# Achieving High Utilization in Guaranteed Services Networks Using Early-Deadline-First Scheduling

Fabio M. Chiussi and Vijay Sivaraman*

High-Speed Networks Research Department

Bell Laboratories, Lucent Technologies

101 Crawfords Corner Road, Holmdel, NJ 07733, USA

*Abstract* — Among packet-scheduling disciplines for providing end-to-end Quality-of-Service (QoS) guarantees to different applications, including real-time services, two classes of algorithms have received particular attention: those based on Generalized Processor Sharing (GPS) and those based on Earliest Deadline First (EDF). One reason for the popularity of GPS-based schemes is that their powerful properties can translate in simple Call Admission Control (CAC) procedures. The intense research on GPS has also resulted in very efficient implementation techniques, which have made the cost of these schedulers very affordable. The EDF discipline in conjunction with per-node traffic shaping (which we refer to as Rate-Controlled EDF (RC-EDF)) has also been recently proposed as a candidate for end-to-end QoS provisioning. However, an appropriate framework for CAC with RC-EDF has not been developed, nor the possible advantages of using RC-EDF in place of GPS have been properly characterized. Furthermore, the implementation complexity of an RC-EDF server is potentially very high, and no technique to reduce cost has been proposed.

In this paper, we first formulate an end-to-end CAC framework for RC-EDF that can be implemented in practice. Then, using this framework, we numerically compare the schedulable regions of RC-EDF and GPS and show that, when the traffic mix in the network consists of connections with both stringent and loose delay requirements, RC-EDF can substantialy outperform GPS in the number of admitted connections, and thus achieve much higher network utilization. Finally, we propose a technique to substantially reduce the implementation complexity of a RC-EDF server.

## 1 Introduction

The provision of Quality-of-Service (QoS) guarantees such as bandwidth, delay, jitter, and cell loss to applications of widely different characteristics is a primary objective in emerging broadband packet-switched networks. In such networks, packet-scheduling disciplines are necessary to satisfy the QoS requirements of delay-sensitive applications, and ensure that real-time traffic and best-effort traffic can coexist on the same network infrastructure. Among the scheduling algorithms that have been proposed in literature (for a survey, see [1, 2]), two classes of schemes have become popular: those based on Generalized Processor Sharing (GPS), and those based on Earliest Deadline First (EDF).

GPS [3, 4] is an idealized fluid discipline with a number of very desirable properties, including the provision of minimum bandwidth guarantees to each connection regardless of the behavior of other connections (perfect isolation), and the provision of deterministic, easily-computable end-to-end delay bounds to connections whose traffic is leaky-bucket constrained. Because of its powerful properties, GPS has become the reference for an entire class of GPS-related packet-scheduling disciplines, which has been extensively studied in literature. This interest in GPS has recently produced a number of implementation techniques which have made the cost of GPS-related schedulers very affordable [5, 6, 7, 8]; the relatively low cost explains the widespread use of these schedulers in the generation of packet switches that is currently reaching the market. Another key reason for this popularity is that GPS scheduling can be coupled with a very simple Call Admission Control (CAC) framework [9], by establishing a one-to-one relation between end-to-end delay requirements and bandwidth guarantees at the network nodes. Although in some situations this may result in a substantial portion of the available bandwidth being wasted (for a discussion, see Section 2.1 below), this simple CAC is arguably the only available framework ready for immediate deployment in practical networks for provisioning end-to-end delay guarantees.

EDF scheduling has been known for many years in the context of processor scheduling [10], and has been more recently proposed as a possible packet-scheduling discipline for broadband networks [11, 12]. In the case of a single node, EDF is known to be the optimal scheduling policy [13, 14] in terms of *schedulable region* for a set of connections with given delay requirements (for details see Section 2.2 below). However, since the computation of the schedulable region in EDF requires precise knowledge of the traffic at the input of a node, the use of EDF for end-to-end guarantees in a multi-node network is problematic. As a possible solution to extend EDF to the multi-node scenario, the work in [15, 16] has suggested the use of EDF in conjunction with per-node traffic shaping (in this paper, we refer to this combination as *Rate-Controlled EDF* (RC-EDF)). In [16], RC-EDF has been shown to at least match GPS performance.

A number of aspects related to RC-EDF have not been addressed in literature. First, to make RC-EDF usable in real networks, a practical end-to-end CAC framework needs to be developed; obviously, the goal is to design a framework more powerful than what possible in GPS. Second, once such framework is available, a characterization of RC-EDF and GPS in terms of schedulable regions must be performed to quantify the potential advantages of RC-EDF. Finally, RC-EDF scheduling can become a candidate as a practical scheduling discipline only if appropriate implementation techniques are devised to make the complexity of a RC-EDF server affordable.

In this paper, we address all these issues. We first formulate a CAC framework for RC-EDF that can be used in practice. Then, we use this framework and numerically compare RC-EDF and GPS in terms of schedulable regions. We show that, when the traffic contains connections with both stringent and loose delay requirements (as is the case most of the times in practice), RC-EDF can considerably outperform GPS by admitting many more connections and thus achieving higher network utilization. (In other scenarios, RC-EDF always at least matches the performance of GPS.) Finally, we propose a implementation technique for a RC-EDF shaper/scheduler that considerably reduces the hardware complexity of the server, and can be used in practical high-speed networks.

The rest of this paper is organized as follows. In Section 2, we review GPS and EDF to the extent that is relevant for this paper. In Section 3, we propose our end-to-end CAC framework for RC-EDF and discuss its implementation complexity. In Section 4, using this framework, we present numerical comparisons of RC-EDF and GPS. In Section 5, we describe our implementation technique for the implementation of a RC-EDF server. Finally, Section 6 offers some concluding remarks.

## 2 Background

### 2.1 Generalized Processor Sharing (GPS)

The GPS service discipline (on which all GPS-related schemes are based) has been described and analyzed in [4, 9]. A GPS server $k$ operating at rate $r^k$ and serving $N$ connections is characterized by $N$ positive real

*This author was with Bell Laboratories, Lucent Technologies, on leave from University of California, Los Angeles, when this work was performed.

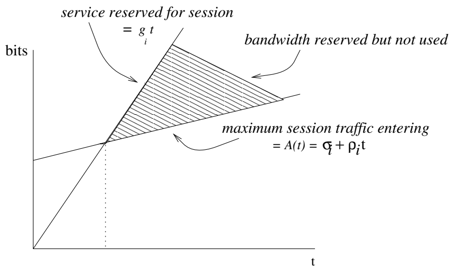


Figure 1: Wasted bandwidth in GPS.

numbers $\phi_1^k, \phi_2^k, \ldots, \phi_N^k$ such that for any connection $i$ continuously backlogged in the interval $(\tau, t]$,

$$\frac{S_i^k(\tau,t)}{S_j^k(\tau,t)} \geq \frac{\phi_i^k}{\phi_j^k}, \quad j = 1, 2, \ldots, N$$

where $S_i^k(\tau,t)$ is the amount of traffic of connection $i$ served by the server $k$ during the interval $(\tau, t]$. The rate $g_i^k = \phi_i^k r^k / \sum_{j=1}^N \phi_j^k$, is the minimum service rate that connection $i$ receives during any period when it is backlogged, *regardless* of the behavior of other connections (perfect isolation). The excess bandwidth available from connections not using their reserved service rate is distributed at each instant among all backlogged connections in proportion to their individual service rates.

