# Performance Analysis of Web Service Architecture for Inter-AS DiffServ-over-MPLS Traffic Engineering

Shanmugham Sundaram [a], Youngsu Chae [b], Young-Tak Kim [c]
Yeungnam University, KOREA

**Abstract**

In this paper, the performance of the WebService architecture for QoS guaranteed connection provisioning in inter-AS domain networks has been measured and analyzed for service publish/inquiry, collection of NMSs ASBR details, source routing by ingress NMS in constraint based routing and connection establishment. From the analysis, it has been found that, the connection between inter-AS domain networks can be established within the usual time limits of 3 seconds by the WebService architecture. Since no standard solutions have been implemented in Interdomain QoS provisioning, this performance analysis assures WebService architecture as a promising solution and can be easily implemented in the early stages of MPLS network employment.

**Keywords:** Distributed NMS, QoS-guaranteed, DiffServ-over-MPLS, interprovider QoS, DMTF CIM.

## 1. Introduction

Since there has been no standardized NNI signaling mechanisms in the QoS guaranteed DiffServ-over-MPLS inter-AS domain networks, WebService architecture (WSDL, SOAP/XML messaging & UDDI registry) based connection management has been proposed [1]. In real-time multimedia service provisioning, there is constraint being imposed on time taken to establish end-to-end connection in inter-AS domain networks. The connection should be established within certain time limits. But the WebService architecture's performance capability to establish end-to-end connection within the usual time limits has not been analyzed in the inter-domain traffic engineering.

For connection provisioning in real-time multimedia services, the time constraint should be satisfied. This constraint has created an importance to analyze the WebService architecture, whether WebService functional modules can deliver the connection establishment within the usual specified time limits of 3 seconds for telephone service.

This paper analyzes the WebService architecture performance on (i) publishing/inquiring NMS services and service end points to/from the service registry (WSDL) by NMSs or CNM, (ii) gathering the information of peer AS domain's ASBR (autonomous system boundary router) ports, available bandwidth, and delay (SOAP/XML), and (iii) end-to-end connection establishment (SOAP/XML) in inter-AS domain networks on the calculated route, i.e. based on CSPF. In this paper, we propose DMTF CIM based MO design and implementation with hierarchical inheritance and polymorphism for inter-AS domain traffic engineering.

The rest of this paper is organized as follows. In Section II the related works are briefly introduced. In section III, the Web service architecture design is explained. In section IV, the Web service implementation is explained in detail, and in section V, the performance of the WebService architecture during connection establishment in inter-AS domain networks is analyzed. In section VI, the conclusion and future works are given.

## 2. Related Works

### 2.1 Web Services

Web service is being standardized by the World Wide Web Consortium (W3C), and promises to provide a single uniform software infrastructure to support a wide range of distributed services. It provides the distributed computing with interoperability between application running in different platforms and in different languages.

WebService share common set of open standard technologies such as SOAP/XML, WSDL, UDDI [2] and ebXML [3]. In WebService paradigm, as shown in Fig. 1, there are three actors: (i) a service provider, (ii) service requestor, and (iii) Service registry. The service provider and service requestor can use any platform or programming languages to interact each other and the messaging between them would be SOAP/XML. The service registry could be implemented either by using UDDI specification or ebXML specification. The ebXML is basically an open XML-based infrastructure enabling the global use of electronic business in an interoperable, secure, and consistent manner by all parties [3].

In DiffServ-over-MPLS traffic engineering, every Network Management System (NMS), which manages an autonomous system (AS), would register their service offerings and other parameters in the common service registry.

*This work has been supported by Yeungnam University IT Research Center (ITRC) Project.

