대규모 이기종 IP 망의 통합품질관리 시스템의 설계 및 구현

최 태 상[†]·정 형 석[†]·최 희 숙[†]·김 창 훈[†]·정 태 수[††]

요 약

인터넷은 더 이상 특정그룹만을 위한 네트워크가 아닌 일상적인 삶에서 이루어지는 광범위한 통신수단이 되었다. 사람들은 이메일을 사용해서 사적인 메시지를 교환하고 학생들은 웹을 통해서 교육적인 도움을 받으며, 사이버 쇼핑몰을 통해 다양한 상품을 구입하기도 한다. 또한 여러 기업들은 인터넷 상에서의 업무를 실시하고 있다. 최근 들어, 이 같은 인터넷 트래픽의 폭발적인 성장은 트래픽의 양, 트래픽의 특성, 다양한 서비스 성능의 필수 요건 등의 관점에서 계속적으로 변화하는 사용자들의 요구를 수용하는 방법에 큰 관심을 올렸다. 망 자원의 Over provisioning방식은 단순한 해결책이 될 수 있지만 너무 비싸고 비효율적인 단점을 가진다. 따라서, 이 당면한 난제를 해결하기위해 많은 신기술이 최근 소개되고 있다. 그러나 어떤 특정 한가지의 해결책은 불충분하므로 다양한 해결방안을 신중하게 고려한 통합 솔루션이 요구되어진다. 본 논문에서는 이 같은 문제에 대한 해결책으로써 policy기반 인터넷 QoS provisioning, 트래픽 엔지니어링과 성능관리 시스템을 통합한 해결책을 제안한다. 본 해결책은 신속하고 자동화된 품질 서비스 provisioning 기능, 실제 망의 성능 정보들에 기반한 보다 현실적인 서비스와 망 자원의 정책 제어 기능 및 이기종의 다양한 IP 망의 중앙 집중식 트래픽 엔지니어링 기능을 제공한다.

An Integrated QoS Management System for Large-Scale Heterogeneous IP Networks : Design and Prototype Implementation

Tae-Sang Choi[†]·Hyung-Seok Chung[†]·Hee-Sook Choi[†]·
Chang-Hoon Kim[†]·Tae-Soo Jeong[††]

ABSTRACT

Internet is no longer a network for special communities but became a global means of communication infrastructure for everyday life. People are exchanging their personal messages using e-mails, students are getting their educational aids through the web, people are buying a variety of goods from cyber shopping malls, and companies are conducting their businesses over the Internet. Recently, such an explosive growth of the traffic in the Internet raised a big concern on how to accommodate ever-changing user's needs in terms of an amount of the traffic, characteristics of the traffic, and various service quality requirements. Over provisioning can be a simple solution but it is too expensive and inefficient. Thus many new technologies to solve this very difficult puzzle have been introduced recently. Any single solution, however, can be insufficient and a carefully designed architecture, which integrates a group of solutions, is required. In this paper, we propose a policy-based Internet QoS provisioning, traffic engineering and performance management system as our solution to this problem. Our integrated management QoS solution can provide highly responsive flow-through service provisioning, more realistic service and resource policy control based on the real network performance information, and centralized control of traffic engineering for heterogeneous networks.

※ The work described in this paper is supported by the Ministry of Information and Communication of Korea.
† 정 회 원 : 한국전자통신연구원 인터넷구조팀
†† 정 회 원 : 한국전자통신연구원 책임연구원
논문접수 : 2000년 10월 30일, 심사완료 : 2000년 12월 29일

1. Introduction

The Internet is growing with enormous speed and is becoming an integral part of everyday business operation. It is undoubtedly becoming the platform of a choice for the emerging innovative network and application services. Recently, however, such an explosive growth of the traffic in the Internet raised a big concern on how to accommodate ever changing user's needs in terms of an amount and characteristics of the traffic, and a wide variety of user's service quality requirements. Over-provisioning can be a simple solution but it is too expensive, time-consuming, and inefficient. Thus many new technologies to solve this very difficult puzzle have been introduced in the past few years.

Integrated Services (IS) was proposed by the Internet Engineering Task Force (IETF) in the mid 1990s to address the end-to-end Quality of Service (QoS) [1]. It utilizes ReSerVation Protocol (RSVP) [2] as a signaling protocol to setup per-flow based resources dynamically. Its intention was to provide the closest way of circuit emulation on IP networks. However, due to the complexities involved to process and maintain flow states, it gives big burdens to both hosts and network elements thus makes hardly scalable in the large-scale IP networks.

Differentiated Services (DiffServ) [3] was introduced as an alternative to provide a simple and coarse method of classifying different user's service quality requirements and thus can be scalable for the large networks. This objective is achieved by pushing most complexity to the edges of the network to keep the network core free from per-flow state processing and maintenance. Two Per Hop Behaviors (PHBs) were standardized by the IETF for traffic classifications. Expedited Forwarding (EF) PHB minimizes delay and jitter and provides the highest level of aggregate QoS, whereas Assured Forwarding (AF) PHB defines four classes and three priorities within each class. The former strictly maintains its service quality by dropping the excessive traffics of the promised profile but the latter may allow demotion of the quality.

IETF yet initiated another effort to define QoS technology called Muti-Protocol Label Switching (MPLS) [4] to provide better QoS for IP networks in terms of the scalability and forwarding speed. Its key concept is to separate IP forwarding and control functions. Its scalability comes from DiffServ like traffic classifications, as it also marks traffic at ingress edge of the network, and un-marks at egress edge points of the network. It also achieves traffic forwarding speed enhancement by making switching decisions of data packets in the core based on 20-bit labels assigned at the ingress edge point. But unlike DiffServ, MPLS can also setup explicit routes based on user's service quality requirements. This explicit routing feature of MPLS makes for Internet Service Providers (ISPs) to engineer their networks based on the QoS requirements in much efficient manner, which allows many of the difficulties associated with current IP routing schemes to be addressed.

Although some layer 2 technologies such as ATM have already been QoS-enabled, other more common ones like Ethernet were not originally designed for this purpose. In order to provide end-to-end QoS, the IEEE defines 802.1p, 802.1Q, and 802.1D standards [5] to classify layer 2 frames for fast delivery of time-critical traffic. IETF is also standardizing the method of mapping between upper-layer QoS protocols and services with those of layer 2 technologies. Subnet Bandwidth Manager (SBM) [6] was defined for one of these purposes for the mapping between shared and switched LANs QoS requirements with high layers. It is a signaling protocol between hosts and the above-mentioned LAN switches.

Besides QoS-enabling technologies mentioned above, there are many more essential component technologies which were not explained such as various traffic conditioning algorithms (e.g., policing, shaping, and queue scheduling algorithms), traffic classification, and congestion control ones. IETF continues to define a wide variety of QoS-enabling protocols and architectures such as Common Open Protocol Service (COPS) [7], Policy-based Network Management (PBNM) [8], and

Directory Enabled Networking (DEN) [9].

Each or some combination of these technologies can provide partial solutions to specific purposes. For examples, integrated service architecture can be used to provide QoS services in small-scale IP networks such as an enterprise environment due to its scalability problem. Differentiated service architecture and MPLS are better suited for a large-scale backbone IP network. PBNM provides mainly a configuration management solution either in enterprise and backbone networks in the scope of single administrative domain. Any single solution, thus, can be insufficient for an end-to-end QoS management. Carefully designed integrated management architecture is required to provide true end-to-end IP QoS management.

