JoBS: Joint Buffer Management and Scheduling for Differentiated Services[⋆]

Jörg Liebeherr and Nicolas Christin

Computer Science Department, University of Virginia, Charlottesville, VA 22904, USA
{jorg, nicolas}@cs.virginia.edu

Abstract. A novel algorithm for buffer management and packet scheduling is presented for providing loss and delay differentiation for traffic classes at a network router. The algorithm, called JoBS (Joint Buffer Management and Scheduling), provides delay and loss differentiation independently at each node, without assuming admission control or policing. The novel capabilities of the proposed algorithm are that (1) scheduling and buffer management decisions are performed in a single step, and (2) both relative and (whenever possible) absolute QoS requirements of classes are supported. Numerical simulation examples, including results for a heuristic approximation, are presented to illustrate the effectiveness of the approach and to compare the new algorithm to existing methods for loss and delay differentiation.

1 Introduction

Quality-of-Service (QoS) guarantees in packet networks are often classified according to two criteria. The first criterion is whether guarantees are expressed for individual end-to-end traffic flows (*per-flow QoS*) or for groups of flows with the same QoS requirements (*per-class QoS*). The second criterion is whether guarantees are expressed with reference to guarantees given to other flows/flow classes (*relative QoS*), or whether guarantees are expressed as absolute bounds (*absolute QoS*).

Efforts to provision for QoS in the Internet in the early and mid-1990s, which resulted in the *Integrated Services* (IntServ) service model [3], focused on per-flow absolute QoS guarantees. However, due to scalability issues and a lagging demand for per-flow absolute QoS, the interest in Internet QoS eventually shifted to relative per-class guarantees. Since late 1997, the *Differentiated Services* (DiffServ) [2] working group has discussed several proposals for per-class relative QoS guarantees, e.g., [4, 17].

With the exception of the Expedited Forwarding service [11], proposals for relative per-class QoS discussed within the DiffServ context define the service differentiation qualitatively, in the sense that some classes receive lower delays and a lower loss rate

[⋆] This work is supported in part by the National Science Foundation through grants NCR-9624106 (CAREER), ANI-9730103, and ANI-0085955.

than others, but without quantifying the differentiation. Recently, research studies have tried to strengthen the guarantees of relative per-class QoS, and have proposed new buffer management and scheduling algorithms which can support stronger notions of relative QoS [6, 7, 15, 16]. Probably the best known such effort is the *proportional service differentiation* model [6, 7], which attempts to enforce that the ratios of delays or loss rates of successive priority classes be roughly constant. For two priority classes such a service could specify that the delays of packets from the higher-priority class be half of the delays from the lower-priority class, but without specifying an upper bound on the delays.

In this paper, we express the provisioning of per-class QoS within a formalism inspired by the network calculus [5]. We present a rate allocation and dropping algorithm for a single output link, called *Joint Buffer Management and Scheduling (JoBS)*, which is capable of supporting a wide range of relative, as well as absolute, per-class guarantees for loss and delay, without assuming admission control or traffic policing. The algorithm operates as follows. Upon each arrival, a prediction is made on the delays of the currently backlogged traffic. Then, the service rates allocation to classes are adjusted to meet delay requirements. If necessary, traffic from certain classes is selectively dropped. A unique feature of the presented algorithm is that rate allocation for link scheduling and buffer management are approached together in a single step. The JoBS algorithm provides delay and loss differentiation independently at each node. End-to-end delays and end-to-end loss rates are thus dependent on the per-node guarantees of traffic and on the number of nodes traversed.

This paper is organized as follows. In Section 2 we give an overview of the current work on relative per-class QoS guarantees. In Sections 3 and 4, we specify our algorithm for buffer management and rate allocation. In Section 5 we propose a heuristic approximation of the algorithm. In Section 6 we evaluate the effectiveness of our algorithm via simulation. In Section 7 we present brief conclusions.

2 Related Work

Due to space considerations, we limit our discussions to the relevant work on scheduling and buffer management algorithms for relative service differentiation.

Scheduling. The majority of work on per-class relative service differentiation suggests to use well-known fixed-priority, e.g., [17], or rate-based scheduling algorithms, e.g., [9]. Only a few scheduling algorithms have been specifically designed for relative delay differentiation. The Proportional Queue Control Mechanism (PQCM, [15]) and Backlog-Proportional Rate scheduler (BPR, [6]) are variations of the GPS algorithm [18]. Both schemes use the backlog of classes to determine the service rate allocation, and bear similarity to the scheduling component of JoBS, in the sense that they dynamically adjust service rate allocations to meet relative QoS requirements.

Different from the rate-based schedulers discussed above, the Waiting-Time Priority scheduler (WTP, [7]) implements a well-known scheduling algorithm with time-dependent priorities ([12], Ch. 3.7). Likewise, the Mean-Delay Proportional scheduler (MDP, [16]) uses a dynamic priority mechanism, but sets priorities based on the average experienced delay of packets. Finally, the Hybrid Proportional Delay scheduler (HPD,

[6]) uses a combination of time-dependent priorities and average experienced delay to set the priority of a given packet.

The Alternative Best-Effort service (ABE, [10]) provides service differentiation for two traffic classes. The first class is provided with absolute delay guarantees, and the second class has guarantees for a lower loss rate. The delay guarantees for the first class are enforced by dropping all traffic that has exceeded the delay bound.

In contrast to the schedulers presented in this section, the scheduling algorithm presented in this paper not only considers the current state and past history of the link, but, in addition, makes predictions on future delays to improve the performance of its scheduling decisions.

