Appeared Infocom '94

Optimal Multiplexing on a Single Link: Delay and Buffer Requirements

Leonidas Georgiadis, Roch Guérin and Abhay Parekh
IBM T. J. Watson Research Center
P. O. Box 704
Yorktown Heights, NY 10598

Abstract

This paper is motivated by the need to support multiple service classes in fast packet-switched networks. We address the problem of characterizing and designing scheduling policies that are optimal in the sense of minimizing buffer and/or delay requirements under the assumption of commonly accepted traffic constraints. We investigate the buffer requirements under three typical memory allocation mechanisms, that represent trade-offs between efficiency and complexity. For classes with delay constraints we provide policies that are optimal in the sense of satisfying the constraints if they are satisfiable by any policy, and they also have low buffer requirements. We also address the issue of designing policies that satisfy delay constraints in a fair manner. We mainly concern ourselves with non-preemptive policies. One of the proposed policies is based on a class of non-preemptive policies that tracks preemptive policies. This class is introduced in this paper and may be of interest in other applications as well.

1 Introduction

Fast packet-switching technology is the basis for the next generation of integrated services networks, which hold the promise of effectively supporting a wide range of traffic classes with different performance requirements. As demonstrated by earlier work [11, 16, 19], the choice of scheduling policy at the output queues of the network switches can play an important role in controlling packet delay, as well as in determining how large buffers should be in order to limit packet loss. Yet what is lacking is a framework within which the "goodness" of various policies can be compared. Such comparisons need to account for how well these policies do in terms of *both* delay and buffer requirements. In this paper, we define a simple analytical model that permits meaningful comparisons, and that also allows the derivation of scheduling polices that are *optimal* in terms of delay and buffer requirements. In addition to considering delay and buffer requirements individually, we study the existence and characteristics of policies that are *jointly delay and buffer optimal*. Finally, we analyze the ability of scheduling policies to fairly allocate lateness when delay requirements cannot be met.

Our study is restricted to the case of a single link (multiplexer), and assumes *zero-loss* multiplexers, i.e., buffers are sized so that space is always available to store incoming data, provided the input traffic satisfies certain constraints. This assumption is probably not severe, given the requirements for low loss probabilities for most service classes.

Our focus is on non-preemptive policies, since preemptive polices are inappropriate in the packet-switching environment. Many of our results are based on Earliest Deadline First (EDF) policies. In addition to the traditional oblivious EDF (NPEDF), we consider a version that attempts to track the behavior of a Preemptive EDF (PEDF) scheduler. In considering buffer requirements we study three buffer structures that represent different trade-offs between efficiency and complexity: Flexible, Semi-flexible and Fixed. Our choice of buffer allocation structures brings out interesting relationships between delay and buffer requirements of scheduling policies.

Before proceeding with the body of the paper, we briefly review significant related prior work. Buffer-optimal policies under the fixed allocation method have been studied in [18, 5, 3, 9, 4], however, our results on semi-flexible allocation are new. All of the results relating buffer and delay optimality are new as well.

The merits of using schedulable regions to guarantee quality of service in networks was recognized in [14], and the problem of scheduling tasks has received

significant attention in the context of (real-time) computing systems, where important results on optimal scheduling policies and the associated schedulable region have been obtained. However many of these results assume more restrictive arrival functions than those employed here. The optimality of the PEDF for the class of preemptive policies was first shown in [15] for periodic arrivals; in [12, 13] the delay-optimality of NPEDF among the class of non-preemptive policies is established for periodic and so-called sporadic arrivals; the schedulable regions for NPEDF and PEDF have been derived in [20] for arrival streams characterized by a minimum inter-packet arrival time that is independent of packet size.

Tracking policies have been proposed and studied in the context of Generalized Processor Sharing in [16, 7], however, results from [16] have been extended here to include all tracking policies that obey a specific Ordering Property. While the optimality of PEDF for the criterion of minimizing the maximum lateness of packets was established in [8], we show here that with respect to this criterion, NPEDF is close to PEDF . Finally, we show that PEDF satisfies the stronger criterion of lexicographic optimality of packet lateness and that T(PEDF) is close to PEDF with respect to this criterion.

We note that space considerations have forced us to omit the proofs of our results—the interested reader is referred to [10].

2 Model, Definitions and Preliminary Results

Assume traffic flows into a multiplexer from K input links, where the speed of the i^{th} link is r_i and the flow on each input link is partitioned into discrete entities or packets. A packet may be infinitesimally small, but can be no larger than $L_{\max}$ bits. Arriving packets are stored in the memory of the multiplexer until they are transmitted on the output link, which is assumed to be of speed r. Note that it takes L/r time units to transmit a packet of length L. A packet becomes eligible for transmission only after its last bit has arrived at the multiplexer (no cut-through). Since there may be several eligible packets at any given time, the multiplexer has a *scheduler* which implements a *service policy*. This policy decides which of the eligible packets to transmit on the output link and then transmits this packet non-preemptively.

