韓國 컴퓨터情報學會 論文誌
第9卷 第2號, 2004. 6.

2004-9-2-3-8

차등적인 QoS의 동적인 제공을 위한 정책 기반 QoS 관리 시스템의 설계 및 구현

차 시 호*, 김 규 호**

Design and Implementation of A Policy based QoS Management System to Support Dynamically Differentiated QoS

Si-Ho Cha*, Kyu-Ho Kim**

요 약

인터넷 QoS를 제공하기 위하여, IETF는 통합 서비스(IntServ)와 차별화 서비스(DiffServ)라는 두 가지 모델을 제안하였다. RSVP에 기반하여 각 패킷 흐름마다 자원 예약과 접근 제어를 수행하는 IntServ와 달리 DiffServ는 개개의 패킷 흐름 대신에 집합적인 패킷 흐름을 지원함으로써 다양한 패킷 클래스들에 대하여 차등적인 QoS를 제공할 수 있다. 그러나 DiffServ 구조가 사용자들의 동적인 QoS 변경 요구를 수용하기 위해서는 각 패킷 클래스에 대하여 동적인 QoS를 설정할 수 있는 관리 시스템이 요구된다. 따라서 본 논문은 DiffServ 네트워크의 QoS를 효율적으로 관리하고 동적으로 구성하기 위한 정책 기반 QoS 관리 시스템을 설계하고 구현하였다. 또한 테스트베드로 구축된 리눅스 기반 DiffServ 네트워크 상에서의 실험을 통해 구현된 QoS 관리 시스템의 타당성을 입증하였다.

Abstract

To provide Internet QoS, the IETF proposed two models of Integrated Services (IntServ) and Differentiated Services (DiffServ). Unlike IntServ where resource reservation and admission control is per-flow based using RSVP, DiffServ supports aggregated traffic classes to provide various QoS to different classes of traffics. However, a dynamic QoS management system is required to dynamically provide differentiated QoS for customers who require dynamic QoS change. This paper designed and implemented therefore a policy-based QoS management system to manage effectively and configure dynamically QoS of DiffServ networks. The validity of the system has been verified by the experimentation on the Linux-based DiffServ network.

▸ Keyword : QoS(Quality of Service), 차별화 서비스(DiffServ), 정책 기반 관리(Policy based Management)

* (주)웨어플러스 네트워크사업본부 기술개발팀 팀장, ** 서울보건대학 인터넷정보과 교수

Ⅰ. Introduction

The best-effort service model in current IP networks does not provide the QoS requirements of next generation network services. To solve this problem, the IETF (Internet Engineering Task Force) proposed two models of Integrated Services (IntServ)[1] and Differentiated Services (DiffServ)[2]. IntServ model is based on per-flow resource reservation and admission control through RSVP (Resource Reservation Protocol). The main disadvantage of IntServ is that the required information of flow states and the QoS treatments in a core IP network raise severe scalability problems[3]. DiffServ model, on the other hand, supports aggregated traffic classes rather than individual flows and provides different QoS to different classes of packets in IP networks. However, a dynamic QoS management system is required to dynamically provide differentiated QoS for customers who require dynamic QoS. From this reasoning, a QoS management system that can manage differentiated QoS provisioning is required. Recently, policy-based management (PBM)[3] has been considered as a technology that can provide QoS management supports. The amount of QoS management task can be reduced by using policies because one policy can be used for many policy targets that are various network nodes and the policy can accept customer's dynamic QoS requirements. There are several research projects for QoS guarantees using the PBM technology going on, but only few of them detail the design and implementation issues of a QoS management platform with PBM concepts.

The implementation of the policy-based QoS management system is build on EJB (Enterprise JavaBeans) framework and uses XML (Extensible Markup Language) to represent and validate high-level QoS policies[4][5].

This paper is structured as follows. Section 2 investigates backgrounds of this work. Section 3 discusses the architecture and components of the proposed QoS management system. Section 4 presents the implementation of the Liuux DiffServ router and the Qos management system and the experimental results in the video streaming system. Finally in section 5 we conclude the paper.

