QoSME: Toward QoS Management and Guarantees*

Phil Y. Wang
pywang@nortelnetowrks.com
Technology Centre, Nortel Networks
3500 Carling Avenue, Ottawa
ON, K2H 8E9, Canada

Yechiam Yemini, Danilo Florissi
{ yemini, df }@cs.columbia.edu
Computer Science Dept
Columbia University, New York,
NY10027, USA

Patricia Florissi
patricia@smarts.com
SMARTS
14 Mamaroneck Avenue
White Plains, NY10601, USA

ABSTRACT

QoS (Quality of Service) is an emergent necessitate of next generation Internet, and has been widely investigated by the computer and communication community. QoSME (Quality of Service Management Environment) is presented in this paper as a general solution of end application QoS management and guarantees. Some related research works briefed firstly demonstrate that delivering QoS to end applications easily is highly demanded. QoSME architecture employs the current prevailing network protocols and technologies (e.g., IP and ATM), and incurs a QoS environment that interfaces end applications with underlying QoS provisioning services and manages application QoS through monitoring real-time network performances. The QoSME system model is deployed with its system components including QoSockets (the socket-like runtime environment), QoS MIBs (Management Information Bases), SNMP (Simple Network Management Protocol) agents and APIs (Application Programming Interfaces). Two typical QoS-demanding multimedia applications built using QoSME are introduced. Finally, some conclusions and future works are presented.

KEYWORDS:
QoS, quality of service, Intserv, RSVP, ATM, network management, distributed multimedia, QoSME, QoSockets, SNMP, MIB

1. INTRODUCTION

As the largest interconnected network around the world, the Internet provides a number of essential communication services such as WWW, email and file transfer (FTP) in our daily life. So far, the Internet has been evolved far beyond our past predictions. However, various mission-critical applications such as multimedia teleconferences, and e-commerce applications such as on-line hotel or ticket reservation and shopping are already surfing over the Internet and demanding that the Internet service providers provide QoS guarantees.

These applications need to acquire their requested service levels and qualities, varying largely one from another. For example, an IP telephony application requires voice signals arriving within a tolerated delay variance (jitter); a video player requires a bandwidth guarantee to convey its images smoothly; a real-time monitor requires a strictly assured delay of communication; and an online shopping service requires all data secured without loss.

Usually, throughput, delay, jitter and reliability (loss) are major concerns of applications' requirements on network performances. The Internet QoS technology is to help application request and meet these network requirements. First, those application QoS requirements should be mapped to the underlying QoS provisioning services such as Intserv (Integrated Services [10]), Diffserv (Differentiated Services [4]) and ATM [6]. Second, network performances behaved by the underlying services need to be monitored to ensure and adapt the application QoS.

In this paper, QoSME (Quality of Service Management Environment) is presented to meet the above challenges. It offers an open solution for delivering QoS to end applications and managing application QoS over network systems. QoSME interfaces applications with appropriate QoS provisioning services and applications request their QoS in terms of QoS characterization. Particularly, it benefits the legacy Internet applications that are based on the well-known Internet technologies. On the other hand, using SNMP [15] agents to access QoS MIBs, QoSME enables applications to monitor real-time network performances for QoS management.

This paper is organized as follows. Section 2 gives a brief review of current QoS-related research works.

* This paper was authored by Phil Wang when he was a research scientist in Computer Science Dept., Columbia University in 1998-2000.

QoSME architecture is introduced in Section 3, and followed by its QoS characterization, provisioning and management. Section 4 describes the QoSME model and its system components such as QoSockets, QoS MIBs and SNMP agents, and APIs. Two multimedia applications using QoSME, which are NetVideo (an Internet Video tool) and DIRM (a resource management system), are introduced in Section 5. Finally, Section 6 summarizes and presents some future works.

2. RELATED WORKS

A large amount of research efforts has been contributed to the QoS investigation, here briefs about some present-day typical projects.

Diffserv (Differentiated Services) that was proposed by Internet Engineering Task Force (IETF) [3][4] is a packet-based priority service that provides several types of premium or assured services to meet differentiated user requirements of network services. According to the SLA (Service Level Agreement), a packet is classified with a given service priority before it is forwarded into a Diffserv network. The DSFILED (Diffserv Field) [3] of the IP header is used to mark a packet, i.e., setting the TOS byte of an IPV4 packet.