In this paper, we propose a policy-based Internet QoS provisioning, traffic engineering and performance management system for this purpose. Our system consists of a GUI server, a policy server, a routing advisor for traffic engineering (RATE), a QoS performance manager, and a common resource information repository. The GUI server interacts with the rest of the servers to control and retrieve information necessary to present to the network manager. The policy server provides policy rule management functionality such as creating, editing, deleting, and conflict checking of QoS and routing policies specific to a QoS policy administration domain such as a diffserv domain and MPLS domain. Enforcement of the defined rules into appropriate network elements is another work. It also plays a role of inter-domain policy broker for SLA provisioning. RATE monitors routing behavior of the target domain and makes optimal decisions for traffic engineering point of view. The QoS performance manager synchronously and asynchronously monitors the status of the QoS provisioned networks to help the policy server and RATE make optimal policy and traffic engineering decisions. It also provides physical network topology information. The common resource information repository consists of a relational DB and LDAP directory to store the defined policy rules and global/local management information.

The main characteristic of our system is an integrated QoS management of policy control, centralized traffic engineering, and real-time resource monitoring. Our integrated management QoS solution can provide highly responsive flow-through service provisioning, more realistic service and resource policy control based on the real network performance information, centralized control of traffic engineering for heterogeneous networks, and a user-friendly and intuitive graphical user interface.

The rest of the paper is organized as follows. Section 2 addresses some of the major related research and development work. Section 3 describes user's and service provider's requirements and some of the important design issues. Section 4 describes our system architecture in detail. Section 5 describes the prototype implementation of our system. Finally, Section 6 summarizes our work and discusses directions for our future research.

2. Related Work

There are a number of related academic research and commercial development efforts in the Internet community. In this section, we describe the architectures and major characteristics of these efforts and examine some of the important lessons we have learned.

Routing and Traffic Engineering Server (RATES) [10] was developed for MPLS traffic engineering. The RATES implementation consists of a policy and flow database, a browser-based interface for policy definition and entering resource provisioning requests, and a COPS server-client implementation for communicating paths and resource information to edge routers. RATES also uses the OSPF topology database for dynamically obtaining link state information. RATES can set up bandwidth-guaranteed label-switched paths (LSPs) between specified ingress-egress pairs. The path selection for LSPs is on a new minimum-interference routing algorithm [11] aimed at making the best use of network infrastructure in an online environment where LSP requests arrive one by one with no a priori infor-

mation about future requests. It uses a relational database for the various resource information to reduce performance burden, which occurs due to frequent data updates. Its main scope is intra-domain MPLS network QoS traffic engineering and thus is limited to provide end-to-end QoS management in heterogeneous IP networks. Our system can perform intra-domain QoS management functionality such as provisioning, performance monitoring, and traffic engineering for both MPLS and traditional IP networks. Furthermore, inter-domain SLA signaling and its associated intra-domain resource provisioning capabilities are supported.

The Internet2 Bandwidth Broker (BB) Advisory Council (BBAC) [12] employs the BB to manage network resources for IP QoS services. It aims to automate the admission control decisions and network device configuration functionality in the context of the Internet QBone architecture [13]. A BB can be a type of policy server and manages the QoS resources within a given domain based on the Service Level Descriptions. BB also gathers and monitors the state of QoS resources within and at the edges of its domain. Two signaling mechanisms are proposed to set up the inter-domain communication for resource allocation. BB will support the best-effort and DiffServ Premium services. Its API allows applications to make BB service requests and to query for existing service commitments. Although the intra-domain QoS resource management is mentioned in the architecture, the specific implementation architecture is completely left upto implementers. Our system completes this missing block by designing the integration architecture to tie the inter-domain signaling functionality and intra-domain resource provisioning and performance monitoring functionality.

QoS agent architecture [14] provides a QoS management solution for a large network. The main thought behind the architecture is that every *Routing Domain* (RD) will have at least one QoS agent that manages admission control and reservations. In each DiffServ domain there can be more than one RD. The QoS agent can handle immediate-, future- and third party reservations. The QoS agent obtains topology information collected from the routing tables by listening to OSPF messages. It also monitors the network links physical bandwidths via SNMP [15]. If a user or application needs a better service quality, the agent has to be contacted to request the new level of service for a flow of data. When the QoS agent receives the request it knows which way the flow will take through the RD and if the links along the path can handle the requested load. If the network is able to handle the extra load, the agent sets up the reservation. QoS agents communicate with each other to handle admission control for reservations spanning more than one RD. This solution still limits its management scope into a single DiffServ domain although the scale of the domain is a large backbone network. It monitors routing topology based on OSPF information and network resource status via SNMP but flow-based monitoring is not supported. Flow-based monitoring is essential for SLA monitoring and billing. Our system is capable of measuring a flow whether it is a DiffServ flow or an MPLS LSP. Measurement is done non-intrusively using SNMP, Diff Serv MIB [16], and MPLS Traffic Engineering MIB [17].

The Distributed Resource Controller (DRC) technology [18] provides a novel approach to interfacing applications with emerging network mechanisms to deliver Quality of Service (QoS) and controlling network resource utilization. DRC aims to unify network services (e.g., Diffserv, Intserv, and ATM) and application QoS provisioning by introducing a middleware system and a set of generic interfaces. DRC middleware is the core part that translates application requests for QoS delivery into respective access to underlying network systems in a manner that optimizes resource utilization and shelters applications from such a complexity. Applications can request QoS to the DRC middleware in two ways : either in-line using the DRC QoS API or off-line via the DRC utility. Application QoS requirements are specified in terms of two categories of parameters : Traffic Profile (quantitative) and User Expectation (qualitative). DRC currently uses the CORBA-based [19] object-oriented technology to develop a CORBA-

based Resource Controller (CRC). Although DRC includes DiffServ management in its architecture, the working prototype manages the Integrated Services only. Its main interest is in QoS management from an application's point of view. Our system is much more than a QoS signaling broker and is designed to be a part of a service provider's operation support system (OSS) to manage next generation IP networks.

Besides the above mentioned research efforts, a dozen of commercial policy server products are available. IPHighway, Ochestream, HP, Intel, Nortel, and Cisco, to name a few, are the major vendors, which are pioneering in this technically emerging field. It is hard to describe all the details of their product's features but a few common properties can be at least mentioned. First of all, most of them are focused to solve QoS management problems within an enterprise network scope. In the TMN [20] hierarchy, their solutions mainly belong to the element management layer. Also, in terms of management functionality point of view, they provide configuration management capability with very few fault and performance coverage. However, end-to-end QoS management for a large-scale IP networks requires network and service management capabilities including intra-domain service and network provisioning, provisioned service quality performance management, network traffic engineering functionality, service and network fault management, inter-domain SLA provisioning and monitoring, and flow-based billing.

In the next section, we summarize important requirements and design considerations for our system architecture to provide optimal end-to-end QoS management.