Let there be N classes and let $A_{ji}(\tau, t)$ be the number of class i bits that arrive on link j in the interval $[\tau, t)$. Also, let

$$A_i(\tau, t) = \sum_{j=1}^{K} A_{ji}(\tau, t)$$

be the number of class i bits that arrive in the interval $[\tau, t)$. For each i, set $A_i(\tau, t+\tau) = 0$ for $\tau < 0$. Finally, let there exist $\sigma_i, \rho_i \geq 0$ such that

$$A_i(\tau, t) \leq \sigma_i + \rho_i(t - \tau) \tag{1}$$

and

$$\sum_{i=1}^{N} \rho_i \leq r. \tag{2}$$

This model for incoming traffic is identical to the one proposed by Cruz [6], and is consistent with the constraints imposed by rate control algorithms that have been proposed in standard bodies [1].

It is important to note that the degree to which these conditions restrict the arrivals varies with the choice of the values of σ_i and ρ_i for each class. For example, they can be used to simply model the input links of the multiplexer. Let the speed of the i^{th} input link be r_i. Then setting, $N = K$, $\sigma_i = 0$, $\rho_i = r_i$ for $i = 1, 2, , .K$, each class is associated with a distinct input link, and the conditions (1) and (2) reduce to the natural condition:

$$\sum_{j=1}^{K} r_j \leq r. \tag{3}$$

More general cases with multiple connections of class i on each incoming link and each connection having constraints similar to those of (1) are also possible. This more general case can be readily handled by either increasing the number of input links, or by modifying the traffic model used for each traffic class to account for the possibility of multiple simultaneous arrivals from different links. Both approaches are straightforward but introduce significant notational complexity without adding much insight. Therefore, for purposes of simplicity we assume in the rest of the paper that each traffic class is identified with a given input link.

Next we introduce some notation. Let the scheduler implement policy π. We denote by $S_i^{\pi}(0, t)$ the number of class i bits served in the interval $[0, t)$. Let also $Q_i^{\pi}(t, \vec{\rho}, \vec{\sigma})$ be the number of class i bits stored at time t, when the input traffic rates and burst size vectors are $\vec{\rho} = \{\rho_1, \ldots, \rho_N\}$ and $\vec{\sigma} = \{\sigma_1, \ldots, \sigma_N\}$ respectively. We define

$$M_i^{\pi}(\vec{\rho}, \vec{\sigma}) = \sup_{t \geq 0} Q_i^{\pi}(t, \vec{\rho}, \vec{\sigma}), \tag{4}$$

We clarify some notation in (4): for fixed $\vec{\rho}, \vec{\sigma}$, the supremum is taken over all $t \geq 0$ and all sample paths

that are consistent with (1) and (2). The same notation will be used throughout this paper, unless specified otherwise. Define the delay of a packet to be the time it spends in the system, i.e. the sum of the time that it spends waiting in the memory since its last bit arrives and the time taken to transmit it on the output link. The maximum delay experienced by packets in class i when the input traffic parameters are $\vec{\rho}, \vec{\sigma}$, is denoted by $D_i^{\pi}(\vec{\rho}, \vec{\sigma})$. For notational convenience, when there is no possibility for confusion we may not indicate explicitly the dependence of the quantities defined above on $\vec{\rho}, \vec{\sigma}$ or π.

2.1 Tracking Service Disciplines

In the following sections we will make use the notion of Tracking Service Disciplines. This concept was developed in [16] in the context of tracking the Generalized Processor Sharing (GPS) discipline. It turns out that the fundamental properties of these policies (see Theorems 1 and 2 below) hold when tracking policies other than GPS and this enables us to prove the delay and buffer optimality of various tracking service disciplines.

Given a preemptive policy π we derive a work-conserving, non-preemptive policy $T(\pi)$ that operates as follows: Let $f_p^{\pi}(t)$ be the time at which packet p departs from a multiplexer that implements policy π assuming that there are no arrivals after time t. Then at each decision epoch t of $T(\pi)$, the server schedules a packet that achieves the minimum value of $f_p^{\pi}(t)$ over all eligible packets present in the system at time t. Thus, $T(\pi)$ attempts to preserve the order in which packets depart under the preemptive system. At each decision epoch t the $T(\pi)$ server picks the next packet that would depart from the system under the preemptive system if no more packets were to arrive after time t. Since more than one packet may leave the preemptive system simultaneously, ties are broken arbitrarily.[1]

When π obeys the following *Ordering Property* we can establish a tight coupling between the sample paths of π and $T(\pi)$:

> Let packets p and p' both be in the system at time τ and suppose that packet p completes service before packet p' if there are no arrivals after time τ. Then packet p will also complete service before packet p' for any pattern of arrivals after time τ. Further, if p and p' leave the system simultaneously when there are no arrivals after time τ, then they leave system simultaneously for any pattern of arrivals after time τ.

A consequence of the ordering property is that if the tracking server schedules a packet p at time τ before another packet p' that is also backlogged at time τ, then packet p cannot leave later than packet p' in the preemptive system.

This leads to the following results (first developed in the context of Generalized Processor Sharing in [16]). Let f_p be the time at which packet p departs from the preemptive system and let $\hat{f}_p$ be the time it departs from the tracking system.

