An Integrated Network Resource and QoS Management Framework*

Ewa Kusmierek, Baek-Young Choi, Zhenhai Duan, and Zhi-Li Zhang
Department of Computer Science & Engineering
University of Minnesota
Minneapolis, MN 55455

Abstract—We present an integrated network resource and QoS management framework based on the idea of decoupling the network control plane from the data plane. Our framework is capable of providing per-flow QoS guarantees using scalable core-stateless packet forwarding mechanism at the network core. Virtual Time Reference System provides the abstraction of the data plane. Bandwidth broker performs all control and resource management functions while router functionality is kept minimal to make the performance of its main function, packet forwarding, efficient. We present the bandwidth broker architecture, its modules and their implementations as well as the implementation of the packet forwarding mechanism at the routers. We describe also the design of the interface between the bandwidth broker and the routers as well as the interface between the bandwidth broker and the users (applications).

I. INTRODUCTION

An increasing number of emerging Internet applications require better than best effort quality of service. Applications such as VoIP, VoD and interactive on-line games need end-to-end QoS guarantees defined in terms of throughput, delay and delay jitter. However, providing hard per-flow QoS guarantees proved to be a challenge. Guaranteed services require a number of resource management functions to be performed such as admission control and resource reservation, generally called QoS control functions. These functions can significantly increase the complexity and affect the scalability of the system.

A number of QoS models have been proposed to address the QoS requirements of the applications, e.g. IntServ and DiffServ. IntServ provides per-flow guarantees but faces some scalability issues. DiffServ, on the other hand, while more scalable, provides only service differentiation for large aggregates of network traffic with coarse-grain QoS.

In this paper we present a QoS model which combines fine granularity of IntServ, i.e., gives per-flow guarantees, with the scalability of DiffServ. Our model addresses granularity and scalability issues by decoupling QoS control plane from the packet forwarding plane, i.e., data plane [1]. Our integrated framework has two main components: the QoS and management plane consisting of bandwidth broker, and the data plane consisting of QoS-enabled configurable routers. A bandwidth broker (BB) performs network management functions and maintains resource reservation information, while core routers implement core stateless packet scheduling [10] to provide guaranteed services. The bandwidth broker interacts with the users/applications in order to accommodate their requests for QoS and data plane elements to implement its admission control and resource reservation decisions.

One of the main advantages of our framework is its flexibility. It can provide service guarantees at various levels of granularity, i.e., per-flow as well as with different levels of aggregation. An interface between the users and the BB allows new applications to be easily introduced without affecting the data plane components. Core routers can be dynamically reconfigured to implement the policy and management decisions regarding services available in the network domain. Therefore a network service provider can implement new services without software/hardware upgrades at the core routers. BB has a network-wide view of the available resources. Because of that and the fact that QoS reservation states are maintained only by the bandwidth broker, it can perform sophisticated QoS provisioning and admission control algorithms to optimize network utilization.

The focus of this paper is on the bandwidth broker architecture and its interaction with the data plane components (configurable routers) and users/applications in the integrated framework. Our goal is to provide a simple implementation of the BB modules using existing network management solutions on one hand, and the implementation of core-stateless packet scheduling mechanism at the routers on the other. We design mechanisms for interaction between control and data plane, and interaction between users (applications) and underlying network to complete the QoS system.

We start with the framework overview in Section II. In Section III we provide a brief description of the VTRS and its implementation. We limit the scope of that description to the details related to the interaction between control and data plane components. Design and implementation of the BB resource management module is presented in Section IV. We describe the implementation of the interface between the BB and core routers in and the interface between a user and BB in Section V.

* This work was supported in part by the National Science Foundation under the grants EIA-9818338, ANI-9903228, ITR-0085824, and CAREER Award NCR-9734428 as well as a University of Minnesota McKnight Land-Grant Professorship. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.
and Section VI, respectively. We conclude the paper in Section VII.

