# Proportional Differentiated Services: Delay Differentiation and Packet Scheduling

Constantinos Dovrolis, *Member, IEEE*, Dimitrios Stiliadis, *Member, IEEE*, and Parameswaran Ramanathan, *Member, IEEE*

***Abstract*—The proportional differentiation model provides the network operator with the ‘tuning knobs' for adjusting the per-hop quality-of-service (QoS) ratios between classes, independent of the class loads. This paper applies the proportional model in the differentiation of queueing delays, and investigates appropriate packet scheduling mechanisms. Starting from the proportional delay differentiation (PDD) model, we derive the average queueing delay in each class, show the dynamics of the class delays under the PDD constraints, and state the conditions in which the PDD model is feasible. The feasibility model of the model can be determined from the average delays that result with the strict priorities scheduler. We then focus on scheduling mechanisms that can implement the PDD model, when it is feasible to do so. The proportional average delay (PAD) scheduler meets the PDD constraints, when they are feasible, but it exhibits a pathological behavior in short timescales. The waiting time priority (WTP) scheduler, on the other hand, approximates the PDD model closely, even in the short timescales of a few packet departures, but only in heavy load conditions. PAD and WTP serve as motivation for the third scheduler, called hybrid proportional delay (HPD). HPD approximates the PDD model closely, when the model is feasible, independent of the class load distribution. Also, HPD provides predictable delay differentiation even in short timescales.**

***Index Terms*—Dynamic priorities, quality of service, resource management algorithms.**

## I. Introduction

PROVIDING some kind of service differentiation in the Internet has been a problem of enormous commercial and research importance in at least the last ten years. The differentiated services (DiffServ) architecture [2] was proposed as a more scalable solution to this problem, compared to previous approaches such as the integrated services (IntServ) architecture [34]. The DiffServ vision is that stateless priority mechanisms at the network core can be combined with stateful mechanisms at the network edges, in order to construct versatile end-to-end services. Among the first DiffServ proposals was the virtual leased line (VLL) service [18] and the assured service [8]. Recent findings, however, have raised concerns about the effect of aggregation on the VLL service [7], and about the provisioning complexity of the assured service [31], [29]. Several other models for scalable differentiated services have been proposed. To name a few, the *SCORE* architecture can provide per-flow service guarantees without per-flow state in the core routers [32], [33]. [6] proposes a core-stateless scheme in which resource management and admission control are performed only at egress routers. The asymmetric best effort [5] provides two service classes: one for delay-sensitive applications and another for throughput-sensitive applications.

Manuscript received April 7, 2001; revised June 21, 2001; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor C. Diot. The work of C. Dovrolis and P. Ramanathan was supported in part by the National Science Foundation under Grant MIP-9526761, and by a research grant from Cisco Systems Inc. An earlier version of this paper was presented at the ACM SIGCOMM conference, Cambridge, MA, 1999.

C. Dovrolis is with the Department of Computer and Information Science, University of Delaware, Newark, DE 19716 USA (e-mail: dovrolis@cis.udel.edu).

D. Stiliadis is with Bell Laboratories, Lucent Technologies, Holmdel, NJ 07733 USA.

P. Ramanathan is with the Department of Electrical and Computer Engineering, University of Wisconsin–Madison, Madison, WI 53706 USA.

Publisher Item Identifier S 1063-6692(02)01079-8.

A fundamentally different approach in the DS framework is the *relative differentiated services*. In this approach, the network traffic is grouped into $N$ classes of service which are ordered, such that Class $i$ is better (or at least no worse) than Class $(i-1)$ for $1 < i \leq N$, in terms of local (per-hop) metrics for the queueing delays and packet losses [12]. The eight *Class Selector PHBs*, standardized by the IETF [25], follow the relative service differentiation model. In this context, applications and users do not get an absolute service level assurance, such as an end-to-end delay bound or throughput, since there is no admission control and resource reservations. Instead, the network assures that higher classes will offer better QoS than lower classes, and so it is up to the applications and users to select the class that best meets their requirements, cost, and policy constraints [15]. From this point of view, the relative service differentiation model follows the architectural principle of *end-system adaptation,* since it provides the class selection as an additional dimension in the end-system adaptation space.

There are several ways in which a network can provide relative differentiated services. For example, the differentiation can be based on appropriate pricing (i.e., higher classes are more expensive) [26], or on capacity provisioning (i.e., higher classes get more bandwidth relative to their expected loads) [17]. Such mechanisms, however, cannot always provide consistent class differentiation, because the relative QoS differentiation between classes varies with the class loads. Other mechanisms, such as strict class prioritization, provide consistent differentiation that does not depend on the load variations, but they do not allow the network operator to adjust the relative QoS between classes. Such tuning knobs are necessary in a practical setting, since the network operators must be able to adjust the QoS difference between classes based on pricing and policy objectives. Our basic premise, starting from these limitations of other relative differentiation models, is that a relative differentiated services architecture has to be:

- *Predictable*, in the sense that the differentiation should be consistent (i.e., higher classes are better, or at least no worse) independent of the class load variations;
- *Controllable*, meaning that the network operators should be able to adjust the relative QoS difference between classes based on their criteria.

As a target for predictable and controllable relative differentiation, we consider the *proportional differentiation model* [12]. According to this model, the basic performance measures for packet forwarding locally at each hop are ratioed proportionally to certain *class differentiation parameters* that the network operator chooses. Even though there is no wide consensus on the most appropriate performance measures for packet forwarding, it is generally agreed that a better network service means lower queueing delays and lower likelihood of packet losses. In this paper, we apply the proportional model in the case of queueing delay differentiation. The sequel paper [13] focuses on loss differentiation and appropriate buffer management, considering also the case of coupled delay and loss differentiation.

In the first part of this paper, we propose and study the proportional delay differentiation (PDD) model. The PDD model controls the ratios of the average queueing delays between classes, based on the specified delay differentiation parameters (DDPs). Starting from the PDD model, we derive the average queueing delay in each class given a class load distribution. We then show certain dynamic properties for the class delays in PDD, and state the conditions under which the PDD model is feasible. In the second part of the paper, we study three schedulers for the PDD model. The proportional average delays (PAD) scheduler appears to always meet the PDD model, when it is feasible to do so. PAD, however, exhibits unpredictable behavior in short timescales. The waiting time priorities (WTP) scheduler, on the other hand, approximates the PDD model closely even in short timescales, but only in heavy load conditions. A third scheduler, called hybrid proportional delays (HPD), combines the operation of PAD and WTP. Our simulation study shows that, in heavy load conditions, above 90% or so, both PAD and WTP meet the PDD model, and so does HPD. HPD approximates the PDD model closely, when the model is feasible, independent of the class load distribution. Also, HPD provides predictable delay differentiation even in short timescales.

The structure of the paper is as follows. In Section II, we study the PDD model and present its dynamic properties and feasibility conditions. Sections III, IV, and V focus on the PAD, WTP, and HPD schedulers, respectively. In Section VI, we review other scheduling disciplines in the area of relative and proportional delay differentiation. We conclude in Section VII.

## II. Proportional Delay Differentiation

We consider the following queueing model of a packet multiplexer (or 'link'). A *lossless*, *work-conserving*, and *nonpreemptive* link with capacity $C$ (bytes per second) services $N$ first-come first-served (FCFS) queues, one for each traffic class[1]. The order in which packets are selected for service is determined by the *packet scheduler* $\mathcal{S}$. The lossless property requires that the utilization of $\mathcal{S}$ is less than 100%, and that there are enough buffers for packets that need to be queued. The work-conserving property means that the scheduler does not idle when there are waiting packets. The nonpreemptive property means that a packet transmission is always carried out to completion.

The interarrivals between packets of class $i$ are IID random variables, generated by a renewal arrival process for that class. The packet sizes in class $i$ are also IID random variables, generated by a renewal packet size process for that class. The arrival and packet size processes of different classes are independent, and, in general, different. Some additional assumptions are made later, where needed. The *aggregate traffic stream* that arrives in the queueing system is determined by the superposition of the $N$ packet streams that arrive in each class. Let $\lambda_i$ be the average input rate in class $i$ (packets per second), and $\lambda = \sum_{i=1}^{N} \lambda_i$ the average rate of the aggregate traffic stream. Let $\bar{L}_i$ be the *average size* (bytes) of class $i$ packets, and $\bar{L} = \sum_{i=1}^{N}(\lambda_i \bar{L}_i)/\lambda$ the average size among all packets. The link *utilization* is $u = \bar{L}\lambda/C < 1$.

The PDD model aims to control the *ratios* of the average class queueing delays based on the DDPs $\{\delta_i, i = 1, \ldots, N\}$. Specifically, let $\bar{d}_i$ be the *average queueing delay*, or simply average delay, of the class $i$ packets. The PDD model requires that the ratio of average delays between two classes $i$ and $j$ is fixed to the ratio of the corresponding DDPs

$$\frac{\bar{d}_i}{\bar{d}_j} = \frac{\delta_i}{\delta_j} \qquad 1 \le i, \quad j \le N. \tag{1}$$

Our convention is that higher classes provide better service, i.e., lower queueing delays, and so $\delta_1 > \delta_2 > \cdots > \delta_N > 0$. In the following, we choose Class-1 as the 'reference class' and set $\delta_1 = 1$. Then, the PDD model requires that the average delay of each class $i$ is a certain fraction $\delta_i$ of the average delay of Class-1

$$\bar{d}_i = \delta_i \bar{d}_1 \qquad i = 2, \ldots, N \tag{2}$$

independent of the aggregate load, or the class load distribution.

### A. Per-Class Average Delays in PDD

In given load conditions, the $N - 1$ ratios of the PDD model specify uniquely the average delays of the $N$ classes. The key additional relation in the mapping from delay ratios to class delays is the *conservation law* [19], [4], which constrains the average class delays in any work-conserving scheduler. Specifically, the conservation law requires that for any work-conserving scheduler $\mathcal{S}$ that causes an average delay $\bar{d}_i$ in class $i$, we must have that

$$\sum_{i=1}^{N} \lambda_i \bar{L}_i \bar{d}_i = \lambda \bar{L} \bar{d}_{\mathrm{ag}} = \bar{q}_{\mathrm{ag}} \tag{3}$$

[1]We use the terms 'class' and 'queue' interchangeably.

where $\bar{d}_{\rm ag}$ and $\bar{q}_{\rm ag}$ is the average delay (in seconds) and the average backlog (in bytes), respectively, in a FCFS scheduler that has the same capacity and services the same aggregate traffic stream as $\mathcal{S}$. The conservation law (in the previous form) holds under arbitrary distributions for the packet interarrivals and packet sizes, as long as the first moment of these distributions ($\lambda_i$ and $\bar{L}_i$) exist. The conservation law implies that even though a scheduler $\mathcal{S}$ can affect the relative magnitude of the class delays, making the delay of one class lower than the delay of another class, this is a 'zero-sum' game because the weighted sum of (3) has to be equal to the average backlog $\bar{q}_{\rm ag}$ of the aggregate traffic stream. The average backlog $\bar{q}_{\rm ag}$ is a significant *invariant* in this balance, as it does not depend on the scheduling discipline $\mathcal{S}$, but only on the aggregate traffic stream and on the capacity $C$.

Suppose now that the scheduler $\mathcal{S}$ satisfies the PDD model of (1). With the additional constraint (3) that the conservation law imposes, we can show that the average delay in class $i$ is

$$\bar{d}_i = \frac{\delta_i \bar{q}_{\rm ag}}{\sum_{n=1}^{N} \delta_n \lambda_n \bar{L}_n} \qquad i = 1, \ldots, N. \tag{4}$$

Consequently, even though the PDD model consists of $N-1$ *relative* constraints, the average delay in each class is determined in an absolute sense when the PDD model is applied to a certain traffic stream with class loads $\{\lambda_i\}$, average packet sizes $\{\bar{L}_i\}$, and average aggregate backlog $\bar{q}_{\rm ag}$.

