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

Zhi-Li Zhang, *Senior Member, IEEE*, Zhenhai Duan, and Yiwei Thomas Hou, *Senior Member, IEEE*

***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 queuing algorithms to support guaranteed services.**

***Index Terms*—Core stateless, differentiated services, guaranteed services, QoS, scheduling.**

I. 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], [11], [16], [19], [20] and references therein), such as virtual clock (VC) and weighted fair queuing (WFQ), have been proposed for the support of *QoS guarantees.* For example, it has been shown [8], [12] 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 IETF has defined a *guaranteed service* [14] 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* 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], [21].

Performing per-flow management inside the network, however, raises the important issue of *scalability.* Due to the complexity 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 IETF [1], [2]. By processing packets based on a number of prespecified *per-hop behaviors* (PHBs) encoded by bit patterns carried inside a packet header, the DiffServ paradigm greatly simplifies the data plane of the network core of a domain, thereby making it more scalable. (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 intradomain QoS provisioning and interdomain service agreement. These *control plane* functions can be performed, for example, by employing a *bandwidth broker* architecture [10]. 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 [18], Stoica and Zhang, using the DiffServ paradigm and the novel notion of *dynamic packet state,* developed several techniques to support end-to-end *per-flow* delay guarantees *without per-flow QoS management.* In the data plane, they designed a new (non-work-conserving) scheduling algorithm, called *Core Jitter VirtualClock* 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.

Manuscript received October 15, 1999; revised April 15, 2000. This work was supported in part by the National Science Foundation under the CAREER Award NCR-9734428. Any opinions, findings, and conclusions or recommendations in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Z.-L. Zhang and Z. Duan are with the Department of Computer Science and Engineering, University of Minnesota, Twin Cities, Minneapolis, MN 55455 USA (e-mail: zhzhang@cs.umn.edu; duan@cs.umn.edu).

Y. T. Hou is with the Fujitsu Labs of America, Sunnyvale, CA 94085 USA (e-mail: thou@fla.fujitsu.com).

Publisher Item Identifier S 0733-8716(00)09236-2.

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 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 queuing algorithms to support guaranteed services.

The objectives of the proposed virtual time reference system are twofold. 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. In this paper, however, we will primarily focus on the theoretical underpinning of the proposed virtual time reference system. Issues related to the control plane management, e.g., bandwidth broker design, will be addressed in the future work (see, e.g., the initial work reported in [23]).

The remainder 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 III 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 IV 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 V, we design new core stateless scheduling algorithms using packet virtual time stamps. We also demonstrate that existing scheduling algorithms can be accommodated in our framework. Related work is discussed in Section VI, and the paper is concluded in Section VII.

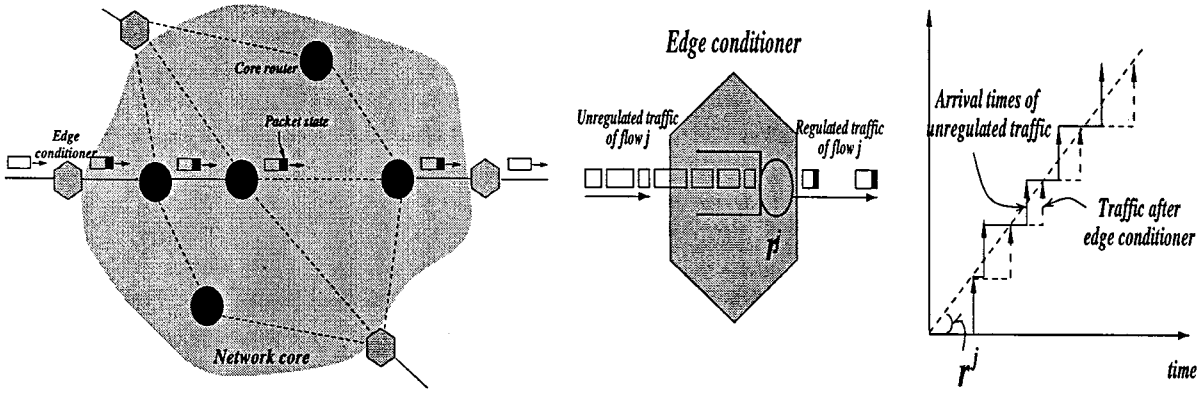
II. Virtual Time Reference System: Basic Architecture

In this section we outline the basic architecture of the proposed unifying scheduling framework—the *virtual time reference system* (VTRS). Conceptually, the virtual time reference system consists of three logical components (see Figs. 1 and 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 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.*