Theorem 1 *Suppose the ordering property holds for the preemptive system and that the tracking service discipline is work conserving. For all packets p,*

$$\hat{f}_p - f_p < \frac{L_{\max}}{r}. \tag{5}$$

Theorem 2 *Suppose the ordering property holds for the preemptive policy π: Then for all times $t \geq 0$ and for each class i:*

$$Q_i^{T(\pi)}(t) - Q_i^{\pi}(t) \leq L_{\max}. \tag{6}$$

3 Buffer Allocation Mechanisms and Buffer Requirements

An important factor that affects the design of the scheduling policy and the sizing of the buffers is the flexibility of the buffer allocation mechanism (the function of assigning memory locations to arriving packets) used in the multiplexer. We consider three natural ways in which the multiplexer can structure its buffers:

1. Flexible Allocation (FL): Packets from all arrival streams share a common pool of memory, i.e. buffers are not allocated by class. This provides the most efficient use of memory, but may be difficult to implement since the multiple input links require that multiple parallel writes be implemented by a single control logic. In addition, a dynamic linked list structure is also needed to maintain packet order. In this case, the minimum multiplexer buffer size needed when policy π is implemented, B_{FL}^{π} is

$$B_{FL}^{\pi} = \sup_{\vec{\rho}} \sup_{t \geq 0} \sum_{i=1}^{N} Q_i^{\pi}(t, \vec{\rho}, \vec{\sigma}), \tag{7}$$

[1] Conceptually, a tracking policy simulates the performance of the preemptive system, but this can sometimes be accomplished without much computational overhead (see [16],[7]).

where $\vec{\rho}$ and $\vec{\sigma}$ are consistent with (1) and (2).

2. Semi-Flexible Allocation (SE): There are b_i^π bits of buffer allocated to packets from class i. The value of b_i^π cannot be changed after $t = 0$, however, the multiplexer is allowed to allocate the buffers based on the knowledge of $\vec{\rho}$ and $\vec{\sigma}$. It is again assumed that $\vec{\rho}$ and $\vec{\sigma}$ are consistent with (1) and (2). This limits the amount of memory sharing, but only requires the multiplexer to be programmable so that the allocations can match the traffic class characteristics. The link list structure then becomes simpler to implement than with a flexible allocation. Also, the multiple parallel writes can now be implemented through separate control logic modules. In this case:

$$B_{SE}^\pi = \sup_{\vec{\rho}} \sum_{i=1}^{N} \sup_{t\geq 0} Q_i^\pi(t, \vec{\rho}, \vec{\sigma}). \tag{8}$$

3. Fixed Allocation (FI): There are $\bar{b}_i^\pi$ bits of buffer allocated to packets from each class i that should be sufficient for all $\vec{\rho}$ and $\vec{\sigma}$ consistent with (1) and (2), i.e.

$$\bar{b}_i^\pi \geq \sup_{\vec{\rho}} \sup_{t\geq 0} Q_i^\pi(t, \vec{\rho}, \vec{\sigma})$$

Therefore,

$$B_{FI}^\pi = \sum_{i=1}^{N} \sup_{\vec{\rho}} \sup_{t\geq 0} Q_i^\pi(t, \vec{\rho}, \vec{\sigma}). \tag{9}$$

Clearly,

$$B_{FL}^\pi \leq B_{SE}^\pi \leq B_{FI}^\pi,$$

while the complexity and cost of implementation reduces from FL to SE to FI. Note that in some cases, it may be desirable to do the buffer allocation by *link* as opposed to class. As mentioned earlier, this is readily incorporated into the model by defining $N = K$ and setting $\rho_i = r_i$, $\sigma_i = 0$ for $i = 1, 2, ..., N$.

Given $\alpha \in$ {Flexible, Semi-Flexible, Fixed}, policy π^* is buffer-optimal policy among the class of admissible policies $\mathcal{A}$, if

$$B_\alpha^\pi \leq B_\alpha^{\pi'}, \text{ for all } \pi' \in \mathcal{A}.$$

We also define,

$$B_\alpha := \inf_{\pi\in\mathcal{A}} B_\alpha^\pi \tag{10}$$

3.1 Buffer-Optimal Multiplexers

In this section we address the issue of determining B_α (as defined in (10)) and the scheduling policies that achieve B_α, for flexible, semi-flexible and fixed buffer allocation multiplexers.

Proposition 1 *For flexible allocation,* $B_{FL} = NL_{\max} + \sum_{i=1}^{N} \sigma_i$, *and this value is achieved by any work-conserving service policy.*

Proposition 2 *For semi-flexible allocation,* $B_{SE} \geq L_{\max}(2N-1) + \sum_{i=1}^{N} \sigma_i$.

Before dealing with the fixed allocation case, we present a preemptive service policy called *Rate Proportional Processor Sharing (RPPS)* that was introduced in [17], and that obeys the ordering property described in Section 2.1. Recall that under our model, bits of a packet p are only eligible for service once the last bit of packet p has arrived. Let a class be backlogged at time t if a positive amount of eligible class i traffic is queued at time t. Then the RPPS server ensures that for any class i, if class i is continuously backlogged in the interval $[\tau, t]$ then

$$\frac{S_i(\tau,t)}{S_j(\tau,t)} \geq \frac{\rho_i}{\rho_j}, \quad j = 1, 2, ..., N. \tag{11}$$

Notice that if i and j are both continuously backlogged in the interval then (11) is met with equality. The following result is adapted from [17].