[a]Dept. of Information and Communication Engineering
214-1, Dae-Dong, Gyeong-San, Gyeongbuk, 712-749 , Korea
shanmughams@yumail.ac.kr
[b]yschae@yu.ac.kr, [c]ytkim@yu.ac.kr

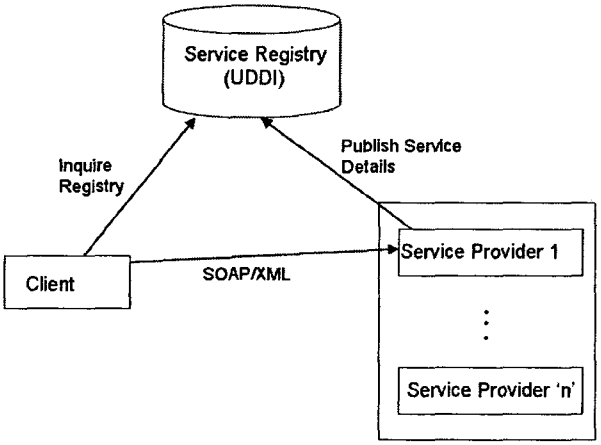


**Fig. 1. Web Service Components**

The details are given in the Web Service Definition Language (WSDL) [4] format. The clients or requestors can inquire the registry and make corresponding service calls to the service providers with the service end point. The Web service can be used in two cases of inter-domain networking: (i) DiffServ-over-MPLS service access by the CNM to the service provider, and (ii) interaction among the NMSs, as standard MPLS NNI signaling is not mature yet.

### 2.2 MESCAL – Solutions for interprovider QoS

The MESCAL [5] project which is funded by EU for IP Premium Service project as the part of the IST program, aims to propose solutions for the deployment and delivery of inter-domain Quality of Service across the Internet. In MESCAL based approach, the service providers (who can have many inter-AS domain networks) co-operate each other to set up QoS capabilities for building an end-to-end QoS delivery chain. The provider can extend the geographical scope of its QoS-based service offering (QC classes) across multiple domains. The scope can be expanded by finding and binding the appropriate peer's service classes. The Fig. 2 shows the QoS provisioning model in MESCAL approach.

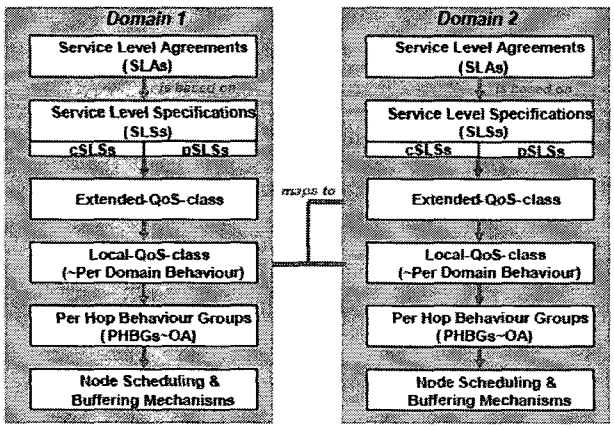


**Fig. 2. MESCAL QoS Service Model**

By the extended-QoS classes, the service provider can offer service to the remote locations even when its network infrastructure is not there. Mescal provides some operations for building interprovider QC operations, such as QC-advertisement, QC-discovery, QC-mapping, QC-binding & QC-implementation. In our implementation and analysis, the QC-advertisement and QC definitions of MESCAL design are considered

### 2.3 DMTF-CIM

Distributed Management Task Force (DMTF) is an industry consortium that develops, supports, and maintains standards for systems management of PC systems and products, to reduce total cost of ownership.

**Table 1. CIM Core data model**

| CIM Schema v2.10 | MOF (Managed Object Format) |
|---|---|
| CIM_Application | Software Feature, Software Element |
| CIM_Core | Managed System Element, Physical Elements, Logical Devices |
| CIM_Database | Database Environment, Database Storage |
| CIM_Device | Power, Controllers, Processors, Logical Ports, Memory |
| CIM_Network | Network Systems, Routing and Forwarding, Routes, OSPF, SNMP |
| CIM_Physical | Physical Packages, Physical Component, Physical Capacity |
| CIM_System | Computer System, Local File System, Operating System, UNIX |
| CIM_Event | SNMPTrapIndication, AlertIndicatio, IndicationHandler |

The CIM is a conceptual information model defined by DMTF for describing managed entities, their composition, and relationships. CIM describes the contents and semantics of manageable entities across an enterprise in an Internet-friendly way. The management models are comprised of a Core Model and a set of Common Models that extend from the Core, which is shown in Table 1. Common models have been defined for systems, services, networks, applications, users, and databases. CIM is defined using a language called Managed Object Format (MOF). MOF is a textual format for describing an information model using an object-oriented design. MOF is used to express the classes, associations, aggregations, properties, and methods that are part of the management domain. CIM schema is extensible which reduces the class design time, compared to the design from the scratch. The operations that are defined for CIM are independent of the protocol used. In this paper, we used CIM MOFs with extensions required for inter-AS traffic engineering. To gather in the information between the NMSs, we used Web service architecture.