### B. Delay Dynamics in the PDD Model

In this section, we further assume that the $N$ classes have the same packet size distribution. We can then set $\bar{L}_i = \bar{L} = 1$, and normalize the backlog measures to *average packet* units. The average delay in each class is then simplified to

$$\bar{d}_i = \frac{\delta_i \bar{q}_{\rm ag}}{\sum_{n=1}^{N} \delta_n \lambda_n} \qquad i = 1, \ldots, N. \tag{5}$$

Additionally, we assume that the $N$ classes have the same interarrival distribution, but with a possibly different average interarrival $1/\lambda_i$.

Based on (5), we can now investigate the variations in the average class delays under the constraints of the PDD model, as the aggregate load, the class load distribution, or the DDPs vary. We refer to the following properties as the *delay dynamics in the PDD model*. The proofs of these properties are given in Appendix I.

*Property 1:* Increasing the input rate of a class, increases (in the wide sense[2]) the average delay of all classes.

This property shows that there is no isolation between classes in the PDD model, i.e., when the delay of a class increases, due to additional load in that class, the delays of all classes will also increase.

*Property 2:* Increasing the rate of a higher class causes a larger increase in the average class delays than increasing the rate of a lower class.

[2]'*Increasing a function $f(x)$ in the wide sense*' means that the corresponding function is nondecreasing, i.e., $df(x)/dx \geq 0$.

In the case of two classes, for instance, suppose that the class rates become either $\lambda_1' = \lambda_1 + \epsilon$ and $\lambda_2' = \lambda_2$, or $\lambda_1'' = \lambda_1$ and $\lambda_2'' = \lambda_2 + \epsilon (\epsilon > 0)$. Even though the conservation law requires that the weighted average of the class delays is the same in both cases $(\lambda_1' \bar{d}_1' + \lambda_2' \bar{d}_2' = \lambda_1'' \bar{d}_1'' + \lambda_2'' \bar{d}_2'')$, the class average delays in the second case are larger, i.e., $\bar{d}_1'' > \bar{d}_1'$ and $\bar{d}_2'' > \bar{d}_2'$, because of the PDD constraints. This property shows that higher classes cost more, in terms of queueing delay, than lower classes.

*Property 3:* Decreasing the delay differentiation parameter of a class increases (in the wide sense) the average delay of all other classes, and decreases (in the wide sense) the average delay of that class.

This property implies that if the delay of a class is reduced, by lowering its DDP, then the delay of all other classes will increase.

The following two properties are important in the dynamic class selection DCS framework [15], in which users with a certain maximum end-to-end delay constraint search for the minimum acceptable class. The first property states that when one ore more users move to a higher class, the delay of all classes increases. The class delays decrease, on the other hand, when one or more users move to a lower class. The second property shows that when a user switches from one class to another, the user still observes a consistent class ordering, i.e., the higher class provides a lower delay.

Suppose that the class load distribution changes from $\{\lambda_n\}$ to $\{\lambda_n'\}$, with $\lambda_i' = \lambda_i - \epsilon, \lambda_j' = \lambda_j + \epsilon$, and $\lambda_k' = \lambda_k$ for all $k \neq i, j (\epsilon > 0)$. Let $\bar{d}_n'$ be the average delay in class $n$ when the class load distribution is $\{\lambda_n'\}$.

*Property 4:* If $i > j$ then $\bar{d}_n' \leq \bar{d}_n$ for all $n = 1 \ldots N$. Similarly, if $i < j$ then $\bar{d}_n' \geq \bar{d}_n$.

*Property 5:* If $i > j$ then $\bar{d}_j' \geq \bar{d}_i$. Similarly, if $i < j$ then $\bar{d}_j' \leq \bar{d}_i$.

### C. Feasibility of the PDD Model

We have assumed so far that the PDD model is *feasible*, i.e., that there exists a work-conserving scheduler that can meet the constraints of (2). However, this is not always possible. Given the load distribution and the average backlog of the aggregate traffic, the PDD model not only specifies the $N-1$ delay ratios, but also the $N$ average delays of (4). However, there may not exist a work-conserving scheduler that can set the average delay of each class as in (4). Intuitively, the reason is that the average delay of a class (or of any set of classes) has a minimum value due to the inherent load in that class (or set of classes). This minimum average delay would result if that class (or set of classes) was given strict priority over the rest of the traffic.

Formally, given the input rates $\{\lambda_i\}$, the average class packet sizes $\{\bar{L}_i\}$, and the average backlog $\bar{q}_{\rm ag}$ of the aggregate traffic stream, we say that a set of DDPs $\{\delta_i, i = 2, \ldots, N\}$ is feasible when there exists a work-conserving scheduler that can set the average delay of each class as in (4). So, the set of DDPs $\{\delta_i, i = 2, \ldots, N\}$ is feasible when the set of class delays $\{\bar{d}_i = \delta_i \bar{q}_{\rm ag} / (\sum_{n=1}^{N} \lambda_n \delta_n \bar{L}_n), i = 1, \ldots, N\}$ is feasible.

The necessary and sufficient conditions for the feasibility of a set of $N$ average class delays, given the $N$ class loads, were derived by Coffman and Mitrani in [9]. Under general assumptions[3], a set of $N$ average delays $\{\bar{d}_i, i = 1, \ldots, N\}$ is feasible if and only if the following $2^N - 2$ inequalities hold:

$$\sum_{i \in \phi} \lambda_i \bar{d}_i \bar{L}_i \geq \bar{d}_\phi^{\mathrm{SP}} \sum_{i \in \phi} \lambda_i \bar{L}_i = \bar{q}_\phi^{\mathrm{SP}} \qquad \text{for all } \phi \in \Phi \quad (6)$$

where $\Phi$ is the set of $2^N - 2$ nonempty proper subsets of the set $\{1, 2, \ldots, N\}$. $\bar{q}_\phi^{\mathrm{SP}}$ and $\bar{d}_\phi^{\mathrm{SP}}$ are the average backlog and the average delay, respectively, of the traffic in the set $\phi$, if $\phi$ was given strict priority over all other classes. The Coffman-Mitrani inequalities state that any set of classes $\phi$ has a lower bound on its average backlog and this minimum value results when the traffic in $\phi$ is serviced with the highest priority in a Strict Priority (SP) scheduler.

In the remainder of this section, we make the additional assumption that all classes have the same packet size distribution, and we set the average packet size to $\bar{L} = \bar{L}_i = 1$. With this assumption, Regnier [27] extended the Coffman–Mitrani results showing that the necessary and sufficient feasibility conditions of (6) can be reduced from $2^N - 2$ to the following $N - 1$ inequalities:

$$\sum_{i=k}^{N} \lambda_i \bar{d}_i \geq \bar{d}_{k,N}^{\mathrm{SP}} \sum_{i=k}^{N} \lambda_i = \bar{q}_{k,N}^{\mathrm{SP}} \qquad k = 2, \ldots, N \quad (7)$$

where $\bar{q}_{k,N}^{\mathrm{SP}}$ and $\bar{d}_{k,N}^{\mathrm{SP}}$ are the average backlog $\bar{q}_\phi^{\mathrm{SP}}$ and average delay $\bar{d}_\phi^{\mathrm{SP}}$, respectively, for $\phi = \{k, \ldots, N\}$. Regnier's inequalities impose a lower bound on the average backlog of the $N-1$ subsets of the $k$ highest classes $(k = 1, \ldots, N-1)$.

Let $\bar{d}_i^{\mathrm{SP}}$ be the average delay of Class-$i$ in the SP scheduler which services class $m$ with higher priority than class $n$ for $m > n$ (i.e., Class-N is given highest priority and Class-1 is given lowest priority). With this notation, we have that $\bar{d}_{k,N}^{\mathrm{SP}} \sum_{i=k}^{N} \lambda_i = \sum_{i=k}^{N} \lambda_i \bar{d}_i^{\mathrm{SP}}$, and so the Regnier inequalities become

$$\sum_{i=k}^{N} \lambda_i \bar{d}_i \geq \sum_{i=k}^{N} \lambda_i \bar{d}_i^{\mathrm{SP}} \qquad k = 2, \ldots, N. \quad (8)$$

From (8) and (4), we can now show that the necessary and sufficient condition for the feasibility of the PDD model when $N = 2$ is

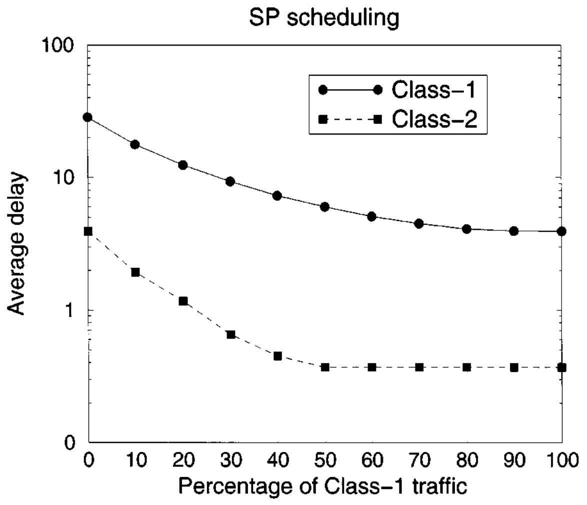
$$\bar{d}_2 = \frac{\delta_2 \bar{q}_{\mathrm{ag}}}{\lambda_1 + \delta_2 \lambda_2} \geq \bar{d}_2^{\mathrm{SP}}. \quad (9)$$

Since $\bar{q}_{\mathrm{ag}} = \lambda_1 \bar{d}_1^{\mathrm{SP}} + \lambda_2 \bar{d}_2^{\mathrm{SP}}$, it is easy to show that the feasibility condition becomes

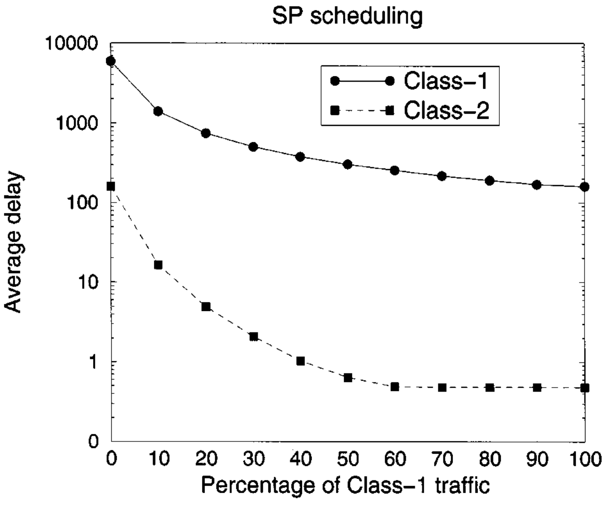
$$\delta_2 \geq \frac{\bar{d}_2^{\mathrm{SP}}}{\bar{d}_1^{\mathrm{SP}}} \quad \text{or} \quad \frac{1}{\delta_2} \leq \frac{\bar{d}_1^{\mathrm{SP}}}{\bar{d}_2^{\mathrm{SP}}}. \quad (10)$$

So, a given delay ratio $\delta_2 < 1$ between Class-2 and Class-1 is feasible if and only if the corresponding average delay ratio in the SP scheduler is not larger than $\delta_2$. Note that the average delays of the two classes in SP are a function of the input rate $\lambda$, of the load distribution $(\lambda_1, \lambda_2)$, and of the aggregate average backlog $\bar{q}_{\mathrm{ag}}$, and so the feasibility of the PDD model depends on the same factors.

[3]Most of [9] assumes Poisson arrivals. The particular result that we include here holds, as [9] also states, for general interarrival and packet size distributions that are different between classes.



(a)



(b)

Fig. 1. Average delays with SP as a function of the load distribution.