Proposition 3 *For fixed allocation,* $B_{FI} = 2NL_{\max} + \sum_{i=1}^{N} \sigma_i$, *and this value is achieved by T(RPPS).*

Since $B_{FI} \geq B_{SE}$, from Propositions 2 and 3 we immediately get the following result.

Corollary 1 $2NL_{\max} + \sum_{i=1}^{N} \sigma_i = B_{FI} \geq B_{SE} \geq L_{\max}(2N-1) + \sum_{i=1}^{N} \sigma_i$

Remarks.

1. From Corollary 1 it appears that the semi-flexible allocation does not provide significant savings in term of buffer requirements over fixed allocation. However, we will show that when packet delays are also considered, the Semi-flexible allocation allows the designing delay-optimal policies with low buffer requirements, while Fixed allocation does not.

2. In [3] it was shown that when $\sigma_i = 0$, $i = 1, \ldots, N$, and when the First-Come-First-Served (FCFS) policy is employed,

$$Q_i(t) \leq L_{\max}\left(1 - \frac{\rho_i}{r}\right) + \frac{\rho_i}{r} NL_{\max}.$$

By summing over all i, we conclude that $B_{SE}^{FCFS} \leq (2N-1)L_{\max}$. Together with Proposition 2, this implies that when $\sigma_i = 0$, $i =$

$1, \ldots, N$, the FCFS is buffer-optimal for semi-flexible allocation. However, this is not true for general σ_i, as the following example shows.

Consider the following arrival pattern. A packet of length $L_{\max}$ together with a burst of size σ_j arrives from each of the classes $j \neq i$ at time 0. At time 0^+ a packet p from class i of length $L_{\max}$ arrives, followed immediately by a burst of size σ_i. After time 0, traffic from class i arrives at rate ρ_i. At time t when packet p enters service:

$$Q_i(t) = \left(\frac{(N-1)L_{\max} + \sum_{j \neq i}^{N} \sigma_j}{r} \right) \rho_i + L_{\max} + \sigma_i.$$

Summing over i

$$\begin{aligned} B_{SE}^{FCFS} &\geq (2N-1)L_{\max} + 2\sum_{i=1}^{N} \sigma_i - \sum_{i=1} \rho_i \sigma_i \\ &\geq (2N-1)L_{\max} + 2\sum_{i=1}^{N} \sigma_i - \max_{1 \leq i \leq N} \sigma_i, \end{aligned}$$

which by Corollary 1 is in general larger than B_{SE}.

4 Buffer requirements v/s Delay. Delay Optimal Policies

In this section we address the issue of designing scheduling policies that provide predetermined delay bounds to each of the classes and have low buffer requirements. We start with some results that will be needed later and are of independent interest. They express the relationship that exists between bounds on the delays and buffer requirements.

Recall the definition of $D_i^{\pi}(\vec{\rho}, \vec{\sigma})$ and $M_i^{\pi}(\vec{\rho}, \vec{\sigma})$ from Section 2. In Theorem 3 we establish a useful lower bound on $D_i^{\pi}(\vec{\rho}, \vec{\sigma})$ as a function of $M_i^{\pi}(\vec{\rho}, \vec{\sigma})$ and the characteristics (σ_i, ρ_i) of traffic class i. Proposition 4 shows that under a fixed buffer allocation, no policy can be designed that has low buffer requirements and keeps the packet delays bounded for all traffic patterns consistent with (1) and (2). Conversely, we will see in Section 4.2 (Proposition 6), that again under a fixed buffer allocation no policy exists that is both delay-optimal and has low buffer requirements for all traffic patterns consistent with (1) and (2).

Theorem 3 *For any zero-loss multiplexer that implements policy π, we have:*

$$D_i^{\pi}(\vec{\rho}, \vec{\sigma}) \geq \frac{M_i^{\pi}(\vec{\rho}, \vec{\sigma}) - \sigma_i}{\rho_i} - L_{\max}\left(\frac{1}{\rho_i} - \frac{1}{r}\right) \quad (12)$$

Proposition 4 *If under fixed buffer allocation and policy π it holds that the buffer allocation $\bar{b}_i^{\pi}$ for each link is such that $\bar{b}_i^{\pi} < NL_{max} + \sigma_i$ for all traffic patterns consistent with (1) and (2), then there is a choice of traffic patterns such that $\sup_j D_j^{\pi} > d$ for any given $d > 0$.*

4.1 Delay-Optimal Policies

To proceed, we need some notation and definitions. Let the non-negative vector $\vec{D} = (D_1, ..., D_N)$ be a list of required upper bounds on delay so that no class i packet is delayed by more than D_i time units in the multiplexer. The *deadline* of packet p from class i that arrives at time a_p is defined as $d_p = a_p + D_i$. If f_p is the finishing time of p, its *lateness* is defined as $l_p = f_p - d_p$. Given a zero-loss multiplexer that implements service policy π, the vector $\vec{D} = (D_1, ..., D_N)$ is *schedulable* under π if for all arrival patterns consistent with (1) and (2), and for all classes i, no class i packet is delayed by more than D_i time units. The *schedulable region* Ω^{π} of the policy π is the set of all vectors schedulable under π. Given a class of admissible policies $\mathcal{A}$, the *schedulable region of* $\mathcal{A}$ is $\bigcup_{\pi \in \mathcal{A}} \Omega^{\pi}$ and a vector is *schedulable in* $\mathcal{A}$ if it belongs to the schedulable region of $\mathcal{A}$. We define a scheduling policy π^* to be delay-optimal in $\mathcal{A}$ if

$$\Omega^{\pi} \subseteq \Omega^{\pi^*} \quad (13)$$

for all policies $\pi \in \mathcal{A}$.