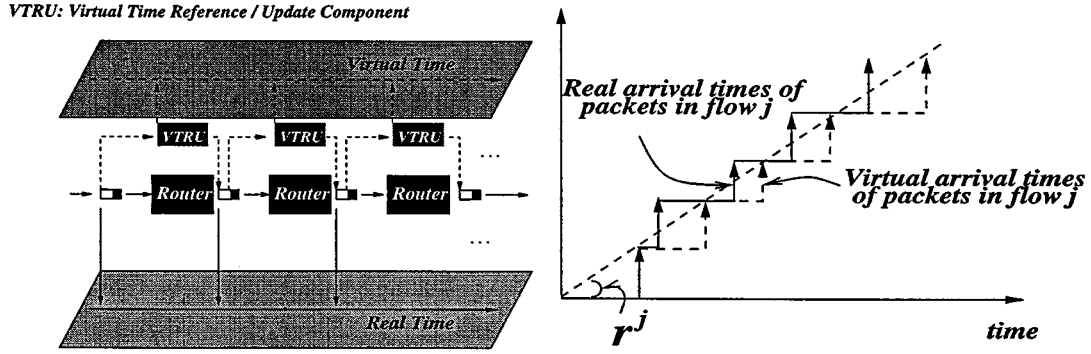
Edge traffic conditioning plays a key role in the virtual time reference system, as *it ensures that traffic of a flow will never be injected into the network core at a rate exceeding its reserved rate.* This traffic conditioning is done by using a traffic shaper [or, more specifically, a rate spacer, see Fig. 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

[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) (b)

Fig. 1. Edge conditioning in the virtual time reference system. (a) A conceptual network model. (b) Edge conditioner and its effect.



(a) (b)

Fig. 2. Illustration of the virtual time reference system. (a) Virtual time reference/update mechanism. (b) Virtual traffic shaping.

hand side of Fig. 1(b). Formally, for a flow j with a reserved rate r^j, the interarrival 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) to (1), we see that *with respect to the virtual time,* the interarrival time spacing is preserved at a core router. Or to put it another way, *the "virtual rate" (as defined with respect to the virtual time) of packets of a flow arriving at a core router does not exceed the reserved rate of the flow.* Clearly, with respect to the *real* arrival times of the packets at a core router, this statement in general does not hold [see Fig. 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 Fig. 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 IV 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

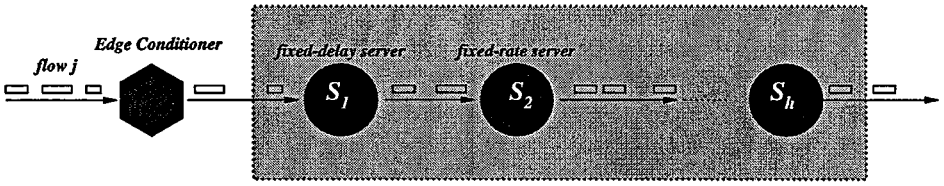


Fig. 3. An ideal per-flow system.

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 [17] does for various fair queuing 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.

TABLE I
NOTATION USED IN THE PAPER

General Notation	
$p^{j,k}$	the kth packet of 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 one server
r^j	reserved rate of flow j
d^j	delay parameter of flow j
h	number of hops along the path of flow j
q	number of 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 $p^{j,k}$: $\delta^{j,k} = \Delta_q^{j,k}/q$
$\tilde{d}_i^{j,k}$	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

III. 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.

Fig. 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 $k = 1, 2, \cdots$, 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 q fixed-rate servers and $h - q$ fixed-delay servers.