Ⅱ. Background

1. Differentiated Service

DiffServ is a technology to provide QoS in an efficient and a scalable way. It supports aggregate traffic classes rather than individual flows and provides a differentiated QoS to a different class of flows in IP networks. DiffServ provides a service differentiation for aggregated IP packet streams with different per hop behaviors (PHBs) by various differentiated service code point (DSCP) values. Edge routers mark each IP packet with a DSCP value and core routers forward the packet according to the corresponding PHB based on the DSCP value in the IP header.

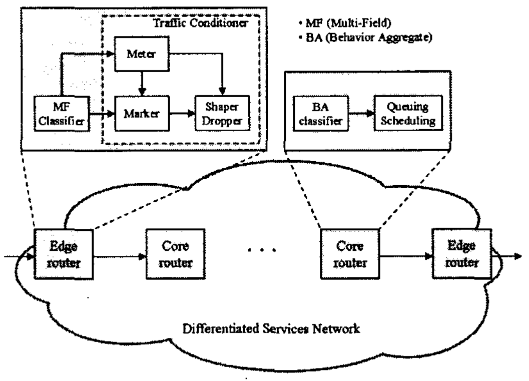


Figure 1. DiffServ architecture

As illustrated in Fig. 1, two main functional components of an edge router are traffic classifier and traffic conditioner block (TCB). The traffic classifier has an alias of multifield (MF) classifier and classifies packets based on their IP header field values such as source address, destination address, DSCP, protocol number or port address. TCB contains the functional blocks such as packet meter, marker, shaper and/or dropper. The work of TCB is to shape a packet to meet the policy requirement associated with the classification of a packet and then assign a DSCP value to the IP packet for a particular behavior. When a stream is passed through a DiffServ network, this DSCP value triggers a selected PHB in all core routers in the network by a behavior aggregate (BA) classifier. DSCP is simply a value set in the first 6 bits of the differentiated services (DS) byte in the IP header. It specifies a particular PHB to a router for a corresponding packet. Currently, four types of PHBs are specified for DiffServ networks: default behavior (best-effort), class selector (CLS), expedited forwarding (EF), and assured forwarding (AF).

2. Policy-based QoS management

The objective of the policy-based QoS management is to manage the QoS of connections using high-level policies that describe the behavior of the network in a way as independently as possible of network devices and topology. The amount of QoS management task can be reduced by using policies because one policy can be used for many policy targets that are various network nodes. The four main functions of the policy-based management, shown in Fig. 2, are graphical user interface, resource discovery, transformation logic, and configuration distributor. The graphical user interface is used by a network administrator to input SLAs (Service Level Agreements) to deploy QoS of a network. The service level that a network offers for a customer is defined by both an SLA, and a high-level QoS policy is defined by an SLA and an information on a network topology that is discovered by the function of resource discovery. The transformation logic verifies the high-level policies, and also translates the high-level policies into low-level policies that are distributed to various devices in the network. The configuration distributor is responsible for ensuring that the low-level policies are distributed to the various devices in the network[6].

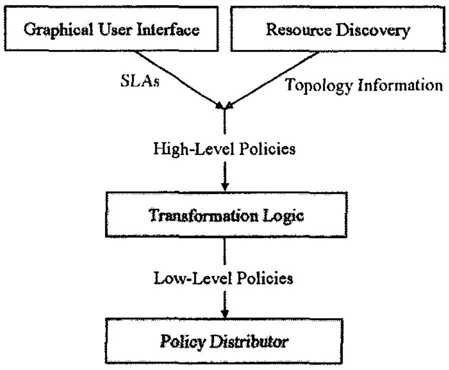


Figure 2. Policy-based management concept

3. EJB-based management architecture

The management architecture based on EJB technology simplifies the complexity of the development of multi-tier management systems, and enhances portability by separating the various complexities inherent in the multi-tier management applications, such as transaction management, life-cycle management, and resource pooling, from specifics such as business logic and user interfaces[7]. There are three main components in EJB technology - EJB component, EJB container, and EJB connector. An EJB component is a Java class that encapsulates functionality which is reusable in many applications. An EJB container provides a runtime environment for EJB components. The EJB connector provides a mechanism for EJB components to access data from legacy systems. EJB technology gives management system developers the ability to model the full range of objects found in network management by defining the following distinct types of EJB components: session beans, entity beans, and message-driven beans. Session beans represent behaviors associated with client sessions and are non-persistent objects. Entity beans represent collections of data and are persistent objects. Message-driven beans represent JMS (Java Message Service) consumers that implement business logic on a server[7]. It provides asynchronous communications between EJB beans. By these specifics, EJB-based management framework has a number of benefits as follows: encapsulation of business logic, simplified application development, extensibility, scalability, consistency, transaction management, database and directory access, container-managed persistence, and distributed object access.

