IEICE TRANS. COMMUN., VOL.E84–B, NO.10 OCTOBER 2001

PAPER

Providing Scalable Support for Multiple QoS Guarantees: Architecture and Mechanisms

Yiwei Thomas HOU[†], ***Regular Member***, **Zhenhai DUAN**[††], **Zhi-Li ZHANG**[††], ***Nonmembers***, ***and*** **Takafumi CHUJO**[†], ***Regular Member***

SUMMARY The IETF Differentiated Services (DiffServ) framework achieves *scalability* by (1) aggregating traffic flows with coarse grain QoS on the data plane, and (2) allocating network resources with a bandwidth broker (BB) on the control plane. However, there are many issues that need to be addressed under such framework. First, it has been shown that the concatenation of strict priority (SP) scheduler of class-based queues (CBQ) can cause delay jitter unbounded under certain utilization, which is not acceptable to support the premium service (PS). Furthermore, it is not clear how such a DiffServ network can support traffic flows requiring the guaranteed service (GS), which is a desirable feature of the future Internet. This paper presents architecture and mechanisms to support multiple QoS under the DiffServ paradigm. On the data plane, we present a node architecture based on the *virtual time reference system* (VTRS). The key building block of our node architecture is the *core-stateless virtual clock* (CSVC) scheduling algorithm, which, in terms of providing delay guarantee, has the same expressive power as a stateful weighted fair queueing (WFQ) scheduler. With the CSVC scheduler as our building block, we design a node architecture that is capable of supporting integrated transport of the GS, the PS, the assured service (AS), and the traditional best effort (BE) service. On the control plane, we present a BB architecture to provide *flexible* resource allocation and QoS provisioning. Simulation results demonstrate that our architecture and mechanisms can provide *scalable* and *flexible* transport of integrated traffic of the GS, the PS, the AS, and the BE services.

key words: *quality of service, differentiated services, scheduling, bandwidth broker, scalability*

1. Introduction

The ability to provide end-to-end *guaranteed services* (GS) [22] (e.g., guaranteed delay or bandwidth) for networked applications is a desirable feature of the future Internet. To enable such services, Quality-of-Service (QoS) support from *both* the network *data plane* (e.g., packet scheduling) and the *control plane* (e.g., admission control and resource reservation) is needed. For example, under the IETF Integrated Services (IntServ) architecture, scheduling algorithms such as *weighted fair queueing* (WFQ) [10], [19], [20] were developed to support the GS. Furthermore, a signaling and reservation protocol, RSVP [6], [26], for setting up end-to-end QoS reservation along a flow's path was also proposed and standardized. However, due to its need for performing *per-flow* management at core routers, the *scalability* of the IntServ architecture has been questioned.

To address the issue of scalability, the IETF has introduced the Differentiated Services (DiffServ) model [3], which achieves scalability by offering services for an aggregate traffic rather than on a per-flow basis. Furthermore, the DiffServ model pushes much complexity out of the core of the network into edge routers, which processes smaller volume of traffic and fewer number of flows. On the data plane, simple *per-hop behaviors* (PHBs) (e.g., expedited forwarding (EF) [16] and assured forwarding (AF) [14]) have been defined to treat traffic aggregate and to provide differentiation in packet forwarding. On the control plane, a centralized *bandwidth broker* (BB) [18] was introduced to perform resource management and allocation within a DiffServ domain (intra-domain) and to maintain *service level agreement* (SLA) between DiffServ domains (inter-domain).

Under such DiffServ paradigm, two new services, namely, the *premium service* (PS) and the *assured service* (AS) have been proposed [18] to provide coarse-grained end-to-end QoS guarantees over the Internet. The PS is expected to offer a guaranteed rate, low delay jitter packet delivery, while the AS is expected to offer a sustainable rate guarantee for a traffic flow*. It has been suggested [18] that a network of routers employing a simple class-based *strict priority* (SP) scheduling between the PS and the AS queues (with FIFO for each queue) may achieve end-to-end support for the PS and the AS through a DiffServ network domain.

Several issues have been raised regarding such class-based SP node architecture in a DiffServ network. First, it has been shown recently [2], [8] that the delay jitter under a concatenation of class-based SP schedulers can be unbounded over a certain utilization, which means that the QoS requirement for the PS cannot always be supported under such node architecture. Second, it is not clear how such class-based SP node architecture can support end-to-end (either per-flow or

Manuscript received September 28, 2000.
Manuscript revised March 12, 2001.

[†]The authors are with Fujitsu Laboratories of America, Sunnyvale, CA, USA.
[††]The authors are with University of Minnesota, Department of Computer Science and Engineering, Minneapolis, MN, USA.

*Here a flow can be either an individual user flow, or an aggregate traffic flow of multiple user flows, defined in any appropriate fashion.

aggregate) GS [22], which is a desirable feature of the future Internet.

The purpose of this paper is to design a node architecture under the DiffServ paradigm to provide *scalable* and *integrated* transport of the GS, the PS, the AS, and the traditional best effort (BE) services. More specifically, we want to achieve the following three design objectives.

Objective 1: Multiple QoS Guarantees. Our network should be capable of simultaneously transporting the GS, the PS, the AS, and the BE services while meeting the QoS requirements of each service. More specifically, (1) For the GS, each flow specifies its traffic behavior through a traffic profile (e.g., $(\sigma, \rho, P, L^{max})$) and a service requirement (e.g., delay requirement) [22]. The network decides whether to admit or reject the call through an *admission control* procedure. If the GS flow is admitted into the network, then the end-to-end delay bound must never be violated and no packet shall be lost for this flow, as long as the source's traffic conforms to its traffic profile. (2) For the PS, each flow specifies its traffic profile and service profile through a peak rate requirement [18]. The network decides whether to admit or reject the call through an admission control procedure. At the network edge, each admitted PS flow is shaped according to its peak rate requirement. An admitted PS flow should experience low delay jitter and low loss when it traverses the network. (3) For the AS, each flow specifies its traffic profile and service profile through a sustainable rate requirement. The network decides whether to admit or reject the call through the admission control procedure. At the network edge, an admitted AS flow is marked according to its sustainable rate. Packets spaced according to its sustainable rate will have their AS bit marked; packets exceeding the sustainable rate will be marked as the BE service. Unlike the PS, *edge shaping is not employed for the AS.* (4) For the BE service, there is no specific traffic profile and QoS service requirement for each flow. Each flow can enter the network freely and share any remaining network resources.

Objective 2: Scalability in Network Core (Core-Stateless). In consistent with the IETF DiffServ model, we identify all the routers within a DiffServ domain and distinguish them between *edge* and *core* routers. To support the GS, the PS, and the AS, we allow edge routers to maintain per-flow state. That is, edge routers perform per-flow traffic classification and conditioning (shaping and marking). Since each edge router typically processes smaller volume of traffic and fewer number of flows as compared with a core node, scalability is *not* a concern at an edge router. On the other hand, we require that core routers maintain no per-flow state, e.g., per-flow reservation state or per-flow scheduling state. Such scalability requirement on the core nodes is also called "core-stateless" approach [15], [23].

Objective 3: Decouple QoS Control from Core Routers for Flexible Resource Allocation and QoS Provisioning. We aim to decouple the QoS control plane from the core routers and use a centralized BB to control and manage domain-wide QoS provisioning. As we shall see in later sections of this paper, there are many *significant advantages* for decoupling QoS control plane from the data plane. One advantage that is enabled by such decoupling is that new network management policies can be *easily* implemented, eliminating the additional complexity (hardware/software upgrade) on the core routers as otherwise would incur. That is, all the policy decision and its enforcement can be performed on the control plane using the BB without any hardware/software change on the data plane.

This paper presents an architecture to meet the above three design objectives. Our node architecture is based on the *virtual time reference system* (VTRS), which has been recently introduced by Zhang et al. [27] as a *unifying* scheduling framework for *scalable* support for the GS. Under the VTRS, a core-stateless scheduler, called *core-stateless virtual clock* (CSVC) has been introduced. The CSVC scheduler has the same expressive power in providing delay and rate guarantee as a *stateful* WFQ scheduler, albeit it does not maintain any reservation or scheduling states in the router. The architecture and mechanisms presented in this paper, which covers both data plane and control plane, builds upon the VTRS/CSVC and aims to achieve the three design objectives listed earlier. In this sense, this paper is a sequel to [27].

The remainder of this paper is organized as follows. In Sect. 2, we first give an overview of the VTRS and the CSVC scheduler. Then we present our node architecture (edge and core) on the data plane for integrated transport of the GS, the PS, the AS, and the BE services. Section 3 presents a BB architecture and admission control algorithms on the control plane. In Sect. 4, we discuss the performance and complexity of our architecture. Section 5 presents simulation results to demonstrate the performance of our node architecture in providing QoS guarantees to each type of services. In Sect. 6, we discuss related work. Section 7 concludes this paper.

2. A Core-Stateless Architecture with Multiple QoS Support

This section presents our core-stateless node architecture on the data plane to support the GS, the PS, the AS, and the BE services. We organize this section as follows. Section 2.1 presents the essential background on the VTRS and introduces the CSVC scheduler. In Sect. 2.2, we propose a node architecture which builds upon the VTRS/CSVC, for integrated transport of the GS, the PS, the AS, and the BE services in a DiffServ network domain.

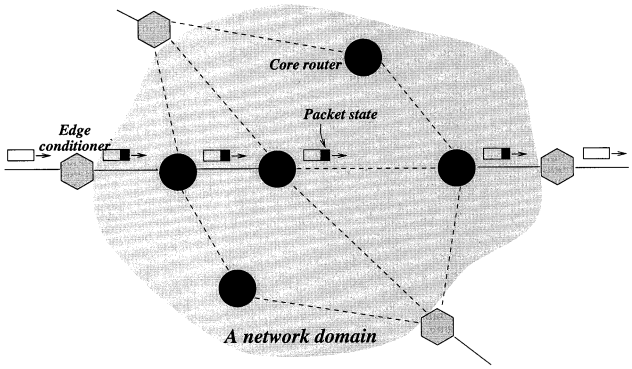


Fig. 1 A network domain where the VTRS is deployed.

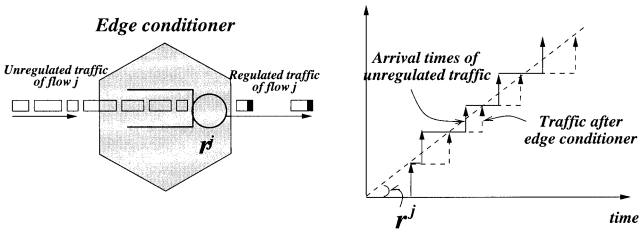


Fig. 2 Edge conditioning and its effect.

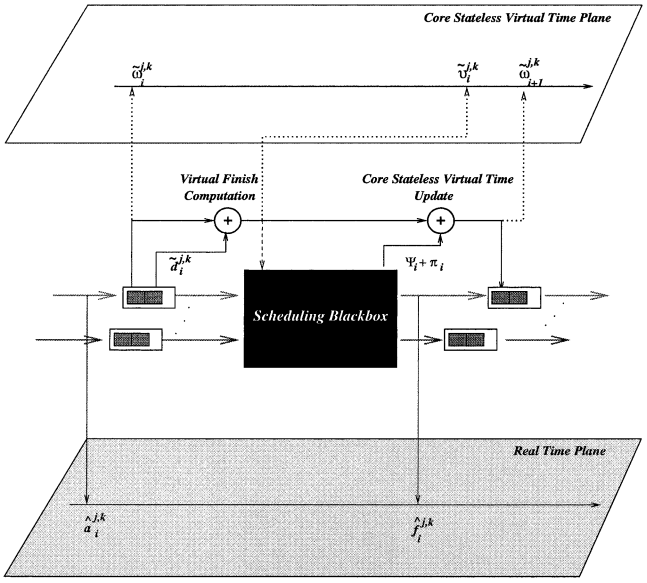


Fig. 3 Per-hop behavior and operations at a core node under the VTRS.

2.1 Virtual Time Reference System: A Background

The VTRS [27] was developed as a *unifying* scheduling framework to provide *scalable support* of the GS. The key construct in the VTRS is the notion of *packet virtual time stamps,* which, as part of the packet state, are referenced and updated as packets traverse each core router. As we will see shortly, the virtual time stamps associated with the packets of a flow form a *thread* which "weaves" together the per-hop behaviors of core routers along the path of the flow to provide the QoS guarantees for the flow. A key property of packet virtual time stamps is that they can be computed using *solely* the packet state carried by packets (plus a couple of fixed parameters associated with core routers). In this sense, the VTRS is *core-stateless,* as no per-flow state is needed at core routers for computing packet virtual time stamps.