3. Requirements and Architectural Design Decisions

We have reviewed pros and cons of some of the important research and commercial development efforts to address various QoS management problems in the previous section. Although each of them claims that its solution provides an end-to-end QoS management, most approaches tackle a limited scope of QoS management. In this section, we describe user and service provider requirements and design decisions to come up with our system architecture to provide end-to-end QoS management.

3.1 Requirements

The requirements for end-to-end QoS management in the Internet come from mainly two different sources : service users and service providers. As new types of applications keep emerging and their traffic types are also changing, best-effort network and service architecture are not sufficient to accommodate such different types of traffics. Users want to specify their desired classes of services according to the types of applications they use. And network providers need to meet these users requirements for the success of their business. So far, there were no efficient means of traffic engineering within user's and service provider's IP network domains respectively, let alone, automated interfaces for service level agreements between them.

To provide true top-to-bottom and end-to-end QoS guarantee, a very well structured and designed architecture from a service management level to a network element management level is required. We explain briefly what is needed for this from user's and service provider's perspectives.

- User Requirements

We define a user in this paper as both an enterprise customer and a service provider that plays a customer role for inter-domain business relationship. Supporting QoS for end-users such as home dial-up users are not technically feasible yet and requires further research and development. User requirements can be summarized as easy service creation and customization both in intra-domain and inter-domain networks, automated trouble administration, and automated performance reporting. The following lists some of the important detailed requirements descriptions from the user's perspective.

1. Users need an easy-to-use and secure user interface to specify and manage what they want for their

applications and services in terms of QoS requirements.

2. This interface should include service and network topologies with managed network resource utilization status. Also users should be able to add, modify, remove new, part of, or entire services through this interface.
3. Users want to subscribe a service via an automated interface throughout the entire service provisioning transaction in a flow-through fashion. They do not want to wait for several days or even several hours before they can use a requested service.
4. When the requested service is activated, they want to monitor or receive contracted level of service performance. Preferably via an automated interface like a web browser.
5. When a fault occurs, users want to report the problem via the same user interface and check the status of the problem resolution processes.
6. Users also need to manage their own networks to support end-to-end QoS guarantee. This includes user-friendly policy definition, policy enforcement, provisioned service utilization monitoring, etc.

• Service Provider Requirements :

A service provider includes an Internet Service Provider (ISP), a Network Service Provider (NSP), and an application service provider (ASP) that provide not only network access services but other value-added services such as VPN and VoIP. The main objective of management automation is to make their customers happy, enhance network resource utilization, and gain the market leadership from competitions. Especially when it comes to a large-scale IP network QoS management issues that haven't been explored much, it is not an easy task at all. It has to take care intra-domain traffic engineering problems efficiently and very timely manner and inter-domain service level agreements to provide end-to-end QoS guarantee. In summary, the service provider requirements are technology independent management, ease of service creation and customization, efficient traffic engineering, scalability, and rated-based billing. Some of the important detailed requirements descriptions are as follows :

1. For QoS service and network provisioning, a service provider should be able to define policies that are easy to describe and are independent of the underlying network technologies.
2. A service provider should be able to store, modify, delete, resolve conflicts of policies and these transactions should be performed in secure manner.
3. A service provider should be able to provision requested services in their intra-domain network using policies defined as above with automation tools. Again such very important and sensitive information should be well authorized and securely transmitted.
4. A service provider should be able to monitor the resource utilization and be able to make appropriate allocation decisions when resource provisioning requests are received. This issue is particularly hard in current IP networks. Traditional IP networks were not built to support cost-effective traffic engineering mechanisms. For instance, enforcing QoS policies in a DiffServ domain requires the advance knowledge of the network to find appropriate policy target network elements such as a best route given a source and a destination pair and link bandwidth availability, etc. Also when the provisioned route are modified for some reason during the service, the related QoS policies have to be updated to appropriate network elements in timely manner. If the underlying network has embedded traffic engineering functionality such as MPLS, the problem can be eased somewhat but the OSS intervention is still required.
5. A service provider should be able to make best possible routing decisions depending on the underlying link layer technology (e.g, Ethernet, ATM, MPLS, or WDM). These decisions should be tightly coupled with monitored network utilization status and routing policies based on a global network topology.
6. A service provider should be able to support scalability in terms of networks and services.
7. A service provider should be able to report the con-

tracted service level in easy to understand forms to their customers using automated tools in timely manner.

8. A service provider should be able to interact with other service providers for admission control, service provisioning, resource monitoring, and rated-based billing on behalf of their customers over secure links.

3.2 Design Decisions

In order to satisfy the requirements mentioned above, we explain our design decisions in this section.

• Management Scope

To support end-to-end QoS guarantee, partial QoS management solutions described in Section 2 are not sufficient. As stated in our requirement section, the objective of our system is to provide an end-to-end QoS management solution. To accomplish this goal, the management scope of our system spans multiple IP network domains including enterprise customer networks and service provider access and core networks. Since the network transmission and switching technologies are rapidly evolving, we would like to accommodate any of them in our design scope. From the management point of view, the underlying network technology should be transparent. In each domain one or more instance of our system manages its own network domain and one leader instance interact with others for inter-domain issues such as admission control, resource allocation, performance reporting, and billing.

• Management Architectural Model and Scalability

Typically, a management system architectural model can be a centralized, a hierarchical, or distributed. Each model has its pros and cons depending on the management scope, performance, reliability, scalability, cost, and interoperability. Our system consists of three distinct servers, one for policy management, the other for traffic engineering including but not limited to routing decision and control, and another for QoS resource monitoring. In our system design, we adopted an integrated approach. For intra-domain management, a centralized model is used. This model is typically used for the small and medium scale management solution. It is easy to implement and has centralized management information that can be shared by different servers easily in our case. However, for the scalability within a domain when needed, each server is designed in an object-oriented way such that more than one instance can be generated depending on the number of managed elements. For inter-domain management, a distributed model is used. Only policy servers interact one another among different domains through a CORBA interfaces. This choice is quite natural because inter-domain issues mainly related with SLA management, which requires transaction-oriented, reliable, secure, location independent, and scalable interface solution. Also the distributed model is used between servers within a system via CORBA interfaces and sharing a directory service and a relational database.

• SLA Management

It is essential to provide SLA management capability to guarantee end-to-end QoS across a wide variety network domains. Among many things for true SLA management, two important aspects are considered, namely, SLA provisioning and SLA monitoring. In our system design, SLA provisioning includes QoS admission control and resource allocation between a customer and service providers and between service providers on behalf of the customer. SLA monitoring is responsible for network-level availability and response time only. There are numerous other aspects to be considered such as application-level SLA management, trouble administration, performance reporting, and charging arbitration in relation with service level agreement violation, etc. But those issues themselves deserve to be researched separately. As mentioned above, CORBA interfaces are used for admission control and resource allocation negotiations. Negotiated SLAs are translated into intra-domain QoS policies and the intra-domain policy server separately does its enforcement. Also network resource utilization information obtained by the resource servers from each domain is aggregated and analyzed to represent service level performance metrics. We also include

the software-based traffic generator and analyzer in our system design to support active SLA performance measurement. It can generate self-similar traffic patterns to simulate the real Internet traffic and can be injected into particular network flows.