"End System QoS Framework" of Washington University [26] is to provide QoS requirements with the end-system for networked multimedia applications and divided into four QoS components: specification, mapping, enforcement and protocol implementation. Important system resources such as CPU, memory and network are considered to meet the QoS requirements for each end-to-end application session.

EPIQ done by University of Illinois at Urbana-Champaign [5] is developing a meta-computing framework, by which end-to-end QoS and resource management can be customized and integrated to provide guaranteed services with negotiated quality to time-critical applications. It is working on an open run-time environment of real-time application, end-to-end QoS characterization and communication protocol design.

"Heidelberg QoS Model" from IBM European Networking Center [8][9] provides QoS guarantees in end-systems and network, and is implemented with a continuous media transport system HeiTS/TP on the top of ST-II (network layer)[27]. HeiRAT (Heidelberg Resource Administration Technique) is used to offer a QoS management scheme of end-to-end guarantees for HeiTS/TP and includes QoS negotiation, computation and admission control from the data link layer to the application layer. Two types of QoS are Guaranteed (at maximum demand) and Statistical (slightly overbooked), and have been experimented on the Token Ring network.

Intserv (Integrated Services) proposed by IETF [10~14] is a flow-based resource reservation service, which employs guaranteed and controlled load services to support end-end mission-critical services such as real-time service. Intserv comes with the resource reservation signaling protocol RSVP (ReSerVation Protocol) [11].

OMEGA of University of Pennsylvania [16] has done an interdisciplinary research work between application QoS requirements and available local and global resources. Two new protocols are proposed for resource reservation and management, RTNP (Real Time Network Protocol) at network layer and RTAP (Real Time Application Protocol) at application layer. A QoS brokerage model is also proposed for resource management including QoS translation and negotiation in the ATM network.

"OSI QoS Framework" proposed by ISO [17] is primary on quality of service support for OSI communication. It defines a set of terminology and concepts for QoS, and provides a model covering objects related to QoS in OSI standards. This framework consists of two types of management entities: layer specific and system-wide, to meet the QoS requirements by monitoring, maintaining and controlling end-to-end QoS.

QoSME from the DCC Group of Columbia University [1][2] is targeted to provide a generic architecture for application QoS demanding and management. It interfaces application with the underlying QoS provider to request QoS and monitors real-time network performances to assure application QoS.

QoS-A (Quality of Service Architecture) proposed by Lancaster University [18][19] is a multi-layer and multi-plane architecture of services and mechanisms for QoS management and control of continuous media flows in multi-service networks. Two important layers are the application platform layer that provides multimedia communications and QoS specification in an object-based environment, and the orchestration layer provides jitter correction and multimedia synchronization services across multiple related application flows over the transport layer. Three planes are the protocol plane which consists of distinct user and control sub-planes for different QoS requirements of control and data, the maintenance plane which contains a number of layer specific QoS managers and

the flow management plane which is responsible for flow establishment, QoS mapping and scaling.

QUASAR (QUAlity Specification and Adaptive Resource) from OGI [23][24] is working on QOS specification for both adaptive and reservation-based resource management, and concentrating on distributed multimedia systems that dynamically adjust to variability in available resources.

QuO (Quality of service Object) proposed by BBN [25] is working on the QoS requirements of distributed multimedia applications with a CORBA dynamic environment through a resource management and availability model. This model is implemented at the CORBA layer, exploits the underlying QoS of the communication layer and allows an application to know of the actual QoS and adapt to changing conditions. This project is integrated with QoSME.

TENET done by University of California at Berkeley [29] is conducted with an experimental wide area ATM network. It sets up a family of protocols in its architecture and provides deterministic and statistical guarantees for hard real-time and continuous media flows. For example, RCAP (Real-time Channel Administration Protocol) spans the transport and network layers for resource reservation and flow setup; RTIP (Real Time Internet Protocol) offers generic connection establishment, resource reservation and signaling functions for the rest of the protocol family; CMTP (Continuous Media Transport Protocol) supports communication over continuous media.