One of the reasons for the popularity of fair-queueing scheduling discipline is that it is possible to couple GPS scheduling with CAC in a rather straightforward manner. This is achieved by assigning to a connection $i$ an *identical* weight (proportional to its rate) at each switch on its path. This restricted form of GPS[1] allocates to a connection a fixed fraction of the capacity of the outgoing link at each switch. The advantage of imposing this restriction is that the end-to-end delay bounds can be promptly computed for a $(\sigma_i, \rho_i)$ leaky bucket[2] constrained connection as [9]:

$$D_i = \frac{\sigma_i}{g_i} + \sum_{k=1}^{K_i} \left(\frac{L}{g_i} + \frac{L}{r^k}\right) \tag{1}$$

where $g_i$ is the rate allocated to connection $i$, $K_i$ is the number of switches in the connection's path, $r^k$ is the service rate at the $k$-th switch, and $L$ is the packet size.

In this way, CAC becomes very simple. For any new connection $j$ with a given delay requirement $D_j$, the CAC procedure computes the corresponding $g_j$ from eq. (1); then, at each switch along the path, it checks for $\sum g_i < r^k$, where $r^k$ is the capacity of the outgoing link at that switch and the summation is over all connections sharing that link, including the new connection $j$. If the condition is verified for all links along the path, the new connection $j$ is accepted, otherwise is rejected.

However, the simplicity of this framework obviously comes at a price. The problem is visually shown in Fig. 1. GPS might have to assign to a connection with bursty traffic a rate $g_i$ much higher than its average rate $\rho_i$, in order for the connection to be able to meet its delay requirements. This overallocation implies that some of the bandwidth will be wasted in the long run (this is represented by the shaded area in Fig. 1). Overall, the bandwidth $r^k - \sum \rho_i$ cannot be used, even though additional connections with larger end-to-end delays guarantees could have been accommodated.

[1] Historically, this form of GPS has been referred to as *Rate Proportional Processor Sharing* (RPPS) [9, 2], but the distinction is not carried over in most of the literature, where RPPS is still called GPS, probably because it is the only way that the GPS framework is used in practice. In this paper, we follow this common but looser usage, and continue to call it GPS.

[2] A connection $i$ is said to be $(\sigma_i, \rho_i)$ leaky bucket constrained if the amount of traffic from connection $i$ entering the network in any interval of length $t$ is bounded by $\sigma_i + \rho_i t$ [17].

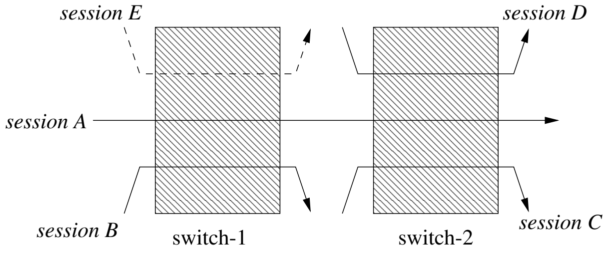


Figure 2: A simple two-switch configuration to illustrate complexity of exact delay computations in GPS. Connections $A$, $B$, $C$ and $D$ are present in the network when connection $E$ joins.

When $g_i$ is much larger than $\rho_i$ (as is often the case with bursty real-time traffic), this leads to low link utilizations.

A more exact delay computation technique, which would lead to a more accurate CAC, is described in [9]. This method allows different weights to be assigned to a connection at each of the switches on its path, and takes into account traffic interactions among connections sharing an output link; this requires the characterization of the traffic of a connection as it crosses successive switches on its path. The end-to-end delay computation for a connection consists of two steps: in the first step, a single-node analysis is used to construct the worst-case *service curve*, which represents the minimal amount of service received by the connection at that switch as a function of time. The worst-case connection backlog determined from this curve is used for characterizing the connection traffic at the next downstream switch. In the second step, the individual service curves for the connection at each switch are aggregated into a *universal service curve*, which represents the minimal amount of end-to-end service received by the connection as a function of time. The end-to-end delay bound is determined from the universal service curve.

Unfortunately, this more exact technique is not applicable in practice due to its computational complexity; in particular, a new connection entering the network could trigger a network-wide recomputation of the end-to-end delay bounds of all connections. As an illustrative example of the problem, consider a simple network with two GPS servers of rate 1 and four connections $A$, $B$, $C$ and $D$, shown in Fig. 2. The leaky-bucket parameters $(\sigma_i, \rho_i)$, end-to-end delay requirements $(\bar{D}_i)$ and GPS weights $(\phi_i)$ for the four connections are shown in Table 1. The end-to-end delay bounds are computed by constructing the universal service curves for each connection and are also show in Table 1 (column 6). All four connections are found to satisfy their delay constraints. When a new connection $E$ (with parameters as shown in Table 1) enters the network, a network-wide recomputation is required to determine the end-to-end delay bounds for each connection (these are shown in the last column in Table 1). Even more importantly, if connection $E$ is allowed to join, the delay requirements for connection $D$ are violated, even though its path is *disjoint* with that of connection $E$.

### 2.2 Earliest Deadline First (EDF)

The EDF scheduling discipline [10, 11, 12] works as follows: each connection $i$ at a switch $k$ is associated with a *local* delay deadline $d_i^k$; then, an incoming packet of connection $i$ arriving to the scheduler at time $t$ is stamped with a deadline $t + d_i^k$, and packets in the scheduler are served by increasing order of their deadline.

For a single switch, EDF is known to be the optimal scheduling policy [13]. Optimality is defined in terms of the *schedulable region* associated with the scheduling policy. Given $N$ connections with traffic envelopes $\bar{A}_i(t)$ $(i = 1, 2, \ldots, N)$ sharing an output link, and given a vector of delay bounds $\vec{D} = (d_1, d_2, \ldots d_N)$, where $d_i$ is an upper bound on the scheduling delay that packets of connection $i$ can tolerate, the schedulable region of a scheduling discipline $\pi$ is defined as the set of all vectors $\vec{D}$ that are schedulable under $\pi$. The authors in [13, 14] show that

| connection | $\sigma_i$ | $\rho_i$ | $\bar{D}_i$ | $\phi_i$ | delay before $E$ joins | delay after $E$ joins |
|---|---|---|---|---|---|---|
| A | 2 | 0.3 | 8.5 | 0.2 | 7.04 | 8.33 |
| B | 1 | 0.4 | 1.7 | 0.6 | 1.33 | 1.67 |
| C | 1 | 0.3 | 2.2 | 0.3 | 2.17 | 2.17 |
| D | 2.5 | 0.2 | 8.0 | 0.15 | 7.26 | 9.76 |
| E | 2 | 0.1 | 8.33 | 0.2 | - | 8.33 |

Table 1: Connection parameters for the scenario of Fig. 2.