The idea of virtual time has been used extensively in the design of packet scheduling algorithms such as the VC [25] and the WFQ [10], [19], [20] schedulers. The notion of virtual time defined in these contexts is used to emulate an ideal scheduling system and is defined *local* to each scheduler. Computation of the virtual time function requires *per-flow* information to be maintained at each node. In contrast, under the VTRS, the notion of virtual time embodied by packet virtual time stamps can be viewed, in some sense, as *global* to an entire domain (see Fig. 1). Its computation is *core-stateless,* relying only on the packet state carried by packets.

Conceptually, the VTRS consists of three logical components: *packet state* carried by packets, *edge traffic conditioning* at the network edge (see Fig. 2), and *per-hop virtual time reference/update mechanism* at core routers (see Fig. 3). These three components are briefly described below.

1. Edge Traffic Conditioning. Edge traffic conditioning plays a key role in the VTRS, as it ensures that the packets of a flow will never be injected into the network core at a rate exceeding its reserved rate (see Fig. 2). Formally, for a flow j with a reserved rate r^j, the inter-arrival time of two consecutive packets of the flow at the first hop core router is such that

$$\hat{a}_1^{j,k+1} - \hat{a}_1^{j,k} \geq \frac{L^{j,k+1}}{r^j}, \tag{1}$$

where $\hat{a}_1^{j,k}$ denotes the arrival time† of the kth packet $p^{j,k}$ of flow j at the first network core router, $L^{j,k}$ the size of packet $p^{j,k}$, and r^j the reserved rate of flow j.

2. Packet State. After going through the edge conditioner at the network edge, packets entering the network core carry in their packet headers certain packet state information that is initialized and inserted at the network edge. The packet state carried by the kth packet $p^{j,k}$ of a flow j contains three types of information: (1) QoS reservation (i.e, the reserved rate r^j) of

†Note that in order to model non-preemptive, non-cut-through network system, throughout the paper we adopt the following convention: a packet is considered to have arrived at a server only when its last bit has been received, and it to have departed the server only when its last bit has been serviced. In addition, we assume that the edge conditioner and the first-hop router (i.e., the first core router) are co-located, and thus the propagation delay from the edge conditioner to the first core router is negligible.

the flow†; (2) the virtual time stamp $\tilde{\omega}_i^{j,k}$ of the packet that is associated with the router i currently being traversed; and (3) the virtual time adjustment term $\delta^{j,k}$ of the packet.

The rate parameter (r^j) is determined by the BB (during call set up time and at the network edge) based on flow j's QoS requirements (see Sect. 3.4 for details), and is inserted into every packet of the flow. For the kth packet of flow j, its virtual time stamp $\tilde{\omega}_1^{j,k}$ is initialized to $\hat{a}_1^{j,k}$, the actual time it leaves the edge conditioner and enters the first core router along the flow's path. That is,

$$\tilde{\omega}_1^{j,k} = \hat{a}_1^{j,k}. \tag{2}$$

The virtual time adjustment term $\delta^{j,k}$ for packet $p^{j,k}$ is set to

$$\delta^{j,k} = \frac{\Delta^{j,k}}{h}, \tag{3}$$

where h is the number of hops along the flow's path, and $\Delta^{j,k}$ is computed at the network edge using the following recursive formula:

$$\Delta^{j,1} = 0 \tag{4}$$

$$\Delta^{j,k} = \max\left\{0,\ \Delta^{j,k-1} + h\frac{L^{j,k-1} - L^{j,k}}{r^j} + \hat{a}_1^{j,k-1} - \hat{a}_1^{j,k} + \frac{L^{j,k}}{r^j}\right\},\ \text{for } k = 2, 3, \ldots. \tag{5}$$

The physical meaning of $\Delta^{j,k}$ is that it represents the cumulative delay experienced by packet $p^{j,k}$ in an *ideal dedicated per-flow system* [27], where packets of flow j are serviced by h tandem servers with capacity r^j.

3. Virtual Time Reference/Update Mechanism and Per-Hop Core Router Behavior Characterization. In the conceptual framework of the VTRS, each core router is equipped with a per-hop virtual time reference/update mechanism to maintain the continual progression of the *(global) virtual time* embodied by the packet virtual time stamps. This virtual time stamp $\tilde{\omega}_i^{j,k}$ represents the arrival time of the kth packet $p^{j,k}$ of flow j at the ith core router *in the (global) virtual time,* and thus it is also referred to as the *virtual arrival time* of the packet at the core router. The virtual time stamps $\tilde{\omega}_i^{j,k}$'s associated with packets of flow j satisfy the following two important properties: (1) *virtual spacing property:* $\tilde{\omega}_i^{j,k+1} - \tilde{\omega}_i^{j,k} \geq L^{j,k+1}/r^j$, and (2) the *reality check property:* $\hat{a}_i^{j,k} \leq \tilde{\omega}_i^{j,k}$, where $\hat{a}_i^{j,k}$ denotes the actual arrival time of packet $p^{j,k}$ at router i. The virtual spacing property says that, when measured according to this (global) virtual time, the traffic flow preserves its reserved rate. These two properties are important in ensuring that the end-to-end delay experienced by packets of a flow across the network core is bounded.

In order to ensure that these two properties are satisfied, the virtual time stamps must be appropriately referenced or updated as packets enter or leave a core router (see Fig. 3). The referencing/updating rule depends on the scheduling algorithm (or *scheduler*) employed by a core router and its characteristics. Since we only consider *rate-based* schedulers in this paper, then, for scheduler $\mathcal{S}_i$ at the ith router, packet $p^{j,k}$ is associated with the *virtual delay* parameter, $\tilde{d}_i^{j,k}$, and

$$\tilde{d}_i^{j,k} = \frac{L^{j,k}}{r^j} + \delta^{j,k}, \tag{6}$$

and its *virtual finish time*, $\tilde{\nu}_i^{j,k}$, is defined as

$$\tilde{\nu}_i^{j,k} = \tilde{\omega}_i^{j,k} + \tilde{d}_i^{j,k} = \tilde{\omega}_i^{j,k} + \frac{L^{j,k}}{r^j} + \delta^{j,k}. \tag{7}$$

The *per-hop behavior* of a core router (or rather, its scheduler) is characterized by an *error term,* which is defined with respect to the virtual finish time and *actual* finish time of packets at the router. Let $\hat{f}_i^{j,k}$ denote the actual time packet $p^{j,k}$ departs the scheduler $\mathcal{S}_i$. We say that $\mathcal{S}_i$ can guarantee flow j its reserved rate r^j with an error term Ψ_i, if for any k, $\hat{f}_i^{j,k} \leq \tilde{\nu}_i^{j,k} + \Psi_i$. In other words, each packet of flow j is guaranteed to depart scheduler $\mathcal{S}_i$ by the time $\tilde{\nu}_i^{j,k} + \Psi_i = \tilde{\omega}_i^{j,k} + \tilde{d}_i^{j,k} + \Psi_i$.

Given the error term Ψ_i of the scheduler $\mathcal{S}_i$, the virtual time stamp of packet $p^{j,k}$ after it has traversed $\mathcal{S}_i$ is updated using the following reference/update rule:

$$\tilde{\omega}_{i+1}^{j,k} = \tilde{\nu}_i^{j,k} + \Psi_i + \pi_i = \tilde{\omega}_i^{j,k} + \tilde{d}_i^{j,k} + \Psi_i + \pi_i, \tag{8}$$

where π_i denotes the propagation delay from the ith router to the next-hop router along the flow's path. It has been shown [27] that by using the reference/update rule in (8), the virtual spacing and reality check properties of virtual time stamps are satisfied at *every* core router.

2.1.1 End-to-End Delay Bound and QoS Abstraction of Data Plane

An important consequence of the VTRS outlined above is that the end-to-end delay bound on the delay experienced by packets of a flow across the network core can be expressed in terms of the reserved rate of a flow and the error terms of the routers along the flow's path. Suppose there are total h hops along the path of flow j, all of which are rate-based schedulers. Then for each packet $p^{j,k}$ of flow j, we have

$$\hat{f}_h^{j,k} - \hat{a}_1^{j,k} \leq h\frac{L^{j,max}}{r^j} + \sum_{i=1}^{h}\Psi_i + \sum_{i=1}^{h-1}\pi_i\,, \tag{9}$$

where $L^{j,max}$ is the maximum packet size of flow j.

†For the purpose of this work, we will only consider *rate-based* schedulers under the VTRS. A more general treatment of the VTRS that includes *delay-based* schedulers can be found in [27].

The VTRS does not mandate any specific scheduling mechanisms to be implemented in a network domain as long as their abilities to provide delay guarantees can be characterized using the notion of error term. In fact, it has been shown [27] that almost all known scheduling algorithms can be characterized, be they *stateless* (e.g., FIFO) or *stateful* (e.g., WFQ).

2.1.2 Core-Stateless Virtual Clock Scheduling Algorithm

The VTRS leads to the design of a set of new core-stateless scheduling algorithms. One of the most important one is the *core-stateless virtual clock* (CSVC) scheduler, which will be the key building block in our node architecture in this paper.

The CSVC is a work-conserving counterpart of the *core-jitter virtual clock* (CJVC) scheduling algorithm [23]. The CSVC scheduler services packets in the order of their virtual finish times (recall that the virtual finish time of packet $p^{j,k}$ is given by $\tilde{\nu}^{j,k} = \tilde{\omega}^{j,k} + L^{j,k}/r^j + \delta^{j,k}$). It has been shown [27] that as long as the total reserved rate of flows traversing a CSVC scheduler does not exceed its capacity (i.e., $\sum_j r^j \leq C$), then the CSVC scheduler can guarantee each flow its reserved rate r^j with the minimum error term $\Psi = L^{*,max}/C$, where $L^{*,max}$ is the largest packet size among all flows traversing the CSVC scheduler.

Theorem 1: The end-to-end delay bound experienced by a flow j with reserved rate r^j in a network of the CSVC schedulers is the same as that under a network of the WFQ schedulers.

The theorem is proved [27] by showing that the WFQ scheduler has the same error term as the CSVC scheduler under the VTRS, which is $\Psi = L^{*,max}/C$.

Theorem 1 shows that the CSVC scheduler has the same expressive power, in terms of providing delay and rate guarantees, as a *stateful* WFQ scheduler, albeit it does not maintain any reservation or scheduling state in the router.

Due to paper length limitation, we can only give some highlights of VTRS here. We strongly encourage readers to [27] for more details on the architectural insights and theoretical underpinning for VTRS.

2.2 A Node Architecture with Scalable Support of Multiple QoS

In this section, we present a node architecture, which builds upon the VTRS/CSVC, for scalable support of integrated traffic of the GS [22], the PS [18], the AS [18], and the BE services. In Sect. 2.2.1, we present the edge conditioning functions. Section 2.2.2 shows the buffering and scheduling mechanisms for both edge and core nodes.

2.2.1 Edge Conditioning

Edge traffic conditioning plays a key role in our architecture. In the following, we show how traffic conditioning is performed for the GS, the PS, the AS, and the BE flows, respectively.

Edge Shaping for the GS Flows. Before entering the traffic shaper, suppose the traffic profile of an GS flow j is specified using the standard dual token bucket regulator (see Fig. 4) $(\sigma^j, \rho^j, P^j, L^{j,max})$ where $\sigma^j \geq L^{j,max}$ is the maximum burst size of flow j, ρ^j is the sustained rate of flow j, P^j is the peak rate of flow j, and $L^{j,max}$ is the maximum packet size of flow j. Then, under the VTRS, we must ensure that packets of this flow will never be injected into the network core at a rate exceeding its reserved rate r^j (see Fig. 2). That is, the shaper in Fig. 5 ensures that, for a flow j with a reserved rate r^j, the inter-arrival time of two consecutive packets of the flow at the first hop core router satisfies Eq. (1).

Note that the edge shaper introduces an additional *edge delay* (due to shaping) to the flow. Such edge delay should be accounted for when measuring the end-to-end delay bound for the GS flow. Denote d^j_{edge} the maximum delay that packets of flow j experienced at the edge shaper. Assume that the flow has a reserved

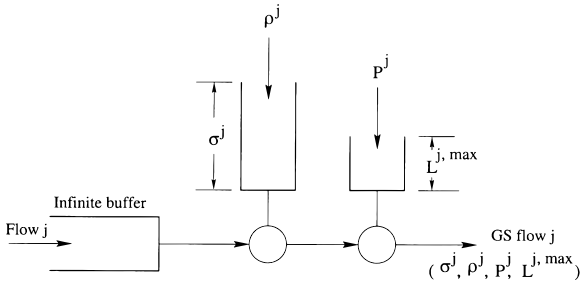


Fig. 4 A dual token bucket regulator for an GS flow j.

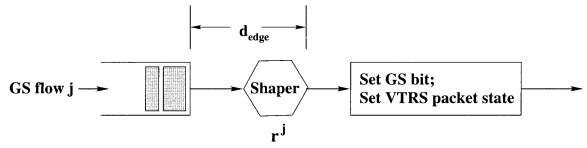


Fig. 5 Edge node shaping and marking for an GS flow.

