A Resource Management Architecture over Differentiated Services Domains for Guarantee of Bandwidth, Delay and Jitter

Koichi Tajima, Manzoor Hashmani, and Mikio Yoshida
NIPPON STEEL Information & Communication Systems Inc.,
E-mail: tajima, manzoor, yoshida@osk.enicom.co.jp

Abstract **— Network-based applications like video and voice over IP (VoIP), require guarantee of end-to-end QoS. The guarantee of end-to-end QoS requires an efficient resource management mechanism that can reserve / control resources like *bandwidth, delay* and *jitter* according to a policy for *immediate* and *future* resource utilization. Only a few resource management architectures called Resource Manager (RM) have been implemented till today but all of them manage only bandwidth in a network. We propose ENICOM's[1] Policy based Resource Manager (PREM) to address the above issues. PREM provides a set of useful API to *make, cancel* and *change* resource reservations and it can manage not only bandwidth but also delay and jitter which is possible mainly due to a highly efficient algorithm called Resource Tracker (RTK). RTK tracks resource reservation and availability in time using timeslices. In this paper, we describe PREM and it's various components, which enable it to provide guarantee of end-to-end QoS in terms of bandwidth, delay and jitter.**

I. Introduction

Emerging performance-oriented applications[1], require guarantee of end-to-end high quality of service (QoS) from the network. We have developed and deployed a prototype system as a Japanese national experiment, which provides QoS enabled network services on top of a high speed network which partly includes JGN (Japanese Gigabit Network) with support of MITI (Ministry of International Trade and Industry) investment. The activity is aimed at contents business, which requires strict guarantee of not only bandwidth but also delay and jitter. Through the experience of the deployment, we recognized that guarantee of bandwidth alone is not sufficient for these promising contents business applications. They need assurance of delay and delay bounds (jitter) because of on-line editing.

To provide guarantee of QoS especially for delay and jitter, the management (reservation, allocation and policing) of delay is necessary. For example, in case of real time applications like video conferencing and voice over IP (VoIP), users need guarantee of bandwidth and jitter. A large jitter may not be acceptable to VoIP applications and may distort communication between users. On the other hand in case of video content transfer guarantee of bandwidth with some level of jitter may be acceptable to users. Hence performance-oriented applications may demand guarantee of one or all of; bandwidth, delay, and jitter. Though delay and jitter can be kept low if bandwidth usage is strictly policed, a guarantee on delay and jitter cannot be provided by simply exercising bandwidth management.

To provide guarantee of delay and jitter, strict admission control and delay calculation is necessary. To the best of our knowledge, all resource management architectures proposed till today only attempt to manage bandwidth and call their implementation a bandwidth broker (BB). Out of the six implementations of BB proposed till today, only one is able to perform bandwidth reservation for future resource utilization. Moreover, many applications and network service providers can adjust their parameters based on the current traffic load. Therefore, a resource manager should be able to provide traffic analysis to its users too.

In this paper, we propose ENICOM's Policy based Resource Manager (PREM) to provide the features we found to be necessary during our experience of QoS enabled network. PREM was also developed because none of the present implementations provides all of these features particularly these do not provide guarantee of delay.

PREM like other existing implementations of resource managers performs distributed management of network resources. Unlike other implementations, however, PREM manages not only bandwidth but also delay according to a policy[9], which may be provided by the network administrator. The basic concept is;

1. To divide a network into manageable DiffServ domains[4].
2. Each domain is managed by a PREM in accordance with SLA (Service Level Agreement) or policy.
3. PREM performs strict admission control in terms of bandwidth reservation and delay. It keeps a record of bandwidth reservation against time and corresponding delay bounds for each flow.

PREM provides the following traffic analysis graphs for each link per service class or application type.

1. Total amount of bandwidth for which reservation requests were rejected, and
2. Total amount of unused bandwidth for which reservation was done.

Besides describing PREM concept and architecture, we report on a prototype implementation. In the end of this paper we describe those desirable features that are not included in PREM's current implementation and will be addressed in near future.

II. Target Issues

As stated in section I, to provide guarantee of end-to-end QoS, requires resource management by a resource manager (RM). There are a few designs and implementations available but lack some necessary features, for example, delay management etc. We describe these lacking features below and give reasons why these are necessary to provide

[1] NIPPON STEEL Information & Communication Systems Inc.

guarantee of end-to-end QoS. We target these issues and provide their solution in PREM.