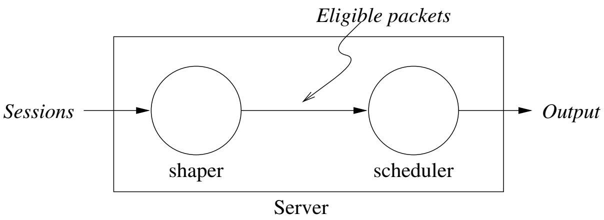


Figure 3: The RCS server.

EDF has the largest schedulable region of all scheduling disciplines, and its Non-Preemptive version (NPEDF) has the largest schedulable region of all the non-preemptive policies. The schedulable region of the NPEDF policy consists of those vectors which satisfy the following constraints [13, 14]:

$$\frac{L}{r} \leq d_1 \tag{2}$$

$$L + \sum_{i=1}^{N} \bar{A}_i(t - d_i) \leq rt, \quad \frac{L}{r} \leq t \leq d_N \tag{3}$$

$$\sum_{i=1}^{N} \bar{A}_i(t - d_i) \leq rt, \quad t \geq d_N \tag{4}$$

where $d_1 \leq d_2 \leq \ldots \leq d_N$, $L$ is the packet size (if the packet size is variable, $L$ is the maximum packet size), $r$ the link rate, and $\bar{A}_i(t) = 0$ for $t < 0$. Within a single node, once the traffic envelopes are known, a 100% link utilization can be achieved (at least in principle) with this characterization.

The difficulties arise in a multinode network where the traffic envelopes are no longer known at the inputs of the nodes inside the network, and the interactions that distort the traffic are not easily characterizable. This problem is not peculiar of EDF, but is common to any scheduling discipline (we have seen a similar situation for the exact delay computation in GPS in Section 2.1 above). As a general framework to handle the multinode problem, Zhang and Ferrari in [15] propose a class of schemes called *Rate-Controlled Service* (RCS) disciplines which reshape the traffic at each hop within the network. As schematically shown in Fig. 3, a RCS server has two components, a *shaper* which reshapes the traffic of each connection, and a *scheduler* which receives packets released by the shaper and schedules them according to a specific scheduling discipline, such as EDF. Georgiadis *et al.* in [16] build upon this model and derive expressions for the end-to-end delay bounds in terms of the shaper envelope and scheduling delay at each node. They also show the following useful properties of RCS:

- Identical shapers at each switch along the path of a connection $i$ (*i.e.*, shapers having identical shaper envelopes for connection $i$) produce end-to-end delays that are no worse than those produced by different shapers at each switch. Therefore, for any given connection, we can use identical shapers at each node. We denote this shaper envelope common to all shapers for connection $i$ as $\bar{E}_i(t)$.
- The end-to-end delay bound for connection $i$ is given by

$$\bar{D}_i = D(\bar{A}_i \| \bar{E}_i) + \sum_{k=1}^{K_i} d_i^k \tag{5}$$

where $\hat{D}_i = D(\bar{A}_i \| E_i)$ denotes the maximum shaper delay and $d_i^k$ is the bound on the scheduler delay for packets of connection $i$ at the $k$-th switch on its path. The maximum shaper delay is incurred only *once*, and is independent of the number of nodes on the path. The total scheduler delay is the sum of the individual scheduling delays $d_i^k$ at each node.

- When EDF scheduling is used together with per-node reshaping (we refer to this combination as RC-EDF), and the delay components in eq. (5) are properly chosen, the same delay bounds as GPS can be achieved.

What has not been shown in literature is whether or not RC-EDF can improve the performance of GPS in terms of network utilization, and more specifically how this can be achieved by devising a proper framework for CAC. Furthermore, the implementation of a RC-EDF server is potentially complex, and proper techniques have to be devised to make the cost of such a server affordable in practice. These issues shall be our focus in the remainder of this paper.

## 3 A Practical CAC Framework for RC-EDF

In this section, we to develop an end-to-end CAC framework for RC-EDF. In the next section, using this framework, we numerically compare the schedulable regions of RC-EDF and GPS in different traffic scenarios.

### 3.1 Framework

The CAC framework needs to characterize the available bandwidth at a switch accurately, and ensure that end-to-end performance constraints are met. We first look at how CAC works for an isolated EDF scheduler, and then proceed to the multinode case.

In a single switch, eq. (3) and eq. (4) immediately lead to a CAC scheme. With RC-EDF, given the traffic envelope $\bar{E}_i(t)$ (enforced by the shaper) and local delay bound $d_i$ for each of the connections being multiplexed on a link of the switch, we just combine the equations into:

$$L + \sum_{i=1}^{N} \bar{E}_i(t - d_i) \leq rt, \quad t \geq \frac{L}{r} \tag{6}$$

where $d_i \geq L/r,\ i = 1, 2, \ldots N$.

The above equation can be graphically interpreted to yield a simple single-switch CAC scheme. $\bar{E}_i(t - d_i)$ is the curve obtained by shifting the connection $i$ arrival envelope curve $\bar{E}_i(t)$ to the right by its delay bound $d_i$, and denotes the *minimal service* $\hat{S}_i(t)$ required by connection-$i$ in order to meet its local delay bound. The *aggregate service demand* of all connections at the scheduler is thus given by $\hat{S}(t) = L + \sum_{i=1}^{N} \hat{S}_i(t)$ (where the term $L$ accounts for the non-preemptive nature of the scheduler). If the aggregate service demand $\hat{S}(t)$ never exceeds the *server capacity* given by $R(t) = rt$ for $t \geq L/r$, the packets can be scheduled such that none misses its deadline. Fig. 4 illustrates the service capacity and

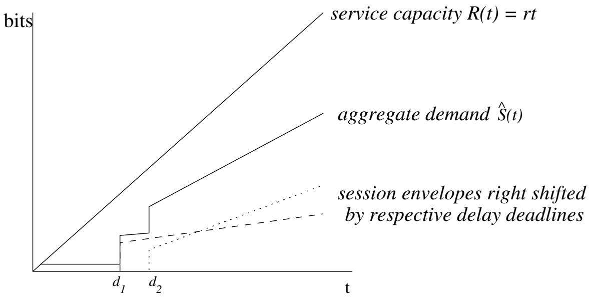


Figure 4: Schedulability test for single-node EDF.

service demand curves for a simple example with two leaky-bucket constrained connections. Since $S(t) = L + \bar{E}_1(t - d_1) + \bar{E}_2(t - d_2) \le R(t)$ for all $t \ge L/r$, the two calls can be admitted with guaranteed delay bounds $d_1$ and $d_2$.

The corresponding aggregate service demand $\hat{S}(t)$ in GPS would be a straight line passing through the origin with slope $\sum_i g_i$, where $g_i$ is the rate allocated to connection-$i$. By allowing a more complex form for the curve $\hat{S}(t)$ (for example piecewise linear if each of the envelope curves is piecewise linear), RC-EDF is able to characterize the available bandwidth more accurately, thereby providing the potential to have wider schedulable regions than GPS.

