edge Router to an egress Edge Router) is relevant, in this chapter, the concept of PDB will be used.

When discussing the traffic, it means that traffic belonging to one specific class corresponds to one PDB in each DS domain traversed.

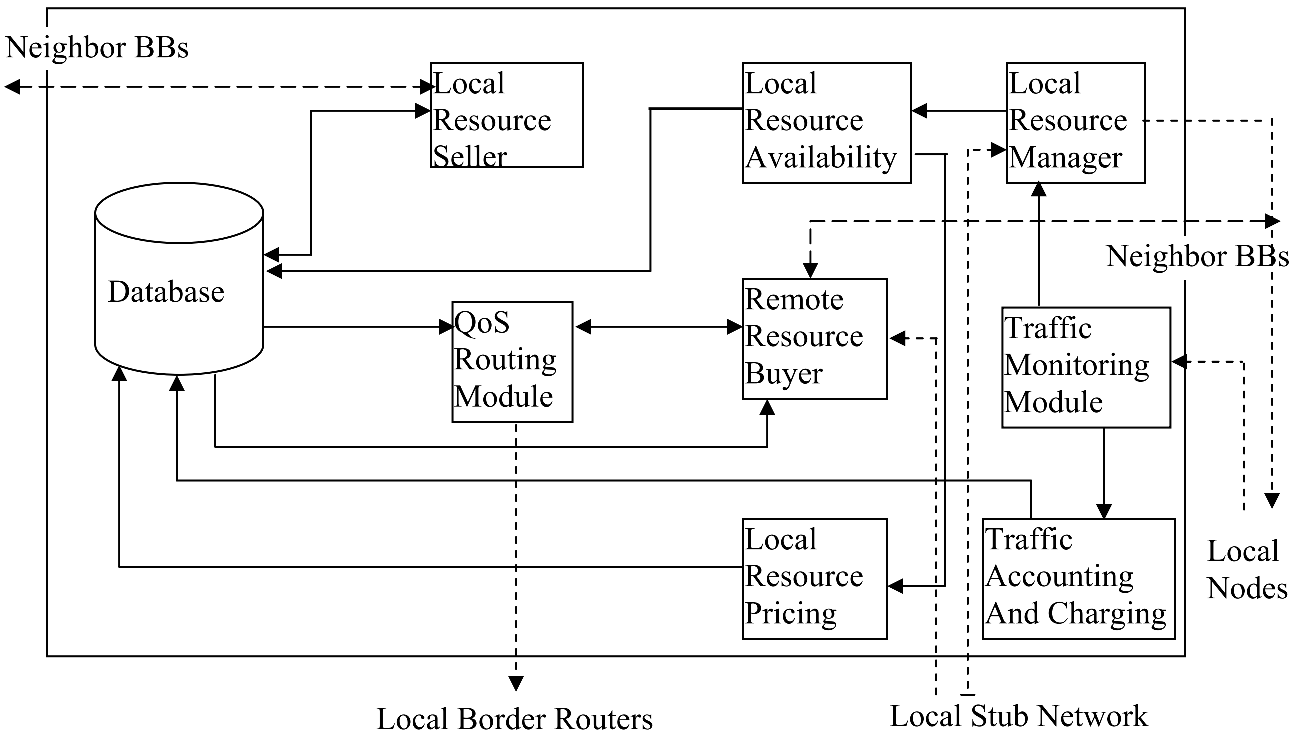


Figure 5.3: A Logical View of Bandwidth Broker Architecture

Currently, the definition of PDB is in its early stage and they are only defined as draft3 [52] [53], so some terminology defined in this thesis is used instead. For simplicity assuming that there exist only three types of PDBs in each DS domain: Golden, Silver and Best-effort, which are built respectively from three types of PHBs: EF, AF and BE. From the point of PDB (i.e. the edge-to-edge pairs), three separate virtual networks over-lays in one physical network: one for Golden, one for Silver, one for Best-effort. From the point of PHB (i.e. the individual links and nodes), three separate virtual networks over-lay in one physical network: one for EF, one for AF, one for BE. For discussions of the resource reservation, the concept of virtual network for PDB is helpful, while for discussion of the computation of local resource availability the concept of virtual network for PHB is beneficial. The components and their functionalities are described as:

Local Resource Manager:

It has been mentioned that there are four types of traffic in a DS domain: the Local Traffic, the Local-Origin Traffic, the Local-Destination Traffic, and the Transit Traffic. All of these four kinds of traffic need consume local resources. Since except for Local Traffic, all other three types of traffic have one or both ends in another DS domain and their resource allocation is related to BBs in other domains that these three types of traffic would have traverse, their resource allocations should be end-to-end and belong to the inter-domain resource management. These issues are left to next section although the Local Resource Manager also allocates the resources needed for these traffics on the portions in local DS domain. The main functions of the Local Resource Manager are to configure Access Routers connecting the local stub network and Border Routers to do classification and traffic conditioning, configure the Core Routers using a PHB or a list of PHB for each PDB, deal with the resource requests from local stub networks and allocate the resources for the Local Traffic. It also gets traffic information from the Traffic-monitoring Module and reports the resource utilization information to the Local Resource Availability module. So a Local Resource Manager knows about the resource utilization of every local link and node. The stub network can be DiffServ capable, IntServ/RSVP capable or neither, which will not affect the basic functionalities of Local Resource Manager. Assuming that the stub networks are IntServ/RSVP capable, when a user in a stub network has traffic to transfer to another user in another stub network, it will issue a RSVP message to signal its requests. The resources the traffic needs are reserved on a per-flow basis, hop-by-hop at each traversing node in the stub network and the egress router of the stub network towards the DS or the ingress router in DS. The traffic is classified and conditioned according to the SLS between the DS and the stub network. The RSVP will tunnel through the DS to the egress router connecting the destination network. When the RSVP enters the destination stub network, it will perform the same reservation as in the origin network. The Local Resource Manager in this scenario mainly performs negotiation of the SLS with its stub network, regulates the traffic in the access router connecting the stub network [54].

Local Resource Availability

When considering local resource availability, it means to find from the point of the PDB how much bandwidth for each PDB per edge-to-edge is left for reservation. In this thesis it is assumed that each PDB only consists of one PHB (Note in real network the same PDB along different edge-to-edge pairs may be implemented by different PHBs) and the resource availability is only restricted by the core resources (since the edge resource constraints can be solved straightforwardly). Since the resources of a link are partitioned to three PHBs and each PHB has different performance and needs different amount of resources to guarantee this performance, each PHB has different view about the resource availability. When three PHBs share a resource, for example: 10% capacity for EF, 30% for AF, the remaining for BE, each PHB will calculate the resource availability based on its own share of these resources. The resources allocated for one PHB can also be increased at the expense of the shares of other PHBs. Also, there is a limit for the share of EF and AF since at higher utilization rate of a link, if most of the traffic is EF or AF, the performance for EF or AF will not be guaranteed. To compute the resource availability, the Local Resource Availability module will get all the relevant information from Local Resource Manager.

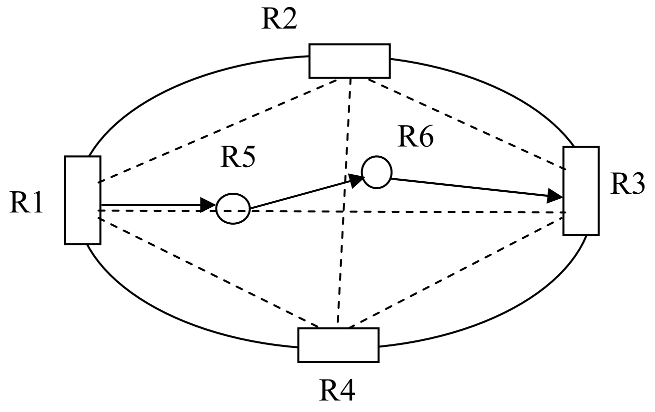


Figure 5.4: Calculation of Resource Availability

The local resource availability on per PDB is calculated per edge-to-edge pair basis. Figure 5.4 is used to illustrate the concepts:

In the DS domain of Figure 5.4, the R1, R2, R3, R4 are edge routers or access routers, the R5 and R6 are core routers. All border or access routers are connected in full-mesh, meaning that all traffic from one edge router can be forwarded to any other edge router directly (Full-meshed is not necessary, but it is used to illustrate the complexity). The dotted line stands for a virtual link. The solid line stands for a physical link. For simplicity only Golden PDB is considered on path from R1 to R3, which is implemented using EF PHB, and Silver PDB on same path, which is implemented using AF PHB. This path includes two core routers: R5 and R6 and three links: (R1-R5), (R5-R6), (R6-R3)). Every edge router is regulated by the SLS with the neighboring domain (DS domain and Stub networks). For simplicity, only the bandwidth requirement is considered and the route in each DS domain is assumed static (This is not a prerequisite, but under this assumption, the following discussions on the bandwidth availability of virtual links make sense. Otherwise, the bandwidth availability of virtual links need to be updated whenever there is a route change.). The traffic volume allowed in each SLS is considered as reserved. So if all the SLSs are not changed for some time, the edge router can be thought as static configured. Every edge router has static reservation bandwidth for each PDB and the corresponding PHB is also static in this small time scale. It is assumed that the link capacity is C_{ij} , the allocated EF traffic TE_{ij} and AF traffic is TA_{ij} for link (i, j) respectively. Also it is assumed there is a percentage limit L_{EF} (a percentage of C_{ij}, same for all links) for EF traffic, which limits the EF traffic not exceeding a level of the total capacity, and a percentage limit for AF or for the total of EF and AF traffic L_{AF} (since under this limit all volume allowed for EF and AF can be allocated to AF, the limit for AF and the total of EF and AF is equal) which limits the AF and the total of the EF and AF. So for a link (i, j) the bandwidth availability for EF is:

$$AE_{ij} = \min(C_{ij} * L_{AF} - TE_{ij} - TA_{ij}, C_{ij} * L_{EF} - TE_{ij}) \quad (1)$$

For AF:

$$AF_{ij} = C_{ij} * L_{AF} - TE_{ij} - TA_{ij} \quad (2)$$

For a path p the bandwidth availability for Golden PDB is:

$$AG_p = \min\{AE_{ij} \mid (i, j) \in p\} \quad (3)$$

For Silver:

$$AS_p = \min\{AF_{ij} \mid (i, j) \in p\} \quad (4)$$

Considering the Golden aggregate from R1 to R3, it is assumed that the links R1-R5, R5-R6, R6-R3 all have capacity 10Mbs, the R1 and R3 have enough capacity, the link utilization rate is 50%, 80% and 60% (including EF, AF and BE) , the EF traffic is 1Mbs, 2Mbs and 2.5Mbs respectively, and the AF traffic is 2Mbs, 3Mbs, 2Mbs respectively. It is also assumed in every link, the EF traffic can not exceed the 40% of the link capacity and the total of AF and EF can not exceed 70%. So the available resource for EF in R1-R5 is 3Mbs, in R5-R6 is 2Mbs, in R6-R3 is 1.5Mbs and the resource availability for Golden is 1.5 Mbs, meaning the network may at most increase 1.5Mbs bandwidth reservation for Golden traffic from R1 to R3 in response to a SLS adjustment. The resource available for AF in R1-R5 is 4Mbs, in R5-R6 is 2Mbs, and in R6-R3 2.5Mbs respectively. So the resource availability for Silver is 2Mbs. Since it is to calculate the resource availability for resource reservation use, only the resource availability for EF and AF is considered. In this way, the traffic availability is calculated for each edge-to-edge pair in both directions. So if there are n edge nodes in one DS domain, there will be *n(n-1)* resource availabilities computed each for Golden and Silver. The results computed are outputted to the database.

Local Resource Pricing:

From the resource availability obtained above, the Local Resource Pricing module determines how to price the local resources for the remote reservation. Viewing each edge-to-edge pair between two edge routers in each direction as a virtual link, the bandwidth availability is the bandwidth the BB wants to sell to the other DS domains. Higher bandwidth availability means there are more supplies and the unit price for the bandwidth should be lower to stimulate the sales. Here a threshold-based approach is used. Assuming there are l thresholds $A_1 < A_2 < ....A_{l-1} < A_l$ (Available Bandwidth) and the corresponding unit prices are $p_1, p_2 ....p_{l-1}, p_l$ respectively, the formula is:

$$p = p_i, \quad A_{i-1} \le A < A_i, 1 \le i \le l \quad (5)$$

Also there must be

$$p_i A_i < p_{i+1} A_{i+1}, \quad 1 \le i \le l \quad (6)$$

Formula (6) means if there are more bandwidths sold, there should be more revenues. Also note the above formulas only determine the prices for one PDB. How to determine the price levels and threshold bandwidth are both a technical and an administrative issue.

Figure 5.5 illustrates a pricing plan for three level thresholds for one PDB of an edge-to-edge pair. All the prices and thresholds for every edge-to-edge pair of each PDB can be listed in a table. So when a price corresponding to bandwidth availability is to be decided, only the table needs to be consulted. In this way all the prices for every class of every virtual link can be determined based on current network status. These unit prices and the amounts of the bandwidth available are stored in the database for distributing to other DS domains and also as a reference for the Local Resource Seller to decide whether a deal with BB in other DS can be made.

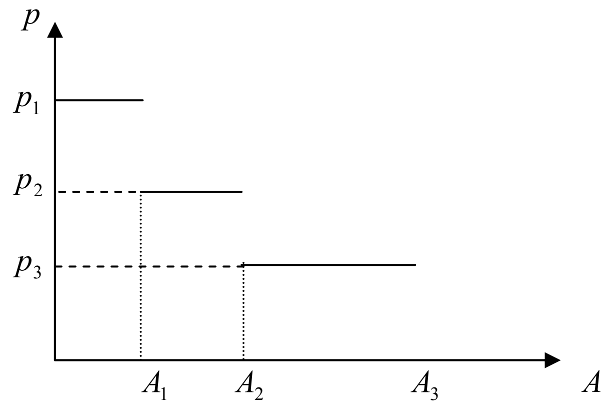


Figure 5.5: Threshold Pricing of the Resource

QoS-based Routing Module

In this thesis, QoS based source routing is used to find the route and update BGP (Border Gateway Protocol) table and the underlying BGP is assumed QoS-unaware. If BGP supports QoS routing, the same functionality can be partially achieved. But there are two advantages for the proposed routing in this thesis over BGP with QoS support. It can easily add in other policy factors except QoS by changing the price of a virtual link through a BB in a domain. The second advantage is that it has no prerequisite to the network infrastructure whereas BGP with QoS support at least demands all edge router have QoS routing capability. MPLS can help in this proposed routing since MPLS can provide the intra-domain routes to guarantee QoS performance while the proposed routing provides the inter-domain routes. Of course, to use MPLS, edge routers or switches must have MPLS capability.

When viewing the path between two edge routers in a DS domain as a virtual link and the entire network as a virtual network based on PDBs, then an aggregate originating from one DS domain to the destination DS domain passes through the path comprised of these links and the links between one DS egress edge router and one ingress edge router of the following neighbor DS. If the packets are routed in the traditional way (i.e. all EF, AF and BE packets from same source to same destination traverses the same route), even the cost is associated with the available bandwidth, both the resource availability through the route and whether this route can really meet the requirements cannot be decided (since one domain does not know the states of other domains). So even the resources are reserved traversing the route, it has a high possibility that the reservation will be unsuccessful. For a suitable path, there are following expectations:

- The path can support the QoS requirements of the aggregate requesting resources such as bandwidth, delay, delay variations etc.
- The path should consume a minimal amount of resources, meaning this path should pass by a least number of DS domains.
- The path should incur the least cost, meaning this path is a route with lowest price.
- The path may consist of different PDBs in different DS domain.

For the fourth goal, it is indeed a problem of Optimal Partition of QoS Requirements with Discrete Cost Function [55], which is computation intensive. In this thesis, only the case that an aggregate uses one PDB end-to-end is considered, thus only the first three goals are left.