III. Design

This section describes the proposed QoS management system. In this section we also discuss the components of the system, and QoS management procedures

1. architecture and component

The architecture of proposed QoS management system is shown in Fig. 3.

The system conforms to the Model-View -Controller (MVC) architecture. Therefore, it is highly manageable and scalable, and provides the overall strategy for the clear distribution of objects involved in managing service. There are two main components in the architecture: a Web server and an EJB-based policy server. A Web server is responsible for the presentation logic of the system. An EJB-based policy server is responsible for the business logic of the system. The system provides a Web-based interface for a network administrator to create and revise high-level QoS policies (HQPs) to be enforced on the DiffServ network. The system uses Java Servlets, JSP pages, and XML technologies to provide for the administrator's view.

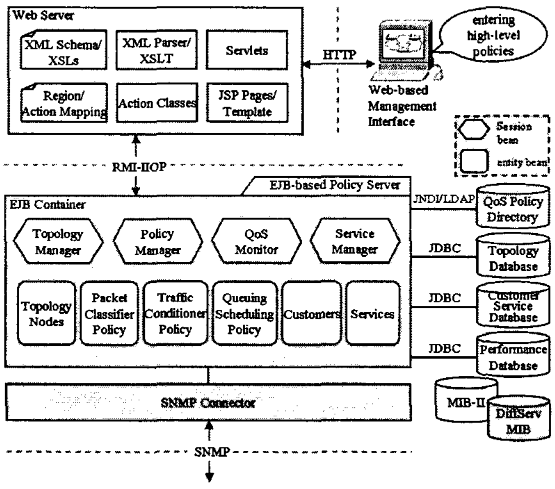


Figure 3. Architecture of a policy-based QoS management system

As illustrated in Fig. 3, there are several functional components in the EJB-based policy server in the system and they are used to discover a network topology, each router type, the composition and distribution of the low-level QoS policies (LQPs), the management of customer services, and the interfaces of the policy server. The system uses the following components to discover network topology and each router type. The topology node (TN) bean is an entity bean containing the information of a network topology and each router type. The information is retrieved using SNMP MIB-II. The topology database (TD) stores the topology information and each router type retrieved from a DiffServ network through SNMP. The topology manager (TM) bean is a session bean responsible for discovering the topology information and each router type and storing them into the TD and setting up the TN beans according to the retrieved information.

The system uses the following components to translate HQPs into LQPs and distribute the LQPs to the DiffServ network. The packet classification policy (PCP) bean is a part of LQP entity bean that classifies packet flows and assigns class identifiers to them. The PCP beans are deployed to edge routers and control the inbound traffics. The traffic conditioning policy (TCP) bean is a part of LQP entity bean that meters the classified packets to check whether they conform to a traffic profile and performs marking, dropping, and/or shaping packets according to the metering results. The TCP beans are deployed to edge routers and control the outbound traffics. The queuing and scheduling policy (QSP) bean is a part of LQP entity bean. It performs queuing, scheduling, and/or dropping packets. The QSP beans are deployed to core routers to control the outbound traffics. The QoS policy directory is a directory storing the LQP beans. The policy manager (PM) bean is a session bean that is responsible for translating HQPs into low-level QoS policy beans and setting the values of DiffServ MIBs of the routers. The PM is also responsible for deploying the LQPs to relevant routers in the DiffServ network. The QoS monitor (QM) bean is a session bean that is responsible for monitoring the QoS resulted from a policy deployment by retrieving the values of DiffServ MIBs and comparing them to the attribute values of the three LQP beans.

The system uses the customer bean, the service bean, and the service manager bean to manage a customer's services. An instance of a customer bean represents a customer. An individual customer's information is maintained in a customer service database. The service bean represents a service provisioning for a customer and the service manager bean guides the installation process of the service bean.