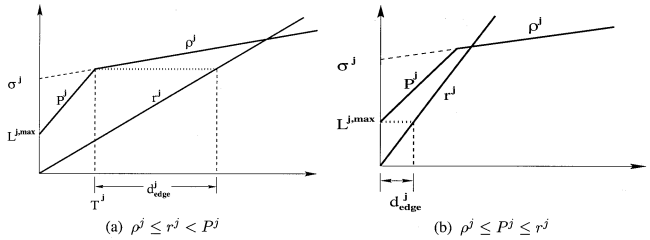


Fig. 6 Edge delay due to edge shaping for an GS flow.

rate r^j (see Sect. 3.4 for r^j calculation during admission control procedure). Then from Fig. 6, we have

$$d^j_{edge} = \begin{cases} \frac{P^j - r^j}{r^j} \cdot \frac{\sigma^j - L^{j,max}}{P^j - \rho^j} + \frac{L^{j,max}}{r^j} & \text{if } \rho^j \le r^j < P^j, \\ \frac{L^{j,max}}{r^j} & \text{if } \rho^j \le P^j \le r^j. \end{cases} \quad (10)$$

Denote T^j as

$$T^j = \frac{\sigma^j - L^{j,max}}{P^j - \rho^j} \quad (11)$$

if $\rho^j \le r^j < P^j$. Then T^j is the maximum duration that flow j can inject traffic at its peak rate (P^j) into the edge shaper if $\rho^j \le r^j < P^j$.

After the shaper, the packet is marked as an GS packet, with its VTRS states set in the packet header as follows (also see Sect. 2.1): (1) $r^j := r^j$, whose calculation is given in Sect. 3.4; (2) $\tilde{\omega}_1^{j,k} := \hat{a}_1^{j,k}$; and (3) $\delta^{j,k}$, which is computed at edge according to Eq. (3).

Edge Shaping/Marking for the PS Flows. For an PS flow j, it only has a peak rate requirement P^j as its traffic profile. According to [18], an PS flow should experience low delay jitter and low packet loss when it traverses the network domain. Under our architecture, we will offer the *same* treatment to an PS flow as that for an GS flow (albeit it does not have a delay bound requirement). More specifically, we will provide an PS flow j a reserved rate equal to its peak rate, i.e., $r^j = P^j$, and let it share the CSVC scheduler with the GS flows in the network core (see Sect. 2.2.2).

Given that we will treat an PS flow the same as if it were an GS flow, the edge traffic conditioning function for an PS flow is very much similar to that for an GS flow, except that traffic is shaped using the traffic's peak rate P^j. The shaper in Fig. 7 ensures that, for an PS flow j with a peak rate P^j, the inter-arrival time of two consecutive packets of the flow at the first hop core router is such that

$$\hat{a}_1^{j,k+1} - \hat{a}_1^{j,k} \ge \frac{L^{j,k+1}}{P^j}. \quad (12)$$

Similar to the GS, the edge shaper introduces an additional *edge delay* (due to shaping) to the flow, which should be accounted for when measuring the end-to-end delay bound for the PS flow. Denote d^j_{edge} the maximum delay that packets of PS flow j experienced at the edge shaper. Assume that the flow has a reserved rate $r^j = P^j$. Then we have

$$d^j_{edge} = \frac{L^{j,max}}{r^j}. \quad (13)$$

After the shaper, the packet is then marked as an PS packet, with VTRS states set into the packet as follows: (1) $r^j := P^j$; (2) $\tilde{\omega}_1^{j,k} := \hat{a}_1^{j,k}$; and (3) $\delta^{j,k}$, which is computed at edge according to Eq. (3).

Edge Marking for the AS Flows. For an AS flow, it only has a sustainable rate requirement ρ^j as its traffic profile and any packet exceeding such sustainable rate will be marked as an BE packet [18]. The edge traffic conditioning is shown in Fig. 8. Unlike for an GS or an PS flow, edge shaping is not performed on an AS flow [18]. The function of the edge conditioner is to examine (test) whether consecutive packets are properly spaced according to the sustainable rate ρ^j. That is,

$$\hat{a}_1^{j,k+1} - \hat{a}_1^{j,k} \ge \frac{L^{j,k+1}}{\rho^j}. \quad (14)$$

It is crucial that proper care needs to be taken when implementing (14). In particular, we must ensure that, in the implementation, the index, k (representing the kth packet) and arrival time $\hat{a}_1^{j,k}$ for the kth packet is updated in such a way that all the packets marked as AS satisfy (14). In particular, if the arrival time of a packet with current index $(k+1)$th satisfies this spacing condition, we will mark such packet as an AS packet and update the variables k with $(k+1)$ and $\hat{a}_1^{j,k}$ with $\hat{a}_1^{j,k+1}$. Otherwise, we will mark this packet as an BE packet and *do not update* the variables k and $\hat{a}_1^{j,k}$, both of which are maintained at the edge conditioner (along with the flow's requested sustainable rate ρ^j). Such update mechanism will ensure that consecutively marked AS packets of flow j satisfy (14) when they enter the core network.

Edge Marking for the BE Flows. Since there is no traffic profile and QoS requirements for an BE flow, the BE packets can enter the network without shaping. The edge conditioner only needs to set the BE bit pattern in the packet header and let it directly enter the network core.

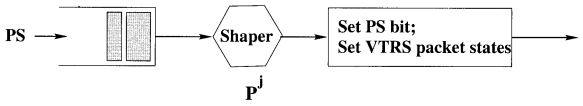


Fig. 7 Edge node shaping and marking for an PS flow.

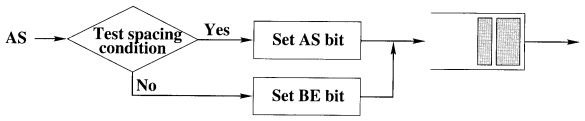


Fig. 8 Edge node marking for an AS flow.

2.2.2 Buffering and Scheduling at Edge and Core Nodes

The key building block in our node architecture is the CSVC scheduler, which is discussed in Sect. 2.1. Recall that CSVC is a work-conserving scheduling algorithm which services packets in the order of their virtual finish times. The virtual finish time of packet $p^{j,k}$ is given by $\tilde{\nu}^{j,k} = \tilde{\omega}^{j,k} + L^{j,k}/r^j + \delta^{j,k}$. As long as the total reserved rate of all flows traversing a CSVC scheduler does not

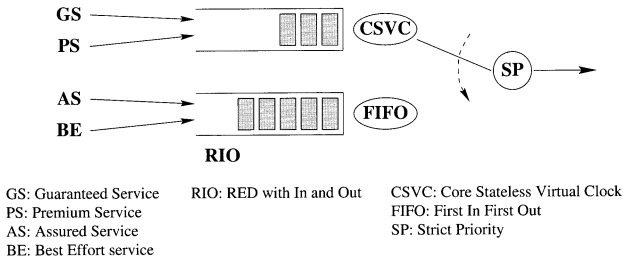


Fig. 9 A stateless node architecture for integrated support of the GS, the PS, the AS, and the BE services.

exceed its capacity (i.e., $\sum_j r^j \leq C$), the CSVC scheduler can guarantee each flow its reserved rate r^j with a minimum error term $\Psi = L^{*,max}/C$, where $L^{*,max}$ is the largest packet size among all the flows traversing the CSVC scheduler.

Figure 9 shows the schematic diagram of our node architecture. We maintain two separate buffers: one for the GS and the PS flows[†], and the other for the AS and the BE flows[††]. Both the GS and the PS traffic is serviced by the CSVC scheduler, while the AS and the BE traffic is serviced by the FIFO scheduler. A strict priority (SP) scheduler is employed between the CSVC and the FIFO queues.

Scheduling for the GS Traffic. Recall that the kth packet $p^{j,k}$ of an GS flow j arriving at the ith core router carries a virtual time stamp $\tilde{\omega}_i^{j,k}$, which represents the arrival time of this packet at the ith core router in the virtual time. Upon arriving at the CSVC queue, the virtual finish time of this packet is calculated by (7). Then this packet is inserted into the CSVC queue in such a way that the virtual finish time of all packets are in ascending order — the packet with the smallest virtual finish time is placed at the front of the queue and is serviced first.

Upon departure from the CSVC queue of the ith node, the virtual time stamp of packet $p^{j,k}$ is updated according to (8).

Note that the above *reference* (in (7)) (for scheduling) and *update* (in (8)) require the field for virtual time stamp to be "touched" twice — once for "read out," i.e., calculating virtual finish time, and once for "written into," i.e., updating virtual arrival time for the next hop (also see Fig. 3). A closer look reveals that these two separate operations are only necessary for conceptual purpose (under VTRS) and can be, for all practical purpose, combined (and thus simplified) into just one operation (by using (8)), which is particularly significant in actual hardware implementation. More specifically, since the term $(\Psi_i + \pi_i)$ is a *common term* added into virtual time stamp for *all* packets at the ith node (independent of the particular flow), we can use the final updated virtual arrival time for the next hop (i.e., $\tilde{\omega}_{i+1}^{j,k}$) directly for insertion (i.e., sorting) and scheduling in the CSVC queue — thus eliminating the step of calculating the virtual finish time $\tilde{\nu}_i^{j,k}$, which is used earlier for insertion and scheduling.

In light of the above observation, the following is a simplified version of the CSVC scheduling algorithm in our implementation. Upon the arrival of the kth packet $p^{j,k}$ of flow j arriving at the CSVC queue of the ith core router, update the packet's virtual time stamp $\tilde{\omega}_i^{j,k}$ with $\tilde{\omega}_{i+1}^{j,k}$ using (8). Then insert the packet into its position in the CSVC queue such that all packets are in increasing order of their virtual time stamp (i.e., $\tilde{\omega}_{i+1}^{j,k}$) — the packet carrying the smallest virtual time stamp being placed at the front of the queue and will be served first.

As far as the GS traffic is concerned, the lower priority queue (for the AS and the BE traffic) does not interfere (or has no effect on) the link access for the CSVC queue — due to the strict priority scheduling between the CSVC queue and the FIFO queue, and the fact that we have considered the error term of the CSVC scheduler. Denote d_{core}^j the maximum delay of all packets in GS flow j traversing the network core. Then d_{core}^j is given by (9), i.e.,

$$d_{core}^j = h\frac{L^{j,max}}{r^j} + \sum_{i=1}^{h}\frac{L^{*,max}}{C_i} + \sum_{i=1}^{h-1}\pi_i. \tag{15}$$

Denote $D_{tot}^{\mathcal{P}}$ as

$$D_{tot}^{\mathcal{P}} = \sum_{i=1}^{h}\frac{L^{*,max}}{C_i} + \sum_{i=1}^{h-1}\pi_i. \tag{16}$$

Note that $D_{tot}^{\mathcal{P}}$ is a path parameter and is independent of the particular GS flow j traversing this path. Combining (10) and (15), the maximum end-to-end delay, d_{e2e}^j, for all packets in flow j is then given by

$$\begin{aligned} d_{e2e}^j &= d_{edge}^j + d_{core}^j \\ &= \begin{cases} \frac{P^j - r^j}{r^j}\cdot\frac{\sigma^j - L^{j,max}}{P^j - \rho^j} + (h+1)\frac{L^{j,max}}{r^j} + D_{tot}^{\mathcal{P}} \\ \qquad\qquad \text{if } \rho^j \leq r^j < P^j, \\ (h+1)\frac{L^{j,max}}{r^j} + D_{tot}^{\mathcal{P}} \\ \qquad\qquad \text{if } \rho^j \leq P^j \leq r^j. \end{cases} \end{aligned} \tag{17}$$

Observe that the end-to-end delay formula in (17) is *precisely the same as* that specified in the IETF IntServ

[†]We assume that the buffer size for the GS and the PS are properly provisioned so that no GS or PS packet will be lost due to buffer overflow.

[††]Note that under the DiffServ model, the buffer for the GS and the PS traffic may be mapped to the EF PHB [16] while the buffer for the AS and the BE may be mapped to the AF PHB [14]. Since the specific implementations of EF and AF PHBs are left to the equipment vendors, node architecture and mechanisms other than the one proposed in this paper may also be employed.

GS [22] using the WFQ as the reference model! In this sense, the CSVC scheduler provides the same expressive power, in terms of supporting end-to-end delay guarantee, as a *stateful* WFQ scheduler, albeit it does not maintain any reservation or scheduling state in the router.

Scheduling for the PS Traffic. Recall that under our architecture, we treat an PS flow the same as an GS flow (albeit it does not have a delay bound requirement). That is, we will provide an PS flow j a reserved rate equal to its peak rate, i.e., $r^j = P^j$, and let it share the CSVC scheduler with the GS flows in the network core. Since the ith CSVC is a rate-based scheduler with an error term $L^{*,max}/C_i$, it will guarantee flow j its reserved rate P^j with an error term $L^{*,max}/C_i$.