Since standard routing algorithm is called single objective optimization, the proposed QoS algorithm can be called three-objective optimization. Achieving three goals simultaneously is impossible since finding a path that meets multiple QoS constraints is a NP hard problem. A heuristic method is used to make trade-offs among these goals in order to get a quasi-optimization result. For the first objective, it is assumed one aggregate uses a kind of PDB of service implicitly, which means that the service can meet the requirements of delay, delay variance since these services may be standardized (for example, by using EF and a lower percentage limit to guarantee the delay and jitter), all the nodes attending the classification are aware of these services and given the bandwidth requirements and the types of the PDBs a BB knows how to allocate other resources such as buffer size. So the delay and delay variance requirements are eliminated from the objectives of the first goal and only the bandwidth requirement need to be mentioned. Since if a link does not have enough bandwidth, this link will never occur in a routed path, by eliminating all the links that do not have sufficient bandwidth from the virtual network based on PDB, a pruned network is generated with every link having enough bandwidth to support the reservation. The last two goals have also been merged into one goal by making a trade-off between them. Since whenever the traffic passing by a DS domain along the path must traverse a link between the egress router in previous domain and the ingress router in the next domain, by giving such links a fixed cost (or price), the number of DS domains the aggregate traverses can be effectively restricted. In this way, in a pruned network, this becomes the standard Shortest Path First Routing (OSPF). The algorithm works in the following way to find a path for an aggregate reservation:

- The QoS routing module runs the algorithm in response to a request of the path for an aggregate originating from local DS from the Remote Resource Buyer.
- The QoS routing module gets the data for local DS and remote DS from the database, which includes two metrics: the bandwidth and the link price.
- The routing process first prunes the network by eliminating all the links without enough bandwidth.
- The module then runs an OSPF algorithm (such as Dijkstra's algorithm) to find the cheapest path. When encountering two equal price links, it is preferred to route to the link with more available bandwidth since this will enhance the load balancing and admit more requests.
- After the above steps, the module gets a best route with all nodes in the pruned virtual network on the route listed. The module then outputs the route information to the Remote Resource Buyer, which will use this to do the reservation.

This on-demand QoS based source routing has the computation complexity comparable to the standard single objective routing. Compared to the pre-computed routing algorithm, there are two benefits for this algorithm. One benefit of this algorithm is by using the most recent link metrics available, it can yield better routes. The other benefit is the associated storage saving [56].

Note in the above discussion, the QoS routing is dealt with based on PDBs, i.e. different PDBs have different network topologies (a virtual link occurring in Best-effort may not occur in Golden because of insufficient capability), state metrics (for example, same link has different cost for three type of PDBs) and so on in the same physical network. In this thesis, three virtual networks Golden, Silver and Best-effort coexist at the same time based on one physical network. The routing for Best-effort may be QoS unaware and the traffic of different PDBs from the same aggregate may travel along different paths.

Remote Resource Buyer

The Remote Resource Buyer in the BB is responsible for the reservation of the remote resources in other transit DSs when it receives a request from local stub networks. Notice the cancellation of the previously reserved resources can also be viewed as a request for reservation for the negative resources. Different from the normal reservation, this kind of reservation always succeeds and the previously reserved resources be released. Since the functionalities of the Remote Resource Buyer are mainly about the inter-domain communication, it is left to the next section for the detailed discussion.

Local Resource Seller

When as a transit network or destination side network for an aggregate, the Local Resource Seller in such a DS domain needs to decide if it will accept the resource request from other BBs, allocate the resources for the successful reservation and release the resources not needed. Since its functionalities are also mainly about inter-domain communication, they are left to the next section for the detailed discussion.

Traffic Monitoring Module

The traffic-monitoring module collects traffic information from local nodes and output to two modules: Local Resource Manager and Traffic Accounting and Charging Module. How the Local Resource Manager uses the information is already described in the Local Resource Manager part. How the Traffic Accounting and Charging Module uses the information will be described in the last section.

Traffic Accounting and Charging

How this module works will be described in details in a separate section since the Accounting and Charging targets are other DS domains reserving the transit services.

The Database

The database is used to store all the information the BB uses to fulfill its various functions. The data are classified into four categories and for each category one table is defined to store the data.

- The aggregate information table: This table is used to store the data from Local Resource Seller, which includes these columns: source DS identification, destination DS identification, PDB, start node, last node, reserved bandwidth, unit price, start time, stop time and status. Source DS identification is the domain number of the DS whose stub network the aggregate originates from. Destination DS identification is the domain number of the DS whose stub network the aggregate destines to. PDB is the service the aggregate receives. Start node is the ingress interface IP address of an aggregate in this DS domain. Last node is the egress interface IP address of an aggregate in this DS domain. Reserved bandwidth is the amount of the resources reserved for the aggregate in the virtual link from start node to last node in this DS domain. Unit price is the bandwidth price per unit when negotiating the deal or fulfilling the deal. Start time is the time the DS begins to service the reservation. Stop time is the time such service stops. There are three values for the status: “Enabled” when a deal has been made and the DS provides the bandwidth to the aggregate, “Processing” when the BB is doing the processing and the decision is not made yet. “Obsolete” when the remote BB has changed the reservation and a new entry emerges to the table and this entry remains only for record use.

- Accounting and charging table: This table stores the data from the Traffic Accounting and Charging module, which includes these columns: remote DS identification, PDB, start node, last node, reserved bandwidth, charge, overused bandwidth, penalty, start time, stop time and status. Remote DS identification is the domain number of DS requesting the service, i.e. the DS domain that is the source DS reserving the PDB from the start node to last node, regardless of their destinations. The PDB, start node and last node has the same meanings as in the first table. Reserved bandwidth is the amount of the bandwidth a remote DS reserved from the start node to the last node in this DS domain, regardless the destination DS domain. Charge is the revenue the DS domain should get from the remote DS domain for the reserved transiting service. The overused bandwidth is the records of the violation of contract by the remote DS domain by overusing the bandwidth it reserves. The penalty is the amounts of money the side that violates the contract pay the other side. A negative overused bandwidth means the DS domain does not provide the sufficient resource for the reservation. A negative penalty means the DS should make compensation to the remote DS. Start time is the beginning time the BB records the charging information. Stop time is the time the reservation changes and a new recording entry should be created. The status column can have two values “Enabled” and “Obsolete”.
- Local Resource Pricing table: This table stores the information from the Local Resource Pricing, which is used by the seller to broadcast the price to other DSs, by Accounting and Charging module to calculate the charges and by the Remote Resource Buyer to calculate the end to end total cost of a request. It has these columns: start node, last node, PDB, available bandwidth, unit price, start time, stop time and status. Available bandwidth is the amounts of the bandwidth the BB

decides can sell to the other DS domains to support their transiting traffic through this virtue link with respect to the current network load. The unit price is the price the Seller advertises to other BBs also with respect to the current resource availability. Status column can have two values: " Enabled" means the information is the most recent and should be used by all relevant modules, "Obsolete" means this entry is only for record use.

- Remote Resource Pricing Table: This table stores the remote resource availability and pricing information, which is used by the QoS Routing Module to compute the best route and also by the Remote Resource Buyer to compose the reservation request message. It has these columns: DS domain number, start node, end node, PDB, available bandwidth, unit price, start time, stop time, status. The DS domain number is the identification for remote DS domain. Other columns have the same meanings with the Local Resource Pricing table except this table is used to store the Remote DS information.

5.3 Resource Reservation in Transit Domains

The Inter-Domain Resource Reservation Protocol:

The reservation goal for an aggregate in a transit network is to reserve enough bandwidth between an ingress point and an egress point [23]. The inter-domain communication is limited only between neighboring domain BBs. Since to guarantee the QoS of an aggregate end to end, a BB must also reserve the bandwidth from the remote BBs that are not its neighbors, the resource request information must be relayed by its neighboring BB and the neighboring BB's neighboring BB and so on to the end along the source routed path, and the decision of each BBs is relayed along the reverse path to the source. For the request message, in each domain, the BB unpacks the request information, takes out the information it needs from previous hop BB, add its own request information to the request message, and relay this new message as a new request to the next hop BB. For the decision message, if the reservation is rejected, the destination BB will relay the reject decision to the source BB and also cancel the request information along the reverse path. If the decision is accepted, the destination BB will also relayed the confirm decision to the source BB. But unlike the reject decision, the BBs along the reverse path also install the reservation state in its relevant devices. To guarantee the accuracy of the remote DS resource availability information, it is necessary to broadcast such information to all the BBs in a timely fashion. So a protocol suitable for such a communication should have request, decision, report messages. There are many protocols with such characteristics, such as RSVP, COPS, and SNMP etc. Here the COPS protocol [28] is chosen as the inter-domain resource allocation protocol to elaborate the resource request and reservation procedure, since it is also a standard protocol recommended by IETF to manage the intra-domain policy and resources [29]. Its request, decision and report message is suitable for this use. The COPS protocol provides an Integrity object that can achieve authentication, message integrity, and replay prevention [28]. So it also satisfies the secure consideration in inter-domain communication. Because it uses the TCP as its transportation base, it is reliable and all the messages are assumed reliably delivered. In chapter 3 and 4 the details of the COPS protocol have been discussed already. The request information could be encapsulated in the CLIENTSI part of the COPS request message, the reservation decision information should be encapsulated in the DECISIONS part of the COPS decision message and the broadcast information about the resources and prices could be encapsulated in the CLIENTSI part of the report message. The client handle object in these three types of the messages could be used to distinguish a request, a report or a decision to the specific request from other same type of messages. Different from the intra-domain resource management framework, in which a BB is always a policy server and the router or other devices such as servers, hosts as the clients [19], a BB in the inter-domain framework is both as a client and a server of its neighboring BBs. When issuing a request or a report message, the BB plays the role of a client of its neighboring BB. When issuing a decision message, the BB plays the role of the server of its neighboring BBs. The architecture of the BB in this model is only a logical view according to the functionalities since both the decision and report message are issued by the Local Resource Seller, i.e. the Seller both plays the role of client and server.

The Reservation Strategy

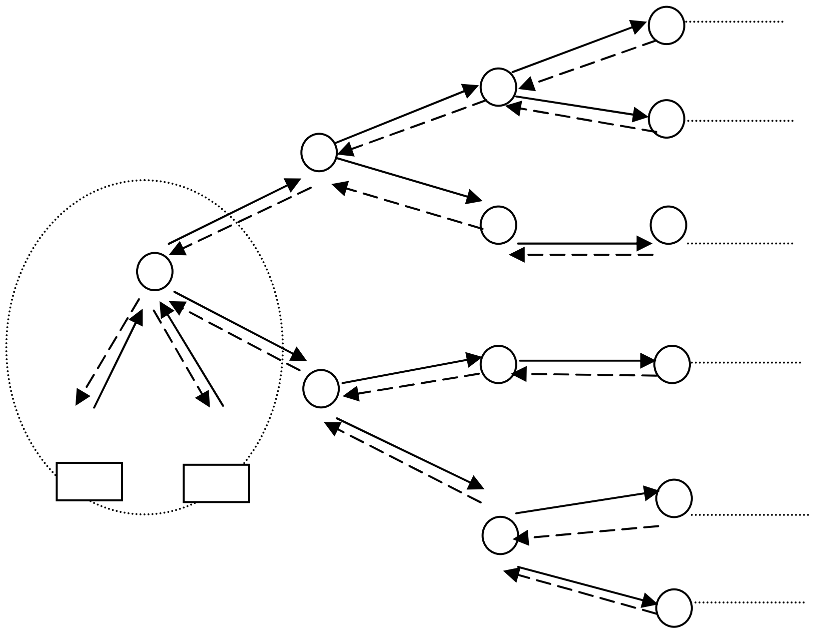


Figure 5.6: Resource Reservation Process

Figure 5.6 is used to illustrate reservation request and decision strategy. In this figure, the circle stands for the BBs in the DS domains related to the reservation. For every pair of circles if a directed line links them it means that these two BBs are neighboring BBs. The big dotted circle stands for a DS domain that issues a reservation request. The rectangle stands for the local nodes in a DS domain. A directed solid line stands for a reservation request from one client BB to one server BB. A directed dotted line stands for a reservation decision from one server BB to one client BB. The undirected dotted line stands for the omission of the reservation path.

When a BB in a local DS domains receives one or more resource requests through the local nodes from its stub networks that want to send traffic outwards, if the destinations of the aggregates are not on the local DS domain, then the Remote Resource Buyer deals with these requests. In the Remote Resource Buyer, the aggregates are further divided into individual aggregate with respect to the destination prefix. Each aggregate is called an O-D (Origin-Destination) pair. For each pair, its source and destination address, class of service and bandwidth requirements are outputted to the QoS routing module, which uses the information from Remote Resource Pricing table to find the best route for this aggregate. The best route then is sent back to the Remote Resource Buyer. After all the routes of the requesting aggregates are obtained, the Buyer first determines the local resources these aggregates need. If the local resource can satisfy these aggregates the BB will freeze this bandwidth until a decision message returns. It then from the routes decides what egress routers in the local DS these aggregates pass by, i.e. what neighboring DS domains these aggregate traverse. For each such neighboring domain, a request is issued to the BB in it.

The CLIENTSI part in the reservation request includes information following the format in Figure 5.7:

Type	O-D pair 1	Source Route	DS number 1	Bandwidth	Unit Price	DS number 2	
Bandwidth	Unit Price	……	DS number n	……·	Allowance	O-D pair 2	……

Figure 5.7: Resource Request Format

For each field, the meaning is described as follows:

- Type: there are four types of request messages in this paper: Golden resource request, Silver resource request, Golden resource release, and Silver resource release.
- O-D pair: the identification of the source and destination of the aggregate, possibly the network prefix pair. There maybe more than one O-D pair in a request.
- Source Route: the route getting from the QoS routing module. This is in the form of the concatenation of ingress and egress router addresses of the DS domains on the path. For the source domain, there is only an egress router address and for destination domain only an ingress router.
- DS number: the identification number of the DS domain along the source route, there are usually more than one DS along the path for each O-D pair.
- Bandwidth: the amounts of bandwidth the Buyer want to reserve from the DS domain identified by the DS number.

- Unit Price: the unit price the buyer can afford for buying the bandwidth from above mentioned DS.
- Allowance: this is a policy-based value used to support the reservation. If a buyer wants to pay a price more than in the unit price field, it may give this field a positive value. When it encounters a BB asking for a price more than the unit price, the difference will be deducted from the allowance.

The DECISIONS part in the reservation decision includes information following the format in the Figure 5.8:

Type	O-D pair 1	Source Route	DS number 1	Decision	Reason	DS number 2
Decision	Reason	…	DS number n	…	O-D pair 2	…

Figure 5.8: Decision Format for Resource Reservation

- Decision: this is the answer whether the resources requested in the DS domains on the path are reserved or rejected.
- Reason: if this reservation request is rejected, the BB in the relevant DS domain will give a reason.

All the fields with the same name as in the request message have the same meanings.

After the BB in the neighboring domain receives the requests, the Local Resource Seller consults the Local Resource Pricing table to decide if this request can be satisfied. Because the information about the resource availability and price may be out of date due to such information is not delivered to the domains issuing the request in time or the resources just have been allocated to other request, the BB may decides that the request can not be satisfied as the resources are not enough or the price of the Buyer can pay is too lower. If it is the first case, the request will be rejected and the reject decision message will be relayed back to the source. If it is the second case, it will check the allowance field. If this field is a positive value and more than the price difference, then it will accept the request and use the value after deduction to update the Allowance field. Otherwise, a reject decision is issued. After the Seller checks all the O-D pair in the message and makes the decisions for each O-D pair, it will output the information for each accepted O-D pair to the Aggregate Information table and set the “Status” field to “Processing”. It also deletes the fields related to its own DS number for each O-D pair from the request. If at this stage, all the O-D pairs in the request still have the same next-hop DS, the request will be forwarded to the BB in the next-hop DS. If not all the O-D pairs have the same next-hop, then for each next-hop, a new request is generated from the old request with the O-D pairs having the same next-hop aggregated in the same new request message. The request eventually gets to the leaf of the request tree. If in each leaf, the request can still be satisfied, then the reservation first be installed in the leaf DS domain, which means the Aggregate Information table is updated, the “Status” field for the updated entry is set to “Enabled” and the Ingress router forwarding table installs the new entry for these aggregates. Then this decision is forwarded to its previous-hop BB. At this stage apart from the actions in the leaf DS domain, the previous-hop BB also installs the new entry in the forwarding tables of relevant egress routers and merges the decision messages from the branches to a new decision message by adding the decisions for each O-D pair having the same last hop BB to a new message. This new message is relayed to its previous hop BB in turn in reverse tree until the last decision message gets to the source BB. After confirming that its local resources are also allocated, the stub networks can send the traffic according to the new bandwidth reserved.