The extension of this scheme to the multinode case is as follows. For an incoming connection $i$ with traffic arrival envelope at the edge of the network $\bar{A}_i(t)$ and end-to-end delay requirement $d_i$, the end-to-end CAC algorithm performs the following steps to determine if the connection can be accommodated in the network:

1. It chooses an appropriate shaper with envelope $\bar{E}_i(t)$ for the connection, and computes the corresponding delay $D(\bar{A}_i \| \bar{E}_i)$ (the delay computation is described in [16]).
2. The $k$-th switch on the path is assigned a delay bound $d_i^k$ such that $D(\bar{A}_i \| \bar{E}_i) + \sum_k d_i^k = \bar{D}_i$. A single-node schedulability check according to the schedulability criteria of eq. (6) is performed (using envelope $\bar{E}_i(t)$ and delay bound $d_i^k$) at each switch on the path.
3. The connection is admitted only if every switch on the path can accommodate the connection.

The results in [16] can be used for choosing the shaper envelope and for splitting the total scheduling delay among the schedulers on the path. For leaky-bucket-constrained sources with traffic arrival envelope $\bar{A}_i(t) = \sigma_i + \rho_i t$, GPS performance can be matched by choosing shaper envelope $\bar{E}_i(t) = \min(L + g_i t, \sigma_i + \rho_i t)$ and assigning local delay bound $d_i^k = L/g_i + L/r^k$ to the $k$-th switch on the path, where $r^k$ is the link rate and $g_i$ the rate allocated to the connection in that switch.

In Section 4 below, we show numerically that, using this call admission strategy, RC-EDF can achieve higher network utilizations than GPS.

### 3.2 Implementation

A key issue that requires further discussion is how this proposed framework can be made practical. As explained above, the whole framework rests on checking the validity of equation (6), which is equivalent to checking that the aggregate service demand $\hat{S}(t)$ lies below the capacity curve $R(t) = rt$. Since $\hat{S}(t)$ is a piecewise linear function with $O(N)$ pieces, where $N$ is the number of connections sharing the link, this task could be quite complex, and rather impractical with the number of connections that are typically supported in current packet switches.

A solution to make the complexity independent of the number of connections sharing the link is to group the connections into a (relatively small) discrete set of *delay classes*. Such a restriction is certainly reasonable in practical switches (this same realistic assumption of discrete delay classes has been used with great success to reduce the implementation complexity of GPS-related schedulers [7, 6]). Let $C$ be the number of delay classes, and $\alpha_1, \alpha_2, \ldots, \alpha_C$ the corresponding delays. Then, the single-node call admission criteria can be restated as:

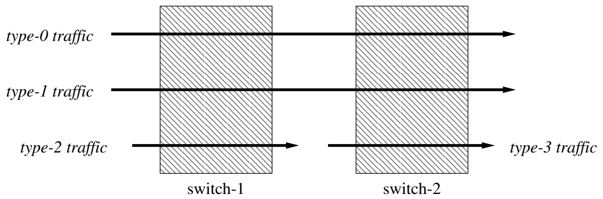


Figure 5: Two-switch network model.

$$L + \sum_{c=1}^{C} \hat{E}_c(t - \alpha_c) \quad \le \quad rt, \quad t = \alpha_1, \alpha_2, \ldots, \alpha_C \tag{7}$$

$$\sum_{c=1}^{C} \frac{\partial \hat{E}_c(t - \alpha_c)}{\partial t} \quad \le \quad r, \quad t = \alpha_1, \alpha_2, \ldots, \alpha_C \tag{8}$$

where $\hat{E}_c(t) = \sum_{i \in c} \bar{E}_i(t)$ is the aggregate envelope of the connections in class $c$ (we make the common assumption of concave, right-continuous, right-differentiable traffic envelope curves $\bar{E}_i(t)$).

Eq. (8) restricts the slope of the aggregate demand curve $\hat{S}(t)$ to be less than $r$ for each delay value $\alpha_c$. Eq. (8), coupled with the concavity property of the traffic envelope curves $\bar{E}_i(t)$, also implies that, once eq. (7) is verified for each delay value $\alpha_c$, it holds for values of $t$ between delay values. Furthermore, the $\hat{E}_c(t)$'s can be precomputed and stored in a $C \times C$ matrix, where the $(i, j)$-th entry stores the value of $\hat{E}_i(\alpha_j - \alpha_c)$. Similarly, the left-hand side in equations (7,8) can be precomputed for each $t = \alpha_1, \alpha_2, \ldots, \alpha_C$. Thus, the CAC procedure, which has to check these equations in order to admit or reject a new connection, is an $O(C)$ operation. Similarly, the process of updating the above variables when a connection joins or leaves is also an $O(C)$ operation. For typical values of $C$ (*e.g.,* 64 or 128), the CAC complexity is rather affordable and independent of the number of connections sharing the link.

## 4 Numerical Results

We compare the RC-EDF scheme with GPS scheduling using the simple two-switch network shown in Fig. 5. This simple network model suffices for our comparison purposes since it captures the interactions of different types of traffic traversing multiple switches and interacting with cross-traffic at each switch; a larger network would only introduce unnecessary complexity and confusion, without providing much additional insight.

In our network model, each connection in the network belongs to one of four types. Types 0 and 1 traverse both switches, type 2 models the cross-traffic at switch 1, and type 3 models the cross-traffic at switch 2. All connections of a type share identical traffic and performance parameters. The link speeds are fixed at 155 Mbps and the packet size at $L = 1000$ bits. We consider leaky-bucket constrained traffic sources since, as mentioned above, this is the traffic model that is most straightforward for the CAC in GPS, and therefore the easiest for comparison with RC-EDF. For RC-EDF, we choose the envelope of the shaper at each node as described in the previous section, *i.e.* $\bar{E}_i(t) = \min(L + g_i t, \sigma_i + \rho_i t)$.

We compare RC-EDF and GPS in the following way. For given input traffic descriptors for all types of traffic, we fix the end-to-end delay requirements and the number of connections for types 1, 2 and 3, and we compute the maximum number of type-0 connections that can be admitted in the network using each of the two schemes. (In other words, we characterize the number of type-0 connections, subject to type-1 background traffic through both switches, and type-2 and type-3 cross-traffic at each switch.) The authors in [16] have already shown that RC-EDF can match GPS performance; we carry the argument further and show that in some situations RC-EDF can accept *many more* connections than GPS. We plot the number of type-0 connections accepted into the network as a function of two parameters: the end-to-end delay bounds for type-0 connections and the number of type-1 connections.

| type | $n$ | $\sigma$(bits) | $\rho$(bps) | $D$(msec) |
|---|---|---|---|---|
| type-0 | computed | 16600 | 16600 | varied |
| type-1 | varied | 1000000 | 1500000 | 120 |
| type-2 | 140 | 18000 | 8000 | 240 |
| type-3 | 85 | 8000 | 4000 | 600 |

Table 2: Connection parameters for Scenario 1.

For the first set of results (Scenario 1), the traffic parameters are shown in Table 2. In this scenario, we have chosen audio-like parameters for type-0 traffic and video-like parameters for type-1 traffic. As usual, the video connections are very demanding in terms of bandwidth and account for most of the traffic at each switch. Since the video connections flow across both switches, the two switches are more or less equally loaded. Accordingly, in the case of RC-EDF, we split the total scheduling delay equally between the two switches. The plot of the schedulable regions is shown in Fig. 6; the lower surface represents the number of type-0 calls accepted by GPS while the upper surface is the number accepted by RC-EDF. For a fixed volume of background type-1 traffic, both GPS and RC-EDF admit about the same number of type-0 connections with low end-to-end delay bounds. However, in the case of type-0 connections with delay constraints above a certain threshold (approximately 1.2s in this case), RC-EDF is able to accept many more connections than GPS (orders of magnitude more in some cases). As mentioned above, the reason why RC-EDF is able to outperform GPS so dramatically in this situation is that GPS allocates to connections with low delay bounds a rate that is much higher than their average rate, and the excess allocated bandwidth goes wasted. On the contrary, by decoupling the rate and delay allocations, RC-EDF is able to recover this bandwidth by squeezing in connections with loose delay bounds.

