Providing Scalable Support for Multiple QoS Guarantees: Architecture and Mechanisms

Y. Thomas Hou * Zhenhai Duan † Zhi-Li Zhang † Takafumi Chujo *

* Fujitsu Laboratories of America, Sunnyvale, CA, USA
† University of Minnesota, Minneapolis, MN, USA

Abstract— **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), which is a unifying scheduling framework for scalable support of the guaranteed service. 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. Based on the CSVC scheduler, we design a node architecture that is capable of supporting integrated transport of the guaranteed service (GS), the premium service (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.**

I. Introduction

The IETF has introduced the Differentiated Services (DiffServ) model [2] to address the issue of *scalability.* The DiffServ achieves scalability by offering services for an aggregate traffic rather than on a per-flow basis. On the data plane, simple *per-hop behaviors* (PHBs) (e.g., expedited forwarding (EF) [7] and assured forwarding (AF) [6]) have been defined to treat traffic aggregate and to provide differentiation in packet forwarding. On the control plane, a centralized *bandwidth broker* (BB) [8] 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 [8] 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[1]. It has been suggested [8] 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) can 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 [1], [4] 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 may not 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 aggregate) guaranteed service (GS) [10], 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 BE services. More specifically, we want to achieve the following three design objectives.

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) 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, an admitted PS flow should experience low delay jitter and low loss when it traverses the network.

2. Scalability in Network Core (Core-Stateless). We require that core routers maintain no per-flow state, i.e., per-flow reservation state or per-flow scheduling state.

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.

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.* [13] as a *unifying* scheduling framework for *scalable* support of 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 that 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 above.

The remainder of this paper is organized as follows. In Section II, 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 III presents a BB architecture and admission control algorithms on the control plane. In Section IV, we present simulation results to demonstrate the performance of our node architecture in providing QoS guarantees to each type of services. Section V concludes this paper.

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

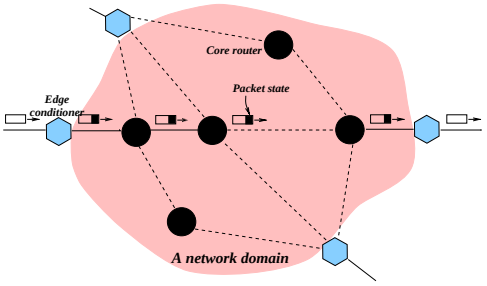


Fig. 1. A network domain where the VTRS is deployed.

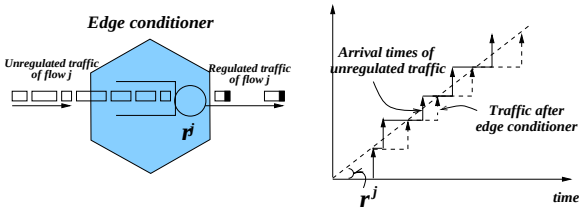


Fig. 2. Edge conditioning and its effect in the VTRS.

II. A Core-Stateless Architecture with Multiple QoS Support

We organize this section as follows. Section II-A presents the background on the VTRS and the CSVC scheduler. In Section II-B, we present our node architecture.

A. Virtual Time Reference System: A Background

The VTRS [13] 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. 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.

Conceptually, the VTRS consists of three logical components: *packet state* (see Fig. 1) carried by packets, *edge traffic conditioning* at the network edge (see Fig. 2), and *per-hop virtual time reference/update mechanism* at core routers. We refer readers to [13] for the details.

An important consequence of the VTRS 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 rate of a flow and the *error terms* of the routers along the flow's path. Furthermore, 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 [13] that almost all known scheduling algorithms can be characterized, be they *stateless* (e.g., FIFO) or *stateful* (e.g., WFQ).

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.

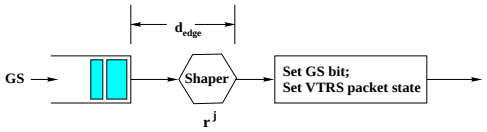


Fig. 3. Edge node shaping and marking for an GS flow.