XRM (eXtended integrated Reference Model) from the COMET Group of Columbia University [22] develops a real-time modeling framework for control and management of multimedia telecommunication in ATM networks and end-systems populated with multimedia devices. It is divided into five distinct functional planes: network and systems management, resource control, data abstraction and management, connection management and binding and use information transport and computing. A database "telebase" is developed to implement data sharing among all the XRM planes.

In summary, the above QoS research works can be divided into three categories: (1) underlying QoS provisioning mechanisms, e.g., ATM, Diffserv and Intserv; (2) QoS control and management of network communication, e.g., QoS-A, TENET and QoSME; (3) End application QoS middleware or environment, e.g., QoSME, OMEGA and EPIQ. In addition, some of them have worked out some fundamental frameworks spanning the above three categories, e.g., OSI and XRM.

Clearly, the end applications requiring QoS do not want to handle the complexity of directly dealing with underlying QoS provisioning services (in the first category). Instead, they need a user-friendly QoS environment to demand and manage their service qualities. QoSME is such an environment described in later sections.

3. QoSME ARCHITECTURE

QoSME is originated from the QuAL (Quality of Service Assurance Language) project [2] of Columbia University. It serves end application QoS by providing a comprehensive architecture (Fig. 3.1) open to applications and network systems. Rather than proposing new protocols, QoSME builds a middleware between applications and the underlying resource providers, which employs current Internet technologies (protocols and standards) of network transmission mechanisms, services and management.

3.1 Overall Structure

QoSME integrates the current Internet protocols and standards that are commonly used in network communications. TCP/IP is the main networking protocol suite of QoSME, and ATM is also supported. SNMP [15] is used to offer MIB-based QoS management. Network protocols, TCP, UDP, ST-II [27] and ATM [6], are available in QoSME. Furthermore, TCP and UDP are provided to support QoS requirements using the reservation protocol RSVP, and called as *R-TCP* and *R-UDP* respectively.

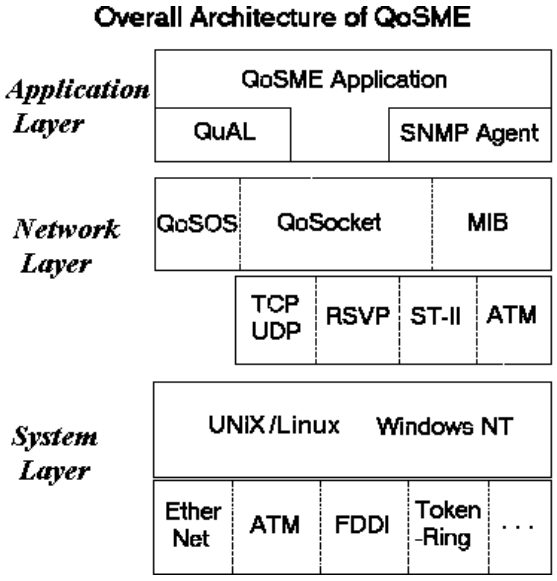

Fig 3.1 QOSME Architecture

At the *network layer* in Fig. 4.1, QoSockets constructs the run-time QoS environment for QoSME applications that use the network protocols (e.g., TCP, UDP, ST-II and ATM). QoSockets establishes connections to ATM [6] or RSVP [11] for QoS reservation and creates MIBs for QoS management. QoSOS is the interface part to interface QoSME with heterogeneous operating systems so as to use those native resources (e.g., threads and signals) from the *system layer*.

The *application layer* consists of end applications, SNMP agents and QuAL. QoSME applications may use QoSME API to communicate with QoSockets, or use QuAL to describe their QoS requirements more conveniently in terms of a friendly macro language that helps users understand detail QoS specifications. An SNMP agent is provided to access QoS MIBs from a remote SNMP-compliant network manager.

3.2 QoS Characterization

The main types of QoS attributes in QoSME are *throughput*, *delay (and jitter)*, and *reliability*. In addition, QoSME introduces the *coerced flag*, to coerce compatible QoS requirements of both senders and receivers of a traffic stream.

- **Throughput**

Four parameters are defined to represent the network throughput.

- ♦ *min_rate*: Lower bound of transmission rate;
- ♦ *max_rate*: Upper bound of transmission rate;
- ♦ *peak_rate*: Peak transmission rate;
- ♦ *size*: Maximal size of transmitted messages.