• Policy-based Management

Policy is used in our system design as specified in policy framework document series defined by IETF [21]. It is a set of information that conveys business decisions for a particular service to be installed in the network(s). It is very important to have an abstract, common, and platform independent information model to represent service policies. Currently, IETF is defining DiffServ-based QoS [22] and security policy information models [23] with the above characteristics. Also policy-based network management (PBNM) framework is defined to configure policies into target network or system elements through automated tools. Our system adopts their framework and extends to add routing policies. PBNM is designed for the element configuration management purpose but it can be used for a network configuration management when it is combined with a global network topology tool. It is our design purpose to expand the scope of PBNM into an intra-domain network configuration management. Also implementable part of SLA, known as Service Level Specification (SLS), needs to be mapped into network policies. This mapping functionality is included in our system design. Besides, we add additional functions for inter-domain SLA admission and resource allocation negotiation similar to a main function of bandwidth brokers defined by Internet2 Qbone BB Advisory Council.

• Monitoring and Topology Discovery

QoS provisioning and monitoring go side by side. After a requested QoS is provisioned, it is essential to monitor the performance of it. Monitored results can be fed back to the route server for optimal routing decision if necessary, to the SLA performance reporting module to measure the contracted service level, and to administrator's GUI for user-friendly view. Unlike typical passive network monitoring, QoS monitoring requires flow-based measurements. Both a micro flow and an aggregated flow measurement should be supported depending on the situation. Typically the former method can be used in a small-scale enterprise network where micro flow classification is possible. Core network mostly depends on the latter. The main issue, though, is how to measure an end-to-end flow. Especially when congestion occurs in a core network, it is very hard to pinpoint the possible cause of the problem. In our system design, thus, we utilize the combination of the information rather than using only a flow measurement information gathered by the resource monitoring server. For this, DiffServ MIB currently under work in IETF DiffServ WG is used to measure flow-based utilization information in the core and other flow measurements collected at the ingress edge points with the help of BGP route reflection mechanism. Also the resource monitoring server constructs intra-domain global topology information based on MIB-II.

• Routing Decision and Control

One of our major system requirements is efficient use of network bandwidth resources to support QoS management. Current IP routing schemes have significant shortcomings by utilizing bandwidth resources based on the routing decision mechanism which depends only on a shortest path computed by a destination address and mostly static and traffic characteristics independent link metrics. To overcome this obstacle, we use an intelligent routing decision and control mechanism by closely working with other system modules, namely, the resource monitoring server and the policy server. In this mechanism, optimal routing decision is made by three types of information. The first is a routing policy decision to make either by the policy server or human manager via a GUI. The second is the global intra-domain network topology information, OSPF [24] based IP layer routing behavior information, and the network resource utilization information collected by the resource monitoring server. The third is the BGP [25] routing behavior information obtained by the BGP route reflector, which is one of a module in the RATE [26]. This decision can be translated into routing policy

information and fed back to the policy server to enforce them to the appropriate target elements. Further comment is worth to mention at this point. When the underlying network is based on conventional router based and OSFP protocols only, it is realistically not easy to enforce the routing policy to gain the desired performance of routing behavior. However, if ATM, MPLS, or WDM based network is supported underneath, an explicit route control is possible for better support of QoS requirements. Our initial design objective is to report the optimal routing decision and present it through the graphical user interface for the monitoring purpose. Explicit routing control with the policy server will be applied to those networks, which supports such explicit path signaling mechanisms [27, 28].

- Repository

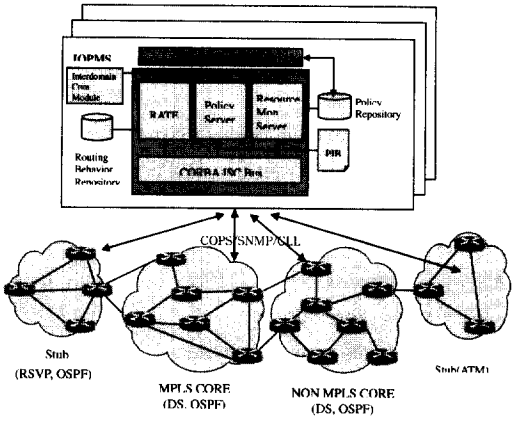
As mentioned in the requirement section, a repository is necessary to store various types of information such as QoS policy, routing policy, security policy, topology data, traffic data, SLA information, network element dependent data for QoS configuration, network element capability data, calculated routing decision data, and etc.. Some of them are static and others are dynamic in nature. PBNM recommends LDAPv3 [29] for storing and managing policy information and we follow this standard track for interoperability purpose. Also some of the information mentioned above requires frequent large volume data updates and, thus, performance is a major decision factor. For this reason, we use a relational database for information with performance constraints. Detailed schema of the information will be described in the implementation section later.

- Security

QoS control automation over IP networks will have great impacts in economics of the next generation Internet. But it doesn't come for free. Other than what was mentioned above, one of the most important issues should be dealt with is the security. Automation typically accompanies with vulnerability with threats. There are several points to be considered to make the system secure. First of all, policy data that contains critical business information should be securely transmitted between a policy server and a policy target element or between policy servers. Repository is also a single point of security threats that should be well protected. But, fortunately, protocols used for QoS management (e.g., COPS) are designed with security features and off-the-self security solutions can be used for most cases. We do not consider this issue further in our architectural design decisions due to the above reasons.

4. System Architecture

We identified the major system requirements and explained design considerations with our decisions. In this section, we propose our system architecture based on the requirements and design decisions. IQPMS consists of an integrated GUI, three servers, a repository, and a policy target including policy target proxy. (Figure 1) shows overall system architecture and its inter-component relationships, and its interaction relationships with a target network infrastructure. This architecture is based on three-tier model : GUI, servers, and agents. Integrated GUI separates GUI-related functionality from servers so that each server can concentrate on its own functions. Servers perform its own functions and also interact with other servers in distributed manner to help each other to achieve overall system objectives. We describe a brief overview of each



(Figure 1) High-level System Architecture of IQPMS

component, its internal modules, and flow relationships among modules. Communication between the servers is achieved through CORBA-based inter-server communication (ISC) bus.

4.1 GUI

A main design philosophy of our GUI is to control, monitor, and present a global view of the QoS services within an intra-domain. GUI has a two-tier architecture : GUI client and GUI server. This architecture provides platform independence, ease of customization, and scalability. GUI client can be either an independent application or applet embedded in the web page. GUI server runs in a background and communicates with other server modules and repository servers via various protocols such as LDAP, SQL, and CORBA. It exports XML-based [30] menu to clients so that dynamic menu update is possible. A human manager can watch its global network topology, resource utilization, routing configuration, and even routing behavior, which are auto-discovered by resource monitoring servers and routing advisors (RATE : Routing Advisor for Traffic Engineering). And policies for a desired service, for instance, VPN service can be defined by using a policy editor based on the view of topology and behavior. When the defined service policies are scheduled and enforced, the topological view of routing behavior and link and flow utilization status can be monitored. Major features supported are as follows :

- Service and Network level topology view presentation
- Network level routing topology visualization ; BGP peering relations, configured OSPF interfaces and metrics etc.
- Network level routing behavior visualization
- Flow specific (DiffServ, MPLS LSP, VPN flow, etc.) view presentation
- Policy domain view presentation
- Policy editor (including QoS policy, security policy, and routing policy)
- Various domain (DiffServ, MPLS, OSPF area and subgroup, etc.) editor
- Enforced service policy viewer and editor
- Managed network elements configuration and capability information viewer
- Other system administration functionality such as server status monitoring, user profile management, resource database initialization and management, etc.

