Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 00-027

Virtual Time Reference System: A Unifying Scheduling Framework for Scalable Support of Guaranteed Services

Zhi-li Zhang, Zhenhai Duan, and Y. Thomas Hou

April 24, 2000

Virtual Time Reference System: A Unifying Scheduling Framework for Scalable Support of Guaranteed Services

Zhi-Li Zhang[†], Zhenhai Duan[†] and Yiwei Thomas Hou[‡]

[†] Dept. of Computer Science & Engineering
University of Minnesota
Minneapolis, MN 55455
{zhzhang,duan}@cs.umn.edu

[‡] Fujitsu Labs of America
595 Lawrence Expressway
Sunnyvale, CA 94086
thou@fla.fujitsu.com

Abstract

We propose and develop a novel *virtual time reference system* as a unifying scheduling framework to provide scalable support for guaranteed services. This virtual time reference system is designed as a conceptual framework upon which guaranteed services can be implemented in a scalable manner using the DiffServ paradigm. The key construct in the proposed virtual time reference system is the notion of *packet virtual time stamps*, whose computation is *core stateless*, i.e., no per-flow states are required for its computation. In this paper, we lay the theoretical foundation for the definition and construction of packet virtual time stamps. We describe how per-hop behavior of a core router (or rather its scheduling mechanism) can be characterized via packet virtual time stamps, and based on this characterization, establish end-to-end per-flow delay bounds. Consequently, we demonstrate that, in terms of its ability to support guaranteed services, the proposed virtual time reference system has the same expressive power and generality as the IntServ model. Furthermore, we show that the notion of packet virtual time stamps leads to the design of *new* core stateless scheduling algorithms, especially work-conserving ones. In addition, our framework does not exclude the use of existing scheduling algorithms such as *stateful* fair queueing algorithms to support guaranteed services.

1 Introduction

The problem of Quality of Service (QoS) provisioning in packet-switched networks has been the focus of networking and telecommunication research communities for the last decade or so. Many new packet scheduling algorithms (see, e.g., [7, 14, 19, 24]) such as Virtual Clock (VC) and Weighted Fair Queueing (WFQ) have been proposed for the support of *QoS guarantees*. For example, it has been shown [8, 15] that in a network where WFQ schedulers (or VC schedulers) are employed at every router, end-to-end delay and bandwidth guarantees can be supported for each user traffic flow. Using these results as a reference model, the Internet IETF has defined a *guaranteed service* [17] under its Integrated Services or IntServ architecture [3, 6], where end-to-end delay and bandwidth guarantees are provided for users on a *per-flow* (either individual or aggregate) basis. To support the IETF IntServ architecture, a signaling protocol, RSVP, for setting up end-to-end QoS reservation along a flow's path has also been proposed and standardized [4, 25].

Performing per-flow management inside the network, however, raises the important issue of *scalability*. Due to the complexities of per-flow operations both in the data plane and QoS control plane, the IntServ architecture *may not* scale well with the size of the Internet and the number of applications. As an alternative to per-flow based QoS provisioning, in recent years a different paradigm — the Differentiated

Services or DiffServ — has been proposed and defined by the Internet IETF [1, 2]. Under this paradigm, services are defined and implemented within individual administrative domains. To provide scalable QoS support, fine-grain user QoS control information is only maintained at the edge routers (i.e., ingress and egress routers) of an administrative domain. The user traffic is appropriately *conditioned* (i.e., shaped) before injected into the network core. At core routers, no per-flow QoS state is maintained. User packets are processed based on a number of pre-specified *Per-Hop Behaviors* (PHBs) encoded by bit patterns carried inside a packet header that convey to core routers the desired levels of QoS support. (We will refer to these bit patterns, or the PHBs they embody, as the *packet state*.) End-to-end, *user-to-user* QoS support is provided through intra-domain QoS provisioning and inter-domain service agreement. These *control plane* functions can be performed, for example, by employing a *bandwidth broker* architecture [13]. The DiffServ paradigm greatly simplifies the data plane of the network core of a domain, thereby making it more scalable. On the other hand, the DiffServ architecture, as it is currently defined, aims to provide only *coarse-grain* QoS support to users. It remains to be seen whether such a service model would be sufficient to meet the potentially diverse user QoS requirements in the future. Furthermore, many issues regarding the design of bandwidth brokers such as admission control and QoS provisioning still need to be addressed, *both theoretically and in practice.*

Recently in an exciting and important piece of work [21], Stoica and Zhang, using theDiffServ paradigm and the novel notion of *dynamic packet state*, developed several techniques to support end-to-end *per-flow* bandwidth and delay guarantees *without per-flow QoS management.* In the data plane, they designed a new (non-work-conserving) scheduling algorithm, called *Core Jitter Virtual Clock* or CJVC, to provide end-to-end per-flow delay guarantees *without per-flow scheduling states at core routers.* (Such scheduling algorithms are referred to as *core stateless*, in contrast to the conventional *stateful* scheduling algorithms such as VC or WFQ, where certain scheduling states must be maintained for each flow.) In the control plane, an *aggregate reservation estimation* algorithm is designed which eliminates the need of maintaining *per-flow QoS reservation states.* Instead, an *aggregate* QoS reservation state is maintained at each core router. A hop-by-hop signaling protocol, however, is still needed to set up QoS reservation for each flow along its path within a domain.

Inspired by the seminal work of Stoica and Zhang, in this paper we propose and develop a novel *virtual time reference system* as a *unifying* scheduling framework to provide scalable support for guaranteed services. In the same way that the WFQ reference system relates to the IntServ architecture, the proposed virtual time reference system is designed as a *conceptual* framework upon which guaranteed services can be implemented in a scalable manner using the DiffServ paradigm. More specifically, this virtual time reference system provides a unifying framework to characterize, in terms of their abilities to provide delay and bandwidth guarantees, both the *per-hop behaviors* of core routers and the *end-to-end properties* of their concatenation. The key construct in the proposed virtual time reference system 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 crucial 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 virtual time reference system is *core stateless*, as no per-flow state is needed at core routers for computing packet virtual time stamps.

In this paper, we lay the theoretical foundation for the definition and construction of packet virtual time stamps. We describe how per-hop behavior of a core router (or rather its scheduling mechanism) can be characterized via packet virtual time stamps, and based on this characterization, establish end-to-end per-flow delay bounds. Consequently, we demonstrate that in terms of support for guaranteed services, the proposed virtual time reference system has the same expressive power as the IntServ model. Furthermore, we show that the notion of packet virtual time stamps leads to the design of *new* core stateless scheduling algorithms, especially work-conserving ones. In addition, our framework *does not exclude* the use of existing scheduling algorithms such as *stateful* fair queueing algorithms to support guaranteed services.

The objectives of the proposed virtual time reference system are two-fold. First, as a reference system, it must not *mandate* any specific scheduling algorithms to be employed in a network in order to provide guaranteed services. In other words, it must allow for diverse scheduling algorithms as long as they are capable of providing QoS guarantees. In fact, we will show that our virtual time reference system can accommodate both *core stateless* scheduling algorithms such as CJVC and *stateful* scheduling algorithms. Second, the virtual time reference system provides a QoS abstraction for scheduling mechanisms that *decouples* the data plane from the QoS control plane. This abstraction facilitates the design of a bandwidth broker architecture (either centralized or distributed), where QoS states are maintained *only* at bandwidth brokers, *while still being capable of providing QoS guarantees with similar granularity and flexibility of the IntServ guaranteed service.* We believe that these two objectives are important in implementing guaranteed services in practice. For example, the ability to employ diverse scheduling algorithms not only encourages choice and competition among equipment vendors and Internet service providers (ISPs), but also, perhaps more importantly, allows a network and its services to evolve. Similarly, by maintaining QoS reservation states only in bandwidth brokers, core routers are relieved of QoS control functions such as admission control, making them potentially more efficient. Furthermore, a QoS control plane which is decoupled from the data plane allows an ISP to deploy sophisticated provisioning and admission control algorithms to optimize network utilization without incurring software/hardware upgrades at core routers. This paper will focus mostly on the theoretical underpinning of the proposed virtual time reference system. We will briefly address the issues regarding its implementation. The problem of designing a bandwidth broker architecture based on the virtual time reference system to support QoS provisioning and admission control will only be briefly discussed; the details are left to a future paper.

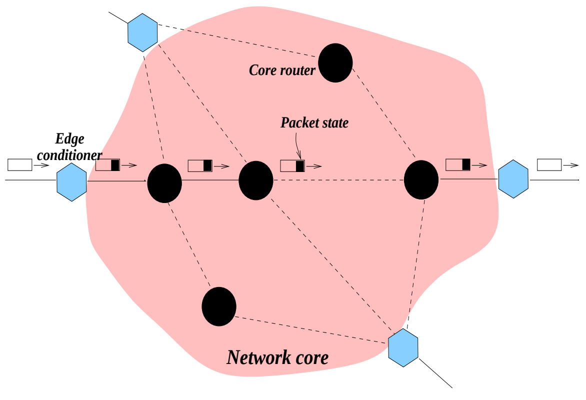
The rest of this paper is organized as follows. In the next section we will briefly outline the basic architecture of the virtual time reference system. In Section 3 we define a virtual time reference system in the context of an ideal per-flow system. This virtual time reference system is extended in Section 4 to account for the effect of packet scheduling. Furthermore, end-to-end per-flow delay bounds are also derived using the virtual time reference system. In Section 5, we design new core stateless scheduling algorithms using packet virtual time stamps. We then show that existing scheduling algorithms can be accommodated in our framework—simple static scheduling algorithms with resource pre-configuration in Section 6 and the generic latency-rate server scheduling framework in Section 7. In Section 8 we briefly discuss various issues regarding implementation and admission control. The paper is concluded in Section 9.

2 Virtual Time Reference System: Basic Architecture

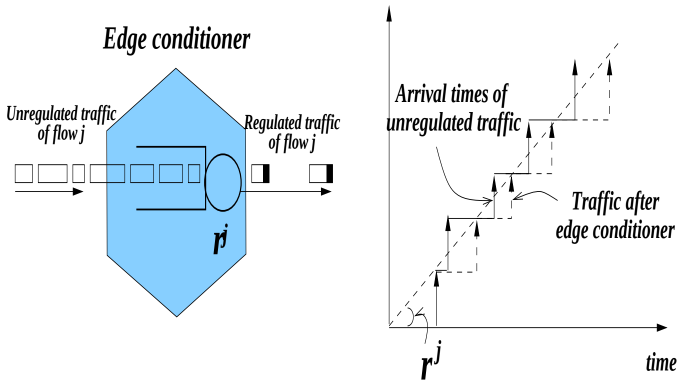
In this section we outline the basic architecture of the proposed unifying scheduling framework—the *virtual time reference system.* Conceptually, the virtual time reference system consists of three logical components (see Figure 1 and Figure 2): *packet state* carried by packets, *edge traffic conditioning* at the network edge, and *per-hop virtual time reference/update mechanism* at core routers. The virtual time reference system is defined and implemented within a *single* administrative domain. In other words, packet state inserted by one administrative domain will not be carried over to another administrative domain.

The packet state carried by a packet contains three types of information: 1) QoS reservation information of the flow[1] the packet belongs to (e.g., the reserved rate or delay parameter of the flow); 2) a virtual time stamp of the packet; and 3) a virtual time adjustment term. The packet state is initialized and inserted

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



(a) A conceptual network model


Edge conditioner
Unregulated traffic of flow j
Regulated traffic of flow j
r^j
Arrival times of unregulated traffic
Traffic after edge conditioner
r^j
time

(b) Edge conditioner and its effect

Figure 1: Edge conditioning in the virtual time reference system.

into a packet at the network edge after it has gone through the traffic conditioner. The *per-hop* behavior of each core router is defined with respect to the packet state carried by packets traversing it. As we will see later, *the virtual time stamps associated with the packets of a flow form the "thread" which "weaves" together the per-hop behaviors of core routers along the flow's path to support the QoS guarantee of the flow.*

Edge traffic conditioning plays a key role in the virtual time reference system, as *it ensures that traffic of a flow will never be injected into the network core at a rate exceeding its reserved rate.* This traffic conditioning is done by using a traffic shaper (or more specifically, a rate spacer, see Figure 1(b)), which enforces appropriate spacing between the packets of a flow based on its reserved rate. This is illustrated in the diagram on the right hand side of Figure 1(b). Formally, for a flow j with a reserved rate r^j, the inter-arrival time of two consecutive packets of the flow is such that

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

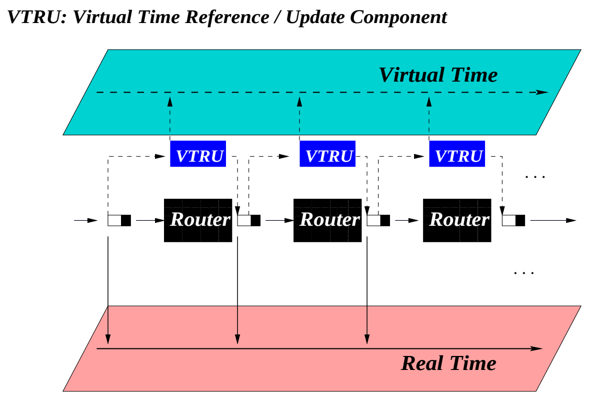
where $a^{j,k}$ denotes the arrival time of the kth packet $p^{j,k}$ of flow j at the network core, and $L^{j,k}$ the size of packet $p^{j,k}$.

In the *conceptual* framework of the virtual time reference system, each core router is equipped with a per-hop virtual time reference/update mechanism to maintain the continual progression of the *virtual time* embodied by the packet virtual time stamps. As a packet traverses each core router along the path of its flow, a virtual time stamp is "attached" to the packet. This virtual time stamp represents the arrival time of the packet at the core router *in the 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 associated with packets of a flow satisfy an important property, which we refer to as the *virtual spacing property*. Let $\tilde{\omega}^{j,k}$ be the virtual time stamp associated with the kth packet, $p^{j,k}$, of flow j. Then

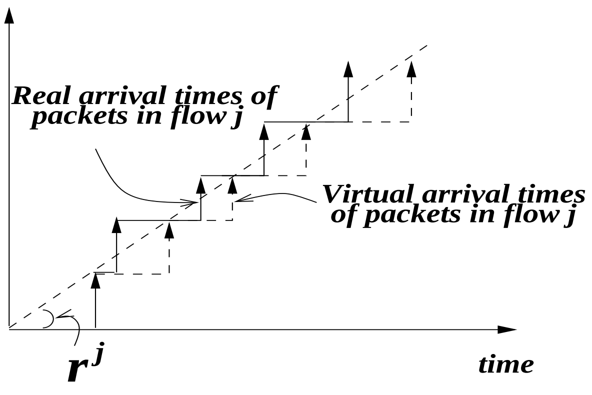
$$\tilde{\omega}^{j,k+1} - \tilde{\omega}^{j,k} \geq \frac{L^{j,k+1}}{r^j} \tag{2}$$

for all k.

Comparing (2) with (1), we see that *with respect to the virtual time*, the inter-arrival time spacing is preserved at a core router. Or to put it another way, *the "virtual rate" (as defined with respect to the virtual*



(a) Virtual time reference/update mechanism


Real arrival times of packets in flow j
Virtual arrival times of packets in flow j
r^j
time

(b) Virtual traffic shaping

Figure 2: Illustration of the virtual time reference system.

time) of packets of a flow arriving at a core router does not exceed the reserved rate of the flow. Clearly this statement in general does not hold with respect to the *real* arrival times of the packets at a core router (see Figure 2(b)). Another key property of packet virtual time stamps is that *at a core router the virtual arrival time of a packet always lags behind its real arrival time.* This property (referred to as the *reality check condition*) is important in deriving end-to-end delay bound experienced by packets of a flow across the network core. The per-hop virtual time reference/update mechanism at a core router is designed in such a manner so as to ensure that these properties of the packet virtual time stamps are satisfied at the entry point and/or exit point of the core router (see the illustration in Figure 2).

The virtual time reference system provides a unifying framework to formalize the per-hop behavior of a core router and to quantify its ability to provide delay guarantees. This formalism is independent of the scheduling mechanism employed by the core routers, *be it stateful or stateless.* Here we briefly describe how this mechanism works (see Section 4 for more details). Conceptually, for each packet traversing a core router, a *virtual finish time* is computed and assigned to it. This virtual finish time is derived from its virtual time stamp and other packet state information. Intuitively, it represents the time the packet finishes its service in an *ideal per-flow reference system*, where the flow to which the packet belongs to is the only flow serviced by the system. *The per-hop behavior of a core router is defined in terms of an upper bound on the difference between the actual departure time and virtual finish time of a packet traversing the core router.* This upper bound is referred to as the *error term* of the core router. Therefore, the scheduling mechanism of the core router can be abstracted into a *scheduling blackbox* characterized by an error term. This simple abstraction enables us to derive *end-to-end delay bounds* for flows traversing an arbitrary concatenation of such scheduling blackboxes, similar to what the notion of latency-rate servers [20] does for various fair queue algorithms.

