SCED: A Generalized Scheduling Policy for Guaranteeing Quality-of-Service

Hanrijanto Sariowan, *Member, IEEE,* Rene L. Cruz, *Senior Member, IEEE,* and George C. Polyzos, *Member, IEEE*

***Abstract*—In this paper, we introduce a new scheduling policy which provides guaranteed service for a session based on a flexible service specification called the *Service Curve*. This policy, referred to as the Service Curve based Earliest Deadline first policy (SCED), is a generalized policy to which well-known policies such as VirtualClock and the Earliest Deadline First (EDF) can be mapped as special cases, by appropriate specification of the service curves. Rather than characterizing service by a single number, such as minimum bandwidth or maximum delay, service curves provide a wide spectrum of service characterization by specifying the service using a *function*. The flexibility in service specification allows a user, or the network, to specify a service that best *matches* the quality-of-service required by the user, preventing an over-allocation of network resources to the user. For a single server, we show that the SCED policy is optimal in the sense of supporting the largest possible schedulability region, given a set of delay-bound requirements and traffic burstiness specifications. For the case of a network of servers, we show that the SCED policy has a greater capability to support end-to-end delay-bound requirements than other known scheduling policies. The key to this capability is the ability of SCED to allocate and guarantee service curves with arbitrary shapes.**

***Index Terms*—Integrated services networks, multiplexing, network calculus, quality-of-service guarantees, scheduling, service curves, traffic envelopes.**

I. Introduction

BROADBAND packet-switched networks are expected to support a large number of concurrent sessions which may have significantly different traffic characteristics and quality-of-service (QoS) requirements. It is desired that such networks be able to deliver a diverse set of QoS guarantees while achieving high bandwidth utilization. In order to meet such objectives, the networks have to implement a scheduling policy, determining which packets to serve at any given time from the various sessions the network serves, as well as an admission control policy, determining and enforcing the maximum set of simultaneous sessions that the network can support.

In this paper, we propose a new scheduling policy which provides guaranteed service for a session based on a flexible service specification called the *Service Curve* [6], [7]. This policy, referred to as the Service Curve based Earliest Deadline first policy (SCED), is a generalized policy to which well-known policies such as VirtualClock and the Earliest Deadline First (EDF) can be mapped as special cases. SCED provides the network with a flexible means for allocating network resources to various sessions in order to meet their diverse QoS requirements while maximizing network utilization.

Most real-time traffic users require the network to provide guarantees on the maximum delay that the traffic will experience as it travels across the network. However, many resource-allocation policies proposed for real-time traffic are based on guaranteeing bandwidth to a specific user. Examples of such policies are VirtualClock [27], Stop-and-Go Queueing [15], Packetized Generalized Processor Sharing (PGPS-RPPS) [20], and Weighted Fair Queueing (WFQ) [9]. These policies guarantee that, within some predefined time intervals, the amount of a user's traffic which is transported by the network is no less than a specified lower bound, which is equal to the guaranteed bandwidth for the user times the length of the interval. While guarantees on bandwidth imply guarantees on delay for the user, provided that the user limits the traffic it sends to the network, this approach can result in over-allocation of network resources, especially if the traffic is very bursty. Scheduling policies such as Rate Controlled Scheduling (RCS) and EDF [26], [13], [14], [19] can overcome these problems in the case of scheduling for a single server. As we shall see, the SCED policy is a generalization of all of these policies, and can in fact schedule sessions even more efficiently in the case of a *network* of servers.

The SCED policy is based on *Service Curves,* which serve as a general measure for characterizing service provided to a user. Rather than characterizing service by a single number, such as minimum bandwidth or maximum delay, service curves provide a wide spectrum of service characterization by specifying the service using a *function.* The flexibility in service specification allows a user, or the network, to specify a service that best *matches* the QoS required by the user, preventing an over-allocation of network resources to the user. For a single server, we show that the SCED policy is optimal in the sense of supporting the largest possible schedulability region, given a set of delay-bound requirements and traffic-burstiness specifications. For the case of a network of servers,

Manuscript received July 21, 1997; revised August 12, 1998 and May 18, 1999; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor R. Guérin. This work was supported in part by the National Science Foundation under Grant NCR 91-58618, Grant NCR 91-19473, Grant NCR 94-15684, and Grant NCR 97-25568, and by the Center for Wireless Communications at the University of California at San Diego.

H. Sariowan is with Tiernan Communications, San Diego, CA 92111 USA (e-mail: sariowan@ucsd.edu).

R. L. Cruz is with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093-0407 USA (e-mail: rcruz@ucsd.edu).

G. C. Polyzos is with the Computer Systems Laboratory, Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA 92093-0114 USA (e-mail: polyzos@cs.ucsd.edu).

Publisher Item Identifier S 1063-6692(99)08526-X.

we show that the SCED policy has a greater capability to support end-to-end delay-bound requirements than other known scheduling policies. The key to this capability is the ability of SCED to allocate and guarantee service curves with arbitrary shapes.

The remainder of the paper is organized as follows. In Section II, we describe the concept of service curves and present results which are needed for the following sections. The SCED policy is defined and analyzed in Section III. In Section IV, we compare the SCED policy to other scheduling algorithms, and demonstrate the unique capabilities of SCED through an example. Finally, in Section V we conclude with a brief discussion.

II. Service Curves: A Review

We would like to succinctly characterize the service the session receives from the network element, so that bounds on the backlog and delay can be obtained. We shall characterize this service in terms of a service curve, defined in the next sub-section.

A. Service Curves, Delay, and Buffering Requirements

Consider a network element which receives, possibly buffers, and eventually serves (sends) packets from a session.[1] Note that a network element can be a single server (switch), or even a full subnet. We adopt a discrete time model in this paper. We assume that time is divided into fixed intervals called *slots*, numbered 0, 1, 2, $\cdots$, and that packets have fixed size. We shall assume that the transmission time of any packet is exactly one slot, i.e. it takes exactly one slot for a packet to enter or exit a network element. Without loss of generality, we assume a "cut-through" model, whereby a packet arriving to a network element during a slot may depart the network element during the same slot. Let $R^{\text{in}}[t]$, where t is a nonnegative integer, denote the number of packets from the session which arrive at the network element during slot t. Similarly, $R^{\text{out}}[t]$ denotes the number of packets from the session which depart from the network element during slot t. There may be several sessions which pass through a network element, but in this section, we focus on a single session. Also, in this paper we assume that only an integral number of packets can arrive or depart from a network element in any slot. Thus, $R^{\text{in}}[t]$ and $R^{\text{out}}[t]$ take on only nonnegative integer values. For $s \leq t$, the "interval" $[s,t]$ is defined to be the set of slots s, $s+1$, $\cdots$, t. Define $R^{\text{in}}[s,t]$ to be the number of packets from the session arriving at the network element during the interval $[s,t]$, i.e., $R^{\text{in}}[s,t] = \Sigma_{m=s}^{t} R^{\text{in}}[m]$. If $s > t$, define $R^{\text{in}}[s,t] = 0$. Let $r^{\text{in}}(t) = R^{\text{in}}[1,t] = \Sigma_{m=1}^{t} R^{\text{in}}[m]$. Similarly, define $R^{\text{out}}[s,t]$ to be the number of packets from the session leaving the network element during the interval $[s,t]$, and $r^{\text{out}}(t) = R^{\text{out}}[1,t]$.

Assuming that there is no packet stored in the network element at the end of slot zero, the number of packets from the session which are stored in the network element at the end of slot t, called the *backlog* $B[t]$ of the session at the end of slot t, is given by

$$B[t] = r^{\text{in}}(t) - r^{\text{out}}(t) \geq 0. \tag{1}$$

The *virtual delay* $d[t]$ suffered by the session through the network element, relative to time t, is defined to be

$$d[t] = \min\{\Delta\colon\ \Delta \geq 0 \text{ and } r^{\text{in}}(t) \leq r^{\text{out}}(t+\Delta)\}. \tag{2}$$

If packets from the session depart the server in the same order in which they arrive (first-in first-out), then the virtual delay $d[t]$ is an upper bound of the delay suffered by any packet from the session that arrives in slot t.

We are now ready to define a service-curve guarantee.

Definition II.1— (Service Curve): Given a nonnegative nondecreasing function $S(\cdot)$, where $S(0) = 0$, we say that the network element guarantees service curve $S(\cdot)$ for the session if for any t, there exists $s \leq t$ such that $r^{\text{out}}(t) - r^{\text{in}}(s) \geq S(t-s)$.

We note that a service curve S only needs to be defined on the nonnegative integers, even though we shall often define a service curve S on the set of all real numbers. Furthermore, in general, we shall allow a service curve to be real-valued.

The notion of a service curve has its roots in the work of Parekh and Gallager [20], who introduced the concept of a universal service curve in the context of a specific scheduling algorithm. The service-curve definition above is slightly less restrictive than the one proposed in [6], [7], which required that the backlog of the session be equal to zero at the end of slot s. Both the definition in this paper and in [6], [7] are considerably less restrictive service definitions than that of the universal service curve in [20], and apply generally to scheduling algorithms other than that considered in [20]. Less general or more restrictive service definitions have also been reported in [5], [16], [17], [23]. The service-curve definition in this paper was reported earlier in [22], and also independently proposed in [1] and [18].

Given two functions f and g defined on the nonnegative integers, define the *convolution* of f and g, $f * g$, to be the function defined on the nonnegative integers such that

$$(f*g)(t) = \min_{0 \leq s \leq t} \{f(s) + g(t-s)\}.$$

It is easy to verify that the convolution operator is commutative and associative, i.e., $f*g = g*f$, and $(f*g)*h = f*(g*h)$.

With this definition, it is clear that a network element guarantees service curve S to a session if and only if[2] $r^{\text{out}} \geq r^{\text{in}} * S$.

As an example, suppose a network element guarantees the service curve $S = \delta_d$, where

$$\delta_d(x) = \begin{cases} 0, & \text{if } x \leq d \\ \infty, & \text{if } x > d. \end{cases} \tag{3}$$

In this case, it follows that for all t we have $r^{\text{out}}(t) \geq (r^{\text{in}} * \delta_d)(t) = r^{\text{in}}(t-d)$, and hence the delay is bounded above by d.

[1] We shall use the terms "session," "stream," and "packet stream" interchangeably.

[2] Throughout this paper, all inequalities and equalities involving functions are defined in a pointwise sense, e.g., $f \leq g$ means that $f(t) \leq g(t)$ for all t.

More generally, if a network element guarantees a session a service curve, we may bound the delay and backlog if the arrival stream of the session is "burstiness constrained" [4]. The burstiness constraint is succinctly summarized in terms of a "traffic envelope," which is defined next. A traffic envelope is also sometimes called an "arrival curve."

Definition II.2— (Traffic Envelope): Given a nondecreasing nonnegative function $b(\cdot)$, called a traffic envelope, we say that the stream R^{in} is b-smooth, or conforms to the envelope b, if

$$R^{\text{in}}[s+1,t] \leq b(t-s) \tag{4}$$

for all s and t satisfying $s < t$. For convenience, unless stated otherwise we define $b(u) = 0$ for all $u \leq 0$. In the special case where b is affine, i.e., $b(t) \equiv \sigma + \rho t$, $t = 1, 2, \cdots$ and $b(0) = 0$, we say that R^{in} is (σ, ρ)-smooth.

In terms of convolution, note that the stream R^{in} is b-smooth if and only if $r^{\text{in}} \leq r^{\text{in}} * b$. In fact, since we assume $b(0) = 0$, we have $r^{\text{in}} * b \leq r^{\text{in}}$ and so R^{in} is b-smooth if and only if $r^{\text{in}} = r^{\text{in}} * b$.

Given two functions f and g defined on the nonnegative integers, we define the "maximum horizontal distance" between f and g to be $\mathcal{D}(f||g)$, where

$$\mathcal{D}(f||g) = \max_{s:s\geq 1} \min\{\Delta\colon \Delta \geq 0 \text{ and } f(s) \leq g(s+\Delta)\}. \tag{5}$$

General results for delay bounds, buffer requirements, and service-curve composition were reported for the earlier service-curve definition in [7]. It turns out that even with the less restrictive service-curve definition considered in this paper, the same results hold. These results are summarized next. Proofs are omitted here, since it is easy to modify the proofs presented in [7] to obtain the corresponding results (see also [8] for short proofs).

Theorem II.1— (Upper Bound on Delay): Suppose a session is guaranteed service curve $S(\cdot)$ by a network element and the input traffic of the session is b-smooth. For every t, the virtual delay $d[t]$ suffered by the traffic of the session through the network element is upper-bounded by the maximum horizontal distance between b and S, $\mathcal{D}(b||S)$.