As an extra benefit for using the CSVC for the PS traffic, each packet of an PS flow j has a *guaranteed* delay bound in the network core, i.e.,

$$d^j_{core} = h\frac{L^{j,max}}{P^j} + \sum_{i=1}^{h}\frac{L^{*,max}}{C_i} + \sum_{i=1}^{h-1}\pi_i = h\frac{L^{j,max}}{P^j} + D^{\mathcal{P}}_{tot}. \quad (18)$$

Combining (13) and (18), the maximum *end-to-end* delay, d^j_{e2e}, for all packets in PS flow j is then given by

$$d^j_{e2e} = d^j_{edge} + d^j_{core} = (h+1)\frac{L^{j,max}}{r^j} + D^{\mathcal{P}}_{tot}\ . \quad (19)$$

We point out that such an end-to-end delay bound offered by the CSVC scheduler effectively circumvents the problem associated with (FIFO) class-based SP scheduling, where it has been shown [2], [8] that the delay jitter can be unbounded over a certain utilization.

Scheduling and Buffer Management for the AS and the BE Traffic. For the AS and the BE service, we employ a simple FIFO queue, which is similar to that in [18]. The FIFO queue is serviced with a strictly lower priority than the CSVC queue (see Fig. 9).

To differentiate packets in the FIFO queue, *RED with In and Out* (RIO) [9] is employed as the buffer management mechanism†. RIO retains all the attractive features of RED [11] and has additional capability of dropping out-of-profile packets (corresponding to the BE packets under our architecture) during congestion.

A schematic diagram for the RIO mechanism is depicted in Fig. 10. RIO employs two RED algorithms for dropping packets, one for the in-profile packets (corresponds to the AS) and one for the out-of-profile packets (corresponds to the BE). By choosing the parameters for the respective RED algorithms differently, RIO is able to preferentially drop the BE packets and support the sustainable rates of the AS flows [9]. To see how this can be done, we first note that RIO starts to drop *all* BE packets when the average queue size exceeds Max_Out, while the AS packets can still enter the buffer while the buffer is being drained by the FIFO scheduler. If (1) we have provisioned enough buffer size and set the average buffer threshold Min_In and Max_In as large as possible; and (2) the link capacity is properly provisioned for the aggregate AS flows (see Sect. 3.4 on admission control for the AS flows), we expect that few AS packet will be dropped and the sustainable rate ρ^j for an AS flow j will be guaranteed (within a properly chosen time scale).

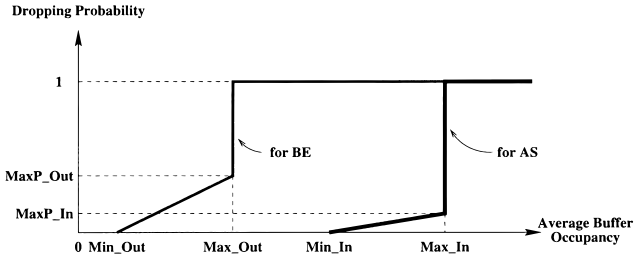


Fig. 10 Buffer management for the AS and the BE traffic with RIO mechanism.

3. Control Plane Operations

The previous section presents the network architecture on the data plane based on the VTRS/CSVC. An attractive feature of the VTRS/CSVC is that it enables the *decoupling* of QoS control functions from core routers, which helps to facilitate the design of a *centralized* bandwidth broker (BB) to control and manage resources in a network domain. In this section, we present control plane operations (in particular, the admission control procedure) in a BB. We organize this section as follows. Section 3.1 discusses the advantages of decoupling QoS control from core routers, which is made possible by the VTRS/CSVC. In Sect. 3.2, we give an overview of a BB. Section 3.3 presents an architectural design for a BB, in particular, those details pertinent to admission control procedure. In Sect. 3.4, we present the admission control procedure for the GS, the PS, and the AS flows.

3.1 Decoupling QoS Control from Data Plane

In the IETF DiffServ framework, a centralized model based on the notion of BB [18] has been proposed for the control and management of QoS provisioning. Under this centralized model, each network domain has a BB (a special network server) that is responsible for maintaining the network QoS states and performing various QoS control and management functions such as admission control, resource reservation and provisioning for the entire network domain. Issues in designing

†We refer interested readers to [13], [15] for more background on various buffer management mechanisms for the Internet.

and building such a centralized BB architecture have been investigated in several recent studies [1], [24].

In this section, we present the control plane operations performed by a BB for the integrated transport of the GS, the PS, the AS and the BE services[†]. This BB architecture relies on the VTRS to provide an QoS abstraction of the data plane. Each router in the network domain employs our node architecture (Fig. 9) at each of its output port, where the CSVC scheduler can be characterized by an error term ($\Psi = L^{*,max}/C$) under the VTRS. The novelty of our BB lies in that all QoS reservation and other QoS control state information (e.g., the amount of bandwidth reserved at a core router) is *removed* from core routers, and is solely maintained at and managed by the BB. In supporting the GS, the PS, the AS, and the BE services in a network domain, core routers perform no QoS control and management functions such as admission control, but only data plane functions such as packet scheduling and forwarding. In other words, the data plane of the network domain is *decoupled* from the QoS control plane. Despite the fact that all the QoS reservation states are removed from core routers and maintained solely at the BB, the proposed BB architecture is capable of supporting the GS with the same QoS granularity and expressive power as the IntServ/GS model. More important, this is achieved without the potential complexity and scalability problems of the IntServ model.

Some Advantages of Decoupling. Because of this decoupling of data plane and QoS control plane, our BB architecture is appealing in several aspects. First of all, by maintaining QoS reservation states only in the BB[††], core routers are relieved of QoS control functions such as admission control, making them potentially more efficient. Second, and perhaps more important, an QoS control plane that is decoupled from the data plane allows a network service provider to introduce new services without necessarily requiring software/hardware upgrades at core routers. In other words, it enables network services to evolve (more or less) *independently* from the underlying network infrastructure (data plane). Third, with QoS reservation states maintained by a BB, it can perform sophisticated (and thus powerful) QoS provisioning and admission control algorithms to optimize network utilization in a *network-wide* fashion. Such network-wide optimization is difficult, if not impossible, under the conventional hop-by-hop reservation set-up approach, where each core router makes its own admission control decisions based on local QoS information. Furthermore, because the network QoS states are centrally managed by the BB, the problems of *unreliable* or *inconsistent* control states [23] are effectively circumvented. This is in contrast to the IETF IntServ QoS control model based on RSVP [5], [26], where every router participates in hop-by-hop signaling for reserving resources and maintains its own QoS state database. Last but not the least, under our approach, the reliability, robustness and scalability issues of QoS control plane (i.e., the BB architecture) can be addressed *separately* from, and without incurring additional complexity to, the data plane[†††].

As an example to illustrate the flexibility that is made possible through the decoupling of QoS control from core routers. We show how various link sharing policy [12] can be easily implemented within a network domain, without incurring any hardware/software upgrades at core routers — only the policy software module in the BB needs to be changed/upgraded. We consider the following link sharing policies for the GS, the PS, the AS, and the BE services, one of which (i.e., Policy A) will be used in our simulation study in Sect. 5.

Policy A: Static Link Sharing Under this policy, the GS, the PS, and the AS each is limited to a *fixed* maximum percentage of link capacity, e.g., 10% for the GS, 20% for the PS, and 30% for the AS. That is, the GS, the PS, the AS cannot exceed its maximum capacity allocation on a link; any remaining capacity can be used by the BE traffic.

Policy B: Dynamic Link Sharing Under this policy, none of the services is limited to a fixed percentage of link capacity. Instead, the link capacity is completely shared by all four services, as long as the aggregate reserved rates for the GS, the PS, and the AS flows do not exceed the link's capacity.

Policy C: Hybrid Link Sharing This policy falls between policy A and policy B, both of which may be considered extreme cases of link sharing for the GS, the PS, the AS, and the BE services. Under policy C, the GS, the PS, and the AS each is guaranteed a certain percentage to share link capacity (similar to that under policy A), e.g., 10% for the GS, 10% for the PS, and 20% for the AS. The remaining capacity can be completely shared by all four types of services (similar to that under policy B), as long as the aggregate required rates for the GS, the PS, and the AS flows do not exceed the link's remaining shared capacity pool.

We do not want to make any comment here on which link sharing policy is better than the other — such matter is a network management policy issue and should be left to the network provider. The point that we want to stress is that, due to the decoupling of QoS control from core routers, policy issues such as link sharing can be easily set up and changed (if nec-

[†]Since the BE service does not have any specific QoS requirements, such flows do not go through a BB for call processing. Instead, the BE traffic can freely enter the network, just as the case under today's Internet.

[††]For simplicity, we assume that there is a single centralized BB for a network domain. In practice, there can be multiple BBs for a network domain to improve reliability and scalability [28].

[†††]In [28], we address the issue of scalable design of BBs.

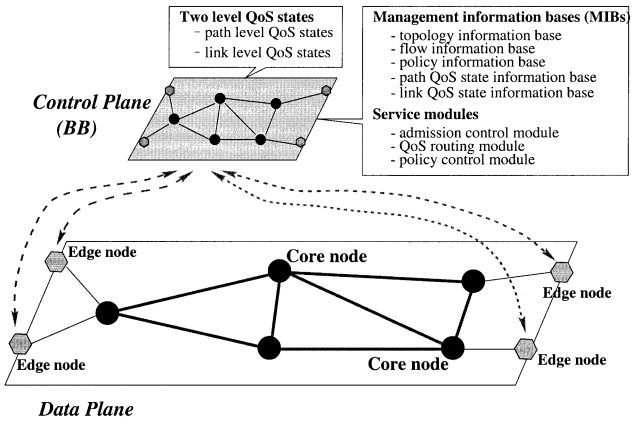


Fig. 11 Illustration of a bandwidth broker (BB) and its operation in a VTRS network domain.

essary) *solely* at the BB, and without incurring any hardware/software upgrade on any core router on the data plane. This is in contrast with the hop-by-hop approach, where any link sharing policy change would require *physical re-configuration* of hardware/software on all relevant core routers (on the data plane), e.g., the weights in class-based (CBQ) WFQ schedulers.

3.2 Bandwidth Broker: An Architectural Overview

As the basis for our study, we first give an overview of the basic centralized BB architectural model and describe how admission control is performed under such a model, thus, setting the stage for the rest of this section.

The basic centralized BB model is schematically depicted in Fig. 11. In this architectural model, the BB centrally manages and maintains a number of management information (data)bases (MIBs) regarding the network domain. As shown in Fig. 11, the BB consists of several modules such as admission control, QoS routing, and policy control. In this paper, we will focus primarily on the *admission control module.* The BB also maintains a number of management information (data)bases (MIB) for the purpose of QoS control and management of the network domain. For example, the *topology information base* contains topology information that the BB uses for route selection and other management and operation purposes; the *policy information base* contains policies (e.g., link sharing policy) and other administrative regulations of the network domain. Among them, the network topology database and network QoS state databases are most relevant to admission control. The network topology database and network QoS state databases together provide a *logical* representation (i.e., an *QoS abstraction*) of the network domain and its entire states. With this QoS abstraction of the network domain, the BB performs QoS control functions by managing and updating these databases. In this sense, the QoS control plane of the network domain is decoupled from its data plane. The core routers of the network domain are removed from the QoS control plane: core routers do not maintain any QoS reservation state, whether per-flow or aggregate, and do not perform any QoS control function such as admission control.

In our BB model, the network QoS states are represented at two levels: *link-level* and *path-level.* The link QoS state database maintains information regarding the QoS states of each link in the network domain, such as the total reserved bandwidth or the available bandwidth of the link. The path QoS state database maintains the QoS state information regarding each path of the network domain, which is *extracted* and *"summarized"* from the link QoS states of the links of the path. An example of the path QoS state is the available bandwidth along a path, which is the minimal available bandwidth among all its links. By maintaining a separate path-level QoS state, the BB can conduct fast admissibility test for flows routed along the path. Furthermore, path-wise resource optimization can also be performed based on the (summarized) path QoS state. Lastly, we note that both the link QoS states and path QoS states are *aggregate* QoS states regarding the links and paths. No per-flow QoS states are maintained in either of the two QoS databases — the QoS and other control state information regarding each flow such as its QoS requirement and reserved bandwidth is maintained in a separate *flow information database* managed by the BB.

We briefly discuss the basic operations of the BB, in particular, those pertinent to the *admission control module.* The details of the admission control procedure will be given in later part of this section. For illustration purpose, we use an GS flow, which is the most sophisticated among all the services.