II. FRAMEWORK OVERVIEW

Our framework consists of two components, QoS and network management plane containing bandwidth broker, and data plane containing configurable routers as presented in Figure 1. The BB performs control functions in a conceptually centralized manner. It can be implemented in a functionally distributed manner with a number of interacting modules. In this section we present a brief description of the key modules and their interaction with the data plane elements in the network space, and between the BB and applications in the user space.

Admission control module interacts with the user through the network management back-end. Generally the MIBs provide an abstract model of the network resources for all BB modules. The resources included in the network model consist of the physical components of the network and its connectivity, as well as logical components such as paths along which the traffic is routed.

The network management back-end is also used by the service management module to configure new services in the network. The BB provides information on the queuing disciplines which should be enabled at the router and their operational parameters. An example of such configuration is given in section V. Core routers perform packet forwarding and all operations required for any queuing discipline they support. No per-flow state is maintained in any core router. Ingress routers, on the other hand, may maintain per-flow state in order to perform per-flow shaping of the traffic entering the network as explained in section III. The per-flow information is provided by the BB through the network front-end. Finally, users/applications interact with the BB either directly or through the third party such as gatekeeper in the case of Voice-over-IP for example. RSVP-like signaling protocol is used for the communication through the application management front-end.

III. VTRS AS QoS ABSTRACTION OF DATA PLANE

The bandwidth broker architecture relies on the Virtual Time Reference System (VTRS) [2] to provide QoS abstraction of the data plan. In this section, we give a brief overview of VTRS and its implementation.

The Virtual Time Reference System was developed as a unifying scheduling framework to provide scalable support for guaranteed services. The key construct in VTRS is the notion of virtual timestamps, which is a part of the state carried by each packet. The timestamps are updated by the routers using only the packet state information and fixed parameters associated with the routers. The computation is therefore core-stateless, i.e., no per-flow states need to be maintained by the core routers. Conceptually, the virtual time reference system consists of three logical components: edge traffic conditioning at the network edge, packet state carried by packets, and perhaps virtual time reference/update mechanism at core routers.

Edge traffic conditioning ensures that the packets of a flow 1 are never injected into the network core at a rate exceeding its reserved rate. After going through the edge conditioner packets entering the network core carry state information that is initialized and inserted at the network edge. The packet state contains three pieces of information: 1) a rate-delay parameter pair determined by the bandwidth broker based on flow’s QoS requirements; 2) the virtual timestamp; and 3) the virtual time adjustment term of the packet, a parameter that is computed at the edge and used to ensure that the virtual spacing property is satisfied (see [2] for details).

Each core router references and updates the virtual timestamps. Packets are scheduled for forwarding in order determined by their timestamps. Using the virtual time reference system outlined above, the delay experienced by packets of a flow across the network core can be upper bounded in terms of the rate-delay parameter pair of a flow and the error terms of the routers along the flow’s path.

VTRS queuing discipline is implemented on two widely used Unix systems: BSD-UNIX and Linux. ALTQ, an alternate queuing in BSD-UNIX [9] is used as a mechanism for implementing packet scheduling disciplines. VTRS is implemented as one of the ALTQ queuing disciplines. Its main

---

1 Here a flow can be either an individual user flow, or an aggregate traffic of multiple user flows, defined in any appropriate fashion.
components are: per-flow packet shaper and packet scheduler. Packet shaper performs three functions: per-flow packet classification, packet state initialization and rate-shaping. Packet scheduler updates packet state, places packet in an output queue in the order determined by their timestamps and dequeues a packet when requested by device driver. It maintains two queues, the queue for VTRS packets and FIFO queue, which has lower priority. Linux offers built-in traffic control framework at outgoing interface including queuing disciplines and the corresponding classes, policers and filters. The arbitrary nesting of queuing disciplines provides a highly flexible framework for new traffic control infrastructure such as VTRS. The VTRS queuing discipline defines three classes: (1) Edge Traffic Conditioner, (2) Core Scheduler, and (3) Best Effort Scheduler for ingress, transit and best effort traffic, respectively. Both BSD-UNIX and Linux implementations have mechanisms to dynamically configure the traffic control elements.