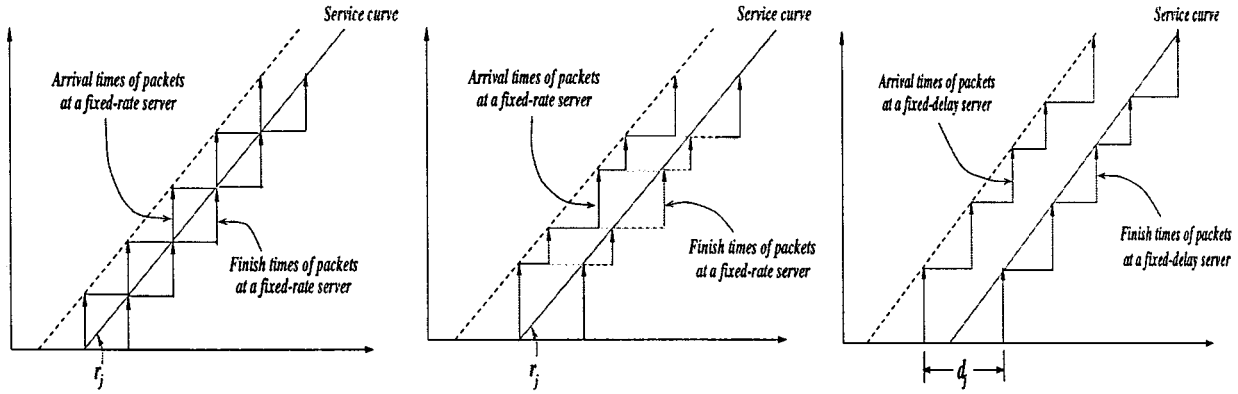
Before we introduce the notion of packet virtual time stamps, we first need to understand and quantify the end-to-end delay experienced by the packets in the ideal per-flow system. We embark on this task in Section III-A. Based on the results obtained thereof, in Section III-B we introduce the ideal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

A. 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,

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

[2]Note that in order to model a nonpreemptive, noncut-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 to have departed the server *only* when its last bit has been serviced.



(a) (b) (c)

Fig. 4. Delay experienced by packets at a server in the ideal per-flow system. (a) Fixed-rate server with constant-size packets. (b) Fixed-rate server with variable-size packets. (c) Fixed-delay server.

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., $q = h$. This result can then be extended to the general ideal per-flow system with mixed fixed-rate and fixed-delay servers.

For $i = 1, 2, \cdots, h$, 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 relations among $a_i^{j,k}$'s and $f_i^{j,k}$'s hold. For any $k = 1, 2, \cdots$,

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

and

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

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 Fig. 4(a)]. In other words, for $i = 1, 2, \cdots, h$, $a_i^{j,k} \ge f_i^{j,k-1}$ and $f_i^{j,k} = a_i^{j,k} + L^j/r^j$. Therefore, in this case, we have $f_h^{j,k} = a_1^{j,k} + hL^j/r^j$. Thus, the *end-to-end delay*, $f_h^{j,k} - a_1^{j,k}$, of packet $p^{j,k}$ in the ideal per-flow system 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 Fig. 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 other subsequent packets to be delayed.

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

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

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 stated in the following theorem, the proof of which can be found in [22].

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

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

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 III-B, this fact is crucial in providing a *core stateless* definition of packet virtual time stamps for a system involving fixed-rate servers with variable packet sizes. From Theorem 1, we have the following two important corollaries, whose proofs are also given in [22].

Corollary 2: For any $q \le h$ and $k = 1, 2, \cdots$, we have

$$\frac{\Delta_q^{j,k}}{q} \le \frac{\Delta_h^{j,k}}{h}. \tag{9}$$

Corollary 3: For any $k \ge 1$, and $i = 1, 2, \cdots, h$, we have

$$\Delta_i^{j,k} + i\,\frac{L^{j,k}}{r^j} \le i\,\frac{L^{j,\max}}{r^j} \tag{10}$$

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

We now consider the general ideal per-flow system with both fixed-rate and fixed-delay servers. Recall that we assume we have q fixed-rate servers and $h-q$ 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 $k = 1, 2, \cdots$,

$$a_i^{j,k} = f_{i-1}^{j,k} \quad \text{and} \quad f_i^{j,k} = a_i^{j,k} + d^j. \tag{11}$$

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 queuing delay due to the packet size difference [see Fig. 4(c)]. It is easy to see that we can rearrange 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 $n-q$ servers are the fixed delay servers. Then from (7), we have

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

Therefore, the end-to-end delay of packet $p^{j,k}$ in the ideal per-flow system is $f_h^{j,k} - a_1^{j,k} = \Delta_q^{j,k} + q(L^{j,k}/r^j) + (h-q)d^j$. In particular, from Corollary 3, we have $\Delta_q^{j,k} + qL^{j,k}/r^j \le qL^{j,\max}/r^j$. Thus, for $k = 1, 2, \cdots$,

$$f_h^{j,k} - a_1^{j,k} \le q\,\frac{L^{j,\max}}{r^j} + (h-q)\,d^j. \tag{12}$$

B. 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 system is referred to as the *ideal per-flow virtual time reference system.*

For $i = 1, 2, \cdots, h$, 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 $k = 1, 2, \cdots$,

$$\tilde{\omega}_i^{j,k+1} - \tilde{\omega}_i^{j,k} \ge \frac{L^{j,k+1}}{r^j}. \tag{13}$$