Buffer Management. For a discussion of buffer management algorithms, we refer to a recent survey article [13]. Many proposals for buffer management in IP networks are motivated with the need to improve TCP performance (e.g., RED [8], REM [1]). Techniques specifically targeted for class-based service differentiation include RIO [4] and multiclass RED [19]. Of these schemes, REM is closest in spirit to the dropping algorithm presented in this paper, since REM treats the problem of marking (or dropping) arrivals as an optimization problem.

The Proportional Loss Rate (PLR) dropper [7] is specifically designed to support proportional differentiated services. PLR enforces that the ratio of the loss rates of two successive classes remains roughly constant at a given value. There are two variants of PLR. PLR(M) uses only the last M packets for estimating the loss rate of a class, whereas PLR(∞) has no such memory constraints.

With the possible exception of [10], the work on relative per-class service differentiation generally considers delay and loss differentiation as orthogonal issues, which are handled by separate algorithms.

3 An Approach to Joint Buffer Management and Scheduling

In this section, we introduce the key concepts of *Joint Buffer Management and Scheduling* (JoBS). Before we provide a detailed description, we first give an informal overview of the operations.

3.1 Overview

We assume that each output link performs per-class buffering of arriving traffic and that traffic is transmitted from the buffers using a rate-based scheduling algorithm [21] with a dynamic, time-dependent service rate allocation for classes. Traffic from the same class is transmitted in a First-Come-First-Served order. There is no admission control and no policing of traffic. The set of performance requirements are specified to the algorithm as a set of per-class QoS constraints. As an example, for three classes, the QoS constraints could be of the form:

– Class-1 Delay $\approx 2 \cdot$ Class-2 Delay,
– Class-2 Loss Rate $\approx 10^{-1} \cdot$ Class-3 Loss Rate, or
– Class-3 Delay $\leq 5\ ms$.

Here, the first two constraints are relative constraints and the last one is an absolute constraint. The set of constraints can be any mix of relative and absolute constraints. Since absolute constraints may render a system of constraints infeasible, some constraints may need to be relaxed. We assume that all QoS constraints are prioritized, so that an order is provided in which constraints are relaxed in case the system of constraints is infeasible.

The time-dependent service rate allocation operates as follows. For every arrival, a prediction is made on the delays of all backlogged traffic. Then, the service rate allocation to traffic classes is modified so that all QoS constraints will be met. If no feasible rate allocation for meeting all constraints exists, traffic is dropped, either from a new arrival or from the current backlog.

We find it convenient to view the service rate allocation in terms of an optimization problem. The constraints of the optimization problem are relative or absolute bounds on the loss and delay as given in the example above (*QoS constraints*) and constraints on the link and buffer capacity (*system constraints*). The objective function of the optimization primarily aims at minimizing the amount of traffic to be dropped, and, as a secondary objective, aims at maintaining the current service rate allocation. The first objective prevents traffic from being dropped unnecessarily, and the second objective tries to avoid frequent fluctuations of the service rate allocation. The solution of the optimization problem yields a service rate allocation of classes and determines how much traffic must be dropped.

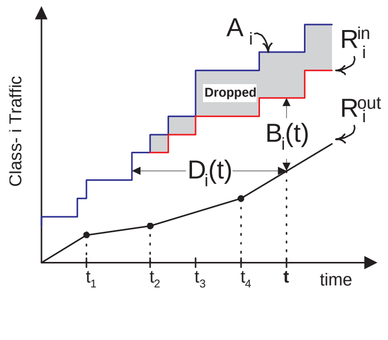
To explore the principal properties of the optimization, we will, at first, assume that sufficient computing resources are available to solve the optimization problem for each arrival to the link. In a later section, we will approximate the optimization with a heuristic which incurs less computational overhead.

3.2 Formal Description

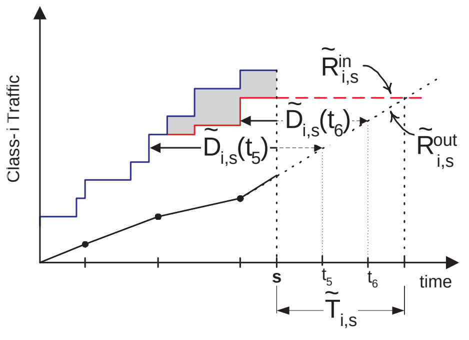
Next we describe the basic operations of the service rate allocation and the dropping algorithms at a link with capacity C and total buffer space B. We assume that all traffic is marked to belong to one of Q traffic classes. In general, we expect Q to be small, e.g., $Q = 4$. Classes are marked by an index. We use a convention, whereby a class with a smaller index requires a better level of QoS. We use $a_i(t)$ and $l_i(t)$ to denote the traffic arrivals and amount of dropped traffic from class i at time t. We use $r_i(t)$ to denote the service rate allocated to class i at time t. We assume that $r_i(t) > 0$ only if there is a backlog of class-i traffic in the buffer (and $r_i(t) = 0$ otherwise), and we assume that scheduling is work-conserving, that is, $\sum_i r_i(t) = C$, if there is at least one backlogged class at time t.

Remark. Throughout this paper, we take a fluid-flow interpretation of traffic, that is, the output link is regarded as serving simultaneously traffic from several classes. Since actual traffic is sent in discrete-sized packets, a fluid-flow interpretation of traffic is idealistic. However, scheduling algorithms that closely approximate fluid-flow schedulers with rate guarantees are available [18, 21].

We now introduce the notions of *arrival curve*, *input curve*, and *output curve* for a traffic class i in the time interval $[0, t]$. The arrival curve A_i and the input curve R_i^{in} of



(a) Delay and backlog.