We have implemented packet states encoding using two methods: IP tunneling and MPLS label. In IP tunneling, the packet state is inserted into encapsulating IP header, which is removed at the egress. Since packet fragmentation can be easily avoided by performing MTU discovery at the source or ingress point, IP header fields designed for fragmentation are available for packet state encoding. More specifically, the 16-bit identification field and the 13-bit fragmentation offset in the encapsulation header can be re-used. We encode the timestamp value in 16-bit identification field. fragment offset bits are split between delay and rate. For the MPLS based implementation, a new MPLS label is defined to carry VTRS packet state. This label is referred to as QoS label. It is appended to an original MPLS label used for switching purposes. VTRS label consists of 4 DS bits, 16-bit timestamp, 6-bit rate index and 6-bit delay value. In addition EXP bits in the regular MPLS switching label are set to denote that QoS label is attached.

Packet per-flow classification performed at the ingress points is based on information in the packet header and the control and policy information provided by the BB. The Recursive Flow Classification (RFC) algorithm [4] is used. The fields used for classification in the packet headers are IP addresses and port numbers of source/destination, protocol and type of service (TOS) field.

VTRS can then be easily implemented in practice. Some of implementation issues include packet state encoding, time granularity used for timestamps, edge router time synchronization, and edge shaper queue structure. For a more detailed description see [3].

IV. RESOURCE MANAGEMENT AND NETWORK MODEL

In order to perform its functions, the BB needs accurate information about both static and dynamic state of the network. The resource management module maintains a number of management information bases (MIBs). It provides access to the network information for other bandwidth broker modules such as admission control and QoS routing. It also updates management information bases with the information provided by these modules and performs consistency and integrity checks on all information in the database.

The information contained in the MIBs reflects the current state of the network, it must be relevant and accurate for the management functions performed by the bandwidth broker. An abstract network-wide view of an autonomous system backbone as seen by an entity managing the system is maintained. It contains information about physical components of the network, network connectivity as well as logical components such as paths established in the system and flows traversing them. We now describe the structure of the network model and the methods used to obtain information about network.

A. Network Model Structure

The network model of an autonomous system contains the representation of the network components as viewed by an Internet Service Provider [6]. We consider first the topology information base which constitutes a static or semi-static part of the network model and contains the representation of its physical components. These components are routers, interfaces and links. Their representation in the database is similar to the one presented in [6]. However, it contains also additional attributes that reflect the fact that routers can be configured by the bandwidth broker to support various types of services.

An IP router receives incoming packets and forwards them toward their destinations. Its attributes include name, loopback IP address, status and a list of interfaces that ties router representation to the interface representation. Most of the information related to the functional representation of a router is provided separately for each of its interfaces. An interface receives incoming packets, queues and transmits them. Its attributes are: the name of the router it is a part of, IP address and prefix, OSPF weight, type and status. Two main types of interfaces are recognized: incoming and outgoing. Each incoming interface is further classified as an ingress interface or a core interface depending whether a link connects it to a router inside or outside of the domain. Similarly, an outgoing interface can be an egress or a core interface. With this classification, a single physical interface may be represented by more than one logical interface. Each interface is also tied to the representation of a queuing discipline it has attached to it. A queuing discipline is a logical component that characterizes behavior of an interface. It has one attribute common to all disciplines - the name of a discipline. Other attributes depend on the type of queuing discipline. Each link over which traffic is exchanged between routers, is characterized by an IP prefix, capacity, propagation delay and MTU. It is also linked to a set of interfaces associated with that link.
The dynamic part of the network model consists of two information bases. A flow information base contains data regarding each active flow such as its identification, traffic profile, service profile and QoS reservation. The second one, path QoS state information base contains information regarding paths established between ingress and egress points. The information consists of the number of hops, types of schedulers along the path, propagation delay, maximum permissible packet size and a number of QoS state parameters regarding the current QoS reservation status. The path QoS state information base reflects the path oriented approach to perform efficient admission control [1]. The elements in the dynamic information bases are tied to corresponding elements in the network topology base, e.g. a path is associated with a number of links it consists of.

