JoBS: Joint Buffer Management and Scheduling for Differentiated Services

Jörg Liebeherr, Nicolas Christin, *Department of Computer Science, University of Virginia*

Abstract— **A novel algorithm, called JoBS (Joint Buffer Management and Scheduling), is presented for loss and delay differentiation of traffic classes in a packet network. JoBS has two unique capabilities: (1) JoBS makes scheduling and buffer management decisions in a single step, and (2) JoBS supports both relative and absolute QoS requirements of classes. The JoBS algorithm is presented in terms of the solution to an optimization problem. Numerical simulation examples, including results for a heuristic approximation of JoBS, are presented to illustrate the effectiveness of the approach and to compare JoBS to existing methods for loss and delay differentiation.**

Keywords— **Buffer Management, Scheduling, Service Curves, Quality-of-Service, Service Differentiation.**

I. Introduction

THERE are two important criteria for classifying Quality-of-Service (QoS) guarantees in packet networks. The first criteria 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 criteria is whether guarantees are expressed with reference to guarantees given to other flows/flow classes (*relative QoS*) or if guarantees are expressed in absolute terms (*absolute QoS*).

Efforts to provision for QoS in the Internet in the early and mid-1990s, which resulted in the IntServ 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. An important argument in favor of relative per-class QoS guarantees is that they do not require admission control or traffic policing. Since late 1997, the *Differentiated Services* (DiffServ) [2] working group has discussed several proposals for per-class relative QoS guarantees [4], [16], [15].

Most 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 than others, but do not quantify the service 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 relative QoS notions [7], [13], [14]. Probably the best known such effort is the *proportional service differentiation* model, proposed by Dovrolis, Stiliadis, and Ramanathan, which tries to enforce that the ratios of delays [7] and loss rates [6] of successive priority classes is 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 relative per-class QoS within a formal framework inspired by Cruz's service curves [5]. Using this approach, we present a scheduling/dropping algorithm, 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. JoBS operates as follows. Whenever there is an arrival jobs makes prediction on the delays of the currently backlogged traffic, and then adjusts the service rate allocation to classes and the amount of traffic to be dropped. A unique feature of JoBS is that it considers scheduling and buffer management (dropping) together in a single step. We also present a heuristic approximation of the JoBS algorithm.

This work is supported in part by the National Science Foundation through grants NCR-9624106 (CAREER), ANI-9730103, and ANI-9903001.

This paper is organized as follows. In Section II we give an overview of the state-of-the-art of relative per-class QoS guarantees. Then, in Sections III and IV, we specify the JoBS framework. In Section V we present a heuristic approximation of JoBS. In Section VI we present simulation scenarios to evaluate the effectiveness of JoBS. In Section VII we present brief conclusions.

II. Related Work

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

SCHEDULING: The majority of work on per-class relative QoS suggests to use a fixed-priority, e.g., [16], or rate-based scheduler, e.g., [9], and the number of scheduling algorithms that have been specifically designed for relative delay differentiation is small. The Proportional Queue Control Mechanism (PQCM) [13] uses the backlog of classes to determine the service rate allocation. Similarly, Backlog-Proportional Rate (BPR) [7] is a variation of the GPS algorithm [17] which sets the service rates of classes proportional to their backlog. Both schemes bear similarity to the scheduling component of JoBS, in the sense that they dynamically adjust service rate allocations to meet relative QoS requirements.

Waiting-Time Priority (WTP), also presented in [7], implements a well-known scheduling algorithm with dynamic time-dependent priorities ([10], Ch. 3.7). The Mean-Delay Proportional scheduler (MDP) [14] also uses a dynamic priority mechanism, but sets priorities based on the average experienced delay of packets.

With respect to these schedulers, a distinguishing feature of JoBS is that it not only considers the current state and the past history of the link, but, in addition, uses the current state to make predictions on future delays and backlog.

BUFFER MANAGEMENT: For a discussion of buffer management algorithms proposed in an IP and/or ATM context, we refer to a recent survey article [11]. Proposals for buffer management (active queue management) in IP networks are often motivated with the need to improve TCP performance (e.g., RED [8], FRED [12], REM [1]), while some techniques have been specifically targeted for class-based service differentiation (RIO [4], multiclass RED [18]). Of these schemes, REM is closest in spirit to the dropping algorithm in JoBS, since it treats the problem of marking (or dropping) arrivals as an optimization problem.

The Proportional Loss Rate (PLR) dropper [6] has been 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 this scheme. PLR(M) uses only the last M packets for estimating the loss rates of a class, whereas PLR(∞) has no such memory constraints.

Most of the work on relative per-class service differentiation considers delay and loss differentiation as orthogonal issues. A notable exception is the most recent revision of the *Proportional Differentiated Services* model [6], which provides mechanisms for both proportional dropping and delay differentiation. However, scheduling and dropping in this scheme are treated independently by separate algorithms.