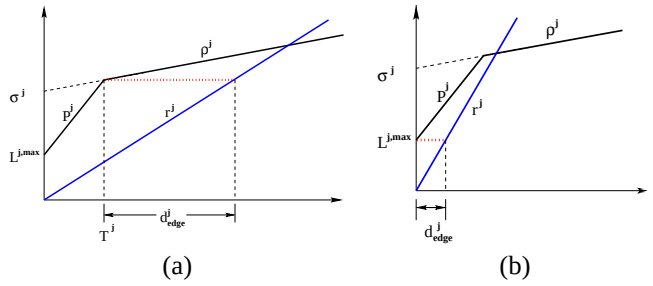


Fig. 4. Edge delay due to edge shaping for an GS flow. (a) $\rho^j \leq r^j < P^j$; and (b) $\rho^j \leq P^j \leq r^j$.

The CSVC is a work-conserving counterpart of the *core-jitter virtual clock* (CJVC) scheduling algorithm [11]. The CSVC scheduler services packets in the order of their virtual finish times. It has been shown [13] 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.

It has been shown in [13] that 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. Thus, the CSVC scheduler has the same expressive power, in terms of providing delay and rate guarantees, as a *stateful* WFQ scheduler, albeit that it does not maintain any reservation or scheduling state in the router. Due to paper length limitation, we strongly refer readers to [13] for the details of the VTRS and CSVC.

B. A Node Architecture for 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 [10], the PS [8], the AS [8], and the BE services. In Section II-B.1, we present the edge conditioning function. Section II-B.2 shows the buffering and scheduling mechanisms for both edge and core nodes.

B.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/Marking for the GS Flows. Before entering the traffic shaper (see Fig. 3), suppose the traffic profile of an GS flow j is specified using the standard dual-token bucket regulator $(\sigma^j, \rho^j, P^j, L^{j,max})$ 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

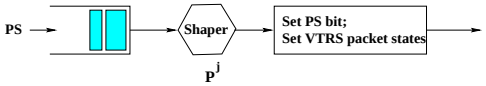


Fig. 5. Edge node shaping and marking for an PS flow.

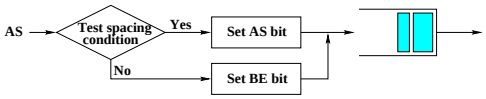


Fig. 6. Edge node marking for an AS flow.

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).

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 rate r^j (see Section III-B for r^j calculation during admission control procedure). Then from Fig. 4, we have

$$d^j_{edge} = \begin{cases} \frac{P^j - r^j}{r^j} \cdot \frac{\sigma^j - L^{j,max}}{P^j - \rho^j} + \frac{L^{j,max}}{r^j} & \text{if } \rho^j \le r^j < P^j, \\ \frac{L^{j,max}}{r^j} & \text{if } \rho^j \le P^j \le r^j. \end{cases} \quad (1)$$

Denote $T^j = (\sigma^j - L^{j,max})/(P^j - \rho^j)$, 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 properly set in the packet header [13].

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 [8], 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 that 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 (see Section II-B.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 (see Fig. 5).

After the shaper, the packet is then marked as an PS packet, with VTRS states properly set into the packet header [13].

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 [8]. The edge traffic conditioning is shown in Fig. 6. Unlike for an GS or an PS flow, an edge shaper is not employed for an AS flow [8]. The function of the edge conditioner is to examine (test) whether consecutive packets are properly spaced according to the sustainable rate ρ^j.

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.

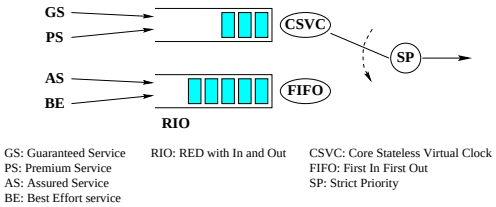


Fig. 7. A stateless node architecture for integrated support of the GS, the PS, the AS, and the BE services.