To keep a high reservation success rate, the remote resource availability information in the databases of all the DS domains must be accurate and timely. The report message takes the same flooding mechanism as the link state update message in the routing protocol. The format of the CLIENTSI in the report messages is shown in Figure 5.9. For each field, the meaning is as follows:

- Type: i.e. Per Domain Behavior, Golden and Silver.
- DS number: the DS identification number of the domain first issuing the report message.
- Virtual Link: the edge-to-edge router pair number, possibly the IP addresses of these pairs.
- Bandwidth: the amount of bandwidth for this type available at this DS domain.
- Unit price: the unit price of the above bandwidth.

Type	DS number	Virtual Link 1	Bandwidth	Unit Price	…	Virtual Link 2
Bandwidth	Unit Price	…	Virtual Link n	…		

Figure 5.9: The CLIENTSI Format

Because for a DS domain with n edge routers and two PDBs, there may be at most 2n(n-1) resource availability entries for each report message if the edge routers are full-meshed. The report information may be tremendous. To reduce the overhead, the BB may not issue periodical refresh message but only issue a report message when there is updated information for the Local Resource Pricing and only includes in the report message the updated entries.

5.4 Traffic Monitoring, Control, Accounting and Charging

Note although the reservation is end to end for an aggregate, there are explicitly only bilateral agreements existing as the result of the reservation. The BBs know the details about the reservation for each aggregate. But for the egress router of an upstream domain and the ingress router of its next hop domain, they are not aware of the details of the individual reservations, i.e. there are no individual QoS mechanism such as classifier, policer, scheduler and so on installed for each aggregate. The router only knows how to forward these aggregates to their next hop. The SLS between the two neighboring domains is still in a form like this: "The EF traffic from the egress router A of Domain X entering the ingress router B of the Domain Y can not exceed an amount of bandwidth C". The ingress router normally performs in the same manner the traffic control functionality, as there is only one aggregate reservation of the amount of C traffic. Or if not incurring the scalability problem, for example if the number of DS domain is not too many, the ingress router may install the individual QoS components for each domain and controls the traffic of all the aggregates from the same DS domain based on the amounts of the reserved bandwidth of this DS domain in the records of the database. In this scenario, the SLS is like this: "The Golden traffic whose source domain is Z from the egress router A of Domain X entering the ingress router B of the Domain Y can not exceed an amount of bandwidth C". For the first case, the BB may collect the traffic statistics in each ingress router for each PDB from each domain. If the total traffic between one egress and the corresponding ingress does not exceed the amounts prescribed in the SLS, the BB does nothing with the traffic. If the total traffic exceeds the prescribed amounts, the BB will find which DS domain originates the violating traffic through the traffic-monitoring module and it will possibly issues a notification message, which will be relayed to the violating DS domain. If upon receiving the message, the violating DS BB does not regulate its traffic, the local BB may take different actions depending on the network status. If the violating traffic does not deteriorate the QoS of other traffic, the BB just record this violating record in its database. If it does deteriorate other traffic, the BB may make the traffic conditioner in the ingress router to drop the packets of the violating traffic and the BB of the violating domain will have to do the traffic control on the violating traffic since if not the QoS of its entire traffic transiting the remote DS domain will degrade.

For the second case, it works in the similar manner as the first case except in this case since the aggregates from the same DS domain have an individual traffic conditioner, it can perform the traffic control directly through the local routers.

The BB can collect the traffic statistics for each DS domain in a local DS domain from the edge routers. For each aggregate, based on its reservation and unit price, the amounts of the money it should pay for this service are calculated. From these data the total amounts incurred for a DS domain for the transiting service can also be obtained. These amounts may also include a penalty charge for its overuse of the reservation. The charging for the reservation may take different approaches. One DS domain may only charges its neighboring DS domains for the transiting service. In this approach, every DS domain acts as the agents of its neighboring domains. The billing issues and the billing relationships between the domains are very complicated since each aggregate traverse different domains and incurs different charges. It is difficult to decide if a charging contract is fair, since the charged DS domain may not send most of the transit traffic used to calculate the charges. An alternative is the remote billing, i.e. for a DS domain providing the transit service, the DS domains sending the aggregate should pay the charges incurred by an aggregate. This scheme is fairer than the first approach since "who use, who pay". Also, because a DS domain must pay for the transit service it reserved in the remote DS domain, the DS domain will tend to choose the path incurring least cost, and there will be more competitions between the DS domains, which make the network more efficient.

5.5 Simulation Results

Network Simulator 2 [57] is used to run the simulations to investigate that how MLTTM improves the load balancing among alternate domains and network efficiency in terms of resource availability and prices, and how different factors affect the effectiveness of this model. The PHB instead of PDB is used in this simulation since this topology is relatively simple.

The simulation topology is shown in Figure 5.10, where SE, SA, SEB, SAB, SBB are sources of EF, AF, EF background, AF background and BE background traffic respectively; RS is the edge router of source domain; E11, C11 and E12 are edge routers and core router of transit domain 1; E21, C21, E22 are edge routers and core router of transit domain 2; DE, DA, DEB, DAB, DBB are destinations of EF, AF, EF background, AF background, BE background traffic respectively.

The bandwidth capacity of links E11-C11 and C11-E12 in transit network 1 is $9Mbits$ and $10Mbits$ respectively. The bandwidth capacity of links E21-C21 and C21-E22 in transit network 1 is $7Mbits$ and $10Mbits$ respectively. So the bottleneck link in DS domain 1 is E11-C11 while in DS domain 2 is E21-C21. AF background traffic is a Star War traffic trace file. All other traffic is Pareto on-off traffic arriving according to a Poisson process with an inter-arrival time $1/\lambda$, and a holding time $1/\mu$ with their parameters shown in Table 5.1. The resource availability is represented by bandwidth price and capacity limit. In this simulation, EF traffic cannot exceed 30% of the link capacity, the total of EF and AF traffic cannot exceed 70% of the link capacity and the remained capacity is left to BE. The scheduler mode is priority queuing with the highest priority to EF and lowest to BE. Before a source generates a flow it first checks with its local store of resource availability information and choose the cheaper transit domain to send the request. When receiving a request the BB in transit domain will check its local resource availability and decide whether to accept this request. If the request is accepted, the source will send traffic across this domain, otherwise it is blocked. Regardless of the status of the network, once resources are reserved for a flow, the resources must be provided during the life time of the flow.

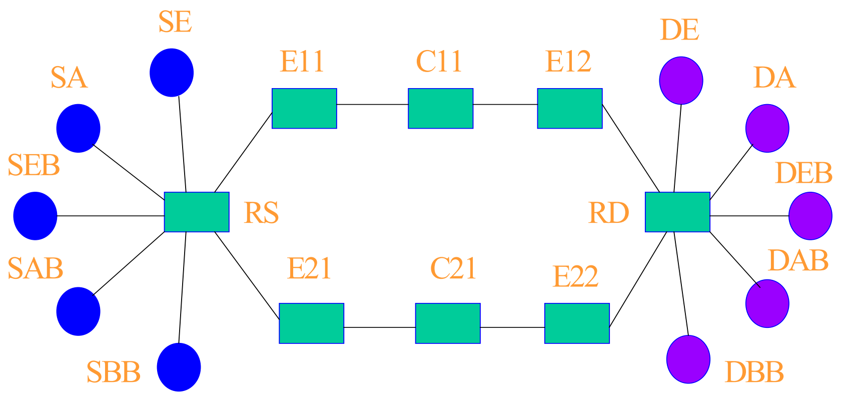


Figure 5.10: Simulation Topology

Traffic Source	SEB	SBB	SE	SA
Arrival Interval (*s*)	4	24	0.6	0.4
Holding Time (*s*)	60	120	20	20
Type & Peak Rate (*k bits*)	UDP (64)	TCP (64)	UDP (64)	UDP (64)
Burst Time (*ms*)	500	500	500	600
Idle Time (*ms*)	500	500	500	400
Packet Size ((*bytes*)	125	125	125	125
Shaper Parameter	1.5	1.5	1.5	1.5