It is well known that under any arrival pattern the Preemptive Earliest Deadline First (PEDF), i.e., the policy that always schedules the packet with the smallest deadline first, minimizes the maximum lateness of all the packets. This implies that the PEDF policy is delay-optimal among all scheduling policies. For non-preemptive policies, no policy is known that minimizes the maximum lateness of all packets over all arrival patterns. However, we will show that under constraints (1) and (2) the non-preemptive EDF (NPEDF) and the PEDF tracking policy (T(PEDF)) are delay-optimal. We will also provide the schedulable region of these non-preemptive policies.

Before we proceed, it is important to note that in general NPEDF may differ significantly from T(PEDF). The following example shows that the order in which packets are scheduled under NPEDF may be quite different from that of T(PEDF):

Example. Assume that the server works at rate 1 and let packets arrive as follows: At time 0 a maximum size packet with deadline $ML_{\max}$ arrives and at time 0^+, K ($< L_{\max}$) 1-bit packets with

deadline $ML^-_{\max}$ arrive. Thereafter, at each time $t = L_{\max}, 2L_{\max}, \ldots, ML_{\max}$, a maximum size packet with deadline $t + L_{\max}$ arrives. Under PEDF, the K 1-bit packets will be transmitted upon arrival and will depart from the system by time K, and under both NPEDF and T(PEDF*) the packet arriving at time 0 goes into service upon arrival and stays in service until it departs at time $L_{\max}$. The difference between the two non-preemptive policies manifests itself after time $L^+_{\max}$, when T(PEDF*) serves the K 1-bit packets, while NPEDF serves the packet with deadline $2L_{\max}$. Observe that in fact under NPEDF, all 1-bit packets leave the system after time $(M-1)L_{\max}$. This example will used again in Section 5.2. □

The example notwithstanding, the next result establishes that when the packet sizes are fixed, both T(PEDF) and NPEDF behave identically. Let f_p be the time at which packet p departs under PEDF and $\widehat{f}_p$ be the time it departs under T(PEF).

Proposition 5 *For fixed packet sizes at $L \leq L_{\max}$, T(PEDF) and NPEDF behave identically.*

We now show the optimality of both the NPEDF and T(PEDF) policies.

Theorem 4 *The NPEDF and T(PEDF) policies are delay-optimal among the class of non-preemptive policies. The schedulable regions of NPEDF and T(PEDF), are nonempty only if $\sum_{i=1}^{N} \rho_i \leq r$, and consists of the set of vectors which satisfy the constraints*

$$(k+1)L_{\max} + \sum_{n=1}^{k} \sigma_{i_n} \leq D_{i_k}\left(r - \sum_{n=1}^{k-1} \rho_{i_n}\right) + \sum_{n=1}^{k-1} \rho_{i_n} D_{i_n}$$

for $1 \leq k \leq N-1$ and

$$NL_{\max} + \sum_{n=1}^{N} \sigma_{i_n} \leq D_{i_N}\left(r - \sum_{n=1}^{N-1} \rho_{i_n}\right) + \sum_{n=1}^{N-1} \rho_{i_n} D_{i_n},$$

whenever $D_{i_1} \leq D_{i_2} \leq \ldots \leq D_{i_N}$.

The schedulable region of PEDF under the arrival patterns considered in this section can be found using similar arguments as those used to prove Theorem 4. In Figures 1 and 2 we show the schedulable regions of PEDF and NPEDF under various parameters. As we see, in both figures the two regions differ by two strips which have width $L_{\max}/r$. In fact, by examining the schedulable regions it is easy to see that if the vector $\{D_1, \ldots, D_N\}$ is schedulable under PEDF, then the vector $\{D_1 + L_{\max}/r, \ldots, D_N + L_{\max}/r\}$ is schedulable under NPEDF. As we will see in the next section, this is a consequence of a general result that holds for any arrival patterns. Also, we see in Figure 1,

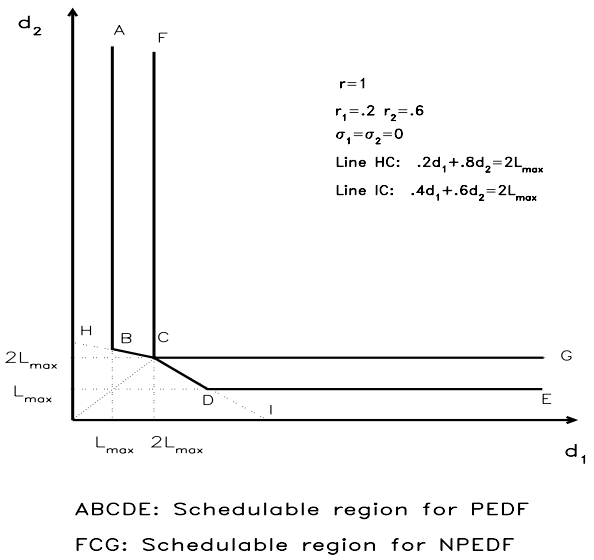

Figure 1: Schedulable regions for $\sigma_1 = \sigma_1 = 0$.