In summary, while based on the DiffServ paradigm, the virtual time reference system renders the same expressive power and generality, in terms of the ability to provide guaranteed services, as the IntServ Model. Furthermore, the virtual time reference system provides a unifying scheduling framework upon which a scalable QoS provisioning and admission control framework can be built, where all QoS reservation states for guaranteed services are eliminated from the network core. The remainder of this paper is devoted to laying formal foundation for the virtual time reference system. We also illustrate how various scheduling algorithms fit into the unifying framework. Issues of implementation and admission control will also be

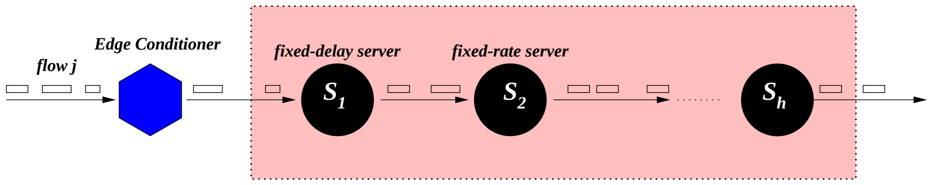


Figure 3: An ideal per-flow system.

briefly discussed.

3 An Ideal Per-flow Virtual Time Reference System

In this section we motivate and introduce the notion of packet virtual time stamps in the context of an *ideal per-flow system.* The virtual time reference system defined in this context is then extended in the next section to account for the effect of packet scheduling in a real network system.

Figure 3 illustrates an ideal per-flow system, where a regulated flow is serviced by a dedicated channel. The dedicated channel consists of a series of servers in tandem. Packets of a flow j are serviced *in order* from server 1 to server h. For $k = 1, 2, \ldots$, the kth packet of flow j is denoted by $p^{j,k}$, and its size by $L^{j,k}$. Let r^j be the reserved rate of flow j, and d^j a delay parameter associated with flow j. For simplicity of exposition, we assume that in this ideal per-flow system the propagation delay from one server to the next server is zero.

We consider two types of servers: *fixed-rate servers* and *fixed-delay servers.* A fixed-rate server has a service capacity equal to the reserved rate r^j of flow j. Hence a fixed-rate server takes $L^{j,k}/r^j$ amount of time to process packet $p^{j,k}$ of flow j. A fixed-delay server has a fixed latency, which equals to the delay parameter d^j of flow j. In other words, a fixed-delay server with latency d^j takes exactly d^j amount of time to process packets of flow j, independent of their packet sizes. We will see later the importance of fixed-delay servers in modeling scheduling algorithms that can provide delay-rate decoupling. Throughout this section, we assume that in the ideal per-flow system, there are p fixed-rate servers and $h-p$ fixed-delay servers.

Before we introduce the notion of packet virtual time stamps, we first need to understand and quantify the end-to-end delay experienced by the packets in the ideal per-flow system. We embark on this task in Section 3.1. Based on the results obtained thereof, in Section 3.2 we will introduce the ideal per-flow virtual time reference system. Table 1 summarizes the important notation used in the paper.

3.1 End-to-end Delay of the Ideal Per-flow System

Recall that before entering this ideal per-flow system, packets from flow j go through an edge conditioner, where they are regulated so that the rate the packets are injected into the ideal per-flow system never exceeds the reserved rate r^j of the flow. Formally, let $a_1^{j,k}$ be the arrival time[2] of packet $p^{j,k}$ of flow j at the first server of the ideal per-flow system. Then the edge spacing condition (3) holds, namely,

$$a_1^{j,k+1} - a_1^{j,k} \geq L^{j,k+1}/r^j, \; k = 1, 2, \ldots \tag{3}$$

[2]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.

Table 1: Notation used in the paper.

General Notation	
$p^{j,k}$	the kth packet flow j
$L^{j,k}$	packet length of $p^{j,k}$
$L^{j,max}$	maximum packet length of flow j
$L^{*,max}$	maximum packet length of all flows at a server/router
r^j	reserved rate of flow j
d^j	delay parameter of flow j
h	number of hops (servers/routers) along the path of flow j
p	number of fixed-rate servers/rate-based schedulers along flow j's path
Notation for the Ideal Per-Flow System	
$a_i^{j,k}$	arrival time of packet $p^{j,k}$ at node i
$f_i^{j,k}$	finish time of packet $p^{j,k}$ at node i
$\Delta_i^{j,k}$	cumulative queueing delay packet $p^{j,k}$ experienced up to server i (inclusive)
Notation for the Virtual Time Reference System	
$\tilde{\omega}_i^{j,k}$	virtual time stamp of packet $p^{j,k}$ at node i
$\tilde{\nu}_i^{j,k}$	virtual finish time of packet $p^{j,k}$ at node i
$\delta^{j,k}$	virtual time adjustment term for packet $p^{j,k}$: $\delta^{j,k} = \Delta_p^{j,k}/p$
$\tilde{d}_i^{j,k}$	virtual delay of packet $p^{j,k}$ at node i: $\tilde{d}_i^{j,k} = \tilde{\nu}_i^{j,k} - \tilde{\omega}_i^{j,k}$
$\hat{a}_i^{j,k}$	actual time packet $p^{j,k}$ arrives at node i
$\hat{f}_i^{j,k}$	actual time packet $p^{j,k}$ departs from node i
Ψ_i	error term of scheduling blackbox at node i
$\pi_{i,i+1}$	propagation delay from the i^{th} node to the $(i+1)^{th}$ node

Let $A^j(\tau,t)$ denote the amount of flow j traffic that is injected into the ideal per-flow system over a time interval $[\tau,t]$. Using (3), it is easy to see that

$$A^j(\tau,t) \le r^j(t-\tau) + L^{j,max} \tag{4}$$

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

In order to derive the end-to-end delay experienced by packets in the ideal per-flow system, we first consider the *pure rate-based* system, where all servers are fixed-rate servers, i.e., $p = h$. This result can then be extended to the general ideal per-flow system with mixed fixed-rate and fixed-delay servers.

For $i = 1, 2, \ldots, h$, let $a_i^{j,k}$ denote the time packet $p^{j,k}$ arrives at server $\mathcal{S}_i$, and $f_i^{j,k}$ the time it leaves server i. In the pure rate-based ideal per-flow system, it is not hard to see that the following recursive relationships among $a_i^{j,k}$'s and $f_i^{j,k}$'s hold. For any $k = 1, 2, \ldots,$

$$a_i^{j,k} = f_{i-1}^{j,k},\ i = 2, \ldots, h, \tag{5}$$

and

$$f_i^{j,k} = \max\{a_i^{j,k}, f_i^{j,k-1}\} + \frac{L^{j,k}}{r^j},\ i = 1, 2, \ldots, h, \tag{6}$$

where in (6) we have used the convention that $f_i^{j,0} = 0$.

Note that in the special case where all packets of flow j have the same size L^j, each packet takes precisely L^j/r^j to be processed at each fixed-rate server. (In this case, a fixed-rate server functions as a fixed-delay server.) Because of the edge spacing property (3), we observe that no packet will ever be delayed in any fixed-rate server (see Figure 4(a)). In other words, for $i = 1, 2, \ldots, h$, $a_i^{j,k} \geq f_i^{j,k-1}$ and $f_i^{j,k} = a_i^{j,k} + L^j/r^j$. Therefore, in this case we have $f_h^{j,k} = a_1^{j,k} + hL^j/r^j$. Hence in this case, the *end-to-end delay* experienced by packet $p^{j,k}$ in the ideal per-flow system, which is defined as $f_h^{j,k} - a_1^{j,k}$, is hL^j/r^j.

In the general case where packets of flow j have variable sizes, the situation becomes somewhat more complicated. As shown in Figure 4(b), a small packet may be delayed at a server due to the longer processing time of a large packet preceding it. This delay can have a cascading effect which may cause more succeeding packets to be delayed.

For $i = 1, 2, \ldots, h$, let $\Delta_i^{j,k}$ denote the cumulative queueing delay experienced by packet $p^{j,k}$ up to server i (inclusive). Formally,

$$\Delta_i^{j,k} = f_i^{j,k} - (a_1^{j,k} + i\frac{L^{j,k}}{r^j}). \tag{7}$$

For the pure rate-based ideal per-flow system, we can derive an important recursive relation, $\Delta_i^{j,k}$ to $\Delta_i^{j,k-1}$ and the arrival times of packets $p^{j,k-1}$ and $p^{j,k}$ at the first-hop server. This recursive relation is given in the following theorem, the proof of which can be found in Appendix A.

Theorem 1 *For any packet $p^{j,k}$, $k = 1, \ldots,$ and $i = 1, 2, \ldots, h$,*

$$\Delta_i^{j,1} = 0$$

and

$$\Delta_i^{j,k} = \max\left\{0, \Delta_i^{j,k-1} + i\frac{L^{j,k-1} - L^{j,k}}{r^j} + a_1^{j,k-1} - a_1^{j,k} + \frac{L^{j,k}}{r^j}\right\}. \tag{8}$$

■

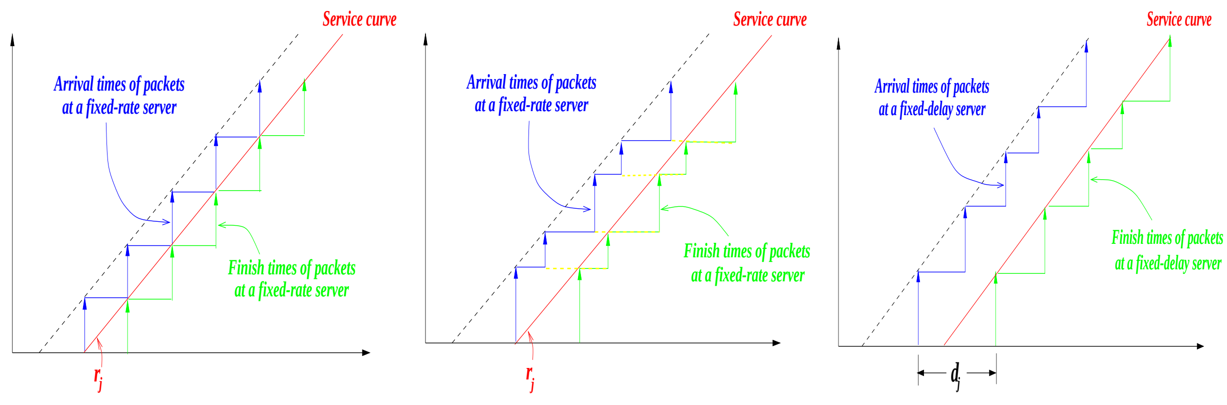
The importance of Theorem 1 lies in the fact that for each $p^{j,k}$, $\Delta_h^{j,k}$ can be calculated (recursively) *at the network edge*. As we will see in Section 3.2, this fact is critical in providing a *core stateless* definition of packet virtual time stamps for a system involving fixed-rate servers with variable packet sizes.

We now consider the general ideal per-flow system with both fixed-rate and fixed-delay servers. Recall that we assume we have p fixed-rate servers and $h - p$ fixed delay servers. As before, let $a_i^{j,k}$ and $f_i^{j,k}$ denote the arrival time and departure time of packet $p^{j,k}$ at server $\mathcal{S}_i$. Clearly, if $\mathcal{S}_i$ is a fixed-rate server, the recursive relation (6) holds among $a_i^{j,k}$'s and $f_i^{j,k}$'s. In the case where $\mathcal{S}_i$ is a fixed-delay server, we have that for $k = 1, 2, \ldots,$

$$a_i^{j,k} = f_{i-1}^{j,k} \text{ and } f_i^{j,k} = a_i^{j,k} + d^j. \tag{9}$$

Unlike a fixed-rate server, every packet of flow j incurs a delay of precisely d^j at a fixed-delay server, regardless of its size. Hence there is no extra queueing delay due to the packet size difference (see Figure 4(c)). It is easy to see that we can re-arrange the location of the fixed-rate servers in the ideal per-flow system without affecting the end-to-end delay experienced by each packet in the system. Hence, without loss of generality, we can assume that the last $n - p$ servers are the fixed delay servers. Then from (7), we have

$$f_h^{j,k} = a_1^{j,k} + \Delta_p^{j,k} + p\frac{L^{j,k}}{r^j} + (h - p)d^j.$$


Service curve
Arrival times of packets at a fixed-rate server
Finish times of packets at a fixed-rate server
r_j
Service curve
Arrival times of packets at a fixed-rate server
Finish times of packets at a fixed-rate server
r_j
Service curve
Arrival times of packets at a fixed-delay server
Finish times of packets at a fixed-delay server
d_j

(a) Fixed-rate server with constant-size packets

(b) Fixed-rate server with variable-size packets

(c) Fixed-delay server

Figure 4: Delay experienced by packets at a server in the ideal per-flow system.

Therefore, the end-to-end delay of packet $p^{j,k}$ in the ideal per-flow system is $f_h^{j,k} - a_1^{j,k} = \Delta_p^{j,k} + p\frac{L^{j,k}}{r^j} + (h-p)d^j$. In particular, from Corollary 11 in Appendix A, we have $\Delta_p^{j,k} + pL^{j,k}/r^j \le pL^{j,max}/r^j$. Thus,

$$f_h^{j,k} - a_1^{j,k} \le p\frac{L^{j,max}}{r^j} + (h-p)d^j. \tag{10}$$

Note that the above end-to-end delay bound holds for all packets of flow j. As an aside, from the perspective of providing end-to-end delay bounds, we can treat a fixed-rate server with service capacity r^j as if it were a fixed-delay server with a latency parameter $L^{j,max}/r^j$. The resulting "pure" delay-based system yields exactly the same delay bound (10). This treatment of fixed-rate servers may simplify the implementation of the virtual time reference system in practice (see Section 8.1).

3.2 Packet Virtual Time Stamps and Ideal Per-flow System

The key construct in the proposed virtual time reference system is the notion of *packet virtual time stamps*. In this section, we formally specify the properties of packet virtual time stamps, and provide a definition in the context of the ideal per-flow system. The resulting virtual time reference is referred to as the *ideal per-flow virtual time reference system.*

For $i = 1, 2, \ldots, h$, let $\tilde{\omega}_i^{j,k}$ denote the virtual time stamp associated with packet $p^{j,k}$ at server $\mathcal{S}_i$. Intuitively, we can regard $\tilde{\omega}_i^{j,k}$ as the (virtual) arrival time of packet $p^{j,k}$ at server $\mathcal{S}_i$ *according to the virtual time.* At server $\mathcal{S}_i$, packet $p^{j,k}$ is also assigned a *virtual finish time*, denoted by $\tilde{\nu}_i^{j,k}$, where $\tilde{\nu}_i^{j,k} \ge \tilde{\omega}_i^{j,k}$. The difference $\tilde{d}_i^{j,k} = \tilde{\nu}_i^{j,k} - \tilde{\omega}_i^{j,k}$ is referred to as the *virtual delay* associated with packet $p^{j,k}$ at server $\mathcal{S}_i$.

We postulate the following properties that packet virtual time stamps (and the corresponding virtual finish times) of flow j must satisfy at each server $\mathcal{S}_i$.

Virtual Spacing: for $k = 1, 2, \ldots,$

$$\tilde{\omega}_i^{j,k+1} - \tilde{\omega}_i^{j,k} \ge \frac{L^{j,k+1}}{r^j}. \tag{11}$$

Reality Check: $\tilde{\omega}_i^{j,k} \ge a_i^{j,k}$, where recall that $a_i^{j,k}$ is the *real* time packet $p^{j,k}$ arrives at server $\mathcal{S}_i$.

Bounded Delay: $f_h^{j,k} = \tilde{\nu}_h^{j,k}$, or more generally, $f_h^{j,k} - \tilde{\nu}_h^{j,k}$ is bounded from above.

Core Stateless: the virtual time stamp $\tilde{\omega}_i^{j,k}$ of each packet $p^{j,k}$ can be calculated at each server $\mathcal{S}_i$ using solely the packet state information carried by the packet (possibly with some additional constant parameters associated with the server).

Intuitively, the *virtual spacing property* ensures that according to the virtual time, the amount of flow j traffic arriving at server i is limited by its reserved rate r^j. To put it formally, consider an arbitrary time interval $[\tau, t]$. We say that *according to the virtual time*, packet $p^{j,k}$ *arrives* at server $\mathcal{S}_i$ during the time interval $[\tau, t]$ (or simply, packet $p^{j,k}$ *virtually arrives* at server $\mathcal{S}_i$ during the time interval $[\tau, t]$), if and only if $\tau \le \tilde{\omega}_i^{j,k} \le t$. Let $\tilde{A}^j(\tau, t)$ denote the amount of flow j traffic arriving virtually in the time interval $[\tau, t]$. It can be shown that (see the *Virtual Shaping Lemma* and its proof in Appendix B)

$$\tilde{A}^j(\tau, t) \le r^j(t - \tau) + L^{j,max}. \tag{12}$$

This bound is analogous to the traffic envelope (4) at the network edge, only that here the amount of flow j traffic is measured according to the virtual time. It suggests that if packet virtual time stamps are used to schedule packets, *explicit rate control or reshaping within the network core is not necessary.*

The *reality check condition* and *bounded delay property* are important in ensuring that end-to-end delay bounds can be derived using the virtual time reference system (both for the ideal per-flow system as well as for a *real* network packet scheduling system, as we will see later). The *core stateless property* is the *key* to the construction of a *scalable* virtual time reference system that does not require per-flow scheduling state information at each core router.[3] In the following we provide a definition of packet virtual time stamps for the ideal per-flow system, and show that it satisfies all the four properties listed above.