(b) Projected input curve, projected output curve, and projected delays.

Fig. 1. Delay, Backlog and Projections. In Figure 1(b), the projection is performed at time s for the time interval $[s, s + \tilde{T}_{i,s}]$.

class i are defined as

$$A_i(t) = \int_0^t a_i(x)dx \ , \ \ R_i^{in}(t) = A_i(t) - \int_0^t l_i(x)dx \ . \tag{1}$$

So, the difference between the arrival and input curve is the amount of dropped traffic. The output curve R_i^{out} of class-i is the transmitted traffic in the interval $[0, t]$, given by

$$R_i^{out}(t) = \int_0^t r_i(x)dx \ . \tag{2}$$

We refer to Figure 1(a) for an illustration. In the figure, the service rate is adjusted at times t_1, t_2, and t_4, and packet drops occur at times t_2 and t_3.

The vertical and the horizontal distance between the input and output curves from class i, respectively, are the backlog B_i and the delay D_i. This is illustrated in Figure 1(a) for time t. The delay D_i at time t is the delay of an arrival which is transmitted at time t. Backlog and delay at time t are defined as

$$B_i(t) = R_i^{in}(t) - R_i^{out}(t) \ , \ \ D_i(t) = \max_{x<t}\{x \mid R_i^{out}(t) \geq R_i^{in}(t-x)\} \ . \tag{3}$$

Upon a traffic arrival, say at time s, the new service rates $r_i(s)$ and the amount of traffic to be dropped $l_i(s)$ for all classes are set such that all QoS and system constraints can be met at times greater than s. If all constraints cannot be satisfied at the same time, then some QoS constraints are relaxed in a predetermined order.

To determine the rate allocation, the scheduler makes a projection of the delays of all backlogged traffic. For the purpose of the projection, it is assumed that the current state of the link will not change after time s. Specifically, indicating projected values by a tilde (˜), for times $t > s$, we assume that (1) service rates remain as they are (i.e., $\tilde{r}_i(t) = r_i(s)$), (2) there are no further arrivals (i.e., $\tilde{a}_i(t) = 0$), and (3) there are no further packet drops (i.e., $\tilde{l}_i(t) = 0$).

With these assumptions, we now define the notions of projected input curve $\tilde{R}^{in}_{i,s}$, projected output curve $\tilde{R}^{out}_{i,s}$, and projected backlog $\tilde{B}_{i,s}$, for $t > s$ as follows:

$$\tilde{R}^{in}_{i,s}(t) = R^{in}_i(s)\ ,\ \tilde{R}^{out}_{i,s}(t) = R^{out}_i(s) + (t-s)r_i(s)\ ,\ \tilde{B}_{i,s}(t) = \tilde{R}^{in}_{i,s}(t) - \tilde{R}^{out}_{i,s}(t)\ . \tag{4}$$

We refer to the *projected horizon* for class i at time s, denoted as $\tilde{T}_{i,s}$, as the time when the projected backlog becomes zero, i.e., $\tilde{T}_{i,s} = \min_{x>0}\{x \mid \tilde{B}_{i,s}(s+x) = 0\}$. With this notation, we can make predictions for delays in the time interval $[s, s+\tilde{T}_{i,s}]$. We define the projected delay $\tilde{D}_{i,s}$ as

$$\tilde{D}_{i,s}(t) = \max_{t-s<x<t}\{x \mid \tilde{R}^{out}_{i,s}(t) \geq R^{in}_i(t-x)\}\ . \tag{5}$$

If there are no arrivals after time s, the delay projections are correct. In Figure 1(b), we illustrate the projected input curve, projected output curve, and projected delays for projections made at time s. In the figure, all values for $t > s$ are projections and are indicated by dashed lines. The figure includes the projected delays for times t_5 and t_6.

4 Service Rate Adaptation and Drop Algorithm

In this section we discuss an algorithm to perform the service rates allocation to classes and the decision to drop traffic in terms of an optimization problem.

Each time s at which an arrival occurs, a new optimization is performed. The optimization variable is a time-dependent vector $\mathbf{x}_s = (r_1(s)\ldots r_Q(s)\ l_1(s)\ldots l_Q(s))^T$, which contains the service rates $r_i(s)$ and the amount of traffic to be dropped $l_i(s)$. The optimization problem has the form

$$\begin{array}{ll} \textbf{Minimize} & F(\mathbf{x}_s) \\ \textbf{Subject to} & g_j(\mathbf{x}_s) = 0,\ j = 1,\ldots,M \\ & h_j(\mathbf{x}_s) \geq 0,\ j = M+1,\ldots,N, \end{array} \tag{6}$$

where $F(.)$ is an objective function, and the g_j's and h_j's are constraints. The objective function, which will be presented in Subsection 4.2, will be chosen so that the amount of dropped traffic and the changes to the current service rate allocation are minimized. The constraints of the optimization problem are QoS constraints and system constraints. The optimization at time s is done with knowledge of the system state before time s, that is the optimizer knows R^{in}_i and R^{out}_i for all times $t < s$, and A_i for all times $t \leq s$.

In the remainder of this section we discuss the constraints and the optimization function. The optimization can be used as a reference system against which practical scheduling and dropping algorithms can be compared.

4.1 System and QoS Constraints

There are two types of constraints. *System constraints* describe constraints and properties of the output link, and *QoS constraints* define the desired service differentiation.