where $\sigma_1 = \sigma_2 = 0$, that any schedulable vector under NPEDF has coordinates larger than $2L_{\max}/r$. Since the vector $\{2L_{\max}/r,\ 2L_{\max}/r\}$ is schedulable under the First-Come-First-Served (FCFS) policy, it follows that in this case from the point of view of schedulability there is no point in employing another scheduling policy. In fact, as can be seen from Theorem 4 this is true always when $N = 2$ and $\sigma_1 = \sigma_2 = 0$.

4.2 Delay-Optimal Policies with Low Buffer Requirements

In this section, we address the issue of designing delay-optimal policies with low buffer requirements. We propose a policy that is delay-optimal and under semi-flexible allocation has low buffer requirements. Note that based on Proposition 1, a delay-optimal policy will also have minimum buffer requirements if a flexible allocation is used. However, we will see that the improvement over the semi-flexible case is small and may, therefore, not warrant the additional cost and complexity.

We first motivate the use of semi-flexible allocation by showing that under fixed allocation the buffer requirements of any delay-optimal policy are at least $O(N^2)$.

Proposition 6 *Let π be any non-preemptive policy that is delay-optimal for all traffic patterns consistent with (1) and (2) . Under fixed-allocation,*

$$B^{\pi}_{FI} \geq N^2 L_{\max} + N \sum_{i=1}^{N} \sigma_i.$$

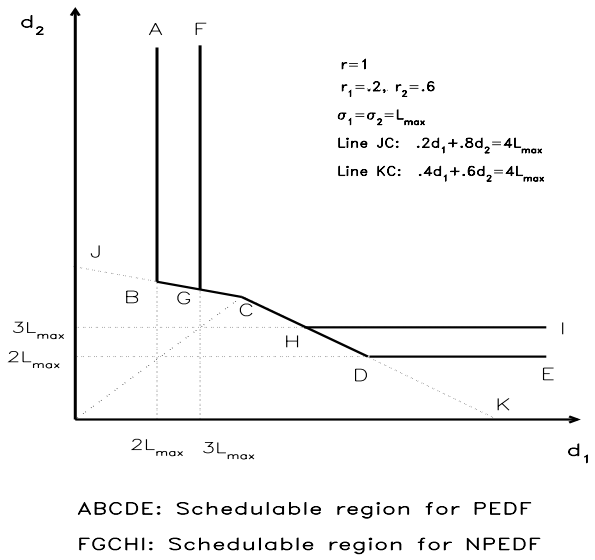

Figure 2: Schedulable regions for $\sigma_1 + \sigma_1 \neq 0$.

The next proposition shows that even under semi-flexible allocation, the delay-optimal policies NPEDF and T(PEDF) still yield buffer requirements of at least $O(N^2)$.

Proposition 7 *With $\alpha \in \{NPEDF, T(PEDF)\}$,*

$$B_{SE}^{\alpha} \geq \frac{N(N-1)}{2} L_{\max} + \sum_{n=1}^{N} n\sigma_{i_n},$$

where $\sigma_{i_1} \leq \ldots \leq \sigma_{i_N}$.

The question now arises whether one can design policies for semi-flexible allocation, that have buffer requirements lower than $O(N^2)$. We show next that this is indeed the case. Specifically, we construct delay-optimal policies with buffer requirements $O(N)$.

Theorem 5 *There is a delay-optimal policy π^* among the class on non-preemptive policies such that for all arrival patterns consistent with (1) and (2) .*

$$B_{SE}^{\pi^*} \leq 2NL_{\max} + 2\sum_{i=1}^{N} \sigma_i.$$

The proof of Theorem 5 provides a simple algorithm for constructing policy π^*.

Algorithm For Constructing Policy π^*. The input to the algorithm is a feasible vector $\vec{D}$. The output is the vector $\vec{D}'$ which is used as the delay vector for NPEDF or T(PEDF).

1. Sort the components of vector $\vec{D}$ so that $D_{i_1} \leq \ldots \leq D_{i_N}$.
2. Set $l = 0$ and $\vec{D}^{(l)} = \vec{D}$.
3. If equality in the N^{th} constraint in Theorem 4 then set $\vec{D}' = \vec{D}^{(l)}$ and stop. Else,
4. Let K be the largest index such that

$$\min\{K+1, N\}L_{\max} + \sum_{n=1}^{K} \sigma_{i_n} < D_{i_K}\left(r - \sum_{n=1}^{K-1} \rho_{i_n}\right) + \sum_{n=1}^{K-1} \rho_{i_n} D_{i_n}.$$

   Define

$$\epsilon := \min_{K \leq k \leq N} \left\{ \frac{D_{i_k}\gamma_k + \sum_{n=1}^{k-1} \rho_{i_n} D_{i_n} - \tau_k}{\gamma_K} \right\}$$

   where $\gamma_j = r - \sum_{n=1}^{j-1} \rho_{i_n}$ and $\tau_k = \min\{K+1, N\}L_{\max} + \sum_{n=1}^{k} \sigma_{i_n}$. Set $D_{i_n}^{(l+1)} = D_{i_n}^{(l)}$, $n = 1, \ldots, K-1$, and $D_{i_n}^{(1)} = D_{i_n} - \epsilon$, $n = K, \ldots, N$.
5. Set $l \leftarrow l + 1$ and go to step 3.