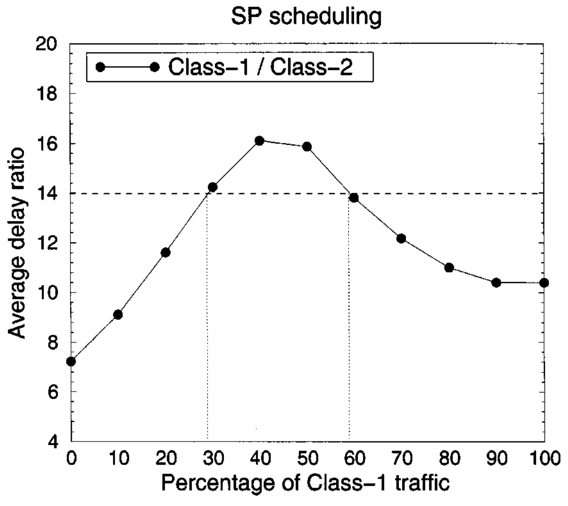
When $N > 2$, the feasibility conditions for the PDD model are the following $N - 1$ inequalities:

$$\sum_{i=k}^{N} \lambda_i \delta_i \geq \frac{S}{\bar{q}_{\mathrm{ag}}} \sum_{i=k}^{N} \lambda_i \bar{d}_i^{\mathrm{SP}} \qquad k = 2, \ldots, N \quad (11)$$

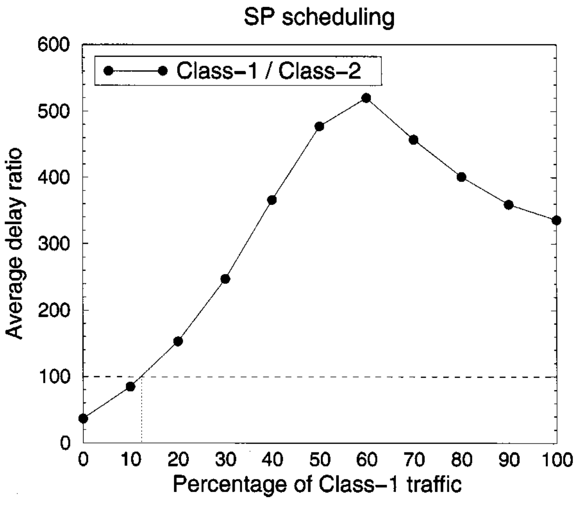
where $S = \sum_{i=1}^{N} \lambda_i \delta_i$ and $\bar{q}_{\mathrm{ag}} = \sum_{i=1}^{N} \lambda_i \bar{d}_i^{\mathrm{SP}}$.

To illustrate the feasibility conditions graphically, Fig. 1 shows the per-class average delays with SP for the case of two classes in two utilization points ($u = 75\%$ and $u = 95\%$). Also, Fig. 2 shows the average delay ratios $\bar{d}_1^{\mathrm{SP}} / \bar{d}_2^{\mathrm{SP}}$ for the same load conditions. These graphs resulted from a simulation of the SP scheduler[4]. When $u = 75\%$ [Fig. 2(a)] and the load distribution is $(\lambda_1, \lambda_2) = (70, 30)$, the DDP $1/\delta_2 = 8$ is feasible, while the DDP $1/\delta_2 = 16$ is infeasible. On the other hand, if $1/\delta_2 = 14$, the load distribution must be such that $0.3 < \lambda_1/\lambda < 0.6$ in order for the PDD model to be feasible. When $u = 95\%$ [Fig. 2(b)], the DDP $1/\delta_2 = 32$ is always

[4]The simulation model and parameters that we used in this paper is described in Appendix II.



(a)



(b)

Fig. 2. Average delay ratios with SP as a function of the load distribution.

feasible, while the DDP $1/\delta_2 = 100$ is only feasible when at least 12% of the traffic belongs to Class-1.

For some queueing models, the average delays with the SP scheduler can be analytically computed. For the M/G/1 queueing model with $N = 2$ classes, for instance, we have that $\bar{d}_1^{\rm SP}/\bar{d}_2^{\rm SP} = 1/(1-\lambda)$ [4]. This result, which is based on Poisson arrivals, shows that the feasibility range increases as the aggregate load increases, but it hides the dependency of the feasibility range on the class load distribution.

If the number of classes is $N > 2$, such a graphical interpretation of the feasibility conditions is not straightforward. One can still use (11) to examine numerically if the $N-1$ conditions are satisfied for the given DDPs, load distribution, and aggregate backlog. In order to examine the feasibility of the PDD model, we need to know the average delays $\bar{d}_k^{\rm SP}$ that would result with a SP scheduler. If the traffic does not conform well to a mathematically tractable queueing model, one needs to estimate or measure them experimentally. Most existing routers provide an SP scheduler, and so a network provider can measure the class average delays $\bar{d}_k^{\rm SP}$ deploying the SP scheduler for short time intervals in the actual workload. Alternatively, if the SP deployment is considered harmful due to its starvation effects, the delays $\bar{d}_k^{\rm SP}$ can be estimated emulating an SP scheduler with the actual workload of the link.

## III. PAD Scheduling

Another way to interpret the PDD model is that the *normalized average delays*, defined as $\tilde{d}_i = \bar{d}_i/\delta_i$, must be equal in all classes, i.e.,

$$\tilde{d}_i = \frac{\bar{d}_i}{\delta_i} = \frac{\bar{d}_j}{\delta_j} = \tilde{d}_j \quad 1 \le i, \quad j \le N. \tag{12}$$

A scheduler that aims to equalize the normalized average delays among all classes is described next. We refer to this algorithm as PAD scheduling.

Let $B(t)$ be the set of backlogged classes at time $t, D_i(t)$ be the sequence of Class-$i$ packets that departed before time $t$ (in order of their departure), and $d_i^m$ be the delay of the $m$'th packet in $D_i(t)$. Assuming that there was at least one departure from class $i$ before $t$, the normalized average delay of class $i$ at $t$ is

$$\tilde{d}_i(t) = \frac{1}{\delta_i} \frac{\sum_{m=1}^{|D_i(t)|} d_i^m}{|D_i(t)|} = \frac{1}{\delta_i}\frac{S_i}{P_i} \tag{13}$$

where $S_i$ is the sum of queueing delays of all packets in $D_i(t)$, and $P_i = |D_i(t)|$ is the number of packets in $D_i(t)$.

Suppose that a packet has to be selected for transmission at time $t$. PAD chooses the backlogged class $j$ with the maximum normalized average delay,

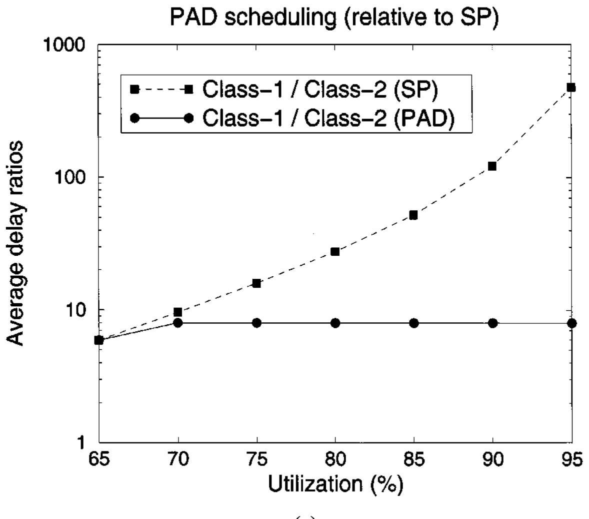
$$j = \arg\max_{i \in B(t)} \tilde{d}_i(t). \tag{14}$$

The packet at the head of queue $j$ is transmitted, after a queueing delay $d_j^{P_j+1}$. The variables $S_j$ and $P_j$ are then updated as $S_j = S_j + d_j^{P_j+1}$ and $P_j = P_j + 1$, and the new normalized average delay $\tilde{d}_j(t)$ is computed from (13).

The selection of the maximum $\tilde{d}_i(t)$ term, as in (14), requires at most $N-1$ comparisons, which is a minor overhead for the small number of classes that we consider here. The main computational overhead of PAD is a division, for the calculation of $S_j/P_j$, after each packet departure.[5] With current commodity microprocessors (≈500 MHz), a floating-point division takes less than 100 ns (≈50 cycles). This delay would not be an issue for network interfaces of up to 1 Gbps, which transmit a (minimum-size) 40-byte packet in about 320 ns. For even faster network interfaces, one can avoid the division operation and compute $\tilde{d}_i(t)$ using an exponential running-average of the form $\tilde{d}_i(t) = (1-2^{-w})\tilde{d}_i(t) + 2^{-w} d_i^{P_i+1}/\delta_i$, where $w$ is a positive integer.

The basic idea in PAD is that if a packet is serviced from class $j$ with the maximum normalized average delay, the delay of that packet will not increase any more (since it is serviced), and thus the increase of $S_j$ due to that packet will be minimized. So, servicing a packet from class $j$ tends to reduce the difference of $\tilde{d}_j(t)$ from the normalized average delays of the other classes. In the long run, if the scheduler always minimizes the differences between the normalized average delays in this manner,

[5]Obviously, the factor $1/\delta_i$ can be precomputed and does not require a division. If it is a power of two, it does not require a multiplication either.



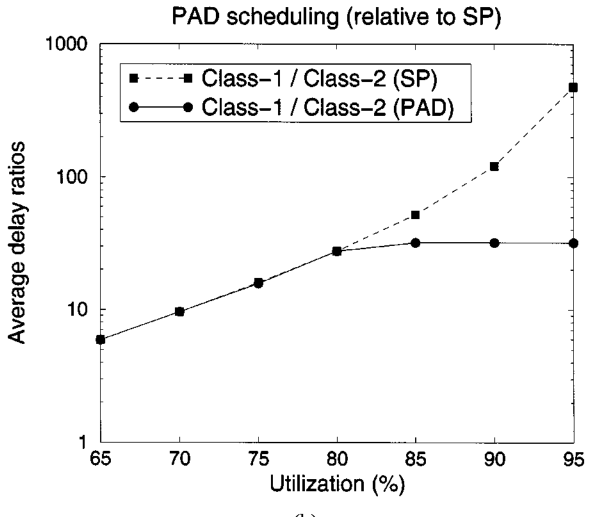
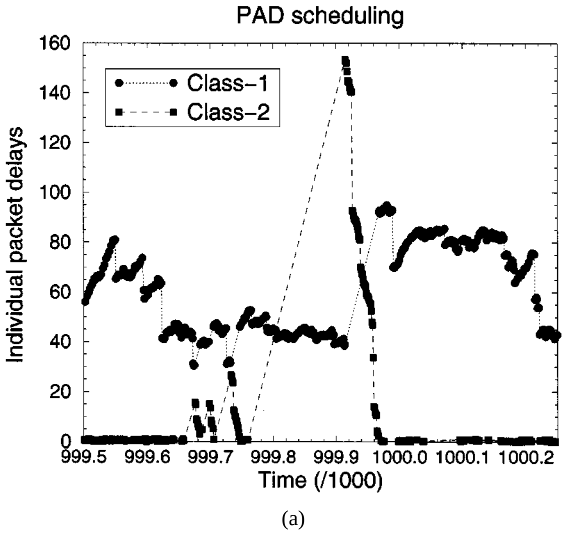


Fig. 3. Average delay ratios with PAD and SP as a function of the utilization.



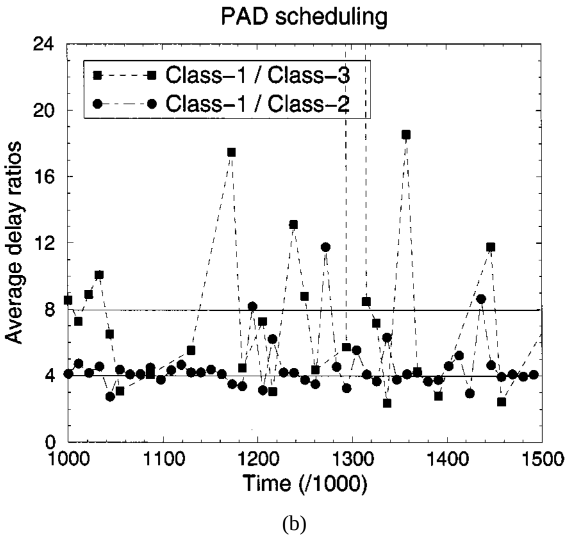


Fig. 4. Individual packet delays and short-term delay ratios with PAD.

we expect that the normalized average delays will be about the same.