Reality Check: $\tilde{\omega}_i^{j,k} \ge a_i^{j,k}$, where $a_i^{j,k}$ 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 [22])

$$\tilde{A}^j(\tau, t) \le r^j(t-\tau) + L^{j,\max}. \tag{14}$$

This bound is analogous to the traffic envelope (4) at the network edge, except 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 property* 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. 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 Fig. 3, where there are q fixed-rate servers and $h-q$ fixed-delay servers in the ideal per-flow system. For each packet $p^{j,k}$, define $\delta^{j,k} = \Delta_q^{j,k}/q$. 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 $i = 1, 2, \cdots, h$, 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:

$$\tilde{d}_i^{j,k} = \begin{cases} L^{j,k}/r^j + \delta^{j,k} & \text{if } \mathcal{S}_i \text{ is a fixed rate server} \\ d^j & \text{if } \mathcal{S}_i \text{ a fixed delay server.} \end{cases} \tag{15}$$

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 $i = 2, \cdots, h$, 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:

$$\tilde{\omega}_i^{j,k} = \tilde{\nu}_{i-1}^{j,k} = \tilde{\omega}_{i-1}^{j,k} + \tilde{d}_{i-1}^{j,k}$$

and

$$\tilde{\nu}_i^{j,k} = \tilde{\omega}_i^{j,k} + \tilde{d}_i^{j,k}. \tag{16}$$

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.

Theorem 4: For $i = 1, 2, \cdots, h$, the virtual spacing property (13) holds at each server $\mathcal{S}_i$. Furthermore,

$$\tilde{\omega}_i^{j,k} \geq a_i^{j,k} \tag{17}$$

and, in particular,

$$\tilde{\nu}_h^{j,k} = f_h^{j,k}. \tag{18}$$

Proof: We first establish that the virtual spacing property holds. Fix i and let q_i be the number of fixed-rate servers along the path from the first hop to the $(i-1)$th hop. Clearly $q_i \leq q$. Note that from (16) and (15), we have

$$\begin{aligned}\tilde{\omega}_i^{j,k} &= \tilde{\omega}_1^{j,k} + \sum_{q=1}^{i-1} \tilde{d}_q^{j,k} \\ &= a_1^{j,k} + q_i\left(\delta^{j,k} + \frac{L^{j,k}}{r^j}\right) + (i-1-q_i)\,d^j. \end{aligned} \tag{19}$$

Hence, to prove (13), it suffices to show

$$\begin{aligned} &a_1^{j,k} + q_i\left(\delta^{j,k} + \frac{L^{j,k}}{r^j}\right) \\ &\quad \geq a_1^{j,k-1} + q_i\left(\delta^{j,k-1} + \frac{L^{j,k-1}}{r^j}\right) + \frac{L^{j,k}}{r^j} \end{aligned}$$

or, equivalently,

$$\delta^{j,k} \geq \delta^{j,k-1} + \frac{L^{j,k-1} - L^{j,k}}{r^j} + \frac{a_1^{j,k-1} - a_1^{j,k} + \frac{L^{j,k}}{r^j}}{q_i}. \tag{20}$$

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

$$\begin{aligned}\delta^{j,k} &= \frac{\Delta_q^{j,k}}{q} \\ &\geq \frac{\Delta_q^{j,k-1}}{q} + \frac{L^{j,k-1} - L^{j,k}}{r^j} + \frac{a_1^{j,k-1} - a_1^{j,k} + \frac{L^{j,k}}{r^j}}{q}. \end{aligned} \tag{21}$$

Using (3) and the fact that $q_i \leq q$, we see that the last term in the right-hand side of (21) is larger than the corresponding term in (20). Hence, (20) holds.

We now establish (17). As $\tilde{\omega}_1^{j,k} = a_1^{j,k}$, (17) holds trivially for $i = 1$. To show that (17) also hold for $i = 2, \cdots, h$, observe that

$$a_i^{j,k} = f_{i-1}^{j,k} = a_1^{j,k} + \Delta_{q_i}^{j,k} + q_i\,\frac{L^{j,k}}{r^j} + (i-1-q_i)\,d^j \tag{22}$$

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

Comparing (22) and (19) and using the fact that $\Delta_q^{j,k}/q \geq \Delta_{q_i}^{j,k}/q_i$ (see Corollary 2), we see that (17) indeed holds.