5 Optimality Criteria for Soft Deadlines

5.1 Minimization of Maximum Lateness

In the previous section we provided the schedulable region of NPEDF, T(PEDF) and PEDF under the assumption that the arriving traffic satisfies certain constraints. In this section we consider the problem of designing scheduling policies when the objective is to keep the lateness of all packets as low as possible. This criterion is of interest in situations where the deadlines represent a desirable time by which the packets should be transmitted and it is important to transmit each packet as early as possible and in a fair manner relative to the transmission times of the rest of the packets. It is well known that PEDF is a good policy with respect to this type of objectives in the sense that among all scheduling policies, it minimizes the maximum lateness of all packets under any arrival pattern. However, it is easy to construct arrival patterns for which the NPEDF is not optimal with respect to the min-max criterion among the non-preemptive policies. In spite of this, we will show in the next Theorem that NPEDF is still a good policy in the sense that the maximum lateness under NPEDF is at most $L_{\max}/r$ larger than the maximum lateness under PEDF under *any* arrival pattern, i.e., even for traffic streams that do not satisfy the conditions of (1) and (2).

Theorem 6 *Let f_p, $\hat{f}_p$ be the finishing time of packet p under the PEDF and NPEDF policies respectively and let d_p be its deadline. Then the following inequality holds under any arrival pattern:*

$$\sup_p \left\{ \hat{f}_p - d_p \right\} \leq \sup_p \{ f_p - d_p \} + \frac{L_{\max}}{r}.$$

Note from Theorem 1, that T(PEDF) has the stronger property,

$$\hat{f}_p - d_p \leq f_p - d_p + \frac{L_{\max}}{r}$$

Corollary 2 *If under any arrival pattern the vector of packet deadlines $\{d_i\}_{i=1}^{\infty}$ is schedulable under PEDF, then the vector $\{d_i + (L_{\max}/r)\}_{i=1}^{\infty}$ is schedulable under NPEDF.*

5.2 Lexicographic Optimization.

A stronger optimality criterion than minimizing the maximum lateness which relates closer to fairness, is the criterion of lexicographic optimization of packet lateness, which is defined below.

Let $\{l_i\}_{i=1}^n$, $\{u_i\}_{i=1}^n$, be two n-dimensional vectors and let $\pi_l(i)$, $\pi_u(i)$, be index permutations such that

$$l_{\pi_l(1)} \geq \ldots \geq l_{\pi_l(n)}, \qquad u_{\pi_u(1)} \geq \ldots \geq u_{\pi_u(n)}.$$

The vector $\{l_i\}_{i=1}^n$ is called *lexicographically smaller* than the vector $\{u_i\}_{i=1}^n$, denoted as $\{l_i\}_{i=1}^n \leq_{lex} \{u_i\}_{i=1}^n$, if 1) $l_{\pi_l(1)} \leq u_{\pi_u(1)}$ and 2) $l_{\pi_l(i)} > u_{\pi_u(i)}$ for some $i = 2, \ldots, n$ implies that $l_{\pi_l(j)} < u_{\pi_u(j)}$ for some $j < i$. Let $\mathcal{A}$ be a set of n-dimensional vectors. Vector $\{l_i^*\}_{i=1}^n \in \mathcal{A}$ is lexicographically optimal in $\mathcal{A}$ if $\{l_i^*\}_{i=1}^n \leq_{lex} \{u_i\}_{i=1}^n$ for all $\{u_i\}_{i=1}^n \in \mathcal{A}$.

The property of the lexicographically optimal vector that relates to fairness is that if one attempts to reduce coordinate i by picking another vector in $\mathcal{A}$, then necessarily another coordinate that is larger than coordinate i will have to be increased (see [2, Section 6.5.2]).

It turns out that if preemptions are allowed, one of the $PEDF$ policies is lexicographically optimal. Specifically, let $PEDF^*$ be the policy that serves preemptively the packets with the earliest deadline first and among the packets with the earliest deadline it serves the packets with shortest remaining service time first. Among packets with the same deadline and the same remaining processing time, $PEDF^*$ selects one in an arbitrary fashion. To provide a precise formulation of the optimality of $PEDF^*$, we will assume that the number of arrivals in finite intervals is finite and

$$\limsup_{t \to \infty} \frac{A(0,t)}{rt} < 1,$$

where $A(0,t)$ is the work that arrives to the system up to time t. These constraints imply that the busy periods of any work-conserving policy, as well as the number of packets served within a busy period are finite.

Theorem 7 *Among all policies, $PEDF^*$ minimizes lexicographically the lateness vector of the packets that arrive during any busy period.*

The next observation, which follows directly from Theorem 1 shows that $T(PEDF^*)$ is "almost" lexicographically optimal.