Theorem II.2— (Upper Bound on Backlog): Suppose a session is guaranteed service curve $S(\cdot)$ by a network element and the input traffic of the session is b-smooth. For every t, the backlog of the session in the network elemen $B[t]$ is upper bounded by the maximum vertical distance between S and b, i.e.,

$$B[t] \leq \max_{s:s\geq 0}\{b(s) - S(s)\}. \tag{6}$$

Consider a session passing through a server that requires an upper bound on delay of $d^{\max}$, whose input traffic is b-smooth. It is clear that if the server guarantees the session the service curve $S(x) = b(x - d^{\max})$, then the maximum delay requirement will be met. If the session requires both an upper bound on delay of $d^{\max}$ and that the backlog be bounded by $B^{\max}$, then both requirements will be met if the session is guaranteed the service curve $\hat{S}(x) = \max\{b(x - d^{\max}), [b(x) - B^{\max}]^+\}$, where we define $x^+ = \max\{x, 0\}$. The SCED scheduling policy defined in this paper yields a means for a server to *allocate* service curves to sessions so that their requirements are met. For a session that traverses a network through a path of servers, a service curve for the session can be allocated at each server along the path, in order to obtain an "end-to-end service-curve guarantee" through the entire network. The end-to-end service curve is obtained by convolving the service curves that are guaranteed at each server along the path, as stated in the next theorem.

Theorem II.3— (End-to-End Service): Consider a session sequentially traversing H network elements in tandem. Suppose the session is guaranteed service curve $S^h(\cdot)$ by network element h for $h = 1, \cdots, H$. Then the entire tandem network guarantees the session the service curve S^{net} where $S^{\text{net}} = S^1 * S^2 * \cdots * S^H$.

B. Regulators

Before proceeding to the definition and analysis of the SCED policy in the Section III, in this sub-section, we review network elements called "regulators," also known as "traffic shapers." Regulators buffer arriving traffic as necessary so that the output traffic stream is conformant to a traffic envelope. Without loss of generality, we may assume that traffic envelopes are sub-additive functions:

Definition II.3— (Sub-additive Function): A nondecreasing function $f(\cdot)$ defined on the nonnegative integers is said to be sub-additive if

$$f(x) + f(y) \geq f(x+y) \tag{7}$$

for any nonnegative integers x, $y \geq 0$.

Note that f is a sub-additive function if and only if $f * f \geq f$. In fact, we have $f * g \leq f$ for any two functions f and g, if $g(0) = 0$. Thus if f is a function such that $f(0) = 0$, then f is sub-additive if and only if $f * f = f$.

The following definition and results for a regulator were obtained independently and simultaneously by Agrawal and Rajan [1], Chang [3], Le Boudec [18], and Sariowan [22], with slightly different models. Here we restate the results from [22]. We use the notation $\lfloor x \rfloor$ to denote the greatest integer less than or equal to x.

Definition II.4— (Regulator): Suppose $b^R(\cdot)$ is a function such that $\lfloor b^R(\cdot) \rfloor$ is a sub-additive function. A b^R-regulator is defined to be a network element whose arrival and departure processes r^{in} and r^{out} are related by the equation

$$r^{\text{out}}(t) = (r^{\text{in}} * \lfloor b^R \rfloor)(t) \quad \text{for all } t \geq 1. \tag{8}$$

Theorem II.4: (Regulator's Output and Service) The output of a b^R-regulator is b^R-smooth. Furthermore, the b^R-regulator guarantees the service curve $\lfloor b^R(\cdot) \rfloor$ to the traffic stream that passes through it.

As we shall see, the regulator is closely related to the SCED scheduling algorithm, which is discussed in the next section. We shall revisit the regulator later in the paper and discuss this further.

III. Scheduling Policy and Schedulability Condition

Consider a set of sessions which share a server, where each session has a pre-specified service curve that needs to be guaranteed by the server. In this section, we introduce a

general scheduling algorithm called SCED which the server can use so that each session is guaranteed its required service curve. In the next sub-section, the SCED algorithm is defined. In Section III-B, we present our key result, which succinctly specifies a condition under which the required service curves can be simultaneously guaranteed by the server. Next, in Section III-C we explore possible algorithms for computing the deadlines associated with the SCED algorithm. In Section III-D we discuss the relationship between the SCED deadline calculations and a possible implementation of the traffic regulators discussed in Section II-B.

A. The SCED Algorithm

Consider M sessions sharing a server. Suppose session i, $i = 1, \cdots, M$, requires service-curve guarantee $\lfloor S_i(\cdot) \rfloor$. The number of arrivals from session i during slot t, number of departures from session i during slot t, and backlog for session i at the end of slot t are denoted by $R_i^{\text{in}}[t]$, $R_i^{\text{out}}[t]$, and $B_i[t]$, respectively.

We say that the server is *empty* at the end of slot t if all sessions have zero backlog at the end of slot t, i.e., $B_i[t] = 0$ for $i = 1, \cdots, M$. If the server is empty at the end of slot t, the server may elect to *reset* at that time. Basically, the purpose of resetting the server is to provide a mechanism for the server to "forget" any "extra" service a session might have received prior to resetting, thereby avoiding the so-called "punishment effect" discussed in [20]. The choice of whether or not to reset the server when it is empty is a philosophical design issue on which we do not take a position in this paper. If the server wishes to "punish" users for using "extra" service, it may choose to never reset when it is empty. On the other hand, if the server wishes to encourage users to utilize "extra" service, it may choose to always reset when it is empty. Our model in fact allows the server to selectively reset or not reset when the server is empty on a case-by-case basis, although it is not clear that there is any advantage to such selective resetting. For each slot t, define $\tau(t)$ to be the index of the latest slot, no larger than t, at the end of which the server resets. By convention the server resets at the end of slot 0, and if $\tau(t) = 0$ for all $t \geq 0$, we say the server never resets.

Define $Z_i(t)$ as

$$Z_i(t) = \min_{\tau(t) \leq s \leq t} \{R_i^{\text{in}}[1, s] + S_i(t - s)\}. \tag{9}$$

We say that $\lfloor Z_i(t) \rfloor$ is the *target* process for stream i, since if the departure process for the ith stream is above $\lfloor Z_i(t) \rfloor$, i.e., we have $R_i^{\text{out}}[1, t] \geq \lfloor Z_i(t) \rfloor$ for all t, then stream i is guaranteed the service curve $\lfloor S_i(\cdot) \rfloor$ as desired. To see this, note that if $R_i^{\text{out}}[1, t] \geq \lfloor Z_i(t) \rfloor$ for all t, we then have

$$\begin{aligned} R_i^{\text{out}}[1, t] &\geq \lfloor Z_i(t) \rfloor \\ &= \min_{\tau(t) \leq s \leq t} \{R_i^{\text{in}}[1, s] + \lfloor S_i(t - s) \rfloor\} \\ &= R_i^{\text{in}}[1, s^*] + \lfloor S_i(t - s^*) \rfloor \end{aligned}$$

and thus, for any t, there exists $s^* \leq t$ such that $R_i^{\text{out}}[1, t] \geq R_i^{\text{in}}[1, s^*] + \lfloor S_i(t - s^*) \rfloor$. In fact this may be slightly stronger than the desired service-curve guarantee, since s^* satisfies $\tau(t) \leq s^* \leq t$.

Fix a session i. Each packet from session i is conceptually assigned a unique *cumulative arrival index*, where the first packet to arrive has index 1 and the indices are assigned in order of packet arrivals. Note that we allow the possibility of greater than one packet arrival from stream i in a single slot. Suppose there is exactly one packet that arrives from session i in slot u. In this case, the packet's index n_i is given by $n_i = R_i^{\text{in}}[1, u]$. Note that if this packet is not served by slot t and $Z_i(t) \geq n_i$, then $R_i^{\text{out}}[1, t] < n_i \leq Z_i(t)$, so that the departure process would fall below the target process for stream i. Therefore, we would like to assign packet n_i a deadline d where d is the first slot, such that $Z_i(d) \geq n_i$. However, from (9), it is evident that in order to determine $Z_i(d)$, we must, in general, have knowledge of the arrival process for stream i up to slot d, which may not be known at time u. Fortunately, $R_i^{\text{in}}[1, s]$ is greater than or equal to n_i for $s \geq u$ and $S_i(\cdot)$ is nonnegative, and hence the terms in the minimum in (9) corresponding to $s \geq u$ do not affect when $Z_i(\cdot)$ first crosses above n_i. In addition, for purposes of estimating $Z_i(t)$ for $t \geq u$ we may make the assumption that $\tau(t) = \tau(u - 1)$, i.e. the server does not reset after time $u - 1$. Therefore, in order to calculate an appropriate deadline for packet n_i we will use $Z_i(t; u - 1)$ as an estimate of $Z_i(t)$, where

$$Z_i(t; u - 1) = \min_{\tau(u-1) \leq s \leq u-1} \{R_i^{\text{in}}[1, s] + S_i(t - s)\}. \tag{10}$$

This motivates the deadline assignments in the SCED policy, defined below.

Definition III.1— (SCED Policy): Given a set of M sessions, where session i, $i = 1, \cdots, M$, requires service-curve guarantee $\lfloor S_i(\cdot) \rfloor$, the SCED policy schedules the departure of packets by assigning a deadline to each incoming packet. In each slot, service priority is given to packets with the earliest deadline in the system. The deadlines are assigned as follows: given a packet from session i which arrives in slot u and has a cumulative arrival index n_i, the deadline $D_i(n_i)$ assigned to the packet is given by

$$D_i(n_i) = \min\{t\colon t \geq u \text{ and } Z_i(t; u - 1) \geq n_i\}. \tag{11}$$

Since a service curve is a nondecreasing function, it can be seen that the sequence of deadlines assigned to packets from a given session is nondecreasing, between consecutive reset points of the server. In other words, suppose that two packets from session i with indices n_i^1 and n_i^2 arrive during slots u_1 and u_2, respectively. If $n_i^1 \leq n_i^2$ then $D_i(n_i^1) \leq D_i(n_i^2)$, assuming that $\tau(u_1) = \tau(u_2)$.

The next theorem shows that if packets from a given session are assigned deadlines according to the SCED policy and each packet departs no later than its assigned deadline, then the session is guaranteed its service curve. Then we derive a schedulability condition which ensures that a packet always meets its deadline.

Theorem III.1— (Service-Curve Guarantees of SCED): Suppose session i is scheduled under the SCED policy defined above. If each packet departs no later than its assigned deadline, then the session is guaranteed service curve $\lfloor S_i(\cdot) \rfloor$.

Lemma III.1: For any slot t, let $N_i(t)$ be the total number of packets from stream i that arrive after slot $\tau(t)$, and that also have deadlines less than or equal to t. Then $N_i(t) = \lfloor \hat{Z}_i(t) \rfloor$, where

$$\hat{Z}_i(t) = \min_{\tau(t) \le s \le t} \{R_i^{\text{in}}[\tau(t)+1, s] + S_i(t-s)\}. \quad (12)$$

Proof: First we show that $N_i(t) \le \lfloor \hat{Z}_i(t) \rfloor$. If $N_i(t) = 0$, this is trivial. If $N_i(t) > 0$, then consider the last packet that arrives after slot $\tau(t)$, and that also has deadlines less than or equal to t. This packet has cumulative arrival index $n_i^* = R_i^{\text{in}}[1, \tau(t)] + N_i(t)$, and suppose it has deadline t^*, $t^* \le t$, and arrives during slot u^*, $\tau(t) + 1 \le u^* \le t^*$. From (11), we have

$$Z_i(t^*; u^* - 1) \ge n_i^*. \quad (13)$$

Since $\tau(t) + 1 \le u^* \le t^* \le t$, it follows that $\tau(u^* - 1) = \tau(t^*) = \tau(t)$. Also, by the definition of cumulative arrival index n_i^*, $R_i^{\text{in}}[1, s] \ge n_i^*$ for $u^* \le s \le t^*$. From (9) and (13), it therefore follows that $Z_i(t^*) = Z_i(t^*; t^*) \ge n_i^*$. Using $t^* \le t$ and $\tau(t^*) = \tau(t)$, it can be seen that $Z_i(t) \ge Z_i(t^*)$. Thus, $Z_i(t) \ge n_i^*$. Subtracting $R_i^{\text{in}}[1, \tau(t)]$ from both sides of this, we obtain $\hat{Z}_i(t) \ge N_i(t)$. Since $N_i(t)$ is integer valued, we thus have $\lfloor \hat{Z}_i(t) \rfloor \ge N_i(t)$.