When a new GS flow j with traffic profile $(\sigma^j, \rho^j, P^j, L^{j,max})$ and end-to-end delay requirement $D^{j,req}$ arrives at an ingress router. The ingress router sends a new flow service request message to the BB (using, e.g., COPS [4]). Upon receiving the service request, the BB first checks for policy and other administrative information bases to determine whether the new flow is admissible. If not, the request is immediately rejected. Otherwise, the BB selects a path[†] (from the ingress to an appropriate egress router in the network domain) for the new flow, based on the network topology information and the current network QoS state information, in

[†]If necessary, a new path may be set up dynamically. The problem of QoS routing, i.e., finding an "optimal" path for the flow can be handled relatively easily under our BB architecture. Since it is beyond the scope of this paper, we will not discuss it here. As an aside, our BB architecture can also accommodate *advance QoS reservation* in a fairly straightforward fashion, thanks to decoupling of the QoS control plane and data plane and the centralized network QoS state information bases.

addition to other relevant information (such as policy constraints applicable to this flow). Once the path is selected, the BB will invoke the admission control module to determine if the new flow can be admitted. The details of admission control procedure for each type of services will be given in Sect. 3.4. Generally speaking, the admission control procedure consists of two phases: (1) *admission control test* phase during which it is determined whether the new flow service request can be accommodated and how much network resources must be reserved if it can be accommodated; and (2) *bookkeeping* phase during which the relevant information bases such as the flow information base, path QoS state information base and node QoS state information base will be updated, if the flow is admitted. If the admission control test fails, the new flow service request will be rejected, no information bases will be updated. In either case, the BB will inform the ingress of the decision. In the case that the new flow service request is granted, the BB will also pass the QoS reservation information (e.g., reserved rate r^j for the GS flow) to the ingress router so that it can set up a new or re-configure an existing edge conditioner (which is assumed to be co-located at the ingress router) for the new flow. The edge conditioner will appropriately initialize and insert the packet states into packets of the new flow once it starts to send packets into the network. When an existing flow departs the network, relevant QoS control states and MIBs should also be updated.

3.3 Management Information Bases

In this subsection, we describe in detail the MIBs that will be used by the admission control module, and set the stage for our discussion on admission control procedure in the subsequent subsection.

Flow Information Base. This MIB contains information regarding individual flows such as service type (e.g., GS, PS, and AS), flow id., traffic profile (e.g., $(\sigma^j, \rho^j, P^j, L^{j,max})$ for GS, P^j for PS, ρ^j for AS), service profile (e.g., end-to-end delay requirement $D^{j,req}$ for the GS, peak rate requirement P^j for the PS, and sustainable rate requirement ρ^j for the AS), route id. (which identifies the path that a flow j traverses in the network domain) and QoS reservation (r^j in the case of the GS, P^j for the PS, and ρ^j for the AS) associated with each flow. Other administrative (e.g., accounting and billing) information pertinent to a flow may also be maintained here.

Network QoS State Information Bases. These MIBs maintain the QoS states of the network domain, and thus are the key to the QoS control and management of the network domain. Under our BB architecture, the network QoS state information is represented in two-levels using two separate MIBs: *path QoS state information base* and *link QoS state information base*. These two MIBs are presented in detail below.

Path QoS state information base maintains a set of paths (each with a route id.) between various ingress and egress routers of the network domain. These paths can be pre-configured or dynamically set up[†]. Associated with each path are certain *static* parameters characterizing the path and *dynamic* QoS state information regarding the path. Examples of static parameters associated with a path $\mathcal{P}$ are the number of hops h on $\mathcal{P}$, sum of the router error terms of all the CSVC schedulers and propagation delay along $\mathcal{P}$, $D^{\mathcal{P}}_{tot}$ (see (16)), and the maximum permissible packet size (i.e., MTU) $L^{\mathcal{P},max}$. The dynamic QoS state information associated with $\mathcal{P}$ include, among others, the set of flows traversing $\mathcal{P}$ (i.e., the GS, the PS, and the AS flows) and a number of QoS state parameters regarding the (current) QoS reservation status of $\mathcal{P}$ such as the minimal remaining (residual) bandwidth along $\mathcal{P}$ for each type of services, i.e., $C^{GS}_{\mathcal{P},res}$ for the GS, $C^{PS}_{\mathcal{P},res}$ for the PS, and $C^{AS}_{\mathcal{P},res}$ for the AS.

Link QoS state information base maintains information regarding the router links in the network domain. Associated with each router link is a set of static parameters characterizing the router and a set of dynamic parameters representing the router's current QoS states. Examples of static parameters associated with a router link are its error term(s) $\Psi = L^{*,max}/C$, propagation delays to its next-hop routers π's, configured total bandwidth and buffer size. The dynamic router QoS state parameter includes the current residual bandwidth and buffer size at the link for each type of services.

3.4 Admission Control

We present a *path-oriented* approach to perform efficient admission control test and resource allocation. Unlike the conventional *hop-by-hop* approach which performs admission control *individually* based on the *local QoS state* at each router along a path, this path-oriented approach examines the resource constraints *along the entire path simultaneously*, and makes admission control decision accordingly. As a result, we can significantly reduce the time of conducting admission control test. Clearly, such a path-oriented approach is

[†]Note that during the process of a path set-up, no admission control test is administered. The major function of the path set-up process is to configure forwarding tables of the routers along the path, and if necessary, provision certain scheduling/queue management parameters at the routers, depending on the scheduling and queue management mechanisms deployed. Hence we refer to such a path a *traffic engineered* (TE) path. Set-up of such an TE path can be done by using a path set-up signaling protocol, say, MPLS [7], [21], or a simplified version (*minus* resource reservation) of RSVP.

possible because the availability of QoS state information of the entire path at the BB.

For the rest of this subsection, we give details on admission control for the GS, the PS, and the AS flows. For the ease of exposition, we employ *static link sharing* policy discussed earlier. That is,

$$\mu_i^{GS} + \mu_i^{PS} + \mu_i^{AS} < 1, \tag{20}$$

where μ_i^{GS}, μ_i^{PS}, and μ_i^{AS} are *fixed* maximum allowed percentage share on the ith link for the GS, the PS, and the AS, respectively. Note that there is no fixed allocation for the BE flows since they are not subject to admission control and can freely enter the network and share any remaining network bandwidth.

Admission Control for the GS Flows. As far as the GS flows are concerned, they are served by a network of CSVC schedulers — since each CSVC queue in our node architecture (Fig. 9) is given strict priority (SP) over the other FIFO queue. Let $j \in \mathcal{F}_i^{GS}$ denote that an GS flow j currently traverses the ith node and C_i^{GS} be the total bandwidth at ith node allocated to the GS flow, i.e., $C_i^{GS} = \mu_i^{GS} C_i$. Then as long as $\sum_{j \in \mathcal{F}_i^{GS}} r^j \leq C_i^{GS}$, the ith node can guarantee each GS flow j its reserved bandwidth r^j. We use $C_{i,res}^{GS}$ to denote the residual bandwidth at the ith node for the GS flows, i.e., $C_{i,res}^{GS} = C_i^{GS} - \sum_{j \in \mathcal{F}_i^{GS}} r^j$. We consider the two phases of the admission control procedure.

1. Admission Test. Let $(\sigma^\nu, \rho^\nu, P^\nu, L^{\nu,max})$ be the traffic profile of a new GS flow ν, and $D^{\nu,req}$ be its end-to-end delay requirement. Let h be the number of hops along $\mathcal{P}$, the path for the new flow. From (17), in order to meet its end-to-end delay requirement $D^{\nu,req}$, the reserved rate r^ν for the new flow ν must satisfy the delay constraint $d_{e2e}^\nu \leq D^{\nu,req}$. That is,

$$\begin{aligned} D^{\nu,req} &\geq d_{e2e}^\nu \\ &= \begin{cases} \frac{P^\nu - r^\nu}{r^\nu} \cdot \frac{\sigma^\nu - L^{\nu,max}}{P^\nu - \rho^\nu} + (h+1)\frac{L^{\nu,max}}{r^\nu} + D_{tot}^{\mathcal{P}} \\ \qquad\qquad\qquad\qquad \text{if } \rho^\nu \leq r^\nu < P^\nu, \\ (h+1)\frac{L^{\nu,max}}{r^\nu} + D_{tot}^{\mathcal{P}} \quad \text{if } \rho^\nu \leq P^\nu \leq r^\nu. \end{cases} \end{aligned} \tag{21}$$

Let r_*^ν be the smallest r^ν that satisfies (21) and recall that $T^\nu = (\sigma^\nu - L^{\nu,max})/(P^\nu - \rho^\nu)$ (see (11)). Then we have

$$r_*^\nu = \begin{cases} \frac{T^\nu P^\nu + (h+1)L^{\nu,max}}{D^{\nu,req} - D_{tot}^{\mathcal{P}} + T^\nu} & \text{if } \rho^\nu \leq r_*^\nu < P^\nu, \\ \frac{(h+1)L^{\nu,max}}{D^{\nu,req} - D_{tot}^{\mathcal{P}}} & \text{if } \rho^\nu \leq P^\nu \leq r_*^\nu. \end{cases} \tag{22}$$

If the above solution for r_*^ν exist, then we have a feasible reserved rate requirement to satisfy the flow's end-to-end delay requirement in (21). Otherwise, the flow is not admissible on this path.

Figure 12 outlines the procedure to calculate the reservation rate r_*^ν for a new flow ν requesting GS. According to Fig. 12, we first check whether the path parameter $D_{tot}^{\mathcal{P}}$ is greater than or equal to the delay requirement $D^{\nu,req}$. If this is true, then no feasible reservation rate r_*^ν exist. If $D_{tot}^{\mathcal{P}}$ is less than $D^{\nu,req}$, we first try to use the upper formula in (22) to calculate r_*^ν (lines 9 to 12 in Fig. 12). If such r_*^ν can be found, we are done; otherwise, we will use the lower formula in (22) (line 14 in Fig. 12) to calculate r_*^ν.

```
0.  Upon the arrival of a new flow ν on path P requesting
1.  GS:
2.      Flow ν's traffic profile: (σ^ν, ρ^ν, P^ν, L^{ν,max});
3.      Delay requirement: D^{ν,req};
4.      Path P: h hops.
5.  if D_tot^P ≥ D^{ν,req}
6.    /* no feasible r_*^ν exist */
7.    End.
8.  else /* D_tot^P < D^{ν,req}, feasible r_*^ν exists */
9.    T^ν = (σ^ν − L^{ν,max}) / (P^ν − ρ^ν);
10.   r_*^ν = (T^ν P^ν + (h+1)L^{ν,max}) / (D^{ν,req} − D_tot^P + T^ν);
11.   if ρ^ν ≤ r_*^ν < P^ν
12.     return r_*^ν;
13.   else /* r_*^ν ≥ P^ν */
14.     r_*^ν = (h+1)L^{ν,max} / (D^{ν,req} − D_tot^P);
15.     return r_*^ν;
16. End.
```

Fig. 12 Computation of the reservation rate for a new flow for GS.

Next, we examine if the path has sufficient remaining bandwidth to accommodate this new rate request. That is, r_*^ν must not exceed the minimal residual bandwidth $C_{\mathcal{P},res}^{GS}$ along path $\mathcal{P}$ for the GS, where $C_{\mathcal{P},res}^{GS} = \min_{i \in \mathcal{P}} C_{i,res}^{GS}$ is maintained (as a path QoS parameter associated with $\mathcal{P}$) in the path QoS state MIB. If this condition cannot be satisfied, the service request of the new flow ν must be rejected. Otherwise, it is admissible, and r_*^ν is the *minimum* feasible reserved rate for the new flow ν. Given that the path QoS parameters $D_{tot}^{\mathcal{P}}$ and $C_{res}^{\mathcal{P}}$ associated with $\mathcal{P}$ are maintained in the path QoS state MIB, the above admission test can be done in $O(1)$.

2. Bookkeeping. If the new GS flow ν is admitted into the network, several MIBs (e.g., the flow MIB, the path and link QoS state MIBs) must be updated. The flow id., traffic profile and service profile of the new flow will be inserted into the flow MIB. The minimal residual bandwidth $C_{\mathcal{P},res}^{GS}$ will be subtracted by r^ν, the reserved rate for the new GS flow ν. Similarly, for each link i along $\mathcal{P}$, its residual bandwidth $C_{i,res}^{GS}$ will also be subtracted by r^ν. Furthermore, for any path $\mathcal{P}'$ that traverses $\mathcal{S}_i$, $i \in \mathcal{P}$, its minimal residual bandwidth $C_{res}^{\mathcal{P}',GS}$ may also be updated, depending on whether the update of $C_{i,res}^{GS}$ changes $C_{\mathcal{P}',res}^{GS}$. Provided that a powerful (and perhaps, parallel) database system is employed, these database update operations can be performed in very short time. Note that when an existing flow departs the network, the relevant MIBs should also

be updated.

Admission Control for the PS Flows. The admission control for the PS flows is simpler than that for the GS flows since a new PS flow ν can explicitly submit a reserved rate requirement, i.e., its peak rate requirement P^ν. Thus, in contrary to the case for a new GS flow, no calculation of the reserved rate is needed for a new PS flow request. Similar to admission control for the GS flows, a path-oriented approach can be applied to a new PS flow.