Each rate is number of messages conveyed per second. The throughput is the product of the rate (*min_rate*, *max_rate*, or *peak_rate*) multiplied by the message *size* (bytes). For the *i*th traffic stream, its throughput is computed as (in bytes/s):

$$\textit{Minimal}: \quad t_m^i = \text{min\_rate}^i \times \text{size}^i \qquad (1a)$$

$$\textit{Maximal}: \quad t_M^i = \text{max\_rate}^i \times \text{size}^i \qquad (1b)$$

$$\textit{Peak}: \quad t_p^i = \text{peak\_rate}^i \times \text{size}^i \qquad (1c)$$

- **Delay & Jitter**

Four parameters are related to the transmission delay and jitter.

- ♦ *min_delay*: Lower bound of transmission delay;
- ♦ *max_delay*: Upper bound of transmission delay;
- ♦ *int_delay*: Maximal time elapsing between two received messages;
- ♦ *jitter*: Maximal delay variance of two consecutive messages

These parameters are metered in milliseconds.

- **Reliability**

The reliability uses three major parameters.

- ♦ *loss*: Percentage of messages lost;
- ♦ *rec_time*: Maximal time elapsed for recovering a disrupted connection;
- ♦ *permt*: Permutable flag indicating if messages can be delivered out of order.

QoSME also provides other parameters (e.g., connection failures) used for monitoring network reliability.

- **Coerced flags**

QoSME allows both senders and receivers of a stream to define their own QoS parameters. Sometimes, the QoS parameters at each end conflict with each other and need to be coerced (downgraded) to a commonly accepted level. Coerced flags are therefore used to indicate which parameters should be coerced. If no coercion is requested, both senders and receivers use their own parameters to request QoS, which may cause resource allocation failure in case of incompatibility.

For example, suppose two ends of a traffic stream want to coerce their peak rates (by setting *coerce_peak_rate* = True), and the rates of the sender and the receiver are 64 and 60 KBps (kilobytes per second) respectively. QoSME internally coerces them to the minimal common rate of 60 KBps, and notifies the new rate to both sender and receiver. The sender effectively downgrades its peak rate to 60 KBps as a result.

3.3 QoS Provision

QoSME provides application QoS by requesting resource allocations of the underlying service providers such as Intserv, Diffserv, and ATM. The current implementation uses ATM and Intserv/RSVP.

- **Types of QoS**
  - ♦ ***Guaranteed:*** guaranteed service
  - ♦ ***Controlled Load:*** controlled-load service

The above two types of QoS are inherited from Intserv/RSVP, while QoSME also has more types of QoS from ATM services.

- **Modes of QoS**
  - ♦ ***Hard:*** QoS provisioning by ATM, in the ATM switch network
  - ♦ ***Soft:*** QoS provisioning by Intserv/RSVP, in the Internet

When an application assigns its QoS requirements, QoSME maps these requirements to QoS parameters of RSVP or ATM. If a resource reservation is made, QoSME associates the network connections of this application with this reservation.

When a resource reservation fails, QoSME will return a message to the application. Combining this

message with QoS MIB, an application can know the current status of available network resource, and then adapt its QoS requirements and make a new request.

3.4 QoS Management

Once an application establishes a QoSME connection (via QoSockets), QoSME management starts. It collects various data from each communication connection, including QoS specifications, connection times, transmission rates and delays, and calculates QoS parameter to detect any QoS violation. All these data are stored into SNMP MIBs (Management Information Base), and can be accessible from a QoSME application, or a remote SNMP manager using an SNMP agent.

With MIBs, QoSME is able to control and adapt the QoS requirements for particular applications. The SNMP agent provided by QoSME can access (read and write) these MIBs, so that an SNMP network manager is able to monitor network status and to adapt QoS requirements at a remote terminal.

4. QoSME SYSTEM and COMPONENTS

In this section, QoSME system and its components are introduced that implement the QoSME architecture described in last section. Currently, QoSME has been available on several OS platforms including SunOS 4.1.x, Solaris 2.5 and Linux 2.0 and 2.2.