B. Database Organization

The network model is implemented as an object oriented library supported by relational database management system. The object oriented model constitutes a volatile representation of the overall network model, while the persistent storage is facilitated in RDBMS environment - Oracle which supports querying in SQL. Since Java has a library for database connectivity and access (JDBC), it was our choice for implementing the volatile part. The resource management module operates, stores information in and retrieves information from the volatile Java model which constitutes the network management back-end in Figure 1. The Java and Oracle implementations of the network model are kept consistent at any time.

C. Populating the Model

The information about the network stored in the database comes from two sources. The static part of the network model is initially populated with information obtained from the configuration files of the routers in the domain [6]. The resource manager analyzes these files, extracts network connectivity information, checks it for the consistency and passes it to the network model. The configuration files do not provide status information though. For example, the topology of the network changes due to changes in the status of routers, interfaces and links.

The resource manager obtains status information through OSPF. Every router in the domain participating in OSPF maintains topology information of its area obtained by exchanging information with other routers. Whenever a change occurs in the status of an interface or a link the router that is associated with the affected interface or link floods the OSPF Link State Advertisement throughout the domain. When a router goes down or comes up, the router it is directly connected to, identifies the change due to the lack or presence of Hello messages. Thus, the current status of the network can be determined by the resource manager by acting as an OSPF listener. It is an effective way to obtain the current topology information without any additional overhead introduced.

V. DATA AND CONTROL PLANE INTERACTION

The proposed bandwidth broker architecture decouples the QoS control plane from the packet forwarding plane. The bandwidth broker stores and maintains the QoS reservation states for the whole domain, performs QoS control function such as admission control and path set-up. Routers on the other hand, perform only data plane functions: packet scheduling and forwarding. Therefore routers must communicate with bandwidth broker to implement its network management decisions such as resource management, service configuration etc.

We have selected Common Open Policy Service (COPS) signaling protocol [8] for the communication between bandwidth broker and the routers. The protocol is designed for the exchange of policy information between the Policy Decision Point (PDP) and the Policy Enforcement Point (PEP). PDP, the bandwidth broker in our case, decides on the policy which is next communicated to and implemented at PEPs, i.e., the packet traffic forwarding devices (routers) in the data plane. The protocol employs a client/server model. The assumption is made that there exists at least one server in the domain and that it interacts with all clients. For reliability reasons, it is recommended that there exists a secondary PDP. COPS uses TCP as its transport protocol for reliable exchange of information and therefore no additional mechanism is necessary for reliable communication. It provides message level security for authentication, reply protection and message integrity.

The communication between the BB and routers is organized in the following way. The router establishes and maintains a TCP connection to the bandwidth broker. Initially it sends a request message for configuration information. The bandwidth broker responds with the requested information and continues by sending unsolicited decision messages to install data or to remove data from router’s Policy Information Base (PIB) whenever a change in the router is required. A change in router’s PIB, a new policy decision, is immediately implemented in the router. Router may feed the state information back to the bandwidth broker. In order for the bandwidth broker to communicate client specific information to the router, a set of new COPS messages and objects is defined.

Consider VTRS as an example of the service provided. Its set of configuration parameters consists of the value of TOS carried by the packets requiring that service, information related to packet state encoding, i.e., a number of bits used to encode delay or rate, and time granularity used for timestamps calculation. VTRS service configuration requires also enabling a VTRS queuing discipline, choosing a specific type of scheduler - rate-based or delay-based, configuring an interface as an edge or core interface and passing parameters such as error term. All of the above mentioned parameters are sent by the PDP to the routers. The routers interpret the parameters and implement them. Once enabled and configured,
an ingress VTRS interface receives flow information from the bandwidth broker as a part of its dynamic configuration. The message is sent upon the acceptance of a new flow by the admission control module and when a flow ends. In this case, the access control table used by RFC classifier at the edge router for VTRS will be modified. Each entry in the access control table contains flow identification information based on a flow specific 5-tuple \( ^2 \). If a new flow is accepted, admission control module initiates a change first in the bandwidth broker’s PIB - a new rule for packet classification is added. That decision is then sent using the COPS protocol to the appropriate edge router. The router updates its PIB, then creates the preprocessed RFC equivalence tables which are finally installed at the packet classifier.