System Constraints. The system constraints specify physical limitations and properties at the output link. The first such constraint states that the total backlog cannot exceed the buffer size B, that is, $\sum_i B_i(t) \leq B$ for all times t. The second system constraint enforces that scheduling at the output link is work-conserving. At a work-conserving link, $\sum_i r_i(t) = C$ holds for all times t where $\sum_i B_i(t) > 0$. Other system constraints enforce that transmission rates and loss rates are non-negative. Also, the amount of traffic that can be dropped is bounded by the current backlog. So we obtain $r_i(t) \geq 0$ and $0 \leq l_i(t) \leq B_i(t)$ for all times t.

QoS Constraints. We consider two types of QoS constraints, relative constraints and absolute constraints. QoS constraints are either constraints on delays or constraints on the loss rate. The number and type of QoS constraints is not limited. Since absolute QoS constraints may result in an infeasible system of constraints, one or more constraints may need to be relaxed at certain times. We assume that the set of QoS constraints is assigned some total order, and that constraints are relaxed in the given order until the system of constraints becomes feasible. In addition, QoS constraints for classes which are not backlogged are simply ignored.

Absolute delay constraints (ADC) enforce that the projected delays of class i satisfy a worst-case bound d_i. That is,

$$\max_{s<t<s+\tilde{T}_{i,s}} \tilde{D}_{i,s}(t) \leq d_i \ , \tag{7}$$

for all $t \in [s, s+\tilde{T}_{i,s}]$. If this condition holds for all s, the delay bound d_i is never violated.

Relative delay constraints (RDC) specify the proportional delay differentiation between classes. As an example, for two classes 1 and 2, the RDC enforces a relationship

$$\frac{\text{Delay of Class 2}}{\text{Delay of Class 1}} \approx \text{constant} \ .$$

Since, in general, there are several packets backlogged from a class, each likely to have a different delay, the notion of 'delay of class i' needs to be further specified. For example, the delay of class i could be specified as the delay of the packet at the head of the class-i queue, the maximum projected delay as in Eqn. (7), or via other measures. We choose a measure, called *average projected delay* $\overline{D}_{i,s}$, which is the time average of the projected delays from a class, averaged over the horizon $\tilde{T}_{i,s}$. We obtain

$$\overline{D}_{i,s} = \frac{1}{\tilde{T}_{i,s}} \int_{s}^{s+\tilde{T}_{i,s}} \tilde{D}_{i,s}(x)dx \ . \tag{8}$$

To provide some flexibility in the scheduling decision, we do not enforce relative delay constraints strictly, but allow for some slack. Using the metric defined in Eqn. (8), and translating the notion of slack into a tolerance level, we can write the relative delay constraints as

$$k_i(1-\varepsilon) \leq \frac{\overline{D}_{i+1,s}}{\overline{D}_{i,s}} \leq k_i(1+\varepsilon) \ , \tag{9}$$

where $k_i > 1$ is a constant defining the proportional differentiation desired, and ε ($0 \leq \varepsilon \leq 1$) indicates a tolerance level. If relative constraints are not specified for some classes, the constraints are adjusted accordingly. Note that in the delay constraints in Eqs. (7) and (9), all values with exception of the components of the optimization variable $\mathbf{x}_s$ are known at time s.

Next we discuss constraints on the loss rate. Similar to delays, there are several sensible choices for defining 'loss'. Here, we select a loss measure, denoted by $p_{i,s}$, which expresses the fraction of lost traffic since the beginning of the current busy period at time t_0.[1] So, $p_{i,s}$ expresses the fraction of traffic that has been dropped in the time interval $[t_0, s]$, that is,[2]

$$p_{i,s} = \frac{\int_{t_0}^{s} l_i(x)dx}{\int_{t_0}^{s} a_i(x)dx} = 1 - \frac{R_i^{in}(s^-) + (a_i(s) - l_i(s)) - R_i^{in}(t_0)}{A_i(s) - A_i(t_0)} \,. \tag{10}$$

In the last equation, all values except $l_i(s)$ are known at time s. With this definition we now specify absolute and relative constraints on the loss rates.
An *absolute loss constraint (ALC)* specifies that the loss rate of class i, as defined above, never exceeds a limit L_i, that is,

$$p_{i,s} \leq L_i \,. \tag{11}$$

Relative loss constraints (RLC) specify the desired proportional loss differentiation between classes. Similar to the RDCs, we provide a certain slack within these constraints. The RLC for classes $(i+1)$ and i has the form

$$k'_i(1 - \varepsilon') \leq \frac{p_{i+1,s}}{p_{i,s}} \leq k'_i(1 + \varepsilon') \,, \tag{12}$$

where $k'_i > 1$ is the target differentiation factor, and ε' ($0 \leq \varepsilon' \leq 1$) indicates a level of tolerance.

4.2 Objective Function

Provided that the QoS and system constraints can be satisfied, the objective function will select a solution for $\mathbf{x}_s$. Even though the choice of the objective function is a policy decision, we select two specific objectives, which, we believe, have general validity: (1) *avoid dropping traffic* , and (2) *avoid changes to the current service rate allocation*. The first objective ensures that traffic is dropped only if there is no alternative way to satisfy the constraints. The second objective tries to hold on to a feasible service rate allocation as long as possible. We give the first objective priority over the second objective.

The following formulation of an objective function expresses the above objectives in terms of a cost function:

$$F(\mathbf{x}_s) = \sum_{i=1}^{Q}(r_i(s) - r_i(s^-))^2 + C^2 \sum_{i=1}^{Q} l_i(s) \,, \tag{13}$$

[1] A busy period is a time interval with a positive backlog of traffic. For time x with $\sum_i B_i(x) > 0$, the beginning of the busy period is given by $\sup_{y<x}\{\sum_i B_i(y) = 0\}$.