B.2 Buffering and Scheduling at Edge and Core Nodes

Figure 7 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[2]. 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. Under VTRS/CSVC, 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 domain. Upon arriving at the CSVC queue, the virtual finish time of this packet, $\tilde{\nu}_i^{j,k}$, is calculated [13] and this packet is then inserted into the appropriate place in the CSVC queue, where 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 [13].

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^j_{core} the maximum delay of all packets in flow j traversing the network core. Then d^j_{core} is given as follows [13]

$$d^j_{core} = h\frac{L^{j,max}}{r^j} + \sum_{i=1}^{h} \frac{L^{*,max}}{C_i} + \sum_{i=1}^{h-1} \pi_i, \quad (2)$$

where h is the total number of hops along path P of flow j, and π_i is the propagation delay from the ith node to the $(i+1)$th node along the path.

Denote D^P_{tot} as

$$D^P_{tot} = \sum_{i=1}^{h} \frac{L^{*,max}}{C_i} + \sum_{i=1}^{h-1} \pi_i. \quad (3)$$

[2] Note that under the DiffServ model, the buffer for the GS and the PS traffic may be mapped to the EF PHB [7] while the buffer for the AS and the BE may be mapped to the AF PHB [6]. 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.

Note that D_{tot}^P is a parameter of path P and is independent of the particular flow j traversing this path. Combining (1), (2), and (3), the maximum end-to-end delay, d_{e2e}^j, for all packets in flow j is then given by

$$\begin{aligned} d_{e2e}^j &= d_{edge}^j + d_{core}^j \\ &= \begin{cases} \frac{P^j - r^j}{r^j} \cdot \frac{\sigma^j - L^{j,max}}{P^j - \rho^j} + (h+1)\frac{L^{j,max}}{r^j} + D_{tot}^P \\ \qquad\qquad \text{if } \rho^j \le r^j < P^j, \\ (h+1)\frac{L^{j,max}}{r^j} + D_{tot}^P \\ \qquad\qquad \text{if } \rho^j \le P^j \le r^j. \end{cases} \end{aligned} \quad (4)$$

Observe that the end-to-end delay formula in (4) is *precisely the same as* that specified in the IETF IntServ Guaranteed Service [10] using the WFQ as the reference system! 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 that 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 that 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 $\frac{L^{*,max}}{C_i}$, it will guarantee flow j its reserved rate P^j with an error term $\frac{L^{*,max}}{C_i}$.

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

$$d_{core}^j = h\frac{L^{j,max}}{P^j} + \sum_{i=1}^{h} \frac{L^{*,max}}{C_i} + \sum_{i=1}^{h-1} \pi_i = h\frac{L^{j,max}}{P^j} + D_{tot}^P. \quad (5)$$

We point out that such a delay bound offered by the CSVC scheduler effectively circumvents the problem associated with (FIFO) class-based SP scheduling, where it has been shown [1], [4] that the delay jitter may be unbounded over a certain utilization.

Scheduling and Buffer Management for the AS and the BE Traffic. For the AS and the BE services, we employ a simple FIFO queue, which is similar to that in [8]. The FIFO queue is serviced with a strictly lower priority than the CSVC queue (see Fig. 7).

To differentiate packets in the FIFO queue, RED with In and Out (RIO) [5] is employed as the buffer management mechanism. A schematic diagram for the RIO mechanism is depicted in Fig. 8. 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 [5].

III. Control Plane Operations

In this section, we present control plane operations in a BB. We organize this section as follows. In Section III-A, we give an architectural overview of a BB, in particular, those details pertinent to admission control procedure. In Section III-B, we present the admission control procedure for the GS, the PS, and the AS flows.

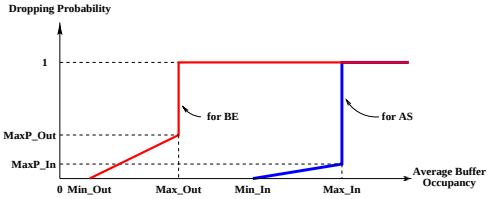


Fig. 8. Buffer management for the AS and the BE traffic with RIO mechanism.

A. Bandwidth Broker: An Architectural Overview

In the IETF DiffServ framework, a centralized model based on the notion of BB [8] 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.