4.1 System Model

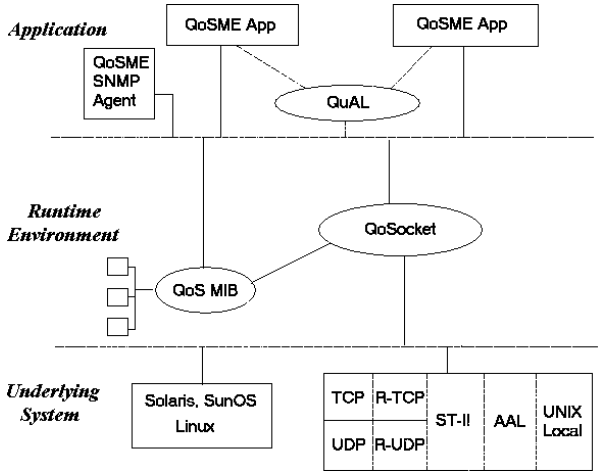

Fig 4.1 QoSME System Model

QoSME has a set of components that compose an open QoS environment for user-end and network management applications. Each component plays an indispensable role in QoS delivery and management, as shown in Fig. 4.1.

Similar to the QoSME architecture, QoSME system is modeled into three sub-systems, and those components are placed in the first two sub-systems: *application* and *runtime environment.* The third sub-system *underlying system*, offers QoSME available resources from operating systems and network systems.

A QoSME application defines its QoS requirements using either QoSME APIs or QuAL, and obtains QoS guarantees via QoSockets at the *runtime environment*. An SNMP V2-compliant agent is also offered at the *application* layer to access QoS MIBs generated by QoSockets in real time.

At the *runtime environment*, QoSockets provides overall QoS functions to QoSME applications and communicates the *underlying system* for QoS guarantees. Another important part is QoS MIB that collects data about QoS requirements and network performances for QoS management. Third, QoSOS that is not depicted in Fig. 4.1 deals with platform-dependent associations so that QoSME applications can be platform-independent.

4.2 QoSockets*: Runtime environment*

QoSockets is the core part and brings QoS to the Berkeley sockets that are widely used nowadays by network applications. It compiles the application specifications into respective transport protocols and mechanisms, when possible. Protocols supported by QoSockets include TCP, UDP, RSVP, ST-II, and ATM. QoSockets also generates instrumentation to monitor the QoS delivered to the application and constructs appropriate QoS Management Information Bases (MIBs) to access this instrumentation.

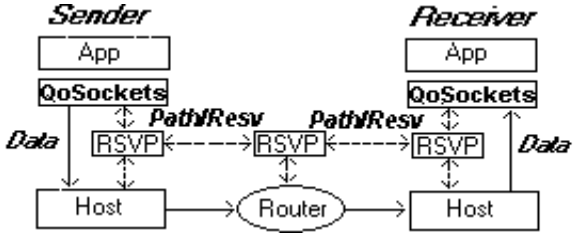

Fig. 4.2 QoSME applications with Intserv/RSVP

Figure 4.2 shows how QoSockets operates above an underlying RSVP service. QoSockets shelters applications from the complexity of the interface details of the specific QoS provisioning mechanism. One could use the same QoSockets specification for RSVP and ATM.

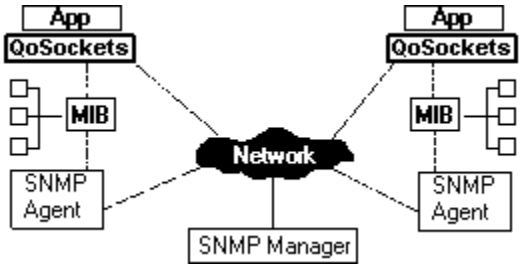

Fig. 4.3 QoS Management in QoSME

Figure 4.3 shows how QoSockets works with QoS MIBs. When an application establishes a QoSockets connection, QoSockets starts collecting the requirement and performance data related to the connection, including QoS specifications, connection duration, transmission rates, delays, etc. It also detects QoS violation by comparing the QoS requirements with real collected performance statistics.

4.3 QoSMIB & Agent: *QoS Management*

QoS management is another important part in QoSME, as shown in Fig 4.3. To do this, QoS MIBs are created automatically by QoSockets, and the SNMP agent is offered as a utility of QoSME to access these MIBs.