III. The Joint Buffer Management and Scheduling Framework

In this section, we introduce our framework of *Joint Buffer Management and Scheduling* (JoBS), for scheduling and buffer management at the output link of a router. We will first give an informal discussion of the operations of JoBS, and then provide a detailed description.

A. Overview of JoBS

JoBS assumes per-class buffering of arriving traffic, and serves traffic from the same class in a First-Come-First-Served order. JoBS also assumes the availability to perform rate-based scheduling with service rate guarantees to classes [5], [17]. Within the context of JoBS, there is no admission control and no policing of traffic.

The set of relative or absolute performance requirements are given to the JoBS algorithms 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 given to JoBS can be any mix of relative and absolute constraints. Note that absolute constraints may render a system of constraints infeasible. Then, some constraints may need to be relaxed. We assume that JoBS is provided with an order in which constraints should be relaxed in case of an infeasible state.

The JoBS algorithm operates as follows. For every arrival, JoBS makes a prediction on the delays of the backlogged traffic, and modifies the service rates so that all QoS and system constraints will be met. If changing the service rates is not sufficient for meeting all constraints, JoBS will drop either the arrival or it will drop queued traffic.

We find it convenient to view the operation of JoBS 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 is such that the amount of dropped traffic and changes to the current service rate allocation are minimized. 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. The optimization is performed for each arrival to the link.

The computational complexity of JoBS is determined by the number and the type of constraints and by the frequency of running the above optimization. To explore the principal properties of JoBS, we will, for now, assume that infinite computing resources are available. In a later section, we will approximate JoBS with a heuristic which incurs less computational overhead.

B. Formal Description of JoBS

Next we describe the basic operations of the algorithms for service rate adjustment and dropping algorithms at a JoBS link with capacity C and total buffer space B.

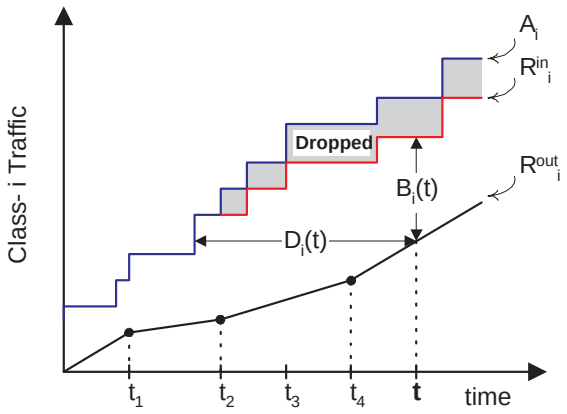


Fig. 1. **Delay and backlog.**

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. Let $a_i(t)$ and $\ell_i(t)$ denote the traffic arrivals and amount of dropped ('lost') traffic from class i at time t.

Let $r_i(t)$ denote the service rate allocated by JoBS to class i at time t. We assume that $r_i(t)$ is nonzero, only if there is a backlog of class-i traffic in the buffer, and we assume that scheduling in JoBS is workconserving, that is, $\sum_i r_i(t) = C$ if the backlog at time t is nonzero.

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. On the other hand, scheduling algorithms that closely approximate fluid-flow schedulers with rate-guarantees are readily available [17].

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 class i are defined as

$$A_i(t) = \int_0^t a_i(x)dx\ , \tag{1}$$

$$R_i^{in}(t) = A_i(t) - \int_0^t \ell_i(x)dx\ . \tag{2}$$

So, the difference between the arrival and input curve

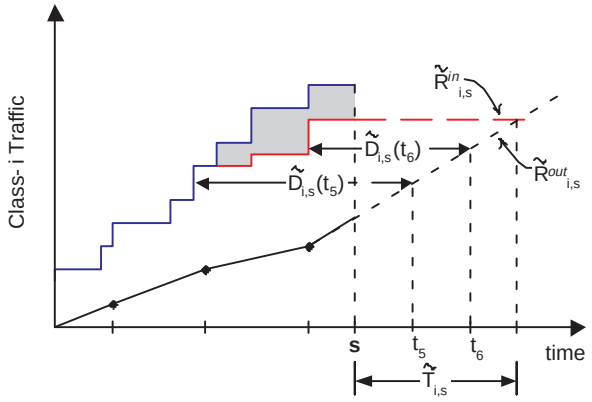


Fig. 2. **Projected input curve, projected output curve, and projected delays.** The projection is performed at time s for the time interval $[s, s + \tilde{T}_{i,s}]$.