[2] $s^- = s - h$, where $h > 0$ is infinitesimally small.

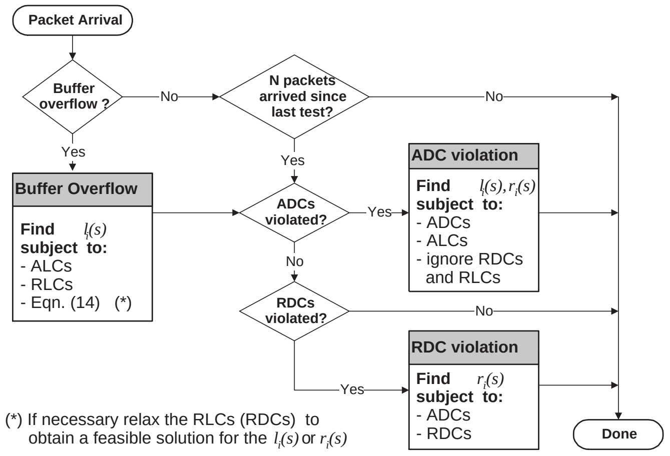

Packet Arrival
Buffer overflow ?
No
Yes
N packets arrived since last test?
No
Yes
Buffer Overflow
Find $l_i(s)$ subject to:
- ALCs
- RLCs
- Eqn. (14) (*)
ADCs violated?
Yes
No
ADC violation
Find $l_i(s), r_i(s)$ subject to:
- ADCs
- ALCs
- ignore RDCs and RLCs
RDCs violated?
No
Yes
RDC violation
Find $r_i(s)$ subject to:
- ADCs
- RDCs
Done
(*) If necessary relax the RLCs (RDCs) to obtain a feasible solution for the $l_i(s)$ or $r_i(s)$

Fig. 2. Outline of the Heuristic Algorithm.

where C is the link capacity. The first term expresses the changes to the service rate allocation and the second term expresses the losses at time s. Note that, at time s, $r_i(s)$ is part of the optimization variable, while $r_i(s^-)$ is a known value. In Eqn. (13) we use the quadratic form $(r_i(s) - r_i(s^-))^2$, since $\sum_i (r_i(s) - r_i(s^-)) = 0$ for a work-conserving link with a backlog at time s. The scaling factor C^2 in front of the second sum of Eqn. (13) ensures that traffic drops are the dominating term in the objective function.

This concludes the description of the optimization process in JoBS. The structure of constraints and objective function makes this a *non-linear optimization problem*, which can be solved with available numerical algorithms [20].

5 Heuristic Approximation

We next present a heuristic that approximates the optimization presented in the previous section, with significantly lower computational complexity. The presented heuristic should be regarded as a first step towards a router implementation.

Approximating a non-linear optimization problem such as the one presented in Section 4 can be performed by well-known techniques such as fuzzy systems, or neural networks. However, these techniques are computationally too expensive if a high accuracy in the approximation is desired. Therefore, we choose a different approach, which decomposes the optimization problem into several computationally less intensive problems. The heuristic algorithm presented here maintains a feasible rate allocation until a buffer overflow occurs or a delay violation is predicted. At that time, the heuristic picks a new feasible rate allocation and/or drops traffic. Unless there is a buffer overflow, the tests for violations of ADCs and RDCs are not performed for every packet arrival, but only periodically.

A set of constraints, which contains absolute constraints (ALCs or ADCs), may be infeasible at certain times. Then, some constraints need to be relaxed. In our heuristic algorithm, the constraints are prioritized in the following order: system constraints have priority over absolute constraints, which in turn have priority over relative constraints. If the system of constraints becomes infeasible, the heuristic relaxes the relative constraints (RLCs or RDCs). If this does not yield a feasible solution, the heuristic relaxes one or more absolute constraints.

A high-level overview of the heuristic algorithm is presented in Figure 2. The algorithm consists of a number of small computations, one for each situation which requires to adjust the service rate allocation and/or to drop packets. We next present each of these situations and the associated computation.

Buffer Overflow. If an arrival at time s causes a buffer overflow, one can either drop the arriving packet or free enough buffer space to accommodate the arriving packets. Both cases are satisfied if

$$\sum_i l_i(s) = \sum_i a_i(s) \ . \tag{14}$$

The heuristic picks a solution for the $l_i(s)$ which satisfies Eqn. (14) and the RLCs in Eqn. (12), where we set $\varepsilon' = 0$ to obtain a unique solution. If the solution violates an ALC, the RLCs are relaxed until all ALCs are satisfied. Once the $l_i(s)$'s are determined the algorithm continues with a test for delay constraint violations, as shown in Figure 2. The algorithm only specifies the amount of traffic which should be dropped from a particular class, however, the algorithm does not select the position in the queue from which to drop traffic. In the present paper, we assume a Drop-Tail dropping policy.

If there are no buffer overflows, the algorithm makes projections for delay violations only once for every N packet arrivals. The selection of N represents a tradeoff between the runtime complexity of the algorithm and performance of the scheduling with respect to satisfying the constraints. Simulation experiments, as described in Section 6, show that the value $N = 100$ provides good performance.

The tests use the current service rate allocation to predict future violations. For delay constraint violations, the heuristic distinguishes three cases.

Case 1: No violation. In this case, the service rates are unchanged.