Our simulation experiments have shown that PAD can achieve the PDD model constraints, when the given set of DDPs is feasible. To illustrate this point, Fig. 3 shows typical simulation results for $N = 2$ classes. The two graphs show the ratio $\bar{d}_1/\bar{d}_2$ of the average delays with PAD and SP for two DDP selections and for a uniform load distribution $(\lambda_1 = \lambda_2)$, as a function of the utilization $u$. Recall from (10) that in the case of two classes, a given DDP $1/\delta_2$ is feasible if and only if it is lower than the ratio $\bar{d}_1/\bar{d}_2$ in SP. As shown in Fig. 3(a), PAD achieves the specified delay ratio $1/\delta_2 = 8$ in all utilization points that this DDP is feasible. The resulting delay ratio is less than eight only when $u < 70\%$, but SP does not lead to a larger delay ratio either, meaning that this DDP is infeasible in that utilization range. Similarly, in Fig. 3(b), PAD achieves the specified delay ratio $1/\delta_2 = 32$ when the utilization is higher than about 80%, which is also when this DDP is feasible.

PAD, however, exhibits a pathological behavior in short timescales. To illustrate this behavior, Fig. 4(a) shows the queueing delays of individual packets at their departure instant. The utilization in this simulation is $u = 90\%$ and the load distribution to $(\lambda_1, \lambda_2) = (70, 30)$. Note that during most of the time, the Class-2 packets depart with minor queueing delays. Occasionally though, as it occurs shortly after time 999.900, some Class-2 packets depart with significantly larger queueing delays, even larger than the Class-1 delays in the corresponding time frame. Let us analyze in detail what happens in the time period 999.760–999.960. Just before 999.760, the Class-2 queue is almost empty and its queueing delays are close to zero. From 999.760 to 999.920 (phase-1), $\tilde{d}_2$ is less than $\tilde{d}_1$, and there are no departures from Class-2. The arriving Class-2 packets during phase-1 accumulate large waiting times, up to the duration of this phase (160 time units), but the normalized average delay $\tilde{d}_2$ does not change since there are no departures from that class. At about 999.920, $\tilde{d}_1$ becomes less than $\tilde{d}_2$, after servicing exclusively packets of that class during phase-1. From that point and until 999.960 (phase-2), all the backlogged Class-2 packets from phase-1 depart back-to-back without interventions from Class-1 packets. At about 999.960, the two normalized average delays become comparable, and the scheduler starts servicing packets again in a more uniform manner. This example shows that because PAD is unaware of the waiting times of backlogged packets, it occasionally allows higher classes to experience much larger delays than their long-term average delays, or the queueing delays of lower classes.

It is also instructive to examine the ratios of short-term average delays between classes with PAD, in relatively short timescales. Fig. 4(b) shows the ratios of average delays, measured in every $K = 10\,000$ successive packet departures, for $N = 3$ classes. The utilization is $u = 90\%$, and the load distribution is $(\lambda_1, \lambda_2, \lambda_3) = (50, 30, 20)$. Note that PAD does a poor job in meeting the PDD constraints in this timescale. This should be expected, since PAD attempts to equalize the *long-term* normalized average delays and not the normalized average delays in the *last $K$ departures*. Another important point in this graph is that there are several time periods in which the ratio $\bar{d}_1/\bar{d}_2$ is larger than the ratio $\bar{d}_1/\bar{d}_3$, meaning that Class-3 experiences higher delays than Class-2. These are incidents of unpredictable delay differentiation. A good scheduler for the PDD model should not only achieve or closely approximate the constraints of (2), but it should also provide predictable delay differentiation in short timescales. A scheduler that emphasizes on this predictability issue is studied next.

## IV. WTP Scheduling

The WTP scheduling algorithm was first studied by L. Kleinrock in 1964 [19] under the name *Time-Dependent Priorities*. A packet is assigned a priority that increases proportionally to the packet's waiting time. Higher classes have larger priority-increase factors. The packet with the highest priority is serviced first (in nonpreemptive order). In the following, we describe WTP in a different way to highlight its similarities and differences with PAD.

Suppose that class $i$ is backlogged at time $t$, and that $w_i(t)$ is the *head waiting time* of class $i$ at $t$, i.e., the waiting time of the packet at the head of class $i$ at $t$. We define the *normalized head waiting time* of class $i$ at $t$ as

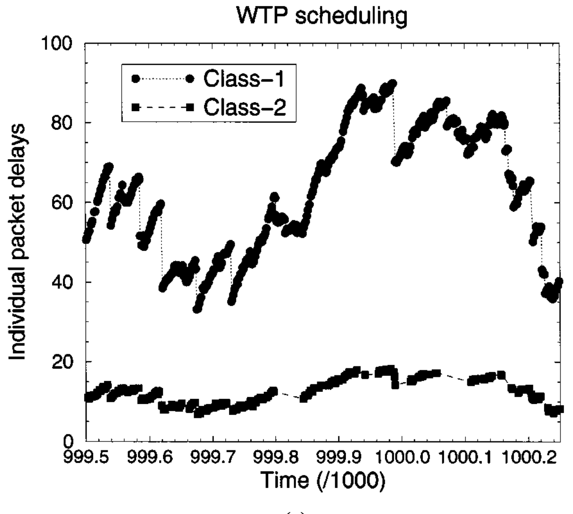
$$\tilde{w}_i(t) = w_i(t)/\delta_i. \tag{15}$$

Every time a packet is to be transmitted, the WTP scheduler selects the backlogged class $j$ with the maximum normalized head waiting time,

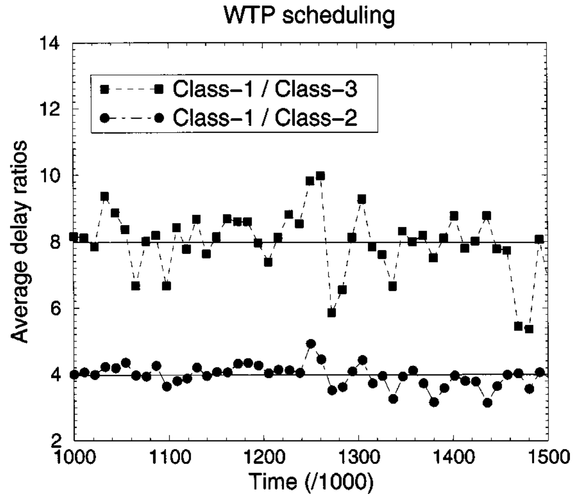
$$j = \arg\max_{i \in B(t)} \tilde{w}_i(t). \tag{16}$$

The similarities with PAD are now obvious. In the same way that PAD chooses for service the class with the maximum normalized average delay, WTP chooses for service the class with the maximum normalized head waiting time. PAD attempts to minimize in this manner the differences of the class normalized average delays, while WTP attempts to minimize the differences between the normalized waiting times of successively departing packets. Suppose, hypothetically, that WTP manages to always reduce these differences to zero. The queueing delays of successively departing packets will be then proportional to the given DDPs,

$$\frac{d^m}{d^{m+1}} = \frac{\delta_{c(m)}}{\delta_{c(m+1)}} \tag{17}$$



(a)



(b)

Fig. 5. Individual packet delays and short-term delay ratios with WTP.

where $d^m$ and $c(m)$ are the queueing delay and the class, respectively, of the $m$th departing packet. Of course, this goal is not always feasible. Equation (17) shows, though, that WTP attempts to achieve a proportional differentiation between the delays of successive packet departures.

In terms of scalability and performance, WTP requires at most $N - 1$ comparisons for each packet transmission, which is a minor overhead for the eight classes or so that we consider. To avoid the multiplication of the waiting time $w_i(t)$ with the factor $1/\delta_i$, the waiting time can be increased by $1/\delta_i$ after each time unit. An implementation requirement, which also applies to PAD, is that packets have to be timestamped upon arrival so that their delays can be measured; we do not expect this requirement to be an important difficulty in practice.

WTP is an excellent scheduler in terms of providing higher classes with lower delays in short timescales. To illustrate the behavior of WTP in short timescales, the two graphs (a) and (b) of Fig. 5 show the individual delays of successive packet departures, and the ratios of short-term class delays averaged in every $K = 10\,000$ packets, respectively. To compare WTP and PAD, these graphs refer to the same time interval and the same traffic stream as the graphs in Fig. 4. First, note that in Fig. 5(a)

WTP services the Class-2 packets with lower queueing delays, even in the shortest timescales of successive packet departures. A more careful examination of that graph also shows that the individual packet delays in Class-2 are approximately proportional to the corresponding Class-1 packet delays, and that the proportionality factor is about five, which is the specified DDP $(1/\delta_2 = 5)$. In other words, as stated in (17), WTP attempts to provide proportional delay differentiation between successive packet departures, and this objective is closely approximated when the delays are sufficiently large.

Second, note that in Fig. 5(b) the correct class ordering is also maintained when the delays are measured over short timescales, in this graph every $K = 10\,000$ packets. Figs. 9 and 10 in the next section, show that this is also true in a wide span of timescales, ranging from a few packets to many thousands of packets. So, with WTP the average queueing delays in higher classes are smaller than the average delays in lower classes, independent of the location or the duration of the averaging time window. Also note that the measured delay ratios in Fig. 5(b) vary around the specified DDPs, $1/\delta_2 = 4$ and $1/\delta_3 = 8$, even though the deviations are sometimes significant. In summary, WTP provides predictable delay differentiation, and it approximates the PDD model in short timescales. Does WTP, however, meet the PDD constraints of (1)?

In the special case of Poisson arrivals, we can show analytically that WTP converges to the PDD model as the utilization approaches 100%. The following result is proved in Appendix III.

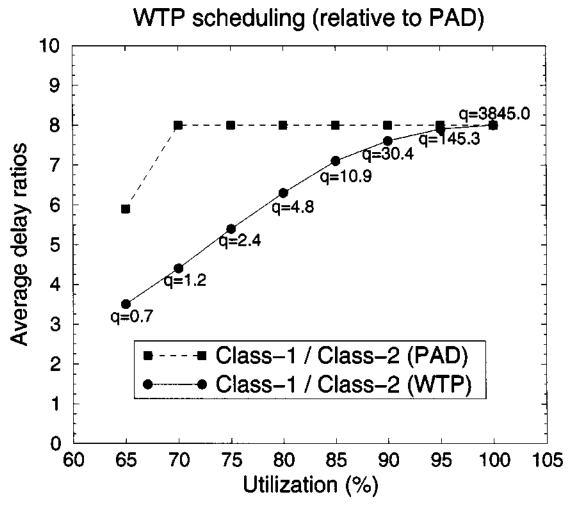
*Proposition 1:* If the packet arrivals in each class are generated from a Poisson distribution, the WTP scheduler converges to the PDD model of (1) as the utilization $u$ tends to 100%

$$\frac{\bar{d}_i^{\mathrm{WTP}}}{\bar{d}_j^{\mathrm{WTP}}} \rightarrow \frac{\delta_i}{\delta_j} \quad \text{as } u \rightarrow 1. \tag{18}$$

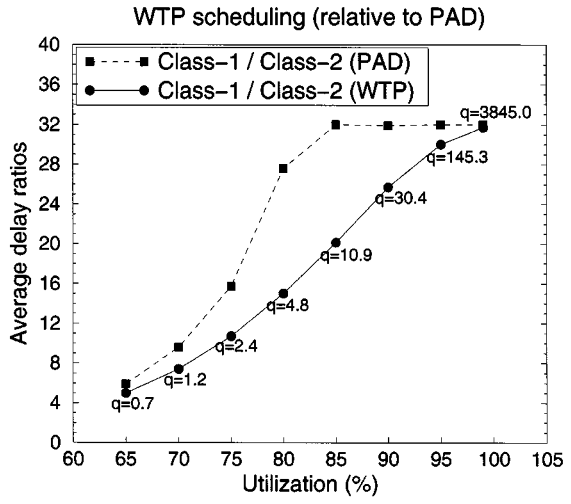
Since the Poisson assumption has been shown to be often invalid in packet networks, we have also examined the validity of (18) with simulations, using the infinite-variance Pareto interarrival distribution. The general observation from our empirical study is that WTP meets the PDD model as the aggregate backlog in the link $\bar{q}_{\mathrm{ag}}$ tends to infinity.