For the second set of results (Scenario 2), we interchange the parameters of traffic types 1 and 3. In this way, the video streams are made to flow through switch 2 only, making it more heavily loaded than switch 1. For RC-EDF, we first consider the case where the scheduling delay is still split 50/50 between the two switches along the path. The corresponding GPS and RC-EDF plots are shown in Fig. 7, and are identical to the ones of Fig. 6. This is expected, since the load on the highly-loaded switch determines the number of calls that can be accepted, independently of the load on the other switches. However, if the scheduling delay with RC-EDF can be split unequally along the switches on the path to take into account the uneven loading of the switches, an improvement in performance is achieved. An example of this improvement is shown in Fig. 8, where we plot the schedulable region for RC-EDF with an 80/20 split in scheduling delay in the same scenario (the plot for the 50/50 split is also shown in the figure for comparison purpose). The plot shows that when the load on both switches is roughly identical, the uneven split of the delay leads to worse performance than the even split, but as switch 2 becomes more heavily loaded than switch 1 (which is the case as the number of type-1 connections increases), it is advantageous to allocate a larger fraction of the delay to the heavily loaded switch. The improvement, however, is rather limited, since the end-to-end delays are dominated by the shaper delays, and scheduling delays are generally quite low.

The above plots show that, as expected, RC-EDF can significantly outperform GPS when handling connections with high end-to-end delay

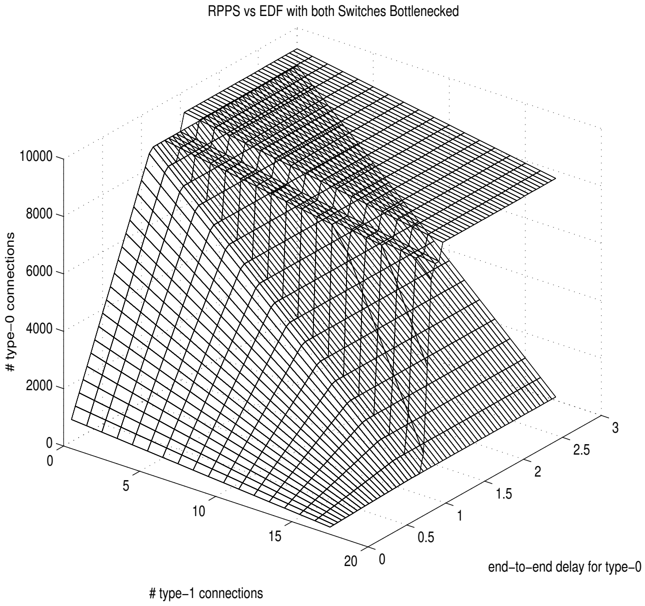


Figure 6: Scenario 1, GPS vs. RC-EDF (similar load on both switches).

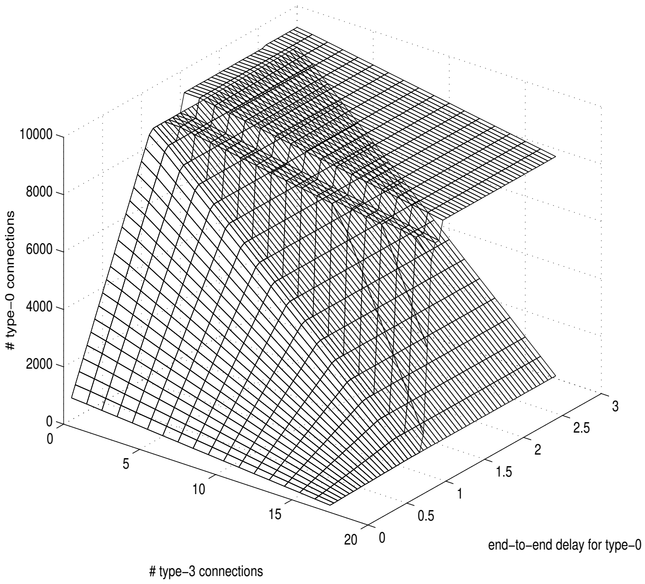


Figure 7: Scenario 2, GPS vs RC-EDF (higher load on switch 2).

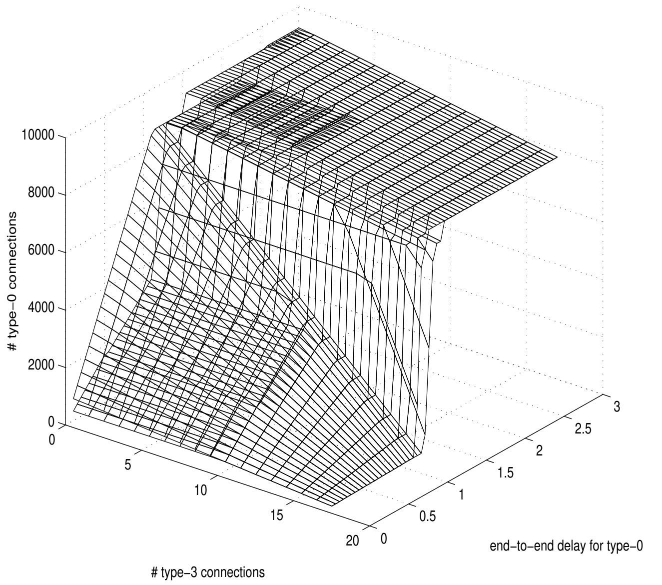


Figure 8: Scenario 2, RC-EDF with 50/50 delay split vs RC-EDF with 80/20 delay split (higher load on switch 2).