Next we show that $\hat{Z}_i(t) < N_i(t) + 1$, which implies the reverse inequality, namely $\lfloor \hat{Z}_i(t) \rfloor \le N_i(t)$. First, consider the case where $R_i^{\text{in}}[\tau(t)+1, t] < N_i(t) + 1$. Since $S_i(0) = 0$, we have $\hat{Z}_i(t) \le R_i^{\text{in}}[\tau(t)+1, t] < N_i(t) + 1$. Second, consider the other case where $R_i^{\text{in}}[\tau(t)+1, t] \ge N_i(t) + 1$. In this case, the packet with index $n_i^{**} = R_i^{\text{in}}[1, \tau(t)] + N_i(t) + 1$ arrives at some time u^{**} that satisfies $\tau(t) + 1 \le u^{**} \le t$ and must have a deadline greater than t. We therefore must have $Z_i(t; u^{**} - 1) < n_i^{**}$. Since $\tau(u^{**} - 1) = \tau(t)$, we therefore have $Z_i(t) \le Z_i(t; u^{**} - 1)$. Thus $Z_i(t) < n_i^{**}$. Subtracting $R_i^{\text{in}}[1, \tau(t)]$ from both sides of this, we obtain $\hat{Z}_i(t) < N_i(t) + 1$. ∎

Proof of Theorem III.1: By Lemma III.1 and the assumption that each packet departs no later than its deadline, we have for any t

$$\begin{aligned} R_i^{\text{out}}[1,t] &= R_i^{\text{out}}[1,\tau(t)] + R_i^{\text{out}}[\tau(t)+1,t] \\ &= R_i^{\text{in}}[1,\tau(t)] + R_i^{\text{out}}[\tau(t)+1,t] \\ &\ge R_i^{\text{in}}[1,\tau(t)] + N_i(t) \\ &= R_i^{\text{in}}[1,\tau(t)] + \lfloor \hat{Z}_i(t) \rfloor \\ &= \lfloor Z_i(t) \rfloor \end{aligned}$$

which implies that the session is guaranteed service curve $\lfloor S_i(\cdot) \rfloor$. □

B. Feasible Service-Curve Allocations

In the next theorem, we provide conditions which guarantee that all packets from a set of sessions scheduled using SCED never miss their deadlines, and thus, by Theorem III.1, guarantee the desired service curve for each session.

For generality, we shall assume that the server has a variable capacity, meaning that the maximum number of packets that can be served during a slot varies with time. We say that the server is continuously backlogged in the interval $[s+1, t]$ if $\Sigma_{i=1}^M B_i[m] > 0$ for all m satisfying $s + 1 \le m \le t$. We say that a server has a *capacity curve* $C(\cdot)$ if within *any* interval $[s+1, t]$ in which the server is continuously backlogged, the amount of output traffic of the server is at least $C(t-s)$. A fixed-rate server, i.e., a server with a fixed capacity to serve c packets in every slot (where c is a constant), is just a special case of a server with a capacity curve $C(x) = cx$. Note that the constraint associated with a capacity curve is a much stricter service characterization than a service curve, and this stricter characterization is consistent with the universal service-curve definition of [20]. By considering a general variable capacity server instead of a fixed-rate server, this extends the scope of our analysis to include hierarchical scheduling architectures [12], [2]. For example, a fixed-rate server could allocate capacity curves to *classes* of sessions. Each class corresponds to a group of sessions, which are conceptually served by an associated variable rate server with its associated capacity curve $C(\cdot)$. The sessions belonging to a given class could then be scheduled using the SCED policy.

Theorem III.2— (Feasible Allocation for the SCED Policy): Consider a variable capacity server with capacity curve $C(\cdot)$ that serves M sessions. The traffic arriving from session i is assumed to be b_i-smooth. Then the SCED policy guarantees service curve $\lfloor S_i(\cdot) \rfloor$ to session i for $i = 1, \cdots, M$ if the following condition is satisfied:

$$\sum_{i=1}^{M} (b_i * S_i)(t) \le C(t) \quad (14)$$

for all positive integers t. With no assumptions on the arriving traffic from each session, a simpler sufficient condition is

$$\sum_{i=1}^{M} S_i(t) \le C(t) \quad (15)$$

for all positive integers t.

The admission control policy says that a set of sessions can be simultaneously served by a server with capacity curve $C(\cdot)$ if the sum of their service curves, i.e., $\Sigma_i S_i(t)$, is below the curve $C(t)$. This condition has a very interesting analogy to circuit-switching. Specifically, a set of circuits, each with bandwidth c_i, can be simultaneously served if the sum of their bandwidth, i.e., $\Sigma_i c_i$, is smaller than the total bandwidth c of the link. Hence, the service curve $S_i(\cdot)$ can be thought of as a *generalization* of the fixed bandwidth c_i. Rather than being represented by a *single number* c_i, the *generalized* bandwidth is represented by a *function* $S_i(\cdot)$.

Proof of Theorem III.2: It suffices to show that if condition (14) is satisfied, then *all* packets depart no later than their assigned deadlines, and hence by Theorem III.1, each session is guaranteed its service curve. To show this, we will show that if a packet misses its deadline, then (14) does not hold.

Let p be the first packet that misses its deadline, and let t_m be the deadline of this packet. Thus, the packet is not served in the interval $[1, t_m]$. Let $\tau^* \ge \tau(t_m)$ be the last slot no later than t_m that the total backlog in the server was zero, i.e., $\tau^* = \max\{s\colon s \le t_m \text{ and } \Sigma_{i=1}^M B_i[s] = 0\}$. Let t_s be the last slot in $[\tau^* + 1, t_m]$ in which a packet with a deadline greater than

t_m is served. If there is no such slot, define $t_s = \tau^*$. Thus, all packets served in the interval $[t_s + 1, t_m]$ have deadlines no greater than t_m. Furthermore, since the total backlog in the system was positive during this interval, it follows that the total number of packets served by the server in this interval is at least $C(t_m - t_s)$. Note that since a deadline is missed in slot t_m, and priority is given to packets with earlier deadlines, it follows that $t_s < t_m$.

Define $T[t_s+1, t_m]$ to be the number of packets which arrive and are assigned deadlines in the interval $[\tau(t_m) + 1, t_m]$, but do *not* depart in $[\tau(t_m) + 1, t_s]$. Each such packet "has to" depart in the interval $[t_s + 1, t_m]$. We will first show that $T[t_s + 1, t_m]$ is strictly greater than $C(t_m - t_s)$.

Since p is counted in $T[t_s + 1, t_m]$, as well as each packet served in the interval $[t_s + 1, t_m]$, we have

$$\begin{aligned} T[t_s + 1, t_m] &\geq 1 + \sum_{i=1}^{M} R_i^{\text{out}}[t_s + 1, t_m] \\ &\geq 1 + C(t_m - t_s) \\ &> C(t_m - t_s). \end{aligned} \tag{16}$$

We now proceed to upper bound $T[t_s + 1, t_m]$. Define $\mathcal{A}$ to be the set of all sessions which have zero backlog at the end of slot t_s *and* have no packets with deadlines greater than t_m departing in $[\tau(t_m) + 1, t_s]$. Define $\mathcal{B}$ to be the *complement* of $\mathcal{A}$.

Claim: Only packets from sessions in $\mathcal{A}$, and none in $\mathcal{B}$, are counted in $T[t_s + 1, t_m]$.

Proof of Claim: The claim is equivalent to saying that none of the packets from sessions in $\mathcal{B}$ are counted in $T[t_s + 1, t_m]$. Suppose a session j belongs to $\mathcal{B}$. Either some packet from session j with deadline greater than t_m is served in the interval $[\tau(t_m) + 1, t_s]$, or $B_j[t_s] > 0$. Suppose the former case is true. Note that packets from a given session are served in order of their deadlines. Thus, all packets from session j which arrived and are assigned deadlines in the interval $[\tau(t_m) + 1, t_m]$ must be served in the interval $[\tau(t_m) + 1, t_s]$. Thus, no packets from session j are counted in $T[t_s + 1, t_m]$. If the latter case is true, then by definition of t_s some packet, say p^*, with a deadline greater than t_m is served in slot t_s. Note that session j has at least one packet queued in the server during slot t_s. All packets from session j that are queued in the server during slot t_s must have a deadline greater than t_m, for otherwise packet p^* would not have been served, since the server gives priority to packets with earlier deadlines. Again, since packets from a given session are served in order of their deadlines, it follows that all packets from session j which arrived and are assigned deadlines in the interval $[\tau(t_m) + 1, t_m]$ must be served in the interval $[\tau(t_m) + 1, t_s]$. Thus, in the latter case, no packet from session j is counted in $T[t_s + 1, t_m]$. ◇

By definition of $\mathcal{A}$, any packet from a session in $\mathcal{A}$ that departs during the interval $[\tau(t_m)+1, t_s]$ must have a deadline that is no greater than t_m. Thus, the number of packets from a session i belonging to $\mathcal{A}$ that are counted in $T[t_s + 1, t_m]$ is equal to the total number of packets from session i that arrive and are assigned deadlines in the interval $[\tau(t_m) + 1, t_m]$, namely $N_i(t_m)$, minus the total number of packets from session i that are served in the interval $[\tau(t_m)+1, t_s]$. Thus, by the claim and the last statement, $T[t_s+1, t_m]$ is upper-bounded by

$$T[t_s + 1, t_m] \leq \sum_{i \in \mathcal{A}} \{N_i(t_m) - R_i^{\text{out}}[\tau(t_m) + 1, t_s]\}. \tag{17}$$

From Lemma III.1, we have

$$\begin{aligned} N_i(t_m) &= \lfloor \hat{Z}_i(t_m) \rfloor \\ &= \min_{\tau(t_m) \leq s \leq t_m} \{R_i^{\text{in}}[\tau(t_m) + 1, s] + \lfloor S_i(t_m - s) \rfloor\} \end{aligned}$$

and hence, we have for $i \in \mathcal{A}$

$$\begin{aligned} &N_i(t_m) - R_i^{\text{out}}[\tau(t_m) + 1, t_s] \\ &\quad \leq \min_{\tau(t_m) \leq s \leq t_m} \{R_i^{\text{in}}[\tau(t_m) + 1, s] + S_i(t_m - s)\} \\ &\qquad - R_i^{\text{out}}[\tau(t_m) + 1, t_s] \\ &\quad = \min_{\tau(t_m) \leq s \leq t_m} \{R_i^{\text{in}}[\tau(t_m) + 1, s] + S_i(t_m - s)\} \\ &\qquad - R_i^{\text{in}}[\tau(t_m) + 1, t_s] \\ &\quad \leq \min_{t_s \leq s \leq t_m} \{R_i^{\text{in}}[t_s + 1, s] + S_i(t_m - s)\} \\ &\quad \leq \min_{t_s \leq s \leq t_m} \{b_i(s - t_s) + S_i(t_m - s)\} \\ &\quad = (b_i * S_i)(t_m - t_s). \end{aligned}$$

It follows that

$$\begin{aligned} T[t_s + 1, t_m] &\leq \sum_{i \in \mathcal{A}} (b_i * S_i)(t_m - t_s) \\ &\leq \sum_{i=1}^{M} (b_i * S_i)(t_m - t_s). \end{aligned} \tag{18}$$

Combining inequalities (16) and (18), we obtain

$$\sum_{i=1}^{M} (b_i * S_i)(t_m - t_s) > C(t_m - t_s)$$

which implies that (14) does not hold. Condition (15) easily follows from (14) by substituting $b_i = \delta_0$. □

C. Algorithms for Calculating Deadlines

In this sub-section we shall use (11) to develop the algorithms depicted in Figs. 1 and 2 for calculating deadlines under the SCED policy.