How close to 'infinity' does the aggregate backlog have to be, though, in order for WTP to closely approximate the PDD model? Fig. 6 shows the achieved delay ratios for the case of two classes and for a uniform load distribution, as a function of the utilization. The aggregate backlog for each utilization point is also shown (denoted by $q$ in the graphs). Note that the aggregate backlog does not depend on the load distribution when all classes have the same statistical characteristics, or on the specified DDPs. The corresponding curves for PAD are also shown, for comparison purposes.

Fig. 6 shows that WTP tends to the PDD model as the utilization approaches 100% and the aggregate backlog increases. For a moderate delay ratio, such as $1/\delta_2 = 8$, the required average backlog for a close approximation of the specified DDP is a few tens of packets. This is not an unusual average backlog for heavily utilized, high-speed links [28]. In order to closely approximate a more drastic delay differentiation with WTP however, such as $1/\delta_2 = 32$, an average backlog of hundreds of



(a)



(b)

Fig. 6. Average delay ratios with WTP and PAD as a function of the utilization.

packets is needed. This may be impractical, especially when there are not enough buffers, or when the network operators prefer to drop packets (and so limit their maximum delay) instead of queueing them for hundreds of packet transmission times. When the utilization is less than about 85%, WTP shows a large deviation from the PDD model, either because the specified DDP is infeasible (as in Fig. 6(b) for $u < 85\%$), or simply because WTP is not a particularly good scheduler in terms of meeting the long-term average delay ratios of the PDD model.

As shown in Section III, PAD attempts to minimize the differences between normalized average class delays. This objective makes PAD capable of meeting the PDD model when the given DDPs are feasible. WTP, on the other hand, attempts to minimize the differences between normalized head waiting times. As a consequence of this behavior, WTP provides higher classes with lower delays even in the shortest timescales. Can we combine the operation of these two scheduling algorithms in order to create a scheduler that is both close to the PDD model, when the DDPs are feasible, and also that provides a predictable delay differentiation in short timescales? This is the objective of the HPD scheduling discipline, described next.

## V. HPD Scheduling

HPD, just like PAD and WTP, maintains a delay metric for each class $i$, normalized by the corresponding DDP $\delta_i$. Specifically, if $\tilde{d}_i(t)$ is the normalized average delay in class $i$ at $t$, as defined in (13), and $\tilde{w}_i(t)$ is the normalized head waiting time in class $i$ at $t$, as defined in (15), the *normalized hybrid delay* in HPD is

$$\tilde{h}_i(t) = g\tilde{d}_i(t) + (1-g)\tilde{w}_i(t) \qquad (19)$$

where $g$ is the *HPD parameter* $(0 \leq g \leq 1)$. When a packet is to be transmitted, HPD chooses the backlogged class $j$ with the maximum normalized hybrid delay

$$j = \arg\max_{i\in B(t)} \tilde{h}_i(t). \qquad (20)$$

When $g$ is zero HPD becomes equivalent to WTP, while when $g$ is one HPD becomes equivalent to PAD. For other values of $g$, HPD combines the operation of PAD and WTP resulting in a hybrid behavior.

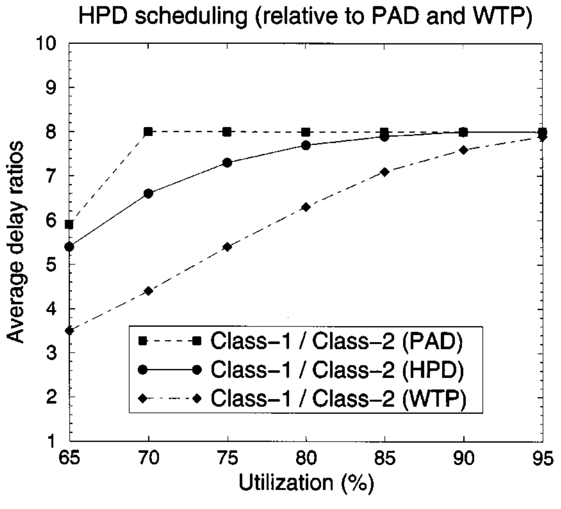
An 'optimal' selection of $g$ would attempt to maximize a given performance metric that evaluates HPD's ability to approximate the PDD model, and at the same time to provide predictable delay differentiation in short timescales. We do not pursue such an optimal selection of $g$ here, however. Instead, through an empirical study, we found that $g = 0.875$ is a 'good' value in the tradeoff between the behaviors of PAD and WTP. Simulation results supporting this choice are given at the end of this section. Note that $0.875 = 1–2^{-3}$, which also means that the multiplication with $g$ and $1-g$ can be replaced with shift and subtract operations.

In the following, we examine different aspects of the HPD behavior and compare it with PAD and WTP using simulation results.

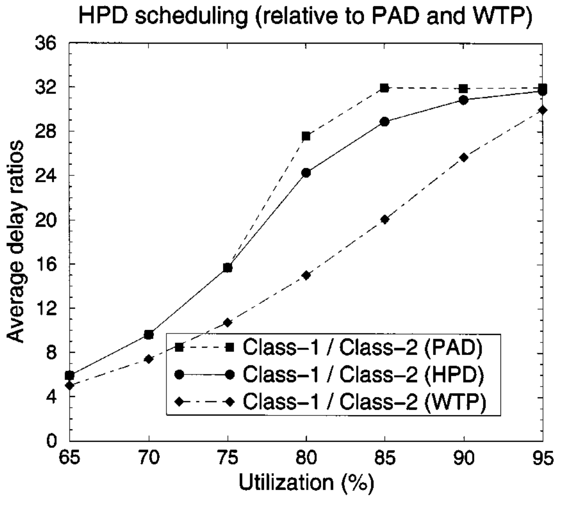
**Effect of the aggregate load on the average delay ratios:** Fig. 7 shows the effect of the utilization $u$ on the HPD average delay ratios. Results are shown for two classes, when the specified delay ratio is either $1/\delta_2 = 8$ or $1/\delta_2 = 32$. The load distribution between the two classes is uniform. For comparison purposes we also show the PAD and WTP curves.

As the utilization increases, all three schedulers tend to the specified delay ratios. The differences between HPD and PAD in heavy load conditions, above 90% or so, are minor. When the utilization is between 70% to 90%, HPD is significantly closer to the PDD constraints than WTP. The largest relative error between HPD and PAD in this operating region is about 20% in these graphs. This error can be further reduced with a higher value of the HPD parameter $g$, if the network operator values the importance of meeting the specified DDPs more than providing a consistent delay ordering between classes in short timescales. In light load conditions, less than about 70% or 75%, the PDD model becomes infeasible, and so the PAD scheduler cannot achieve the specified delay ratios either.

**Effect of the load distribution on the average delay ratios:** Fig. 8 shows the effect of the class load distribution on the HPD average delay ratios. Results are shown for four classes and seven class load distributions, when the utilization is 75% and 95%. The DDPs are: $\delta_1/\delta_2 = \delta_2/\delta_3 = \delta_3/\delta_4 = 2$, i.e., each class should have a twice as large average delay than the next higher class.



(a)

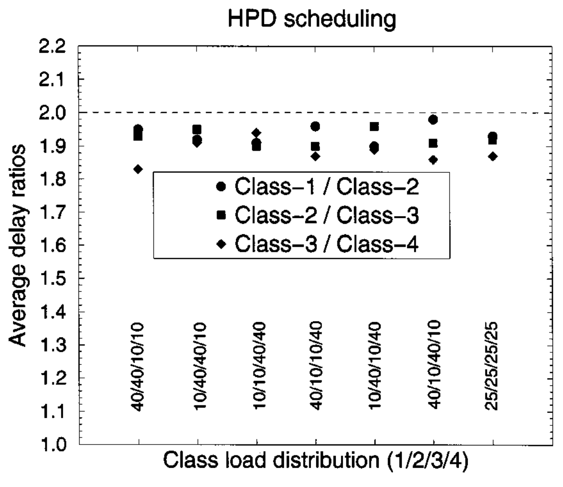


(b)

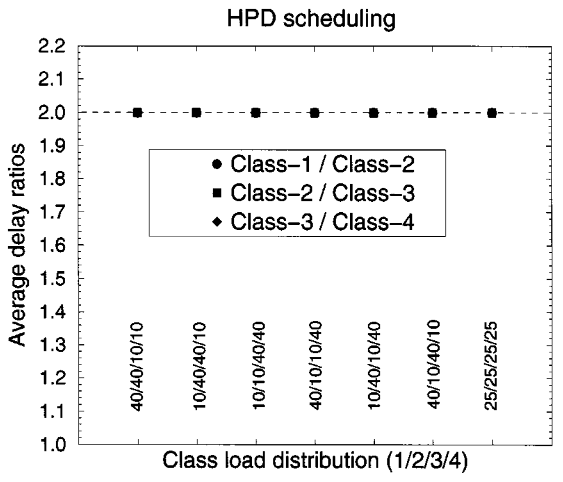
Fig. 7. Average delay ratios with HPD as a function of the utilization.

Ideally, HPD should meet the PDD model independent of the class load distribution. The set of DDPs is feasible in both load conditions. When the utilization is 75% [Fig. 8(a)], HPD closely approximates the specified delay ratio between successive classes, and the deviations from the PDD model are less than 10%. In heavy load conditions [Fig. 8(b)], HPD accurately meets the specified delay ratios in all load distributions. In general, HPD manages to closely approximate the PDD model, when it is feasible, independent of the class load distribution. The approximation errors increase as the utilization decreases, and as the required delay differentiation between classes becomes more extreme.

**Delay differentiation with HPD and WTP in short timescales:** In evaluating the predictability of a scheduler, our primary focus is to examine whether higher classes encounter lower delays in all timescales. It has already been illustrated in the previous graphs that HPD provides higher classes with lower delays, but only in terms of long-term averages. Is the delay ordering between classes, however, also met in short timescales?

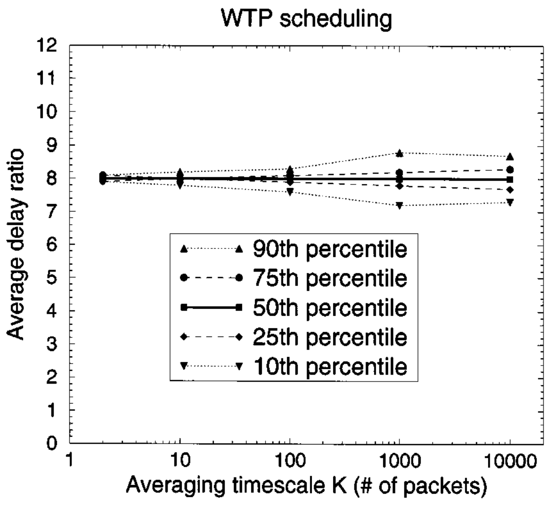


(a)

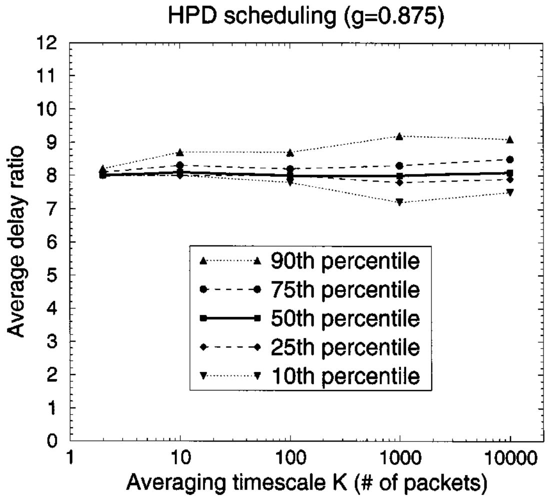


(b)

Fig. 8. Average delay ratios with HPD as a function of the load distribution.