2. QoS management process

To process a policy-based QoS management efficiently and correctly, the following procedures are required: topology discovery, policy definition and validation, policy translation, policy deployment, and QoS monitoring. To process a policy-based QoS management efficiently and correctly, the following procedures are required:

Topology discovery: In order to describe the QoS of a DiffServ network, the system should have the knowledge of the routing topology and each router's role. The TM session bean accomplishes the discovery of the routing topology and router type discovery by using two SNMP MIB-II tables, ipAddrTable and ipRouteTable. The ipAddrTable contains IP addresses of all network interfaces in a router and the ipRouteTable contains an IP routing table that has a next hop host and a network interface for a set of destination IP addresses. The topology and router information discovered from the network are stored in the TD, and are represented as TN entity beans.

Policy definition and validation: The system defines HQPs as valid XML documents and validates the XML documents. A Java Servlet on the Web server receives QoS policy data from a Web browser and creates valid XML documents and then validates them. Once the HQPs are validated, a Java Servlet requests a PM session bean to create the instances of LQP entity beans, PCP bean, TCP bean, and QSP bean.

Policy translation: The translation of a QoS policy from HQPs to LQPs is done by the PM session bean on the EJB-based policy server. The PM session bean translates HQPs to LQPs by properly setting the attributes of the three LQP entity beans. The attributes of the LQP entity beans are mapped into the device configuration parameters to configure the DiffServ routers for provisioning QoS requirements.

Policy deployment: The deployment of LQPs is done by the three LQP entity beans. These three LQP entity beans perform SNMP operations for deploying each LQP. The PCP bean and the TCP bean are deployed to edge routers to control the functions of the edge routers, whereas the QSP bean is deployed to core routers to control the functions of the core routers. The PCP bean classifies packet flows and the TCP bean performs the traffic conditioning such as metering, marking, dropping, and/or shaping packets. The QSP bean performs queuing, scheduling, and/or dropping packets. A set of these actions is accomplished by using the DiffServ MIB. The DiffServ MIB describes a configuration and management aspect of DiffServ routers.

QoS monitoring: A deployed QoS policy might not behave as defined in the policy. The QoS monitoring in the system uses the same DiffServ MIB as in the policy deployment. The QM session bean accesses the policy definition in the three LQP beans and compares the observed behavior of a network to the one defined in the policy. If any QoS degradation is observed, the QM session bean notifies an administrator by alerting messages and updates the performance database.

Ⅳ. Implementation and experiment

1. Linux-based DiffServ routers

Linux-based routers are used for our DiffServ network testbed. Supporting differentiated services are already incorporated in the mainstream Linux kernel source code version 2.4 and later[8]. Fig. 4 shows the implementation architecture of a Linux DiffServ router. An SNMP agent with a MIB-II and a DiffServ MIB receives management operations from the EJB-based policy server and performs appropriate parameter changes in the Linux traffic control (TC) kernel. Communication between the SNMP agent and the Linux TC kernel is achieved through Netlink socket[9]. An SNMP agent containing a MIB-II and a DiffServ MIB has been implemented by using the UCD-SNMP package 4.2.2[10] that provides an agent extension capability.

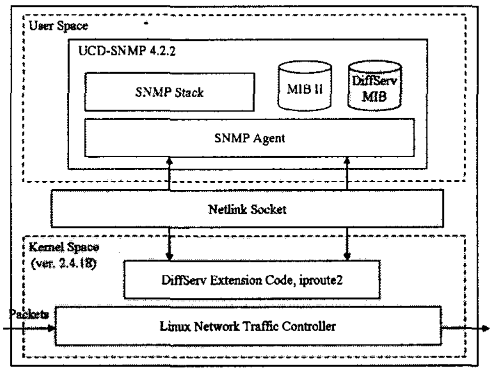

User Space
UCD-SNMP 4.2.2
SNMP Stack
MIB II
DiffServ MIB
SNMP Agent
Netlink Socket
Kernel Space (ver. 2.4.18)
DiffServ Extension Code, iproute2
Packets
Linux Network Traffic Controller