The deadline of a packet $n_i = R_i^{\text{in}}[1, u-1] + l$ from stream i which arrives during slot u is given by

$$\begin{aligned} D_i(n_i) &= \min\{t\colon\ t \geq u \text{ and } Z_i(t; u-1) \geq n_i\} \\ &= \max\{u, Y_i(l; u)\}, \end{aligned} \tag{19}$$

where $Y_i(l; u) = \min\{t\colon\ Z_i(t; u-1) \geq R_i^{\text{in}}[1, u-1] + l\}$. Define

$$S_i^{-1}(n) = \min\{m\colon\ m \geq 1 \text{ and } S_i(m) \geq n\}. \tag{20}$$

If the server is empty at the end of current slot and the server resets **then**
  $RESET\_FLAG_i \leftarrow 1$
end if

In each slot u where packets arrive from stream i:
  If $RESET\_FLAG_i = 1$ **then**
    $RESET\_FLAG_i \leftarrow 0$
    $\Delta_i \leftarrow u - 1 + \Delta_i^0$
    For $l = 1$ to M_i
      $Table_i(l) \leftarrow u - 1 + S_i^{-1}(l)$
    end for
  else
    $\Delta_i \leftarrow \max\{\Delta_i, u - 1 + \Delta_i^0\}$
    For $l = 1$ to M_i
      $Table_i(l) \leftarrow \max\{Table_i(l), u - 1 + S_i^{-1}(l)\}$
    end for
  end if
  For each of the $R_i^{in}[u]$ packets that have arrived, assign the deadlines as follows:
    For $l = 1$ to $R_i^{in}[u]$
      If $l \le M_i$ **then**
        Assign the deadline $\max\{u, Table_i(l)\}$ to packet $r_i^{in}(u-1) + l$.
      else
        Assign the deadline $\max\{u, \lceil \Delta_i + \eta_i l \rceil\}$ to packet $r_i^{in}(u-1) + l$.
      end if
    end for
  Update $Table_i$ and Δ_i as follows:
    For $l = 1$ to M_i
      If $l + R_i^{in}[u] \le M_i$ **then**
        $Table_i(l) \leftarrow Table_i(l + R_i^{in}[u])$
      else
        $Table_i(l) \leftarrow \lceil \Delta_i + \eta_i(l + R_i^{in}[u]) \rceil$
      end if
    end for
    $\Delta_i \leftarrow \Delta_i + \eta_i R_i^{in}[u]$

Fig. 1. Algorithm for computing the deadlines assigned to packets from session i, in the case where the inverse of the service curve, $S_i^{-1}(\cdot)$, is parameterized by $S_i^{-1}(l) = \lceil \Delta_i^0 + \eta_i l \rceil$ for $l > M_i$. In this algorithm, at the end of a slot u where packets from stream i arrive, $Table_i(l)$ holds the value of $Y_i(l + R_i^{\text{in}}[u]; u)$ for $l = 1, 2, \cdots, M_i$, and Δ_i holds the value of $\Delta_i(u) + \eta_i R_i^{\text{in}}[u]$, relative to the definitions in the text, implementing (23) and (25).

We now show that $Y_i(l; u)$ can be computed recursively. For $l \ge 1$, we have

$$\begin{aligned} Y_i(l;u) &= \min\{t:\ Z_i(t; u-1) \ge R_i^{\text{in}}[1, u-1] + l\} \\ &= \min\{t:\ \min_{\tau(u-1) \le s \le u-1} \{R_i^{\text{in}}[1,s] + S_i(t-s)\} \\ &\ge R_i^{\text{in}}[1, u-1] + l\} \\ &= \max_{\tau(u-1) \le s \le u-1} \{\min\{t:\ R_i^{\text{in}}[1,s] + S_i(t-s) \\ &\ge R_i^{\text{in}}[1, u-1] + l\}\} \\ &= \max_{\tau(u-1) \le s \le u-1} \{s + S_i^{-1}(l + R_i^{\text{in}}[s+1, u-1])\}. \end{aligned} \tag{21}$$

Suppose that the server resets at the end of slot τ_0, i.e. $\tau(\tau_0) = \tau_0$. Furthermore, suppose that after slot τ_0 there are no arrivals

If the server is empty at the end of current slot and the server resets **then**
  $RESET\_FLAG_i \leftarrow 1$
end if

In each slot u where packets arrive from stream i:
  If $RESET\_FLAG_i = 1$ **then**
    $RESET\_FLAG_i \leftarrow 0$
    $\delta_{i,k} \leftarrow u - 1 + \delta_{i,k}^0$ for $k = 1, \ldots K_i$
  else
    $\delta_{i,k} \leftarrow \max\{\delta_{i,k}, u - 1 + \delta_{i,k}^0\}$ for $k = 1, \ldots K_i$
  end if
  For each of the $R_i^{in}[u]$ packets that have arrived, assign the deadlines as follows:
    For $l = 1$ to $R_i^{in}[u]$
      $\delta_{i,k} \leftarrow \delta_{i,k} + \eta_{i,k}$ for $k = 1, \ldots K_i$
      Assign the deadline $\max\{u, \max_{k=1}^{K_i} \lceil \delta_{i,k} \rceil\}$ to packet $r_i^{in}(u-1) + l$.
    end for

Fig. 2. Algorithm for computing the deadlines assigned to packets from session i, in the case where the service curve is a CPL curve $S_i(x) = \max\{0, \min_{k=1}^{K_i}\{\alpha_{i,k} + \beta_{i,k} x\}\}$. As in the text, $\delta_{i,k}^0 = -\alpha_{i,k}/\beta_{i,k}$ and $\eta_{i,k} = 1/\beta_{i,k}$. In this algorithm, at the end of a slot u, where packets from stream i arrive, $\delta_{i,k}$ holds the value of $\delta_{i,k}(u) + \eta_{i,k} R_i^{\text{in}}[u]$, implementing (27).

from stream i until slot u_1, i.e., $R_i^{\text{in}}[\tau_0 + 1, u_1 - 1] = 0 < R_i^{\text{in}}[u_1]$. Then, it follows from (21) that

$$Y_i(l; u_1) = u_1 - 1 + S_i^{-1}(l). \tag{22}$$

Proceeding iteratively, suppose there are no arrivals from stream i after slot u_m until slot u_{m+1}, i.e., $R_i^{\text{in}}[u_m + 1, u_{m+1} - 1] = 0 < R_i^{\text{in}}[u_m]$, and the server does not reset in the interval $[u_m, u_{m+1} - 1]$, i.e., $\tau(u_m - 1) = \tau(u_{m+1} - 1)$. In this case, using (21), we have

$$\begin{aligned} &Y_i(l; u_{m+1}) \\ &\quad = \max\{Y_i(l + R_i^{\text{in}}[u_m, u_{m+1} - 1]; u_m), \\ &\qquad \max_{u_m \le s \le u_{m+1} - 1} \{s + S_i^{-1}(l + R_i^{\text{in}}[s+1, u_{m+1} - 1])\} \\ &\quad = \max\{Y_i(l + R_i^{\text{in}}[u_m]; u_m), u_{m+1} - 1 + S_i^{-1}(l)\}. \end{aligned} \tag{23}$$

Therefore, in order to implement the calculation of deadlines for stream i, we may recursively calculate $Y_i(\cdot; u)$, using (22) and (23), in every slot u where packets from stream i arrive, and then use (19). In general, this may be a difficult task unless $Y_i(\cdot; u)$ can be parameterized.

One case where this is possible is the case where $S_i^{-1}(l) = \lceil \Delta_i^0 + \eta_i l \rceil$ for all $l > M_i$, for some integer M_i and constants Δ_i^0 and η_i. In this case we claim that $Y_i(l; u_{m+1}) = \lceil \Delta_i(u_{m+1}) + \eta_i l \rceil$ for $l > M_i$, where $\Delta_i(u_{m+1})$ can be recursively computed in terms of $\Delta_i(u_m)$ for $m \ge 1$. To see this, proceed by induction. For $m = 1$ and $l > M_i$ we have from (22) that

$$\begin{aligned} Y_i(l; u_1) &= u_1 - 1 + S_i^{-1}(l) = u_1 - 1 + \lceil \Delta_i^0 + \eta_i l \rceil \\ &= \lceil (u_1 - 1 + \Delta_i^0) + \eta_i l \rceil = \lceil \Delta_i(u_1) + \eta_i l \rceil \end{aligned}$$

where

$$\Delta_i(u_1) = u_1 - 1 + \Delta_i^0. \tag{24}$$

Assume inductively that $Y_i(l; u_m) = \lceil \Delta_i(u_m) + \eta_i l \rceil$ for some $m \geq 1$ and $l > M_i$. From (23) we then obtain for $l \geq M_i$ that

$$\begin{aligned} &Y_i(l; u_{m+1}) \\ &\quad = \max\{Y_i(l + R_i^{\text{in}}[u_m]; u_m), u_{m+1} - 1 + S_i^{-1}(l)\} \\ &\quad = \max\{\lceil \Delta_i(u_m) + \eta_i(l + R_i^{\text{in}}[u_m]) \rceil, \\ &\qquad u_{m+1} - 1 + \lceil \Delta_i^0 + \eta_i l \rceil\} \\ &\quad = \lceil \max\{\Delta_i(u_m) + \eta_i(l + R_i^{\text{in}}[u_m]), \\ &\qquad u_{m+1} - 1 + \Delta_i^0 + \eta_i l\} \rceil \\ &\quad = \lceil \Delta_i(u_{m+1}) + \eta_i l \rceil \end{aligned}$$

where

$$\Delta_i(u_{m+1}) = \max\{\Delta_i(u_m) + \eta_i R_i^{\text{in}}[u_m], u_{m+1} - 1 + \Delta_i^0\}. \tag{25}$$

The values of $Y_i(l; u)$ for $l = 1, 2, \cdots M_i$ can be stored in a table with M_i elements and recursively updated. This leads to the algorithm for calculating deadlines illustrated in Fig. 1. This algorithm requires $O(M_i)$ storage space for "state" variables associated with session i, and requires $O(M_i)$ comparison and memory access operations in each slot where packets from session i arrive. With appropriate hardware support, the comparison and memory operations could be executed in parallel in $O(1)$ time. Finally, we note that this algorithm is closely related to the "hybrid" FIR/IIR regulator introduced by Chang [3].

Bear in mind that the service curve $S_i(x)$ may take on an arbitrary (nondecreasing) shape for $x = 1, 2, \cdots, M_i$. Thus, the size of M_i is dependent on the duration of time for which we want the service curve to take on an arbitrary shape, roughly speaking. As we shall see later, the capability of supporting service curves with an arbitrary shape may be important in a network of servers.

Nevertheless, an important special case is where a service curve is "concave piecewise linear," defined below. In this case, the deadline calculations can be simplified considerably, as we discuss next.

Definition III.2: (CPL Curves) Define $\boldsymbol{P}$ to be the vector $\{(\alpha_k, \beta_k)\}_{k=1}^K$. Without any loss of generality, we assume $\beta_1 > \beta_2 > \cdots > \beta_K > 0$. Furthermore, we assume that the following inequality is satisfied:

$$1 \leq \frac{\alpha_2 - \alpha_1}{\beta_1 - \beta_2} < \frac{\alpha_3 - \alpha_2}{\beta_2 - \beta_3} < \cdots < \frac{\alpha_K - \alpha_{K-1}}{\beta_{K-1} - \beta_K}. \tag{26}$$

A Concave Piecewise Linear (CPL) curve with parameter $\boldsymbol{P}$, denoted by $S^{\boldsymbol{P}}(\cdot)$, is defined to be

$$S^{\boldsymbol{P}}(t) = \max\{0, \min_{k=1,\cdots,K}\{\alpha_k + \beta_k t\}\}.$$

Note that $S^{\boldsymbol{P}}(t)$ is defined not only for integers, but for all real t. We note that numbers $\alpha_{i+1} - \alpha_i/\beta_i - \beta_{i+1}$ are the values of t where the slope of $S^{\boldsymbol{P}}(t)$ changes. Without loss of generality, we can assume that $S^{\boldsymbol{P}}(\alpha_2 - \alpha_1/\beta_1 - \beta_2) > 0$.

Suppose that $S_i(\cdot) = S^{\boldsymbol{P}_i}(\cdot)$ where $\boldsymbol{P}_i = \{(\alpha_{i,k}, \beta_{i,k})\}_{k=1}^{K_i}$ is the parameter vector. We shall find that in this case, we can obtain a simpler algorithm for computing deadlines, which requires only $O(K_i)$ storage space for state variables, and requires only $O(K_i)$ comparison and memory access operations. Note that in this case, we have $S^{\boldsymbol{P}_i}(t) = \min_{k=1,\cdots,K_i} S_{i,k}(t)$, where $S_{i,k}(t) = \max\{0, \alpha_{i,k} + \beta_{i,k} t\}$. Upon examination of (21), it is evident that if we separately calculate the deadlines assuming that $S_i(\cdot) = S_{i,k}(\cdot)$ for each k, then the deadlines assuming that $S_i(\cdot) = S^{\boldsymbol{P}_i}(\cdot)$ can be obtained by taking the maximum of the deadlines calculated separately for each k. Thus, let us temporarily assume that $S_i(\cdot) = S_{i,k}(\cdot)$. In this case, note that $S_{i,k}^{-1}(l) = \lceil \delta_{i,k}^0 + \eta_{i,k} l \rceil$ for $l \geq 1$, where $\delta_{i,k}^0 = -\alpha_{i,k}/\beta_{i,k}$ and $\eta_{i,k} = 1/\beta_{i,k}$. Repeating the same arguments as in (24) and (25), it follows that $Y_i(l; u_m) = \lceil \delta_{i,k}(u_m) + \eta_{i,k} l \rceil$ for all $l > 0$, where

$$\delta_{i,k}(u_1) = u_1 - 1 + \delta_{i,k}^0$$

and

$$\delta_{i,k}(u_{m+1}) = \max\{\delta_{i,k}(u_m) + \eta_{i,k} R_i^{\text{in}}[u_m], u_{m+1} - 1 + \delta_{i,k}^0\}. \tag{27}$$

Using (19) and then combining all the calculations for $k = 1, \cdots, K_i$, we obtain the algorithm in Fig. 2.