## 3. Architecture of Web Service Based Distributed Network Management System

### 3.1 WebService Architecture for distributed NMS for DiffServ-over-MPLS TE

The distributed network management systems can be implemented by using the standardized web service tools. Each NMS which manages the underlying autonomous systems is integrated by the Web service mechanisms. Each NMS would publish the services it offers to the customers, either peer domains or real customers. The CNM/NMS can query registry about the other NMSs service details and can make use of them for intra-AS domain connection provisioning. The services can be defined in terms of service class with each class having the unique network parameters of bandwidth offered, jitter, and delay.

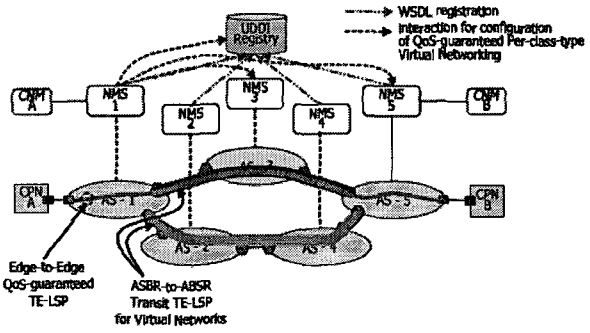


**Fig. 3. Web Service architecture based distributed NMS**

The Fig. 3 explains the overview of the WebService architecture based distributed network management system for DiffServ-over-MPLS traffic engineering which has been designed, and its performance is analyzed. Each NMS that manages a domain registers their services to the service registry (UDDI). For each service class type the NMS offers, there can be many LSPs available. The details of the corresponding ingress/egress IP also need to be published to the service registry, in case of making multiple paths for handling faults. The service details and the service enquiry points (i.e., service access points) are described in the Web Service Definition Language (WSDL) and published in the registry. The data actually transferred during publish/inquiry are in XML format.

### 3.2 Interaction sequence for creating end-to-end QoS guaranteed service with CSPF routing

For the connection establishment across multiple AS domain network, firstly the ingress NMS must find the shortest path route that satisfy the requested constraints. Fig. 4 depicts an example of route computation. The NMS of the ingress AS domain network, to which the connection establishment requester (user A) is connected, and performs source routing. It finds the URL and related MIB/MIT contact points of the AS network to which the destination terminal or site (user B) is connected, and retrieves the ASBR port list. Then, it confirms the connectivity and availability of the requested bandwidth from the egress NMS. Also the neighbor AS networks (i.e., intermediate AS domains) of the egress AS domain are searched, and their ASBR ports are evaluated. The constraint-satisfied shortest path is selected.

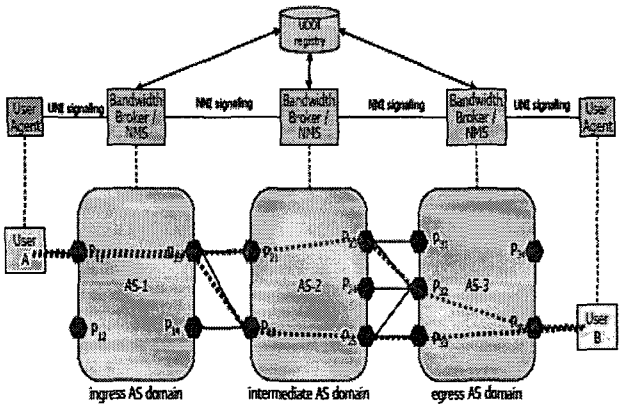


**Fig. 4. Constrained based shortest path routing**

This procedure is repeated until the ingress AS domain is reached through multiple possible routes. Finally, the call requesting user can be connected to a port of the ingress AS domain network. For fault restoration, the working path and the backup path are individually processed, with SRLG (shared risk link group) constraint. The collection of all network resource information for CSPF route calculation depends on the network topology and the number of AS networks along the route.