Table 5.1: Traffic in the Simulation

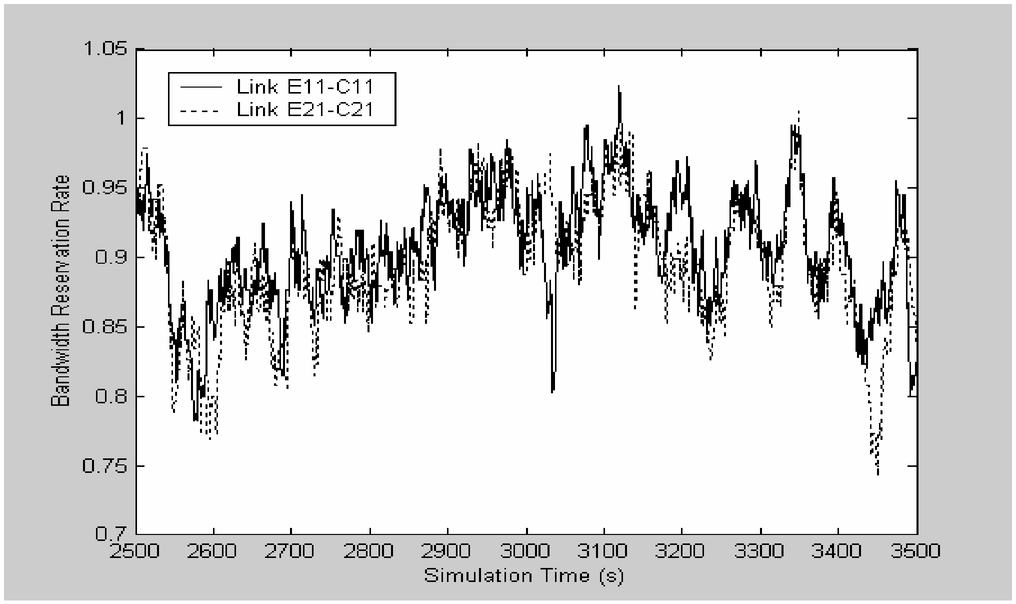


Figure 5.11: Bandwidth Reservation Rate under MLTTM

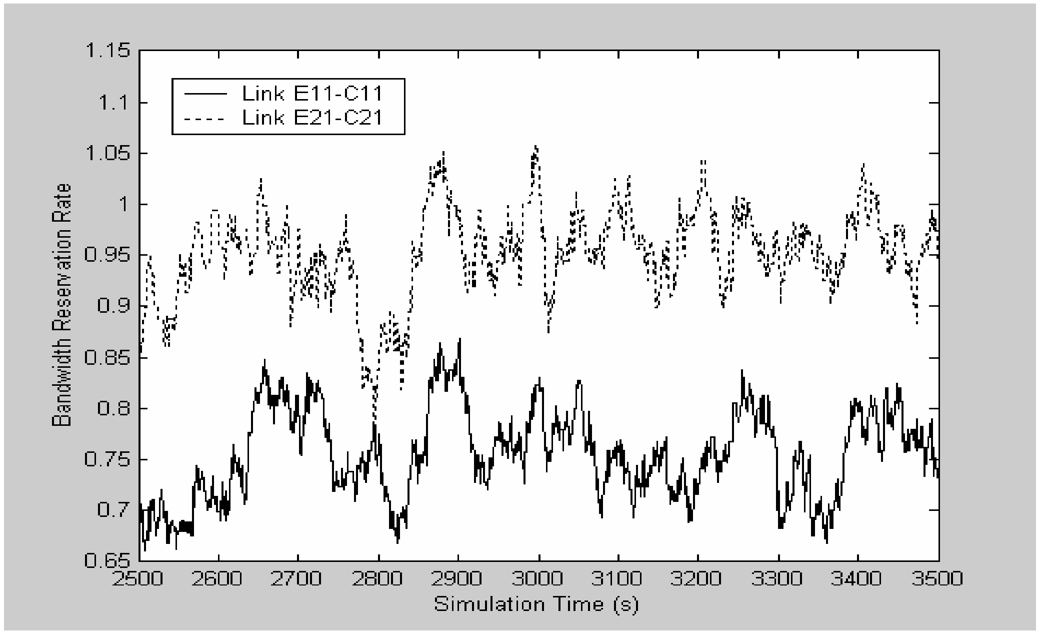


Figure 5.12: Bandwidth Reservation Rate under Random Model

The bandwidth reservation rate for high priority traffic (the ratio of the total reserved EF and AF traffic to their limits) under MLTTM is shown in Figure 5.11, and that of the two-tier model in which a source randomly selects a transit domain is shown in Figure 5.12, which is called a Random Model. The pricing scheme for these two figures is shown in Table 5.2. Availability is the ratio of available bandwidth for AF or EF to the bandwidth limit for AF or EF. For instance, the column value 0:1/32 corresponding to 5.0 means if there are resources not exceeding 1/32 of its limit available, the price should be 5.0. In Figure 5.11, load is balanced in MLTTM because of source-based QoS routing and threshold-based pricing. In Figure 5.12, because load is not balanced, some resources are wasted in transit domain 1 while at the same time the resources in transit domain 2 are heavily used.

To investigate the efficiency of the MLTTM, the network utilization rates between these two models are compared in Figure 5.13. The network utilization rate is the ratio of total reserved traffic to the total limit for the reservation. A higher utilization rate means the entire network has a high throughput, i.e. higher efficiency. Note in this simulation the ratio of link capacity limit of two domains is 9:7. If the ratio is larger, the effect will be more obvious.

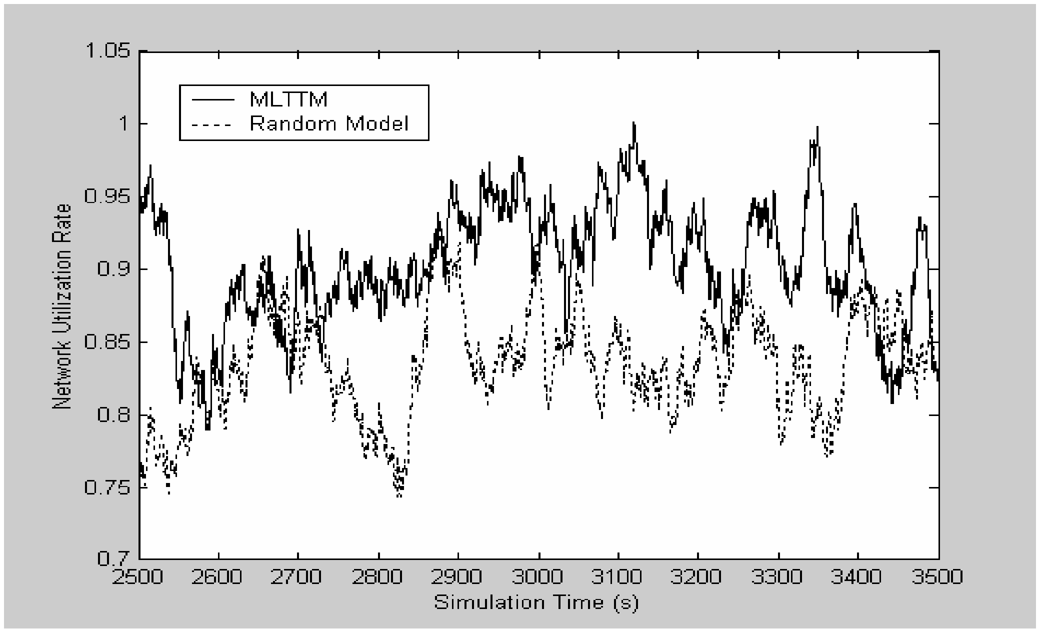


Figure 5.13: Network Utilization Rate

Availability	0	0:1/32	1/32:1/16	1/16:1/8	1/8:1/4	1/4:1/2	1/2:1
Price	6.0	5.0	4.0	3.0	2.0	1.5	1.0

Table 5.2: Fine-grained Pricing

Availability	0	0:1/4	1/4:1/2	1/2:1
Price	6.0	2.0	1.5	1.0

Table 5.3: Coarse-grained Pricing

Line	1	2	3	4	5	6
Min Interval (s)	0.4	1.2	1.2	0.4	1.2	1.2
Graininess	fine	fine	coarse	fine	fine	coarse
Ratio	block	block	block	total	total	total