(a)



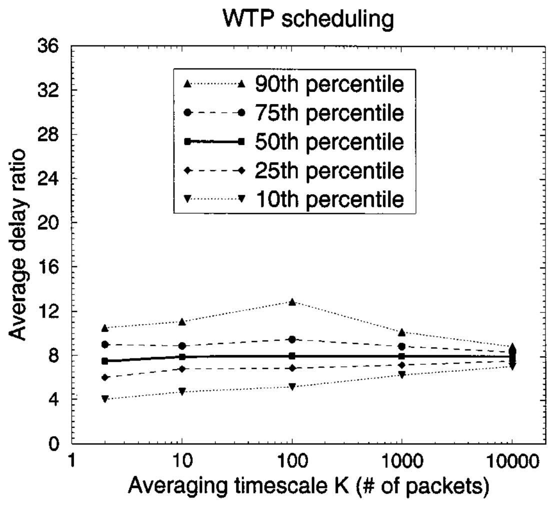
(b)

Fig. 9. Percentiles of the average delay ratios with WTP and HPD as a function of the averaging timescale $K (u = 95\%)$.

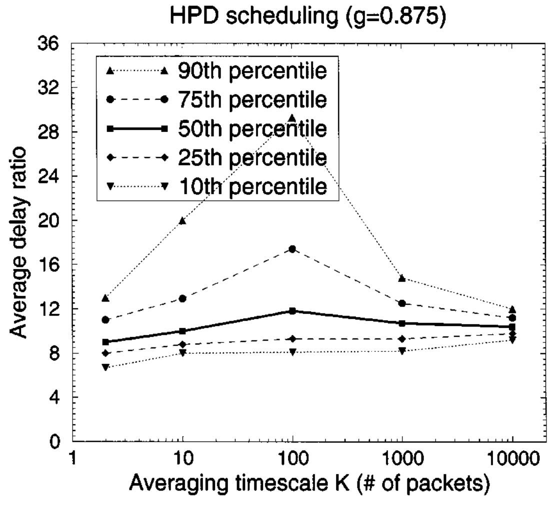
Figs. 9 and 10 investigate the delay differentiation that HPD and WTP produce in short timescales of different length. The graphs in these two figures show five percentiles of the distribution of delay ratios between two classes, when the delay ratios are measured in successive time windows of $K$ packet departures. $K$ determines the timescale in which we measure the class delays and compute their ratio. In these experiments, $K$ ranges from successive packet departures $(K = 2)$ to several thousands of packet departures $(K = 10\,000)$. The five percentiles shown give a comprehensive view of the distribution of the resulting delay ratios, providing the median delay ratio (50th percentile), the 25th and 75th percentiles, as well as two tail delay ratios (10th and 90th percentiles). Note that when there are no packet departures from both classes in a time period of $K$ packet departures, a delay ratio is not measured. Also, the duration of the simulation runs is adjusted so that we get approximately the same number of delay ratios for all values of $K$. In both figures, the specified DDP is $\delta_2 = 1/8$ and the class load distribution is uniform.

Fig. 9 shows the results of these experiments in heavy load conditions $(u = 95\%)$. As shown, the resulting distributions of delay ratios are quite narrow, centered around the specified DDPs $\delta_1/\delta_2 = 8$, in both HPD and WTP. This means that in heavy load conditions, HPD inherits the excellent predictability of WTP, i.e., it provides lower delays to higher classes, and it approximates closely the PDD model. This behavior is consistent in a range of timescales, from successive departures to several thousands of packet departures. In Fig. 10, on the other hand, the utilization is $u = 80\%$. In this case, the HPD delay ratio distributions are significantly wider than the WTP distributions. Additionally, the HPD distributions are not centered exactly around the specified $\delta_1/\delta_2 = 8$, but they indicate slightly higher delay ratios. Even if HPD does not approximate the PDD model as closely as WTP does in moderate or low load conditions, however, it still provides lower delays to the higher class, and so it produces predictable delay differentiation.

**Choosing a 'good' HPD parameter value:** Fig. 11 shows the effect of the HPD parameter $g$ on the ratio of average delays for two classes. In heavy load conditions $(u = 95\%)$, the selection of $g$ does not matter much, and the PDD model is closely approximated for practically any $g$. The reason is that in heavy load conditions both PAD and WTP meet the PDD model, and so does HPD. In lower load conditions $(u = 75\%)$ though, $g$ needs to be close to one in order for HPD to approximate the PDD model well. Note that the dependency of the delay ratio

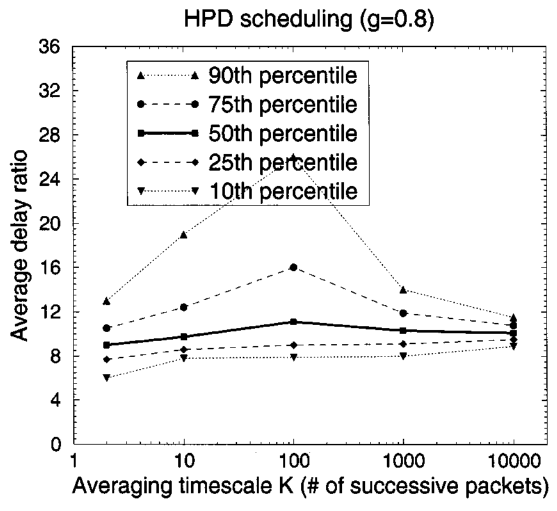


(a)

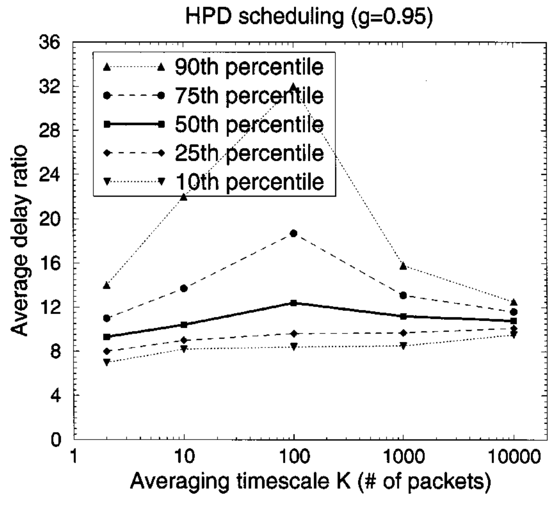


(b)

Fig. 10. Percentiles of the average delay ratios with WTP and HPD as a function of the averaging timescale $K(u = 80\%)$.



(a)



(b)

Fig. 12. Percentiles of the average delay ratios with HPD for two values of $g$, as a function of the averaging timescale $K(u = 80\%)$.

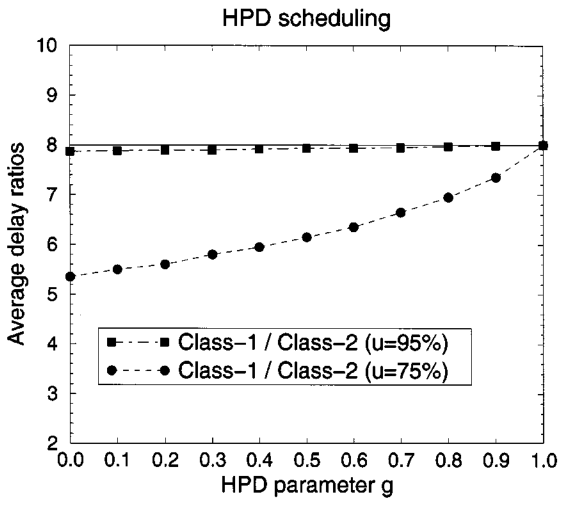


Fig. 11. Effect of the HPD parameter $g$ on the average delay ratios $(1/\delta_2 = 8)$.

on $g$ is convex, meaning that the approximation error decreases faster as $g$ approaches one.

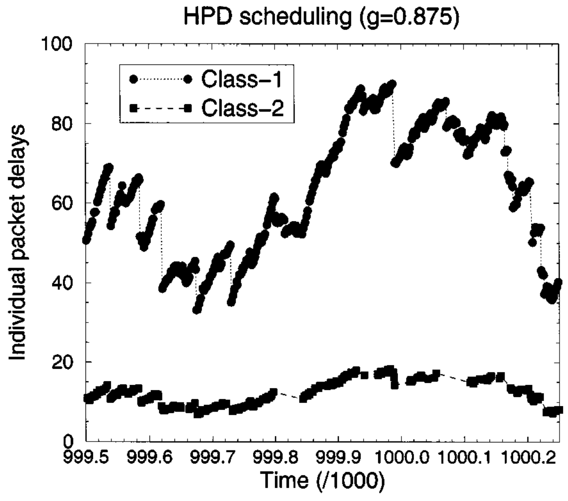
A good value of $g$ has to also take into account the predictability of HPD. Fig. 12 shows five percentiles for the average delay ratios with HPD for $g = 0.8$ and $g = 0.95$, as it was done in Fig. 10(b) for $g = 0.875$. The class load distribution is uniform, and the utilization is $u = 80\%$. Note that as $g$ decreases, the width of the resulting delay ratio distributions decreases as well. This is expected, since as $g$ decreases, HPD becomes closer to WTP. As a good and practical tradeoff between approximating the PDD model and achieving consistent delay differentiation in short timescales, we choose $g = 0.875$.

To further illustrate that HPD provides consistent delay differentiation in short timescales when $g = 0.875$, Fig. 13 shows the same sample paths of individual delays and short-term delay ratios for HPD, as Fig. 5 shows for WTP, and Fig. 4 shows for PAD. The HPD parameter is set to $g = 0.875$. Comparing carefully the two graphs in Figs. 13(a) and 5(a) shows that, with $g = 0.875$, HPD provides almost indistinguishable results with WTP.