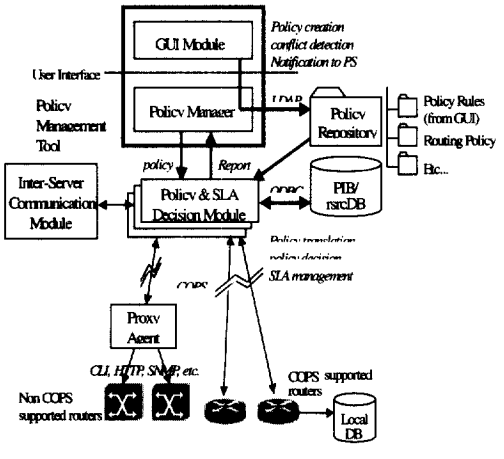
4.2 Policy Server

As explained in the design decision section on policy-based network management, our system adopts IETF's PBNM framework and extends to add routing and security policies. PBNM is designed for the element configuration management purpose. We expand the scope of PBNM into an intra-domain service and network configuration management. Besides, we add additional functions for inter-domain SLA admission and resource allocation negotiation similar to a main function of bandwidth brokers defined by Internet2Qbone BB Advisory Council. It consists of policy manager, policy decision module, inter-server communication module, policy agent and proxy agent, policy repository, and PIB database. Policy manager receives user inputs through the GUI and checks any policy conflicts with the existing ones before it stores into policy repository with the help of policy decision module. Policy decision module retrieves stored policies and translates them into network dependent policies in PIB format and sends them into appropriate policy targets via COPS protocol. Inter-server communication module is used for SLA negotiation and inter-server communications with other server modules. Proxy agent module talks with non COPS-supported routers via any other protocols such as CLI, HTTP, and HTTP.

Architecture of our policy server is shown in (Figure 2) The following lists major functionality of the server :

- Intra-domain QoS, security, and routing policy creation, conflict detection, and storage to LDAP server.

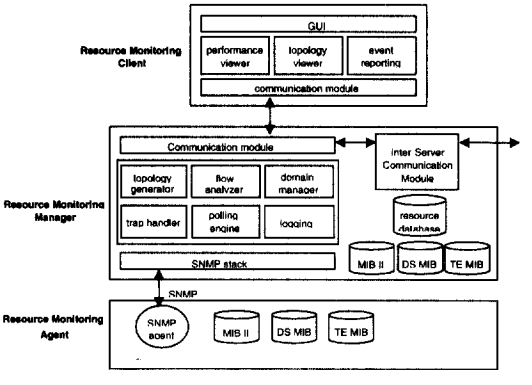
- Policy retrieval from LDAP server, decision of appropriate network or system elements to enforce policy, and translation into Policy Information Base (PIB).
- Enforce the translated policies into the appropriate target elements directly to COPS-enabled elements or indirectly via a proxy to legacy target elements.
- SLA admission control, resource allocation, and SLA performance reporting with other domains
- SLA mapping into domain specific policy rules.
- Interaction with the routing server or the resource monitoring server to help making a routing decision and to retrieve resource utilization information through inter-server communication module.



(Figure 2) Architecture of the Policy Server

4.3 Resource Monitoring Server

Resource can mean many different things. In our system, it is mainly defined for QoS management purposes such as network link capacity, processing power of routing, switching, and transmission equipment, and service node system processing power (e.g., Remote Access Server (RAS), VoIP Gateway, and Gatekeeper). The resource monitoring server's main functionality is to monitor and analyze utilization of these resources. It has four distinct components, as depicted in (Figure 3) : a resource monitoring client, a resource monitoring manager, a resource monitoring agent, and inter-server communication module. The resource monitoring manager further consists of topology generator, traffic collector, traffic analyzer, domain specific flow analyzer, and SNMP manager/polling engine. It is a centralized manager used to manage a set of DiffServ and MPLS routers and to provide management interfaces to a set of resource monitoring clients. Its main role is to perform domain and flow management and flow-based performance measurement. The resource monitoring client is a user interface to control domain and flows to manage and to present the analyzed performance results received from the resource monitoring server. The resource monitoring agent resides in the network element to collect flow-based performance metric via various MIBs. Inter-server communication module is used to interact with other servers in the system.



(Figure 3) Architecture of the Resource Monitoring Server

(Figure 3) shows the architecture of the resource monitoring server. Its main features are described below :

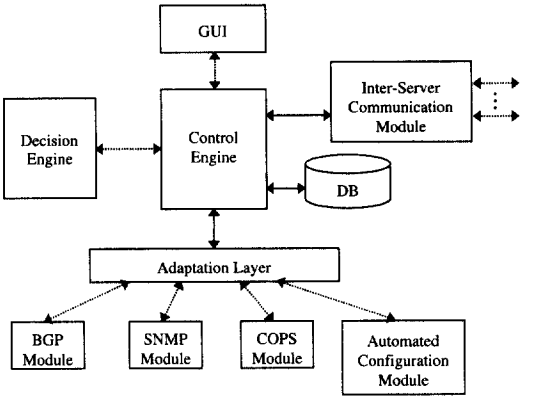
- Intra-domain network topology (layer 3 physical view) construction.
- Domain creation : DiffServ or MPLS domain creation. Network manager who knows its own managed network can group a set of network elements into a particular domain to meet a specific requirement. During the creation, the network manager assigns the role of each network elements within

the boundary of the domain. For example, edge ingress/egress of DiffServ domain or core Level Switch Router (LSR) of MPLS domain, etc.

- Flow creation : flow(s) can be created when a particular service policy is enforced, for example, VPN setup request is received from a policy server. Depends on the type of underlying networks, the flow can be DiffServ flow or MPLS LSP. Flows are very important unit of management in this system because it is the basic unit of performance measurement and billing. The created flow information is stored in a flow table in a global database so that other servers can share it.
- Network performance data collection and analysis such as end-to-end flow utilization, packet drop rates and the number of drops on a DiffServ aggregated flow or an MPLS LSP. Resource monitoring manager collects raw performance data from DiffServ MIB, MPLS traffic engineering MIB, and MIB II via SNMP protocol and summarize all the data to provide the aggregated performance metrics. Active measurement method is going to be used to provide true end-to-end performance metrics for flows as well.
- Network elements fault detection.
- Translation of performance data into a common form to share with other servers such as the routing server and the policy server.

4.4 Routing Advisor for Traffic Engineering (RATE)

Automated QoS policy provisioning and monitoring is very important to provide QoS support. Yet, they are not sufficient to support an end-to-end QoS guarantee. Traffic engineering based on the optimal routing decisions in the network infrastructure is the essential functionality to achieve full-featured end-to-end QoS guarantee. Thus, three servers should work in harmony for this ultimate goal. RATE's major functions are identifying routing topology, monitoring routing behavior, and issuing advanced routing decisions for traffic engineering in an intra-domain network especially at the core with lots of composite transit traffics. RATE consists of GUI, control & decision engine, adaptation layer, various communication modules, and inter-server communication module. Like other two servers, this server is based-on three-tier architecture. The detailed functionalities of this server are described below.