Consider the ideal per-flow system shown in Figure 3. Recall that we assume that there are p fixed-rate servers and $h-p$ fixed-delay servers in the ideal per-flow system. For each packet $p^{j,k}$, define $\delta^{j,k} = \Delta_p^{j,k}/p$. We refer to $\delta^{j,k}$ as the *virtual time adjustment term* for packet $p^{j,k}$. It is calculated at the network edge and inserted into the packet state in addition to the reserved rate r^j and delay parameter d^j. For $i = 1, 2, \ldots, h$, the *virtual delay* $\tilde{d}_i^{j,k}$ associated with packet $p^{j,k}$ at server $\mathcal{S}_i$ is computed from the packet state information using the following formula:

$$\tilde{d}_i^{j,k} = \begin{cases} L^{j,k}/r^j + \delta^{j,k} & \text{if } \mathcal{S}_i \text{ is a fixed rate server,} \\ d^j & \text{if } \mathcal{S}_i \text{ a fixed delay server.} \end{cases} \tag{13}$$

At the first-hop server $\mathcal{S}_1$, the virtual time stamp of packet $p^{j,k}$ is defined to be $\tilde{\omega}_1^{j,k} = a_1^{j,k}$, which is the time packet $p^{j,k}$ is injected to the ideal per-flow system and arrives at $\mathcal{S}_1$. This value is inserted into the packet state of $p^{j,k}$ at the network edge. The corresponding virtual finish time of $p^{j,k}$ at server $\mathcal{S}_i$ is given by $\tilde{\nu}_i^{j,k} = \tilde{\omega}_i^{j,k} + \tilde{d}_i^{j,k}$.

For $i = 2, \ldots, h$, the virtual time stamp $\tilde{\omega}_i^{j,k}$ and the corresponding virtual finish time $\tilde{\nu}_i^{j,k}$ associated with packet $p^{j,k}$ at server $\mathcal{S}_i$ are defined as follows:

$$\tilde{\omega}_i^{j,k} = \tilde{\nu}_{i-1}^{j,k} = \tilde{\omega}_{i-1}^{j,k} + \tilde{d}_{i-1}^{j,k} \text{ and } \tilde{\nu}_i^{j,k} = \tilde{\omega}_i^{j,k} + \tilde{d}_i^{j,k}. \tag{14}$$

From the above definition, it is clear that the core stateless property holds trivially. In the rest of this section we show that the other three properties are also satisfied. This fact is stated in the following theorem.

[3]As an example of the importance of the core stateless property, consider the following "definition" of virtual time stamps. At each server $\mathcal{S}_i$, we define $\tilde{\omega}_i^{j,k} = a_i^{j,k}$ and $\tilde{\nu}_i^{j,k} = f_i^{j,k}$ for packet $p^{j,k}$. Then from (3), (5), (6), and (9), we see that the virtual spacing property holds. Furthermore, the reality check condition and the bounded delay property also hold trivially. However, using this definition, computation of packet virtual time stamps at a core router requires maintenance of per-flow scheduling state (i.e., the value of the virtual time stamp of the previous packet). Therefore, such a definition is *not* core stateless.

Theorem 2 *For $i = 1, 2, \ldots, h$, the virtual spacing property (11) holds at each server $\mathcal{S}_i$. Furthermore,*

$$\tilde{\omega}_i^{j,k} \geq a_i^{j,k} \tag{15}$$

and in particular,

$$\tilde{\nu}_h^{j,k} = f_h^{j,k}. \tag{16}$$

Proof: We first establish that the virtual spacing property holds. Fix i and let p_i be the number of fixed-rate servers along the path from the first hop to the $(i-1)$th hop. Clearly $p_i \leq p$. Note that from (14) and (13), we have

$$\tilde{\omega}_i^{j,k} = \tilde{\omega}_1^{j,k} + \sum_{q=1}^{i-1} \tilde{d}_q^{j,k} = a_1^{j,k} + p_i(\delta^{j,k} + \frac{L^{j,k}}{r^j}) + (i-1-p_i)d^j. \tag{17}$$

Hence to prove (11), it suffices to show

$$a_1^{j,k} + p_i(\delta^{j,k} + \frac{L^{j,k}}{r^j}) \geq a_1^{j,k-1} + p_i(\delta^{j,k-1} + \frac{L^{j,k-1}}{r^j}) + \frac{L^{j,k}}{r^j},$$

or equivalently,

$$\delta^{j,k} \geq \delta^{j,k-1} + \frac{L^{j,k-1} - L^{j,k}}{r^j} + \frac{a_1^{j,k-1} - a_1^{j,k} + \frac{L^{j,k}}{r^j}}{p_i}. \tag{18}$$

From the definition of $\delta^{j,k}$ and Theorem 1, we have

$$\delta^{j,k} = \frac{\Delta_p^{j,k}}{p} \geq \frac{\Delta_p^{j,k-1}}{p} + \frac{L^{j,k-1} - L^{j,k}}{r^j} + \frac{a_1^{j,k-1} - a_1^{j,k} + \frac{L^{j,k}}{r^j}}{p}. \tag{19}$$

Using (3) and the fact that $p_i \leq p$, we see that the last term in the right hand side of (19) is larger than the corresponding term in (18). Hence (18) holds.

We now establish (15). As $\tilde{\omega}_1^{j,k} = a_1^{j,k}$, (15) holds trivially for $i = 1$. To show that (15) also hold for $i = 2, \ldots, h$, observe that

$$a_i^{j,k} = f_{i-1}^{j,k} = a_1^{j,k} + \Delta_{p_i}^{j,k} + p_i \frac{L^{j,k}}{r^j} + (i-1-p_i)d^j, \tag{20}$$

where recall that p_i is the number of fixed-rate servers among $\mathcal{S}_1, \ldots, \mathcal{S}_{i-1}$.

Comparing (20) and (17) and using the fact that $\Delta_p^{j,k}/p \geq \Delta_{p_i}^{j,k}/p_i$ (see Corollary 10 in Appendix A), we see that (15) indeed holds.

Lastly, (16) follows easily from the definitions, as

$$\tilde{\nu}_h^{j,k} = a_1^{j,k} + \Delta_p^{j,k} + p\frac{L^{j,k}}{r^j} + (h-p)d^j = f_h^{j,k}. \tag{21}$$

■

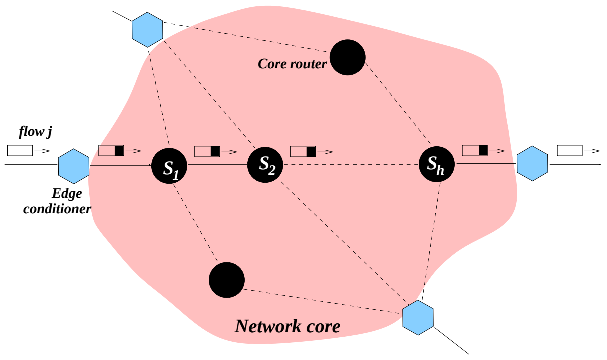


Figure 5: A flow traverses a network core.

4 Virtual Time Reference System and Packet Scheduling

In this section we extend the virtual time reference system defined in the context of ideal per-flow system to a network system where each core router is shared by multiple flows. The key notion we will introduce is the *error term* of a core router (or rather, of its scheduling mechanism), which accounts for the effect of packet scheduling in providing delay guarantees for a flow. Based on this notion of error term, we define a generic virtual time reference system, which provides a unifying scheduling framework to characterize the end-to-end behavior of core routers in providing delay guarantees.

Consider a flow j, whose path through a network core is shown in Figure 5. Flow j has a reserved rate r^j and a delay parameter d^j. The traffic of flow j is regulated at the network edge such that for $k = 1, 2, \ldots,$

$$\hat{a}_1^{j,k+1} - \hat{a}_1^{j,k} \geq \frac{L^{j,k+1}}{r^j} \tag{22}$$

where $\hat{a}_1^{j,k}$ is the *actual* time packet $p^{j,k}$ of flow j arrives at the first router along its path, after being injected into the network core.

As shown in Figure 5, the path of flow j consists of h core routers, each of which employs certain scheduling mechanism to provide guaranteed service for flow j. For $i = 1, 2, \ldots, h$, we will refer to the scheduler at core router i as a scheduling *blackbox*, and denote it by $\mathcal{S}_i$. In the following, we will first characterize the *per-hop behavior* of the scheduling blackboxes, and then show how end-to-end delay bounds can be derived based on this characterization of their per-hop behavior.

4.1 Scheduling Blackbox: Per-Hop Behavior

Corresponding to the fixed-rate servers and fixed-delay servers in the ideal per-flow system, we categorize the scheduling blackboxes into two types: *rate-based* scheduling blackbox and *delay-based* scheduling blackbox. They are distinguished by how the virtual delay parameter is computed, as in the ideal per-flow system. For a rate-based scheduling blackbox $\mathcal{S}_i$, packet $p^{j,k}$ of flow j is assigned a virtual delay $\tilde{d}_i^{j,k} = L^{j,k}/r^j + \delta^{j,k}$, where $\delta^{j,k}$ is the virtual time adjustment term carried in the packet state. For a delay-based scheduling blackbox $\mathcal{S}_i$, packet $p^{j,k}$ of flow j is assigned a virtual delay $\tilde{d}_i^{j,k} = d^j$. In other words, the virtual delay $\tilde{d}_i^{j,k}$ is given by the same formula as in (13). In either case, we see that the virtual delay $\tilde{d}_i^{j,k}$ can be computed using only the packet state information carried by the packet.

Now fix an index i, where $i = 1, 2, \ldots, h$, and consider the scheduling blackbox $\mathcal{S}_i$. For any flow j traversing the scheduling blackbox $\mathcal{S}_i$, let $\tilde{\omega}_i^{j,k}$ be the virtual time stamp associated with packet $p^{j,k}$ as it enters $\mathcal{S}_i$. We will provide a definition for $\tilde{\omega}_i^{j,k}$ shortly and establish its properties. At this point, we only assume that the *reality check condition* holds at $\mathcal{S}_i$, namely,

$$\hat{a}_i^{j,k} \leq \tilde{\omega}_i^{j,k} \tag{23}$$

where $\hat{a}_i^{j,k}$ is the *actual* time that packet $p^{j,k}$ enters the scheduling blackbox $\mathcal{S}_i$. Hence upon its arrival at $\mathcal{S}_i$, the virtual time stamp associated with packet $p^{j,k}$ is never smaller than its real arrival time.

At $\mathcal{S}_i$, packet $p^{j,k}$ is assigned a virtual finish time $\tilde{\nu}_i^{j,k}$, where $\tilde{\nu}_i^{j,k} = \tilde{\omega}_i^{j,k} + \tilde{d}_i^{j,k}$. Let $\hat{f}_i^{j,k}$ denote the *actual* time packet $p^{j,k}$ departs $\mathcal{S}_i^{j,k}$, i.e., $\hat{f}_i^{j,k}$ is the *real finish time* of $p^{j,k}$. We say that the scheduling blackbox $\mathcal{S}_i$ can *guarantee* packets of flow j their virtual delays with an *error term* Ψ_i, if for any k,

$$\hat{f}_i^{j,k} \leq \tilde{\nu}_i^{j,k} + \Psi_i. \tag{24}$$

In other words, each packet is guaranteed to depart the scheduling blackbox $\mathcal{S}_i$ by the time $\tilde{\nu}_i^{j,k} + \Psi_i = \tilde{\omega}_i^{j,k} + \tilde{d}_i^{j,k} + \Psi_i$.

By using the packet virtual finish time as a reference point to quantify the real finish time of a packet at a core router, we are able to abstract and characterize the per-hop behavior of a core router via an error term. This error term captures the ability of the core router to provide guaranteed services to a flow. In particular, for a rate-based scheduling blackbox $\mathcal{S}_i$, we say that $\mathcal{S}_i$ guarantees flow j its reserved rate r^j with an error term Ψ_i if (24) holds. For a delay-based scheduling blackbox $\mathcal{S}_i$, we say that $\mathcal{S}_i$ guarantees flow j its delay parameter d^j with an error term Ψ_i if (24) holds.

4.2 Virtual Time Reference System and End-to-End Delay Bounds

We now extend the virtual time reference system defined earlier to account for the effect of packet scheduling by incorporating the error terms of core routers into the system. In particular, we illustrate how packet virtual time stamps associated with flow j should be referenced and updated as packets of flow j traverse the core routers along the flow's path. We also derive and characterize the end-to-end behavior of these core routers in concatenation.

Consider the path of flow j shown in Figure 5. Suppose there are p rate-based scheduling blackboxes and $h-p$ delay-based scheduling blackboxes. For $i = 1, 2, \ldots, h$, let Ψ_i be the error term associated with the scheduling blackbox $\mathcal{S}_i$. In other words, $\mathcal{S}_i$ can guarantee flow j either its reserved rate r^j or its delay parameter d^j with an error term Ψ_i. The virtual delay $\tilde{d}_i^{j,k}$ associated with packet $p^{j,k}$ at $\mathcal{S}_i$ is given below:

$$\tilde{d}_i^{j,k} = \begin{cases} L^{j,k}/r^j + \delta^{j,k} & \text{if } \mathcal{S}_i \text{ is rate-based,} \\ d^j & \text{if } \mathcal{S}_i \text{ is delay-based} \end{cases}$$

where $\delta^{j,k} = \Delta_p^{j,k}/p$ is the virtual time adjustment term of packet $p^{j,k}$.

For $i = 1, 2, \ldots, h$, let $\tilde{\omega}_i^{j,k}$ denote the virtual time stamp associated with packet $p^{j,k}$ of flow j at $\mathcal{S}_i$, and $\tilde{\nu}^{j,k}$ be the virtual finish time of packet $p^{j,k}$ at $\mathcal{S}_i$. Then

$$\tilde{\nu}^{j,k} = \tilde{\omega}_i^{j,k} + \tilde{d}_i^{j,k}.$$

We now define $\tilde{\omega}_i^{j,k}$ and show that this definition satisfies the four requirements of packet virtual time stamps, namely, the *virtual spacing property*, the *reality check condition*, the *bounded delay property* and the *core stateless property*. Here in defining the reality check condition and bounded delay property, the quantities $a_i^{j,k}$ and $f_i^{j,k}$ defined in Section 3.2 are replaced by $\hat{a}_i^{j,k}$ and $\hat{f}_i^{j,k}$, which denote the *real arrival time* and *real finish time* of packet $p^{j,k}$ at $\mathcal{S}_i$, respectively.

As in the ideal per-flow system, the virtual time stamp associated with packet $p^{j,k}$ at the first-hop router $\mathcal{S}_1$ is set to its (real) arrival time, i.e.,

$$\tilde{\omega}_1^{j,k} = \hat{a}_1^{j,k}. \tag{25}$$

Thus $\tilde{\nu}_1^{j,k} = \tilde{\omega}_1^{j,k} + \tilde{d}_1^{j,k} = \hat{a}_1^{j,k} + \tilde{d}_1^{j,k}$.

From (22), the virtual spacing property is clearly met at the first-hop router. Furthermore, the reality check condition also holds trivially. Therefore, by the definition of Ψ_1, we have

$$\hat{f}_1^{j,k} \leq \tilde{\nu}_1^{j,k} + \Psi_1.$$

For $i = 1, 2, \ldots, h-1$, let $\pi_{i,i+1}$ denote the *propagation delay*[4] from the i^{th} hop router $\mathcal{S}_i$ to the $(i+1)^{th}$ hop router $\mathcal{S}_{i+1}$. Then

$$\hat{a}_{i+1}^{j,k} = \hat{f}_i^{j,k} + \pi_{i,i+1}.$$

By the definition of Ψ_i, we have

$$\hat{a}_{i+1}^{j,k} \leq \tilde{\nu}_i^{j,k} + \Psi_i + \pi_{i,i+1}. \tag{26}$$

In order to ensure that the reality check condition holds as packet $p^{j,k}$ enters the $(i+1)^{th}$ hop router $\mathcal{S}_{i+1}$, the relation (26) suggests that the virtual time stamp $\tilde{\omega}_{i+1}^{j,k}$ associated with packet $p^{j,k}$ at $\mathcal{S}_{i+1}$ should be defined as follows:

$$\tilde{\omega}_{i+1}^{j,k} = \tilde{\nu}_i^{j,k} + \Psi_i + \pi_{i,i+1} = \tilde{\omega}_i^{j,k} + \tilde{d}_i^{j,k} + \Psi_i + \pi_{i,i+1}. \tag{27}$$

Then $\hat{a}_{i+1}^{j,k} \leq \tilde{\omega}_{i+1}^{j,k}$.