Case 2: RDC violation. If some RDC (but no ADC) is violated, the heuristic algorithm determines new rate values. Here, the RDCs as defined in Eqn. (9) are transformed into equations by setting $\varepsilon = 0$. Together with the work-conserving property, one obtains a system of equations, for which the algorithm picks a solution. If the solution violates an ADC, the RDCs are relaxed until the ADCs are satisfied.

Case 3: ADC violation. Resolving an ADC violation is not entirely trivial as it requires to recalculate the $r_i(s)$'s, and, if traffic needs to be dropped to meet the ADCs, the $l_i(s)$'s. To simplify the task, our heuristic ignores all relative constraints when an ADC violation occurs, and only tries to satisfy absolute constraints.

The heuristic starts with a conservative estimate of the worst-case delay for the class-i backlog at time s. For this, the heuristic uses the fact that for all $x \in [s, s+\tilde{T}_{i,s}]$, $\tilde{D}_{i,s}(x) \leq D_i(s) + \frac{B_i(s)}{r_i(s)}$, which can be verified by referring to Figures 1(a) and 1(b). Then, using $B_i(s) = B_i(s^-) + a_i(s) - l_i(s)$, we can write a sufficient condition for

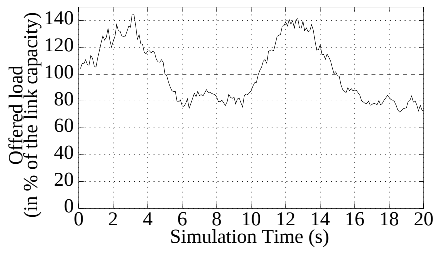


Fig. 3. Offered Load.

satisfying the ADC of class i with delay bound d_i at time s,

$$\underbrace{\frac{1}{r_i(s)}\frac{B_i(s^-)+a_i(s)-l_i(s)}{d_i-D_i(s)}}_{\rho_i} \leq 1\ . \tag{15}$$

The heuristic algorithm will select the $r_i(s)$ and $l_i(s)$ such that Eqn. (15) is satisfied for all i. Initially, rates and traffic drops are set to $r_i(s) = r_i(s^-)$ and $l_i(s) = 0$. Since at least one ADC is violated, there is at least one class with $\rho_i > 1$, where ρ_i is defined in Eqn.(15). Now, we apply a greedy method which tries to redistribute the rate allocations until $\rho_i \leq 1$ for all classes. This is done by reducing $r_i(s)$ for classes with $\rho_i < 1$, and increasing $r_i(s)$ for classes with $\rho_i > 1$. If it is not feasible to achieve $\rho_i \leq 1$ for all classes by adjusting the $r_i(s)$'s, the $l_i(s)$'s are increased until $\rho_i \leq 1$ for all i. To minimize the number of dropped packets, $l_i(s)$ is never increased to a point where an ALC is violated.

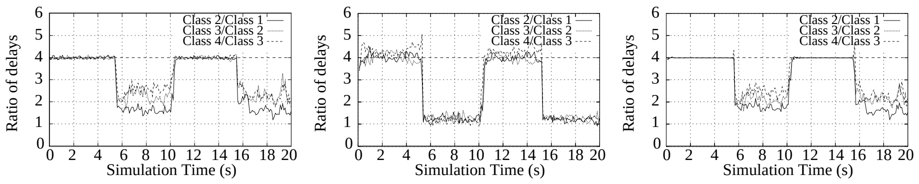
6 Evaluation

We present an evaluation of the algorithms developed in this paper via simulation. Our goals are (1) to determine if and how well the desired service differentiation is achieved; (2) to determine how well the heuristic algorithm from Section 5 approximates the optimization from Section 4; and (3) to compare our algorithm with existing proposals for proportional differentiated services.

We present two simulation experiments. In the first experiment, we compare the relative differentiation provided by the optimization algorithm described in Section 4, *JoBS (optimization)*, the heuristic approximation of Section 5, *JoBS (heuristic)*, and *WTP/PLR(∞)* [7], which provided uniformly the best results among previously proposed schemes for relative service differentiation. In the second experiment, we augment the set of constraints by absolute loss and delay constraints on the highest priority class, and show that JoBS can effectively provide both relative and absolute differentiation.

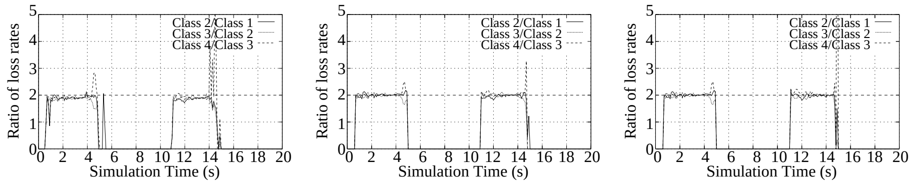
6.1 Experimental Setup

We consider a single output link with capacity $C = 1$ Gbps and a buffer size of 6.25 MB. We assume $Q = 4$ classes. The length of each experiment is 20 seconds of



(a) JoBS (optimization). (b) JoBS (heuristic). (c) WTP/PLR(∞).

Fig. 4. Experiment 1: Relative Delay Differentiation. The graphs show the ratios of the delays for successive classes. The target value is $k = 4$.



(a) JoBS (optimization). (b) JoBS (heuristic). (c) WTP/PLR(∞).

Fig. 5. Experiment 1: Relative Loss Differentiation. The graphs show the ratios of loss rates for successive classes. The target value is $k' = 2$.

simulated time, starting with an empty system. In all experiments, the incoming traffic is composed of a superposition of Pareto sources with $\alpha = 1.2$ and average interarrival time of 300 μs. The number of sources active at a given time oscillates between 200 and 550, following a sinusoidal pattern. All sources generate packets with a fixed size of 125 bytes. The resulting offered load is plotted in Figure 3. At any time, each class contributes 25% of the aggregate load, yielding a symmetric load. In a realistic environment, one would expect to have "less" high priority traffic than low priority traffic. Therefore, a symmetric load can be regarded as a realistic worst-case that can occur during bursts of high-priority traffic.