VI. APPLICATION AND BANDWIDTH BROKER INTERACTION

We illustrate the application-network interaction using VoIP as an example of the application whose successful functioning depends on the quality of service guarantee provided by the network. We present the modules of VoIP application first, then describe its interaction with the bandwidth broker.

Most VoIP applications implement the H.323 protocol stack [7]. Four logical entities are required in the implementation. They are terminals, gateways, gatekeepers and multipoint control units. A terminal is an endpoint where H.323 data streams and control signaling originate and terminate. A gateway, which is an optional component, provides data format translation, control signaling translation, audio and video codec translation, call setup and termination functionality. Gatekeepers are needed to ensure reliable, commercially feasible communication. A gatekeeper is often referred to as the brain of H.323 enabled network because of the central management and control services it provides. All endpoints must be registered with a gatekeeper if it exists, and their control messages are routed through the gatekeeper. A multipoint control unit enables conferencing between three or more endpoints.

Because of its role in H.323 stack, the gatekeeper is the most interesting entity for us. It performs admission and access control of endpoints. This control is based on the bandwidth availability, limit on the number of simultaneous H.323 calls, and registration privileges of endpoints. It does bandwidth management and has routing capabilities. All of these functions make the gatekeeper a suitable counterpart for the bandwidth broker, a counterpart that has application specific information. To perform its role the gatekeeper interacts with the bandwidth broker of each domain a flow will traverse. Hence a new module is added: call-qos that aggregates the individual calls into calls between any two gateways that route them. It reserves aggregated bandwidth for the path between the two gateways with the bandwidth broker. The call-qos module acts then as “middle man” between the user application and the bandwidth broker. It makes the QoS reservation transparent to end-users, and uses its knowledge of application specific requirements to give the bandwidth broker the information it needs to provide the QoS guarantees.

More specifically, the call-qos module performs the following functions. It maintains information about each call such as source address and port, destination address and port, protocol id, actual bandwidth used, codec used, gateway through which the call is sent. It keeps track of the calls going through each gateway and whether the aggregation is performed or not; if aggregation is performed, what the specification of the aggregated traffic is. It also allocates bandwidth to new calls, if it can accommodate their requirements using bandwidth already reserved for the aggregate traffic. Otherwise it sends a request to the bandwidth broker, and communicates flow information when a call is added or deleted. The communication between the call-qos module and the bandwidth broker is based on RSVP signaling protocol.

The user/application can communicate directly with the BB or rely on the third party that has a knowledge of the application characteristic and its requirements. In the latter case the interaction with the network control plane is transparent to user and a new abstraction level is defined. In either case the communication is possible through the application/service management front-end of the BB and using the RSVP based protocol.

VII. CONCLUSIONS

We have described and implemented the integrated network resource and QoS management framework. It realizes the idea of decoupling the network control plane from the data plane. Bandwidth broker performs all network and QoS control functions, maintaining all necessary information about network state, while core routers perform core-stateless packet forwarding capable of supporting guaranteed service. Our framework combines scalability of DiffServ with the service granularity of IntServ.
The bandwidth broker has a modular structure. We proposed a way to implement some of the key components of the BB, namely its management information base, network management front- and back-end, and application/service management front-end. The bandwidth broker relies on the Virtual Time Reference System to provide the abstraction of the data plane. We used VTRS as an example of the service to illustrate the bandwidth broker interaction with routers in order to enable that service in the network. We have illustrated the interaction between the bandwidth broker and an application with the QoS requirements with an example of Voice-over-IP. Our future work includes a quantitative evaluation of the effectiveness of the system.

ACKNOWLEDGMENT

The following students participated in the VTRS and BB implementation project: Nilesh Bagad, Anish Desai, Ameeta Deshpande, Shirisha Kaipa, Jeff Krasky, Kondapally Kumar, Ravishankar Rajamani, Bruce Rovner, Gayathri Seetharaman, Pratik Shah, and Rajasekhar Sistla. We want to acknowledge their contribution to the project.

REFERENCES