### 4. Implementation

The implementation of inter-AS TE involves many steps. Since DMTF's CIM network model does not provide all the necessary MOFs to represent the MOs for inter-AS TE, we extended the existing CIM for representing the DiffServ-over-MPLS TE. The MOFs (i.e., CIM with extensions) is converted to java class files by using the MOF compiler [13] mof2bean that is provided for WBEM solutions.

The NMS uses the java files created by mof2bean compiler. Then NMS generates WSDL file for the service it is offering, and publishes the WSDL URL reference to the service registry. All the NMSs would do the steps of publishing of their available transport service provisioning, as mentioned above. Whenever it is required, the NMS would query the service registry (stored in UDDI), retrieve the contact points of each AS domain networks, and make a WebService request to other NMSs.

Fig. 5 shows the implementation sequence of MO creation from the extension of DMTF-CIM, WSDL creation, publishing, retrievals of contact points of available services, and request via XML/SOAP.

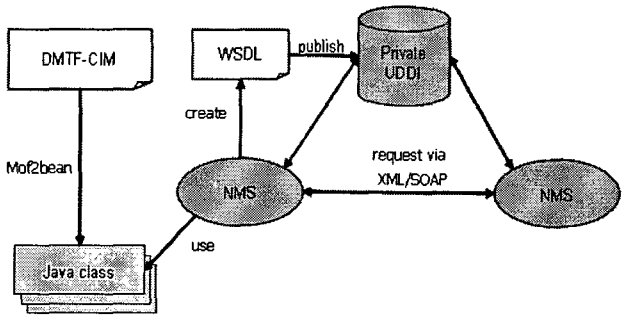


**Fig. 5. MO creation and management**

### 4.1 Implementation of CIM-based MO

The CIM MOF schema has been used to implement and design Managed Objects. The MO design with hierarchical inheritance and polymorphism among MO classes will make easy implementation of new MOs for new management function with reusable common information of base MO, as the object-oriented design approach provides. Fig. 5 depicts the basic structure of inheritance for basic MOs in inter-AS traffic engineering. The MOFs NMS, MPLSNetService, MPLSNetwork, ASBRMPLSRouter, ASBRMPLSRouterService, and ASBRPort.