QoS MIBs contain QoS statistics per application (Fig 4.4), and integrates application level QoS management into standard network management frameworks, such as SNMP.

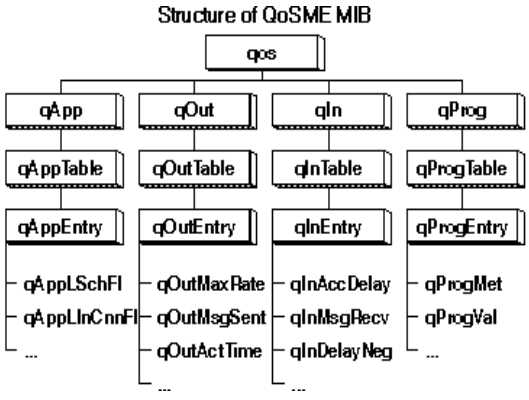

Fig. 4.4 QoS MIBs in QoSME

The data stored in the QoS MIBs are accessed inside the application using the QoSME APIs and from remote network managers using the QoSME SNMP agents. Thus, QoSockets allows applications to control and adapt to QoS performance by using application exception handling procedures (locally) or by requiring assistance from network managers (remotely).

The SNMP agent offered by QoSME is SNMP-V2 compliant, and accesses QoS MIB access under control of a remote SNMP-based network managers. That is, a network administrator, working at a distant site, can control and monitor the QoS of QoSME applications.

4.4 APIs: *Application Programming Interfaces*

QoSME provides APIs using a C library similar to the interfaces of Berkeley sockets. Those APIs include the following abstractions.

- Connection Establishment: initialize and establish connections and reserve the application QoS requirements specified.
- Selection of Protocols: select a specific transport protocol and bind a socket address to a QoSockets connection endpoint.
- Monitoring QoS delivery: monitor the QoS performance of applications communications, and store the sampled performance statistics into QoS MIBs.
- QoS MIB Access: access values of QoS MIBs using SNMP-based interfaces.

Alternatively, applications can use QuAL to specify QoS requirements that are compiled into run time components that map and monitor the actual QoS.

5. QoSME Applications

Recently, the QoS performances of two multimedia applications Using QoSME have been investigated in [30]. The first is NetVideo [29], a UDP-based real-time video tool; and the other is DIRM (Dynamic Integrated Resource Management) [25], a TCP-based resource management system for CORBA (Common Object Request Broker Architecture [31])-based applications. Both applications were originally developed using sockets and have been easily modified to use QoSME so that they take advantage of its powerful QoS infrastructure.

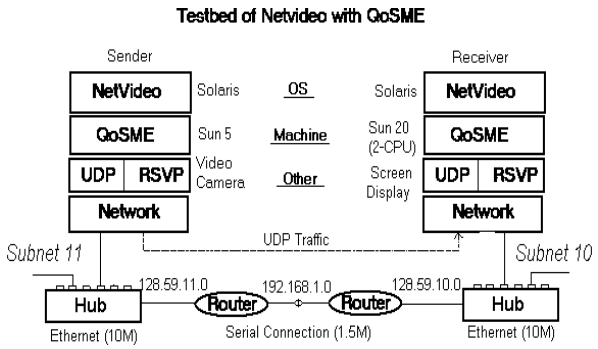

Fig. 5.1 Net Video with QoSME

- **Application 1: *Net Video***

NetVideo is a multimedia tool for the Internet that captures, transfers, and receives real-time video pictures using *UDP*, in either unicast or multicast. The proposed version uses the QoSME API to request QoS for UDP transport (*R-UDP*), and is tested in a test bed in which two different LANs are connected through a serial link (a bandwidth bottleneck) between two CISCO routers (shown in Fig. 5.1). Two programs run under two Solaris machines, one acts as a video sender and takes pictures with video camera, and the other acts as a video receiver and displays received pictures onto screen.

- **Application 2: *DIRM***

DIRM is a resource management environment and provides a set of high-level APIs that assists object-oriented applications to acquire QoS at runtime. Upon application request, DIRM allocates and manages network resources (e.g., bandwidth) dynamically using the IIOPGW resource manager that bridges two ORBs (Object Request Brokers). IIOPGW uses the QoSME APIs for resource allocations to assure QoS of its TCP transport (R-TCP).