Corollary 3 *Let $\{t_1\}_{p=1}^n$ and $\{l_1\}_{p=1}^n$ be the lateness vectors for a given set of arrivals when the service policy is T(PEDF) and PEDF respectively. Then for each packet arrival p:*

$$l_p \leq t_p - L_{\max}.$$

The natural question to ask at this point is the performance of NPEDF with respect to the lexicographic criterion. The following example shows that for certain arrival patterns the policy is far from being lexicographically optimal.

Example. Consider the same arrival stream as in the example of Section 2.1. Since the K 1-bit packets will will be transmitted upon arrival by $PEDF^*$, therefore their lateness is $-M$. However, under any of the NPEDF policies, the lateness of the K packets is 0 while the lateness of the other packets remains essentially the same as when the $PEDF^*$ policy is employed. In other words, $PEDF^*$ improves the lateness of K packets by M, and yet only minimally penalizes the other packets. □

The example notwithstanding, NPEDF *is* almost lexicographically optimal for fixed size packets since from Proposition 5 it behaves identically to T(PEDF).

6 Conclusions and Extensions

The main conclusions of this paper are the following. If the only objective is to have low buffer requirements the fixed allocation mechanism is adequate in practice. If however, good delay performance is also required, fixed allocation leads to large buffer requirements. In contrast, under the semi-flexible allocation, delay-optimal policies with low buffer requirements can be designed. While it is easier to implement NPEDF than T(PEDF), T(PEDF) may be the delay optimal policy of choice if it is desirable to apportion lateness in packet finishing times in a fair manner.

In general, the definitions and results of this paper establish a framework within which various service policies can be evaluated and compared. The class of tracking policies that was introduced may be of independent interest in other applications. The natural direction in which the results should be extended is to multiple links, and this is our focus for future work.

References

[1] ATM UNI specification version 3.0. Technical report, ATM Forum, September 1993.

[2] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, second edition, 1992.

[3] A. Birman, P. C. Chang, J. S.-C. Chen, and R. Guérin. Buffer sizing in an ISDN frame relay switch. Technical Report RC 14386, IBM Research, IBM T. J. Watson Research Center, August 1989.

[4] A. Birman, H. R. Gail, S. L. Hantler, Z. Rosberg, and M. Sidi. An optimal service policy for buffer systems. Technical Report RC 16641, IBM Research, IBM T. J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, April 1991.

[5] I. Cidon, I. Gopal, G. Grover, and M. Sidi. Real-time packet switching: A performance analysis. *IEEE J. Sel. Areas Commun.*, SAC-6(9):1576–1586, December 1988.

[6] R. L. Cruz. A calculus of delay, Part I: Network element in isolation. *IEEE Trans. Inform. Theory*, IT-37(1):114–131, January 1991.

[7] Alan Demers, Srinivasan Keshav, and Scott Shenkar. Analysis and simulation of a fair queueing algorithm. *Internetworking Research and Experience*, 1(1), 1990.

[8] M. L. Dertouzos and A. K.-L. Mok. Multiprocessor on-line scheduling of hard-real-time tasks. *IEEE Trans. Softw. Eng.*, 15(12):1497–1506, December 1992.

[9] H. R. Gail, G. Grover, R. Guérin, S. L. Hantler Z. Rosberg, and M. Sidi. Buffer size requirements under longest queue first. *Performance Evaluation*, 18(2), September 1993.

[10] L. Georgiadis, R. R. Guérin, and A. Parekh. Optimal multiplexing on a single link: Delay and buffer requirements. Technical report, IBM Research, IBM T. J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, March 1994.

[11] S. J. Golestani. Duration-limited statistical multiplexing of delay sensitive traffic in packet networks. In *Proceedings of IEEE INFOCOM '91*, 1991.

[12] K. Jeffay and R. Anderson. On optimal scheduling of periodic and sporadic tasks. Technical Report TR 88–11–06, University of Washington, University of Washington, Dept. Comput. Science FR-35, Seattle, WA 98195, November 1988.

[13] K. Jeffay, D. F. Stanat, and C. U. Martel. On non-preemptive scheduling of periodic and sporadic tasks. In *IEEE Proc. Real-Time Systems Symposium*, pages 129–139, San Antonio, TX, 1991.

[14] A. Lazar and G. Pacifici. Control of resources in broadband networks with quality of service guarantees. *IEEE Communications Magazine*, September 1991.

[15] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the Association for Computing Machinery*, 20(1):46–61, 1973.

[16] A. K. Parekh. *A Generalized Processor Sharing Aproach to Flow Control in Integrated Services Networks*. PhD thesis, Department of Electrical Engineering and Computer Science, MIT, February 1992.

[17] A. K. Parekh and R. G. Gallager. A Generalized Processor Sharing approach to flow control in Integrated Services Networks—The Single Node Case. *ACM/IEEE Transactions on Networks*, 1(3):344–357, June 1993.

[18] G. Sasaki. Input buffer requirements for round robin polling systems. In *Proc. Allerton Conference on Communication, Control and Computing*, 1989.

[19] Lixia Zhang. *A New Architecture for Packet Switching Network Protocols*. PhD thesis, Department of Electrical Engineering and Computer Science, MIT, August 1989.

[20] Qin Zheng. *Real-time Fault-tolerant Communication in Computer Networks*. PhD thesis, University of Michigan, 1993.