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

$$R_i^{out}(t) = \int_0^t r_i(x)dx \ . \tag{3}$$

We refer to Figure 1 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 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

$$B_i(t) = R_i^{in}(t) - R_i^{out}(t) \ , \tag{4}$$
$$D_i(t) = \max_{x<t}\{x \mid R_i^{out}(t) \geq R_i^{in}(t-x)\} \ . \tag{5}$$

Upon a traffic arrival, say at time s, JoBS sets new service rates $r_i(s)$ and the amount of traffic to be dropped $\ell_i(s)$ for all classes, such that all QoS and system constraints can be met at times $> s$. To determine the rates, JoBS projects the delays of all queued traffic. For the projections, JoBS assumes that the current state of the link will not change after time s. Specifically, JoBS makes the following assumptions on the service, the arrival, and the drops (we indicate projected values by a tilde ($\tilde{}$)) for times $t > s$:

1. Service rates remain as they are: $\tilde{r_i}(t) = r_i(s)$,
2. There are no further arrivals: $\tilde{a_i}(t) = 0$,
3. There are no further packet drops: $\tilde{\ell_i}(t) = 0$.

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:

$$\begin{aligned}\tilde{R}^{in}_{i,s}(t) &= R_i^{in}(s) \ , \\ \tilde{R}^{out}_{i,s}(t) &= R_i^{out}(s) + (t-s)r_i(s) \ , \\ \tilde{B}_{i,s}(t) &= \tilde{R}^{in}_{i,s}(t) - \tilde{R}^{out}_{i,s}(t) \ .\end{aligned}$$

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.,

$$\tilde{T}_{i,s} = \min_{x>0}\{x \mid \tilde{B}_{i,s}(s+x) = 0\} \ . \tag{6}$$

With this notation, we can now make predictions for the (future) delays in the time interval $t \in [s, s + \tilde{T}_{i,s}]$. We define the projected delay $\tilde{D}_{i,s}$ as

$$\tilde{D}_{i,s}(t) = \max_{s<x<s+\tilde{T}_{i,s}} \{x \mid \tilde{R}^{out}_{i,s}(t) \geq R_i^{in}(t-x)\} \ . \tag{7}$$

Note that, if there are no arrivals after time s, the delay projections are correct. In Figure 2, we illustrate the projected input curve, projected output curve, 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.

IV. Service Rate Adaptation and Drop Algorithm in JoBS

In this section we discuss how JoBS adjusts the service rates of classes and decides if and how much traffic to drop. The algorithm will be expressed in terms of an optimization problem.

Each time s, when an arrival occurs, a new optimization is performed. The optimization variable is a time-dependent vector $\mathbf{x}_s$ which contains the service rates $r_i(s)$ and the amount of traffic to be dropped $\ell_i(s)$,

$$\mathbf{x}_s = (r_1(s) \dots r_Q(s) \ \ell_1(s) \dots \ell_Q(s))^T \ . \tag{8}$$

The optimization problem has the form

$$\begin{array}{lll} \text{MINIMIZE} & F(\mathbf{x}_s) & \\ \text{SUBJECT TO} & g_k(x) = 0, & k = 1, \ldots, M \\ & h_k(x) \geq 0, & k = M+1, \ldots, N. \end{array}$$

where $F(.)$ is an objective function, and the g_k's and h_k's are constraints.

The objective function of JoBS will be stated such that JoBS minimizes the amount of dropped traffic, and keeps the changes to the current service rate allocation small. The constraints are QoS constraints and system constraints. The optimization at time s is done with knowledge of the system state before times s, that is, the optimizer knows R_i^{in} and R_i^{out} 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. We will use the optimization problem as a reference system that provides a benchmark, against which practical scheduling and dropping algorithms can be compared.

A. 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.

- *Buffer size*: The total backlog cannot exceed the buffer size B, that is, $\sum_i B_i(t) \leq B$ for all times t.
- *Workconserving property:* At a workconserving link $\sum_i r_i(t) = C$ holds for all times t where $\sum_i B_i(t) > 0$. This constraint is stronger than the limit given by the link capacity C, i.e., $\sum_i r_i(t) \leq C$.
- *Other bounds:* Rates and packet drops are nonnegative. Also, the amount of traffic that can be dropped is bounded by the current backlog. So we obtain $r_i(t) \geq 0$ and $0 \leq \ell_i(t) \leq B_i(t)$ for all times t.

QOS CONSTRAINTS. JoBS allows two types of QoS constraints, relative constraints and absolute constraints. Also, QoS constraints can be expressed for delays and for loss rates. The number and type of QoS constraints is not limited by JoBS. However, absolute QoS constraints may result in an infeasible system of constraints. In such a situation, one or more constraints must be relaxed or eliminated. 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 is feasible. In addition, QoS constraints for classes which are not backlogged are simply ignored.