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[3]. Our 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. 7) 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 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.

The basic centralized BB model is schematically depicted in Fig. 9. In this architectural model, the BB centrally maintains and manages a number of management information (data)bases (MIBs) regarding the network domain. Also shown in Fig. 9, the BB consists several modules such as admission control, QoS routing, and policy 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

[3] 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.

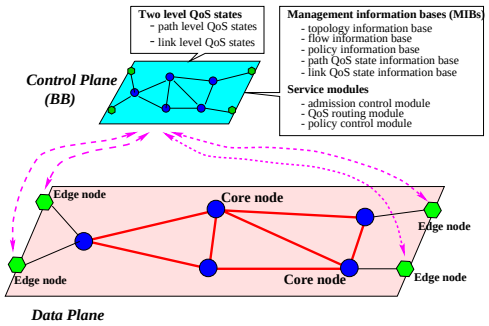


Fig. 9. Illustration of a bandwidth broker (BB) and its operation in a VTRS network domain.

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 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[4]. 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 P are the number of hops h on P, sum of the router error terms of all the CSVC schedulers and propagation delay along P, D^P_{tot} (see (3)), and the maximum permissible packet size (i.e., MTU) $L^{P,max}$. The dynamic QoS state information associated with P include, among others, the set of flows traversing 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 P such as the minimal remaining bandwidth along P for each type of services, i.e., $C^{GS}_{P,res}$ for the GS, $C^{PS}_{P,res}$ for the PS, and $C^{AS}_{P,res}$ 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 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 is the current residual bandwidth and buffer size at the link for each type of services.

B. 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 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 ease of exposition, we employ *static link sharing* policy. That is, $\mu^{GS}_i + \mu^{PS}_i + \mu^{AS}_i < 1$, where μ^{GS}_i, μ^{PS}_i, and μ^{AS}_i are *fixed* maximum allowable 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 serviced by a network of CSVC schedulers since each CSVC queue in our node architecture (Fig. 7) is given strict priority (SP) over the other FIFO queue. Let $j \in F^{GS}_i$ denote that an GS flow j currently traverses the ith node and C^{GS}_i be the total bandwidth at ith node allocated to the GS flow, i.e., $C^{GS}_i = \mu^{GS}_i C_i$. Then as long as $\sum_{j \in F^{GS}_i} r^j \leq C^{GS}_i$, the ith node can guarantee each GS flow j its reserved bandwidth r^j. We use $C^{GS}_{i,res}$ to denote

[4] 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. 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 [3], [9], or a simplified version (*minus* resource reservation) of RSVP.

the residual bandwidth at the ith node for the GS flows, i.e., $C_{i,res}^{GS} = C_i^{GS} - \sum_{j \in F_i^{GS}} r^j$. 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 flow ν, and $D^{\nu,req}$ be its end-to-end delay requirement. Let h be the number of hops in P, the path for the new flow. From (4), 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 $d_{e2e}^\nu \leq D^{\nu,req}$. Denote r_*^ν the smallest r^ν that satisfies the delay constraint and recall that $T^\nu = (\sigma^\nu - L^{\nu,max})/(P^\nu - \rho^\nu)$. Then we have

$$r_*^\nu = \begin{cases} \frac{T^\nu P^\nu + (h+1)L^{\nu,max}}{D^{\nu,req} - D_{tot}^P + T^\nu} & \text{if } \rho^\nu \leq r^\nu < P^\nu, \\ \frac{(h+1)L^{\nu,max}}{D^{\nu,req} - D_{tot}^P} & \text{if } \rho^\nu \leq P^\nu \leq r^\nu. \end{cases} \quad (6)$$

If the above solution for r_*^ν can be found, then we have a feasible reserved rate request to satisfy the flow's end-to-end delay constraint requirement.

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 $C_{P,res}^{GS}$ along path P for the GS, where $C_{P,res}^{GS} = \min_{i \in P} C_{i,res}^{GS}$ is maintained (as a path QoS parameter associated with 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}^P and C_{res}^P associated with P are maintained in the path QoS state MIB, the above admission test can be done in $O(1)$.