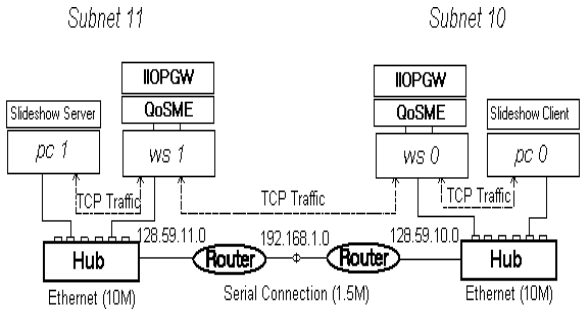

Fig. 5.2 DIRM with QoSME

Slideshow is a CORBA-based Java application, and consists of two parts: a remote server and a local client. The server manages a repository of pictures, and the client requests those pictures and displays then onto screen. Since two IIOPGW programs request network resource reservations for all traffic streams between them, the communications between Slideshow server and client are provided with QoS.

By experiments, these two applications show that they (UDP and TCP applications) benefit from QoSME (e.g., resource reservations) and experience significant improvement in their performances. QoSME generates very important MIBs of real-time network performances, which is used for QoS monitoring. However, TCP applications require bi-directional traffic guarantees which are not directly provided by the QoS provisioning mechanisms.

6. SUMMARY

This paper presents QoSME as a potential solution for the Internet application QoS management and guarantees. It provides actually a middleware between applications and underlying resource systems, which is built over prevailing network protocols. Moreover, QoSME (using QoSockets) easily brings QoS to those applications using the Berkeley sockets. Regarding the QoS management, QoSME generates QoS MIBs and uses SNMP agents to access real-time system parameter and network performance data.

Future works include extending QoSME to support flexible QoS adaptation and negotiation. End applications usually have to adapt their QoS requirements to the varying network conditions. On the other hand, flexible QoS adaptation and negotiation enable applications to use network resources efficiently.

Another topic is to enable applications obtaining appropriate network service guarantees using uniform interfaces, regardless what system platforms, provisioning services and programming languages are. It also benefits legacy applications to acquire QoS even they do not include those QoS APIs. Some earlier work has been published in [32].

ACKNOWLEDGEMENTS

The authors sincerely thank John Zinky, Frank Bronzo at GTE/BBN Technologies for their valuable contribution in the DIRM project.

REFERNCES

[1] Florissi, P. and Yemini, Y., “Management of Application Quality of Service”, Technical Report DCC-03-94, Columbia University, 1994

[2] Florissi, P., “QuAL: Quality Assurance Language”, Ph.D. Thesis, Columbia University, 1996

[3] Nichols, K., Blake, S., Baker, F. and Black, D., “Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers”, Internet RFC 2474, Dec. 1998

[4] Blake, S., Black D., Carlson, M. Davies, E., Wang, Z. and Weiss, W., “An Architecture for Differentiated Services”, Internet RFC 2475, Dec. 1998

[5] Hull, D., Shankar, A., Nahrstedt K., and Liu J., “An end-to-end QoS model and management architecture”, Proc. of IEEE Workshop on Middleware for Distributed Real-time Systems and Services, San Francisco, December 1997

[6] ATM Forum, “ATM User-Network Interface Specification”, Version 3.1, 1994

[7] Yemini, S., Goldszmidt, G., Stoyenko, A. and Wei, Y., “Convert: A High Level Language Approach to Heterogeneous Distributed Systems”, Proceedings of the 9th International Conference on Distributed Computing Systems, Newport Beach, CA, June 1989

[8] Vogt, C., Wolf, L., Herrtwich, R., and Wittig, H., "HeiRAT -- Quality-of- Service Management for Distributed Multimedia Systems", Special Issue on QoS Systems of ACM Multimedia Systems Journal, Vol. 6, No. 3, May 1998

[9] Delgrossi, L., Halstrinck, C., Hehmann, D.B., Herrtwich, R.G., Krone, J., Sandvoss, C and Vogt, C., “Media Scaling for Audiovisual Communication with the Heidelberg Transport System”, Proc. ACM Multimedia’93 Anaheim, August, 1993