- *Absolute Delay Constraints (ADC):* These constraints enforce that the projected delays of class i satisfy a worst-case bound d_i.

$$\max_{s<t<s+\tilde{T}_{i,s}} \tilde{D}_{i,s}(t) \leq d_i \ , \tag{9}$$

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

- *Relative Delay Constraints (RDC):* These constraints specify the proportional delay differentiation between classes. As an example, for two classes 1 and 2, the proportional delay differentiation 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, it could be specified as the delay of the packet at the head of the class-i queue, the maximum projected delay as in Eqn. (9), 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:

$$\overline{D}_{i,s} = \frac{1}{\tilde{T}_{i,s}} \int_s^{s+\tilde{T}_{i,s}} \tilde{D}_{i,s}(x)dx \ . \tag{10}$$

To provide some flexibility in the scheduling decision, we do not enforce relative delay constraints strictly, but allow for some slack. The relative delay constraints are of the form

$$k_i(1-\varepsilon) \leq \frac{\overline{D}_{i+1,s}}{\overline{D}_{i,s}} \leq k_i(1+\varepsilon) \tag{11}$$

for $i = 1, \ldots, Q-1$, where $k_i > 1$ is the target differentiation factor and ε ($0 \leq \varepsilon \leq 1$) indicates

a tolerance level. If relative constraints not specified for some classes, the constraints are adjusted accordingly.
Note that in the delay constraints in Eqs. (9) and (11), 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' in this context. For this paper, we select one specific 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]

$$\begin{aligned} p_{i,s} &= \frac{\int_{t_0}^{s} \ell_i(x)dx}{\int_{t_0}^{s} a_i(x)dx} \\ &= 1 - \frac{R_i^{in}(s^-) + (a_i(s) - \ell_i(s)) - R_i^{in}(s - t_0)}{A_i(s) - A_i(s - t_0)} \end{aligned} \tag{12}$$

In the last equation, all values except $\ell_i(s)$ are known at time s.

- *Absolute Loss Constraints (ALC):* An ALC specifies that the loss ratio of class i, as defined above, never exceeds a limit L_i. That is, if $B_i(s) > 0$, then

$$p_{i,s} \leq L_i \ . \tag{13}$$

- *Relative Loss Constraints (RLC):* The RLCs specify the desired proportional loss differentiation between classes. Similar to the RDCs, we provide a certain slack within these constraints. If, at time s, $B_i(s) > 0$ and $B_{i+1}(s) > 0$, then the RLC for classes $i+1$ and i has the form

$$k_i'(1 - \varepsilon') \leq \frac{p_{i+1,s}}{p_{i,s}} \leq k_i'(1 + \varepsilon') \ , \tag{14}$$

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

Remark: JoBS only specifies the amount of traffic which should be dropped from a particular class, however, JoBS does not select the position in the queue from which to drop traffic. For example, drops may occur from the head of the queue (Drop-from-Front) or from the tail (Drop-Tail) [11]. Note that such a policy has an impact on the shape of the input curve. The definitions in Section III assume a Drop-Tail policy.

[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 $\max_{y<x}\{\sum_i B_i(y) = 0\}$.

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

B. Objective Function

Provided that the QoS and system constraints can be satisfied, the objective function of JoBS selects 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:

- OBJECTIVE 1: Avoid dropping traffic,
- OBJECTIVE 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 formulation of the objective function expresses the above objectives in terms of a cost function.

$$F(\mathbf{x}_s) = \sum_{i=1}^{Q} (r_i(s) - r_i(s^-))^2 + C^2 \sum_{i=1}^{Q} \ell_i(s) \ , \tag{15}$$

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. (15) we need to use the quadratic form $(r_i(s) - r_i(s^-))^2$, since $\sum_i (r_i(s) - r_i(s^-)) = 0$ for a workconserving link with a backlog at time s. The scaling factor C^2 in front of the second sum of Eqn. (15) 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 [19].

V. HEURISTIC APPROXIMATION OF JOBS

We next present a heuristic that approximates the JoBS algorithm, yet, which has significantly lower computational complexity. Our goal is not to present an algorithm that is readily implementable and can operate at current line rates. Instead, we want to demonstrate that it is feasible to find relatively simple algorithms that can closely approximate the idealized JoBS system. The presented heuristic can be thought of as the first step towards a router implementation.

Remark: The translation of the fluid-flow service model of JoBS into a packet-level architecture is done using the well-known technique of assigning virtual deadlines to packets which are computed from the current rate allocation [17], [20]. Note that a change to the service rate of a class may require to update the virtual deadlines of already queued packets.

Our heuristic algorithm completely avoids running the optimization from Section IV. Instead, the heuristic maintains the current rate allocation until a buffer overflow occurs or a delay violation is predicted. At that time, the heuristic picks a new feasible rate allocation. Unless there is a buffer overflow, the tests for violations of ADCs and RDCs[3] 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 3. The algorithm is broken up into a number of small computations.