We note that the algorithm in Fig. 2 was independently proposed in [21], where it was called "multirate scheduling." Finally, we note that the two proposed algorithms are used only for computing the deadline assignment for each packet. Another algorithm is needed to sort the deadlines of queued packets in order to determine the order of departure of these packets. Typically, such sorting algorithms require $O(\log M)$ operations in each slot, where M is the number of sessions sharing the server.

D. Implementing Regulators with Deadlines

We now discuss a close relationship between the deadline calculations for the SCED algorithm and a possible implementation of regulators. From (12), assuming that the server does not reset, i.e., $\tau(t) = 0$ for all t, we have that the number of packets that are assigned deadlines in the interval $[1, t]$ is given by

$$N_i(t) = \min_{0 \leq s \leq t}\{R_i^{\text{in}}[1, s] + \lfloor S_i(t - s) \rfloor\}. \tag{28}$$

Comparing this with (8), and identifying departure times with deadlines, it is immediately evident that a b^R-regulator can be implemented by calculating deadlines as in the SCED algorithm for a stream with service curve $S_i(\cdot) = b^R(\cdot)$, and then using these deadlines as the departure times for the regulator. In other words, if a packet is assigned the "deadline" d, then it departs the regulator during slot d.

A special case of particular interest is the case where $b^R(\cdot)$ is a CPL curve. The next lemma shows that, in fact, $\lfloor b^R(\cdot) \rfloor$ is sub-additive in this case, as long as $b^R(0) \geq 1$. The reader is referred to [22] for a formal proof.

Lemma III.2: (Sub-additive CPL Curves) Let $S^{\boldsymbol{P}}(\cdot)$ be a CPL curve, defined on the nonnegative real line, with parameter vector $\boldsymbol{P} = \{(\alpha_k, \beta_k)\}_{k=1}^{K}$. If $S^{\boldsymbol{P}}(0) \geq 1$, then $\lfloor S^{\boldsymbol{P}}(\cdot) \rfloor$ is sub-additive.

Combining these observations, it follows that the algorithm of Fig. 2 can be used to implement a regulator with $b^R(x) = \min_{k=1,\cdots,K}\{\alpha_i + \beta_i x\}$, as long as $\alpha_i \geq 1$ for each i. We note that this is equivalent to K "leaky bucket" regulators in series, as discussed in [7]. We also note that this differs slightly from the regulators defined in [3], since there the output streams of regulators are allowed to contain fractional packets in each slot, whereas here we constrain an output stream so that the number of packets in each slot is an integer. Note also that the service curve $\lfloor b^R(\cdot) \rfloor$ of a regulator here is integer valued, whereas the service curves of regulators in [3] may take on real values.

IV. Comparison with Other Scheduling Algorithms

In this section, we demonstrate that in certain cases, the SCED policy reduces to or is closely related to other well-known policies. We also demonstrate that the flexibility of SCED to allocate and guarantee arbitrarily specified service curves endows it with a greater capability to support *end-to-end* delay guarantees than other algorithms.

A. Reduction to VirtualClock

Consider the SCED algorithm whereby each service curve S_i is a "straight line," i.e., $S_i(x) = \rho_i x$ for all $x \geq 0$ and all i. Thus, the service curves are all of the CPL type and we may use Fig. 2 to calculate deadlines, where we set $K_i = 1$, $\delta_{i,1} = 0$, and $\eta_{i,1} = 1/\rho_i$. If we assume that the server never resets, it is seen that in this case the SCED algorithm reduces to the discrete time equivalent of the "VirtualClock" scheduling discipline [27] (see also [10], [24]). If the server resets at the end of each server busy period, we obtain a variation of VirtualClock whereby the "punishment" effect described in [20] is reduced, since the server effectively "forgets" about extra service a session may have received when a server busy period ends.

We note that the service curves guaranteed by a general class of schedulers called the "Latency-Rate Servers" [23], which is a generalization of VirtualClock, PGPS, Weighted Round Robin, Deficit Round Robin, and other schedulers, are a special case of the service curves guaranteed by SCED in which the curves are affine.

B. Reduction to EDF

Next consider the case where[3] $S_i = \delta_{d_i}$ for all i, where d_i is a nonnegative integer constant for all i. In this case, $S_i^{-1}(l) = d_i + 1$ for all $l > 0$, and hence Fig. 1 can be used to calculate deadlines, with $M_i = 0$, $\Delta_i^0 = d_i + 1$, and $\eta_i = 0$, which yields the following reduction: A packet from stream i that arrives during slot u is assigned the deadline $u + d_i$. Thus, in this case, the SCED algorithm reduces to the EDF algorithm.

[3] Recall the definition in (3).

Assume that each session i is b_i-smooth and that the server capacity function is $C(x) = x$. In this case, we have $(b_i * S_i)(x) = b_i(x - d_i)$ for all x, and hence Theorem III.2 implies that all deadlines will be met if

$$\sum_{i=1}^{M} b_i(t - d_i) \leq t \tag{29}$$

for $t = 1, 2, \cdots$ Note that this is a discrete time equivalent of the admission control condition for the EDF algorithm found in [13], [19], [25].

Note also that (29) is a necessary condition to guarantee that no deadlines will be missed under the assumed traffic model.[4] To see this, suppose that $R_i^{\text{in}}[1, t] = b_i(t)$ for all $t \geq 1$. In this case, the total number of packets that must be served by time t is given by the left hand side of (29), so if this quantity is greater than t it is clear that a deadline will be missed. Thus, the EDF and SCED policies are "optimal" scheduling policies in the sense of having the largest possible schedulability region, defined by (29), in the case of a single server.

Finally, we note that if no deadlines are missed for session i under the EDF policy then we have $r_i^{\text{out}}(t) \geq r_i^{\text{in}}(t - d_i) = (r_i^{\text{in}} * \delta_{d_i})(t)$ and hence session i is guaranteed the service curve $S_i = \delta_{d_i}$.

C. Relationship to RCS-EDF

Another class of scheduling policies, called RCS [26], is obtained by passing packets from each session through a regulator before allowing entry into the scheduler. If the scheduler implements the EDF policy, then we use the acronym RCS-EDF to denote the overall scheduling policy. In an RCS-EDF policy, each session is passed through a b_i-regulator[5] before entering the EDF scheduler, and the deadlines for the ith session are equal to the arrival times at the scheduler plus a constant d_i. The service curve guaranteed to each session with this policy is obtained by convolving the service curve of the associated regulator with the service curve guaranteed by the EDF scheduler. Thus, under this policy, session i receives service curve $\hat{S}_i(x) = (b_i * \delta_{d_i})(x) = b_i(x - d_i)$, where we assume that b_i is integer valued. Since b_i is a sub-additive function, we say the service curve $\hat{S}_i(x) = b_i(x - d_i)$ guaranteed by an RCS-EDF policy is a "shifted sub-additive" function. Note that an RCS-EDF policy is not work-conserving, in the sense that packets may be buffered in the system, in particular packets may be buffered in a regulator, while the server transmits fewer packets than it is capable of.

It is interesting to compare this to the SCED policy whereby we set $S_i(x) = b_i(x - d_i)$ for each i, and whereby the server never resets. Both policies guarantee the same service curve to each session. The difference is that the SCED policy is work conserving, in the sense that the server serves as many packets as possible in each slot. For example, in the RCS-EDF policy, consider a packet from session i that arrives at the regulator in slot u and departs the regulator in slot s. This

[4] Here we also assume that the traffic envelopes b_i are integer valued.

[5] In this context, the arriving traffic for session i is not necessarily b_i-smooth.

packet is assigned the deadline $s+d_i$ in the EDF scheduler. In the SCED scheduler, the same packet would also be assigned the deadline $s + d_i$ when it arrives at the scheduler in slot u. Thus, this packet is assigned the same deadline in both systems. The difference is that this packet will never leave the RCS-EDF system before slot s (because it does not enter the EDF scheduler until slot s), whereas in the SCED system it is possible that the packet may depart at any slot after slot u, in particular it may depart the SCED scheduler before slot s.

A work-conserving version of the RCS-EDF policy was discussed in [14], wherein packets are allowed to leave a regulator if the server would otherwise serve fewer packets than its capacity. The SCED policy is similar to this policy, but different in the following sense. The work conserving RCS-EDF policy in [14] does not specify the exact order in which packets may leave the regulator when there is excess capacity, whereas in the SCED policy this "excess" bandwidth is distributed according to deadlines, and hence according to the allocated service curves. In this sense, a SCED policy which guarantees "shifted sub-additive" service curves can be considered as a special case of the work conserving RCS-EDF policy.

Since the service curve guaranteed by an RCS-EDF policy is a shifted sub-additive function and the same service curve can also be guaranteed by the SCED policy, the SCED policy can guarantee the same delay bounds as that of the RCS-EDF policy. The main advantage of the SCED policy compared to the RCS-EDF policy is that the SCED policy can allocate more general service curves than the shifted sub-additive curves guaranteed by the RCS-EDF policy. In the next sub-section, we explore the benefits of SCED's flexibility for allocating general service curves, which ultimately results in a SCED's larger schedulability region than that of RCS-EDF in the network case.

D. End-to-End Scheduling

Next, we consider a more general problem, where a set of sessions shares a *network* of servers. Each session traverses a fixed path through a subset of servers in the network. Each session generates traffic that conforms to a known envelope. We are interested in the set of end-to-end delay bounds that can be obtained for each of the sessions, for various scheduling policies used at each of the servers. We formally define a schedulability region $\mathcal{R}_\pi$ for a given scheduling policy π as the set of end-to-end delay bounds that can be obtained for each of the sessions.

In particular, we formally define the schedulability region corresponding to the SCED policy, $\mathcal{R}_{\text{SCED}}$, as follows. Suppose there are n sessions, and they are indexed by the integers $1, 2, \cdots, n$. It is given that the traffic offered to the network by session i conforms to the envelope b_i. Suppose that session i travels through a total of H_i servers in the network, and that it is allocated the service curve S_i^h at the server at hop h along the path of servers it traverses, $h = 1, 2, \cdots, H_i$. The end-to-end service curve allocated to the session is thus $S_i^{\text{net}} = S_i^1 * S_i^2 * \cdots * S_i^{H_i}$, and the end-to-end delay bound for session i is $\mathcal{D}(b_i||S_i^{\text{net}})$. Suppose there are a total of Q servers in the network, indexed by the integers $1, 2 \cdots Q$. Let $p(i,h)$ be the index of the server traversed by session i at hop h. In accordance with Theorem III.2, we require that the service curves $\{S_i^h\}$ allocated at each server satisfy the feasibility condition

$$\sum_{(i,h):p(i,h)=q} S_i^h(x) \le c_q x, \qquad \text{for all } x, q = 1, 2, \cdots, Q \tag{30}$$

where c_q is the number of packets that server q can serve in each slot. The schedulability region $\mathcal{R}_{\text{SCED}}$ is the set of delay bounds that can be realized by some allocation of service curves $\{S_i^h\}$ at each server that satisfies the feasibility condition (30), i.e.,

$$\mathcal{R}_{\text{SCED}} = \{(d_1, \cdots, d_n)\text{: there exists } \{S_i^h\} \text{ satisfying (30) and } d_i \ge \mathcal{D}(b_i||S_i^{\text{net}})\}. \tag{31}$$

Next, we consider the schedulability region corresponding to the RCS-EDF policy, $\mathcal{R}_{\text{RCS}}$. Suppose that session i passes through a b_i^h-regulator before entering the EDF scheduler at hop h, where the deadlines of packets are equal to their arrival times at scheduler plus the constant d_i^h. Recall from our discusion of the RCS-EDF policy earlier that session i is guaranteed the shifted sub-additive service curve $\hat{S}_i^h = b_i^h * \delta_{d_i^h}$ at server $p(i,h)$, as long as no deadlines are missed. In turn, no deadlines are missed as long as the feasibility condition

$$\sum_{(i,h):p(i,h)=q} b_i^h(x - d_i^h) \le c_q x, \quad \text{for all } x \tag{32}$$

is satisfied at each server q. The end-to-end service curve guaranteed to session i is $\hat{S}_i^{\text{net}} = \hat{S}_i^1 * \hat{S}_i^2 * \cdots * \hat{S}_i^{H_i}$, and the end-to-end delay bound is $\mathcal{D}(b_i||\hat{S}_i^{\text{net}})$. The schedulability region for the RCS-EDF policy is therefore

$$\mathcal{R}_{RCS} = \{(d_1, \cdots, d_n)\text{: there exists } \{b_i^h, d_i^h\} \text{ satisfying (32) and } d_i \ge \mathcal{D}(b_i||\hat{S}_i^{\text{net}})\}. \tag{33}$$

It is clear that since the SCED policy can allocate the same service curve $\hat{S}_i^h = b_i^h * \delta_{d_i^h}$ at server $p(i,h)$ that the RCS-EDF policy can guarantee at the same server, the schedulability region for SCED is at least as large as that for RCS-EDF, i.e.,

$$\mathcal{R}_{\text{RCS}} \subset \mathcal{R}_{\text{SCED}}. \tag{34}$$

In fact, we shall see that the scedulability region for SCED can be *strictly* larger than that for RCS-EDF. The reason for this is that the service curves $\hat{S}_i^h$ guaranteed at each hop by the RCS-EDF policy are shifted sub-additive, whereas the generality of the SCED policy allows allocation of service curves that are not necessarily shifted sub-additive.