Since Ψ_i's and $\pi_{i,i+1}$'s are *fixed* parameters associated with the core routers and the path of flow j, it is clear that the packet virtual time stamps defined using (27) are *core stateless*. Namely, they can be computed at each core router using only the packet state information carried by the packets (in addition to the two fixed parameters associated with the routers and the flow's path). Thus *no per-flow state needs to be maintained at these core routers.*

Since $\Psi_i + \pi_{i,i+1}$ is a constant independent of $p^{j,k}$, comparing the definition of $\tilde{\omega}_i^{j,k}$ in (27) and that in (14), it is easy to see that the virtual spacing property also holds at each core router $\mathcal{S}_i$. Furthermore, we have

$$\tilde{\omega}_{i+1}^{j,k} = \tilde{\nu}_i^{j,k} + \Psi_i + \pi_{i,i+1} = \hat{a}_1^{j,k} + \sum_{q=1}^{i} \tilde{d}_q^{j,k} + \sum_{q=1}^{i} \Psi_q + \sum_{q=1}^{i} \pi_{q,q+1}.$$

In particular, we see that the bounded delay property holds, as

$$\hat{f}_h^{j,k} \leq \tilde{\nu}_h^{j,k} + \Psi_h = \hat{a}_1^{j,k} + \sum_{q=1}^{i} \tilde{d}_q^{j,k} + \sum_{q=1}^{h} \Psi_q + \sum_{q=2}^{h} \pi_{q-1,q}.$$

This completes the construction of packet virtual time stamps for flow j. In a nutshell, packet virtual time stamps are initialized using (25) at the network edge, and are referenced and updated using (27) at each core router. The reference and update mechanism of the resulting virtual time reference system is schematically shown in Figure 6.

[4]Here for simplicity, we assume that the propagation delay experienced by each packet of flow j from $\mathcal{S}_i$ to $\mathcal{S}_{i+1}$ is a constant. In case this is not true, we can assume $\pi_{i,i+1}$ to be the *maximum* propagation delay from $\mathcal{S}_i$ to $\mathcal{S}_{i+1}$. Then for any packet $p^{j,k}$, $\hat{a}_{i+1}^{j,k} \leq \hat{f}_i^{j,k} + \pi_{i,i+1}$.

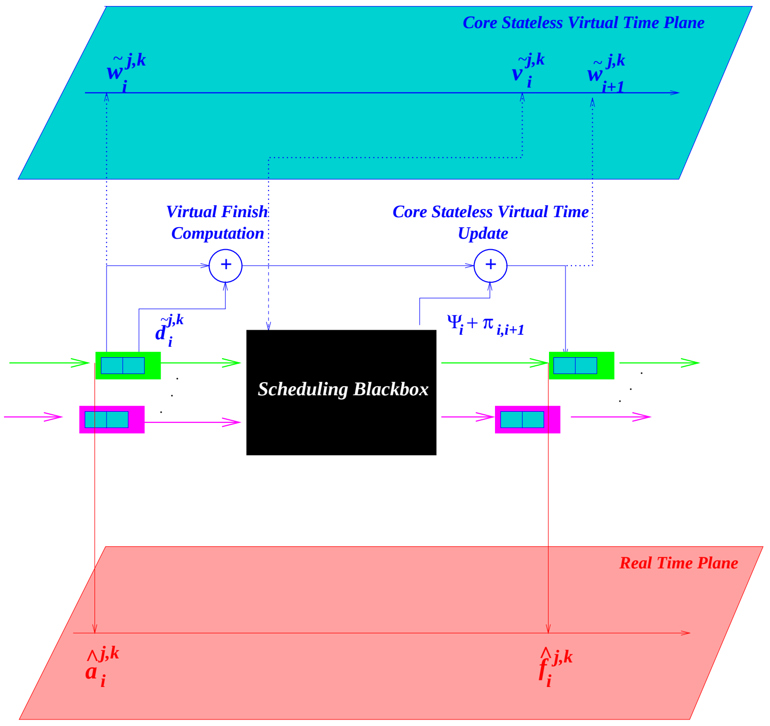


Figure 6: Virtual time reference system: per-hop behavior and operations.

Using the virtual time reference system, the following end-to-end delay bound for flow j can be easily derived from the bounded delay property of packet virtual time stamps:

$$\hat{f}_h^{j,k} - \hat{a}_1^{j,k} \le \Delta_p^{j,k} + p\frac{L^{j,k}}{r^j} + (h-p)d^j + \sum_{i=1}^{h}\Psi_i + \sum_{i=1}^{h-1}\pi_{i,i+1} \le p\frac{L^{j,max}}{r^j} + (h-p)d^j + \sum_{i=1}^{h}\Psi_i + \sum_{i=1}^{h-1}\pi_{i,i+1}, \quad (28)$$

where the last inequality follows from Corollary 11 in Appendix A.

In the special case where only rate-based scheduling algorithms are employed at core routers (i.e., $p = h$), we have

$$\hat{f}_h^{j,k} - \hat{a}_1^{j,k} \le h\frac{L^{j,max}}{r^j} + \sum_{i=1}^{h}\Psi_i + \sum_{i=1}^{h-1}\pi_{i,i+1}. \quad (29)$$

This bound is analogous to those derived for fair-queueing/latency-rate-server based scheduling algorithms [8, 15, 20]. In particular, if $\Psi_i = L^{*,max}/C_i$, where $L^{*,max}$ is the maximum packet size permissible at the i^{th} router and C_i is its service capacity, then the above inequality yields precisely the same delay bound as is obtained for a flow in a network of Weighted Fair Queueing (WFQ) schedulers [15] (or Virtual Clock (VC) schedulers [8] for that matter).

By incorporating delay-based scheduling algorithms into our framework, we can provide a certain degree of *rate and delay decoupling*. To see this, let D^j be such that $D^j \ge pL^{j,max}/r^j$. Set

$$d^j = \frac{1}{h-p}[D^j - p\frac{L^{j,max}}{r^j}]. \quad (30)$$

Suppose we can design delay-based scheduling algorithms that can support the delay parameter d^j for flow

j. Then from (28) we have

$$\hat{f}_h^{j,k} - \hat{a}_1^{j,k} \leq D^j + \sum_{i=1}^{h} \Psi_i + \sum_{i=1}^{h-1} \pi_{i,i+1}.$$

In the special case where $p = 0$ (i.e., only delay-based scheduling algorithms are employed along the path of flow j), setting $d^j = D^j/h$ yields

$$\hat{f}_h^{j,k} - \hat{a}_1^{j,k} \leq D^j + \sum_{i=1}^{h} \Psi_i + \sum_{i=1}^{h-1} \pi_{i,i+1}.$$

Clearly, this delay bound is completely decoupled from the reserved rate of flow j. Hence using pure delay-based scheduling algorithms, it is possible to support an arbitrary delay bound $D^j \geq 0$ for flow j (apart from the constant term $\sum_{i=1}^{h} \Psi_i + \sum_{i=1}^{h-1} \pi_{i,i+1}$ associated with the path of flow j).

Before we leave this section, it is interesting to compare our virtual time reference system with the WFQ-based reference system used in the Internet IntServ model [3]. From (28), we see that the constant term $\sum_{i=1}^{h} \Psi_i + \sum_{i=1}^{h-1} \pi_{i,i+1}$ is equivalent to the D_{total} term defined in the IntServ guaranteed service, whereas the rate-dependent term $p\frac{L^{j,max}}{r^j}$ corresponds to the C_{total} term in the IntServ guaranteed service. Furthermore, the delay parameter d^j, similar to the *slack term* in the IntServ guaranteed service, can be used to take advantage of the delay-rate decoupling offered by delay-based scheduling algorithms. Therefore, in terms of providing delay guaranteed services, our virtual time reference system has essentially the same expressive power as the IntServ guaranteed service model. What distinguishes the virtual time reference system from the IntServ guaranteed service model is its *core stateless* nature. Using the Internet DiffServ paradigm and the notion of packet virtual time stamps, our virtual time reference system allows for more scalable scheduling mechanisms (e.g., *core stateless* scheduling algorithms) to be employed for the support of guaranteed services. In addition, our virtual time reference system can accommodate both core stateless and stateful scheduling algorithms, thereby making it a unifying scheduling framework. In the next section we demonstrate that the notion of packet virtual time stamps leads to the design of new *core stateless* scheduling algorithms. In particular, we design both rate-based and delay-based scheduling algorithms with the *minimum* error term $\Psi = L^{*,max}/C$. In Section 6 we illustrate how some simple static scheduling algorithms can be used in our framework to provide scalable scheduling support with resource pre-configuration. In Section 7 we show that the generic scheduling framework based on latency-rate servers (examples of which include VC, WFQ and various variations of fair queueing algorithms) can also be accommodated in our framework.

5 Core Stateless Scheduling Algorithms: Examples

In this section we illustrate how the notion of packet virtual time stamps can be used to design new *core stateless* scheduling algorithms. In particular, we design a number of rate-based and delay-based core stateless scheduling algorithms, and establish their error terms using the properties of packet virtual time stamps.

5.1 Rate-Based Core Stateless Scheduling Algorithms

5.1.1 Core Stateless Virtual Clock Scheduling Algorithm

A *core stateless virtual clock* ($C_{\mathcal{S}}$VC) scheduler $\mathcal{S}$ is a rate-based scheduler. It services packets in the order of their virtual finish time. For any packet $p^{j,k}$ traversing $\mathcal{S}$, let $\tilde{\omega}^{j,k}$ be the virtual time carried by $p^{j,k}$

as it enters $\mathcal{S}$, and $\tilde{d}^{j,k} = \frac{L^{j,k}}{r^j} + \delta^{j,k}$ be its virtual delay. Then the virtual finish time $\tilde{\nu}^{j,k}$ of $p^{j,k}$ is given by $\tilde{\omega}^{j,k} + \tilde{d}^{j,k}$. We claim that the $C_{\not{S}}$VC scheduler can guarantee each flow j its reserved rate r^j with the minimum error term $\Psi_{C_{\not{S}}VC} = L^{*,max}/C$, provided that an appropriate schedulability condition is met. This fact is stated formally in the following theorem, the proof of which can be found in Appendix B.

Theorem 3 *Consider N flows traversing a $C_{\not{S}}VC$ scheduler $\mathcal{S}$ such that the schedulability condition $\sum_{j=1}^{N} r^j \le C$ is satisfied. Suppose that $\hat{a}^{j,k} \le \tilde{\omega}^{j,k}$ for any packet $p^{j,k}$ of flow j, $j = 1, 2, \ldots, N$. Then*

$$\hat{f}^{j,k} \le \tilde{\nu}^{j,k} + \frac{L^{*,max}}{C}. \tag{31}$$

In other words, $\Psi_{C_{\not{S}}VC} = \frac{L^{*,max}}{C}$.

5.1.2 Core-Jitter Virtual Clock (CJVC) Scheduling Algorithm

In [21], the core-jitter virtual clock (CJVC) scheduling algorithm is presented, which can be considered as a non-work-conserving version of the $C_{\not{S}}$VC scheduling algorithm. In the CJVC scheduling algorithm, if a packet $p^{j,k}$ arrives too early, it is held in a rate controller until its *eligibility time*, namely, when the real time reaches $\tilde{e}^{j,k} = \tilde{\omega}^{j,k} + \delta^{j,k}$ (see Appendix B). It can be shown that the CJVC scheduler has the same error term as the $C_{\not{S}}$VC scheduler does, i.e., $\Psi_{CJVC} = L^{*,max}/C$. The proof follows a similar argument as used in the case of $C_{\not{S}}$VC. Comparing $C_{\not{S}}$VC and CJVC, we see that the *work-conserving* $C_{\not{S}}$VC scheduling algorithm provides the same delay bound as the *non-work-conserving* CJVC scheduling algorithm, without the additional complexity of rate controller (*albeit core stateless*) at each core router as is required by CJVC. This is achieved because of the virtual shaping property of the core stateless virtual time.

5.1.3 Calendar Queue Approximation to Core Stateless Virtual Clock

In order to reduce the overhead of sorting, we can use a series of FIFO queues to approximate the $C_{\not{S}}$VC scheduling algorithm. These FIFO queues are referred to as *calendar queues*, as they are labeled by discrete time epochs. Conceptually, the virtual time is divided into fixed time slots with a slot unit of ι: $\tau_0, \tau_1, \ldots, \tau_p, \ldots$, where $\tau_p = p\iota$. Each time slot τ_p has an associated FIFO queue[5]. Upon its arrival, a packet $p^{j,k}$ is placed in the queue associated with time slot τ_p if $\tau_p \le \tilde{\nu}^{j,k} < \tau_{p+1}$. The calendar queues are serviced in the order of τ_p. Moreover, if queue τ_p is empty, the packet at the head of line from the next non-empty queue is serviced. However, suppose a new packet arrives at the previously empty queue τ_p while this packet (from a queue $\tau_{p'}$ where $\tau_{p'} > \tau_p$) is being serviced. After this packet departs, queue τ_p will be serviced next.

We call the above calendar queue approximation to $C_{\not{S}}$VC the *slotted* $C_{\not{S}}$VC. It can be shown that a slotted $C_{\not{S}}$VC scheduler has an error term $\Psi_{slotted\text{-}C_{\not{S}}VC} = L^{*,max}/C + \iota$. This result is stated in the following Theorem, the proof of which can be found in Appendix B.

Theorem 4 *Consider N flows traversing a slotted-$C_{\not{S}}$ VC scheduler $\mathcal{S}$ such that the schedulability condition $\sum_{j=1}^{N} r^j \le C$ is satisfied. Suppose that $\hat{a}^{j,k} \le \tilde{\omega}^{j,k}$ for any packet $p^{j,k}$ of flow j, $j = 1, 2, \ldots, N$. Then*

$$\hat{f}^{j,k} \le \tilde{\nu}^{j,k} + \frac{L^{*,max}}{C} + \iota. \tag{32}$$

In other words, $\Psi_{slotted\text{-}C_{\not{S}}VC} = \frac{L^{*,max}}{C} + \iota$.

[5]In reality, only a limited number of FIFO queues are needed. Whenever the current time t passes τ_p, the queues can be reused for the future time slots. This implementation is similar to the rotation priority queue proposed by Lieberherr *et al* [11, 12].

5.2 Delay-Based Core Stateless Scheduling Algorithms

5.2.1 Virtual Time Earliest Deadline First Scheduling Algorithm

A *virtual time earliest deadline first* (VT-EDF) scheduler $\mathcal{S}$ is a delay-based scheduler. It services packets in the order of their virtual finish time. Recall that the virtual finish time of $p^{j,k}$ is given by $\tilde{\nu}^{j,k} = \tilde{\omega}^{j,k} + d^j$, where $\tilde{\omega}^{j,k}$ is the virtual time carried by $p^{j,k}$ as it enters $\mathcal{S}$ and d^j is the delay parameter associated with its flow. Provided that an appropriate schedulability condition is met, it can be shown that the VT-EDF scheduler can guarantee each flow j its delay parameter d^j with the minimum error term $\Psi_{VT\text{-}EDF} = L^{*,max}/C$. This fact is stated formally in the following theorem, the proof of which is relegated to Appendix B.

Theorem 5 *Consider N flows traversing a VT-EDF scheduler $\mathcal{S}$, where d^j is the delay parameter associated with flow j, $1 \le j \le N$. Without loss of generality, assume $0 \le d^1 \le d^2 \le \cdots \le d^N$. Suppose the following* schedulability condition *holds:*

$$\sum_{j=1}^{N} [r^j(t - d^j) + L^{j,max}] \mathbf{1}_{\{t \ge d^j\}} \le Ct, \text{ for any } t \ge 0 \tag{33}$$

where the indicator function $\mathbf{1}_{\{t \ge d^j\}} = 1$ if $t \ge d^j$, 0 otherwise. Suppose that $\hat{a}^{j,k} \le \tilde{\omega}^{j,k}$ for any packet $p^{j,k}$ of flow j, $j = 1, 2, \ldots, N$. Then

$$\hat{f}^{j,k} \le \tilde{\nu}^{j,k} + \frac{L^{*,max}}{C}. \tag{34}$$

In other words, $\Psi_{VT\text{-}EDF} = \frac{L^{*,max}}{C}$.

We observe that the schedulability condition is essentially the same as the one derived for the standard EDF scheduling algorithm with flow traffic envelopes of the form given in (4) (see, e.g., [9, 12]). When using the standard EDF scheduling algorithm in a network to support guaranteed delay services, *per-hop re-shaping at core routers is required.* This is to ensure that the flows conform to their traffic envelopes so that the schedulability condition still holds at each hop [10, 23]. This rendition of the standard EDF is sometimes referred to as rate-controlled EDF, or RC-EDF. In contrast, using VT-EDF only requires shaping at the network edge to control the rate of each flow (i.e., to ensure (3) holds). As long as the the schedulability condition (33) holds at every VT-EDF scheduler, *no per-hop re-shaping is needed for any core router.* This is because the VT-EDF scheduler services packets using the packet virtual arrival times, not their real arrival times. From (12), we see that *according to the virtual time*, the traffic envelope for each flow is still preserved at each core router.