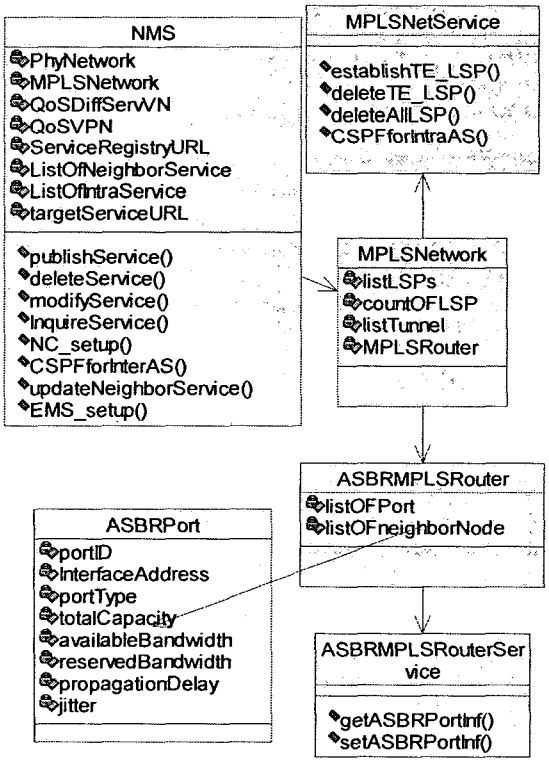


**Fig. 6. Extended CIM classes for inter-AS TE**

Fig. 6 show the detailed attributes and actions of MOFs designed for inter-AS traffic engineering purpose. NMS has all the layered network objects and it will handle them. Also NMS has several attributes like ServiceRegistryURL, ListOfNeighborService, ListOfIntraService, targerServiceURL. ServiceRegisryURL attribute holds the address of UDDI registry and it could be used while publishing and inquiring. In the proposed Web service architecture, private UDDI was used for security. Since it is private UDDI, only registerd NMSs could access the UDDI registry. ListOfNeighborService attribute holds WSDL URL information of the neighbor NMS. It could be a collection type. When NMS needs to make Web service request to the neighbors, it does not need to inquire UDDI for the details. Instead, it can directly make a request by using ListOfNeighborService attribute. This reduces the proessing time by avoiding UDDI registry inquiry to gather the neighbor details. ListOfIntraService attribute has the details of services which one NMS can publish to the UDDI registry. TargetServiceURL attribute has the specific neighbor WSDL URL for which the Web Service request can be made. The ASBRPort class has several attributes to represent the port status.

**Table 2. Extended CIM MO actions**

| Action on MOs | | Description |
|---|---|---|
| NMS | publishService | Publishing NMS Service to service registry |
| | deleteService | Deleting the NMS Service from service registry |
| | modifyService | Update the existing service with new details in th service registry. |
| | InquireService | Gather other NMS service from the service registry |
| | NC_Setup | Establish connection for inter-AS domain network. Requested by CNM. |
| | CSPFforInterAS | For inter-AS CSPF computation |
| | updateNeighborService | Maintain the neighbor service details in the local cache |
| | EMS_setup | Establish connection in the underlying managed sub-network |
| MPLSNetService | establishTE_LSP | Establish TE_LSP |
| | deleteTE_LSP | Remove the established TE_LSP |
| | deleteAllLSP | Remove all established TE_LSP |
| | CSPFforIntraAS | For intra-AS CSPF computation |
| ASBRMPLSRouterService | getASBRPortInf | Gather ASBR port information |
| | setASBRPortInf | Set ASBR port information |

Table.2 shows explanation of necessary actions with the corresponding MOFs. The publishService action of NMS MO is used to publish NMS service to the service registry with the details of ListOfIntraService. DeleteService action is used to delete the service, when there is no availability of the registred service. ModifyService action is for modifying the existing service which was already registered. This may be due to changes happening in the existing service. InquireService action is used by NMS to gather other NMS service from the service registry. These inquired details will be saved in the ListOfNeighborService attribute and will be used in future. NC_Setup action is used to create connection

in inter-AS domain network and it will be used by the CNM. When this is called, it makes CSPF computation. This will result to gather ASBRport information of other NMS by making Web service request. The updateNeighborService action is used store the neighbor service details in the local cache for the future use. The EMS_setup will be used to call the EMS for establishing the connection in the intra-AS domain network. The establishTE_LSP, deleteTE_LSP, and deleteAllLSP actions are used to handle the TE_LSP connections in the underlying domain network. In ASBRMPLSRouterService class has overloaded actions to gather the ASBR port information. When getASBRPortInf action with single parameter of portAddress is passed, it would gather the local ASBR from the managed NMS. When it is called with two parameters (with the collection attribute), it makes Web service request to peer AS and passes the ASBR information in the collection. The setASBTPortInf action would set the ASBR port information

### 4.2 Implementation of NMS

The NMS has been implemented by Java 5 SDK for GUI part, C++ as the backend core, and the open source software toolkits such as Apache Axis (for java 1.2) [7], and Tomcat 5.0 [8] (as Web service container) for the Web service part. The MO classes are obtained by compiling CIM MOF by mof2bean compiler [9]. Every NMS has both client and server part for the Web service implementation. Fig. 7. shows the NMS with WebService tools implementation. The JAXR APIs are used for both Web services publishing and inquiring. The dynamic invocation interface (DII) method is used to invoke the Web service call from the remote NMS. The interactions among NMSs are in SOAP/XML over HTTP.