Next, we consider the use of the SCED policy at each server, but where the allocated service curves $\{S_i^h\}$ at each hop are constrained to be shifted sub-additive. We shall call this the SCED-S policy. From the discussion in the previous paragraph, it follows that the corresponding schedulability region for SCED-S is the same as that for RCS-EDF, i.e.,

$$\mathcal{R}_{\text{RCS}} = \mathcal{R}_{\text{SCED-S}}. \tag{35}$$

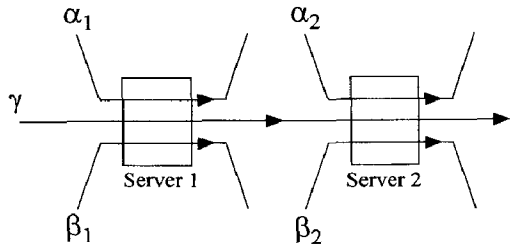


Fig. 3. Network configuration used to demonstrate that the SCED policy has a larger schedulability region than other policies.

We remark that in a continuous time context, it is shown in [14] that the class of RCS-EDF policies outperforms policies based on Generalized Processor Sharing (GPS) [20], such as WFQ [9] and Packetized-GPS [20], with respect to the schedulability region. In the following example we will show that $\mathcal{R}_{\text{SCED}}$ is strictly larger than $\mathcal{R}_{\text{RCS}}$. It is not known, however, whether or not $\mathcal{R}_{\text{SCED}}$ is the largest possible schedulability region.

1) Flexibility of SCED: An Example: Consider a network of two servers, labeled 1 and 2, each with a constant transmission capacity of one packet per slot, i.e., each server has a capacity function $C(x) = x$ for all x. The network is illustrated in Fig. 3. There are four "local" sessions arriving to the network. These sessions are labeled α_i and β_i, $1 \le i \le 2$. Traffic from sessions α_i and β_i arrives to the network at server i, and departs from the network after departing server i. There is one "global" session, γ, which arrives to the network at server 1. After the global session departs from server 1, it is fed to server 2 with zero propagation delay. After the global session departs from server 2, it leaves the network. Thus, each server supports two local sessions as well as the global session γ. For this network, we shall show that $\mathcal{R}_{\text{SCED}}$ is strictly larger than $\mathcal{R}_{\text{RCS}}$.

In order to simplify the discussion, we shall consider the SCED-S policy instead of the RCS-EDF policy and show that $\mathcal{R}_{\text{SCED}}$ is strictly larger than $\mathcal{R}_{\text{SCED-S}}$. In view of (35), this shows that $\mathcal{R}_{\text{SCED}}$ is strictly larger than $\mathcal{R}_{\text{RCS}}$. Also, in order to simplify the exposition of this example, throughout this subsection we ignore the integer constraints from all variables and equations. A more detailed analysis of the example, taking into account the integer constraints, is outlined in the Appendix.

Suppose the traffic arriving to the network from session α_i is b_{α_i}-smooth, where b_{α_i} is sub-additive. Each session α_i requires a delay bound of d_{α_i} slots. Similarly, suppose the traffic arriving to the network from session β_i is b_{β_i}-smooth, and requires a delay bound of d_{β_i}.

Assuming that the SCED policy is used at the servers, the local sessions α_i and β_i can be allocated the shifted sub-additive service curves $S_{\alpha_i}(x) = b_{\alpha_i}(x - d_{\alpha_i})$ and $S_{\beta_i}(x) = b_{\beta_i}(x - d_{\beta_i})$ at server i, respectively, for $i = 1, 2$. The "excess capacity curve" at each server i is defined to be

$$E_i(x) = x - b_{\alpha_i}(x - d_{\alpha_i}) - b_{\beta_i}(x - d_{\beta_i}) \tag{36}$$

which will be nonnegative for all x in our example. In accordance with Theorem III.2, we shall allocate the global session γ a service curve S_γ^i at server i, where $S_\gamma^i(x) \le E_i(x)$ for all x. Defining $S_\gamma = S_\gamma^1 * S_\gamma^2$, the delay bound for the global session under the SCED policy with general service curves is given by $d_\gamma^G = \mathcal{D}(b_\gamma || S_\gamma)$. Thus, under the SCED policy, by Theorem III.2, the local sessions α_i and β_i have delay upper bounded by d_{α_i} and d_{β_i}, $i = 1, 2$, and the global session has an end-to-end delay upper bounded by d_γ^G. In other words, we have $(d_{\alpha_1}, d_{\alpha_2}, d_{\beta_1}, d_{\beta_2}, d_\gamma^G) \in \mathcal{R}_{\text{SCED}}$.

We shall discuss a particular case where the delay bounds obtained above for all sessions under SCED with general service curves can not be achieved by *any* SCED-S policy. More specifically, consider any SCED-S policy that is able to guarantee the local sessions the same upper bounds on delay that the SCED policy is able to guarantee, namely d_{α_i} and d_{β_i}, $i = 1, 2$. Let d_γ^S be the end-to-end delay bound for the global session γ under the SCED-S policy. We shall give an example where d_γ^S must be at least approximately $2d_\gamma^G$. Thus, under any SCED-S (and hence any RCS-EDF) policy, the end-to-end delay for the global session is at least roughly twice what is achievable under the SCED policy, assuming that the delay requirements for the local sessions are met. This shows that $(d_{\alpha_1}, d_{\alpha_2}, d_{\beta_1}, d_{\beta_2}, d_\gamma^S) \notin \mathcal{R}_{\text{RCS}}$ if d_γ^S is less than approximately $2d_\gamma^G$, and in particular that $\mathcal{R}_{\text{SCED}}$ is strictly larger than $\mathcal{R}_{\text{RCS}}$.

We first compute d_γ^G, the upper bound for the maximum end-to-end delay for the global session γ. Let μ_H and μ_L be positive constants satisfying $\mu_H > \mu_L$ and $\mu_H + \mu_L = 1$. Suppose that b_{α_i} is the sub-additive function

$$b_{\alpha_i}(x) = \mu_H x^+ + (\mu_L - \mu_H)(x - \tau_{\alpha_i})^+$$

and τ_{α_i} is a nonnegative constant, sometimes called the maximum burst length.

If we set the delay bound for session α_i to zero, we may assign session α_i the service curve to $S_{\alpha_i}(x) = b_{\alpha_i}(x)$. Furthermore, set $\tau_{\alpha_1} = 0$ and $\tau_{\alpha_2} = \tau_\alpha > 0$. In this case we have

$$S_{\alpha_1}(x) = \mu_L x^+$$
$$S_{\alpha_2}(x) = \mu_H x^+ + (\mu_L - \mu_H)(x - \tau_\alpha)^+.$$

Similarly, suppose that

$$b_{\beta_i}(x) = (\mu_H - \mu_L)x^+.$$

The delay bound for session β_i is $d_{\beta_1} = \tau_\beta$, $d_{\beta_2} = \tau_\alpha + \tau_\beta$, and τ_β is a positive constant. Thus we may assign to session β_i the service curve $S_{\beta_i}(x) = b_{\beta_i}(x - d_{\beta_i})$, i.e.,

$$S_{\beta_1}(x) = (\mu_H - \mu_L)(x - \tau_\beta)^+$$
$$S_{\beta_2}(x) = (\mu_H - \mu_L)(x - \tau_\alpha - \tau_\beta)^+.$$

With these definitions, we have for $i = 1, 2$

$$S_{\beta_i}(x) = x - b_{\alpha_i}(x) - b_{\beta_i}(x - d_{\beta_i})$$
$$= E_i(x)$$

where

$$E_1(x) = \mu_H x^+ + (\mu_L - \mu_H)(x - \tau_\beta)^+$$
$$E_2(x) = \mu_L x^+ + (\mu_H - \mu_L)(x - \tau_\alpha)^+ + (\mu_L - \mu_H) \cdot (x - \tau_\alpha - \tau_\beta)^+$$

as illustrated in Figs. 4 and 5.

Note that $E_1(x)$ has slope μ_H for $0 < x < \tau_\beta$ and slope μ_L for $x > \tau_\beta$. Also, $E_2(x)$ has slope μ_L for $0 < x < \tau_\alpha$, slope μ_H for $\tau_\alpha < x < \tau_\alpha + \tau_\beta$, and again slope μ_L for $x > \tau_\alpha +$

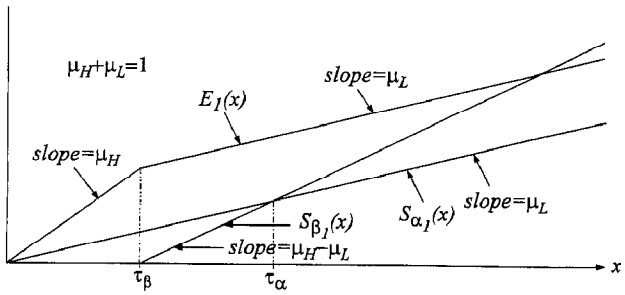


Fig. 4. Illustration of the calculation of the excess capacity curve E_1 at server 1.

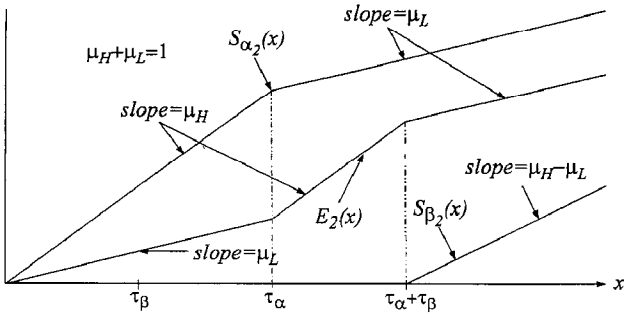


Fig. 5. Illustration of the calculation of the excess capacity curve E_2 at server 2.

τ_β. For a general SCED policy at each server, according to Theorem III.2, each session will receive their corresponding service curves as long as $S_{\alpha_i}(x) + S_{\beta_i}(x) + S^i_\gamma(x) \le x$ for all x. In view of (36), this will be satisfied if $S^i_\gamma(x) \le E_i(x)$. We will therefore set $S^i_\gamma(x) = E_i(x)$.

The end-to-end service curve for the global session γ, say S_γ, is[6]

$$\begin{aligned} S_\gamma(x) &= (S^1_\gamma * S^2_\gamma)(x) = (E_1 * E_2)(x) = E_2(x) \\ &= \mu_L x^+ + (\mu_H - \mu_L)(x-\tau_\alpha)^+ + (\mu_L - \mu_H) \\ &\quad \cdot (x - \tau_\alpha - \tau_\beta)^+. \end{aligned}$$

Next, we assume the arrival curve b_γ of the global session is given by

$$b_\gamma(x) = \mu_H x^+ + (\mu_L - \mu_H)(x - \tau_\gamma)^+.$$

We set the burst length τ_γ so that $b_\gamma(\tau_\gamma) = S_\gamma(\tau_\alpha + \tau_\beta)$, or equivalently

$$\tau_\gamma = \tau_\beta + \tau_\alpha \frac{\mu_L}{\mu_H}. \tag{37}$$

Referring to Fig. 6, this implies that the maximum end-to-end delay for the global session γ is upper bounded by $\mathcal{D}(b_\gamma || S_\gamma) = d^G_\gamma$, where

$$\begin{aligned} d^G_\gamma &= \tau_\alpha + \tau_\beta - \tau_\gamma \\ &= \tau_\alpha \left(1 - \frac{\mu_L}{\mu_H}\right). \end{aligned} \tag{38}$$

Thus, we have shown that by using the general SCED policy at each server, it is possible to guarantee the local sessions

[6]Recall that here, we are ignoring integer constraints, so that the convolutions below are performed over the set of real numbers.