1) *Delay Management:* Current implementations of resource manager manage only bandwidth as network resource and, therefore, can provide guarantee of only bandwidth. Guarantee of only bandwidth may not result in satisfactory performance for many applications like video conferencing and telephony over IP networks. These applications are particularly sensitive to variation in delay of communication that is jitter. The management and guarantee of bandwidth may not necessarily reduce jitter to an acceptable level because of the worst-case scenarios of head of line (HOL) blocking due to statistical fluctuations. The worst-case scenario of HOL is more likely to occur in the core routers and therefore an assurance jitter cannot be provided (section V.D.2).

2) *Future Reservation:* To achieve efficient use of the network resources, future reservation of network resources is necessary. For example a video contents provider may have an SLA with a TV station to provide some content on every Wednesday from 18:00 to 18:30. To realize this SLA a resource manager must be able to perform future reservation of network resources. The BB architecture of Qbone and most of the implementations based on it do not provide this feature (section V.D.1).

3) *Inter-Domain Reservation:* In DiffServ architecture a network is divided into DS (DiffServ) domains, therefore, for any system to provide guarantee of end-to-end QoS, needs to make resource reservation in all domains between a source and a destination. Most of the available implementations of resource manager provide only intra-domain resource reservation. In PREM, we provide support to inter-domain reservation so that guarantee of end-to-end reservation can be provided (section V.F).

4) *Traffic Analysis:* One purpose of a resource manager is to help design future networks or redesign current networks so that its users or user applications can get a satisfactory performance in terms of guarantee of end-to-end QoS. Moreover, many applications can adjust their rate of transmission depending on the current traffic load in the network. The traffic analysis of reservation requests and of actual traffic can be used for this purpose. Therefore, we have included a traffic analysis module in PREM, which addresses this issue (section V.I).

5) *Heterogeneous Router Support:* In networks, many different types of routers co-exist with different QoS capabilities. The idealistic situation may be to have all DiffServ compliant routers in a network with same QoS features. However, we think it is a valid assumption that a resource manager may have to tackle routers, which are not DiffServ compliant. Therefore, a resource manager may provide support to heterogeneous router support. PREM provides this support using RT (Router Translator) module (section V.J).

PREM addresses above the stated issues and provides many other features which are essential but are not available in most other designs and implementations of resource managers.

III. Resource Manager

The basic components of the system, which provides guarantee of end-to-end QoS, are;

1. A QoS architecture (DiffServ)
2. Resource Manager

Whereas, the main functions of a resource manager are;

1. To enforce rules (policy) of each class of service.
2. Admission control to protect and manage resources.

We now briefly explain these components and functions before further discussion.

A. *DiffServ Architecture*

The DiffServ architecture [3][4][5][7] requires that the network may be divided into manageable domains called DS domains as shown in Figure 1. This architecture achieves scalability by implementing complex classification and conditioning functions only at network boundary (ingress and egress) nodes, and by applying per-hop behaviors (PHBs) to aggregates of traffic in the core routers. Per-application flow or per-customer forwarding state is maintained only at domain boundary nodes and need not be maintained within the core of the network. Per-hop behaviors are defined to permit a reasonably granular means of allocating buffer and bandwidth resources at each node among competing traffic streams using a policy.

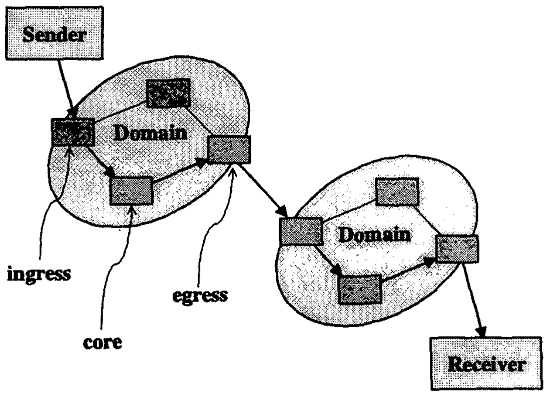


Figure 1: DiffServ Architecture

B. *Policy Architecture*

Since, there are varying circumstances in which traffic owners are entitled to the QoS services they request, there is a need for rules (policy or SLA), and a system to decide when and how to enforce these rules.

The framework provided by the IETF (Figure 2) comprises of a policy repository, policy consumers and policy targets. The Policy Consumer or a Policy Decision Point (PDP) receives policy intended for a Policy Target (router) and processes the policy to allow the router to enforce the policy.

The whole system is called a policy system and is essential to provide guarantee of end-to-end QoS.