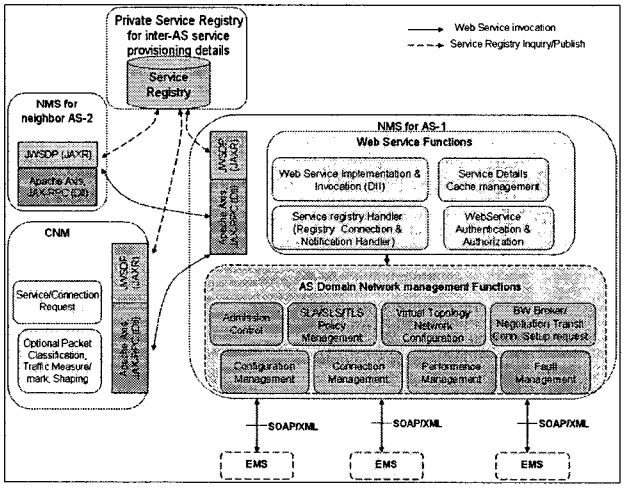


**Fig. 7. NMS with Web Service – overview**

The Java API packages have been built for handling the service inquiring, publishing to the service registries. Other packages are developed for (i) the client/service implementation of the Web service, (ii) registry handling, (iii) cache for service registry information, and (iv) performance analysis test cases. The integrated development environment has been Netbeans 4.1, and Borland's JBuilder X with Java 5 SDK.

### 4.3 Implementation of Service Registry

The service registry is configured by Sun Java system application server 8.1[10] and the Java Web Services developer pack (JWSDP) 1.6 [11]. Service registry has been implemented with ebXML 3 specification. The database used by service registry is Derby.

All the NMSs register themselves to the service registry through the web interface. NMS can either programmatically or manually publish their details to the registry. For publishing details in the service registry, user account need to be created and for every update in the registry, the user has to be authenticated by X.509 certificates. The keytool provided by JAVA SDK is used for making the X.509 certificates in a keystore file.

## 5. Performance Analysis of Web service Architecture

### 5.1 Distributed NMS test bed setup

The distributed NMSs have been setup with a single service registry, five NMS systems and two CNM systems as shown in Fig. 3. There are three cases considered for the analysis and the performance of the WebService components for delivering the requests/response. The performance details are measured and plotted in the graphs shown below.

### 5.2 UDDI Registry – Service Publish/Inquiry

The first case, the time taken for the NMS to publish the service details to the service registry is measured and is shown in Fig. 8. The details of service contact points, and network parameters are registered along with the ingress/egress IPs. The measurements are in milliseconds. The service publishing involves two main activities: (i) user authentication, and (ii) storing the details in the Derby database by the registry.

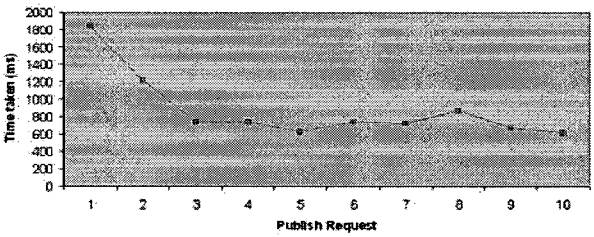


**Fig. 8. Time taken by NMS for subsequent publishing to the service registry**

The time taken to publish details to the service registry in the first request is high (say 1800 ms). This is because of the user authentication step involved while setting up connection with the service registry. The user authentication is done by using X.509 user certificates. The subsequent interactions does not involve user authentication and only database search is made. It has also been found that on the average it takes 800 ms. There is always an initial setup delay which is unavoidable (i.e., in terms of authenticating the user and database search).

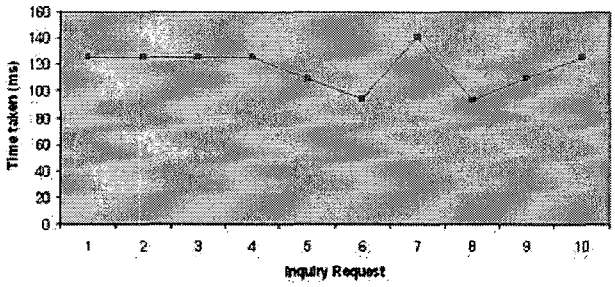


**Fig. 9. Time taken by NMS for subsequent inquiries**

The second case, the time taken to inquire the service by the NMS is measured and is provided in Fig. 9. During connection establishment, the NMS contacts the service registry to gather the service details and the service contact points of the other NMSs. The inquiry to the registry does not involve any user authentication. The service registry just does the database search for the registry objects being requested by the NMS. The matched registry objects are populated and sent to the requested client. On the average, the inquiry takes 130 ms.