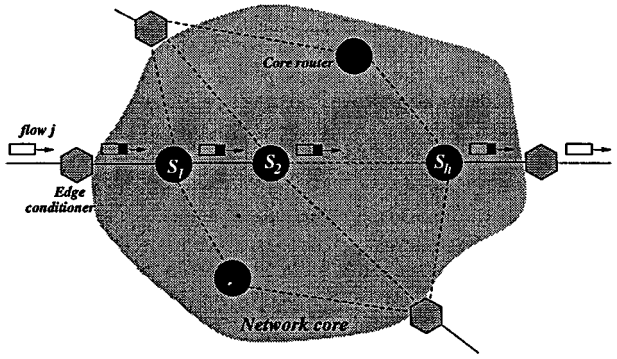


Fig. 5. A flow traverses a network core.

Last, (18) follows easily from the definitions, as

$$\tilde{\nu}_h^{j,k} = a_1^{j,k} + \Delta_q^{j,k} + q\,\frac{L^{j,k}}{r^j} + (h-q)d^j = f_h^{j,k}. \tag{23}$$

■

IV. Virtual Time Reference System and Packet Scheduling

In this section we extend the virtual time reference system defined in the context of the 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 Fig. 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 $k = 1, 2, \cdots$,

$$\hat{a}_1^{j,k+1} - \hat{a}_1^{j,k} \geq \frac{L^{j,k+1}}{r^j} \tag{24}$$

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 Fig. 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 $i = 1, 2, \cdots, h$, 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.

A. Scheduling Blackbox: Per-Hop Behavior Characterization

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 (15). 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 i, $i = 1, 2, \cdots, h$, 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* property holds at $\mathcal{S}_i$, namely,

$$\hat{a}_i^{j,k} \leq \tilde{\omega}_i^{j,k} \tag{25}$$

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 $\mathcal{S}_i$, 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,

$$\hat{f}_i^{j,k} \leq \tilde{\nu}_i^{j,k} + \Psi_i. \tag{26}$$

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 (26) 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 (26) holds.

B. 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 Fig. 5. Suppose there are q rate-based scheduling blackboxes and $h - q$ delay-based scheduling blackboxes. For $i = 1, 2, \cdots, h$, 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:

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

For $i = 1, 2, \cdots, h$, 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—the *virtual spacing, reality check, bounded delay,* and *core stateless* properties. Here in defining the reality check and bounded delay properties, the quantities $a_i^{j,k}$ and $f_i^{j,k}$ defined in Section III-B 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.,

$$\tilde{\omega}_1^{j,k} = \hat{a}_1^{j,k}. \tag{27}$$

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 (24), 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 $i = 1, 2, \cdots, h-1$, let $\pi_{i,i+1}$ denote the *propagation delay*[3] from the ith 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

$$\hat{a}_{i+1}^{j,k} \leq \tilde{\nu}_i^{j,k} + \Psi_i + \pi_{i,i+1}. \tag{28}$$

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 (28) 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:

$$\begin{aligned} \tilde{\omega}_{i+1}^{j,k} &= \tilde{\nu}_i^{j,k} + \Psi_i + \pi_{i,i+1} \\ &= \tilde{\omega}_i^{j,k} + \tilde{d}_i^{j,k} + \Psi_i + \pi_{i,i+1}. \end{aligned} \tag{29}$$

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 (29) 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

[3]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 $p^{j,k}$, $\hat{a}_{i+1}^{j,k} \leq \hat{f}_i^{j,k} + \pi_{i,i+1}$.

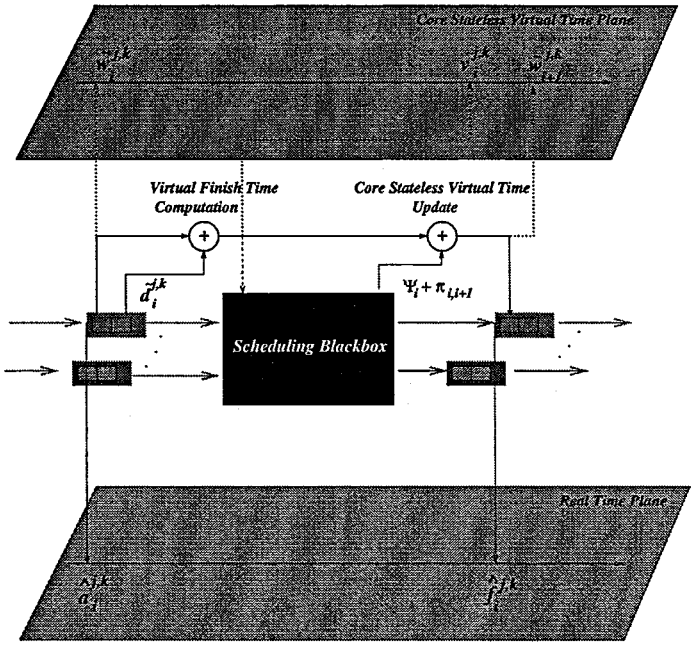


Fig. 6. Virtual time reference system: per-hop behavior and operations.