It is also interesting to compare the schedulability condition of the Core Stateless Virtual Clock with that of the Virtual-Time EDF. For each flow j, set $d^j = L^{j,max}/r^j$. Then the condition $\sum_{j=1}^{N} r^j \le C$ is equivalent to $\sum_{j=1}^{N}[r^j(t - d^j) + L^{j,max}] \le Ct$ for all $t \ge 0$. Comparing this condition with (33), we see that for the *same* delay parameters $d^j = L^{j,max}/r^j$, the schedulability condition of Core Stateless Virtual Clock follows from that of VT-EDF.

In fact, because the left hand side of (33) is a piece-wise linear function, the schedulability condition (33) can be simplified into the following set of *closed-form* conditions on the rates r^j's and delay parameters d^j's of the flows:

$$\sum_{j=1}^{N} r^j \le C$$

and for $j = 1, 2, \ldots, N$,

$$d^j \ge \frac{L^{j,max} + \sum_{m=1}^{j-1}[L^{m,max} - r^m d^m]}{C - \sum_{m=1}^{j-1} r^m}.$$

In particular, if $d^j = L^{j,max}/r^j$, $j = 1, 2, \ldots, N$, then the above conditions on d^j's hold trivially if the condition on r^j's holds, i.e, $\sum_{j=1}^{N} r^j \leq C$. Hence for this set of delay parameters, the rate condition $\sum_{j=1}^{N} r^j \leq C$ is sufficient. This is why the core stateless virtual clock can guarantee each flow a maximum delay $d^j = L^{j,max}/r^j$ as long as $\sum_{j=1}^{N} r^j \leq C$. In general, the VT-EDF scheduler can guarantee each flow a delay parameter d^j as long as the above set of conditions on the rates r^j's and delay parameters d^j's are satisfied.

5.2.2 Calendar Queue Approximation to VT-EDF

As in the case of the core stateless virtual clock scheduling algorithm, we can also design a calendar-queue approximation to the VT-EDF scheduling algorithm. This approximation scheme is referred to as the *slotted* VT-EDF scheduling algorithm. This scheme works exactly the same as the slotted $C\mathcal{S}$VC scheduling algorithm, except that the virtual finish time $\tilde{\nu}^{j,k}$ of a packet $p^{j,k}$ is computed using $\tilde{\nu}^{j,k} = \tilde{\omega}^{j,k} + d^j$ instead of $\tilde{\nu}^{j,k} = \tilde{\omega}^{j,k} + L^{j,k}/r^j + \delta^{j,k}$.

The schedulability condition for the slotted VT-EDF scheduling algorithm becomes

$$\sum_{j=1}^{N} [r^j(t + \iota - d^j) + L^{j,max}]\mathbf{1}_{\{t \geq d^j\}} \leq Ct, \text{ for any } t \geq 0.$$

We claim that if the above schedulability condition holds, then a slotted-VT-EDF scheduler can guarantee flow j its delay parameter with an error term $\Psi_{slotted\text{-}VT\text{-}EDF} = L^{*,max}/C + \iota$.

5.3 Virtual Time Rate Control and Delay-Based Scheduling Algorithms

Any delay-based scheduling algorithm is *in essence* core stateless in that it does not need to maintain *per-flow scheduling state* for determining when to service a packet so long as the delay parameter can be inferred from the packet directly (say, either from an explicit delay parameter carried by the packet or implicit from the flow label carried by the packet). What makes *conventional* delay-based scheduling algorithms such as EDF *stateful* is the need to perform *per-flow shaping* at each hop in order to support guaranteed delay services. For example, in the rate-controlled EDF (RC-EDF) scheduling algorithm, the rate controller needs to maintain per-flow state information to ensure that the flow traffic envelope process is satisfied at each hop. The VT-EDF scheduler proposed in Section 5.2.1 circumvents this problem by scheduling packets using their virtual finish time, a quantity that can be computed directly from the packet state information. An alternative approach is to use a *(core stateless) virtual time rate controller*, which is described below. Replacing the conventional rate controller with this virtual time rate controller, we can convert any rate-controlled, delay-based scheduling algorithm into a *core stateless* scheduler.

The operation of a virtual time rate controller is very simple. Upon the arrival of packet $p^{j,k}$, the virtual time rate controller assigns to it an *eligibility time* $e^{j,k}$, which is equal to its virtual time stamp, i.e., $e^{j,k} = \tilde{\omega}^{j,k}$. The packet is held at the virtual time rate controller until the real time reaches its eligibility time. The packet is then released to the delay-based scheduler. Clearly, this virtual time rate controller does not need to maintain any per-flow state information. It can be easily implemented using a sorted list and a single timer.

For any flow j, let $A^j_{RT-RC}(\tau, t)$ denote the amount of flow j traffic released by the virtual time rate controller into the scheduler over any time interval $[\tau, t]$. We claim that

$$W^j_{VT\text{-}RC}(\tau, t) \leq r^j(t - \tau) + L^{j,max}. \quad (35)$$

This is because any packet $p^{j,k}$ released by the virtual time rate controller to the scheduler during the time interval $[\tau, t]$ must satisfy the condition: $\tau \leq \tilde{\omega}^{j,k} \leq t$. Applying the Virtual Shaping Lemma in Appendix B to the time interval $[\tau, t]$, we see that (35) holds.

As an example to illustrate its usefulness, we design a new delay-based core stateless scheduler by incorporating the virtual time rate controller into the conventional static priority scheduling algorithm. The resulting core stateless scheduling algorithm is referred to as the *virtual-time rate-controlled static priority* (VT-RC-SP) scheduler. Its "stateful" counterpart is the rate-controlled static priority (RC-SP) scheduler proposed in [23].

In the following we present the schedulability condition and the error term for the VT-RC-SP scheduler. For $1 \leq q \leq M$, let D_q be a delay parameter associated with queue q, where $0 \leq D_1 < D_2 < \cdots < D_M$. Suppose a packet from flow j with a delay parameter d^j is placed into queue q if $D_q \leq d^j < D_{q+1}$. Let $\mathcal{F}_q$ be the set of flow j sharing queue q. Assuming that the following condition on D_q's holds:

$$D_q \geq \frac{\sum_{p=1}^{q} \sum_{j \in \mathcal{F}_p} L^{j,max}}{C - \sum_{p=1}^{q-1} \sum_{j \in \mathcal{F}_p} r^j} \text{ for } q = 1, 2, \ldots, M,$$

then the VT-RC-SP scheduler can guarantee each flow j in $\mathcal{F}_q$ a delay parameter $D_q \leq d^j$ with an error term $\Psi_{VT\text{-}RC\text{-}SP} = L^{*,max}/C$.

The above schedulability condition for VT-RC-SP can be established by applying the schedulability result obtained for RC-SP in [12] with the traffic envelope (35).

Lastly, we comment that we can also combine the virtual time rate controller with a rate-based scheduler. The core jitter virtual clock (CJVC) scheduling algorithm proposed in [21] and discussed in Section 5.1.2 is such an example. Note that in this case, the virtual time rate controller has a somewhat *different* definition: the eligibility time $\tilde{e}^{j,k}$ of a packet is defined as $\tilde{\omega}^{j,k} + \delta^{j,k}$ instead of $\tilde{\omega}^{j,k}$(See the Virtual Rate Control Lemma in Appendix B).

6 Static Scheduling Algorithms with Resource Pre-configuration

Another way to achieve the objective of scalable scheduling is to employ *static* scheduling algorithms. Here by a *static* scheduler we mean a scheduling algorithm which does not *directly* use flow-dependent packet state information such as the packet virtual time stamp, reserved rate or delay parameter of a flow. Examples of static scheduling algorithms are FIFO or simple priority-based scheduling schemes. In contrast, scheduling algorithms which *do* use this information are referred as *dynamic*. For example, the core stateless scheduling algorithms designed in the previous section are dynamic. Observe that static scheduling algorithms by definition are *core stateless*, as no per-flow states need to be maintained. Static scheduling algorithms are typically employed to support *traffic aggregation* and to provide *class of services.* In the following, we will use a number of examples to illustrate how static scheduling algorithms can be accommodated in our framework. In these examples, we assume that resources associated with the scheduler are *pre-configured*, namely, they are not dynamically allocated or de-allocated to flows as they arrive or depart.

6.1 FIFO

An FIFO scheduler $\mathcal{S}$ services packets in the order of their actual arrival time, regardless of their virtual time. We can view an FIFO scheduler as a delay-based scheduler with a *fictitious* delay parameter $d^j = 0$ assigned to each flow passing through the scheduler. We now determine the error term Ψ_{FIFO} for an FIFO scheduler with pre-configured service and buffer capacities.

Consider an FIFO scheduler $\mathcal{S}$ with a service capacity C and a total buffer size B. Suppose that N flows share $\mathcal{S}$, where the sum of the reserved rates $\sum_{j=1}^{N} r^j \leq C$. Furthermore, we assume that the buffer capacity B is appropriately provisioned such that no packet from any flow will ever be lost. (We will illustrate how to provision buffer and bandwidth resources for scheduling algorithms such as FIFO under the virtual time reference system in a future paper.) For any packet $p^{j,k}$, let $\hat{a}^{j,k}$ be the actual arrival time at the scheduler, and $\hat{f}^{j,k}$ be its actual departure time. It is clear that $\hat{f}^{j,k} \leq \hat{a}^{j,k} + B/C + L^{*,max}/C$. Since $\tilde{\omega}^{j,k} \geq \hat{a}^{j,k}$ and $\tilde{\nu}^{j,k} = \tilde{\omega}^{j,k}$, we have

$$\hat{f}^{j,k} \leq \tilde{\nu}^{j,k} + \frac{B}{C} + \frac{L^{*,max}}{C}.$$

Therefore $\Psi_{FIFO} = B/C + L^{*,max}/C$.

By updating the virtual time stamp of packet $p^{j,k}$ using $\tilde{\omega}^{j,k}_{next\text{-}hop} = \tilde{\omega}^{j,k} + \frac{B}{C} + \frac{L^{*,max}}{C} + \pi$, where π is the propagation delay to the next hop of packet $p^{j,k}$, it is clear that $\tilde{\omega}^{j,k}_{next\text{-}hop}$ not only preserves the virtual spacing property of packet virtual time stamps but also meets the reality check condition at the next hop.

We can also use the FIFO scheduling algorithm as a *dynamic* delay-based scheduler with a fixed delay parameter: only flows with a delay parameter d^j such that $d^j \geq B/C$ can be supported. Under this interpretation, $\Psi_{FIFO} = L^{*,max}/C$. Like other dynamic delay-based schedulers, the virtual time stamp of a packet is updated as follows: $\tilde{\omega}^{j,k}_{next\text{-}hop} = \tilde{\omega}^{j,k} + d^j + \frac{L^{*,max}}{C} + \pi$. Clearly, a scheduler needs to choose one of the two interpretations so that the packet virtual time stamps can be updated appropriately and consistently.

6.2 Static WFQ with Pre-Configured Rates

To provide more service differentiation than the simple FIFO scheduler, we can employ a fixed number of FIFO queues with pre-configured service and buffer capacities. Bandwidth sharing among the FIFO queues can be implemented using, e.g., a Weighted Fair Queuing (WFQ) scheduler, or any of its variations. Suppose we have M FIFO queues which are serviced by a WFQ scheduler with a total service capacity of C. For $q = 1, 2, \ldots, M$, let $C_q = \phi_q C$ be the pre-configured service rate of queue q, where $0 \leq \phi_q \leq 1$ and $\sum_{q=1}^{M} \phi_q = 1$. In other words, each queue q is guaranteed a minimum service rate of C_q. In addition, we assume that the buffer capacity for queue q is B_q.

Let $\mathcal{F}_q$ denote the class of flows sharing queue q. We assume the schedulability condition $\sum_{j \in \mathcal{F}_q} r^j \leq C_q$ holds for each queue q. Furthermore, the buffer capacity B_q is also appropriately provisioned such that no packet from any flow in $\mathcal{F}_q$ will ever be lost. Then using the same argument as in the FIFO case, we can show that each queue has an error term $\Psi_q = B_q/C_q + L^{*,max}/C$. Namely, for any packet $p^{j,k}$ of flow $j \in \mathcal{F}_q$,

$$\hat{f}^{j,k} \leq \tilde{\nu}^{j,k} + \frac{B_q}{C_q} + \frac{L^{*,max}}{C}$$

where as in the case of FIFO, we define $\tilde{\nu}^{j,k} = \tilde{\omega}^{j,k}$.

For $1 \leq q \leq M$, let $D_q = B_q/C_q$. Without loss of generality, assume that $0 \leq D_1 < D_2 < \cdots < D_M$. Then $\Psi_q = D_q + L^{*,max}/C$. In other words, using this static WFQ scheduler with multiple FIFO queues, we can support a fixed number of guaranteed delay classes. Note that using this multiple-queue WFQ scheme, flows from different queues can have different error terms. Hence the packet virtual time stamp must be updated accordingly, depending on from which queue it is serviced.

As in the case of FIFO, we can also view this multiple-queue WFQ scheme as a *dynamic* delay-based scheduler with a set of fixed delay parameters: a flow with a delay parameter d^j can be supported and placed into queue q, $1 \leq q \leq M$, if $D_{q-1} \leq d^j < D_q$. Under this interpretation, the multiple-queue scheme has an error term $\Psi_{FIFO} = L^{*,max}/C$. Like other dynamic delay-based schedulers, the virtual time stamp of a packet is updated as follows: $\tilde{\omega}^{j,k}_{next\text{-}hop} = \tilde{\omega}^{j,k} + d^j + \frac{L^{*,max}}{C} + \pi$.

6.3 Static Priority with Pre-Configured Rates

As an alternative to the static WFQ scheduler with multiple FIFO queues, we can also implement a static priority scheduler with multiple FIFO queues. Again suppose we have M FIFO queues, where queues of lower index have higher priority. Namely, queue 1 has the highest priority, whereas queue M has the lowest priority. The queues are serviced in the order of their priorities: whenever queue q is not empty, it is serviced before queue $q+1$, $q+2$, ..., q_M. The scheduler is assumed to be non-preemptive.

The queues of the static priority scheduler is configured and provisioned in the following manner. For $q = 1, 2, \ldots, M$, $C_q \geq 0$ is a *nominal* service rate assigned to queue q, where $\sum_{q=1}^{M} C_q = C$. The nominal service rate of a queue is used to control the aggregate rate of flows sharing the queue. Let $\mathcal{F}_q$ denote the class of flows sharing queue q, we control the number of flows sharing queue q to ensure that $\sum_{j\in\mathcal{F}_q} r^j \leq C_q$ holds for each queue q. Furthermore, the buffer capacity B_q is also appropriately provisioned such that no packet from any flow in $\mathcal{F}_q$ will ever be lost.

For $1 \leq q \leq M$, let D_q be a delay parameter associated with queue q. We assume that $0 \leq D_1 < D_2 < \cdots < D_M$. Using the result obtained in [12], we can show the following schedulability condition. If the following condition on $D_q's$ holds:

$$D_q \geq \frac{\sum_{p=1}^{q} B_p}{C - \sum_{p=1}^{q-1} C_p} \text{ for } q = 1, 2, \ldots, M,$$

then each queue has an error term $\Psi_q = \frac{B_q}{C_q} + \frac{L^{*,max}}{C}$. Namely, for any packet $p^{j,k}$ of flow $j \in \mathcal{F}_q$,

$$\hat{f}^{j,k} \leq \tilde{\nu}^{j,k} + \frac{B_q}{C_q} + \frac{L^{*,max}}{C},$$

where $\tilde{\nu}^{j,k} = \tilde{\omega}^{j,k}$.

Clearly, we can also view this static priority queue scheme as a dynamic scheduler with an error term $\Psi = L^{*,max}/C$. The decision for placing a flow into an appropriate queue and mechanism for updating packet virtual time stamps are exactly the same as used in the static WFQ scheme with multiple queues.

7 Latency-Rate Servers and the Virtual Time Reference System

The virtual time reference system proposed in this paper does not exclude the use of *stateful* scheduling algorithms, namely, those scheduling algorithms that maintain per-flow state information in order to provide guaranteed services. Such per-flow state information, for example, is imperative if "bounded fairness" in bandwidth sharing among flows is desired, in addition to delay and rate guarantees. To accommodate these stateful scheduling algorithms into our framework, it suffices to identify the error term incurred by these stateful scheduling algorithms. As an example to show how this can be done generally, we consider the class of scheduling algorithms introduced in [18, 20]—the *latency-rate servers*. This class encompasses virtually all known fair-queueing algorithms and its variations.