1. Admission Test. Let peak rate P^ν be the traffic profile of a new PS flow ν. To admit the new PS flow ν, P^ν must not exceed the minimal residual bandwidth $C^{PS}_{\mathcal{P},res}$ along path $\mathcal{P}$ for PS, where $C^{PS}_{\mathcal{P},res} = \min_{i\in\mathcal{P}} C^{PS}_{i,res}$ is maintained (as a path QoS parameter associated with $\mathcal{P}$) in the path QoS state MIB. If this condition cannot be satisfied, the service request of the new PS flow ν must be rejected. Otherwise, it is admissible, and P^ν will be the reserved rate for the new flow ν.

2. Bookkeeping. If the new flow ν is admitted into the network, several MIBs (e.g., the flow MIB, the path and link QoS state MIBs) must be updated. Such operations are very similar to that for the GS flows.

Admission Control for the AS Flows. The admission control for an AS flow is very similar to that for an PS flow, except that the traffic profile and service profile for a new AS flow ν is its sustainable rate ρ^ν. Due to paper length limitation, we omit to discuss it further.

4. Discussions

In this section, we discuss the performance and complexity of our architecture.

4.1 Performance

Under our architecture, core routers of the network domain are relieved from the QoS control functions: core routers do not maintain any QoS reservation state, whether per-flow or aggregate. Therefore, our core network is *core-stateless* and fulfills our second design requirement — *scalability.*

Since our network architecture builds upon the VTRS, which is capable of providing an QoS abstraction of the network domain, we can use a centralized BB to perform QoS control functions by managing and updating relevant databases. Therefore, the QoS control plane of the network domain is *decoupled* from its data plane. Furthermore, any new policy or service can be introduced solely by appropriate software installation/update in the BB, without incurring any hardware/software re-configuration on the core routers. Therefore, our architecture meets our third design requirement — *decoupling QoS control from core routers for flexible resource allocation and QoS provisioning.*

Now we show that our architecture also meets our first design requirement — *integrated transport of the GS, the PS, the AS, and the BE services with QoS guarantees for each service.* The following corollary summarizes the behavior of admitted GS and PS flows under our node architecture.

Corollary 1.1: An VTRS network domain where each core routers employing our node architecture (Fig. 9) provides guaranteed rate and end-to-end delay to each admitted GS and PS flows, respectively. In particular, we have: (1) For an admitted GS flow j, the end-to-end delay of each packet is bounded by (17); and (2) For an admitted PS flow j, its peak rate P^j is guaranteed. Moreover, the end-to-end delay of each packet of the admitted PS flow j is bounded by (19).

Proof: As far as an admitted GS or PS flow is concerned, it is served by a network of CSVC schedulers. The lower priority queue for the AS and the BE traffic does not interfere (or has no effect on) the link access for the CSVC scheduler. Given the fact that we have considered the error term of the CSVC scheduler, the corollary then follows from Theorem 1. □

Now we examine the performance for the AS and the BE flows. Note that at node i, although the AS and the BE share the same FIFO queue, which is served with a strictly lower priority than the CSVC queue, the rate allocated to the FIFO queue is at least $(C_i - C_i^{GS} - C_i^{PS})$, which is *strictly greater than* C_i^{AS}, the allocated rate to the AS flows (see (20)). Now we show how the RIO mechanism can provide each AS flow j its sustainable rate ρ^j. Suppose we have sufficient buffer size so that we can set the average buffer thresholds for the AS (i.e., Min_In and Max_In) reasonably large and average buffer thresholds for the BE (i.e., Min_Out and Max_Out) reasonably small (see Fig. 10). Then all of the BE packets will be dropped if the average buffer occupancy (aggregated traffic of the AS and the BE) exceeds Max_Out. Once the average buffer occupancy reaches or exceeds this point, any BE traffic will cease to enter the buffer (until the average buffer occupancy decreases below Max_Out) — but the AS packets can still enter the buffer while the buffer is being drained by the FIFO scheduler, with a rate of *at least* $(C_i - C_i^{GS} - C_i^{PS})$, which is greater than C_i^{AS}. Therefore, the sustainable rate for each AS can be guaranteed within a certain time scale.

4.2 Complexity

We first discuss implementation complexity on the data plane. A crucial question on implementing the VTRS/CSVC is the overhead of *packet state encoding.* The packet state contains three parameters: (1) packet virtual time stamp $\tilde{\omega}^{j,k}$, (2) reserved rate r^j, and (3) virtual time adjustment term $\delta^{j,k}$. There are several options that we can use to encode the packet state: using an IP option, using an MPLS label, using the IP

fragment offset field. Issues on efficient packet state encoding will be investigated in our future work.

Another complexity associated with the VTRS/CSVC is per-packet processing[†] at core routers: (1) packet state lookup, (2) virtual time stamp update (for the next hop), and (3) insertion (i.e., sorting) into the CSVC queue. The last operation, i.e., packet insertion (i.e., sorting) in the CSVC queue, is perhaps the most complex step in hardware implementation. We believe that it is possible to reduce the complexity associated with sorting by using a notion of *slotted virtual time*. Such simplified implementation is similar to the *rotation priority queue* introduced by Liebeherr and Wrege [17], which is a trade-off between *performance* (delay bound) and *implementation complexity* (sorting).

We now discuss complexity associated with the BB on the control plane. There are several aspects regarding scalability that need to be addressed: (1) link/path QoS control states updates after either admitting a new GS/PS/AS flow or the departure of an existing flow; (2) the size of flow information base used to maintain each flow's traffic profile and service profile; and (3) the potential bottleneck between the BB's input/output interface and flow requests sent from all the edge routers. To address these issues, a *path-oriented* and *quota-based* approach (or "PoQ" for short) has been proposed [28]. It has been shown [28] that such PoQ approach can be used to design multiple and hierarchical BB architecture, under which all the issues raised above regarding scalability and reliability of a centralized BB can be satisfactorily resolved. Since the focus of this paper is on the performance of data plane for scalable and integrated support of the GS, the PS, the AS, and the BE traffic, we will not elaborate further on scalable design of the BBs. But we stress that, due to the decoupling of data plane and control plane, which is made possible by the VTRS, issues related to the design of BBs can be addressed *separately* from, and without incurring additional complexity to, the data plane.

5. Simulation Investigation

In this section, we conduct simulations to investigate the performance of the integrated framework for supporting diverse service guarantees. The focus of this simulation study is twofold: (1) the effectiveness of the proposed node architecture in satisfying different user requirements; and (2) the efficacy of the BB in controlling the network load for traffic requiring different service guarantees.

5.1 Simulation Settings

The network topologies that we will use for the simulations are depicted in Figs. 13, 14, and 15, which are referred to as the *peer-to-peer*, *parking-lot*, and *chain* network, respectively. In these networks, a node labeled with Ii denotes an ingress edge router i, Cj a core router j, Ek an egress edge router k, and Gn an end host group n[††]. Flows generated from a source node group Gn (S) will be destined to the destination node group Gn (D), traversing the shortest path from the source node to the destination node.

Fig. 13 A peer-to-peer network.

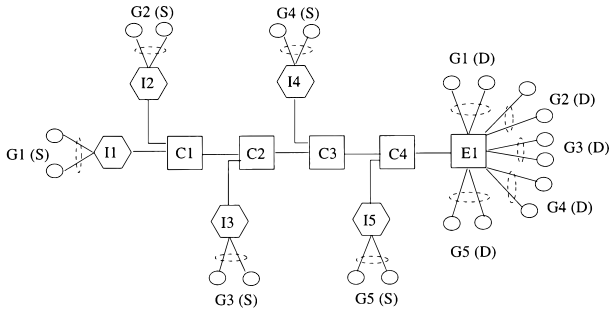


Fig. 14 A parking-lot network.

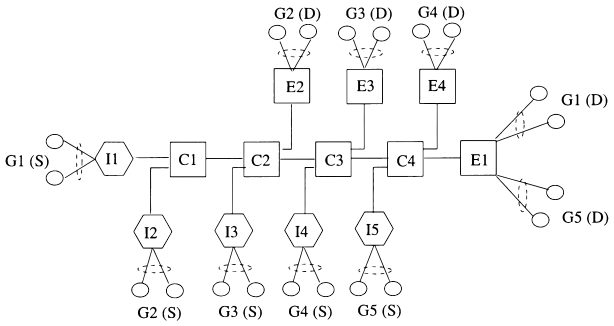


Fig. 15 A chain network.

Each ingress edge router (Ii) contains two functionality blocks: an edge conditioner (see Sect. 2.2.1) and the node architecture in Fig. 9. On the other hand, each core router (Cj) employs only the node architecture in Fig. 9.

For all network configurations, the capacity of the links between an end host and an edge (either an ingress or egress) router is assumed to be infinite and the propagation delay is negligible. In the peer-to-peer network,

[†]Such complexity comes from the fact that, in order to achieve scalability, we are making a trade-off between *time* (i.e., per-packet processing) and *space* (i.e., per-flow reservation state and per-flow scheduling state maintained at a core router as in the case of WFQ scheduling).

[††]Under the peer-to-peer network, we only have one group of flows (i.e., $n = 1$) traversing G1(S) to G1(D), whereas under the parking-lot or chain network, there are multiple groups of flows traversing different paths of the respective network.

Table 1 Traffic patterns for a flow used in the simulation study.

Traffic name	Traffic type	Bucket depth (σ) in bit (b)	Mean rate (ρ) (kb/s)	Peak rate (P) (kb/s)	Mean on time in seconds	Mean off time in seconds	Packet size in byte (B)	Window size (packets)
exp1	UDP/EXP	1536	16	64	0.25	0.75	96	–
exp2	UDP/EXP	20480	128	512	0.25	0.75	512	–
exp3	UDP/EXP	–	16	64	0.25	0.75	96	–
exp4	UDP/EXP	–	128	512	0.25	0.75	512	–
ftp	TCP/FTP	–	–	–	–	–	1024	50

the propagation delay of the link between I1 and E1 is 10 ms. While in the parking-lot and chain networks, all the links between routers have a propagation delay equal to 5 ms. We will specify the capacity of the links between the routers when we present the simulation results.

We assume that the CSVC queue in the node (Fig. 9) has infinite buffer size (i.e., it will never drop packets). On the other hand, the maximum buffer size of the low priority RIO queue is set to 100 packets. For the RIO queue, the buffer configuration for the *in-profile* AS traffic is (40, 70, 0.02), i.e., the minimum buffer threshold (Min_In) is 40 packets, the maximum buffer threshold (Max_In) is 70 packets, and the maximum dropping probability ($MaxP\_In$) is 2%. Meanwhile, the buffer configuration for the *out-of-profile* traffic is (10, 30, 0.5), i.e., the minimum buffer threshold (Min_Out) is 10 packets, the maximum buffer threshold (Max_Out) is 30 packets, and the maximum dropping probability ($MaxP\_Out$) is 50%. The weight parameter (*q_weight*) [9] used for estimating the average queue lengths is set to 0.02.

The traffic patterns used for the flows in our simulations are listed in Table 1. The fields marked as "–" means that it is not applicable to the corresponding traffic type. As shown in the table, the *exponential* on-off traffic types *exp1* and *exp2* are token bucket constrained, while *exp3* and *exp4* are not. For the traffic type *ftp,* the TCP version is *Reno,* and there is infinite traffic to be sent, i.e., persistent source. As shown in Fig. 4, if a flow requests GS, then a dual-token bucket regulator is attached to the traffic source to ensure that the traffic conforms to the $(\sigma, \rho, P, L^{max})$ traffic envelop. There is an infinite buffer at the entrance of the dual-token bucket regulator to hold the out-of-constraint traffic of the flow, i.e., out-of-constraint packets from the source is shaped instead of being dropped.

5.2 Simulation Results

We present simulation results as follows.

5.2.1 Data Plane

In this first set of simulations, we will examine the effectiveness of the proposed node architecture. The performance metrics that are of interest are the end-to-end packet delay, traffic throughput, and packet loss rate.

In this set of simulations, the capacity of the links between edge (ingress or egress) routers and core routers, and the links between core routers is 10 Mb/s in all three network configurations. We employ *static link sharing* policy and set the targeted traffic load on a link for each service type (either GS, PS, or AS) to be 30%, i.e., $\mu_i^{GS} = \mu_i^{PS} = \mu_i^{AS} = 0.3$ for link i. The simulated time is 200 seconds, of which the first 100 seconds are the simulation warm-up period.

To examine the key properties of the proposed node architecture, flows with a traffic profile and service requirement will be generated randomly between the time interval [0, 1] seconds from a source node group Gn (S) to a destination group Gn (D), and once admitted by the BB, they will never terminate, i.e., they have infinite holding time. Such long holding time of a flow will provide us a steady period where various packet level statistics (e.g., per-packet delay, flow throughput, and packet loss rate) can be collected and analyzed[†].

For the BE traffic, we only activate four ftp flows randomly within the time interval [0, 0.5] seconds from the source node group G1(S) to the destination group G1(D); while between other source and destination groups (in the case of parking-lot and chain network configurations), two BE ftp flows are randomly activated within the time interval [0, 0.5] seconds.