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

$$\sum_i \ell_i(s) = \text{Size of arriving packet} . \qquad (16)$$

The heuristic picks a solution for the $\ell_i(s)$ which satisfies Eqn. (16) and the RLCs in Eqn. (14), where ε' is set to zero to simplify the search for a solution. If the solution violates an ALC, the RLCs are relaxed until all ALCs are satisfied. Once the $\ell_i(s)$'s are determined the algorithm continues with a test for delay constraint violations, as shown in Figure 3.

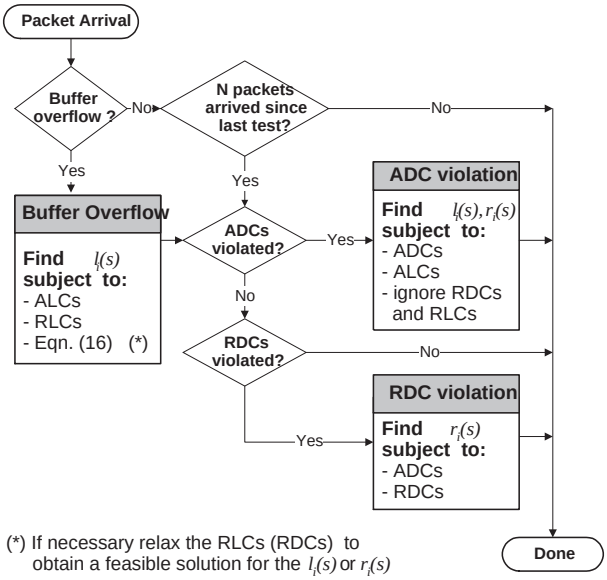


Fig. 3. **Outline of the Heuristic algorithm**.

If there are no buffer overflows, the algorithm makes projections for delay violations (ADC and RDC) only once for every N packets. The tests use the current service rate allocation to predict future violations. For delay constraint violations, the heuristic distinguishes the following three cases:

NO VIOLATION: In this case, the service rate allocation remains unchanged.

RDC VIOLATION: If some RDC (but no ADC) is violated, the heuristic algorithm determines new rate values. Here, the RDCs as defined in Eqn. (11) are transformed into equations by setting $\varepsilon = 0$. Together with the workconserving property ($\sum_i r_i(s) = C$), 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.

[3] Recall: *ADC* = absolute delay constraint, *RDC* = relative delay constraint, *ALC* = absolute loss constraint, and *RLC* = relative loss constraint.

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 simply ignores all relative QoS constraints (RLCs, RDCs) when an ADC violation occurs, and only tries to satisfy ALCs and ADCs.

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 following bound, which is easily verified by referring to Figures 1 and 2.

$$\max_{s<x<s+\tilde{T}_{i,s}} \tilde{D}_{i,s}(x) \leq D_i(s) + \frac{B_i(s)}{r_i(s)} \,. \qquad (17)$$

Then, using $B_i(s) = B_i(s^-) + a_i(s) - l_i(s)$, the following is a sufficient condition for satisfying the ADC of class i with delay bound d_i at time s.

$$\underbrace{\frac{1}{r_i(s)} \frac{B_i(s^-) + a_i(s) - l_i(s)}{d_i - D_i(s)}}_{\rho_i} \leq 1 \,. \qquad (18)$$

The heuristic algorithm will select the $r_i(s)$ and $l_i(s)$ such that Eqn. (18) 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.(18). 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, after the redistribution of rates, there are still classes i with $\rho_i > 1$, we increase $l_i(s)$ until $\rho_i < 1$ for those classes. To minimize the number of dropped packets, $l_i(s)$ is never increased to a point where an ALC is violated.

VI. EVALUATION

We present an evaluation of the JoBS algorithm via simulation. Our goals are (1) to determine if and how well JoBS provides the desired service differentiation; (2) to determine how well the heuristic algorithm from Section V approximates the optimization of JoBS; and (3) to compare JoBS with existing proposals for proportional differentiated services.

In the simulations, we compare the performance of the following three schemes.