Table 5.4: Legend for Figure 5.14

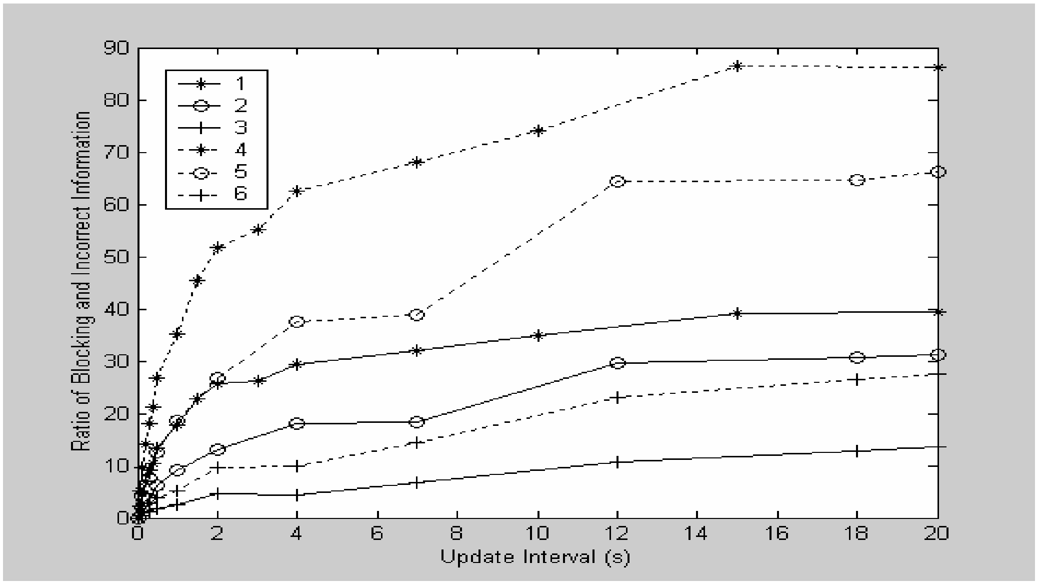


Figure 5.14: Reservation Performance Degradation under Different Situations

The performance of MLTTM is shown in Figure 5.14 in terms of the factors: the dynamic nature of the network traffic reservation (Minimum Flow Interval), the resource availability information update interval, and graininess of the price (fine or coarse, i.e. the number of thresholds for setting price). The graininess of price is shown in Table 5.2 and 5.3 respectively. Prices in Table 5.2 are fine grained while in Table 5.3 they are coarse grained. The legend of Figure 5.14 is shown in table 5.4. For MLTTM to be successful, the distributed information store should be relatively accurate and consistent, which will result in a lower blocking rate and lower waste of the resources. When current bandwidth price in a transit domain is higher than the outdated price by the source, there is a high probability for the request to be blocked (*block*). When it is lower, there is a high priority for the source not to issue the request (*miss*), which results in resource waste. In Table 5.4, *total* means the sum of *blocking* and *missing*. From this figure, it can be concluded that the more dynamic of the network traffic reservation, the longer of the update interval and the finer of the graininess, the more depredated the performance of MLTTM.

5.6 Conclusions

In this chapter, a DiffServ model MLTTM based on the two-tier framework to provide dynamic SLA negotiations is proposed. The way is described in details that the BBs in different DS domains cooperate to make the bilateral agreements consistent by performing the aggregate reservation of the resources. Specifically the issues are illustrated such as how to manage the intra-domain resources, how to price the resources, how to make the resource reservation reservations and how to monitor, control, and charge the traffic. There are alternatives for schemes proposed and the scheme in this chapter are mainly used to illustrate concepts. In this framework, the individual reservation complexity is left to the BBs and no complexities are added to the edge routers, so this framework is scalable. Also, it is compatible with the standards of the IETF and needs modifications for the current protocols. It should be noted by now, in current Internet, both the common data store of global resource availability and dynamic BGP table do not exist. To deploy this model still needs many efforts.

The resource availability scheme might be oversimplified since the intra-domain resource management and edge provisioning are among the most challenged issues in resource management area. Also, since in a very dynamic network, the resource availability and price information change frequently and may not be flooded to all other domains in a timely manner, it may results in high rate reservation failure. Thus the dynamic time scale of this model should be on several minutes.

Chapter 6

Conclusions

Policy-based networking has become increasingly important with the dramatically growth of Internet, diverse technology and vendor-specific devices, new emerging QoS-sensitive applications, threat to network security, quickly evolving nature of technology and lack of experienced network managers. Since DiffServ is possibly the standard future architecture for providing QoS, QoS policy-based networking for DiffServ attracts intensive investigation. This thesis has two contributions to this research aspect: implementation of a policy-based networking system, which provides policy-based admission control, and policy provisioning based on DiffServ PIB; and a novel model MLTTM for providing end-to-end QoS in DiffServ networks.

The concepts of COPS-RSVP are used to implement the policy-based admission control. It is novel in that a Bandwidth Broker acts as a policy client to allocate bandwidth in terms of the policies defined by a policy manager.

The DiffServ QoS PIB is used to define policy for provisioning edge devices of a DiffServ domain. Since the DiffServ QoS PIB is not yet a standard and no device is currently PIB-aware, some mapping work from PIB to Cisco commands has been performed.

The components of the implementation of policy-based admission control and policy provisioning have been described. Examples for both implementations are experimented in the test-bed and their results are validated.

A novel policy-based resource allocation model MLTTM is proposed. This model tries to use distributed common data stores to share resource availability information among BB peers and by pricing the resource in terms of availability to achieve fairness and efficiency for a DiffServ Internet. By using an explicit signaling reservation strategy for resource reservation and a flooding method to update the information in the common store, the model tries to establish a multilateral agreement among transit networks along the path for an aggregate.

Policy-based networking is still at its infancy stage. Many of the protocols for policy-based networking in Internet society are still Internet drafts. The commercial implementations of policy-based networking are proprietary and incompatible to each other. The scope for using policy-based networking is still limited to the intra-domain case.

The Internet-information superhighway is not built on single technology. The Internet infrastructure in nature is heterogeneous, which includes Packet-Switched technology such as Ethernet, Token Ring, FDDI, Frame Relay, Circuit Switched network such as SONET, DSL, CATV, connection-oriented such as Asynchronous Transfer Mode (ATM) and Optical Network WDM. Although the infrastructure of networks is transparent for IP, the features of the underlying infrastructure decide whether it is suitable for providing QoS. For example, ATM can provide QoS for flows with various characteristics such as Constant Bit Rates, Variable Bit Rates, Committed Bit Rates etc but it is difficult for Ethernet to do the same thing. This heterogeneity of infrastructure inevitably influences the standardization of the policy-based networking.

Until now all the policy-based networking systems currently released are limited to provide QoS in one administrative domain for the following reasons:

- There is no standard protocol defined for exchanging inter-domain policies.
- The resource allocation is complicated and difficult even in one domain because of the unpredictable nature of the traffic, not to mention multiple domains. Providing end-to-end QoS is a challenging issue in computing the resources needed.
- There is no feasible method for dynamic edge-to-edge provisioning.
- Although DiffServ is considered as the architecture of next generation Internet, it yet is not widely deployed in the Internet.
- Most of the applications are still not QoS capable, i.e. they are unaware of their resource needs.

- The policy-based networking systems from different vendors may be incompatible.
- The different network may not want to share information for security reasons.

Although faced with these difficulties, the trend for policy-based networking is to evolve slowly towards inter-domain implementations in some way in the future, enabling the Internet to finally be a global intelligent information system.