In defining a latency-rate server, a key notion introduced in [20] is the concept of *burst period*[6]. For any flow j, a *flow j burst period* is a maximal time interval $(\tau_1, \tau_2]$ such that for any time $t \in (\tau_1, \tau_2]$, packets

[6]Actually, the term "busy period" instead of "burst period" is used in [18, 20]. In order to avoid confusion with the standard definition and usage of "busy period" in queueing theory, we opt to use the term "burst period." Incidentally, the term "backlogged period" is used in [18, 20] to refer to the standard notion of "busy period."

of flow j arrive with rate greater than or equal to its reserved rate r^j, or

$$A^j(\tau_1, t) \geq r^j(t - \tau_1) \tag{36}$$

where $A^j(\tau_1, t)$ denote the amount of flow j traffic arriving during the timer interval $[\tau_1, t]$.

Consider a server $\mathcal{S}$. Suppose that the mth burst period of flow j starts at time τ. Let τ^* be the time that the last packet of the mth burst period of flow j departs server $\mathcal{S}$. For $\tau \leq t \leq \tau^*$, denote by $W^{j,m}(\tau, t)$ the total service provided to the packets of the pth burst period of flow j up to time t by server $\mathcal{S}$. We say $\mathcal{S}$ is a latency-rate server (with respect to flow j) if and only if for any time t, $\tau \leq t \leq \tau^*$,

$$W^{j,m}(\tau, t) \geq \max\{0, r^j(t - \tau - \Theta^j)\},$$

where Θ^j is the minimum non-negative number such that the above inequality holds. It is referred to as the *latency* of server $\mathcal{S}$.

To relate a latency-rate server to the virtual time reference system, we provide an alternative definition of the latency-rate server.

Consider the mth burst period of flow j. Let $\hat{a}^{j,k}$ denote the actual arrival time of the kth packet in the mth burst period of flow j at server $\mathcal{S}$. Then clearly, $\hat{a}^{j,1} = \tau$, where recall that τ is the beginning of the mth bursty period of flow j. For each packet $p^{j,k}$ in the mth burst period, define $\nu^{j,k}$ recursively as follows:

$$\nu^{j,1} = \hat{a}^{j,1} + \frac{L^{j,1}}{r^j} \text{ and } \nu^{j,k} = \max\{\nu^{j,k-1}, \hat{a}^{j,k}\} + \frac{L^{j,k}}{r^j} \quad k \geq 2. \tag{37}$$

From the definition of burst period, it is not too hard to see that for $k \geq 1$, we must have $\hat{a}^{j,k} \leq \nu^{j,k-1}$. In other words, $\nu^{j,k} = \nu^{j,k-1} + \frac{L^{j,k}}{r^j}$. Consequently, we have

$$r^j(\nu^{j,k} - \hat{a}^{j,1}) = \sum_{q=1}^{k} L^{j,q}. \tag{38}$$

For each packet $p^{j,k}$, let $\hat{f}^{j,k}$ be the actual time it finishes service at server $\mathcal{S}$. The following lemma provides an alternative definition of a latency-rate server, the proof of which can be found in Appendix C.

Lemma 6 *A server $\mathcal{S}$ is a latency-rate server with a latency parameter Θ^j (with respect to flow j) if and only if for any packet $p^{j,k}$ of flow j, the following inequality holds:*

$$\hat{f}^{j,k} - \nu^{j,k} \leq \Theta^j - \frac{L^{j,k}}{r^j}.$$

■

Using Lemma 6, we now determine the error term for the latency rate server $\mathcal{S}$. For each packet $p^{j,k}$ of flow j, let $\tilde{\omega}^{j,k}$ be its virtual time as it enters $\mathcal{S}$. Define its virtual finish time $\tilde{\nu}^{j,k}$ by $\tilde{\nu}^{j,k} = \tilde{\omega}^{j,k} + \tilde{d}^{j,k}$. Using the fact that $\tilde{\omega}^{j,k} \geq \hat{a}^{j,k}$ and the virtual spacing property of $\tilde{\omega}^{j,k}$, it is not too hard to prove by induction that

$$\tilde{\nu}^{j,k} \geq \nu^{j,k}, \text{for all } k \geq 1.$$

Then from Lemma 6, we have

$$\hat{f}^{j,k} \leq \tilde{\nu}^{j,k} + \Theta^j - \frac{L^{j,k}}{r^j} \leq \tilde{\nu}^{j,k} + \Theta^j.$$

Hence we see that $\mathcal{S}$ has an error term Ψ such that $\Psi \leq \Theta^j$. This leads to the following theorem.

Table 2: Error terms of latency-rate ($\mathcal{LR}$) servers.

$\mathcal{LR}$ server	PGPS/WFQ, VC, FFQ, SPFQ	SCFQ
Latency Θ^j	$\frac{L^{j,max}}{r^j} + \frac{L^{*,max}}{C}$	$\frac{L^{j,max}}{r^j} + \frac{L^{*,max}}{C}(N-1)$
Error term Ψ	$\frac{L^{*,max}}{C}$	$\frac{L^{*,max}}{C}(N-1)$

Theorem 7 *Any latency-rate server with a latency Θ^j (with respect to flow j) has an error term such that*

$$\Psi \le \Theta^j.$$

■

For several well-known scheduling algorithms studied in [18, 20], we can actually show that $\Psi = \Theta^j - \frac{L^{j,max}}{r^j}$. Θ^j and its corresponding error term for these scheduling algorithms are listed in Table 2.

8 Discussions and Related Work

8.1 Implementation Issues

So far we have focused on the theoretical foundation for the virtual time reference system. In this section we will briefly discuss issues regarding its implementation. Future work will further explore these issues.

A straightforward method to implement the virtual time reference system is to use the *dynamic packet state* (DPS) technique proposed in [21, 22]. Using this technique, the packet virtual time stamp is updated at every core router as a packet enters or departs. An alternative method is to use *static packet state.* This technique requires an explicit path set-up procedure, which can be done, for example, using a simplified RSVP or MPLS (Multiprotocol Label Switching) [5, 16]. Note that this path set-up is *different* from a reservation set-up for a flow. In fact, multiple flows can share the same path. During the path set-up, the i^{th} router along the path is configured with a parameter $D_i = \sum_{q=1}^{i-1} \Psi_q + \sum_{q=1}^{i-1} \pi_{q,q+1}$, which represents the cumulative error term and propagation delay along the path up to router i. The i^{th} router is also configured with another parameter, p_i, which is the number of rate-based schedulers along the path up to router i (exclusive). Note that both D_i and p_i are parameters that are related only to the path characteristics, and are independent of any flow traversing the path. At the network edge, the virtual time stamp (i.e., $\tilde{\omega}_1^{j,k}$) of a packet is initialized to the time it is injected into the network core. At the i^{th} router along the path of its flow, the virtual time stamp associated with the packet is computed as $\tilde{\omega}_i^{j,k} = \tilde{\omega}_1^{j,k} + D_i + C_i$, where $C_i = p_i(\frac{L^{j,k}}{r^j} + \delta^{j,k}) + (i-1-p_i)d^j$. Using this approach, we see that once the packet state is initialized at the network edge, it will not be modified or updated inside the network core. This may speed up the packet forwarding operation of core routers. In particular, for a core router whose scheduling mechanism does not use the packet virtual time stamp information (e.g., a FIFO scheduler), there is no need to compute packet virtual time stamps.

The virtual time reference system is amenable to incremental deployment. Consider, for instance, the scenario where, between two core routers which employ the VT-EDF scheduling algorithms, there is one core router which does not support packet virtual time stamps. As long as the scheduling mechanism of this router can be characterized by an error term Ψ, its effect on the virtual time can be treated as if it were part of the link propagation delay between the two virtual-time-aware routers, and can be absorbed into the propagation delay between these two routers.

A critical question in implementing the virtual time reference system is how to encode the packet state. In its most general form, the packet state contains four parameters: packet virtual time stamp $\tilde{\omega}^{j,k}$, reserved rate r^j, delay parameter d^j and virtual time adjustment term $\delta^{j,k}$. Observe that in terms of providing end-to-end delay bounds, a rate-based scheduler can be treated as if it were a delay-based scheduler with a virtual delay parameter $L^{j,max}/r^j$ (see the comment at the end of Section 3.1). Hence we can eliminate the virtual time adjustment term $\delta^{j,k}$ completely. As discussed in [21], 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. Using the last option, for example, the packet state information can be efficiently encoded using floating point representation with 17 bits [21]. Our virtual time reference system allows for additional flexibility in packet state encoding. In the case where only coarse-grain QoS guarantees (say, a fixed number of bandwidth or delay classes) is to be supported (e.g., in a DiffServ domain), the rate and delay parameters can be encoded in the TOS bits of the IP header as part of PHBs. Thus only the packet virtual time stamp needs to be carried in a separate field. It is possible to represent packet virtual time stamps *approximately*, using the notion of *slotted virtual time.* Accuracy of the approximation clearly hinges on the number of bits available to represent the virtual time. These issues will be investigated further in the future work.

8.2 QoS Provisioning and Admission Control

As stated in the introduction, one major objective of our work is to use the virtual time reference system as a QoS abstraction to decouple the data plane from the QoS control plane so as to facilitate the design of a bandwidth broker architecture for guaranteed services. In this section we briefly describe how this may be achieved. The details are left to a future paper.

In our bandwidth broker architecture, *no QoS reservation state, whether per-flow or aggregate reservation state, is maintained at any core router.* Bandwidth brokers have the topology information of a network domain and *dynamically* maintain all the QoS states regarding the flows and routers. When a request for setting QoS reservation for a flow arrives, a bandwidth broker looks up its QoS database, finds an appropriate path and checks whether sufficient resources are available at each router along the path. In case the flow can be admitted, the bandwidth broker would choose a reserved rate r and a delay parameter d for the flow, and informs the edge router to configure the edge conditioner accordingly. No QoS configuration or "state update" is needed at core routers *on a per-flow basis.* As a result, the problem of *robustness* facing the conventional hop-by-hop admission control approach [21], e.g., *inconsistent* QoS databases due to loss of signaling messages, is significantly alleviated. Because of the virtual time reference system, this bandwidth broker architecture is capable of supporting QoS provisioning and admission control with similar granularity and flexibility of the IntServ guaranteed service. The decoupling of QoS control plane and data plane enables sophisticated admission control algorithms to be employed in powerful bandwidth brokers for network-wide optimization of resource utilization. This is generally *infeasible* in the conventional hop-by-hop admission control approach. Furthermore, the bandwidth broker architecture makes it easy to implement policy-based admission control or advanced reservation.

8.3 Related Work

The idea of virtual time has been used extensively in the design of packet scheduling algorithms such as VC and WFQ. 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. In contrast, in this paper the notion of virtual time embodied by packet virtual time stamps can be viewed, in some sense, as *global* to an entire domain. Its computation is *core* *stateless*, relying only on the packet state carried by packets. Furthermore, when measured according to this "global" virtual time, the traffic flow preserves its reserved rate, due to the virtual spacing property of packet virtual time stamps.

In [21], the first core stateless scheduling algorithm, CJVC, is designed and shown to provide the same end-to-end delay bound as the stateful VC-based reference network. In addition, an aggregate reservation estimation algorithm is developed for performing admission control without per-flow state. Our work is motivated and inspired by the results in [21]. However, our work differs from [21] in several aspects. The virtual time reference system we developed is defined to serve as a unifying scheduling framework where diverse scheduling algorithms can be employed to support guaranteed services. The virtual spacing property of the packet virtual time stamps and the abstraction of core routers with error terms greatly simplify the design and management of the QoS provisioning. The notion of packet virtual time stamps we introduced also leads to the design of new core stateless scheduling algorithms. Furthermore, the virtual time reference system is defined with an aim to decouple the data plane and the QoS control plane so that a flexible and scalable QoS provisioning and control architecture can be developed at the network edge, without affecting the core of the network.

9 Conclusions and Future Work

In this paper we have proposed and developed a novel virtual time reference system as a unifying scheduling framework to provide scalable support for guaranteed services. This virtual time reference system is designed as a conceptual framework upon which guaranteed services can be implemented in a scalable manner using the DiffServ paradigm. The key construct in the proposed virtual time reference system is the notion of packet virtual time stamp, whose computation is core stateless, i.e., no per-flow states are required for its computation. In the paper, we have laid the theoretical foundation for the definition and construction of packet virtual time stamps. We described how per-hop behavior of a core router (or rather its scheduling mechanism) can be characterized via packet virtual time stamps, and based on this characterization, establish end-to-end per-flow delay bounds. Consequently, we demonstrated that, in terms of its ability to support guaranteed services, the proposed virtual time reference system has the same expressive power as the IntServ model. Furthermore, we showed that the notion of packet virtual time stamps leads to the design of new core stateless scheduling algorithms, especially work-conserving ones. In addition, our framework does not exclude the use of existing scheduling algorithms such as stateful fair queueing algorithms to support guaranteed services.

Using the virtual time reference system, we are currently designing a bandwidth broker architecture to support flexible and scalable QoS provisioning and admission control. We are also exploring various issues regarding the implementation of the virtual time reference system and its implications in QoS provisioning. In particular, we plan to investigate methods to implement a "coarse" grain version of the virtual time reference system using the current DiffServ architecture as well as MPLS.

A Proof of Theorem 1

In this appendix, we will prove Theorem 1 in Section 3. This theorem states an important recursive relation between $\Delta_i^{j,k}$ and $\Delta_i^{j,k-1}$. The proof of the theorem is based on two preliminary lemmas.

Lemma 8 below relates the cumulative queueing delay experienced by $p^{j,k}$ up to server i (i.e., $\Delta_i^{j,k}$), to either its queueing delay up to server $i-1$ (i.e., $\Delta_{i-1}^{j,k}$), or the queueing delay experienced by the previous packet $p^{j,k-1}$ up to server i (i.e., $\Delta_i^{j,k-1}$).

Lemma 8 *For $i \geq 1$ and $k \geq 2$,*

$$\Delta_i^{j,k} = \max\{\Delta_{i-1}^{j,k}, \Delta_i^{j,k-1} + i\frac{L^{j,k-1} - L^{j,k}}{r^j} + a_1^{j,k-1} - a_1^{j,k} + \frac{L^{j,k}}{r^j}\}. \tag{39}$$

In the above, we have defined $\Delta_0^{j,k} = 0$ for all k's. ∎

Proof: First note that as $f_i^{j,1} = a_i^{j,1} + \frac{L^{j,1}}{r^j} = f_{i-1}^{j,1} + \frac{L^{j,1}}{r^j}$, $\Delta_i^{j,1} = 0$ for all i, which is also intuitively obvious.

For any $k \geq 2$, we show (39) holds by induction on i.

Basis ($i = 1$). Note that $\Delta_0^{j,k} = 0$.

$$\begin{aligned}
\Delta_1^{j,k} = f_1^{j,k} - (a_1^{j,k} + \frac{L^{j,k}}{r^j}) &= \max\{a_1^{j,k}, f_1^{j,k-1}\} + \frac{L^{j,k}}{r^j} - (a_1^{j,k} + \frac{L^{j,k}}{r^j}) \\
&= \max\{a_1^{j,k}, a_1^{j,k-1} + \frac{L^{j,k-1}}{r^j} + \Delta_1^{j,k-1}\} + \frac{L^{j,k}}{r^j} - (a_1^{j,k} + \frac{L^{j,k}}{r^j}) \\
&= \max\{0, \Delta_1^{j,k-1} + \frac{L^{j,k-1} - L^{j,k}}{r^j} + a_1^{j,k-1} - a_1^{j,k} + \frac{L^{j,k}}{r^j}\}.
\end{aligned}$$

Inductive Step. Now suppose (39) holds true up to server i for $k \geq 2$. We show that it is also true at server $i + 1$ for $k \geq 2$.

$$\Delta_{i+1}^{j,k} = f_{i+1}^{j,k} - [a_1^{j,k} + (i+1)\frac{L^{j,k}}{r^j}] = \max\{a_{i+1}^{j,k}, f_{i+1}^{j,k-1}\} + \frac{L^{j,k}}{r^j} - [a_1^{j,k} + (i+1)\frac{L^{j,k}}{r^j}]. \tag{40}$$

Now we consider two cases according to the relationship between the arrival time of packet $p^{j,k}$ and the finish time of $p^{j,k-1}$ at the server $i + 1$.