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. The flow id., traffic profile and service profile of the new flow will be inserted into the flow MIB. The minimal residual bandwidth $C_{P,res}^{GS}$ will be subtracted by r^ν, the reserved rate for the new GS flow ν. Similarly, for each link i along P, its residual bandwidth $C_{i,res}^{GS}$ will also be subtracted by r^ν. Furthermore, for any path P' that traverses S_i, its minimal residual bandwidth $C_{res}^{P',GS}$ may also be updated, depending on whether the update of $C_{i,res}^{GS}$ changes $C_{P',res}^{GS}$. Provided that a powerful (and perhaps, parallel) database system is employed, these database update operations can be performed in a 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, there is no need to calculate reserved rate requirement as in the case for an GS flow. Similar to admission control for the GS flows, a path-oriented approach can be applied to a new PS flow.

1. Admission Test. Let 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 $C_{P,res}^{PS}$ along path P for PS, where $C_{P,res}^{PS} = \min_{i \in P} C_{i,res}^{PS}$ is maintained (as a path QoS parameter associated with 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.

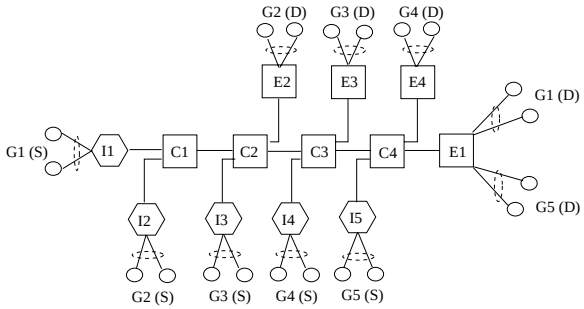


Fig. 10. A chain network.

IV. Simulation Investigation

In this section, we conduct simulations to investigate the performance of the integrated framework for supporting diverse service guarantees.

A. Simulation Settings

The network topology that we will use for the simulations is depicted in Fig. 10, which is referred to as the *chain* network. In this network, 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.

Each ingress edge router (Ii) contains two functionality blocks: an edge conditioner (see Section II-B.1) and the node architecture in Fig. 7. On the other hand, each core router (Cj) employs only the node architecture in Fig. 7.

All the links between routers have a propagation delay equal to 5 ms. The capacity of the links between edge (ingress or egress) routers and core routers, and the links between core routers is 10 Mb/s. 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.

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.

We assume that the CSVC queue in the node (Fig. 7) has sufficiently large 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

TABLE I

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)	Packet size in Byte (B)	Window size (packets)
exp1	UDP/EXP	1536	16	64	96	–
exp2	UDP/EXP	20480	128	512	512	–
exp3	UDP/EXP	–	16	64	96	–
exp4	UDP/EXP	–	128	512	512	–
ftp	TCP/FTP	–	–	–	1024	50

TABLE II

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)	120	120	–	–	–	–	–	–	–

TABLE III

FLOW THROUGHPUT 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 (Kb/s)	Flow id.	Throughput (Kb/s)	Loss rate (%)	Flow id.	Throughput (Kb/s)	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

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*) [5] 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 I. The fields marked as "–" means they do not apply 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.

B. Simulation Results

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. The simulated time is 200 seconds, of which the first 100 seconds are the simulation warm-up period.

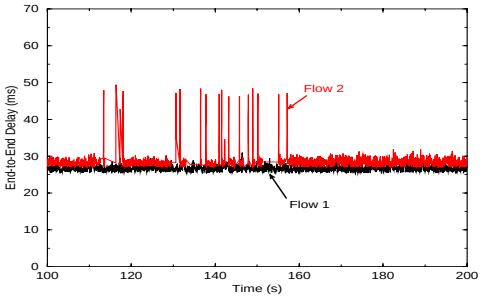
To examine the key properties of the proposed node architecture, flows with a traffic profile and service requirement will be generated randomly between the time interval [0, 1] seconds from a source node group Gn (S) to a destination group Gn (D), and once admitted by the BB, they will never terminate, 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, two BE ftp flows are randomly activated within the time interval [0, 0.5] seconds.