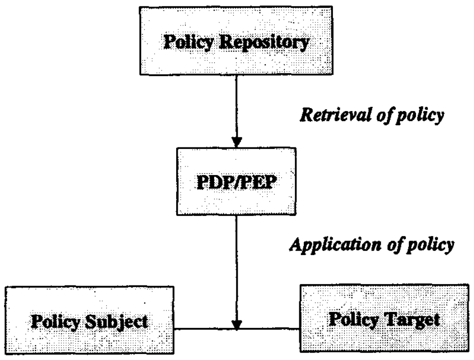


Figure 2: Policy Architecture

C. *Admission Control*

Admission Control is the main function of a resource manager. Admission control means to admit or reject a resource reservation request based on the resource availability and policy. To perform admission control, tracking of resource availability is necessary. There are two methods to track resource availability, namely;

1. Track resource availability using real time measurement of network traffic.
2. By keeping a record of reservations.

If only method 1 is used, resource reservation for future cannot be made. Therefore, method 2 is also necessary.

D. *Current Works*

Let us now summarize the discussion given in section III. To provide guarantee of end-to-end QoS in IP networks, the following are essential.

1. QoS architecture like RSVP[2] and/or DiffServ.
2. Policy enforcement mechanism.
3. A record of availability and allocation of network resources.

A lot of research has already been done in various forums like IETF etc., on QoS architectures like IntServ and DiffServ, which are now well defined. On the other hand the participants of the Internet2 QoS Working Group (Qbone) are trying to develop and implement new IP QoS technologies. The Qbone has decided to adopt the DiffServ approach due to its flexibility, scalability and per hop behavior. To achieve guarantee of end-to-end QoS, Qbone architecture defines a BB, which must possess the following five functions[10].

1. Can receive a bandwidth request from an Application (section V.D) or a router.
2. Should respond to requester after having set-up the QoS with the routers.
3. Should be able to reject over-booking of bandwidth.
4. Should be able to reconfigure routers.
5. Should handle the application terminating its use of the bandwidth and return the bandwidth to the pool.

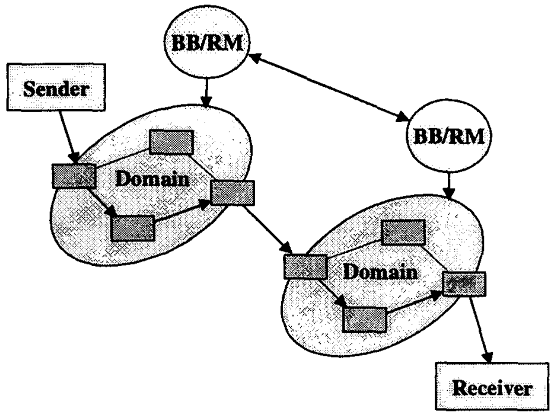


Figure 3: BB/RM Architecture

The BB defined by Qbone is a kind of resource manager. Six organizations participating in Qbone have produced implementations of BB till now. These are the only implementations available besides PREM described in this paper. The names of the organizations/projects are as follows

1. Globus architecture for reservation & allocation (GARA).
2. MCI/WorldCom
3. Merit Networks
4. Siemens
5. Telia
6. University of Kansas

IV. Our Proposal

A. *Analysis for Required Features*

We summarize our discussion till now and state the features/functions, which are necessary to provide guarantee of end-to-end QoS. The following table lists these functions. The second column states the status of support available in the designs and implementations of resource managers listed in section III.D.

functionality	support in existing RMs
Admission Control	Yes
Bandwidth Management	Not comprehensive
Delay Management	No
Policy Translation	Yes
Policy Conflict Resolution	No
Heterogeneous Router Support	Not comprehensive
Traffic Analysis	No

We now briefly describe each function and why it is necessary. As stated in the previous sections, without ***admission control***, guarantee of QoS, cannot be provided by a resource manager. Also there are applications that need guarantee of not only bandwidth but also delay and ***jitter***, therefore, a resource manager needs to perform bandwidth and delay management. The delay management is sufficient to enable a resource manager to provide guarantee of delay and jitter. The ***policy translation*** and ***policy conflict resolution*** are also essential for a resource manager to perform a smooth and optimal use of QoS enabled network. Moreover, in real life, a combination of heterogeneous QoS routers coexist, therefore, support for heterogeneous routers is also a necessary feature of resource managers. Lastly, a resource manager must also be able to analyze reservation information in such a way so as to help redesign a network to meet future needs of its users. To do this we think, ***traffic analysis*** must also be a necessary feature.