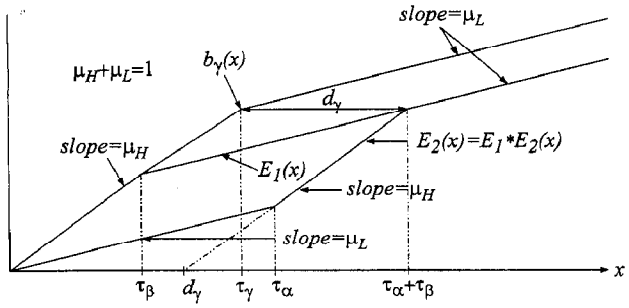


Fig. 6. Illustration of delay calculations for global session γ in network of Fig. 3. Under the SCED policy, the global session γ is assigned the service curve E_1 and E_2 at servers 1 and 2, ignoring integer constraints. The end-to-end service curve for session γ under SCED is therefore approximately $E_1 * E_2 = E_2$, which leads to an end-to-end delay bound of approximately d_γ. Under any SCED-S policy, the delay at server 1 is at least $\mathcal{D}(b_\gamma || E_1) = d_\gamma$, as is evident in the illustration. The dashed line with slope μ_h extending the curve E_2 helps to illustrate why in the best case, the delay bound in the SCED-S scheduler at server 2 must be at least d_γ. Hence, under any SCED-S policy, the delay of the global session is at least $2d_\gamma$.

α_i and β_i the delay bounds of d_{α_i} and d_{β_i}, respectively, and guarantee the global session γ an upper bound on end-to-end delay of d^G_γ. We remark that the deadline calculations for the global session can be done using the algorithm in Fig. 1, but not using the algorithm in Fig. 2 since the service curve for the global session at server 2 is not a CPL curve.

Next, we examine the delay bounds achievable by any SCED-S policy. We assume that at each server i, the global session γ is guaranteed shifted sub-additive service curve $g_i * \delta_{d_i}$, where g_i is some sub-additive function and d_i is some nonnegative integer. Under the SCED-S policy, the global session γ receives the end-to-end service curve $S^S_\gamma(x) = (g_1 * \delta_{d_1}) * (g_2 * \delta_{d_2})(x) = g(x - d_1 - d_2)$, where $g = g_1 * g_2$. The end-to-end delay bound for the global session γ under the SCED-S policy is $d^S_\gamma = \mathcal{D}(b_\gamma || S^S_\gamma)$.

In order for sessions α_i and β_i to be guaranteed the delay bounds of d_{α_i} and d_{β_i}, it is necessary that they be allocated service curves no smaller than $b_{\alpha_i}(x - d_{\alpha_i})$ and $b_{\beta_i}(x - d_{\beta_i})$, respectively, for $i = 1, 2$. Thus, in order to satisfy the feasibility condition, the allocated service curves for the global session γ at server i must satisfy

$$g_i(x - d_i) \le E_i(x), \quad \text{for all } x \ge 1. \tag{39}$$

Given any η such that $0 < \eta < 1$ we shall show that if τ_α and τ_β are sufficiently large, then $d^S_\gamma \ge (1+\eta) d^G_\gamma$. Thus, in this case, the smallest end-to-end delay for the global session γ that is possible to achieve under any SCED-S (and hence any RCS-EDF) policy must be roughly twice the end-to-end delay bound that could be guaranteed with a general SCED policy.

The main idea is as follows. First, we may assume without loss of generality that $g_i = g$, since we may replace each g_i with the smaller envelope $g = g_1 * g_2$ and the end-to-end service curve S^S_γ remains the same. Assuming that $g_1 = g_2 = g$, then constraint (39) is given by $g(x - d_i) \le E_i(x)$. From Fig. 6, since the slope of the dashed line segment is μ_H, it is evident that if d_2 is set significantly smaller than d_γ, then the constraint $g(x - d_2) \le E_2(x)$ at $x = \tau_\alpha$ implies that the slope of g must be strictly less than μ_H because of the sub-additivity of g. In turn, this would imply that $\mathcal{D}(b_\gamma || g)$ is

large, and hence the maximum delay for the global session at server 1 is large, since the initial slope of b_γ is equal to μ_H and we choose τ_α and τ_β to be large. Thus if the maximum end-to-end delay is minimized, the maximum delay at the second server is at least on the order of d_γ. On the other hand, the maximum delay for the global session at server 1 is at least $d_\gamma = \mathcal{D}(b_\gamma||E_1)$. Thus, the total end-to-end delay for the global session γ under any SCED-S policy is at least on the order of $2d_\gamma$.

We now argue this more carefully. We have

$$\begin{aligned} d_\gamma^S &= \mathcal{D}(b_\gamma||S_\gamma^S) \\ &= \mathcal{D}(b_\gamma||g(\cdot - d_1 - d_2)) \\ &= \max_{x:x\geq 1} \min\{d\colon d \geq 0 \text{ and } b_\gamma(x) \leq g(x - d_1 - d_2 + d)\} \\ &\geq \max_{x:x\geq 1} \min\{d\colon b_\gamma(x) \leq g(x - d_1 - d_2 + d)\} \\ &= d_2 + \max_{x:x\geq 1} \min\{d\colon b_\gamma(x) \leq g(x - d_1 + d)\} \\ &= d_1^S + d_2 \end{aligned} \tag{40}$$

where we define

$$d_1^S = \max_{x\colon x\geq 1} \min\{d\colon b_\gamma(x) \leq g(x - d_1 + d)\}. \tag{41}$$

Fix any η such that $0 < \eta < 1$. We now distinguish two cases. In the first case, suppose that $d_2 \geq \eta d_\gamma^G$. Note that since (39) holds we have

$$\begin{aligned} d_1^S &= \max_{x:x\geq 1} \min\{d\colon b_\gamma(x) \leq g(x - d_1 + d)\} \\ &\geq \max_{x:x\geq 1} \min\{d\colon b_\gamma(x) \leq g_1(x - d_1 + d)\} \\ &\geq \max_{x:x\geq 1} \min\{d\colon b_\gamma(x) \leq E_1(x + d)\} \\ &\geq \min\{d\colon b_\gamma(\tau_\gamma) \leq E_1(\tau_\gamma + d)\} \\ &= \min\left\{d\colon \mu_H\tau_\beta + \tau_\alpha\mu_L \leq \mu_H\tau_\beta + \mu_L\tau_\alpha\frac{\mu_L}{\mu_H} + \mu_L d\right\} \\ &= \tau_\alpha\left(1 - \frac{\mu_L}{\mu_H}\right) \\ &= d_\gamma^G. \end{aligned}$$

Using (40), it therefore follows that in this case we have $d_\gamma^S \geq d_1^S + d_2 \geq (1+\eta)d_\gamma^G$.

Next we consider the other case, where $d_2 < \eta d_\gamma^G$. In this case, we will show that d_1^S is large, in particular larger than $(1+\eta)d_\gamma^G$. First, note that since g is a sub-additive function, then for any $\Delta > 0$ we have

$$\begin{aligned} g(x) &\leq \left\lfloor\frac{x}{\Delta}\right\rfloor g(\Delta) + g\left(x - \left\lfloor\frac{x}{\Delta}\right\rfloor\Delta\right) \\ &\leq \frac{g(\Delta)}{\Delta}x + g(\Delta). \end{aligned}$$

From (39), we have

$$g(\tau_\alpha - d_2) \leq g_2(\tau_\alpha - d_2) \leq E_2(\tau_\alpha).$$

Setting $\Delta = \tau_\alpha - d_2$ in the above upper bound for g yields

$$g(x) \leq E_2(\tau_\alpha)\left(\frac{x}{\tau_\alpha - d_2}\right) + E_2(\tau_\alpha).$$

We thus have

$$\begin{aligned} d_1^S &= \max_{x:x\geq 1} \min\{d\colon b_\gamma(x) \leq g(x - d_1 + d)\} \\ &\geq \max_{x:x\geq 1} \min\{d\colon b_\gamma(x) \leq g(x + d)\} \\ &\geq \max_{x:x\geq 1} \min\left\{d\colon b_\gamma(x) \leq E_2(\tau_\alpha)\left(\frac{x+d}{\tau_\alpha - d_2}\right) + E_2(\tau_\alpha)\right\} \\ &\geq \min\left\{d\colon b_\gamma(\tau_\gamma) \leq E_2(\tau_\alpha)\left(\frac{\tau_\gamma + d}{\tau_\alpha - d_2}\right) + E_2(\tau_\alpha)\right\} \\ &= (b_\gamma(\tau_\gamma) - E_2(\tau_\alpha))\left(\frac{\tau_\alpha - d_2}{E_2(\tau_\alpha)}\right) - \tau_\gamma \\ &= (\mu_H\tau_\gamma - \mu_L\tau_\alpha)\left(\frac{\tau_\alpha - d_2}{\mu_L\tau_\alpha}\right) - \tau_\gamma \\ &= \mu_H\tau_\beta\left(\frac{\tau_\alpha - d_2}{\mu_L\tau_\alpha}\right) - \tau_\beta - \tau_\alpha\frac{\mu_L}{\mu_H} \\ &\geq \mu_H\tau_\beta\left(\frac{\tau_\alpha\left(1 - \eta\left(1 - \frac{\mu_L}{\mu_H}\right)\right)}{\mu_L\tau_\alpha}\right) - \tau_\beta - \tau_\alpha\frac{\mu_L}{\mu_H}. \end{aligned} \tag{42}$$

Define $\theta = 0.5((\mu_H/\mu_L) - 1)(1 - \eta)$, and note that $\theta > 0$. Defining $k = (1 + 2\theta/1 + \theta)$, note that $k > 1$ and

$$k = \frac{\frac{\mu_H}{\mu_L} - \eta\left(\frac{\mu_H}{\mu_L} - 1\right)}{1 + \theta}. \tag{43}$$

Using $\mu_L\tau_\alpha \leq \mu_L k\tau_\alpha$ in (42), and then substituting (43) we have

$$\begin{aligned} d_1^S &\geq \mu_H\tau_\beta\left(\frac{\tau_\alpha\left(1 - \eta\left(1 - \frac{\mu_L}{\mu_H}\right)\right)}{\mu_L k\tau_\alpha}\right) - \tau_\beta - \tau_\alpha\frac{\mu_L}{\mu_H} \\ &= \tau_\beta\left(\frac{\mu_H\left(1 - \eta\left(1 - \frac{\mu_L}{\mu_H}\right)\right)}{\mu_L k}\right) - \tau_\beta - \tau_\alpha\frac{\mu_L}{\mu_H} \\ &= \tau_\beta(1+\theta) - \tau_\beta - \tau_\alpha\frac{\mu_L}{\mu_H} \\ &= \theta\tau_\beta - \tau_\alpha\frac{\mu_L}{\mu_H}. \end{aligned} \tag{44}$$

Now fixing τ_α, we may make the right hand side of (44) arbitrarily large by making τ_β sufficiently large. In particular, by choosing τ_β sufficiently large, we have $d_\gamma^S \geq d_1^S \geq (1+\eta)d_\gamma^G$.

It follows that τ_α and τ_β can be chosen such that $d_\gamma^S > (1+\eta)d_\gamma^G$ in any case. Thus, choosing η near unity, any SCED-S policy yields a maximum delay for the global session which must be approximately twice that achievable using the SCED policy. In particular, this proves that the schedulability region in the network with a general SCED policy at each server is strictly greater than the schedulability region with a RCS-EDF policy at each server.

2) Comparison of Schedulability Regions: It is not known in general how much smaller the achievable end-to-end delays are for the SCED policy, as compared to the RCS-EDF policy. However, we now present a simple analysis that may provide an answer to this question. In particular, we show that if $(d_1, d_2, \cdots, d_n) \in \mathcal{R}_{\text{SCED}}$ then $(H_1 d_1, H_2 d_2, \cdots, H_n d_n) \in \mathcal{R}_{\text{RCS}}$, where we use the notation defined at the beginning of Section IV-D. For simplicity, assume that each server has a capacity function of $C(x) = x$. Suppose we use the SCED policy to allocate the service curve S_i^h to session i at the hth> server along its route through the network. The end-to-end service curve for session i under the SCED policy is then $S_i^{\text{net}} = S_i^1 * S_i^2 * \cdots * S_i^{H_i}$. Thus, we have $(d_1, d_2, \cdots d_n) \in \mathcal{R}_{\text{SCED}}$, where $d_i = \mathcal{D}(b_i || S_i^{\text{net}})$.

In order to guarantee the allocated service curves at each server, we are assuming that we have

$$\sum_{(i,h):p(i,h)=q} S_i^h(x) \le x \tag{45}$$

for each server q. Note that for all i and h we have

$$S_i^h(x) \ge b_i(x - d_i) \tag{46}$$

for otherwise we would have $S_i^{\text{net}}(x) \le S_i^h(x) < b_i(x - d_i)$ for some i, which would contradict the definition of d_i.