[10] Braden, R., Clark, D. and Shenker, S., “Integrated Services in the Internet Architecture: Overview”, Internet RFC 1633, June 1994

[11] Zhang, L., Berson, S., Herzog, S. and Jamin, S., “Resource ReSerVation Protocol (RSVP) – Version 1 Function Specification”, Internet RFC-2205, 1997

[12] Wroclawski, J., “The Use of RSVP with IETF Integrated Services”, Internet RFC 2210, Sept., 1997

[13] Wroclawski, J., “Specification of the Controlled-Load Network Element Service”, Internet RFC 2211, Sept., 1997

[14] Shenker, S., Patridge, C. and Guerin, R., “Specification of Guaranteed Quality of Service”, Internet RFC 2212, Sept., 1997

[15] SNMPv2 Working Group, “Protocol Operations for Versions 2 of the Simple Network Management Protocol (SNMPv2)”, Internet RFC-1905, January 1990

[16] Nahrstedt, K. and Smith, J., “The QoS Broker”, IEEE Multimedia, Spring 1995

[17] ISO, “ Quality of Service Framework”, ISO/IEC JTC 1/SC21/WG1 N9680, International Standard Organization, UK, 1995

[18] Colulson, G., Campbell, A. and Robin, P., “Design of a QoS Controlled ATM Based Communication System in Chorus”, IEEE Journal of Selected Areas in Communications (JSAC), Special Issue on ATM LANs: Implementation and Experiences with Emerging Technology, May 1995

[19] Campbell, A., Coulson, G., Garcia, F., Hutchinson, D. and Leopold, H., “Integrated Quality of Service for Multimedia Communications”, Proc. IEEE INFOCOM’93, San Francisco, USA, April 1993

[20] Lazar, A.A., Bhonsle S., Lim, K.S., “A Binding Architecture for Multimedia Networks”, Proceedings of COST-237 Conference Transport and Teleservices, Vienna, Austria, 1994

[21] Lazar, A., “Challenges in Multimedia Networking”, Proc. International Hi-Tech Forum, Osaka, Japan, February 1994

[22] Lazar, A., Ngoh, L. and Sahai, A., “Multimedia Networking Abstraction with Quality of Services Guarantees”, Proc. SPIE Conference on Multimedia Computing and Networking, San Jose, February 1995

[23] Cen, S., Pu, C. and Walpole, J., "Flow and Congestion Control for Internet Streaming Applications", Proceedings of Multimedia Computing and Networking (MMCN98), 1998

[24] Staehli, R., Walpole, J. and Maier, D., "Quality of Service Specification for Multimedia Presentations", Multimedia Systems, Vol. 3, No. 5/6, November 1995

[25] Zinky, J., Bakken, D. and Schantz R., “Architectural Support for Quality of Service for CORBA Objects”, Theory and Practice of Object Systems, January 1997.

[26] Gopalakrishna, G. and Parulkar, G., “A Real-time Upcall Facility for Protocol Processing with QoS Guarantees”, 15th ACM Symposium on Operating Systems Principles, December 1995

[27] Dekgrossi, L. and Berger, L., “Internet Stream Protocol Version 2(ST2) – Protocol Specification – Version ST2+”, Internet RFC-1819, 1995

[28] Ferrari, D., “The Tenet Experience and the Design of Protocols for Integrated Services internetworks”, Multimedia Systems Journal, 1996

[29] Xerox Corporation, NetVideo, Version 3.3, 1994

[30] Wang, P.Y., Yemini, Y., Florissi, D., Zinky, J. and Florissi P., “Experimental QoS Performances of Multimedia Applications”, IEEE Infocom 2000, Tel Aviv, Israeli, March 2000

[31] Object Management Group, “The Common Object Request Broker: Architecture and Specification”, Rev. 2.2, Feb. 1998

[32] Wang, P.Y., Yemini, Y., Florissi, D., and Zinky, J., “A Distributed Resource Controller for QoS Applications”, IEEE/IFIP NOMS 2000, Hawaii, USA, April 2000

[33] Wang, P., “Linux Port Of RSVP r4.2a3”, http://www.cs.columbia.edu/~yhwang/ftp/qos/rsvp, Aug. 1998