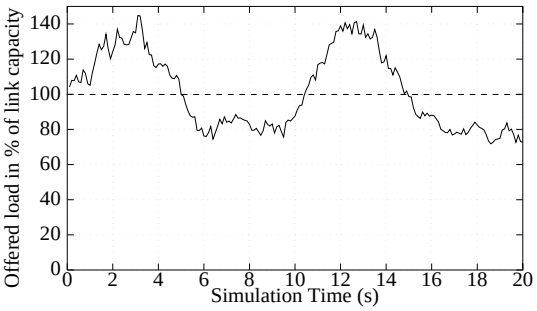


Fig. 4. **Offered Load.**

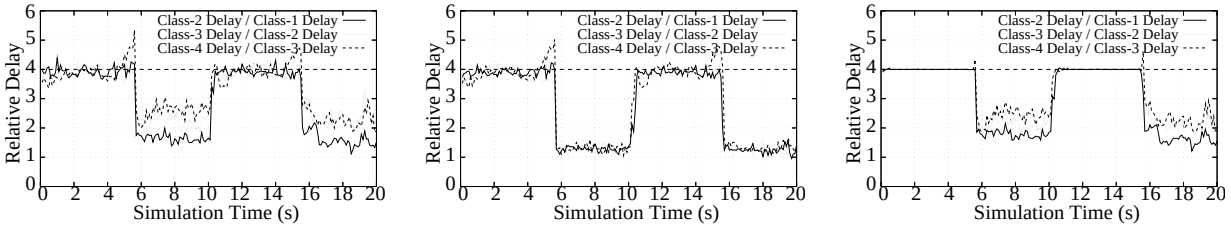
- JOBS (OPTIMIZATION): This is the optimization described in Section IV.
- JOBS (HEURISTIC): This is the heuristic algorithm discussed in Section V. The rates are recalculated every $N = 100$ packets, unless there is a buffer overflow.
- WTP/PLR(∞) [7]: Among the considered schemes for relative service differentiation (for scheduling: MDP [14], WTP [7], BPR [7], for dropping: PLR(M), PLR(∞) [6], Drop-Tail), we found that WTP/PLR(∞) provided uniformly the best results. Thus, we use this scheme to represent the state-of-the-art.

We present two simulation experiments. In the first experiment, we compare and contrast the relative differentiation provided of JoBS (optimization), JoBS (heuristic), and WTP/PLR(∞) without specifying absolute constraints. In the second experiment, we augment the set of constraints by absolute delay constraints on the highest priority class, and show that JoBS can effectively provide both relative and absolute differentiation.

A. Experimental Setup

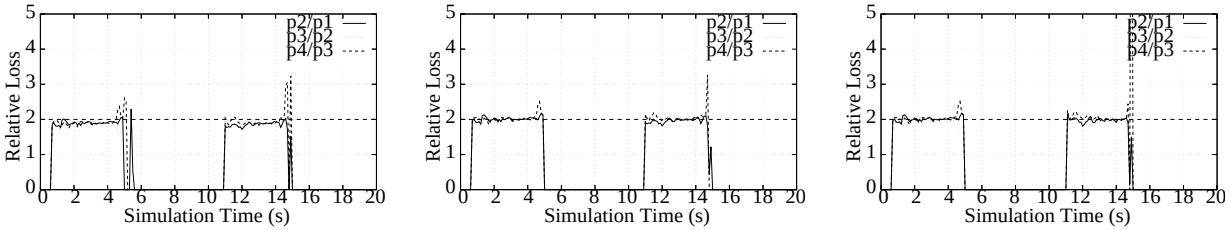
We consider a single output link with capacity $C = 1$ Gbps and a buffer size of 6.25 MByte. We have $Q = 4$ classes. We use the same load curve in all experiments. The length of each experiment is 20 seconds of simulated time, starting at time 0 with an empty system.

The incoming traffic is composed of a superposition of Pareto sources with a parameter $\alpha = 1.2$ and an average interarrival time of 300 μs. These sources generate packets with a fixed size of 125 Byte. As offered load, we generate a time-varying load curve, where the number of active sources follows a sinusoidal pattern with period $T = 10s$. The offered



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

Fig. 5. **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. 6. **Experiment 1: Relative Loss Differentiation.** The graphs show the ratios of loss rates for successive classes. The target value is $k' = 2$.

load used in our experiments is plotted in Figure 4. Between 200 and 550 sources are active at the same time, resulting in an offered load comprised between 75% and 145% of the link capacity. The load from the classes is symmetric, that is, at each time, each class generates 25% of the aggregate load.

B. Experiment 1: Relative Differentiation Only

The first experiment focuses on relative service differentiation, and does not include absolute constraints. The objective for the relative differentiation are

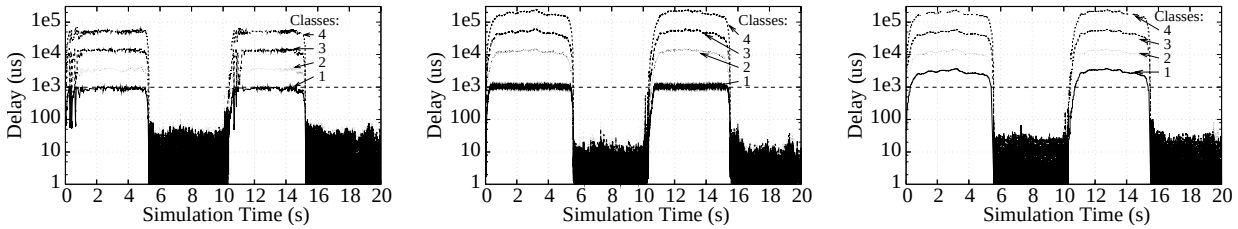
$$\text{Delay of Class } (i+1)/\text{Delay of Class } i \approx 4 \ ,$$
$$\text{Class-}(i+1) \text{ Loss Rate}/\text{Class-}i \text{ Loss Rate} \approx 2 \ .$$