(Figure 4) Architecture of RATE

(Figure 4) illustrates the architecture of RATE. Its main features are listed as follows

- Gathering lots of routing information via SNMP and BGP peering ; RATE takes participate in the BGP operation as a BGP route reflector [31] itself. This method enables RATE to monitor and control inter-domain routing behavior more efficiently and seamlessly, since RATE can filter and modify BGP route updates directly, and even issue some new route updates on purpose.
- Creation of integrated routing information DB and its management. It performs synthesis and analysis of the raw routing data collected via various means, such as BGP peering and MIB variable fetching from OSPF MIB, BGP MIB, and MIB-II routing data, etc., to achieve optimally engineered traffic condition.
- Detection of any routing behavior changes and notifies to other servers to act on the updates.
- Explicit routing path selection. When the underlying network supports explicit route signaling capability such as MPLS, RATE can find the op-

timal path based on the given user's QoS requirements and current routing behavior which are collected by a resource monitoring server and RATE itself.

- Load balancing for equilibrium. RATE detects the congestion status in the target network and performs necessary controls such as changing the routing metrics, aggregation or segregation of inter-domain route updates, and so forth.
- Suggesting optimal routing metrics by massive simulations. Off-line simulation capability which is equipped in the Decision Engine module of RATE can provide several simulation results which can be used for long-term traffic engineering.

4.5 LDAP Directory Server and Relational Database System

As mentioned in the requirements and the system design decision sections, we use two types of information repository systems : a LDAPv3 directory server and a relational database system. Both of them are critical system components to provide platform independent, scalable, user-friendly, efficient, and interoperable QoS management functionality within and across (an) administration IP network domain(s). Large portion of server communications is performed over this common repository. LDAP directory stores fairly static information such as policy rules. Any other information that requires frequent updates are stored in a relational database.

We defined a number of tables to be shared among servers. Some of them are global and others are server specific tables. We categorized six groups to represent the all the managed information : router, interface, routing, flow, layer3 link and domain. Each group has global and server specific tables. There are 5 global tables : GlobalRouterTable, GlobalInterfaceTable, GlobalRoutingTable, GlobalFlowTable, and Layer3LinkTable. Resource monitoring server and RATE fill in the contents periodically. Router group has server specific tables : RMSRouterTable, OSPFRouterTable, BGPRouterTable, and GUIRouterTable. Interface group's server specific tables are OSPFInterfaceTable, DiffServInterfaceTable, and PSInterfaceTable. Flow group has DiffServFlowTable and MPLSFlowTable. Finally, domain group has DiffServDomainTable, PolicyDomainTable, and OSPFSubGroupTable. (Figure 5) shows GlobalRoutingTable as an example. It contains information collected from RMS and RATE and main user of this table is GUI to present router information to the human

Field Name	Field Type	Field Constraint	Field Description
routerId。	serial	Pk (Primary Key), auto-increment	globally unique id of this router
snmpVersion	varchar(5)		supported SNMP version
nofIntf	int2		number of interfaces that this router has
routingProtocol	int2		OSPF, BGP, etc. (predefined numerical value)
community	varchar(30)		SNMP community
controlPortIpAddress	varchar(15)	Unique, NOT NULL	IP address of control port
controlPortName	varchar(50)	Unique	hostname corresponding to the controlPortIpAddress
sysDescr	varchar(100)		fields imported from system MIB
sysObjectid	varchar(50)		
sysUpTime	varchar(50)		in "mmddhhmmss" form
sysContact	varchar(100)		
sysName	varchar(50)	Unique	
sysLocation	varchar(100)		
sysServices	int4		
ospfRouterId	varchar(15)	Unique, NULL	for the sake of OSPF module
bgpRouterId	varchar(15)	Unique, NULL	for the sake of BGP module

(Figure 5) Global Routing Table

network manager.

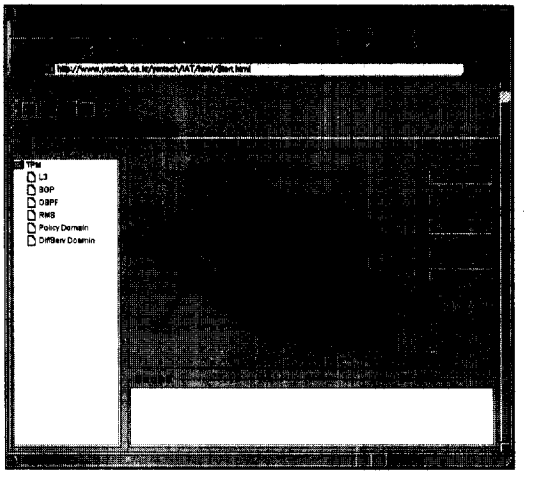
4.6 Inter-Server Communication Module

Information stored in a relational database can be shared among three servers but its not enough for full server interactions. Servers need to exchange messages to perform certain actions that require more than one server involvement. For example, when RATE found a route changes in the target network. It has to notify a Policy Server to modify the enforced policy rules into updated route and remove the rules from the existing route. There are numerous inter-server interactions possible among four servers : GUI server, Policy server, RMS, and RATE. We use CORBA for these interactions and defined IDL interfaces and information objects to be exchanged. Some of the interfaces, for example, are AutoDiscoveryStart(), DiffServDomainCreate(), Flow Create(), DBFillInReq(), RouteChangeNotify(), NewRouterAdded(), etc. Also, for inter-domain SLA signaling, we are defining IDL interfaces and information objects to represent SLA. Active research is still going on this area by several consortiums such as Qbone BB Advisory board and TEQUILA project [32]. They try to come up with standard interfaces and information object to promote interoperability. We try to actively follow these activities to align with standard approach.

5. Prototype Implementation

We are currently implementing our system based on the proposed architecture. We are in the second phase of the development. The prototype completed in the first phase (in 1999) realized a subset of policy server functionality targeted for intra-domain QoS policy provisioning and a resource monitoring server based on a web-based SNMP manager and an agent with DiffServ MIB [33].

GUI has a two-tier architecture : GUI client and GUI server. GUI client can be either a standalone java application or java applet embedded in the web page. GUI server runs in a background and communicates with other server modules and repository servers via various protocols such as LDAP, SQL, and CORBA. It exports XML-based menu to clients so that dynamic menu update is possible. Communication between GUI client and server uses RMI protocol [34]. JDK 1.3 is used to build GUI client and server. It has topology manager which displays L3 physical topology view, BGP & OSFP topology view, DiffServ domain topology view, and Policy domain topology view. Human network managers can use these user-friendly views to check the current status of the network, performance metrics, fault status, etc and to help making right QoS policy decisions. Figure 6 shows the snapshot of our GUI. It shows the layer 3 topology view of our testbed.