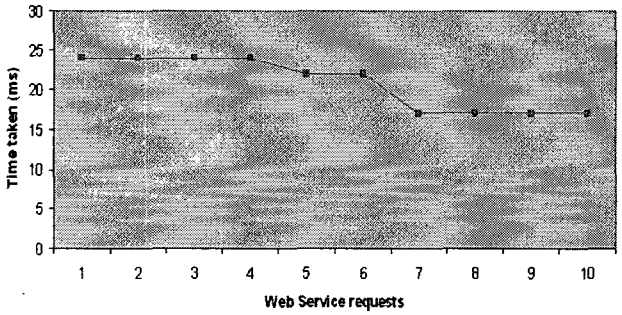


**Fig. 10. Time taken for NMS-NMS Web services interaction.**

The final case, the time taken to make a service call between the peer NMS systems are measured. When NMS needs to find the other NMSs details (ASBR, availability bandwidth, and the delay for the specified DiffServ class type), it will make a Web service request and gets the details. Fig. 10 depicts the scenario of this NMS-NMS Web service interaction. They are comparatively faster than other test cases, and on the average it takes around 20 ms.

### 5.3 CSPF Routing

Fig. 11 shows the CSPF routing performance time. This CSPF routing computation process includes one UDDI Query, four Web service requests and Dijkstra computation. On the average, it takes 218 ms for CSPF computation. If each NMS has multiple neighbor NMS, it could make scalability problem during CSPF computation. To solve the scalability issue, getASBRPortInf action can make threaded Web service request to each of the connected neighbors.

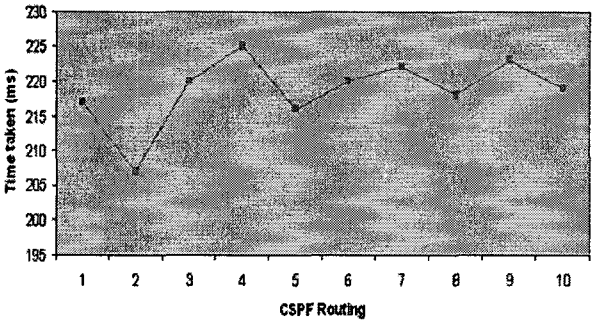


**Fig. 11. CSPF computation**

### 5.4 End-to-End Connection Establishment

Fig. 12 shows the overall picture of the performance of the WebService architecture for end-to-end connection provisioning in the DiffServ-over-MPLS TE. From the analysis, it has been found that the connection establishment in inter-AS domain could be made within the usual time limits of 3 seconds.

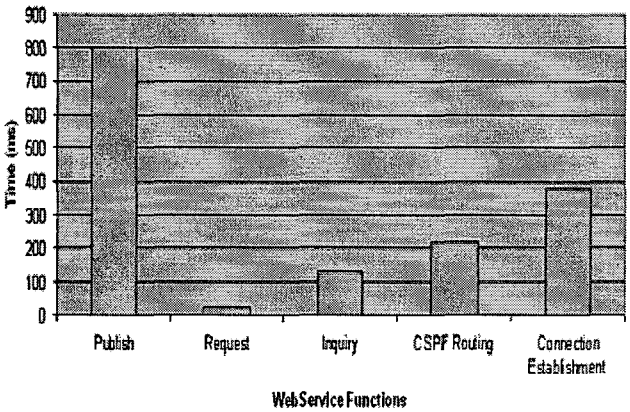


**Fig. 12. End-to-end connection establishment**

The performance of the Web service architecture for end-to-end QoS guaranteed service provisioning in DiffServ-over-MPLS TE can be improved by avoiding the service registry interactions as much as possible. This can be done

by building a local cache in NMS, which resembles details in the registry. When NMS is starting up, it gathers the registry details and builds the local cache. This local will be consulted by each & every service request and the corresponding NMS will be contacted, thereby avoiding the service registry interaction. The implementation of Notification mechanism in the NMS will also reduce the interaction between the NMS and service registry. This makes the NMS not to query the service registry for the updates, instead, it will become service registry's responsibility for updating the NMS, whenever there is any change, and that kind of changes has been interested by the NMS.