Thus, for JoBS, the parameters in the RDCs and RLCs are set to $k_i = 4$ and $k'_i = 2$ for all i. The tolerance levels are set to $\varepsilon = 0.001$ in JoBS (optimization), $\varepsilon = 0.01$ in JoBS (heuristic), $\varepsilon' = 0.05$ in JoBS (heuristic) and JoBS (optimization). The results of the experiment are presented in Figures 5 and 6, 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 present averages over moving time windows of size 0.1 s (This measure is adopted from [6]).

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 mostly due to the tolerance values ε and ε'. Note that 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 algorithm.

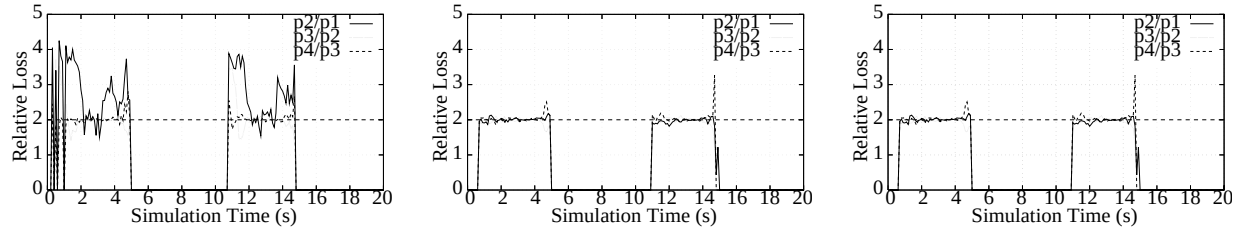
When the system load is low, that is, in time intervals $[6\ s, 10\ s]$ and $[16\ s, 20\ s]$, JoBS (heuristic) is not effective for providing delay differentiations. Here, JoBS (optimization) and WTP/PLR(∞) still manage to achieve some delay differentiation, albeit far from the target values. One should note, however, that, at an underloaded link, the absolute values of the delays are very small for all classes. In Figure 6, we observe that both WTP/PLR(∞) and JoBS (optimization) show some transient oscillations in the time interval $[5\ s, 6\ s]$, while JoBS (heuristic) does not seem to suffer from this problem as much.

We have not presented graphs for the total loss rate in the simulations. 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. Without presenting additional plots, we state that, in the simulations, the loss rate of all schemes are very similar. Likewise, the absolute values for the delays are comparable in all schemes.



(a) With ADC, all RDCs. (b) With ADC, one RDC removed. (c) No ADC, all RDCs.

Fig. 7. **Experiment 2: Absolute Delay Differentiation.** The graphs show the delays of all packets. All results are for JoBS (heuristic).



(a) With ADC, all RDCs. (b) With ADC, one RDC removed. (c) No ADC, all RDCs.

Fig. 8. **Relative Loss Differentiation.** The graphs show the ratios of loss rates for successive classes. The target value is $k' = 2$. All results are for JoBS (heuristic).

C. Experiment 2: Relative and Absolute Differentiation

In this experiment, we evaluate how well JoBS can satisfy a mix of absolute and relative delay constraints. In this experiment, we only present results for JoBS (heuristic). Note that WTP/PLR(∞), or other schemes from the literature, do not support both relative and absolute guarantees.

We consider the same simulation setup and the same relative constraints (RDCs and RLCs) as in Experiment 1, but add an absolute delay constraint (ADC) for Class 1 with a delay bound of

$$d_1 = 1,000 \mu s.$$

We call this scenario "with ADC, all RDCs". Note that, with the given relative delay constraints from Experiment 1, the other classes have implicit absolute delay constraints, which are approximately[4] 4,000 μs for Class 2, 16,000 μs for Class 3, and 64,000 μs for Class 4.

These 'implicit' absolute constraints can be avoided, by removing the RDC which governs the ratio of the delays between Class 2 and Class 1. We present results for such a constraint system as well, and denote this constraint set as "with ADC, one RDC removed". For reference purposes, we also include the results for JoBS (heuristic) from Experiment 1. We refer to this constraint set as "no ADC, all RDCs".