Case 1: $a_{i+1}^{j,k} \leq f_{i+1}^{j,k-1}$. Then by definition,

$$f_{i+1}^{j,k-1} = \Delta_{i+1}^{j,k-1} + a_1^{j,k-1} + (i+1)\frac{L^{j,k-1}}{r^j}.$$

Therefore, from (40), we have

$$\Delta_{i+1}^{j,k} = \Delta_{i+1}^{j,k-1} + (i+1)\frac{L^{j,k-1} - L^{j,k}}{r^j} + a_1^{j,k-1} - a_1^{j,k} + \frac{L^{j,k}}{r^j}.$$

Note that, in this case, we have

$$\begin{aligned}
&\Delta_{i+1}^{j,k-1} + (i+1)\frac{L^{j,k-1} - L^{j,k}}{r^j} + a_1^{j,k-1} - a_1^{j,k} + \frac{L^{j,k}}{r^j} \\
= \; & f_{i+1}^{j,k-1} - [a_1^{j,k-1} + (i+1)\frac{L^{j,k-1}}{r^j}] + (i+1)\frac{L^{j,k-1} - L^{j,k}}{r^j} + a_1^{j,k-1} - a_1^{j,k} + \frac{L^{j,k}}{r^j} \\
= \; & f_{i+1}^{j,k-1} - (a_1^{j,k} + i\frac{L^{j,k}}{r^j}) \\
\geq \; & a_{i+1}^{j,k} - (a_1^{j,k} + i\frac{L^{j,k}}{r^j}) \\
= \; & f_i^{j,k} - (a_1^{j,k} + i\frac{L^{j,k}}{r^j}) \\
= \; & \Delta_i^{j,k}
\end{aligned}$$

Hence (39) holds.

Case 2: $a_{i+1}^{j,k} > f_{i+1}^{j,k-1}$. Then since

$$a_{i+1}^{j,k} = f_i^{j,k} = \Delta_i^{j,k} + a_1^{j,k} + i\frac{L^{j,k}}{r^j},$$

we have

$$\Delta_{i+1}^{j,k} = \Delta_i^{j,k}.$$

Furthermore,

$$\begin{aligned}
& \Delta_{i+1}^{j,k-1} + (i+1)\frac{L^{j,k-1} - L^{j,k}}{r^j} + a_1^{j,k-1} - a_1^{j,k} + \frac{L^{j,k}}{r^j} \\
= \; & f_{i+1}^{j,k-1} - [a_1^{j,k-1} + (i+1)\frac{L^{j,k-1}}{r^j}] + (i+1)\frac{L^{j,k-1} - L^{j,k}}{r^j} + a_1^{j,k-1} - a_1^{j,k} + \frac{L^{j,k}}{r^j} \\
= \; & f_{i+1}^{j,k-1} - (a_1^{j,k} + i\frac{L^{j,k}}{r^j}) \\
< \; & a_{i+1}^{j,k} - (a_1^{j,k} + i\frac{L^{j,k}}{r^j}) \\
= \; & f_i^{j,k} - (a_1^{j,k} + i\frac{L^{j,k}}{r^j}) \\
= \; & \Delta_i^{j,k}
\end{aligned}$$

Hence (39) also holds in this case. This completes the proof of Lemma 8. ■

From the above proof, we observe that for $i \geq 2$ and $k \geq 2$, if $a_i^{j,k} > f_i^{j,k-1}$, we have

$$\Delta_i^{j,k} = \Delta_{i-1}^{j,k}, \tag{41}$$

otherwise,

$$\Delta_i^{j,k} = \Delta_i^{j,k-1} + i\frac{L^{j,k-1} - L^{j,k}}{r^j} + a_1^{j,k-1} - a_1^{j,k} + \frac{L^{j,k}}{r^j}. \tag{42}$$

Note also that $\Delta_{i+1}^{j,k} - \Delta_i^{j,k}$ represents the queueing delay experienced by packet $p^{j,k}$ at server $i+1$. The following lemma shows that as a packet goes through the servers in the ideal reference system, the queueing delay it experiences at each individual server cannot decrease.

Lemma 9 *For $i \geq 1$ and $k \geq 2$,*

$$\Delta_{i+1}^{j,k} - \Delta_i^{j,k} \geq \Delta_i^{j,k} - \Delta_{i-1}^{j,k}. \tag{43}$$

(Note that $\Delta_0^{j,k} = 0$.) ■

Proof: Proof by induction on k.

Basis($k = 1$). Since $\Delta_i^{j,1} = 0$, the inequality (43) holds trivially.

Inductive Step. Now suppose (43) holds for $k-1$, $k \geq 2$. We show it also holds for k.

$$\begin{aligned}
\Delta_{i+1}^{j,k} - \Delta_i^{j,k} &= f_{i+1}^{j,k} - [a_1^{j,k} + (i+1)\frac{L^{j,k}}{r^j}] - [f_i^{j,k} - (a_1^{j,k} + i\frac{L^{j,k}}{r^j})] \\
&= f_{i+1}^{j,k} - f_i^{j,k} - \frac{L^{j,k}}{r^j} \\
&= (\max\{a_{i+1}^{j,k}, f_{i+1}^{j,k-1}\} + \frac{L^{j,k}}{r^j}) - (\max\{a_i^{j,k}, f_i^{j,k-1}\} + \frac{L^{j,k}}{r^j}) - \frac{L^{j,k}}{r^j} \\
&= \max\{a_{i+1}^{j,k}, f_{i+1}^{j,k-1}\} - \max\{a_i^{j,k}, f_i^{j,k-1}\} - \frac{L^{j,k}}{r^j}.
\end{aligned}$$

We consider two cases.

CASE 1: $a_i^{j,k} > f_i^{j,k-1}$. Under this condition, we have

$$\begin{aligned}\Delta_{i+1}^{j,k} - \Delta_i^{j,k} &= \max\{a_{i+1}^{j,k}, f_{i+1}^{j,k-1}\} - a_i^{j,k} - \frac{L^{j,k}}{r^j} \\ &\geq a_{i+1}^{j,k} - a_i^{j,k} - \frac{L^{j,k}}{r^j}.\end{aligned}$$

Note that for $i > 1$,

$$a_i^{j,k} = f_{i-1}^{j,k} = \Delta_{i-1}^{j,k} + a_1^{j,k} + (i-1)\frac{L^{j,k}}{r^j} \tag{44}$$

where as $\Delta_0^{j,k} = 0$, the last equality above also holds for $i = 1$.

On the other hand,

$$a_{i+1}^{j,k} = f_i^{j,k} = \Delta_i^{j,k} + a_1^{j,k} + i\frac{L^{j,k}}{r^j} \tag{45}$$

From (44) and (45), we have

$$\Delta_{i+1}^{j,k} - \Delta_i^{j,k} \geq \Delta_i^{j,k} - \Delta_{i-1}^{j,k}.$$

CASE 2: $a_i^{j,k} \leq f_i^{j,k-1}$. Under this condition, we have

$$\begin{aligned}\Delta_{i+1}^{j,k} - \Delta_i^{j,k} &= \max\{a_{i+1}^{j,k}, f_{i+1}^{j,k-1}\} - f_i^{j,k-1} - \frac{L^{j,k}}{r^j} \\ &\geq f_{i+1}^{j,k-1} - f_i^{j,k-1} - \frac{L^{j,k}}{r^j}.\end{aligned}$$

Since $f_{i+1}^{j,k-1} = \Delta_{i+1}^{j,k-1} + a_1^{j,k-1} + (i+1)\frac{L^{j,k-1}}{r^j}$ and $f_i^{j,k-1} = \Delta_i^{j,k-1} + a_1^{j,k-1} + i\frac{L^{j,k-1}}{r^j}$, we have

$$\begin{aligned}\Delta_{i+1}^{j,k} - \Delta_i^{j,k} &\geq \Delta_{i+1}^{j,k-1} - \Delta_i^{j,k-1} + \frac{L^{j,k-1} - L^{j,k}}{r^j} \\ &\geq \Delta_i^{j,k-1} - \Delta_{i-1}^{j,k-1} + \frac{L^{j,k-1} - L^{j,k}}{r^j} \text{ (from inductive hypothesis)} \\ &= \Delta_i^{j,k} - (i\frac{L^{j,k-1} - L^{j,k}}{r^j} + a_1^{j,k-1} - a_1^{j,k} + \frac{L^{j,k}}{r^j}) - \Delta_{i-1}^{j,k-1} + \frac{L^{j,k-1} - L^{j,k}}{r^j} \text{ (from (42))} \\ &= \Delta_i^{j,k} - (\Delta_{i-1}^{j,k-1} + (i-1)\frac{L^{j,k-1} - L^{j,k}}{r^j} + a_1^{j,k-1} - a_1^{j,k} + \frac{L^{j,k}}{r^j}) \\ &\geq \Delta_i^{j,k} - \Delta_{i-1}^{j,k} \text{ (from Lemma 8)}\end{aligned}$$

■

A direct consequence of Lemma 9 is that if for some i, $1 \leq i \leq h$, $\Delta_{i+1}^{j,k} = \Delta_i^{j,k}$, then $\Delta_{i+1}^{j,k} = \Delta_i^{j,k} = \Delta_{i-1}^{j,k} = \cdots = \Delta_1^{j,k} = \Delta_0^{j,k} = 0$. In other words, for any packet $p^{j,k}$, either $\Delta_i^{j,k} = 0$ for $i = 1, 2, \ldots, h$ (i.e., no queueing delay is experienced by the packet); or there exists i^*, $1 \leq i^* \leq h$, such that $\Delta_i^{j,k} = 0$ for $i = 1, \ldots, i^* - 1$ and $\Delta_i^{j,k} > 0$ for $i = i^*, i^* + 1, \ldots, h$ (i.e., the packet starts experiencing queueing delay at server i^* and onwards). Intuitively, the latter case happens because the packets preceding packet $p^{j,k}$ have eventually accumulated enough delay at server i^* to affect packet $p^{j,k}$ (see Figure 4(b)). Applying the above fact to Lemma 8, we see that Theorem 1 holds.

Another consequence of Lemma 9 is the following corollary.

Corollary 10 *For any $p \le h$ and $k = 1, 2, \ldots,$ we have*

$$\frac{\Delta_p^{j,k}}{p} \le \frac{\Delta_h^{j,k}}{h}. \tag{46}$$

Proof: Note that for $k = 1$, since $\Delta_p^{j,1} = \Delta_h^{j,1} = 0$, (46) holds trivially. For $k \ge 2$, let $\delta_p^{j,k} = \Delta_p^{j,k}/p$. Then $\Delta_p^{j,k} = p\delta_p^{j,k}$. From Lemma 9, it is easy to see that $\Delta_p^{j,k} - \Delta_{p-1}^{j,k} \ge \delta_p^{j,k}$. Furthermore, for any $i = p+1, \ldots, h$, we have $\Delta_i^k - \Delta_{i-1}^k \ge \Delta_p^{j,k} - \Delta_{p-1}^{j,k} \ge \delta_p^{j,k}$. Hence

$$\Delta_h^{j,k} = \sum_{i=p+1}^{h} (\Delta_i^k - \Delta_{i-1}^k) + \Delta_p^{j,k} \ge (h-p)\delta_p^{j,k} + p\delta_p^{j,k} = h\delta_p^{j,k}.$$

Therefore (46) also holds for any $k \ge 2$. ■

We now apply Theorem 1 to derive the following important corollary.

Corollary 11 *For any $k \ge 1$, and $i = 1, 2, \ldots, h$,*

$$\Delta_i^{j,k} + i\frac{L^{j,k}}{r^j} \le i\frac{L^{j,max}}{r^j}. \tag{47}$$

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

Proof: Since $\Delta_i^{j,1} = 0$, the inequality (47) holds trivially. Suppose (47) holds for $1, 2, ..., k-1$, we show that it also holds for k. Note that if $\Delta_i^{j,k} = 0$, then (47) holds trivially. Now consider the case $\Delta_i^{j,k} > 0$. From Theorem 1, we have

$$\begin{aligned}
\Delta_i^{j,k} + i\frac{L^{j,k}}{r^j} &= \Delta_i^{j,k-1} + i\frac{L^{j,k-1} - L^k}{r^j} + a_1^{j,k-1} - a_1^k + \frac{L^{j,k}}{r^j} + i\frac{L^{j,k}}{r^j} \\
&= \Delta_i^{j,k-1} + i\frac{L^{j,k-1}}{r^j} + a_1^{j,k-1} - a_1^k + \frac{L^{j,k}}{r^j}. \qquad (48)
\end{aligned}$$

From (3) and the inductive hypothesis, we see that (47) also holds for k.

■

B Virtual Shaping Lemma and Its Applications

An important consequence of the virtual spacing property of packet virtual time stamps is the following *Virtual Shaping Lemma.* Intuitively, it states that according to the virtual time, the amount of flow j traffic arriving at server j is "constrained" by its reserved rate r^j. This lemma is critical in designing delay-based core stateless scheduling algorithms that can provide delay guarantees *without explicit rate control or reshaping within the network core.*

Lemma 12 (Virtual Shaping Lemma) *Consider an arbitrary time interval $[\tau, t]$. Let $\tilde{T}^j$ denote the set of the packets of flow j which virtually arrives during $[\tau, t]$, i.e., $k \in \tilde{T}^j$ if and only if $\tau \le \tilde{\omega}^{j,k} \le t$. Let $\tilde{A}^j(\tau, t) = \sum_{k \in \tilde{T}^j} L^k$ be the corresponding amount of traffic of flow j arriving virtually between the time interval $[\tau, t]$. Then*

$$\tilde{A}^j(\tau, t) = \sum_{k \in \tilde{T}^j} L^{j,k} \le r^j(t - \tau) + L^{j,max}$$

Proof: Let k_0 be the smallest index k of the packets $p^{j,k}$ in $\tilde{A}^j$ and k_1 be the largest index k of the packets $p^{j,k}$ in $\tilde{A}^j$. From $k = k_1$ down to k_0, recursively applying the fact that $\tilde{\omega}^{j,k} \geq \tilde{\omega}^{j,k-1} + \frac{L^{j,k}}{r^j}$, we have

$$t \geq \tilde{\omega}^{j,k_1} \geq \tilde{\omega}^{j,k_1-1} + \frac{L^{j,k_1}}{r^j} \geq \tilde{\omega}^{j,k_0} + \frac{\sum_{k=k_0+1}^{k_1} L^{j,k}}{r^j} \geq \tau + \frac{\sum_{k=k_0+1}^{k_1} L^{j,k}}{r^j}.$$

Hence,

$$\tilde{A}^j(\tau, t) = \sum_{k \in \tilde{T}^j} L^{j,k} = \sum_{k=k_0+1}^{k_1} L^{j,k} + L^{j,k_0} \leq r^j(t-\tau) + L^{j,k_0} \leq r^j(t-\tau) + L^{j,max}.$$

■

Using the Virtual Shaping Lemma, we now prove that the VT-EDF scheduling algorithm we designed in Section 5 has the minimum error term $\Psi_{VT\text{-}EDF} = L^{*,max}/C$ (see Theorem 5).

Proof of Theorem 5: Fix an arbitrary packet $p^{j,k}$ from a flow j. We show that (34) holds.

Consider the system busy period containing packet $p^{j,k}$. Without loss of generality, assume that the beginning of the system busy period starts at time 0. Let τ be the last time before the arrival of packet $p^{j,k}$ such that the scheduler *starts* servicing a packet whose virtual finish time is larger than that of $p^{j,k}$ (i.e., $\tilde{\nu}^{j,k}$). If such a packet does not exist, set $\tau = 0$, the beginning of the busy period. Hence $0 \leq \tau \leq \hat{a}^{j,k} \leq \tilde{\omega}^{j,k}$.