flow's path). Thus, *no per-flow state needs to bemaintained 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 (29) and that in (16), it is easy to see that the virtual spacing property also holds at each core router $\mathcal{S}_i$. Furthermore, we have

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

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

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

This completes the construction of packet virtual time stamps for flow j. In summary, packet virtual time stamps are initialized using (27) at the network edge, and are referenced and updated using (29) at each core router. The reference and update mechanism of the resulting virtual time reference system is schematically shown in Fig. 6.

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:

$$\begin{aligned}&\hat{f}_h^{j,k} - \hat{a}_1^{j,k} \\ &\quad \le \Delta_q^{j,k} + q\frac{L^{j,k}}{r^j} + (h-q)\,d^j + \sum_{i=1}^{h}\Psi_i + \sum_{i=1}^{h-1}\pi_{i,i+1} \\ &\quad \le q\,\frac{L^{j,\max}}{r^j} + (h-q)d^j + \sum_{i=1}^{h}\Psi_i + \sum_{i=1}^{h-1}\pi_{i,i+1} \qquad (30)\end{aligned}$$

where the last inequality follows from Corollary 3.

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

$$\hat{f}_h^{j,k} - \hat{a}_1^{j,k} \le h\,\frac{L^{j,\max}}{r^j} + \sum_{i=1}^{h}\Psi_i + \sum_{i=1}^{h-1}\pi_{i,i+1}. \qquad (31)$$

This bound is analogous to those derived for fair-queuing/latency-rate-server based scheduling algorithms [8], [12], [17]. In particular, if $\Psi_i = L^{*,\max}/C_i$, where $L^{*,\max}$ is the maximum packet size permissible at the ith 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 queuing (WFQ) schedulers [12] or virtual clock (VC) schedulers [8].

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 $D^j \ge qL^{j,\max}/r^j$. Set

$$d^j = \frac{1}{h-q}\left[D^j - q\,\frac{L^{j,\max}}{r^j}\right]. \qquad (32)$$

Suppose we can design delay-based scheduling algorithms that can support the delay parameter d^j for flow j. Then from (30) we have

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

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

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

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 $D^j \ge 0$ 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 to the WFQ-based reference system used in the Internet IntServ model [3]. From (30), 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 $q(L^{j,\max}/r^j)$ 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.

V. Design and Characterization of Packet Schedulers Using VTRS

In this section we first design two new *core stateless* scheduling algorithms—one rate-based and one delay-based packet scheduler—using the notion of packet virtual time stamps. We then illustrate through examples how the per-hop behavior of existing (*static* and *stateful*) scheduling algorithms can also be characterized using the virtual time reference system. These examples demonstrate the ability of the virtual time reference system as a unifying scheduling framework for support of guaranteed services. For more examples, as well as proofs of the theorems included in this section, we refer interested readers to [22].

A. $C_{\not s}$VC: A Rate-Based Core Stateless Scheduling Algorithm

A *core stateless virtual clock* ($C_{\not s}$VC) scheduler $\mathcal{S}$ is a work-conserving, rate-based scheduler. It services packets in the order of their virtual finish times. 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 $\tilde{d}^{j,k} = (L^{j,k}/r^j) + \delta^{j,k}$ 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 $\Psi_{C_{\not s}\mathrm{VC}} = L^{*,\max}/C$, provided that an appropriate schedulability condition is met.

Theorem 5: Consider N flows traversing a $C_{\not s}$VC scheduler $\mathcal{S}$ such that the schedulability condition $\sum_{j=1}^{N} r^j \leq C$ is satisfied. Suppose that $\hat{a}^{j,k} \leq \tilde{\omega}^{j,k}$ for any packet $p^{j,k}$ of flow j, $j = 1, 2, \cdots, N$. Then

$$\hat{f}^{j,k} \leq \tilde{\nu}^{j,k} + \frac{L^{*,\max}}{C}. \tag{33}$$

In other words, $\Psi_{C_{\not s}\mathrm{VC}} = L^{*,\max}/C$.