The resource managers implemented till today possess some of these functions but lack others. The features that we did not find in any implementation are delay management, policy conflict resolution, and traffic analysis. In section V, we present our design that has all of the above functions.

V. PREM

In this section, we present a full design and architecture of PREM, which solves the issues, described in section IV.A. First, we briefly explain the common operation. Next, we describe the main modules of our design, their importance and interaction between them. We then explain detailed reservation operation in which we describe how all these modules work together to provide guarantee of end-to-end QoS.

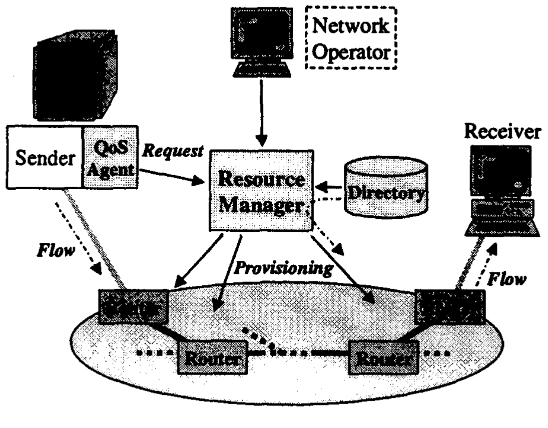


Figure 4: PREM Architecture

A. Common Operation

First of all, the static policy is stored in the repository by the network administrator via the resource manager. Based on the static policy, a sender can make a resource reservation request to the resource manager. The sender or its QoS agent can directly send this request to the resource manager. The resource manager returns a reservation handle if sufficient resources are available in the path from the sender to the receiver and corresponding QoS configuration is successfully done. After receiving the reservation handle, the sender can start sending data from the time specified in the reservation request.

B. PREM API

In this section, we briefly describe the PREM-API, which are used by the Applications to access the server functions.

Table 1: Main API, shows main PREM-API. The **doReservation** API is used to request resource reservation. A sender usually initiates the reservation request using path specification (**pathSpec**), traffic specification (**trafficSpec**) and time condition (**timeCondition**). The **pathSpec** contains path information between sender(s) and receiver(s). The **trafficSpec** specifies the details of resources to be reserved including service class (for example, EF or AF of DiffServ).

The **changeReservation** API can change the traffic specification or time condition of a previous reservation if sufficient resources are available. The **cancelReservation** API is used to cancel a previous reservation.

Table 1: Main API

name	inputs	return value	comment
doReservation	pathSpec trafficSpec timeCondition	reservation id	make resource reservation with given input parameters.
cancelReservation	reservationId	*none*	cancel existing reservation.
changeReservation	reservationId trafficSpec timeCondition	*none*	change properties of an existing reservation.
getTrafficSpec	pathSpec timeCondition	traffic speci-fication	estimate traffic specification for a given flow and a given time condition.
getTimeCondition	pathSpec trafficSpec	time condi-tion	estimate time condition for which requested traffic specification can be reserved (for a given flow).

C. Admission Controller

The Admission Controller (AC) is the module, which makes resource reservations in a network against policy and/or resource allocation requests (RAR) for immediate or future reservations. The admission controller performs admission control based on the information provided by the Resource Tracker (RTK) and the Policy Manager (PM). A RAR can be initiated by either a sender Application or a network device (for example a router). However, in the current design we consider RARs from Applications only. In the near future, we will enhance this module to accommodate RARs from routers (due to RSVP signaling).

D. Resource Tracker

The ability of PREM and similar architectures to provide a guarantee of end-to-end QoS depends on their resource tracking capability. Without resource tracking, admission control is not possible. In PREM we have a Resource Tracker (RTK) module to record allocation and availability of network resources. The RTK is responsible to maintain a record over two resources, namely, bandwidth and delay. Management of these two resources can be used to provide end-to-end QoS guarantee of not only bandwidth but also delay and jitter. For bandwidth management we have introduced a Bandwidth Calculator (BC) and for delay management, a Delay Calculator (DC) is introduced. Each is respectively described below.