To demonstrate packet-level QoS performance for a flow with a particular traffic profile and service requirement, we focus on the admitted flows traversing the path G1(S) to G1(D), and purposely choose only 12 flows among all the admitted flows along this path to present our simulation results. These 12 flows are listed in Table 2 and represent traffic profile and service profile of all four types of services of our interest. In particular, flows 1 to 8 require a particular service guarantee while flows 9 to 12 are BE ftp traffic, which do not require any service guarantee.

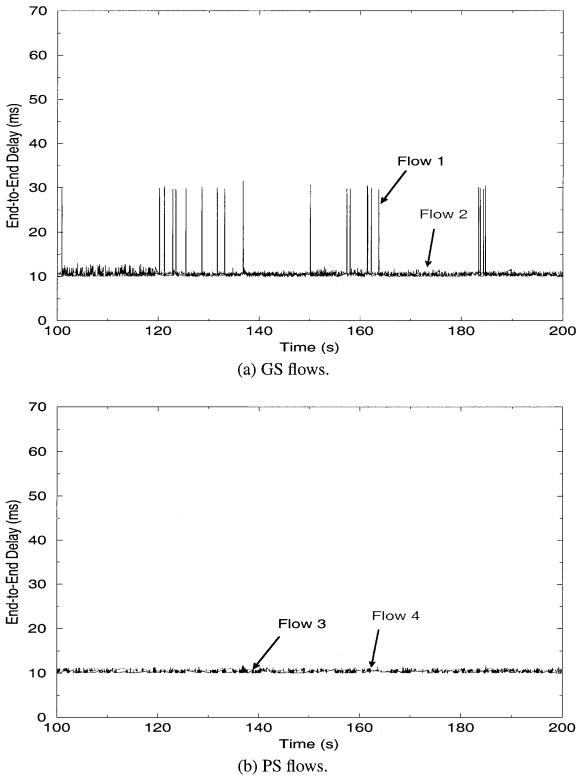
For a flow requesting GS, the BB will use the algorithm in Fig. 12 to calculate the reservation rate. As an example, in the parking-lot network, flow 1 (see Table 2), which traverses the path from G1(S) to G1(D) and has a delay requirement of 120 ms, we obtain a reservation rate of 50.926 Kb/s with Fig. 12.

Figure 16(a) shows the end-to-end packet delays of flows 1 and 2, which request GS in the peer-to-peer

[†]In the next subsection, when we investigate the performance of the BB for controlling the network load for each type of service, we will use shorter flow holding time so as to generate dynamics on the flow level.

Table 2 Traffic profiles and service requirements of the 12 flows chosen on the path traversing G1(S) to G1(D) in the simulations.

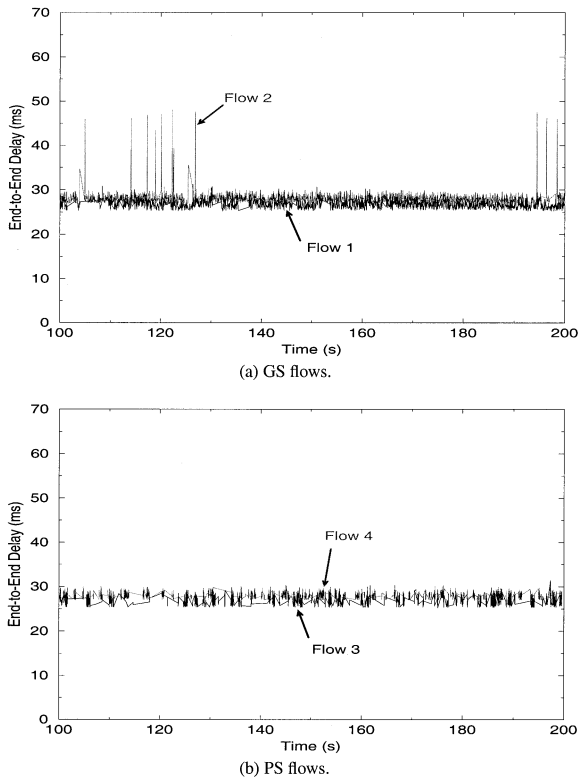
Flow id.		1	2	3	4	5	6	7	8	9 – 12
Required service		GS	GS	PS	PS	AS	AS	AS	AS	BE
Traffic profile		exp1	exp2	exp3	exp4	exp3	exp4	exp1	exp2	ftp
For GS, end-to-end delay bound requirement (ms)	peer-to-peer	100	100	–	–	–	–	–	–	–
	parking-lot	120	120	–	–	–	–	–	–	–
	chain	120	120	–	–	–	–	–	–	–



(a) GS flows.

(b) PS flows.

Fig. 16 End-to-end packet delays of the GS/PS flows on the path traversing G1(s) to G1(D) in the peer-to-peer network.



(a) GS flows.

(b) PS flows.

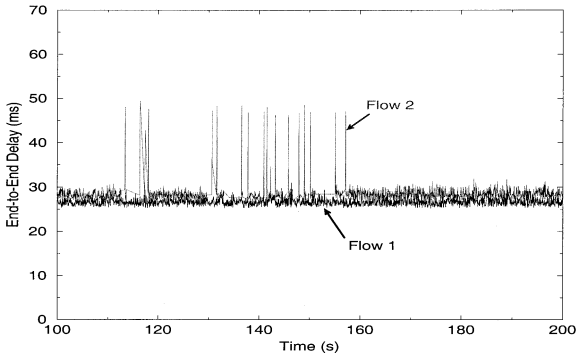
Fig. 17 End-to-end packet delays of the GS/PS flows on the path traversing G1(s) to G1(D) in the parking-lot network.

network. Comparing the end-to-end delay requirement listed in Table 2 (100 ms), we observe that the delay requirements of both flows are satisfied. Figure 16(b) presents the end-to-end packet delays for the two PS flows 3 and 4 in the same simulation. Recall that one of the objectives of the PS is to achieve a relatively small packet queueing delay throughout the network domain. From Fig. 16(b), we see that the traffic of the PS flows experiences almost constant delay, with almost no jitter! It is interesting to note that even though both GS and PS traffic share the same high priority queue, the end-to-end packet delay jitter of GS traffic is relatively higher than that of the PS traffic. A close examination reveals that the larger delay jitter of the GS traffic comes from the more conservative traffic shaping at the network edge (see (10)). That is, the GS traffic is released into the network according to the reserved rate of the flow, while PS traffic is released according to its peak rate. Therefore, a relatively longer packet queue can accumulate at the *edge conditioner* for an GS flow.

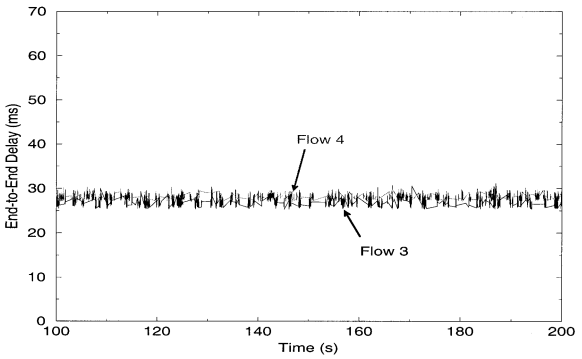
The same observations on the end-to-end packet delays hold for the simulations with the parking-lot and chain network configurations, each of which are presented in Figs. 17 and 18, respectively.

So far we have confirmed that the end-to-end delay requirements for the GS and PS are satisfied. Next, we investigate the traffic throughput and packet loss rate of the 12 flows during the same simulation run under the three network configurations.

Table 3 presents the corresponding data with the peer-to-peer network for the simulation run (i.e., 100 s – 200 s). As expected, there is no packet lost in both GS and PS flows. Therefore, we omit to list the packet loss ratio (0) for the GS and the PS flows in the table. From the table, we see that our framework provides the protection for the GS, the PS, and the AS traffic from the burst of the BE ftp traffic, therefore, achieving some degree of rate guarantees. Note that flows 5 and



(a) GS flows.



(b) PS flows.

Fig. 18 End-to-end packet delays of the GS/PS flows on the path traversing G1(s) to G1(D) in the chain network.

6 are burst traffic, which are more aggressive than the token bucket constrained flows 7 and 8. That is, in flows 5 and 6, there are considerable packets marked as BE traffic at the network edge because of their bursty behavior; while in flows 7 and 8, few packets are marked as BE traffic. Therefore, flows 5 and 6 have a higher dropping rate than flows 7 and 8. It is worth noting that all the dropped packets in flows 5 and 6 are *out-of-profile,* therefore marked as BE packets. Thus, by properly configuring the RIO, we can indeed protect the *in-profile* AS packets from both *out-of-profile* (although they are within the same AS flow, they are actually marked as BE traffic) and BE ftp traffic.

In Table 3, flows 9 to 12 are BE flows. We see that these BE flows have a similar packet dropping rate. Moreover, their throughput are reasonably close to each other. Thus, we conclude that the residual bandwidth of the link is distributed among the BE ftp flows in a relatively fair manner, which is achieved by the *random* early dropping property of the RED buffer management mechanism.

Again, the same observations on the traffic throughput and packet loss rate hold for the simulations with the parking-lot and chain network configurations, which are listed in Tables 4 and 5, respectively.

Table 3 Flow throughput (kb/s) and packet loss rates for the 12 flows chosen from the path traversing G1(S) to G1(D) in the peer-to-peer network.

GS/PS		AS			BE		
Flow id.	Throughput	Flow id.	Throughput	Loss rate	Flow id.	Throughput	Loss rate
1	14.29	5	14.08	2.7%	9	792.66	5.1%
2	128	6	112.07	2.6%	10	858.60	4.5%
3	22.80	7	16	0	11	845	4.8%
4	124.48	8	108.34	0	12	825.92	4.8%

Table 4 Flow throughput (kb/s) and packet loss rates for the 12 flows chosen from the path traversing G1(S) to G1(D) in the parking-lot network.

GS/PS		AS			BE		
Flow id.	Throughput	Flow id.	Throughput	Loss rate	Flow id.	Throughput	Loss rate
1	13.66	5	15.58	5.2%	9	314.49	7.0%
2	115.92	6	120.3	4.0%	10	293.68	7.1%
3	13.26	7	16	0	11	306.3	6.7%
4	122.47	8	119.6	0	12	310.07	7.0%

Table 5 Flow throughput (kb/s) and packet loss rates for the 12 flows chosen from the path traversing G1(S) to G1(D) in the chain network.

GS/PS		AS			BE		
Flow id.	Throughput	Flow id.	Throughput	Loss rate	Flow id.	Throughput	Loss rate
1	16	5	15.74	4.6%	9	335.63	5.7%
2	103.92	6	107.27	4.3%	10	334.4	6.5%
3	14.78	7	14.45	0	11	355.29	6.0%
4	124.15	8	119.44	0	12	340.21	6.5%

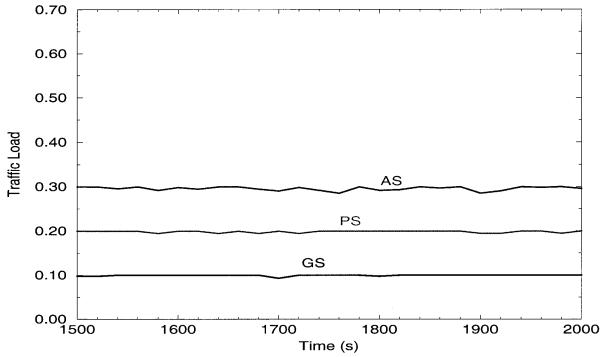
5.2.2 Control Plane

In the second set of simulations, we examine the capability of the BB entity in controlling the network load for each type of services. Because, among all the links in the network, only the bottleneck link will play a role in flow admission control, we will only conduct simulations with the peer-to-peer network (Fig. 13) here. In the following simulations, the capacity of the link between the ingress router $I1$ and the egress router $E1$ is set to 100 Mb/s. The simulated times are 2000 seconds, of which the first 1500 seconds are the simulation warm-up period.

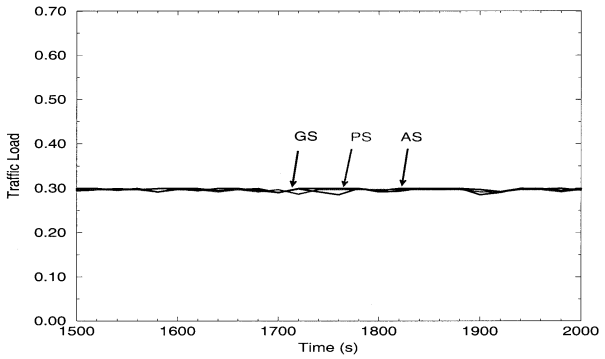
Because the BE traffic does not require any service guarantee and does not contact the BB before sending traffic, there is no need to consider the BE traffic in this set of simulations. Therefore, we will focus on the three types of traffic that does require admission control: the GS, the PS, and the AS.