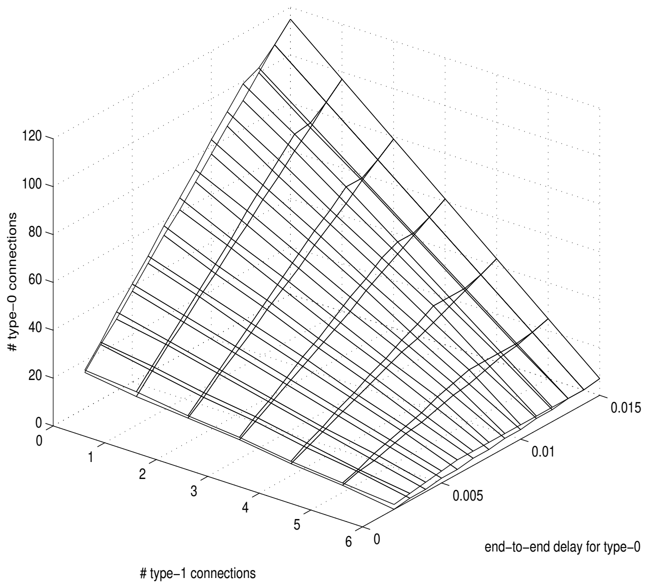


Figure 9: Scenario 3, GPS vs RC-EDF for low delay bounds.

| type | $n$ | $\sigma$(bits) | $\rho$(bps) | $D$(msec) |
|---|---|---|---|---|
| type-0 | computed | 16600 | 16600 | varied |
| type-1 | varied | 2700000 | 1500000 | 120 |
| type-2 | 140 | 18000 | 8000 | 240 |
| type-3 | 85 | 8000 | 4000 | 600 |

Table 3: Connection parameters for low-delay region scenario.

requirements. For low delay requirements there is no perceptible difference, since RC-EDF needs a longer time scale to capitalize on the bandwidth that goes wasted in GPS. Fig. 9 focuses on the low-delay region in more detail. For this plot (Scenario 3), to make the difference between the two schemes more prominent, we have reduced the link rates to 64 Mbps and used the connection parameters of Table 3. We observe that even in this region RC-EDF does accept more connections than GPS, though the difference is very small. This is the result of the optimality of EDF at a single switch, and with the help of reshaping at each node this difference carries on also to the multinode setting.

In summary, from the numerical comparisons presented in this section, we conclude that:

1. Due to its tighter characterization of the traffic, RC-EDF can accommodate orders of magnitude more connections with loose end-to-end delay bounds than GPS.
2. If the switches are unevenly loaded, an unequal split of the end-to-end delay among the switches can further increase the number of connections accepted by RC-EDF. However, the improvement is limited when the delays are dominated by the shaper delays.
3. Because of the optimality of single-node EDF, RC-EDF can accept more connections than GPS even for connections with low end-to-end delay constraints. However, this difference is marginal.

## 5 Implementation of a RC-EDF Server

A potentially serious problem is that the implementation of an RC-EDF server, which consists of a traffic shaper and an EDF scheduler, can be quite complex. In this section, we propose a technique to reduce the complexity of the server. Similarly to what we have done in Section 3.2 above for the call admission implementation, the idea here is that the server only supports a discrete set of $C$ delay classes. (Again, the assumption of discrete classes has been used in [7, 8] for the implementation of GPS-related schedulers; however, the application of a similar idea to RC-EDF is far from straightforward, and requires a complete redesign of the server.)

The overall architecture of the server is schematically shown in Fig. 10. Both shaper and scheduler are implemented using FIFO queues. The following discussion assumes fixed packet lengths, as is the case in ATM. Most of the complexity resides in the implementation of the shaper, which is described first.

### 5.1 The Shaper

In the shaper, one scheduling structure per delay class (referred to as *delay-class shaper*) is provided. Each delay-class shaper shapes all the connections in the corresponding delay class, and supports a discrete number $G$ of *shaping rates* $r_1, r_2, \ldots, r_G$. The total rate of each delay-class shaper is equal to the link rate (this is because eq. (8) restricts the sum of the rates of the connections in a delay class to be less than $r$). In each delay-class shaper, each shaping rate is associated with a corresponding *shaping FIFO queue*; connections that need to be shaped at a certain rate $r_i$ are queued in the correponding shaping FIFO queue and scheduling is only performed among connections at the head of their respective shaping FIFO queues. Each shaping FIFO queue is associated with a *queue timestamp* $F_i$ and an aggregate service rate $n_i r_i$, where $n_i$ is the number of connections currently queued in the shaping FIFO queue.

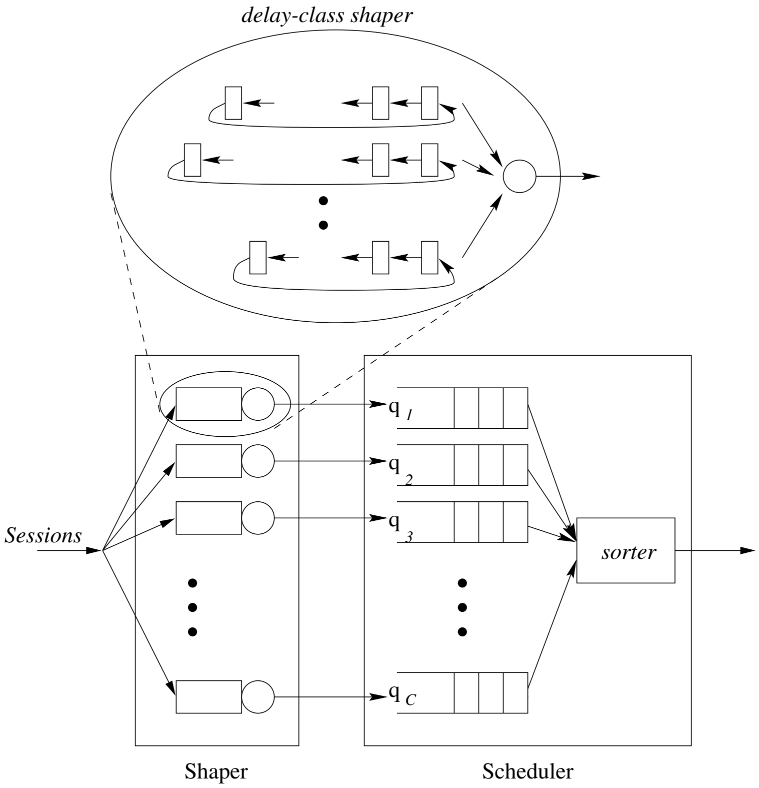


Figure 10: Implementation of the shaper and the scheduler.

As mentioned above, each connection is shaped according to a piecewise linear shaping envelope, which is the combination of one or more leaky-bucket envelopes, each with a corresponding shaping rate. When a packet of a connection arrives at the server, a leaky-bucket computation is used to determine the shaping rate $r_i$ desired by the packet (we refer to the associated leaky bucket as the *dominant leaky bucket* for that packet) Then, if the connection was not previously backlogged, it is queued in the corresponding shaping FIFO queue.

Scheduling among the shaping FIFO queues is performed at each time slot by selecting the connection at the head of the shaping FIFO queue having the minimum *eligible* queue timestamp (an eligible queue timestamp is one whose value is not greater than the current time). Every time a shaping FIFO queue is served, its corresponding timestamp is incremented by $L/n_ir_i$. If the connection that has been served remains backlogged, and requires the same service rate $r_i$, it is reinserted at the tail of the shaping FIFO queue. An other way to see this delay-class shaper is as a hierarchical scheduling structure, where at the lower level of the hierarchy the connections within each shaping FIFO queue are served in a round-robin fashion, and at the top level of the hierarchy the shaping FIFO queues are shaped according to their aggregate rate.

Due to the fact that the shaping envelopes for the connections are piecewise linear functions, the desired shaping rate may change while a connection is backlogged. Consequently, a connection that is queued in a given shaping FIFO queue may, once served, have to be enqueued in a different shaping FIFO queue (we refer to such an occurrence as a *rate jump*).