B. VT-EDF: A Delay-Based Core Stateless Scheduling Algorithm

A *virtual time earliest deadline first* (VT-EDF) scheduler $\mathcal{S}$ is a work-conserving, delay-based scheduler. It services packets in the order of their virtual finish times. 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 $\Psi_{\text{VT-EDF}} = L^{*,\max}/C$.

Theorem 6: Consider N flows traversing a VT-EDF scheduler $\mathcal{S}$, where d^j is the delay parameter associated with flow j, $1 \leq j \leq N$. Without loss of generality, assume $0 \leq d^1 \leq d^2 \leq \cdots \leq d^N$. Suppose the following *schedulability condition* holds:

$$\sum_{j=1}^{N} [r^j(t - d^j) + L^{j,\max}]\mathbf{1}_{\{t \geq d^j\}} \leq Ct, \qquad \text{for any } t \geq 0 \tag{34}$$

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

$$\hat{f}^{j,k} \leq \tilde{\nu}^{j,k} + \frac{L^{*,\max}}{C}. \tag{35}$$

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

C. Static Scheduling Algorithms

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 static priority-based scheduling schemes. 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 use the FIFO scheduling algorithm as an example to illustrate how static scheduling algorithms can be accommodated into our framework.

An FIFO packet scheduler $\mathcal{S}$ services packets in the order of their actual arrival times, regardless of their virtual times. 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 preconfigured 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. 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, $\Psi_{\text{FIFO}} = B/C + L^{*,\max}/C$.

D. Stateful Scheduling Algorithms

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. 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 [15] and [17], namely, the *latency-rateservers.* This class encompasses virtually all known fair-queuing algorithms and its variations. The following theorem establishes the relationship between the latency for a latency-rate server and its error term in the VTRS framework.

TABLE II
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

$$\Psi \leq \Theta^j. \quad \blacksquare$$

For a number of well-known scheduling algorithms studied in [15] and [17], we can show that $\Psi = \Theta^j - (L^{j,\max}/r^j)$. Θ^j and its corresponding error term for these scheduling algorithms are listed in Table II.

VI. RELATED WORK

The idea of virtual time has been used extensively in the design of packet scheduling algorithms (see, e.g., [7], [11], [12], [15], [17], [19], and [20]). The notion of virtual time defined in these contexts is used to emulate an ideal scheduling system and is defined *local* to each scheduler. Computation of the virtual time function requires *per-flow* information 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 [18], 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. (Since [18], a couple of new core stateless scheduling algorithms (see, e.g., [9]) have been designed, in addition to ours.) Our work is primarily motivated and inspired by the results in [18]. However, our work differs from [18] 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 may greatly simplify the design and management of QoS provisioning architectures. 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.

VII. CONCLUSION 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 the 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 queuing 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 (see the initial work reported in [23]). We are also exploring various issues regarding the implementation of the virtual time reference system and its implications in QoS provisioning. In particular, we are interested in applying the VTRS framework to the design of coarse-grain QoS control mechanisms under the current DiffServ architecture as well as the multiprotocol label switching (MPLS) architecture [5], [13].

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, work in progress, Feb. 1999.
[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," RFC 2475, Dec. 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, Sept. 1997.
[5] R. Callon, P. Doolan, N. Feldman, A. Fredette, G. Swallow, and A. Viswanathan, "A framework for multiprotocol label switching," Internet Draft, work in progress, Sept. 1999.
[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*, Aug. 1992.
[7] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queuing algorithm," in *Proc. ACM SIGCOMM*, Austin, TX, Sept. 1989, pp. 1–12.
[8] N. Figueira and J. Pasquale, "An upper bound on delay for the virtual clock service discipline," *IEEE/ACM Trans. Networking*, vol. 3, pp. 399–408, Aug. 1995.
[9] T. Nandagopal, N. Venkitaraman, R. Sivakumar, and B. Vaduvur, "Relative delay differentiation and delay class adaptation in core-stateless networks," in *Proc. IEEE INFOCOM*, Mar. 2000.
[10] K. Nichols, V. Jacobson, and L. Zhang, "A Two-bit differentiated services architecture for the Internet," RFC 2638, July 1999.