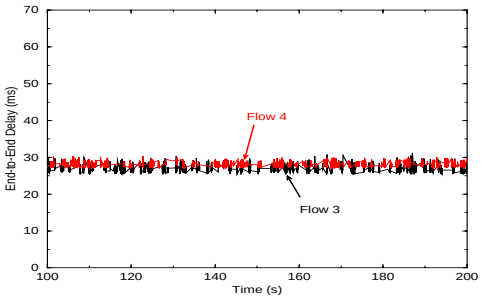
To demonstrate packet-level QoS performance for a flow with a particular traffic profile and service requirement, we focus on the admitted flows traversing the path G1(S) to G1(D), and purposely choose only 12 flows among all the admitted flows along this path to present our simulation results. These 12 flows are listed in Table II 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.

Fig. 11(a) shows the end-to-end packet delays of flows 1 and 2, which request GS in the chain network. Comparing the end-to-end delay requirement listed in Table II (100 ms), we observe that the delay requirements of both flows are satisfied. Fig. 11(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. 11(b), we see that the traffic of the PS flows experiences almost constant delay, as promised under our architecture. 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 (1)). 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.

So far we have confirmed that the end-to-end delay require-



(a) GS flows.



(b) PS flows.

Fig. 11. End-to-end packet delays of the GS/PS flows on the path traversing G1(s) to G1(D) in the chain network.

ments for the GS are satisfied. Next, we investigate the traffic throughput and packet loss rate of the 12 flows during the same simulation run.

Table III presents the corresponding data for the same simulation run (i.e., 100s–200s). 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. Note that flows 5 and 6 are burst traffic, which are more aggressive than the token bucket constrained flows 7 and 8. In flows 5 and 6, there are considerable packets marked as BE traffic at the network edge because of the burstiness; 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 BE traffic. 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 III, flows 9 to 12 are BE flows. We see that these BE flows have a similar packet dropping rate. Moreover, their throughput are reasonable close to each other. Therefore, 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.

V. 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 BE service. On the control plane, we designed a BB 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.

There are still many challenging problems, both theoretical and practical, in the design and implementation of scalable data and control plane architectures, such as bit requirements for packet state encoding, scalability issue in a centralized BB. Currently we are investigating a number of approaches to resolve these issues and these results will be reported in our future publications [14], [15].

References

[1] 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," in *Proc. IEEE INFOCOM 2001*, Anchorage, Alaska, April 2001.

[2] 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.

[3] 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, work in progress.

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

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

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

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

[8] K. Nichols, V. Jacobson, and L. Zhang, "A two-bit differentiated services architecture for the Internet," *RFC 2638* Internet Engineering Task Force, July 1999.

[9] E.C. Rosen, A. Viswanathan, and R. Callon "Multiprotocol label switching architecture," *Internet Draft*, Internet Engineering Task Force, August 1999, work in progress.

[10] S. Shenker, C. Partridge, and R. Guerin, "Specification of guaranteed quality of service," *RFC 2212*, Internet Engineering Task Force, Sept. 1997.

[11] I. Stoica and H. Zhang, "Providing guaranteed services without per flow management," in *Proc. ACM SIGCOMM 99*, Sept. 1999, Cambridge, MA.

[12] Z.-L. Zhang, Z. Duan, L. Gao, and Y.T. Hou, "Decoupling QoS control from core routers: a novel bandwidth broker architecture for scalable support of guaranteed services," in *Proc. ACM SIGCOMM 2000*, pp. 71–83, Stockholm, Sweden, August 2000.

[13] Z.-L. Zhang, Z. Duan, and Y.T. Hou, "Virtual time reference system: A unifying scheduling framework for scalable support of guaranteed services," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 12, pp. 2684–2695, Dec. 2000.

[14] Z.-L. Zhang, Z. Duan, and Y.T. Hou, "On scalable design of bandwidth brokers," unpublished.

[15] Z.-L. Zhang, Z. Duan, and Y.T. Hou, "Fundamental trade-offs in aggregate packet scheduling," unpublished.