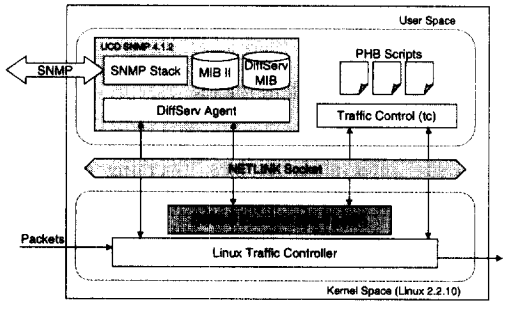


(Figure 6) A Snapshot of GUI

The policy server has been implemented in Linux environment. It consists of three modules. A java-based GUI, a policy management module to create, edit, modify, and store QoS policies, and a policy decision point (PDP) module to retrieve policies from the LDAP directory and translates them into PIB for enforcement to policy target elements. COPS-based policy target has been implemented in Linux and FreeBSD-based PC routers. Since the Cisco7505 and Juniper M5 [35] router did not support COPS agent functionality at the time of our development, we implemented a Linux PC-based proxy policy agent. PDP module talks to target routers via COPS-PR protocol. This proxy agent communicates

with the Cisco router by using expect telnet scripts [36] and controls Cisco's class-based priority queuing. Currently, we are upgrading it to be capable of providing DiffServ service based on its class based weighted fair queuing per ATM VCs.

The resource monitoring server that includes an SNMP manager for the DiffServ MIB has been implemented in Linux environment and we have added an SNMP agent for the DiffServ MIB in each router. A java-based management interface which is a part of our integrated GUI is implemented for delivering various management services to users very conveniently. A Diff Serv agent is an SNMP agent with MIB II and DiffServ MIB running on the Linux DiffServ router. Basically the agent extracts DiffServ parameters from the Linux traffic control kernel. The organization of our Linux DiffServ router implementation is explained in (Figure 7). There are two process spaces in the Linux operating system, the user space and the kernel space. Extending from Linux traffic control framework, the Linux DiffServ implementation resides in the kernel space. In the user space, the DiffServ SNMP agent is implemented, combined with the Linux traffic control program. Communication between the DiffServ agent and the Linux traffic control kernel is effected via NetLink sockets [37]. The NetLink socket is a socket-type bidirectional communication link located between kernel space and user space. It transfers information between them.



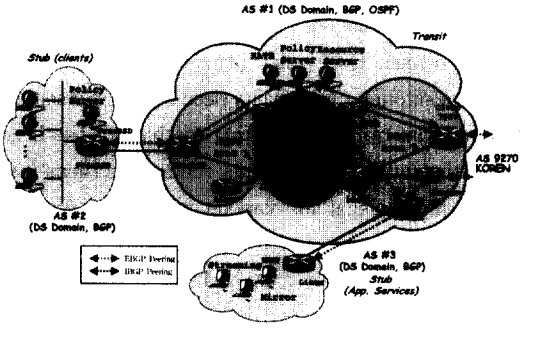
(Figure 7) Organization of Linux DiffServ Router Implementation

The agent has been implemented by using UCD SNMP agent extension package [38]. UCD SNMP 4.1.2 provided the agent development environment. The DiffServ agent uses the traffic control program (tc) or NetLink socket directly for accessing DiffServ parameters in kernel space and manipulates the values of MIB II and DiffServ MIB.

RATE is under development in Linux environment. We will soon finish the functionality of collecting and analyzing routing behavior data via BGP reflector and SNMP. Optimal explicit route path calculation and route control for load balancing functions will be added in the near future.

For policy repository, we are using IBM's Secureway LDAP server [39] which is running in Windows NT4.0. All the common information to be shared between servers is stored in a PostgreSQL database of version 7.0.2 [40].

(Figure 8) shows our testbed configuration. The testbed is built in the lab environment that simulates a real next generation Internet. It is made of four ASs and each represents DiffServ domain except AS#4. AS#1, which is the biggest one, represents a transit core domain, and runs OSPF over IP/ATM within the domain. It has 11 routers : 1 Cisco 7505, 1 Juniper M5, 4 FreeBSD-based routers, and 5 Linux-based routers. There are two stub domains one with a number of client hosts and another with a number of servers to test various applications and services. EBGP and IBGP connections and physical connections between them are illustrated in detail. We are experimenting a number of scenarios. Provisioning of DiffServ flows into our



(Figure 8) Our Testbed Configuration

testbed using our system, monitoring of the provisioned flow performance metrics, monitoring routing behaviors are some of them.

6. Conclusion and Future Work

In this paper, we explained our research and development efforts to support seamless end-to-end QoS management. It described the design requirements, considerations and decisions, and our system architecture. Detailed features of the system components were explained to meet the design requirements. We also described our prototype implementation.

Currently we are at the second phase of research and development. It still requires significant additional work. It currently provides functionality such as intra-domain policy-based QoS service provisioning, DiffServ flow monitoring, routing behavior monitoring and analysis. During prototyping phase, we made some modifications for inter-server communications(ISCs) for fast prototyping. ISCs designed to use CORBA & SQL but the current prototype is built based on socket-based TCP message communications. This module will be eventually replaced by the CORBA-based ISC when the system matures. Also current version's policy server is implemented to handle multiple clients based-on process-forking mechanism. But, in reality, typical ISPs have several-hundred routers and, thus, the number of possible simultaneous connections between the policy server and policy targets can be fairly large. To deal with this issue, we are considering prethreaded policy server implementation.

Inter-domain QoS policy provisioning and performance monitoring functionality need to be implemented. Intra-domain traffic engineering mechanism has been added that utilizes the optimal routing decisions and control mechanisms for various underlying network switching and transmission technology. Those works are underway currently and will be incorporated in the next version of the paper. Integration of MPLS and WDM network management into our implementation is another next big step to follow. It is very important to provide easy-to-use, consistent, integrated, and scalable QoS management solution to next generation IP networks, most likely equipped with optical backbone with intelligent control functionality, MPLS network, and network with various routers ranging from low-end routers to ultra high-speed new next generation routers.

Finally, a performance evaluation of our system is being considered. As the size of the target network becomes bigger, our system has to deal with a huge amount of traffic and should be scalable. Also the control data generated by the system shouldn't affect the underlying target network's performance. We need to conduct network performance evaluation based on both simulations and real network traffic analysis by applying various test traffics for this reason. Our system supposed to handle a large size of routing table and other performance related data. Current prototype collects these information from our testbed and some of the routers have several thousands routing table entries. It takes 5-10 minutes to collect and analyze them mainly due to heavy DB operations. Real ISP routers typically handle more than 60,000 routing table entries, which is more than 10 times of our testbed size. To make our system to scale to accommodate the real ISP networks, we are currently considering testing it with real time DB and trying to enhance algorithms for collection and analysis. These performance evaluation results will be reported in the future version of our paper.

References

[1] IETF "Integrated Services," working group. See http://www.ietf.org/html.charters/intserv-charter html and http://www.ietf.org/ids.by.wg/intserv.html.

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

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang,

W. Weiss, "An Architecture for Differentiated Services," RFC 2475, December 1998.

[4] E. Rosen, A. Viswanathan, R. Callon, "Multiprotocol Label Switching Architecture," April 1999, <draft-ietf-mpls-arch-05.txt>, Work in Progress.

[5] For information on the 802 LAN standards such as 802.1p, 802.1Q and 802.1D, see http://standards.ieee.org/catalog/IEEE802.1.html.