As we quantify in the sequel, such a rate jump may introduce a delay degradation for the connection. Furthermore, as the system progresses, the number of connections $n_i$ queued in shaping FIFO queue $i$ may increase or decrease as new connections become backlogged, served connections become idle, and connections require to change their rate. If the aggregate rate $n_ir_i$ of the shaping FIFO queue keeps changing indiscriminately as connections enter or leave the FIFO queue, the actual service rate experienced by the connections in the FIFO queue may not correspond to the desired rate $r_i$, and further delay degradation is introduced.

In order to minimize these degradations, we introduce the concept of a *pivot connection*, which is a "pseudo connection" that is always present in each of the shaping FIFO queues. The role of the pivot connection is to act as a reference that appears at regular intervals of time at the head of each FIFO queue; new connections are always inserted in the FIFO queue immediately behind the pivot connection, and updates of the aggregate rate $n_ir_i$ are performed only when the pivot connection is at the head of the FIFO queue.

To properly cope with connections changing their desired shaping rates, we perform a rate jump only *from a higher rate to a lower rate*. If a packet for a connection arrives at the server and requests a shaping rate that is higher than the shaping rate requested by the previous packet for that connection, the connection is considered newly backlogged, and immediately queued in the shaping FIFO queue corresponding to the new rate. This implies that, at any given time, a connection may be waiting in *more that one* shaping FIFO queue.

The rationale behind this idea can be understood by considering again the piecewise-linear shaping envelope for a connection, which corresponds to an ideal system of $n$ leaky buckets connected in series [18], with parameters $(\sigma_1, \rho_1), (\sigma_2, \rho_2), \ldots, (\sigma_n, \rho_n)$ (it has been shown that the order of interconnection of the leaky buckets is immaterial, therefore without loss of generality we assume $\rho_1 < \rho_2 < \ldots < \rho_n$). Before proceeding with this discussion, we recall that in such system, the delay $d^j$ incurred by an arriving packet $j$ is given by:

$$d^j = \max_{k=1,\ldots,n} \{d_k^j\} \tag{9}$$

where $d_k^j$ denotes the delay incurred in the isolated $k$-th leaky bucket $(\sigma_k, \rho_k)$; the leaky bucket which maximizes the right side of eq. (9) is the dominant leaky bucket for packet $j$ (we denote the corresponding value of $k$ as $k_{\max}^j$). We define a *busy period* for the system of multiple leaky buckets as a maximal interval of time during which the system is non-empty, *i.e.,* there is at least one leaky bucket holding packets waiting to be released.

It is fairly straightforward to prove the following theorem.

**Theorem 1:** For a connection shaped by $n$ leaky buckets $(\sigma_1, \rho_1), (\sigma_2, \rho_2), \ldots (\sigma_n, \rho_n)$ connected in series where $\rho_1 < \rho_2 < \ldots < \rho_n$, the dominant leaky buckets for the $i$-th and $j$-th packets within a busy period, where $i < j$, satisfy $k_{\max}^i \geq k_{\max}^j$.

**Proof:** Consider packets $i$ and $(i+1)$ with arrival times $s^i$ and $s^{i+1}$ within the busy period. By definition of $k_{\max}^i$:

$$d_{k_{\max}^i}^i > d_k^i \quad k = k_{\max}^i + 1, \ldots, n \tag{10}$$

Moreover, since packets $i$ and $(i+1)$ are in the same busy period:

$$s^{i+1} < s^i + d_{k_{\max}^i}^i \tag{11}$$

The release time of packet $(i+1)$ by leaky bucket $k$ ($k = k_{\max}^i + 1, \ldots, n$) is bounded by $\max\{s^{i+1}, s^i + d_k^i\} + L/\rho_k$, which from eq. (11) is no greater than $\max\{s^i + d_{k_{\max}^i}^i, s^i + d_k^i\} + L/\rho_k$; from eq. (10), this is equal to $s^i + d_{k_{\max}^i}^i + L/\rho_k$. Since $\rho_k > \rho_{k_{\max}^i}$ for $k = k_{\max}^i + 1, \ldots, n$, this quantity is lower than $s^i + d_{k_{\max}^i}^i + L/\rho_{k_{\max}^i}$, which corresponds to the release time by leaky bucket $k_{\max}^i$. Therefore, the leaky bucket with index $k_{\max}^i$ delays packet $(i+1)$ more than the leaky buckets with larger index, and thus $k_{\max}^{i+1} \leq k_{\max}^i$. Transitivity extends this relation to the $j$-th packet. $\triangle$

The above theorem implies that within a busy period of the system of multiple leaky buckets, the shaping rate for the connection is a *non-increasing* function of time. In other words, in an ideal shaper, a connection would request rate jumps only from higher to lower rates during a busy period of the shaper. The connection would increase its desired shaping rate only at the beginning of a new busy period (it should be clear that at the beginning of a busy period, a connection starts from one of the shaping rates, depending on its shaping rate at the end of the previous busy period, and on how much time has passed since the end of the previous busy period). In such an ideal shaper, the connection would always be queued in only one shaping FIFO queue at any time.

However, in our implementation, the shaper is not ideal (an ideal shaper would immediately serve all eligible connections, while in our implementation we only serve a connection at a time) and a connection may have to wait in a shaping FIFO queue longer than it would in an ideal shaper (this additional time corresponds to the delay degradation that we derive below). For this reason, a new busy period may start while the connection is still queued in the shaping FIFO queue corresponding to the rate at the end of the previous busy period (loosely, we could say that in our structure the busy periods may overlap). We show below that the number of queues in which the connection may be waiting simultaneously is bounded, and in general quite small.

We now compute the maximum delay degradation that a connection can experience in our shaping structure, *i.e.,* the maximum difference between the time when a packet is released by our implementation and the time when that packet would be released by an ideal shaper. Such degradation for a packet $p$ belonging to connection $i$ is given by the sum of two components:

1. $T_s^p - \delta^p$, where $T_s^p$ is the timestamp of the corresponding shaping FIFO queue $s$ when $p$ is the head-of-line packet for connection $i$ and connection $i$ is at the head of shaping FIFO queue $s$, and $\delta^p$ is the release time of $p$ by an ideal shaper for connection $i$. This quantity is bounded as follows. For a newly-backlogged connection entering shaping FIFO queue $s$ at time $t_s$, the queue timestamp $T_s$ when that connection reaches the head of the queue is bounded by $t_s + L/r_s$. This degradation of $L/r_s$ with respect to the ideal scheduler remains constant while the connection stays at the same shaping rate $r_s$. By a similar argument, it can be shown that when the connection jumps to a new shaping rate $r_{s'}$, an additional worst-case degradation of $L/r_{s'}$ is incurred. From Theorem 1, we know that within a busy period the connection can never jump back to a higher shaping rate. Thus, $T_s^p - \delta^p \leq \sum_n L/\rho_i$, where $\rho_i$, $i = 1, 2, \ldots, n$ correspond to the leaky bucket shaping rates for the connection.