## 6. Conclusion

The performance of WebService architecture based distributed network management system for QoS guaranteed end-to-end DiffServ-over-MPLS TE has been studied and analyzed. We have proposed CIM extensions for inter-AS traffic engineering. By using the WebService, the QoS guaranteed end-to-end connection between inter-AS domain networks could be established within the usual time limits of 3 seconds. There are other distributed computing architectures like CORBA or RMI that can be used for the interprovider QoS. RMI has a limitation that it is not interoperable between applications running in different languages. CORBA is more like client/server driven and the applications are tightly coupled. WebService overcome these drawbacks and proves itself a promising candidate for interprovider QoS.

Some fine tunings are required to be done in the areas of WebService client side implementation, which will definitely improve the performance of the system. The future plans are to improve the performance of the system by (i) design and implementation of cache system, (ii) design and implementation of subscription notification system, so that whenever there is any change in the service registry, the NMS will be notified with the changes.

## References


[1] Youngtak Kim, Hyun-Ho Shin, "Web Service based Inter-AS Connection Managements for QoS-guaranteed DiffServ-aware-MPLS Internetworking," Proc. of International Conference on Software Engineering Research, Management & Applications (SERA 2004), San Francisco, pp. 256-261.

[2] http://www.uddi.org/.

[3] http://www.ebxml.org/specs/index.htm.

[4] http://www.w3.org/TR/wsdl.

[5] http://www.mescal.org/presentations/mescal-introduction.pdf.

[6] The Java Web Services tutorial http://java.sun.com/webservices/docs/1.6/tutorial/doc/index.html

[7] The Apache User Guide http://ws.apache.org/axis/java/user-guide.html

[8] http://jakarta.apache.org/tomcat/.

[9] http://www.sun.com/software/xml/developers/jaxb/.

[10] Sun java system application server 8.1 http://java.sun.com/j2ee/1.4/download.html.

[11] http://java.sun.com/webservices/jwsdp/index.jsp.

[12] Netbeans Quick start guide http://www.netbeans.org/kb/41/quickstart.html.


**Shanmugham Sundaram** is a Ph.D. candidate in Information and Communication Engineering at the Yeungnam University, Kyoungsan, Korea. His research interests are in the areas of Qos-guaranteed DiffServ-over-MPLS, sensor networks, and distributed computing. He received his M.C.A from the Bharathiyar University (India) in 1996.

**Youngsu Chae** received the B.S. and the M.S. degrees in computer science from Pohang University of Science and Technology, Pohang, Korea, in 1994 and 1996, respectively. He is currently a faculty member of School of EECS, Yeungnam University, Kyoungsan, Korea. His research interests include mobility management and QoS support in mobile networks, large scale Internet service architecture, and 4G networks.

**Young-Tak Kim** is a professor in the school of electrical engineering and computer science (EECS) of Yeungnam University, Korea. He graduated Yeungnam Univ. in 1984, and received Master Degree and Ph.D. degree from KAIST (Korea Advanced Institute of Science and Technology) in 1986 and 1990, respectively. He joined Korea Telecom (KT) in March 1990, where he had researched and developed an ATM Metropolitan Area Network (MAN) Switching System (ATM-MSS). He transferred to Yeungnam University in September 1994. He has performed many research projects in the area of "High-speed Telecommunications Networking" and "Network Operations and Management." Currently he is the director of government supported University IT Research Center (ITRC) which is developing "QoS-guaranteed Traffic Engineering and Multimedia Service Platform in the Broadband convergence Network (BcN)." His research interests include QoS-guaranteed inter-AS traffic engineering and broadband mobile Internet service provisioning on a broadband converged wired & wireless network environment. Prof. Young-Tak Kim is a member of IEEE Communication Society, KICS (Korea Institute of

Communication Society), KISS (Korea Information Society), KIPS (Korea Information Processing Society), and Korea Multimedia Society. He has been working as an organization committee member of APNOMS (Asia Pacific Network Operations and Management Symposium), and IEEE NOMS-2004 (Network Operations and Management Symposium).respectively. He is currently a faculty member of School of EECS, Yeungnam University, Kyoungsan, Korea. His research interests include mobility management and QoS support in mobile networks, large scale Internet service architecture, and 4G networks.