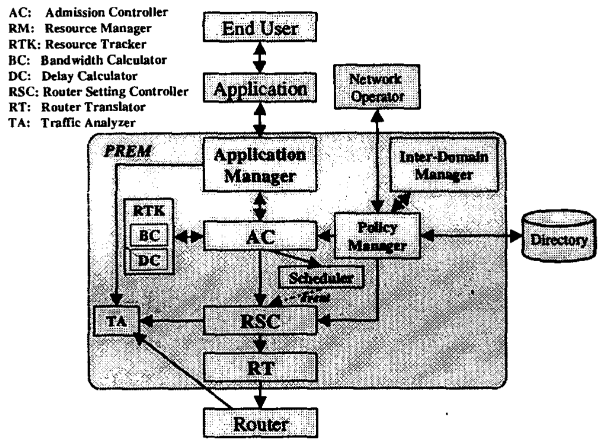


Figure 5: PREM Architecture

1) Bandwidth Calculator

BC is responsible to keep track of allocation of network resources (bandwidth). Keeping a record of allocation in timeslices does the tracking of bandwidth. The timeslices are created and destroyed dynamically, that is, in real time. When a new bandwidth reservation request from time t_1 to t_2 arrives at AC, it requests BC to find out if sufficient bandwidth is available in all time slices from t_1 to t_2 for all links between a sender and a receiver. If sufficient bandwidth is available, AC commits resources and requests BC of RTK to reserve bandwidth. If t_1 and/or t_2 do not coincide with the boundary timeslices, BC creates new timeslices and updates all others in between. Similarly, when current time crosses a timeslice or if a cancel reservation request arrives, BC may destroy some timeslices and update others. BC can create and update a large number of timeslices in parallel to ensure timely response to resource reservation requests.

The resource reservation for the immediate resource usage in general and for future resource usage in particular depends on BC. Note that the reservation for immediate resource usage can be done by traffic measurement but it cannot be done for future resource usage if reservation information is not maintained for future as well. Since, BC is able to maintain this information using timeslices of future, it is able to provide a guarantee of end-to-end QoS for future resource usage as well.

2) Delay Calculator

The Delay Calculator (DC) is much similar in operation to BC. At present we consider only the CBR like traffic for delay calculation because the properties of CBR like traffic are calculable. We do not consider bursty traffic for delay calculation because it is unpredictable. The DC calculates three kinds of delay for each microflow, namely, propagation delay, processing delay and queuing delay at each router. It then calculates total delay using these three quantities. The propagation delay and processing delay are constant and are not affected by traffic statistics. The queuing delay depends on the traffic pattern. The DC has the information of all the traffic entering at all nodes in domain, therefore, it can calculate average and maximum queuing delay observed by a microflow.

To account for the misbehaving users, we plan to get the queue size information from the heavily used nodes to check correctness of calculations. However, routers of some vendors do not provide this information, therefore, we are still refining this feedback mechanism.

The calculation of delay is used to provide a guarantee of end-to-end delay and jitter.

E. Policy Manager

As described earlier, a resource manager must be able to provide guarantee of QoS according to a policy. In PREM, policy is enforced by a Policy Manager (PM). The PM contains a rule-based engine to resolve policy conflicts when multiple conflicting policies exist. Network operators do not store policies in the repository directly. Rather, all policies are stored via PM. If a sender wants to reserve resources from this sender to a receiver, which exists in a domain other than that of this PM, then the PM performs the following two steps.

1. Perform policy check against the SLA of this sender.
2. Request Inter-Domain Manager to perform inter-domain resource reservation, if required.

1) Static Policy & Dynamic Policy

There are two types of policy; namely, static and dynamic. The static policy is the one, which changes relatively infrequently, for example, in the time scale of hours and is usually stored in a repository before being enforced. On the other hand the dynamic policy may change very frequently, for example, on the time scale of minutes and may come directly from the Application in real time. However, the dynamic policy may also be stored in the repository for the accounting purposes.

The PREM is able to handle both static and dynamic policy in the PM. Regarding static policy, the PM receives it from the network operator and after conflict resolution with already existing policies, it stores the new policy in the repository. On the other hand, the dynamic policy reaches PM via AM (Application Manager) and AC (Admission Controller).

F. Inter-Domain Manager

A resource reservation request may not be limited to only one domain rather it may require reservation of resources in other domains as well which are managed by separate PREMs. Therefore, each PREM has an Inter-Domain Manager (IDM) whose functions are given below.

1. Handle reservation requests from the adjacent domain(s).
2. Send reservation requests to the adjacent domain(s).

Regarding 1, when a resource reservation request is received from the adjacent domain, it is handled in this manner. First, it is checked against the SLA of the requesting domain to see if the request is acceptable. If in fact the request is acceptable, then if necessary, reservation request is again sent to the adjacent domain, which is determined by the destination host. If a positive response is received from that domain, then PM is contacted to commit resources. The PM may refuse the request if sufficient resources are not available. In that case a negative response and the reason are communicated to the requesting domain otherwise positive response is communicated.