Figure 4. Architecture of a Linux DiffServ router

2. A QoS Management System

The system is implemented on a Windows 2000 server system. It consists of a Web server and an EJB-based policy server. The Web server is responsible for the presentation logic of the system and the EJB-based policy server is responsible for the business logic of the system. In the MVC architecture, the view is implemented using the JSP template mechanism and the Composite View pattern. The controller is implemented using the Front Controller pattern (a Action Servlet) and the Session Facade pattern (EJB Session Beans). The model is implemented using EJB Entity Beans and Service Locator pattern. We use Apache Tomcat 4.0.1 for the Servlet and JSP container. An EJB-based policy server within the business-tier runs an EJB server to manage EJB components. We use JBoss 2.4.10[11]for an EJB-based policy server and use EJB 1.1 to implement EJB beans. AdventNet SNMP API[12] written in Java are used for handling SNMP operations. The Oracle 8i

Enterprise Edition 8.1.6 for Windows NT is used for storing the performance and topology information derived from MIB tables.

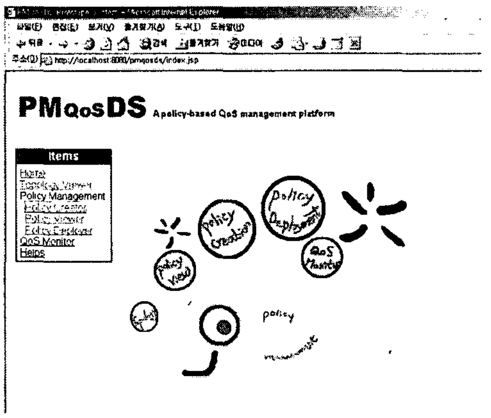
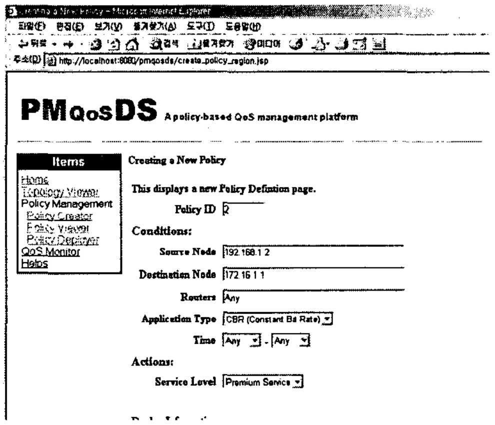


Figure 5. Snapshot of the system (index.jsp)


http://localhost:8080/pmqosds/create_policy_region.jsp

PMQoSDS A policy-based QoS management platform

Items	Creating a New Policy
Home Topology Viewer Policy Management Policy Creator Policy Viewer Policy Deployer QoS Monitor Helps	This displays a new Policy Defintion page.

Policy ID: 2

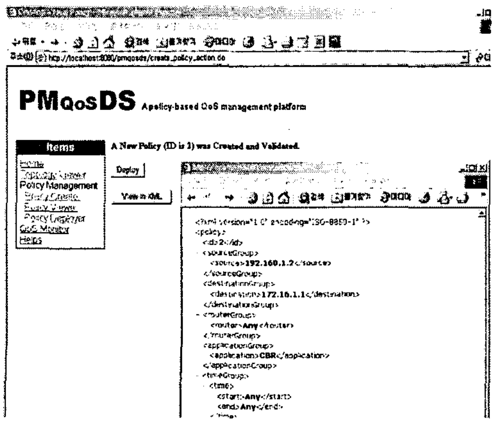
Conditions:

Source Node: 192.168.1.2
Destination Node: 172.16.1.1
Routers: Any
Application Type: CBR (Constant Bit Rate)
Time: Any - Any

Actions:

Service Level: Premium Service

Figure 6. Snapshot of the system


http://localhost:8080/pmqosds/create_policy_action.do

PMQoSDS A policy-based QoS management platform

A New Policy (ID is 2) was Created and Validated.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<policy>
  <id>2</id>
- <sourceGroup>
    <source>192.160.1.2</source>
  </sourceGroup>
  <destinationGroup>
    <destination>172.16.1.1</destination>
  </destinationGroup>
- <routerGroup>
    <router>Any</router>
  </routerGroup>
  <applicationGroup>
    <application>CBR</application>
  </applicationGroup>
- <timeGroup>
  - <time>
      <start>Any</start>
      <end>Any</end>
```

Figure 7. Snapshot of the system

Fig. 5 shows the main screen (index.jsp) of the policy-based QoS management system. Fig. 6 shows the input forms for the high-level policy information and Fig. 7 shows the result of the request for the policy creation of Fig. 6.

3. Experiment

To show the effectiveness of the system, we apply Windows Media Streaming Services to our DiffServ network. To do that, we configure a testbed shown in Fig. 8.

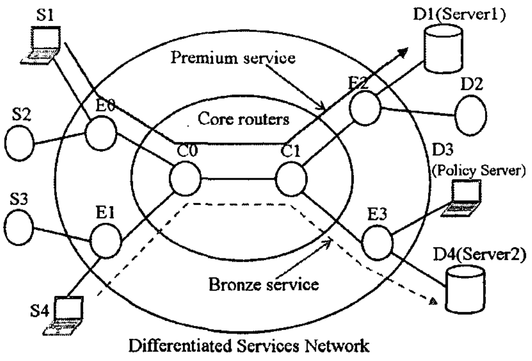


Figure 8. Experiment environment

Two streaming servers are attached to the network - one server to D1 and the other to D4. Fig. 9 shows the snapshot of the streaming server applied to the testbed. They have exactly the same hardware and software system configurations. A policy server is attached to D3. The systems in the testbed are running on the following hardware configurations. The core routers are running on Pentium IV 1.8 GHz with 512 MB main memory, the edge routers on Pentium IV 1.5 GHz with 512 MB main memory, two VOD servers on Pentium IV 2.0 GHz with 512 MB main memory, and the other systems on Pentium III 1.0 GHz with 256 MB main memory. All the links in Fig. 8 are connected via FastEthernet NICs.

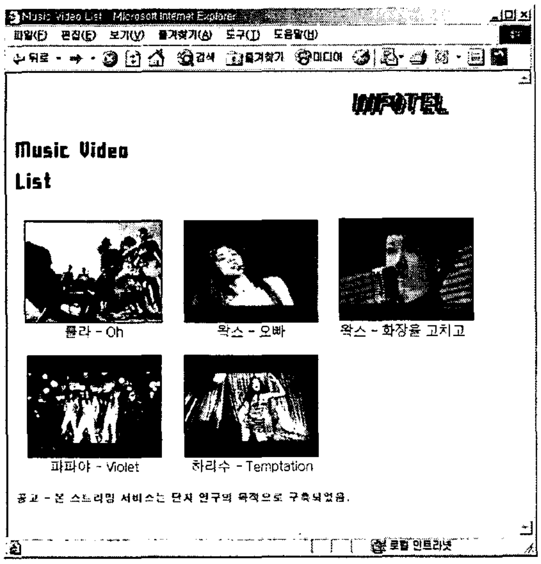

Music Video List - Microsoft Internet Explorer
INFOTEL
Music Video List
클라 - Oh
왁스 - 오빠
왁스 - 화장을 고치고
파파야 - Violet
하리수 - Temptation
공고 - 본 스트리밍 서비스는 단지 연구의 목적으로 구축되었음.

Figure 9. Snapshot of the streaming server

In the configuration, there are three connections running - two for multimedia connections and the other one for cross traffic. Two connections for multimedia traffics are the connection between S1 and Server1, and the one between S4 and Server2. Those connections share a link between C0 and C1. To differentiate the services between them, the connection between S1 and Server1 is applied by Premium service, while the other multimedia connection between S4 and Server2 is applied by Bronze service. To make the sharing link congested, MGEN toolset[13] is used to generate cross traffics on that link, and CBR traffics are used to achieve that goal. Cross traffics are generated at C0 and sinked at C1 router. By doing this, the service levels and the resulted QoS can be explicitly demonstrated.

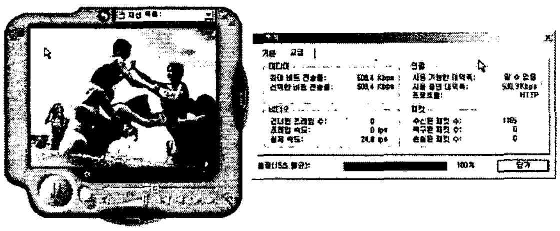


Figure 10. Snapshot of the client S1

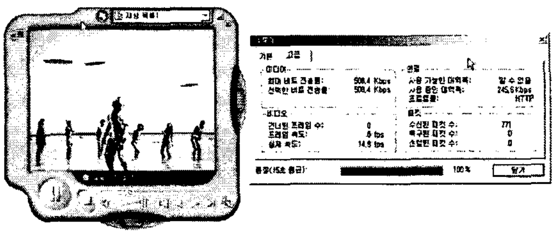


Figure 11. Snapshot of the client S4