2. $\tau^p - T_s^p$, where $\tau^p$ denotes the actual release time of the packet in our implementation. This quantity is bounded as follows. Consider the time interval $[x, x+t]$, where $x$ marks the start of a busy period for the delay-class shaper (a busy period for the delay-class shaper is a maximal interval of time during which a packet is released every slot). The maximum number of times that the timestamp value for shaping FIFO queue $s$ can change within this interval, such that the value does not exceed $x + t$, is equal to $\lceil \frac{t}{L/n_s r_s} \rceil$, since the timestamp increases at each update by $L/n_s r_s$. Summing over all the shaping FIFO queues, this yields $\sum_s \lceil \frac{t}{L/n_s r_s} \rceil$, which is bounded by $\sum_s (1 + \frac{t n_s r_s}{L})$; in turn, this is no greater than $G + \frac{tr}{L}$, since eq. (8) restricts the sum of shaping rates to be no more than $r$. This implies that the total number of eligible packets in $[x, x+t]$ is no more than $G + \frac{t}{L/r}$. Since $[x, x+t]$ constitutes a busy period for the delay class shaper, a packet is released every $L/r$ units of time, and therefore by time $x + t + GL/r$, all packets eligible for service in the interval $[x, x+t]$ have been released. Thus, $\tau^p - T_s^p \leq GL/r$.

Adding these bounds on the two delay components yields an upper bound on the delay degradation equal to $GL/r + \sum_{i=1}^n L/\rho_i$.

Having computed the delay degradation with respect to an ideal shaper, we can easily derive that, in our implementation, the maximum number of queues in which a connection may be simultaneously waiting as a result of the overlap of busy periods is equal to $n + \rho_k G/r$, where $n$ is the number of shaping rates corresponding to the traffic envelope for the connection.

This can be seen by focusing on one busy period for the connection in the shaper. Let $F_b$ denote the time at which the busy period ends in the ideal shaper and $F_b + \Delta$ the corresponding time in our implementation. Also let $\rho_k$ denote the rate at which the last packet in the busy period was served. The bound on the delay degradation derived above implies that $\Delta \leq GL/r + \sum_{i=1}^k L/\rho_i$. Also, at time $F_b$, the $k$-th leaky bucket is out of tokens and hence can release no more than $\frac{\Delta}{L/\rho_k}$ packets in the next $\Delta$ units of time. Consequently, no more than $\rho_k(\sum_{i=1}^k 1/\rho_i + G/r) < n + \rho_k G/r$ packets can be released by the ideal shaper within this interval of time. In other words, the degradation due to our implementation may cause at most $n + \rho_k G/r$ busy periods to overlap.

### 5.2 The Scheduler

The scheduler supports the same delay classes used by the call admission algorithm, *i.e.,* $C$ delay classes with associated delays $\alpha_1, \alpha_2, \ldots, \alpha_C$. The idea is that delay class $c$ is associated with a *delay FIFO queue* $q_c$, and scheduling is only performed among packets at the head of their respective delay FIFO queues. If connection $i$ belongs to delay class $j$ (*i.e.,* has a local scheduler delay bound equal to $\alpha_j$), then every time a packet of connection $i$ enters the scheduler (coming from the shaper) is assigned a timestamp $\alpha_j$ units ahead, and linked at the tail of delay FIFO queue $q_j$. Each delay queue is served as a FIFO because all packets in the queue have identical delay constraints. At each decision instant, the scheduler selects the packet with the minimum timestamp among the packets at the head of each delay FIFO queue. Since the number of queues $C$ is small (*e.g.*, 64), the complexity of the sorting operation is also relatively small, and is independent of the number of connections sharing the link.

## 6 Conclusions

EDF scheduling with per-node traffic shaping (which we have called RC-EDF) has been proposed as a solution to use EDF in a multi-node network [16]. In this paper, we have addressed a number of issues to make RC-EDF scheduling a practical candidate for deployment in broadband networks. We have first formulated a CAC framework for RC-EDF that can be implemented in practice and is more powerful than what achievable in GPS. Then, we have used this framework to numerically compare RC-EDF with GPS. We have shown that, when the traffic contains connections with a wide range of delay requirements, RC-EDF can admit orders of magnitude more connections than GPS, and thus achieve considerably higher network utilization. Finally, we have introduced a low-complexity technique to make a RC-EDF server implementable in high-speed networks.

## References


[1] S. Keshav. *An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network*. Addison-Wesley, Ithaca, New York, 1996.

[2] H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks," *Proceedings of the IEEE*, pp. 1374–1396, Oct. 1995.

[3] A. Demers and S. Keshav and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," *Proc. ACM SIGCOMM '89*, pp. 3–12, 1989.

[4] 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*, pp. 344–357, June 1993.

[5] J. L. Rexford, A. G. Greenberg and F. G. Bonomi, "Hardware-Efficient Fair Queueing Architectures for High-Speed networks," *Proc. INFOCOM '96*, pp. 638–646, March 1996.

[6] F. M. Chiussi, A. Francini and J. G. Kneuer, "Implementing Fair Queueing in ATM Switches – Part 2: The Logarithmic Calendar Queue," *Proc. GLOBECOM '97*, pp. 519–525, Nov. 1997.

[7] J. C. R. Bennett, D. C. Stephens and H. Zhang, "High Speed, Scalable, and Accurate Implementation of Fair Queueing Algorithms in ATM Networks," *Proc. ICNP '97*, pp. 7–14, Oct. 1997.

[8] F. M. Chiussi and A. Francini, "Implementing Fair Queueing in ATM Switches: The Discrete-Rate Approach," *Proc. INFOCOM '98*, March 1998.

[9] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple-Node Case," *IEEE/ACM Trans. Networking*, pp. 137–150, April 1994.

[10] C. L. Liu and J. W. Wayland, "Scheduling algorithms for multiprogramming in a hard real time environment," *Journal of ACM*, pp. 46–61, Jan. 1973.

[11] D. Ferrari and D. Verma, "A Scheme for Real-Time Channel Establishment in Wide-Area Networks," *IEEE Jour. Sel. Areas Commun.*, pp. 368–379, April 1990.

[12] D. Verma, H. Zhang, D. Ferrari, "Guaranteeing Delay Jitter Bounds in Packet Switching Networks," *Proc. TRICOMM '91*, pp. 35–46, Chapel Hill, NC, Oct. 1991.

[13] L. Georgiadis, R. Guerin, and A. Parekh, "Optimal Multiplexing on a Single Link: Delay and Buffer Requirements," *RC 19711 (97393)*, IBM T. J. Watson Research Center, Aug. 1994.

[14] J. Liebeherr and D. Wrege and D. Ferrari, "Exact Admission Control for Networks with a Bounded Delay Service," *IEEE/ACM Trans. Networking*, pp. 885–901, Dec. 1996.

[15] H. Zhang and D. Ferrari, "Rate-Controlled Service Disciplines," *Jour. High Speed Networks*, pp. 389–412, 1994.

[16] L. Georgiadis, R. Guerin, V. Peris, and K. Sivarajan, "Efficient Network QoS Provisioning Based on per Node Traffic Shaping," *IEEE/ACM Trans. Networking*, pp. 482–501, Aug. 1996.

[17] R. Cruz, "A Calculus for Network Delay, Part I: Network Elements in Isolation," *IEEE Trans. Information Theory*, pp. 114–121, Jan. 1991.

[18] R. Cruz, "A Calculus for Network Delay, Part II: Network Analysis," *IEEE Trans. Information Theory*, pp. 121–141, Jan. 1991.