Regarding 2, a resource reservation request is initiated by this IDM for the IDM of the PREM of adjacent domain. This is done on behalf of the PM, which determines if such negotiation is necessary.

G. Resource Policing

The guarantee of end-to-end QoS depends not only on the resource tracking but also corresponding resource policing. For example, if a commitment for 10 Mbps has been made by the PREM from a source ***SRC*** to a destination ***DST*** that traverses domains $\boldsymbol{D_1}$ and $\boldsymbol{D_2}$, strict policing must be performed at some or all locations (e.g. routers) between ***SRC*** and ***DST***. Since, for the present implementation we only consider DiffServ architecture, we perform policing at the ingress of each domain only. In PREM, router setting controller (RSC) performs QoS setting in ingress router of each domain to ensure proper resource policing.

To ensure scalability, DiffServ proposes a policing method, which is a combination of per microflow policing and per aggregate policing. At first ingress to a source, per microflow (determined by source address, source port, destination address, destination port and protocol) policing is performed but in the core routers of this domain policing is performed on only a class (EF and AF of DiffServ) based aggregated traffic. On the other hand on the boundaries of a domain, policing on a per-domain per-class based aggregated traffic is performed.

However, this mechanism of policing can still lead to a large number of microflows being handled in an ingress router. This may lead to a large unwanted overhead. On the other hand traffic from the same sender to multiple destinations may not require microflow based policing. We are, therefore, considering a sender-based aggregation of microflows in the ingress router. More work has to be done to determine the impact of such aggregation on the microflows from the same sender.

H. Traffic Analyzer

All of the resource managers (RMs) proposed and implemented till today only perform resource allocation and track resource availability. However, we think that one of the desirable features is their ability to perform traffic analysis and make it useful information in order to help network designers to design future networks or redesign existing ones. The PREM is the only RM, which has the following functionality.

1. The traffic Analyzer (TA) in the PREM keeps a record of reservation requests rejected for many reasons. TA uses the information of rejected requests due to unavailability of bandwidth and produces graphs using showing the amount of bandwidth required to satisfy all requests.
2. The TA also analyzes the actual traffic to see actual utilization of reserved bandwidth. Service providers can use this information to make a better and optimal policy.

I. Detailed Reservation Operation

This section describes common operation of resource reservation for guarantee of end-to-end QoS. It consists of the following steps.

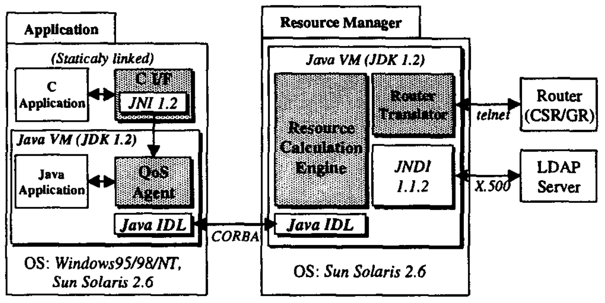
1. The network operator inputs static policy to the PM to be stored in the policy repository, currently an LDAP[11] directory. PM stores this policy after performing conflict resolution with already existing policies.
2. At the time of actual utilization of reserved resources, a resource allocation request (RAR) or dynamic policy may be sent to PREM via doReservation API.
3. In PREM, PM performs checks on the incoming resource reservation request against static policy to resolve any conflicts or to check its authentication.
4. If reservation is required beyond this domain, PM sends a request to IDM for negotiation with PREM of the proper adjacent domain. Similarly the IDM of the adjacent domain contacts the IDM of the next domain if required and so on.
5. If PM gets a successful commitment of resources from the IDM and if sufficient resources are available, then the resource reservation record is updated in the RTK.
6. After successful resource reservation in RTK, if it is a request for immediate reservation, a corresponding QoS setting for the microflow or source-based aggregated traffic is done in the ingress router and if it is a request for future reservation, the request is sent to a scheduler to be activated at the time determined by the reservation request. In the Router Translator (RT), proper router commands are generated and sent to the router via CLI (Command Line Interface) or COPS protocol[12].
7. After the whole process a positive or a negative acknowledgement is sent back to the Application.

J. Heterogeneous Router Support

A large network infrastructure already exists which deploys different types of routers. Any system, which attempts to provide a guarantee of QoS and requires a homogeneous network (same routers) to implement it, is not desirable because it will require a new infrastructure or large-scale modification of existing infrastructure. PREM provides heterogeneous router support by introducing a router translator module for each type of router. Each router translator is responsible for translating a logical policy into policy rules, which are router dependent. In our present implementation we have introduced only two router translators for, namely, silicon router (GR2000) and MPLS based router (CSR).