We now show the existence of a RCS-EDF policy which is able to guarantee session i the end-to-end delay bound $H_i d_i$. Specifically, the traffic from session i is passed through a g_i^h-regulator before entering the EDF scheduler at the hth server along its path, where $g_i^h(x) = b_i(x)$, and it is assigned the delay bound d_i^h at this EDF scheduler, where $d_i^h = d_i$. For all servers q, we then have

$$\begin{aligned}\sum_{(i,h):p(i,h)=q} g_i^h(x - d_i^h) &= \sum_{(i,h):p(i,h)=q} b_i(x - d_i) \\ &\le \sum_{(i,h):p(i,h)=q} S_i^h(x) \\ &\le x\end{aligned}$$

where the inequalities follow from (45) and (46). Since this is the sufficient condition for schedulability of the delay vector $\{d_i^j\}$ under the RCS-EDF policy at each server, it follows that no deadlines will be missed at any EDF scheduler. The end-to-end service curve for session i under the RCS-EDF policy is $\hat{S}_i^{\text{net}}(x) = b_i(x - H_i d_i)$, and thus the RCS-EDF policy is capable of guaranteeing the delay bound $H_i d_i$ for session i. In other words, we have $(H_1 d_1, H_2 d_2, \cdots, H_n d_n) \in \mathcal{R}_{\text{RCS}}$.

It may appear that this argument significantly underestimates the schedulability region of the RCS-EDF policies with respect to that of the SCED policies. On the other hand, the example of Section IV-D.1 shows that for a session that traverses two hops, the best end-to-end delay for RCS-EDF may be a factor of two larger than that achievable under the SCED policy. It is an open question whether or not there exists an example whereby a session that traverses H hops has a best-case end-to-end delay under the RCS-EDF policy that is H times larger than what is achievable under the SCED policy, for $H > 2$.

V. Discussion and Conclusion

We have presented SCED, a versatile scheduling policy which is a generalization of both the VirtualClock and the EDF algorithms. Like the EDF algorithm, the SCED policy has the optimal schedulability region for the case of a single server. The deadline assignments in SCED are more flexible than in EDF, which may be of some value for other performance measures besides maximum delay, such as average delay. We have seen that if the allocated service curves are parameterized by concave piece-wise linear functions, the SCED deadline assignment algorithm reduces to a simple generalization of the VirtualClock policy.

As with the EDF policy, it is necessary to check several inequalities in order to determine whether or not it is feasible for a single server to support a given set of sessions with pre-specified performance requirements. This is more complex than what is necessary for scheduling algorithms which are based on bandwidth-allocations. It may be possible to use techniques as introduced in [11] to reduce the computational complexity of admission control.

In the case of a network of servers, we have shown by example that the SCED policy has a greater capability than other known policies to support end-to-end delay bounds. The key to this capability is the ability of SCED to efficiently support service curves with arbitrary shapes. However, by allocating service curves with a general shape, the deadline assignments under SCED are more complex. It is not known whether or not using the SCED policy at each server yields an optimal schedulability region for a network of servers. It is worth noting that since we have defined schedulability regions in terms of easily obtained delay *bounds*, we have not shown that the actual worst-case end-to-end delay for the SCED policy can be strictly less than for other policies. We conjecture, however, that this is indeed the case. Such a "sample path" result is beyond the scope of this paper.

The problem of synthesizing a rational approach to the allocation of service curves in the context of a network of servers is a wide open problem. The problem appears difficult, because there are many degrees of freedom in choosing a service curve, since it is a function. In particular there are many possible service curves that could be allocated for a session at each hop, that result in the same end-to-end service curve. It is not clear how the service curves should be allocated at each hop in order to maximize the number of sessions that can be supported by the network.

These issues need to be investigated further, before the practical utility of the SCED policy can be accurately evaluated. At the least, the framework of allocating service curves offers new insight to the synthesis of network scheduling algorithms.

Appendix

In this section, we outline how the argument in Section IV-D.1 can be modified to take into account the integer constraints.

First, we perform the substitution to the following quantities defined in Section IV-D.1:

$$\begin{aligned} b_{\alpha_i}(x) &\leftarrow \lfloor 1 + b_{\alpha_i}(x) \rfloor \\ b_{\beta_i}(x) &\leftarrow \lfloor 1 + b_{\beta_i}(x) \rfloor \\ b_{\gamma}(x) &\leftarrow \lfloor 1 + b_{\gamma}(x) \rfloor \\ S_{\alpha_i}(x) &\leftarrow \lfloor S_{\alpha_i}(x) \rfloor \\ S_{\beta_i}(x) &\leftarrow \lfloor S_{\beta_i}(x) \rfloor \\ S^i_{\gamma}(x) &\leftarrow \lfloor S^i_{\gamma}(x) \rfloor \\ d_{\alpha_i} &\leftarrow \frac{2}{\mu_L} \\ d_{\beta_i} &\leftarrow d_{\beta_i} + \frac{2}{\mu_H - \mu_L}. \end{aligned}$$

Essentially the same arguments can be made, taking into account the modifications above and by setting τ_α and τ_β sufficiently large. The key idea is to use the inequality $\lfloor 1 + x \rfloor \geq x$ which holds for any real x, and to lower bound convolutions over integer sets by convolutions over sets of real numbers. In particular, the delay bound for the global session under the SCED policy becomes

$$d^G_\gamma = \tau_\alpha \left(1 - \frac{\mu_L}{\mu_H}\right) + \frac{3}{\mu_L} \tag{47}$$

and we can show that if τ_α and τ_β are sufficiently large, then the maximum end-to-end delay d^S_γ for the global session under any SCED-S (and hence any RCS-EDF) policy is lower bounded as follows:

$$d^S_\gamma \geq (1+\eta)\tau_\alpha \left(1 - \frac{\mu_L}{\mu_H}\right) - \frac{\epsilon_0}{\mu_L} \tag{48}$$

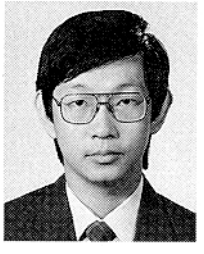
where $\epsilon_0 = 2 + 2(\mu_H/\mu_L)$. The terms $3/\mu_L$ and ϵ_0/μ_L become negligible if we choose τ_α and τ_β to be sufficiently large, so that d^S_γ is at least approximately $(1+\eta)d^G_\gamma$.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees and the editor, Prof. R. Guérin, for their generous comments and suggestions that have improved the paper.

REFERENCES

[1] R. Agrawal and R. Rajan, "A general framework for analyzing schedulers and regulators," in *Proc. 34th Annu. Allerton Conf. Communication, Control, and Computing*, 1996, pp. 239–248.
[2] J. Bennett and H. Zhang, Hierarchical packet fair queueing algorithms, *IEEE/ACM Trans. Networking*, vol. 5, pp. 675–689, Oct. 1997.
[3] C.-S. Chang, "On deterministic traffic regulation and service guarantees: a systematic approach by filtering," *IEEE Trans. Inform. Theory*, vol. 44, pp. 1097–1110, May 1998.
[4] R. L. Cruz, "A calculus for network delay, Part I: Network Elements in Isolation," *IEEE Trans. Inform. Theory*, vol. 37, pp. 114–131, Jan. 1991.
[5] ——, "Service burstiness and dynamic burstiness measures: A framework," *J. High Speed Networks*, vol. 1, no. 2, pp. 105–127, 1992.
[6] R. L. Cruz and H.-N. Liu, "End-to-end queueing delay in ATM networks," *J. High Speed Networks*, vol. 3, no. 4, 1994.
[7] R. L. Cruz, "Quality of service guarantees in virtual circuit switched networks," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 1048–1056, 1995.
[8] ——, "SCED$^+$: Efficient management of quality of service guarantees," in *Proc. IEEE INFOCOM*, 1998, pp. 625–642.
[9] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *Proc. ACM SIGCOMM*, 1989, pp. 1–12.
[10] 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.
[11] V. Firoiu, J. Kurose, and D. Towsley, "Efficient admission control of piecewise linear traffic envelopes at EDF schedulers," *IEEE/ACM Trans. Networking*, vol. 6, pp. 558–570, Oct. 1998.
[12] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Trans. Networking*, vol. 3, pp. 365–386, Aug. 1995.
[13] L. Georgiadis, R. Guérin, and A. Parekh, "Optimal multiplexing on a single link: Delay and buffer requirements," *IEEE Trans. Inform. Theory*, vol. 43, pp. 1518–1535, Sept. 1997.
[14] L. Georgiadis, R. Guérin, V. Peris, and K. Sivarajan, "Efficient network QoS provisioning based on per node traffic shaping," *IEEE/ACM Trans. Networking*, vol. 4, pp. 482–501, Aug. 1996.
[15] S. J. Golestani, "Congestion-free communication in high-speed packet networks," *IEEE Trans. Commun.*, vol. 39, pp. 1802–1812, Dec. 1991.
[16] P. Goyal, S. Lam, and H. Vin, "Determining end-to-end delay bounds in heterogeneous networks," in *Proc. 5th Int. Workshop Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, 1995, pp. 287–298.
[17] A. Hung and G. Kesidis, "Bandwidth scheduling for wide-area ATM networks using virtual finishing times," *IEEE/ACM Trans. Networking*, vol. 4, pp. 49–54, Feb. 1996.
[18] J.-Y. Le Boudec, "Application of network calculus to guaranteed service networks," *IEEE Trans. Inform. Theory*, vol. 44, pp. 1087–1096, May 1998.
[19] J. Liebeherr, D. E. Wrege, and D. Ferrari, "Exact admission control for networks with a bounded delay service," *IEEE/ACM Trans. Networking*, vol. 4, pp. 885–901, Dec. 1996.
[20] 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, June 1993.
[21] D. Saha, S. Mukherjee, and S. K. Tripathi, "Multirate scheduling of VBR video traffic in ATM networks," *IEEE J. Select. Areas Commun.*, vol. 15, no. 6, pp. 1132–1147, Aug. 1997.
[22] H. Sariowan, "A service curve approach to performance guarantees in integrated-service networks," Ph.D. dissertation, Dept. Elect. Comput. Eng., Univ. California at San Diego, 1996.
[23] D. Stiliadis and A. Varma, "Latency-Rate Servers: A general model for analysis of traffic scheduling algorithms," *IEEE/ACM Trans. Networking*, vol. 6, pp. 611–624, Oct. 1998.
[24] G. Xie and S. Lam, "Delay guarantee of virtual clock server," *IEEE/ACM Trans. Networking*, vol. 3, pp. 683–689, Dec. 1995.
[25] Q. Z. Zheng and K. G. Shin, "On the ability of establishing real-time channels in point-to-point packet switching networks," *IEEE Trans. Commun.*, vol. 42, pp. 1096–1105, Mar. 1994.
[26] H. Zhang and D. Ferrari, "Rate-controlled service disciplines," *J. High Speed Networks*, vol. 3, no. 4, 1994, pp. 389–412.
[27] L. Zhang, "VirtualClock: A new traffic control algorithm for packet-switched networks," *ACM Trans. Comput. Syst.*, vol. 9, no. 2, pp. 101–124, May 1991.

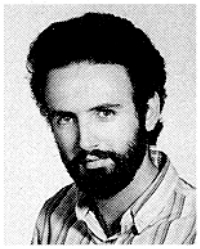


Hanrijanto Sariowan (S'91–M'95) received the B.S. (*cum laude*) degree from the Sepuluh-Nopember Institute of Technology, Surabaya, Indonesia, in 1987, the M.S. degree from Columbia University, NY, in 1990, and the Ph.D. degree from the University of California at San Diego in 1996, all in electrical engineering.

From 1996 to 1998, he was with Linkabit Wireless (previously Titan Information Systems), San Diego, working on the architecture design of a very small aperture terminal network. Since 1998, he has been with Tiernan Communications, San Diego, working on networking, multiplexing, and data broadcasting for high-definition television systems. His research interests include quality-of-service provisioning for multimedia broadband networks, wireless networks, and protocol design.

Rene L. Cruz (S'80–M'84–SM'90) received the B.S. and Ph.D. degrees in electrical engineering from the University of Illinois, Urbana, and the S.M. degree in electrical engineering from the Massachusetts Institute of Technology, Cambridge.

Currently, he is a Professor in the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla. His present research interests include resource allocation and scheduling in integrated broadband networks, wireless networks, and high-speed switching systems.



George C. Polyzos (M'88) received the Dipl. E.E. degree from the NTUA, Athens, Greece, and the M.A.Sc. degree in electrical engineering and the Ph.D. degree in computer science, both from the University of Toronto, Toronto, Canada.

He joined the faculty of the Department of Computer Science and Engineering, University of California at San Diego, La Jolla, in July 1988. His research interests include communications networks and multimedia systems design and performance evaluation.