Let $t = \tilde{\omega}^{j,k} + d^j - \tau$. Since $\tilde{\omega}^{j,k} \geq \hat{a}^{j,k} \geq \tau$ and $d^j \geq d^1 \geq 0$, we have that

$$t = \tilde{\omega}^{j,k} + d^j - \tau \geq d_1 \geq 0. \tag{49}$$

For each flow m, $m = 1, 2 \ldots, N$, let S^m denote the set of packets of flow m that are serviced after τ and but no later than packet $p^{j,k}$ (i.e., they are serviced during the time interval $(\tau, \hat{f}^{j,k}]$). From the definition of τ and S^m, we see that for any $p^{m,l} \in S^m$, $\hat{a}^{m,l} \geq \tau$ and $\tilde{\nu}^{m,l} \leq \tilde{\nu}^{j,k}$ hold. Since $\tilde{\omega}^{m,l} \geq \hat{a}^{m,l}$, $\tilde{\nu}^{m,l} = \tilde{\omega}^{m,l} + d^m$ and $\tilde{\nu}^{j,k} = \tilde{\omega}^{j,k} + d^j = t + \tau$, we have

$$\tau \leq \tilde{\omega}^{m,l} \leq t + \tau - d^m. \tag{50}$$

From (50), it is clear that if $d^m > t = \tilde{\omega}^{j,k} + d^j - \tau$, then $S^m = \emptyset$. Therefore for any m such that $S^m \neq \emptyset$, we must have $d^m \leq t$. Applying the Virtual Shaping Lemma to flow m over the time interval $[\tau, t - d^m + \tau]$, we have

$$\sum_{l \in S^m} L^{m,l} \leq L^{m,max} + r^m(t - d^m).$$

Summing over all m and using the schedulability condition (33), we have

$$\sum_{m=1}^{N} \Big(\sum_{l \in S^m} L^{m,l}\Big) \mathbf{1}_{\{S^m \neq \emptyset\}} \leq \sum_{m=1}^{N} [L^{m,max} + r^m(t - d^m)] \mathbf{1}_{\{t \geq d^m\}} \leq Ct$$

Note that packet $p^{j,k} \in S^j$. Hence packet $p^{j,k}$ must have departed the scheduler when all the packets in $\cup_{m=1}^{N} S^m$ have been serviced by the server. Furthermore, the packet which is being serviced at time τ (if it exists) has a size of at most $L^{*,max}$. Therefore, we have

$$\hat{f}^{j,k} \leq \tau + \frac{L^{*,max}}{C} + \frac{\sum_{m=1}^{N} (\sum_{l \in S^m} L^{m,l}) \mathbf{1}_{\{S^m \neq \emptyset\}}}{C} \leq \tau + \frac{L^{*,max}}{C} + t = \tilde{\omega}^{j,k} + d^j + \frac{L^{*,max}}{C} = \tilde{\nu}^{j,k} + \frac{L^{*,max}}{C}.$$

■

For a rate-based scheduler $\mathcal{S}$, an alternative form of the Virtual Shaping Lemma can be established. Let $\tilde{\omega}^{j,k}$ be the virtual time stamp associated with $p^{j,k}$ at the entry point of $\mathcal{S}$. Define the *virtual eligibility time* $\tilde{e}^{j,k}$ of packet $p^{j,k}$ to be $\tilde{e}^{j,k} = \tilde{\omega}^{j,k} + \delta^{j,k}$. Then the virtual finish time of packet $p^{j,k}$ is equal to

$$\tilde{\nu}^{j,k} = \tilde{\omega}^{j,k} + \tilde{d}^{j,k} = \tilde{e}^{j,k} + \frac{L^{j,k}}{r^j}. \tag{51}$$

Using a similar proof as in the proof of (15) in Theorem 2, we can show that

$$\tilde{e}^{j,k} \geq \tilde{\nu}^{j,k-1}. \tag{52}$$

The intuitive meaning of $\tilde{e}^{j,k}$ can be interpreted as follows. Imagine that there were a *virtual rate controller* attached to the rate-based scheduler $\mathcal{S}$. Packet $p^{j,k}$ arriving at the virtual time $\tilde{\omega}^{j,k}$ would be held at the virtual rate controller for $\delta^{j,k}$ amount of time and released to the server $\mathcal{S}$ at the virtual eligible time $\tilde{e}^{j,k} = \tilde{\omega}^{j,k} + \delta^{j,k}$. The packet were then to finish its service at the scheduler $\mathcal{S}$ by the virtual finish time $\tilde{\nu}^{j,k}$. From (52), we see that a packet were never released to be serviced at the scheduler $\mathcal{S}$ before its previous packet had finished its service. Using this observation, we establish the following alternative form of the Virtual Shaping Lemma — the *Virtual Rate Control Lemma.*

Lemma 13 (Virtual Rate Control Lemma) *Consider an arbitrary time interval $[\tau, t]$. We say that packet $p^{j,k}$ of flow j is* virtually eligible *for service during $[\tau, t]$ if $\tilde{e}^{j,k} \geq \tau$ and $\tilde{\nu}^{j,k} = \tilde{e}^{j,k} + L^{j,k}/r^j \leq t$. Let $\tilde{S}^j$ denote the set of the packets of flow j which are virtually eligible for service in $[\tau, t]$. Define $\tilde{W}^j(\tau, t) = \sum_{k \in \tilde{S}^j} L^k$. We refer to $\tilde{W}^j(\tau, t)$ as the* virtual eligible work *of flow j over $[\tau, t)$. Then*

$$\tilde{W}^j(\tau, t) = \sum_{k \in \tilde{S}^j} L^{j,k} \leq r^j(t - \tau). \tag{53}$$

Proof: Let k_0 be the smallest index k of the packets p^k in $\tilde{S}^j$ and k_1 be the largest index k of the packets $p^{j,k}$ in $\tilde{S}^j$. From $k = k_1$ down to k_0, recursively applying the fact that $\tilde{e}^{j,k} \geq \tilde{\nu}^{j,k-1} = \tilde{e}^{j,k-1} + L^{j,k-1}/r^j$, we have

$$t \geq \tilde{\nu}^{j,k_1} = \tilde{e}^{j,k_1} + \frac{L^{j,k_1}}{r^j} \geq \tilde{e}^{j,k_0} + \frac{\sum_{k=k_0}^{k_1} L^{j,k}}{r^j} \geq \tau + \frac{\sum_{k \in \tilde{S}^j} L^{j,k}}{r^j}.$$

Hence (53) follows. ∎

The Virtual Rate Control Lemma states that the amount of virtual eligible work of a flow over any time interval is always limited by its reserved rate. This lemma enables us to design rate-based core stateless scheduling algorithms that can support rate guarantees *without using explicit rate control within the network core.* One such an example is the $C_{\mathcal{S}}$VC scheduling algorithm we designed in Section 5. We now prove that it has the minimum error term $\Psi_{C_{\mathcal{S}}VC} = L^{*,max}/C$ (see Theorem 3).

Proof of Theorem 3: Fix an arbitrary packet $p^{j,k}$ from any flow j. We show that (31) holds.

Consider the system busy period containing packet p_j^k. Without loss of generality, we assume that the beginning of the system busy period starts at time 0. Let τ be the last time before the arrival of packet $p^{j,k}$ such that the scheduler *starts* servicing a packet whose virtual finish time is larger than that of $p^{j,k}$. (In other words, no packet queued at time τ has a virtual finish time smaller than $\tilde{\nu}^{j,k}$.) If such a packet does not exist, set $\tau = 0$, the beginning of the busy period. Hence $0 \leq \tau \leq \hat{a}^{j,k}$.

For each flow m, $m = 1, 2 \ldots, N$, let S^m denote the set of packets of flow m that are serviced by the scheduler after τ but no later than $\hat{f}^{j,k}$, i.e., they are serviced during the time interval $(\tau, \hat{f}^{j,k}]$. From the definition of τ, we have that for any $p^{m,l} \in S^m$, $\hat{a}^{m,l} \geq \tau$ and $\tilde{\nu}^{m,l} \leq \tilde{\nu}^{j,k}$. Now applying the Virtual Rate Control Lemma to flow m over the time interval $[\tau, \tilde{\nu}^{j,k}]$, we have

$$\sum_{l \in S^m} L^{m,l} \leq \tilde{W}^j(\tau, \tilde{\nu}^{j,k}) \leq r^m(\tilde{\nu}^{j,k} - \tau).$$

Summing over all m and using the schedulability condition, we have

$$\sum_{m=1}^{N} \sum_{l \in S^m} L^{m,l} \leq (\sum_{m=1}^{N} r^m)(\tilde{\nu}^{j,k} - \tau) \leq C(\tilde{\nu}^{j,k} - \tau).$$

Note that packet $p^{j,k} \in S^j$. Hence packet $p^{j,k}$ must have departed the scheduler when all the packets in $\cup_{m=1}^{N} S^m$ have been serviced by the server. Furthermore, the packet which is being serviced at time τ (if it exists) has a size of at most $L^{*,max}$. Therefore,

$$\hat{f}^{j,k} \leq \tau + \frac{L^{*,max}}{C} + \frac{\sum_{m=1}^{N} \sum_{l \in S^m} L^{m,l}}{C} \leq \tilde{\nu}^{j,k} + \frac{L^{*,max}}{C}.$$

■

Theorem 4 in Section 5 can also be proved using the Virtual Rate Control Lemma.

Proof of Theorem 4: We prove that a slotted $C_{\not{S}}$VC scheduler has an error term $\Psi_{slotted-C_{\not{S}}VC} = L^{*,max}/C + \iota$. To proceed, consider an arbitrary packet $p^{j,k}$ from a flow j.

Consider the system busy period containing packet $p^{j,k}$. Suppose that $\tau_p \leq \tilde{\nu}^{j,k} < \tau_{p+1}$. Without loss of generality, we assume that the system busy period starts at time 0. Let τ be the last time before the arrival of packet $p^{j,k}$ such that the scheduler starts servicing a packet from a queue $\tau_{p'}$ such as $\tau_{p'} > \tau_p$. If such a packet does not exist, set $\tau = 0$, the beginning of the busy period. Hence $0 \leq \tau \leq \hat{a}^{j,k}$. For each flow m, $m = 1, 2 \ldots, N$, let S^m denote the set of packets of flow m that are serviced after τ and no later than packet $p^{j,k}$ (i.e., they are serviced during the time interval $(\tau, \hat{f}^{j,k}])$. From the definition of τ, we see that for any $p^{m,l} \in S^m$, $\tilde{\omega}^{m,l} \geq \hat{a}^{m,l} \geq \tau$ and $\tilde{\nu}^{m,l} < \tau_{p+1} = \tau_p + \iota$. Applying the Virtual Rate Control Lemma to flow m over the time interval $[\tau, \tau_p + \iota]$, we have

$$\sum_{l \in S^m} L^{m,l} \leq \tilde{W}^m(\tau, \tau_p + \iota) \leq r^m(\tau_p + \iota - \tau).$$

Summing over all m and using the schedulability condition, we have

$$\sum_{m=1}^{N} \sum_{l \in S^m} L^{m,l} \leq (\sum_{m=1}^{N} r^m)(\tau_p + \iota - \tau) \leq C(\tau_p + \iota - \tau).$$

Note that packet $p^{j,k} \in S^j$. Hence packet $p^{j,k}$ must depart the scheduler after all the packets in $\cup_{m=1}^{N} S^m$ have been serviced by the server. Furthermore, the packet which is being serviced at time τ (if it exists) has a size of at most $L^{*,max}$. Therefore,

$$\hat{f}^{j,k} \leq \tau + \frac{L^{*,max}}{C} + \frac{\sum_{m=1}^{N} \sum_{l \in S^m} L^{m,l}}{C} \leq \tau_p + \iota + \frac{L^{*,max}}{C} \leq \tilde{\nu}^{j,k} + \iota + \frac{L^{*,max}}{C}.$$

■

C An Alternative Definition of Latency-Rate Servers

In this appendix, we show that the alternative definition of latency-rate servers is indeed equivalent to the one given in [18, 20].

Proof of Lemma 6: Consider the q^{th} burst period of flow j. Let $p^{j,1}, p^{j,2}, \ldots, p^{j,m}$ denote the first, the second, ..., and the last packet of flow j arriving during this burst period. Let $\hat{a}^{j,1}, \hat{a}^{j,2}, \ldots, \hat{a}^{j,m}$ and $\hat{f}^{j,1}, \hat{f}^{j,2}, \ldots, \hat{f}^{j,m}$ denote the actual arrival and departure times of these packets. By the definition of latency-rate server, we have that for any t, $\hat{a}^{j,1} \leq t \leq \hat{f}^{j,m}$,

$$W^{j,q}(\hat{a}^{j,1}, t) \geq r^j(t - \hat{a}^{j,1} - \Theta^j). \tag{54}$$

For $k = 1, 2, \ldots, m$, let $\nu^{j,k}$ be given in (37). We show that (54) is equivalent to

$$\hat{f}^{j,k} \leq \nu^{j,k} + \Theta^j - \frac{L^{j,k}}{r^j}. \tag{55}$$

We first show that (54) implies (55). For any $k = 1, 2 \ldots, m$, let $t = \hat{f}^{j,k} - \epsilon$, where $0 < \epsilon < \hat{f}^{j,k} - \hat{f}^{j,k-1}$. Since by time t, the scheduler has not finished servicing packet $p^{j,k}$ yet, we have

$$W^{j,q}(\hat{a}^{j,1}, t) = W^{j,q}(\hat{a}^{j,1}, \hat{f}^{j,k-1}) = \sum_{l=1}^{k-1} L^{j,l}.$$

From (54), we have

$$\sum_{l=1}^{k-1} L^{j,l} \geq r^j(t - \hat{a}^{j,1} - \Theta^j). \tag{56}$$

On the other hand, $\nu^{j,k-1} - \hat{a}^{j,1} = \frac{\sum_{l=1}^{j,k-1} L^{j,l}}{r^j}$. Therefore

$$t \leq \nu^{j,k-1} + \Theta^j \leq \nu^{j,k} - \frac{L^{j,k}}{r^j} + \Theta^j.$$

Letting $\epsilon \to 0$ yields (55).

We now show that (55) implies (54). For any t, $\hat{f}^{j,k-1} \leq t < \hat{f}^{j,k}$, we have

$$\begin{aligned}
W^{j,q}(\hat{a}^{j,1}, t) &\geq W^{j,q}(\hat{a}^{j,1}, \hat{f}^{j,k}) - L^{j,k} \\
&= \sum_{l=1}^{k} L^{j,l} - L^{j,k} \\
&= r^j(\nu^{j,k} - \hat{a}^{j,1}) - L^{j,k} \\
&\geq r^j(\hat{f}^{j,k} + \frac{L^{j,k}}{r^j} - \Theta^j - \hat{a}^{j,1}) - L^{j,k} \\
&= r^j(\hat{f}^{j,k} - \hat{a}^{j,1} - \Theta^j) \geq r^j(t - \hat{a}^{j,1} - \Theta^j).
\end{aligned}$$

This completes the proof of Lemma 6. ■

References

[1] Y. Bernet, J. Binder, S. Blake, M. Carlson, B. E. Carpenter, S. Keshav, E. Davies, B. Ohlman, D. Verma, Z. Wang, and W. Weiss. A framework for differentiated services. Internet Draft, February 1999. Work in Progress.

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. RFC 2475, December 1998.

[3] R. Braden, D. Clark, and S. Shenker. Integrated services in the internet architecture: An overview. RFC 1633, June 1994.

[4] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource reservation protocol (rsvp) – version 1 functional specification. RFC 2205, September 1997.

[5] R. Callon, P. Doolan, N. Feldman, A. Fredette, G. Swallow, and A. Viswanathan. A framework for multiprotocol label switching. Internet Draft, September 1999. Work in Progress.

[6] D. D. Clark, S. Shenker, and L. Zhang. Supporting real-time applications in an integrated services packet network:architecture and mechanism. In *Proc. ACM SIGCOMM*, August 1992.

[7] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *Proceedings of Sigcomm'89: Communication Architectures and Protocols*, pages 1–12, Austin, TX, September 1989.

[8] N. Figueira and J. Pasquale. An upper bound on delay for the Virtual Clock service discipline. *IEEE/ACM Transactions on Networking*, 3(4):399–408, August 1995.

[9] L. Georgiadis, R. Guérin, and A. Parekh. Optimal multiplexing on a single link: Delay and buffer requirements. In *Proc. IEEE INFOCOM*, pages 524–532, 1994.

[10] L. Georgiadis, R. Guérin, V. Peris, and K. N. Sivarajan. Efficient network qos provisioning based on per node traffic shaping. *IEEE/ACM Transactions on Networking*, 4(4):482–501, 1996.

[11] J. Liebeherr and D. E. Wrege. A versatile packet multiplexer for quality-of-service networks. In *Proc. 4th International Symposium on High Performance Distributed Computing (HPDC-4)*, pages 148–155, August 1995.

[12] J. Liebeherr, D. E. Wrege, and D. Ferrari. Exact admission control for networks with a bounded delay service. *IEEE/ACM Transactions on Networking*, 4(6):885–901, 1996.

[13] K. Nichols, V. Jacobson, and L. Zhang. A two-bit differentiated services architecture for the internet. Internet Draft, November 1997. Work in Progress.

[14] 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 Transactions on Networking*, 1(3):344–357, 1993.

[15] 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 Transactions on Networking*, 2(2):137–150, 1994.

[16] E. C. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. Internet Draft, August 1999. Work in Progress.

[17] S. Shenker, C. Partridge, and R. Guérin. Specification of guaranteed quality of service. RFC 2212, September 1997.

[18] D. Stiliadis. *Traffic Scheduling in Packet-Switched Networks: Analysis, Design, and Implementation.* PhD thesis, Computer Science and Engineering Department, University of California at Santa Cruz, June 1996.

[19] D. Stiliadis and A. Varma. Efficient fair queueing algorithms for packet-switched networks. *IEEE/ACM Transactions on Networking*, 6(2):175–185, 1998.

[20] D. Stiliadis and A. Varma. Latency-rate servers: A general model for analysis of traffic scheduling algorithms. *IEEE/ACM Transactions on Networking*, 6(5):611–624, 1998.

[21] I. Stoica and H. Zhang. Providing guaranteed services without per flow management. In *Proceedings of ACM SIGCOMM 1999*, Boston, MA, September 1999.

[22] I. Stoica, H. Zhang, S. Shenker, R. Yavatkar, D. Stephens, A. Malis, Y. Bernet, Z. Wang, F. Baker, J. Wroclawski, C. Song, and R. Wilder. Per hop behaviors based on dynamic packet states. Internet Draft, February 1999. Work in Progress.

[23] H. Zhang and D. Ferrari. Rate-controlled static-priority queueing. In *IEEE INFOCOM'93*, pages 227–236, April 1993.

[24] L. Zhang. Virtual clock: A new traffic control algorithm for packet switching networks. In *Proc. ACM SIGCOMM*, pages 19–29, September 1990.

[25] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A new resource reservation protocol. *IEEE Network*, pages 8–18, September 1993.