6.2 Simulation Experiment 1: Relative Differentiation Only

The first experiment focuses on relative service differentiation, and does not include absolute constraints. The objectives for the relative differentiation are so that we want to have a ratio of four between the delays of two successive classes, and a ratio of two between the loss rates of two successive classes. Thus, for JoBS, we set $k_i = 4$ and $k'_i = 2$ for all i. The tolerance levels are set to $(\varepsilon, \varepsilon') = (0.001, 0.05)$ in JoBS (optimization), and to $\varepsilon = 0.01$ in JoBS (heuristic). The results of the experiment are presented in Figures 4 and 5, where we graph the ratios of delays and loss rates, respectively, of successive classes for JoBS (optimization), JoBS (heuristic), and WTP/PLR(∞). The plotted delay and loss values are averages over moving time windows of size 0.1 s.

When the link load is above 90% of the link capacity, that is, in time intervals $[0\ s, 6\ s]$ and $[10\ s, 15\ s]$, all methods provide the desired service differentiation. The oscillations around the target values in JoBS (optimization) and JoBS (heuristic) are

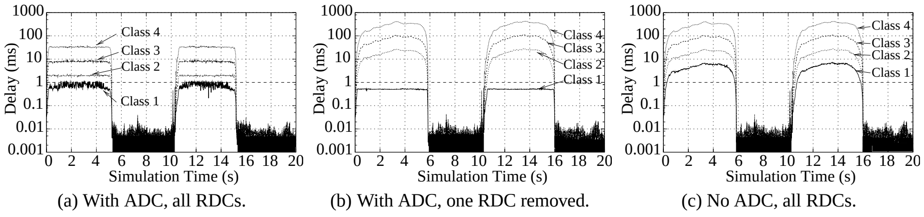


Fig. 6. Experiment 2: Absolute Delay Differentiation. The graphs show the delays of all packets. All results are for JoBS (heuristic).

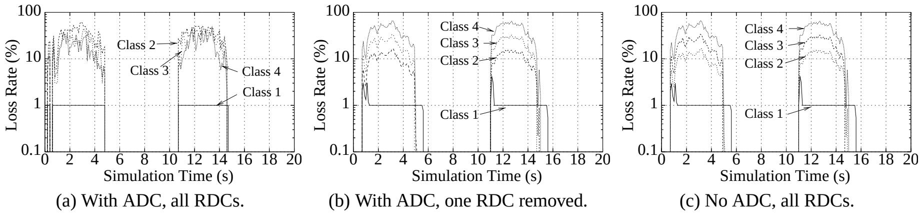


Fig. 7. Experiment 2: Absolute Loss Differentiation. The graphs show the loss rates of all classes. All results are for JoBS (heuristic).

mostly due to the tolerance values ε and ε'. The selection of the tolerance values ε and ε' in JoBS presents a tradeoff: smaller values for ε and ε' reduce oscillations, but incur more work for the algorithms. When the system load is low, that is, in time intervals $[6\ s, 10\ s]$ and $[16\ s, 20\ s]$, only JoBS (optimization) and WTP/PLR(∞) manage to achieve some delay differentiation, albeit far from the target values. However, at an underloaded link, the absolute values of the delays are very small for all classes.

Finally, one should note that the total loss rate is of interest, as a scheme may provide excellent proportional loss differentiation, but have an overall high loss rate. Additional plots provided in [14] show that the loss rates and the absolute values for the delays are very similar in all schemes.

6.3 Simulation Experiment 2: Relative and Absolute Differentiation

In this second experiment, we evaluate how well our algorithm can satisfy a mix of absolute and relative constraints on both delays and losses. Here, we only present results for JoBS (heuristic). WTP/PLR(∞) does not support absolute guarantees.

We consider the same simulation setup and the same relative delay constraints as in Experiment 1, but add an absolute delay constraint (ADC) for Class 1 such that d_1=1 ms, and we replace the relative loss constraint (RLC) between Classes 1 and 2 by an absolute loss constraint (ALC) for Class 1 such that L_1=1%. We call this scenario "with ADC, all RDCs". With the given relative delay constraints from Experiment 1, the other classes have implicit absolute delay constraints, which are approximately[3] 4 ms for Class 2, 16 ms for Class 3, and 64 ms for Class 4. Removing the RDC between

[3] Due to the tolerance value ε, the exact values are not integers.

Class 1 and Class 2, we avoid the 'implicit' absolute constraints for Classes 2, 3, and 4, and call the resulting constraint set "with ADC, one RDC removed". We also include the results for JoBS (heuristic) from Experiment 1, with the ALC on Class 1 replacing the RLC between Classes 1 and 2, and refer to this constraint set as "no ADC, all RDCs". In Figure 6 we plot the absolute delays of all packets, and in Figure 7 we plot the loss rates of all classes, averaged over time intervals of length 0.1 s. We discuss the results for each of the three constraint sets proposed.