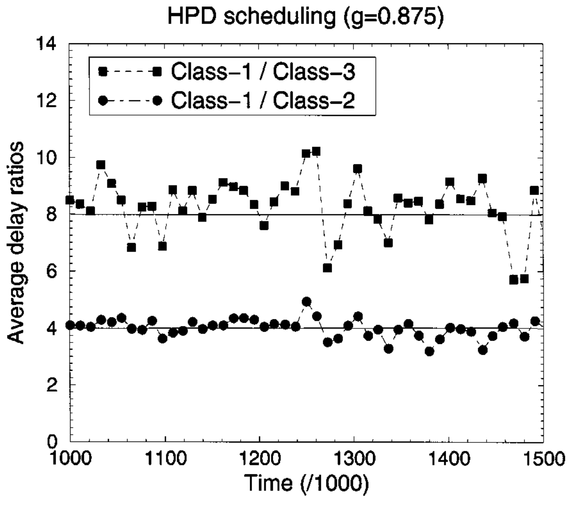
## VI. Related Work

### A. Link Sharing Schedulers

Several packet schedulers aim to provide each class with a minimum bandwidth share of the link's capacity. Examples of

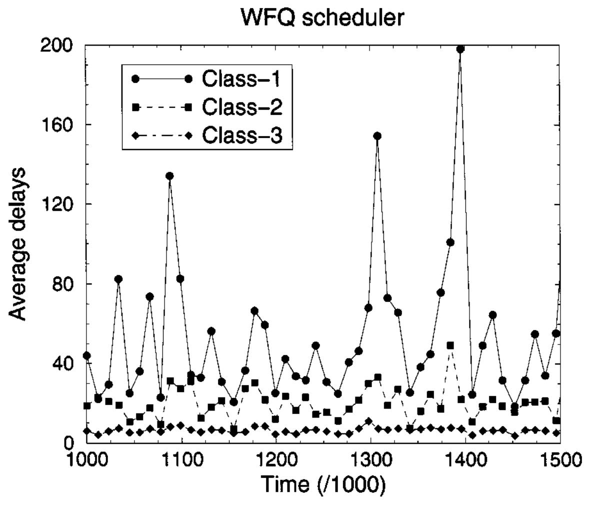


(a)

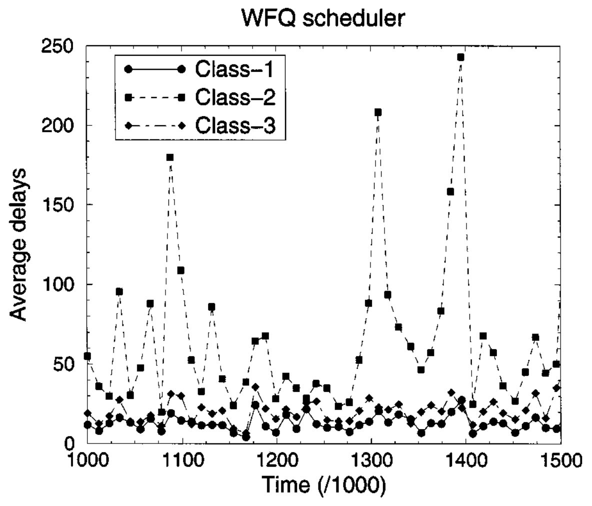


(b)

Fig. 13. Individual packet delays and short-term delay ratios with HPD.



(a)



(b)

Fig. 14. Average delays with a three-class WFQ scheduler in two slightly different load distributions.

such schedulers are the packet-based approximations of generalized processor sharing (GPS) (such as WFQ [10]), as well as the class-based queueing (CBQ) [16] and hierarchical packet fair queueing (H-PFQ) [1] link sharing schedulers. Such schedulers have been proposed for providing relative delay differentiation. For example, [17] proposes the use of such a scheduler to implement the *Olympic service model*, which consists of the 'Gold', 'Silver', and 'Bronze' classes.

A drawback of the link sharing schedulers is that slight changes in the class load distribution affect dramatically the resulting delay differentiation. Fig. 14 shows short-term average delays in a three-class WFQ scheduler, measured in every $K = 10\,000$ packet departures. The WFQ weights are selected as $(w_1, w_2, w_3) = (0.5, 0.35., 0.3)$ so that, when the load distribution is $(\lambda_1, \lambda_2, \lambda_3) = (50, 30, 20)$, the average delay in Class-1 is about 50, in Class-2 about 20, and in Class-3 about 7 time units. In Fig. 14(a), notice that the selected WFQ weights create a predictable delay differentiation, providing the maximum delays to Class-1 and the lowest delays to Class-3. If, however, only 10% of the Class-1 traffic moves to Class-2 and Class-3, leading to the load distribution $(\lambda_1, \lambda_2, \lambda_3) = (40, 35, 25)$, the class ordering is violated, causing the maximum delays for Class-2 and the minimum delays for Class-1 [see Fig. 14(b)].

Obviously, such a 'priority inversion' between the offered classes would be unacceptable for the higher class users. The previous example illustrates that link sharing schedulers are quite sensitive to changes in the class load distribution. Such 'priority inversions' can occur when the load distribution is not as it was expected when the weights were selected. This can occur when the load distribution is nonstationary, or when the load distribution varies in short timescales due to traffic burstiness.

## *B. Recent Contributions in PDD*

The WTP scheduler received further attention in [20]. The authors, starting from Kleinrock's analysis of WTP [19] and assuming Poisson arrivals, derived the feasible load distribution range for a given set of DDPs. They also proposed a numerical method for calculating the scheduling weights (when $N > 2$), so that WTP can achieve a feasible set of DDPs for any given load conditions. Another scheduler for proportional delay differentiation, called *mean-delay proportional* (MDP), was proposed in [24]. MDP is similar to HPD, in the sense that it also chooses for service at time $t$ the class with the maximum normalized 'delay' metric $\tilde{D}_i(t)$. The average delay metric in MDP takes into account both the previously departed packets, and also *all the backlogged packets*. [23] and [21] proposed two proportional delay schedulers (called *proportional queue control mechanism* (PQCM) and *dynamic weighted fair queueing* (D-WFQ), respectively) which are based on the GPS rate allocation mechanism.

Other researchers proposed different relative differentiation models than the PDD model of Section II. For instance, [30] proposed the *local optimal proportional differentiation* (LOPD) scheduler, which attempts to provide proportional average delay differentiation *in each busy period*. [3] extended the PDD model in the direction of *deadline violation probabilities*, proposing a scheduler that aims for proportional delay violation probabilities among classes. Finally, [22] proposed a *joint buffer management and Scheduling* (JoBS) mechanism which determines both the scheduling order and the packet drop decisions.

## VII. Conclusion

This paper applied the general model of proportional differentiation in the context of queueing delays. The dynamics and feasibility of the PDD model were derived, and three schedulers that approximate the PDD model in heavy load conditions were designed. The three schedulers differ in the tradeoff between approximating the PDD model closely, and providing consistent delay differentiation in short timescales.

The proportional differentiation architecture has three additional components. First, it can be applied in the context of coupled delay and loss differentiation, considering a lossy packet multiplexer. The PLD model, as well as the design of appropriate packet droppers was studied in [13]. Second, the relative differentiation that the proportional model offers can be combined with dynamic class selection at the end-points, in order to meet absolute and end-to-end QoS objectives. A first investigation of this technique has appeared in [15]. Third, a network operator can use the PDD model and a scheduler such as HPD to achieve a certain average delay in each class, as described in [14]. This provisioning process determines the link capacity and the scheduling DDPs, given the desired average delay and the target traffic load in each class.

## Appendix I

In the following proofs, $S = \sum_{n=1}^{N} \lambda_n \delta_n$.

*Proof of Property (1):* Since the DDPs are positive, an increase in the average delay of one class causes an increase in the average delays of all classes. Since an increase in the average rate of the aggregate traffic stream cannot cause a decrease in all class delays, we have that

$$\frac{\partial \bar{d}_i}{\partial \lambda_j} \geq 0 \qquad 1 \leq i, \quad j \leq N. \tag{21}$$

◇

*Proof of Property (2):* From (5) we have that

$$\frac{\partial \bar{d}_i}{\partial \lambda_j} - \frac{\partial \bar{d}_i}{\partial \lambda_k} = \frac{\delta_i}{S^2}\left[S\left(\frac{\partial \bar{q}_{\text{ag}}}{\partial \lambda_j} - \frac{\partial \bar{q}_{\text{ag}}}{\partial \lambda_k}\right) + \bar{q}_{\text{ag}}(\delta_k - \delta_j)\right]. \tag{22}$$

Since the interarrivals and packet sizes follow the same distributions in all classes, the average backlog $\bar{q}_{\text{ag}}$ does not depend on the class that the traffic belongs to, and so $\partial \bar{q}_{\text{ag}}/\partial \lambda_j = \partial \bar{q}_{\text{ag}}/\partial \lambda_k$. If $k < j, \delta_k > \delta_j$, and so

$$\frac{\partial \bar{d}_i}{\partial \lambda_j} > \frac{\partial \bar{d}_i}{\partial \lambda_k} \quad \text{when} \quad k < j. \tag{23}$$

◇

*Proof of Property (3):* From (5) we have that

$$\frac{\partial \bar{d}_i}{\partial \delta_j} = \frac{-\bar{q}_{\text{ag}} \delta_i \lambda_j}{S^2} \leq 0 \qquad i \neq j \tag{24}$$

and

$$\frac{\partial \bar{d}_i}{\partial \delta_i} = \frac{\bar{q}_{\text{ag}} S - \bar{q}_{\text{ag}} \delta_i \lambda_i}{S^2} \geq 0. \tag{25}$$

The equalities are necessary for the case $\lambda_j = 0$ and $\lambda_i = \lambda$, respectively. ◇

*Proof of Property (4):* Let $\lambda'_k$ be the rate and $\bar{d}'_k$ be the average delay of a class $k$ after the load transition from class $i$ to class $j$. We have that $\lambda'_i = \lambda_i - \epsilon (0 < \epsilon \leq \lambda_i), \lambda'_j = \lambda_j + \epsilon$, and $\lambda'_k = \lambda_k$ for all $k \neq i, j$. So

$$S' = \sum_{n=1}^{N} \delta_n \lambda'_n = S + \epsilon(\delta_j - \delta_i). \tag{26}$$

When $i > j, \delta_j > \delta_i$, and so $S' \geq S$. Note that $S' = S$ when $\lambda_i = \lambda'_j = \lambda$. Thus, from (5), $\bar{d}'_k \leq \bar{d}_k$ for all $k = 1 \ldots N$. Similarly, when $i < j$, it follows that $\bar{d}'_k \geq \bar{d}_k$. Note that both load distributions have the same average aggregate backlog $\bar{q}_{\text{ag}}$, because the interarrivals and packet sizes follow the same distributions in all classes. ◇

*Proof of Property (5):* As in the proof of the previous property, $S' = S + \epsilon(\delta_j - \delta_i)$. Suppose that $i > j$ and thus $\delta_j > \delta_i$. It is easy to see that $\delta_j S \geq \delta_i S'$, because $S \geq \epsilon \delta_i$. Consequently

$$\bar{d}'_j = \delta_j \bar{q}_{\text{ag}}/S' \geq \delta_i \bar{q}_{\text{ag}}/S = \bar{d}_i. \tag{27}$$

Similarly, when $i < j$, it follows that $\bar{d}'_j \leq \bar{d}_i$.

## Appendix II

A random source generates the traffic for each source. The interarrivals between packets of the same class follow a Pareto distribution with shape parameter $\alpha = 1.5$. The distribution has infinite variance when $\alpha < 2$, leading to bursty interarrivals. Pareto interarrivals with infinite variance is probably an appropriate model for highly aggregated network traffic. We have experimented with several distributions for the packet sizes, including the exponential distribution, multimodal distributions around certain common values (e.g., 40, 550, and 1500 bytes), and also with fixed-size packets. In this paper, we only include simulation results for fixed-size packets; we did not observe any qualitative differences in the results when simulating variable-size packets. In all simulation results presented in this paper, the interarrival and packet size distributions are the same for all classes.

The capacity of the scheduler is normalized to $C = 1$ average-size packet per time unit. The utilization of the scheduler is $u = \lambda/C$, where $\lambda$ is the average rate of packet arrivals. The class load distribution is specified in terms of packets. For example, when we say that Class-1 is 70% of the aggregate traffic, we mean that 70% of the packets belong to Class-1. The time unit in all graphs is the time it takes to transmit an average-size packet. The reported delays in all graphs are in these time units. In the case of fixed-size packets, when we say that the queueing delay of a packet is 10 units, we mean that the packet waited in the queue for the transmission of ten other packets.

## APPENDIX III

Kleinrock derived the average queueing delays with WTP [19] in the special case of Poisson arrivals. Assuming that all classes have the same average packet size, and normalizing the service rate to $C = 1$, Kleinrock's result states that:

$$\bar{d}_i^{\rm WTP} = \bar{d}_i = \frac{\bar{d}_0/(1-\lambda) - \sum_{k=1}^{i-1}\lambda_k\delta_k(1-\delta_i/\delta_k)}{1-\sum_{k=i+1}^{N}(1-\delta_k/\delta_i)} \qquad i = 1\ldots N \tag{28}$$

where $\bar{d}_0$ is the average remaining service time for the packet that is being transmitted upon the arrival of a new packet. Recall that $\delta_1 = 1 > \delta_2 > \cdots > \delta_N > 0$. We prove (18) using induction in the following manner: after showing that $\bar{d}_2/\bar{d}_1 = \delta_2$, we will assume that $\bar{d}_k/\bar{d}_1 = \delta_k$ for all $k = 2\ldots m < N$ with $m \in \{2,\ldots,N-1\}$, and then show that $\bar{d}_{m+1}/\bar{d}_1 = \delta_{m+1}$.