We set the GS flows be the *exp2* type traffic, and all the PS and AS flows be the *exp4* type traffic. Flows have an exponentially distributed inter-arrival time with a mean of 0.133 seconds and an exponential holding time with a mean of 120 seconds. Each new flow arrival will request the GS, the PS, or the AS with equal probability.

Again, we set a target network load ratio for a service requirement type in the BB using the *static link sharing* policy. Figures 19(a) and (b) present the network load for the following two cases: (1) Case 1: $\mu^{GS} = 0.1$, $\mu^{PS} = 0.2$, and $\mu^{AS} = 0.3$; and (2) Case 2: $\mu^{GS} = \mu^{PS} = \mu^{AS} = 0.3$. As shown in the figures, the BB entity can indeed control the traffic load on the data plane so that each type of service does not exceed its target ratio.



(a) Case 1: $\mu^{GS} = 0.1$, $\mu^{PS} = 0.2$, and $\mu^{AS} = 0.3$.



(b) Case 2: $\mu^{GS} = \mu^{PS} = \mu^{AS} = 0.3$.

Fig. 19 Traffic load on the link between ingress and egress nodes for each type of services in the peer-to-peer network.

6. Related Work

A node architecture based on strict priority (SP) scheduling was proposed previously [18] to provide integrated transport of the PS, the AS, and the BE services. However, it has been shown [2], [8] that, once over certain utilization, the concatenation of such class-based strict priority (SP) schedulers can cause delay jitter unbounded! This means that such node architecture [18] cannot always support the PS, which should experience low delay jitter. On the other hand, the node architecture proposed in this paper, which builds upon the VTRS/CSVC, can provide hard delay jitter bound to the PS (see (19)). Furthermore, our node architecture can also support the GS, which is otherwise not possible under the FIFO CBQ and SP architecture proposed in [18].

The idea of virtual time has been used extensively in the design of packet scheduling algorithms such as VC [25] and WFQ [10], [19], [20]. The notion of virtual time defined in these contexts is used to emulate an ideal scheduling system and is defined *local* to each scheduler. Computation of the virtual time function requires *per-flow* information (e.g., per-flow reservation state and per-flow scheduling state) to be maintained. In contrast, in this paper the notion of virtual time embodied by packet virtual time stamps can be viewed as *global* to an entire domain. Its computation is *core-stateless,* relying only on the packet state carried by packets.

In [23], the first core-stateless scheduling algorithm, *core jitter virtual clock* (CJVC), was designed and shown to provide the same end-to-end delay bound as a stateful VC-based reference network. In addition, an aggregate reservation estimation algorithm was developed for performing admission control without per-flow state. Our VTRS/CSVC was motivated and inspired by the results in [23]. However, the VTRS differs from [23] in several aspects. The VTRS is defined to serve as a *unifying scheduling framework* where *diverse* scheduling algorithms can be employed to support the GS [27]. The notion of packet virtual time stamps under the VTRS can lead to the design of new core-stateless scheduling algorithms, e.g., *core-stateless virtual clock* (CSVC). Furthermore, the VTRS is defined with an aim to decouple the data plane and the QoS control plane so that a flexible QoS provisioning and control architecture can be developed at the net-

work edge, without affecting the core of the network.

This paper is a sequel to our previous work on VTRS [27]. In particular, this paper shows that, in addition to the GS, we can also support the PS, the AS, and the BE services under the same node architecture. Furthermore, we also present detailed design and operations on the control plane (BB) and show how QoS control and provisioning can be performed solely at the BB without incurring any additional complexity on the core routers.

The BB model was first proposed in [18] for the PS and the AS using the DiffServ model. Under the BB architecture, admission control, resource provisioning, and other policy decisions are performed by a centralized BB in each network domain. However, many issues regarding the design of the BB such as admission control and QoS provisioning have not been addressed adequately. Furthermore, it is not clear, under the BB architecture proposed in [18], what level of QoS guarantees can be supported and whether or not core routers are still required to perform *local* admission control and QoS state management. One the other hand, the BB architecture proposed in this paper have clearly addressed the above issues. In particular, we show that a BB architecture, which builds upon VTRS/CSVC, is capable of decoupling QoS control functions from core routers. That is, our BB performs admission control and QoS state management solely at the BB, without incurring any additional complexity to the core routers on the data plane. Furthermore, our BB architecture supports per-flow GS (in addition to the PS and the AS).

7. Conclusion

We presented architecture and mechanisms to support multiple QoS under the DiffServ paradigm. On the data plane, we presented a node architecture based on the virtual time reference system (VTRS). The key building block of our node architecture is the core-stateless virtual clock (CSVC) scheduling algorithm, which, in terms of providing delay guarantee, has the same expressive power as a stateful weighted fair queueing (WFQ) scheduler. With the CSVC scheduler as our building block, we designed a node architecture that is capable of supporting integrated transport of the GS, the PS, the AS, and the BE service. On the control plane, we designed a bandwidth broker architecture to provide flexible resource allocation and QoS provisioning. Simulation results demonstrated that our architecture and mechanisms can indeed provide scalable and flexible support for integrated traffic of the GS, the PS, the AS, and the BE services.

Acknowledgments

The authors wish to thank the anonymous reviewers for their thorough review and helpful comments and suggestions, which helped to improve the presentation of this paper.

References

[1] P. Aukia, M. Kodialam, P.V.N. Koppol, T.V. Lakshman, H. Sarin, and B. Suter, "RATES: A server for MPLS traffic engineering," IEEE Network Magazine, pp.34–41, March/April 2000.

[2] J.C.R. Bennett, K. Benson, A. Charny, W.F. Courtney, and J.Y. Le Boudec, "Delay jitter bounds and packet scale rate guarantee for expedited forwarding," Proc. IEEE INFOCOM 2001, Anchorage, Alaska, April 2001.

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," RFC 2475, Internet Engineering Task Force, Dec. 1998.

[4] J. Boyle, R. Cohen, D. Durham, S. Herzog, R. Rajan, and A. Sastry, "The COPS (common open policy service) protocol," RFC 2748, Internet Engineering Task Force, Jan. 2000.

[5] R. Braden, D. Clark, and S. Shenker, "Integrated services in the Internet architecture: An overview," RFC 1633, Internet Engineering Task Force, July 1994.

[6] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP)—Version 1 functional specification," RFC 2205, Internet Engineering Task Force, Sept. 1997.

[7] R. Callon, P. Doolan, N. Feldman, A. Fredette, G. Swallow, and A. Viswanathan, "A framework for multiprotocol label switching," Internet Draft, Internet Engineering Task Force, Sept. 1999.

[8] A. Charny and J.Y. Le Boudec, "Delay bounds in a network with aggregate scheduling," Proc. First International Workshop of Quality of future Internet Services (QofIS'2000), Berlin, Germany, Sept. 25–26, 2000.

[9] D.D. Clark and W. Fang, "Explicit allocation of best-effort packet delivery service," IEEE/ACM Trans. Networking, vol.6, no.4, pp.362–373, Aug. 1998.

[10] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulations of a fair queueing algorithm," Proc. ACM SIGCOMM'89, pp.1–12, Austin, TX, 1989.

[11] S. Floyd and V. Jacobson, "On random early detection gateways for congestion avoidance," IEEE/ACM Trans. Networking, vol.1, no.4, pp.397–413, Aug. 1993.

[12] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," IEEE/ACM Trans. Networking, vol.3, no.4, pp.365–386, Aug. 1995.

[13] R. Guerin and V. Peris, "Quality-of-service in packet networks: basic mechanisms and directions," Computer Networks Journal (Elsevier Science), vol.31, no.3, pp.169–189, Feb. 1999.

[14] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured forwarding PHB group," RFC 2597, Internet Engineering Task Force, June 1999.

[15] Y.T. Hou, D. Wu, B. Li, T. Hamada, I. Ahmad, and H.J. Chao, "A differentiated services architecture for multimedia streaming in next generation Internet," Computer Networks Journal (Elsevier Science), vol.32, no.2, pp.185–209, Feb. 2000.

[16] V. Jacobson, K. Nichols, and K. Poduri, "An expedited forwarding PHB," RFC 2598, Internet Engineering Task Force, June 1999.

[17] J. Liebeherr and D.E. Wrege, "Priority queue schedulers with approximate sorting in output buffered switches," IEEE J. Sel. Areas Commun., vol.17, no.6, pp.1127–1145,

June 1999.
[18] K. Nichols, V. Jacobson, and L. Zhang, "A two-bit differentiated services architecture for the Internet," RFC 2638, Internet Engineering Task Force, July 1999.
[19] A.K. Parekh and R.G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single-node case," IEEE/ACM Trans. Networking, vol.1, no.3, pp.344–357, June 1993.
[20] A.K. Parekh and R.G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the multiple node case," IEEE/ACM Trans. Networking, vol.2, no.2, pp.137–150, April 1994.
[21] E.C. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," RFC 3031, Internet Engineering Task Force, Jan. 2001.
[22] S. Shenker, C. Partridge, and R. Guerin, "Specification of guaranteed quality of service," RFC 2212, Internet Engineering Task Force, Sept. 1997.
[23] I. Stoica and H. Zhang, "Providing guaranteed services without per flow management," Proc. ACM SIGCOMM 99, Cambridge, MA, Sept. 1999.
[24] A. Terzis, L. Wang, J. Ogawa, and L. Zhang, "A two-tier resource management model for the Internet," Proc. IEEE Global Internet 99, Rio de Janeiro, Brazil, Dec. 1999.
[25] L. Zhang, "VirtualClock: A new traffic control algorithm for packet switching networks," ACM Trans. Computer Syst., vol.9, pp.101–124, May 1991.
[26] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A new resource ReSerVation protocol," IEEE Network Magazine, vol.7, no.5, pp.8–18, Sept. 1993.
[27] Z.-L. Zhang, Z. Duan, and Y.T. Hou, "Virtual time reference system: A unifying scheduling framework for scalable support of guaranteed services," IEEE J. Sel. Areas Commun., vol.18, no.12, pp.2684–2695, Dec. 2000.
[28] Z.-L. Zhang, Z. Duan, and Y.T. Hou, "On scalable design of bandwidth brokers," IEICE Trans. Commun., vol.E84-B, no.8, Aug. 2001.

Yiwei Thomas Hou obtained his B.E. degree (*Summa Cum Laude*) from the City College of New York in June 1991, the M.S. degree from Columbia University in May 1993, and the Ph.D. degree from Polytechnic University, Brooklyn, New York, in January 1998, all in Electrical Engineering. He was awarded a five-year National Science Foundation (NSF) Graduate Research Traineeship for pursuing Ph.D. degree in high speed networking, and was recipient of the Alexander Hessel award for outstanding Ph.D. dissertation (1997–1998 academic year) from Polytechnic University. Since September 1997, Dr. Hou has been a Research Scientist at Fujitsu Laboratories of America (FLA), Sunnyvale, California. At FLA, he has been conducting and leading research efforts in the areas of scalable architecture, protocols, and implementations for differentiated services Internet; and quality of service (QoS) support for media streaming over wired and wireless IP-based networks; and service overlay architecture over the Internet. Dr. Hou is a co-recipient of the 2001 IEEE Transactions on Circuits and Systems for Video Technology best paper award. He is a member of the IEEE, ACM, Sigma Xi and New York Academy of Sciences.

Zhenhai Duan received a B.S. degree from Shandong University, China, and M.S. degree from Peking University, China, in 1994 and 1997, respectively, both in Computer Science. Currently he is pursuing a Ph.D. degree in the Department of Computer Science and Engineering at the University of Minnesota. His research interests are in the areas of computer and communications networks, currently in QoS provisioning over the Internet, traffic measurements and modeling. Mr. Duan is a student member of IEEE and ACM.

Zhi-Li Zhang received a B.S. degree in Computer Science from Nanjing University, China and his M.S. and Ph.D. degrees in Computer Science from University of Massachusetts, Amherst, in 1992 and 1997 respectively. In January 1997, he joined the Computer Science faculty at the University of Minnesota, where he is currently an Assistant Professor. From 1987 to 1990, he conducted research in the Computer Science Department at Århus University, Denmark, under a fellowship from the Chinese National Commission on Education. Dr. Zhang's current research interests include high speed networks, multimedia and real-time systems, and modeling and performance evaluation of computer and communication system. He received the National Science Foundation CAREER Award in 1997. He is also a co-recipient of 1996 ACM SIGMETRICS Conference best paper award, and a member of IEEE, ACM and INFORMS.

Takafumi Chujo is Director of Advanced Internet Research at Fujitsu Laboratories of America, Sunnyvale, CA. His research interests include IP/photonic networking infrastructure and next-generation Internet architecture. Mr. Chujo received his B.E. degree from Kanazawa University (Japan) in 1976 and M.E. degree from Nagoya University (Japan) in 1978. He served as a tutorial co-chair for NOMS'96 and a TPC co-chair for APNOMS'99. He is a member of IEEE.