VI. Implementation of PREM Prototype

Figure 6, shows the implementation diagram of PREM. The main API is coded in C as well as Java. These API are used by the QoS agent to access services of the Resource Manager, which is also implemented in JDK 1.2. The repository is a Directory, which can be communicated through LDAP protocol. We use JNDI 1.1.2 to communicate with LDAP server and accommodate any other types of servers as well. Finally we use CORBA for communications between Application and Resource Manager for interoperability purposes.



JNI: Java Native Interface

JNDI: Java Naming and Directory Interface

Figure 6: PREM Prototype Implementation

To check the efficiency of our algorithm in the resource calculation engine, we performed a simulation of bandwidth calculation. The results are as under.

Simulation Environment & Assumptions:

Nodes:	two
Links:	one
Platform:	Java2
Machine:	UltraSPARC-II 400MHz
Memory:	512 MB

The graph in Figure 7 shows the performance of our algorithm against mainly two types of requests, i.e., doReservation and getAllocableBw (an internal method to find the allocable bandwidth).

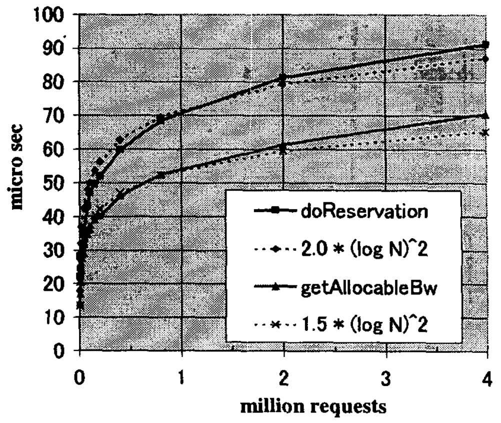


Figure 7: Simulation Results

The x-axis in the graph represents the number of requests in millions and the y-axis, the time in microseconds consumed by the bandwidth calculator to handle these requests. We can see from the figure, that the simulation results for doReservation almost coincide with a curve plotted for 2.0x(log N)^2 and for getAllocableBw with 1.5x(log N)^2. Here N is the number of requests. We can see from the graph that BC can handle reservation requests in time which is an order of square of the log number of requests or O((log N)^2).

From these results we can see that BC can handle a very large number of reservation requests in real time. However, the arriving number of requests at anytime is not completely predictable. The minimum and the maximum number of requests during a day (or some other time interval) cannot be generalized, but can be estimated for a particular network by its administrators. The administrators then can decide the size of domains to be managed by a BC.

VII. Summary & Conclusion

Many performance-critical applications like video and audio over IP networks require guarantee of end-to-end QoS. To provide guarantee of end-to-end QoS, network resource management is necessary which is performed by a resource manager. Some resource managers (also called bandwidth brokers or BBs) have been presented till today which are based on the DiffServ architecture.

In this paper, we described some features which are not included in the existing implementations and which are highly desirable for satisfactory performance of the performance-critical applications. These features are given below.

1. Delay Management
2. Future Reservation
3. Inter-Domain Reservation
4. Traffic Analysis
5. Heterogeneous Router Support

We proposed a resource manager architecture called PREM to address these issues and gave its detailed description. We also showed that the resource calculation engines and the strict admission control based on them helps in efficient but flexible policing which results in a guarantee of end-to-end QoS in terms of not only bandwidth but also delay and jitter.

VIII. Future Work

A. QoS Routing

The QoS routing is a desirable feature of a resource manager. The objectives of QoS routing[15] are as follows.

1. Dynamic determination of feasible paths.
2. Optimization of resource usage.
3. Graceful performance degradation

These objectives can be achieved in PREM using its resource tracker. However, after determining a specific path, route fixing must be done in the router. Many routers except MPLS based routers do not support route fixing. We plan to provide support for QoS routing in prototype of PREM using MPLS[13] enabled routers like CISCO routers, CSR of Toshiba and other route fixing routers like GR2000 of Hitachi.

B. Multicasting

The resource manager operates over DiffServ architecture. The DiffServ's simplicity is necessary for high scalability, but it also causes fundamental problems in conjunction with the use of IP Multicast in DS domains[16]. These problems have to be solved and standardized before support is provided in resource manager.

C. Inter-Domain SLA

This area is under research these days. An SLA is to be determined which can be used to indicate the needs of many different applications and would enable interoperability between domains managed by altogether different approaches.