Fig. 10 and Fig. 11 show the results of the experiment. Fig. 10 is the snapshot of the connection with Premium Service between S1 and the streaming Server1. Fig. 11 is that of the connection with Bronze Service between S4 and the streaming Server2. As shown in Fig. 10 and Fig. 11, the difference in the video quality of each connection is explicit. The client S1 with Premium service receives a video stream with 530.3 kbps, while the client S4 with Bronze service receives a video stream with 245.6 kbps.

From the experiment, we can verify that the proposed policy-based QoS management system provides differentiated QoS levels to the contending connections using the management platform. Obviously, this work can be extended to a network with more complicated connections.

V. Conclusion

In this paper, we proposed and implemented a policy-based QoS management platform for DiffServ enabled IP networks. The system integrated the functions of policy management and QoS monitoring by extending the original IETF PBM architecture to the policy-based QoS management. We presented both a QoS management model for policy-based QoS provisioning and a QoS management mechanism. We showed the QoS management procedures as well as the structures and components of the system.

To show the effectiveness of our system, we experimented with video streaming systems in our Linux-based DiffServ testbed. In the experiment, we demonstrated that our system is able to manage differentiated QoS provisioning in a

DiffServ network. Because this work can be obviously extended to a network with more complicated connections, we are currently extending our system with the various QoS policies on the testbed to study how the system can provide differentiated QoS guarantees with various service requirements. We expect our system to be successfully integrated in the service management systems used by the service providers in order to meet various dynamic QoS requirements from their customers.

참고문헌

[1] R. Braden, D. Clark, S. Shenker, "Integrated Services in the Internet Architecture: an Overview", IETF RFC 1633, June 1994.

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

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

[4] Si-Ho Cha, Kuk-Hyun Cho, "A Policy-Based QoS Management Framework for Differentiated Services Networks", Springer-Verlag's Lecture Notes in Computer Science (LNCS) Vol. 2662, August 2003.

[5] Si-Ho Cha, Kuk-Hyun Cho, "Policy-based Differentiated QoS Provisioning for DiffServ Enabled IP Networks", Springer-Verlag's Lecture Notes in Computer Science (LNCS), accepted for publication, June 2004.

[6] D. C. Verma, "Policy-Based Networking: Architecture and Algorithms", New Riders, November 2000.

[7] Sun Microsystems, Inc, "Telecom Network Management With EnterpriseJavaBeansTM Technology", Technical White Paper, May 2001.

[8] Differentiated Services on Linux, http://diffserv.sourceforce.net.

[9] G. Dhandapani, A. Sundaresan, "Netlink Sockets-Overview, University of Kansas", September 1999, http://qos.ittc.ukans.edu/netlink.

[10] UCD-SNMP Package, University of California at Davis, http://www.net-snmp.org/.

[11] JBoss Application Server,

저 자 소 개

차 시 호

1995년 순천대학교 전자계산학과 (이학사)

1997년 광운대학교 전자계산학과 (이학석사)

2004년 광운대학교 컴퓨터과학과 (공학박사)

2000년 대우통신 종합연구소 선임연구원

현 재 (주)웨어플러스 기술개발팀 팀장

〈관심분야〉 네트워크 소프트웨어, 네트워크 관리, 차세대 인터넷, 소프트웨어 컴포넌트 기술

김 규 호

광운대학교 전자계산학과 졸업 (이학사)

광운대학교 대학원 전자계산학과 (이학석사)

광운대학교 대학원 전자계산학과 (이학박사)

서울보건대학 인터넷정보과 교수

한국컴퓨터정보학회 이사

〈관심분야〉 네트워크 관리, 유비쿼터스 네트워킹, 액티브 네트워킹 기술