Bibliography

1. A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Herzog, A. Huynh, M. Carlson, J. Perry, S. Waldbusser, "Terminology for Policy-Based Management", RFC 3198, November 2001.
2. B. Moore, E. Ellesson, J. Strassner, A. Westerinen, "Policy Core Information Model – Version 1 Specification", RFC 3060, February 2001.
3. J. Strassner, A. Westerinen, Bob Moore, David Durham, Walter Weiss, "Information Model for Describing Network Device QoS Mechanisms for Differentiated Services", <draft-ietf-policy-qos-device-info-model-02.txt>, work in progress, November 2000.
4. Hugh Mahon, Yoram Bernet, Shai Herzog, John Schnizlein, "Requirements for a Policy Management System", <draft-ietf-policy-req-02.txt>, work in progress, November 2000.
5. R. Braden, D. Clark, S. Shenker, "Integrated Services in the Internet Architecture: An Overview", RFC 1633, June 1994.
6. J. Wroclawski, "The use of RSVP with IETF Integrated Services", RFC 2210, September 1997.
7. S. Shenker, C. Partridge, R. Guerin, "Specification of Guaranteed Quality of Service", RFC 2212, September 1997.
8. J. Wroclawski, "Specification of the Controlled-Load Network Element Service", RFC 2211, September 1997.
9. R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Procotol (RSVP) Version 1 Functional Specification", RFC 2205, September 1997.
10. Gene Gaines, Marco Festa, "A Survey of RSVP/QoS Implementation", Update 2, RSVP Working Group, July 1, 1998.
11. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

12. Information Sciences Institute, university of Southern California, "Internet Protocol, DAPRA Internet Program Protocol Specification", RFC 791, September 1981.
13. S. Deering, R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
14. V. Jacobson, K. Nichols, K. Poduri, "An Expedited Forwarding PHB", RFC 2598, June 1999.
15. J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, "Assured Forwarding PHB Group", RFC 2597, June 1999.
16. Y. Bernet, S. Blake, D. Grossman, A. Smith, "An Information Management Model for DiffServ Routers", RFC 3290, May 2002.
17. Dinesh C. Verma, "Policy-Based Networking: Architecture and Algorithms", ISBN: 1-57870-226-7.
18. Dave Kosiur, "Understanding Policy-Based Networking", ISBN: 0-471-38804-1
19. R. Yavatkar, D. Pendarrakis, R. Guerin, "A Framework for Policy-based Admission Control", RFC 2753, January 2000.
20. R. Droms, "Dynamic Host Configuration Protocol", RFC 2131, March 1997.
21. P. Vixie, S. Thomson, Y. Rekhter, J. Bound, "Dynamic Updates in Domain Name System (DNS UPDATE)", RFC 2136, April 1997.
22. Chen-Nee Chuah, Lakshminarayanan Subramanian, Randy H. Katz and Anthony D. Joseph, "QoS Provisioning Using A Clearing House Architecture", International Workshop on Quality of Service, 2000.
23. A. Terzis, L. Wang, J. Ogawa, L. Zhang, "A Two-Tier Resource Management Model for the Internet", Global Internet, 1999.
24. K. Nichols, S. Blake, F. Baker, D. Black, "Definition of the Differentiated Services Fields (DS Fields) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.
25. Klara Nahrstedt, Jonathan M. Smith, "The QoS Broker", IEEE Multimedia, 1995.
26. Dinesh Verma, "Supporting Service Level Agreements on IP Networks", ISBN: 1-57870-146-5.

27. "Cisco Policy Networking Solutions with PeopleSoft Financials", http://www.cisco.com/warp/public/cc/pd/wr2k/qoppmn/prodlit/ensps_br.pdf
28. D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, A. Sastry, "The COPS (Common Open Policy Service) Protocol", RFC 2748, January 2000.
29. K. Chan, J. Seligson, D. Durham, S. Gai, K. Mccloghrie, S. Herzog, F. Reicmeyer, R. Yavatkar, A. Smith, "COPS Usage for Policy Provisioning (COPS-PR)", RFC 3084, March 2001.
30. D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, J. McManus, "Requirements for Traffic Engineering over MPLS", RFC 2702, September 1999.
31. S. Herzog, "RSVP Extensions for Policy Control", RFC 2750, January 2000.
32. S. Herzog, J. Boyle, R. Durham, R. Rajan, A. Sastry, "COPS Usage for RSVP", RFC 2749, January 2000.
33. "COPS for RSVP", from Cisco Documentation, http://www.cisco.com/univercd/cc/td/doc/product/software/ios121/121newft/121t/121t1/copsrsvp.htm#xtocid235380
34. MySQL Reference Manual, http://www.mysql.com/documentation/mysql/bychapter/
35. QBone Architecture (v1.0), Internet2 QoS Working Group Draft, August, 1999, Editor: Ben Teitelbaum, http://www.internet2.edu/qos/wg/papers/qbArch/1.0/draft-i2-qbone-arch-1.0.html
36. QBone Bandwidth Broker Architecture, http://qbone.internet2.edu/bb/bboutline2.html
37. Draft Codes for a simple inter-domain BB specification, Rüdiger Geib, Document Version 2, http://qbone.internet2.edu/bb/parameter-requirementsV2.html
38. Draft SIBBS Version 1 RAR Code, Author: Rüdiger Geib, Document Version 1.0, http://qbone.internet2.edu/bb/RAR_code.html
39. Erik Lidén, Anders Torger, "Implementation and Evaluation of the Common Open Policy Service (COPS) Protocol and its use for Policy Provisioning", master thesis, Computer Science and Electrical Engineering Department, Luleå University of Technology, January 2000.
40. COPS 1.2.0, http://www.vovida.org/protocols/downloads/cops/

41. John Larmouth, “ASN. 1 Complete”, ISBN: 0122334353.
42. M. Fine, K. McCloghrie, J. Seligson, K. Chan, S. Hahn, A. Smith, Francis Reichmeyer, “Differentiated Services Quality of Service Policy Information Base”, <draft-ietf-diffserv-pib-02.txt>, work in progress, November 2000.
43. Y. Snir, Y. Ramberg, J. Strassner, R. Cohen, B. Moore, “Policy QoS Information Model”, <draft-ietf-policy-qos-info-model-04.txt>, work in progress, November 2000.
44. K. McCloghrie, M. Fine, J. Seligson, K. Chan, S. Hahn, R. Sahita, A. Smith, F. Reichmeyer, “Structure of Policy Provisioning Information (SPPI)”, RFC 3159, August 2001.
45. K. McCloghrie, D. Perkins, J. Schoenwaelder, J. Case, K. McCloghrie, M. Rose, S. Waldbusser, “Structure of Management Information Version 2 (SMI v2)”, RFC 2578, April 1999.
46. J. Case, K. McCloghrie, M. Rose, S. WaldBusser, “Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)”, RFC 1907, January 1996.
47. F. Baker, K. Chan, A. Smith, “Management Information Base for the Differentiated Services Architecture”, <draft-ietf-diffserv-mib-06.txt>, work in progress, November 2000.
48. M. Fine, K. McCloghrie, J. Seligson, K. Chan, S. Hahn, A. Smith, F. Reichmeyer, “Framework Policy Information Base”, <draft-ietf-rap-frameworkpib-03.txt>, work in progress, November 2000.
49. Don Libes, “Exploring Expect: a Tcl-based toolkit for automating interactive programs”, ISBN: 1565920902.
50. Sugih Jamin, Scott J. Shenker, Peter B. Danzig, “Comparison of Measurement-based Admission Control Algorithms for Controlled-Load Service”, INFOCOM, 1997.
51. K. Nichols, B. Carpenter, “Definition of Differentiated Services Per Domain Behaviors and Rules for their Specification”, RFC 3086, April 2001.

52. N. Seddigh, B. Namdy, J. Heinanen, "An Assured Rate Per-Domain Behaviour for Differentiated Services", <draft-ietf-diffserv-pdb-ar-01.txt>, work in progress, July 2001.
53. Van Jacobson, Kathleen Nichols, Kedar Poruri, "The 'Virtual Wire' Per-Domain Behavior", <draft-ietf-diffserv-pdb-vw-00.txt>, work in progress, July 2000.
54. Y. Bernet, P. Ford, Y. Yavatkar, F. Baker, L. Zhang, M. Speer, R.Braden, B. Davie, J. Wroclawski, E. Failsteine, "Integrated Services over DiffServ Networks", RFC 2998, November 2000
55. Danny Raz, Yuval Shavitt, "Optimal Partition of QoS Requirements with Discrete Cost", IEEE Journal on Selected Areas in Communications, Vol. 18, NO. 12, December 2000.
56. G. Apostlopoulos, D. Williams, S. Kamat, R. Guerin, A. Orda, T. Przygienda, "QoS Routing Mechanisms and OSPF Extensions", RFC 2676, August 1999.
57. The VINT project, "The *ns* Manual", Kevin Fall (editor), Kannan Varadhan (editor), February 2002.
58. Joel Conover, "Policy-Based Network Management", November 29,1999, http://www.networkcomputing.com/1024/1024f1.html
59. Marcus Goncalves, "Directory-enabled Networks", ISBN: 0071349510