For the initial induction step, it is easy to show from (28) that

$$\frac{\bar{d}_2}{\bar{d}_1} = \frac{1-\sum_{k=2}^{N}\lambda_k(1-\delta_k) - \lambda_1(1-\delta_2)}{1-\sum_{k=3}^{N}\lambda_k(1-\delta_k/\delta_2)} \tag{29}$$

and so,

$$\frac{\bar{d}_2}{\bar{d}_1} = \frac{(1-\lambda) + \sum_{k=1}^{N}\lambda_k\delta_k + \lambda_1(\delta_2-1)}{(1-\lambda)+\sum_{k=1}^{N}\lambda_k\delta_k/\delta_2 + \lambda_1(1-1/\delta_2)}. \tag{30}$$

As $\lambda \to 1$, and thus, $u \to 100\%$,

$$\frac{\bar{d}_2}{\bar{d}_1} = \frac{\tilde{S} + \lambda_1(\delta_2-1)}{\tilde{S}/\delta_2 + \lambda_1(1-1/\delta_2)} = \delta_2 \tag{31}$$

where $\tilde{S} = \lim_{\lambda\to 1}\sum_{k=1}^{N}\lambda_k\delta_k$. This completes the proof for $N = 2$.

For $N > 2$, the inductive assumption is that, as $\lambda \to 1$

$$\frac{\bar{d}_k}{\bar{d}_1} \to \delta_k \qquad \text{for all } k = 2\ldots m < N \tag{32}$$

with $m \in \{2,\ldots,N-1\}$. Then, the average delay of the $(m+1)$'th class is

$$\bar{d}_{m+1} = \frac{\bar{d}_0/(1-\lambda) - \sum_{k=1}^{m}\lambda_k\delta_k\bar{d}_1(1-\delta_{m+1}/\delta_k)}{1-\sum_{k=m+2}^{N}\lambda_k(1-\delta_k/\delta_{m+1})} \tag{33}$$

and the ratio of $\bar{d}_{m+1}$ and $\bar{d}_1$ becomes

$$\frac{\bar{d}_{m+1}}{\bar{d}_1} = \frac{1-\sum_{k=1}^{N}\lambda_k(1-\delta_k) - \sum_{k=1}^{m}\lambda_k\delta_k(1-\delta_{m+1}/\delta_k)}{1-\sum_{k=1}^{N}\lambda_k(1-\delta_k/\delta_{m+1}) + \sum_{k=1}^{m}\lambda_k(1-\delta_k/\delta_{m+1})} \tag{34}$$

As $\lambda \to 1$, let $\tilde{S} = \lim_{\lambda\to 1}\sum_{k=1}^{N}\lambda_k\delta_k, \tilde{S}_m = \lim_{\lambda\to 1}\sum_{k=1}^{m}\lambda_k\delta_k$, and $\tilde{\lambda}_m = \lim_{\lambda\to 1}\sum_{k=1}^{m}\lambda_k$. Using this notation, we have that as $\lambda \to 1$,

$$\frac{\bar{d}_{m+1}}{\bar{d}_1} \to \frac{\tilde{S} - \tilde{S}_m + \delta_{m+1}\tilde{\lambda}_m}{\tilde{S}/\delta_{m+1} - \tilde{S}_m/\delta_{m+1} + \tilde{\lambda}_m} = \delta_{m+1} \tag{35}$$

which completes the proof.

## REFERENCES

[1] J. C. R. Bennett and H. Zhang, "Hierarchical packet fair queueing algorithms," *IEEE/ACM Trans. Networking*, vol. 5, pp. 675–689, Oct. 1997.
[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services,", IETF RFC 2475, Dec. 1998.
[3] S. Bodamer, "A scheduling algorithm for relative delay differentiation," in *Proc. IEEE Conf. High Performance Switching and Routing (ATM 2000)*, June 2000, pp. 357–364.
[4] G. Bolch, S. Greiner, H. Meer, and K. S. Trivedi, *Queueing Networks and Markov Chains*. New York: Wiley, 1999.
[5] J. Y. Le Boudec, M. Hamdi, L. Blazevic, and P. Thiran, "Asymmetric best effort service for packet networks," in *Proc. Global Internet Symp.*, Dec. 1999.
[6] C. Cetinkaya and E. W. Knightly, "Egress admission control," in *Proc. IEEE INFOCOM*, Mar. 2000, pp. 1471–1480.
[7] A. Charny and J. Y. Le Boudec, "Delay bounds in a network with aggregate scheduling," in *Proc. QOFIS*, Oct. 2000.
[8] D. D. Clark and W. Fang, "Explicit allocation of best effort packet delivery service," *IEEE/ACM Trans. Networking*, vol. 6, pp. 362–373, Aug. 1998.
[9] E. G. Coffman and I. Mitrani, "A characterization of waiting time performance realizable by single-server queues," *Oper. Res.*, vol. 28, no. 3, pp. 810–821, May 1980.
[10] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *Internetworking: Res. Exp.*, pp. 3–26, 1990.
[11] C. Dovrolis, D. Stiliadis, and P. Ramanathan, "Proportional differentiated services: Delay differentiation and packet scheduling," in *Proc. ACM SIGCOMM*, Sept. 1999, pp. 109–120.
[12] C. Dovrolis and P. Ramanathan, "A case for relative differentiated services and the proportional differentiation model," *IEEE Network*, vol. 13, pp. 26–34, Sept./Oct. 1999.
[13] ——, "Proportional differentiated services, part II: Loss rate differentiation and packet dropping," in *IEEE/IFIP Int. Workshop Quality of Service (IWQoS)*, June 2000, pp. 52–61.
[14] ——, "Class provisioning in proportional differentiated services networks," presented at the Scalability and Traffic Control in IP Networks (SPIE ITCOM304), Aug. 2001.
[15] ——, "Dynamic class selection: From relative differentiation to absolute QoS," in *Proc. Int. Conf. Network Protocols*, Nov. 2001.
[16] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Trans. Networking*, vol. 3, pp. 365–386, Aug. 1995.
[17] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured Forwarding PHB Group RFC 2597,", June 1999.
[18] V. Jacobson, K. Nichols, and K. Poduri, "An Expedited Forwarding PHB,", RFC 2598, June 1999.
[19] L. Kleinrock, *Queueing Systems*. New York: Wiley, 1976, vol. II.
[20] M. K. H. Leung, J. C. S. Lui, and D. K. Y. Yau, "Characterization and performance evaluation for proportional delay differentiated services," in *Proc. Int. Conf. Network Protocols (ICNP)*, Oct. 2000.
[21] C. C. Li, S.-L. Tsao, M. C. Chen, Y. Sun, and Y.-M. Huang, "Proportional delay differentiation service based on weighted fair queueing," in *Proc. IEEE Int. Conf. Computer Communications and Networks (ICCCN)*, Oct. 2000, pp. 418–423.
[22] J. Liebeherr and N. Christin, "JoBS: Joint buffer management and scheduling for differentiated services," in *Proc. IWQoS*, June 2001, pp. 404–418.
[23] Y. Moret and S. Fdida, "A proportional queue control mechanism to provide differentiated services," in *Proc. Int. Symp.Computer and Information Systems (ISCIS)*, Oct. 1998.
[24] T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Bharghavan, "Delay differentiation and adaptation in core stateless networks," in *Proc. IEEE INFOCOM*, Mar. 2000, pp. 421–430.
[25] K. Nichols, S. Blake, F. Baker, and D. L. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers,", IETF RFC 2474, Dec. 1998.

[26] A. M. Odlyzko, “Paris metro pricing: The minimalist differentiated services solution,” in *Proc IEEE/IFIP Int. Workshop Quality of Service*, June 1999, pp. 159–161.
[27] J. Regnier, “Priority Assignment in Integrated Services Networks,”, Tech. Rep. LIDS-TH-1565, Dec. 1986. LIDS-MIT.
[28] B. Reynolds, “RED Analysis for Congested Network Core and Customer Egress,” North American Network Operators’ Group (NANOG) meeting, QualNet, Tech. Rep., Jan. 1999.
[29] S. Sahu, P. Nain, D. Towsley, C. Diot, and V. Firoiu, “On achievable service differentiation with token bucket marking for TCP,” in *Proc. ACM SIGMETRICS*, June 2000.
[30] H. Saito, C. Lukovszki, and I. Moldovan, “Local optimal proportional differentiated services scheduler for relative differentiated services,” in *Proc. IEEE Int. Conf. Computer Communications and Networks (ICCCN)*, Nov. 2000.
[31] I. Stoika and H. Zhang, “LIRA: An approach for service differentiation in the internet,” in *Proc. NOSSDAV*, 1998, pp. 115–128.
[32] ——, “Providing guaranteed services without per flow management,” in *ACM SIGCOMM*, Sept. 1999, pp. 81–94.
[33] I. Stoika, S. Shenker, and H. Zhang, “Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks,” in *ACM SIGCOMM*, Sept. 1998, pp. 118–130.
[34] P. P. White, “RSVP and integrated services in the internet: A tutorial,” *IEEE Commun. Mag.*, pp. 100–106, May 1997.

**Constantinos Dovrolis** (M’93) received the computer engineering degree from the Technical University of Crete, Crete, Greece, the M.S. degree from the University of Rochester, Rochester, NY, and the Ph.D. degree from the University of Wisconsin–Madison, in 1995, 1996, and, 2000, respectively.

Currently, he is an Assistant Professor at the Computer Science Department, University of Delaware, Newark. His research interests include network measurements, bandwidth estimation algorithms and tools, quality-of-service mechanisms, and router architectures.

Dr. Dovrolis is a member of the Association for Computing Machinery.

**Dimitrios Stiliadis** (M’96) received the Diploma in computer engineering from the University of Patras, Patras, Greece, in 1992, and the Ph.D. and M.S. degrees in computer engineering from the University of California at Santa Cruz, in 1996 and 1994 respectively.

Since 1996, he has been with the High-Speed Networks Research Department of Bell Laboratories, Holmdel, NJ, where he is currently a Distinguished Member of Technical Staff. During these years he has been leading the architecture of several generations of packet switching equipment. His recent research has been in issues related to traffic management, switch scheduling, and applications of optical technologies to packet networks.

Dr. Stiliadis is a co-recipient of the 1998 IEEE Fred W. Ellersik Award.

**Parameswaran Ramanathan** (S’85–M’86) received the B.Tech. degree from the Indian Institute of Technology, Bombay, India, in 1984, and the M.S.E. and Ph.D. degrees from the University of Michigan, Ann Arbor, in 1986 and 1989, respectively.

Since 1989, he has been a faculty member in the Department of Electrical and Computer Engineering, University of Wisconsin, Madison, where he is presently a Full Professor. He leads research projects in the areas of sensor networks and next generation cellular technology. From 1997 to 1998, he took a sabbatical leave to visit research groups at AT&T Laboratories and Telcordia Technologies. His research interests include wireless and wireline networking, real-time systems, fault-tolerant computing, and distributed systems.

Dr. Ramanathan served as an Associate Editor for IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED COMPUTING, from 1996 to 1999. He has also served on program committees of conferences such as International Conferences on Distributed Systems and Networks, Distributed Computing Systems, Fault-tolerant Computing Symposium, Real-time Systems Symposium, Conference on Local Computer Networks, and International Conference on Engineering Complex Computer Systems. He was the Finance and Registration Chair for the 1999 Fault-tolerant Computing Symposium. He was the program chairman of the Workshop on Architectures for Real-time Applications, 1994 and the program vice-chair for the International Workshop on Parallel and Distributed Real-time Systems, 1996. He is a member of the Association of Computing Machinery.