Concerning the experiment "with ADC, all RDCs", Figure 6(a) shows that the heuristic maintains the relative delay differentiation between classes, thus, enforcing the 'implicit' delay constraints for Classes 2, 3, and 4. With a large number of absolute delay constraints, the system of constraints easily becomes infeasible, which brings two observations. First, Figure 7(a) shows that the loss rates of Classes 2, 3 and 4 are similar. This result illustrates that the heuristic relaxes relative loss constraints to meet the absolute delay constraints. Second, Figure 6(a) shows that the absolute delay constraint d_1 is sometimes violated. However, such violations are rare (over 95% of Class-1 packets have a delay less than 900 μs), and Class-1 packet delays always remain reasonably close to the delay bound d_1. For the experiment "with ADC, one RDC removed", Figure 6(b) shows that, without an RDC between Classes 1 and 2, the ratio of Class-2 delays and Class-1 delays can exceed a factor of 10 at high loads. With this constraint set, the absolute delay constraint d_1 is never violated, and Figure 7(b) shows the RLCs are consistently enforced during periods of packet drops. Finally, for the experiment "no ADC, all RDCs", Figure 6(c) shows that, without the ADC, the delays for Class 1 are as high as 5 ms.[4]

7 Conclusions

We proposed an algorithm, called JoBS (Joint Buffer Management and Scheduling), for relative and absolute per-class QoS guarantees without information on traffic arrivals. At times when not all absolute QoS guarantees can be satisfied simultaneously, JoBS selectively ignores some of the QoS guarantees. The JoBS algorithm reconciles rate allocation and buffer management into a single scheme, thereby acknowledging that scheduling and dropping decisions at an output link are not orthogonal issues, but should be addressed together. JoBS implements the desired service differentiation based on delay predictions of backlogged traffic. The predictions are used to update service rate allocations to classes and the amount of traffic to be dropped. We showed in a set of simulation experiments, that JoBS can provide relative and absolute per-class QoS guarantees for delay and loss.

In future work, we will extend the approach presented in this paper to TCP congestion control. As a point of departure, we will attempt to express existing active queue management schemes, e.g., RED [8] and RIO [4], within the formal framework introduced in this paper.

[4] The delay values for Classes 2, 3, and 4 in Figures 6(b) and (c) appear similar, especially since we use a log-scale. We emphasize that the values are *not* identical, and that the results are consistent.

References

1. S. Athuraliya, D. Lapsley, and S. Low. An enhanced random early marking algorithm for internet flow control. In *Proceedings of IEEE INFOCOM 2000*, pages 1425–1434, Tel-Aviv, Israel, April 2000.
2. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. IETF RFC 2475, December 1998.
3. R. Braden, D. Clark, and S. Shenker. Integrated services in the internet architecture: an overview. IETF RFC 1633, July 1994.
4. D. Clark and W. Fang. Explicit allocation of best-effort packet delivery service. *IEEE/ACM Transactions on Networking*, 6(4):362–373, August 1998.
5. R. Cruz, H. Sariowan, and G. Polyzos. Scheduling for quality of service guarantees via service curves. In *Proceedings of the International Conference on Computer Communications and Networks (ICCCN)*, pages 512–520, Las Vegas, NV., September 1995.
6. C. Dovrolis. *Proportional Differentiated Services for the Internet*. PhD thesis, University of Wisconsin-Madison, December 2000.
7. C. Dovrolis and P. Ramanathan. Proportional differentiated services, part II: Loss rate differentiation and packet dropping. In *Proceedings of IWQoS 2000*, pages 52–61, Pittsburgh, PA., June 2000.
8. S. Floyd and V. Jacobson. Random early detection for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, July 1993.
9. S. Floyd and V. Jacobson. Link-sharing and resource management models for packet networks. *IEEE/ACM Transactions on Networking*, 3(4):365–386, August 1995.
10. P. Hurley, M. Kara, J.-Y. Le Boudec, and P. Thiran. ABE: Providing a low delay service within best-effort. Technical Report DSC/2000/34, EPFL-DI-ICA, September 2000.
11. V. Jacobson, K. Nichols, and K. Poduri. An expedited forwarding PHB. IETF RFC 2598, June 1999.
12. L. Kleinrock. *Queueing Systems. Volume II: Computer Applications*. John Wiley & Sons, New York, NY, 1976.
13. M. A. Labrador and S. Banerjee. Packet dropping policies for ATM and IP networks. *IEEE Communications Surveys*, 2(3), 3rd Quarter 1999.
14. J. Liebeherr and N. Christin. Buffer management and scheduling for enhanced differentiated services. Technical Report CS-2000-24, University of Virginia, August 2000.
15. Y. Moret and S. Fdida. A proportional queue control mechanism to provide differentiated services. In *Proceedings of the International Symposium on Computer and Information Systems (ISCIS)*, pages 17–24, Belek, Turkey, October 1998.
16. T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Bharghavan. Delay differentiation and adaptation in core stateless networks. In *Proceedings of IEEE INFOCOM 2000*, pages 421–430, Tel-Aviv, Israel, April 2000.
17. K. Nichols, V. Jacobson, and L. Zhang. Two-bit differentiated services architecture for the Internet. IETF RFC 2638, July 1999.
18. A. K. Parekh and R. G. Gallagher. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.
19. S. Sahu, P. Nain, D. Towsley, C. Diot, and V. Fioroiu. On achievable service differentiation with token bucket marking for TCP. In *Proceedings of ACM SIGMETRICS 2000*, pages 23–33, Santa Clara, CA, June 2000.
20. K. Schittkowski. NLPQL: A FORTRAN subroutine solving constrained nonlinear programming problems. *Annals of Operations Research*, 5:485–500, 1986.
21. L. Zhang. Virtual clock: A new traffic control algorithm for packet switched networks. *IEEE/ACM Trans. Comput. Syst.*, 9(2):101–125, May 1991.