In Figure 7 we plot the absolute delays of all packets, and in Figure 8 we plot the ratios of the rates for successive classes. The plotted ratios of loss rates are time averages over intervals of length 0.1 s.

Our discussion will focus on the delay values in Figure 7. Figures 7(a) and 7(b) show that the absolute delay constraint of $d_1 = 1000$ is enforced in both cases. Figure 7(a) also shows that the JoBS heuristic maintains the relative delay differentiation for the other classes, thus, enforcing the 'implicit' delay constraints of (approximately) 4,000, 16,000, and 64,000 μs for Classes 2, 3, and 4, respectively. A problem with having a large number of absolute delay constraints is that the system of constraints easily becomes infeasible. Indeed, the delay fluctuations in Figure 7(a) and the violations of the RLCs in Figure 8(a) are due to an infeasible set of constraints.

In the constraint set 'with ADC, one RDC removed', we remove the RDC for the delay ratio between Class 2 and Class 1, thereby, removing the 'implicit' bounds on the worst case delays for Classes 2, 3, and 4. Figure 7(b) shows that the JoBS heuristic handles this set of constraints as one would expect. The ADC for Class 1 is enforced, as are the RDCs for Classes 2, 3, and 4. Note that no proportional delay differentiation is enforced for Class 1. In fact the ratio of Class 2 delays and Class 1 delays exceeds a factor of 10 at high loads.

[4] Due to the tolerance value ε the exact values are multiples of 1,000.

For reference purposes, we present the results for the constraint system from Experiment 1, that is, without having any ADCs. Figure 7(c) shows that, without the ADC, the delays for Class 1 are as high as $5,000\mu s$. [5]

VII. Discussion and Conclusions

The main contribution of this paper is a new framework, referred to as JoBS (Joint Buffer Management and Scheduling), for reasoning about relative and absolute per-class service differentiation in a network without information on traffic arrivals. JoBS reconciles scheduling and buffer management into a single algorithm, thus, acknowledging that scheduling and buffer management are not orthogonal issues, but should be dealt with in concert. JoBS makes predictions on the delays of backlogged traffic, and uses the predictions to update the service rates of classes and the amount of traffic to be dropped. A unique capability of JoBS is its ability to provide relative and absolute per-class service differentiation for delays and loss rate. We have demonstrated the effectiveness of JoBS in a set of simulation experiments.

As future work, we are interested in extending the JoBS approach to support TCP congestion control. As a point of departure, we conjecture that many active queue management algorithms, e.g., RED [8] and RIO [4], can be expressed within the JoBS framework. We are also working towards an implementation of JoBS-style algorithms on PC-based IP routers.

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. RFC 2475: An architecture for differentiated services, December 1998.

[3] R. Braden, D. Clark, and S. Shenker. RFC 1633: Integrated services in the internet architecture: an overview, 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] The delay values for Classes 2, 3, and 4 in Figures 7(b) and (c) appear similar, especially, since we use a log-scale. We emphasize that the values are consistent.

[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 and P. Ramanathan. Proportional differentiated services, part II: Loss rate differentiation and packet dropping. In *Proceedings of IWQoS*, Pittsburgh, PA., June 2000.

[7] C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional differentiated services: Delay differentiation and packet scheduling. In *Proceedings of ACM SIGCOMM '99*, pages 109–120, Boston, MA., August 1999.

[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] L. Kleinrock. *Queueing Systems. Volume II: Computer Applications*. John Wiley & Sons, New York, NY, 1976.

[11] M. A. Labrador and S. Banerjee. Packet dropping policies for ATM and IP networks. *IEEE Communications Surveys*, 2(3), 3rd Quarter 1999. http://www.comsoc.org/pubs/surveys.

[12] D. Lin and R. Morris. Dynamics of random early detection. In *Proceedings of ACM SIGCOMM '97*, pages 127–137, Cannes, France, September 1997.

[13] 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.

[14] T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Barghavan. Delay differentiation and adaptation in core stateless networks. In *Proceedings of IEEE INFOCOM 2000*, pages 421–430, Tel-Aviv, Israel, April 2000.

[15] K. Nichols, S. Blake, F. Baker, and D. Black. RFC 2474: Definition of the differentiated services field (DS field) in the IPv4 and IPv6 headers, December 1998.

[16] K. Nichols, V. Jacobson, and L. Zhang. RFC 2638: Two-bit differentiated services architecture for the Internet, July 1999.

[17] 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.

[18] 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.

[19] K. Schittkowski. NLPQL: A FORTRAN subroutine solving constrained nonlinear programming problems. *Annals of Operations Research*, 5:485–500, 1986. Edited by Clyde L. Monma.

[20] L. Zhang. Virtual clock: A new traffic control algorithm for packet switched networks. *IEEE/ACM Trans. Comput. Syst.*, 9(2):101–125, May 1991.