[11] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks—The single node case," *IEEE/ACM Trans. Networking*, vol. 1, pp. 344–357, 1993.
[12] ——, "A generalized processor sharing approach to flow control in integrated services networks—The multiple node case," *IEEE/ACM Trans. Networking*, vol. 2, pp. 137–150, 1994.
[13] E. C. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," Internet Draft, work in progress, Aug. 1999.
[14] S. Shenker, C. Partridge, and R. Guérin, "Specification of guaranteed quality of service," RFC 2212, Sept. 1997.
[15] D. Stiliadis, "Traffic scheduling in packet-switched networks: Analysis, design, and implementation," Ph.D. thesis, Computer Science Engineering Dep., Univ. Calif. Santa Cruz, June 1996.
[16] D. Stiliadis and A. Varma, "Efficient fair queuing algorithms for packet-switched networks," *IEEE/ACM Trans. Networking*, vol. 6, pp. 175–185, 1998.
[17] ——, "Latency-rate servers: A general model for analysis of traffic scheduling algorithms," *IEEE/ACM Trans. Networking*, vol. 6, pp. 611–624, 1998.
[18] I. Stoica and H. Zhang, "Providing guaranteed services without per flow management," in *Proc. ACM SIGCOMM*, Boston, MA, Sept. 1999.
[19] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proc. IEEE*, Oct. 1995.
[20] L. Zhang, "Virtual clock: A new traffic control algorithm for packet switching networks," in *Proc. ACM SIGCOMM*, Sept. 1990, pp. 19–29.
[21] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A new resource reservation protocol," *IEEE Network*, pp. 8–18, Sept. 1993.
[22] Z.-L. Zhang, Z. Duan, and Y. T. Hou. (1999, Sept.) Virtual time reference system: A unifying scheduling framework for scalable support of guaranteed services. Tech. Rep. 00-027, Computer Science Dep., Univ. Minnesota. [Online]. Available: http://www.cs.umn.edu/~zhzhang/papers.html
[23] Z.-L. Zhang, Z. Duan, Y. T. Hou, and L. Gao. (2000, Mar.) Decoupling QoS control from core routers: A novel bandwidth broker architecture for scalable support of guaranteed services. Tech. Rep. 00-028, Computer Science Dep., Univ. Minnesota. [Online]. Available: http://www.cs.umn.edu/~zhzhang/papers.html

Zhi-Li Zhang (M'97–SM'96) received the B.S. degree in computer science from Nanjing University, China, and the M.S. and Ph.D. degrees in computer science from the University of Massachusetts, Amherst, in 1992 and 1997, respectively.

In January 1997, he joined the Computer Science faculty at the University of Minnesota, where he is currently an Assistant Professor. From 1987 to 1990, he conducted research in the Computer Science Department at Århus University, Denmark, under a fellowship from the Chinese National Commission on Education. His current research interests include high speed networks, multimedia and real-time systems, and modeling and performance evaluation of computer and communication system.

Dr. Zhang received the National Science Foundation CAREER Award in 1997. He is also a co-recipient of a 1996 ACM Sigmetrics Conference best paper award. He is a member of ACM, IEEE, and INFORMS.

Zhenhai Duan received the B.S. degree from Shandong University, China, and the M.S. degree from Peking University, China, in 1994 and 1997, respectively, both in computer science.

Currently he is pursuing the Ph.D. degree in the Department of Computer Science and Engineering, the University of Minnesota. His research interests are in the areas of computer and communications networks, currently in QoS provisioning over Internet, traffic measurements and modeling.

Mr. Duan is a Student Member of ACM.

Yiwei Thomas Hou (M'98–SM'91) received the B.E. degree (*summa cum laude*) from the City College of New York in 1991, the M.S. degree from Columbia University in 1993, and the Ph.D. degree from Polytechnic University, Brooklyn, NY, in 1997, all in electrical engineering.

While a graduate student, he worked at AT&T Bell Labs, Murray Hill, NJ, during the summers of 1994 and 1995, on internetworking of IP and ATM networks; he conducted research at Bell Labs, Lucent Technologies, Holmdel, NJ, during the summer of 1996, on fundamental problems on network traffic management. Since September 1997, he has been a Research Scientist at Fujitsu Laboratories of America, Sunnyvale, CA. His current research interests are in the areas of scalable architecture, protocols, and implementations for differentiated services Internet; terabit switching architecture; and quality of service (QoS) support for multimedia over IP networks.

Dr. Hou is a Member of ACM, Sigma Xi, and New York Academy of Sciences. He was awarded a five-year National Science Foundation Graduate Research Traineeship for pursuing the Ph.D. degree in high speed networking, and was a recipient of the Alexander Hessel award for outstanding Ph.D. dissertation (1997–1998 academic year) from Polytechnic University. He received Intellectual Property Contribution Award from Fujitsu Laboratories of America in 1999.