[6] R. Yavatkar, D. Hoffman, Y. Bernet, F. Baker, "SBM (Subnet Bandwidth Manager) : A Protocol for RSVP-based Admission Control over IEEE 802-style networks," May 1999, <draft-ietf-issll-is802-sbm-08.txt>, Work in Progress.

[7] J. Boyle, "The COPS (Common Open Policy Service) Protocol," Internet Draft : draft-ietf-rap-cops-03.txt, 1998.

[8] Moore, B., Ellesson, E., Strassner, J., "Policy Framework Core Information Model," Internet-Draft, <draft-ietf-policy-core-info-model-01.txt>, September 1999. See also http://www.ietf.org/html.charters/policy-charter.html.

[9] For information on DEN, see http://www.dmtf.org/spec/denh.html.

[10] Petri Aukia, et al., "RATES : A Server for MPLS Traffic Engineering," IEEE Network, March/April 2000.

[11] M. Kodialam and T. V. Lakshman, "Minimum Interference Routing with Applications to MPLS Traffic Engineering," Proc. INFOCOM, Mar. 2000.

[12] Internet2 Bandwidth Broker (BB) Advisory Council.

[13] Internet2 QoS Working Group Draft, Draft Qbone Architecture, May, 1999.

[14] O. Schelen, "Quality of Service Agents in the Internet," Ph.D. thesis, Lulea University of Technology, August 1998.

[15] J. Case, K. McCloghrie, M. Rose, S. Waldbusser, "Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2)," IETF RFC1448, April 1993.

[16] F. Baker, K. H. Chan, and A. Smith, "Management Information Base for Differentiated Services Architecture," IETF Internet-Draft, draft-ietf-diffserv-mib-03.txt, May 2000.

[17] C. Srinivasan, A. Viswanathan, T. D. Nadeau, "MPLS Traffic Engineering Management Information Base Using SMIv2," IETF Internet-Draft, draft-ietf-mpls-te-mib-04.txt, July 2000.

[18] P. Wang, Y. Yemini and D. Florissi and John Zinky, "A Distributed Resource Controller for QoS Applications," NOMS2000, April 2000.

[19] OMG, "The Common Object Request Broker : Architecture and Specification," Revision 2.2, Feb. 1998.

[20] ITU-T Recommendation M.3010, "Principles for a Telecommunications Management Network," 1996.

[21] For information, see http://www.ietf.org/html.charters/policy-charter.html.

[22] M. Fine, K. McCloghrie, et al, "Differentiated Services Quality of Service Policy Information Base," IETF Internet Draft, draft-ietf-diffserv-pib-00.txt, Internet Draft, March 10. 2000.

[23] M. Li, A. Doria, J. Jason, "IPSec Policy Information Base," IETF Internet Draft, draft-ietf-ipsp-ipsecpib-00.txt, July 2000.

[24] J. Moy, "OSPF Version 2," IETF RFC 2178, July 1997.

[25] Y. Rekhter, T. Li, "Border Gateway Protocols (Ver 4)," IETF RFC 1654, July 1994.

[26] C. Kim, H. Choi, "Functional Specification of Routing Advisor for Traffic Engineering (RATE)," Technical Report, ETRI, June, 2000.

[27] B. Jamoussi, et. al., "Constraint-Based LSP Setup using LDP," IETF Internet Draft, draft-ietf-mpls-cr-ldp-04.txt, July 2000.

[28] D. O. Awduche, et. al., "RSVP-TE : Extensions to RSVP for LSP Tunnels," IETF Internet Draft, draft-ietf-mpls-rsvp-lsp-tunnel-07.txt, August 2000.

[29] Y. Yaacovi, et. al., "Lightweight Directory Access

Protocol (v3) : Extensions for Dynamic Directory Services," IETF RFC2589, May 1999.

[30] OMG, "Extensible Markup Language (XML) 1.0 (Second Edition)," October 2000.

[31] T. Bates, R. Chandra, "BGP Route Reflection : An Alternative to full mesh IBGP," IETF RFC 1966, June 1996.

[32] For information, see http://www.ist-tequila.org.

[33] J. Kim, J. W. Hong, T. Choi, "Constructing End-to-End Traffic Flows for Managing Differentiated Services Networks," DSOM2000, December 2000.

[34] Javasoft, "Java 2 SDK v1.3 Java™Remote Method Invocation (RMI)," see http://java.sun.com/j2se/1.3/docs/guide/rmi/index.html.

[35] For information, see http://www.juniper.com

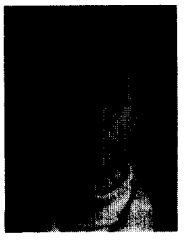
[36] D. Libes, "Exploring Expect," O'Reilly & Associates, Inc., 1995.

[37] G. Dhandapani and A. Sundaresan, "Netlink Sockets Overview," a technical paper of Department of Electrical Engineering and Computer Science, University of Kansas, September 13, 1999.

[38] For information, see UCD-SNMP Homepage, http://ucd-snmp.ucdavis.edu/.

[39] For information, see http://www-4.ibm.com/software/network/directory/.

[40] For information, see http://www.postgresql.org/.



최 태 상

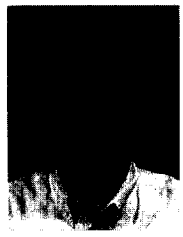
e-mail : choits@etri.re.kr

1995년 미주리-캔사스 주립대학 컴퓨터통신학과 공학박사

1996년~1998년 한국전자통신연구원 멀티미디어통신팀

1999년~현재 한국전자통신연구원 인터넷구조팀

관심분야 : QoS Management and Traffic Engineering in IP Networks, Interactive Multimedia Service System, Network, System, and Service Management



정 형 석

e-mail : chunghs@etri.re.kr

1986년~1993년 광운대학교 전자통신공학과 학사

1993년~1995년 광운대학교 전자통신공학과 대학원 석사

1995년~2000년 광운대학교 전자통신공학과 대학원 박사

1999년~현재 한국전자통신연구원 인터넷구조팀

관심분야 : QoS Management and Policy-based Network Management

최 희 숙

e-mail : hschoi@etri.re.kr

1992년 충남대학교 전산학 석사

1992년~현재 한국전자통신연구원 광대역통신망연구부

1999년~현재 한국전자통신연구원 인터넷구조팀

관심분야 : QoS Management and Traffic Engineering in IP Networks, Routing, Protocol and Network Management

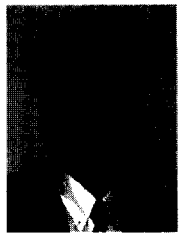
김 창 훈

e-mail : kimch@etri.re.kr

1993년~1999년 서울대학교 컴퓨터공학과 학사 및 석사

1999년~현재 한국전자통신연구원 인터넷구조팀

관심분야 : IP Traffic Engineering and Routing, Traffic Measurement



정 태 수

e-mail : tsjeong@etri.re.kr

1981년 경북대학교 전자공학과 (학사)

1983년 경북대학교 전자공학과 (석사)

1983년~현재 한국전자통신연구원, 책임연구원, 팀장

1991년~1992년 미국 UMass 대학 방문연구원

관심분야 : 통신망구조, 인터넷 시스템, LAN 시스템, 망관리