IX. ACKNOWLEDGMENT

This research is done in collaboration with the CKP/NGI (Cyber Kansai Project/Next Generation Internet) project, which is partly supported by MITI (Ministry of International Trade and Industry) and Research for the Future Program of JSPS (Japan Society for the Promotion of Science) under the Project "Integrated Network Architecture for Advanced Multimedia Application Systems" (JSPS-RFTF97R16301). We thank TAO (Telecommunications Advancement Organization) and MPT (Ministry of Posts and Telecommunications) Japan for providing network links of JGN for experiments. We also thank Professor Shinji Shimojo of Osaka University, Professor Yuji Oie of Kyushu Institute of Technology, and Professor Tatsuya Suda of University of California, Irvine, for their continuous support and valuable comments.

X. REFERENCES

[1] I. Foster and C. Kesselman, editors. *"The Grid: Blueprint for a Future Computing Infrastructure,"* Morgan Kaufmann Publishers 1999.

[2] S. Shenker and J. Wroclawski, *"General Characterization Parameters for Integrated Service Network Elements,"* RFC2215, September 1997.

[3] K. Nichols, V. Jacobson and L. Zhang, *"A Two-bit Differentiated Services Architecture for the Internet,"* RFC 2638, July 1999.

[4] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, *"An Architecture for Differentiated Services,"* RFC2475, Dec 1998.

[5] K. Nichols, S. Blake, F. Baker, and D. Black, *"Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers,"* RFC2474, December 1998.

[6] S. Floyd and V. Jacobson, *"Link-sharing and Resource Management Models for Packet Networks,"* IEEE/ACM Transactions on Networking, pp 365-386, August 1995

[7] V. Jacobson, K. Nichols, and K. Poduri, *"An Expedited Forwarding PHB,"* RFC2598, June 1999.

[8] Akira Shirahase, Manzoor Hashmani, Mikio Yoshida, et al., *"Design and Deployment of QoS Enabled Network for Contents Businesses,"* International Conference on Computer Communication, Sep 14-16, 1999, Tokyo, Japan.

[9] B. Moore, E. Ellesson, and J. Strassner, *"Policy Framework Core Information Model -- Version 1 Specification,"* <draft-ietf-policy-core-info-model-03.txt>, January 2000, Work in progress.

[10] R. Neilson, J. Wheeler, F. Reichmeyer and S. Hares, *"A Discussion of Bandwidth Broker Requirements for Internet2 Qbone Deployment Version 0.6,"* available at http://www.merit.edu/working.groups/i2-qbone-bb/doc/BB_Req6.doc, March 1999.

[11] M.Wahl, T. Howes and S. Kille, *"Lightweight Directory Access Protocol,"* RFC1777, March 1995.

[12] D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan and A. Sastry, *"The COPS (Common Open Policy Service) Protocol,"* RFC2748, January 2000.

[13] E. Rosen, A. Viswanathan and R. Callon, *"Multiprotocol Label Switching Architecture,"* <draft-ietf-mpls-arch-06.txt>, August 1999, Work in progress.

[14] S. Shenker, C. Partridge, and R. Guerin, *"Specification of Guaranteed Quality of Service,"* RFC2212, September 1997.

[15] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick, *"A Framework for QoS-based Routing,"* RFC2386, August 1998

[16] R. Bless, and K. Wehrle, *"IP Multicast in Differentiated Services Networks,"* <draft-bless-diffserv-multicast-00.txt>, September 1999, Work in progress.

XI. BIOGRAPHY

Koichi Tajima received the B. Eng and M. Eng degrees from Osaka University in 1992 and 1994 respectively. He currently works as a system engineer at NIPPON STEEL Information & Communication Systems Inc. His research interests include QoS enabled networking, mobile agent systems and multi-agent systems.

Manzoor Hashmani received the Ph.D. and M.E. degrees from Nara Institute of Science & Technology, Ikoma, Japan in 1999 and 1997 respectively. He currently works as a system engineer at NIPPON STEEL Information & Communication Systems Inc. His research aims at high-speed communication, multimedia, and computer networks.

Mikio Yoshida received the B. Eng., M. Eng degrees from Kyoto University in 1981, 1983 respectively. He joined the IBM Tokyo Research Laboratory in 1986 and engaged in research on object-oriented programming languages. From 1994, he is with NIPPON STEEL Information & Communication Systems Inc. His current research interests include distributed computing, constraint logic programming and policy based networking systems.