A Combined Delay and Throughput Proportional Scheduling Scheme for Differentiated Services[*]

Ahmed E.Kamal[†] and Samyukta Sankaran
Department of Electrical and Computer Engineering
Iowa State University
Ames, IA 50011
{kamal,samyukta}@iastate.edu

Abstract

The proportional differentiation model is a service differentiation approach for differentiated services networks that includes several desirable features such as controllability and predictability. This paper proposes and evaluates a scheduling mechanism for the combined control of delay and throughput metrics, within the context of the proportional differentiation model. The scheme is based on the well known Little's Law, and always exercises average delay differentiation, while it implements throughput differentiation only during overload periods, and if certain conditions as satisfied. A moving window averaging mechanism and an active queue management scheme are simultaneously, and respectively used to achieve control over the relative throughput values as well as the relative delay values between classes. The scheme does away with measurement of the actual packet delays, and state information is minimized. Some feasibility bounds are presented under the assumption of Poisson arrivals, and a simulation study shows the effectiveness of this scheme.

Keywords: Differentiated services (DiffServ); Proportional differentiation; Quality of Service (QoS); Little's result; Performance evaluation; Simulation.

I Introduction

This section gives a brief introduction to Internet Quality of Service, and overviews some recently proposed approaches towards achieving this goal.

I.1 Internet QoS: Needs and Solutions

The current model of the Internet provides only for best-effort service, which makes no guarantees on packet delivery and provides no bounds on the time of packet arrival. While this is adequate for non-real time applications (such as file transfer and e-mail), the proliferation of new applications with varying time and capacity requirements has exposed two weaknesses of this model [29]:

[*]This work was sponsored in part by a Carver Trust grant from Iowa State University.
[†]Corresponding author: telephone: (515) 294-3580; fax: (515) 294-1152; e-mail: kamal@iastate.edu

- *lack of performance assurance*: A critical limitation of the packet switched IP network is that it cannot, by itself, provide assurances that resources will be available at the time communication is desired.

- *lack of service differentiation*: Currently, all users experience the same level of (best-effort) service. However, service providers would like to be able to offer their customers different service tiers, depending on the needs of the user (e.g. a controlled jitter service for streaming video applications).

Research efforts towards achieving QoS, have therefore focused on addressing these limitations, mostly using resource allocation techniques. The Internet Engineering Task Force (IETF) has introduced the following two models that can be used to achieve QoS:

- *Integrated Services (IntServ)*: The primary motivation of *IntServ* [3] was to satisfy the end-to-end requirements of real-time applications, which needed absolute performance guarantees. Individual user flows use a reservation setup protocol (such as RSVP [4]) to reserve the required resources along the routing path. The IntServ approach was the first viable option to achieve QoS, and has the advantages of providing absolute guarantees with a fine per flow granularity. However, per flow book keeping and scheduling is needed at each router, and such overhead may be impractical for shortlived flows. The lack of scalability of this model hinders its widespread deployment.

- *Differentiated Services (DiffServ)*: The *DiffServ* architecture [2] is a lightweight and scalable approach, in which user flows with similar performance requirements are aggregated into a limited number of service classes. Each class is marked with a suitable DiffServ codepoint (DSCP), and DiffServ-aware routers employ class sensitive packet forwarding mechanisms that provide service differentiation on a per hop basis (called per hop behavior (PHB)). Since state information is maintained for a limited number of classes, DiffServ has better scalability than IntServ.

I.2 Types of Differentiated Services

Research within DiffServ has been proceeding along three broad directions [14]:

i. *Absolute DiffServ*: Absolute DiffServ is an attempt to meet the same goals as IntServ, of providing absolute performance levels, but without the taxing maintenance of per flow state at the routers. This can be achieved by providing a strict limit on the resources that the user can expect to receive from the network. The Expedited Forwarding PHB approach [12, 5][1] as specified by the IETF, is an example of Absolute DiffServ which allocates minimum amounts of service rates to different EF classes at core routers.

ii. *Relative DiffServ*: This is a more flexible approach than the Absolute model, which provides assurances for the relative quality according to some performance metric, rather than absolute service levels. Traffic is grouped into N classes such that class i is better than, or at least no worse than, class *(i-1)*, for $1 < i \leq N$. The comparison is made on the basis of some meaningful performance metric, such as queuing delay, aggregate throughput, or packet loss. Users cannot be guaranteed a specific level of performance, but get the assurance that higher classes are allocated more resources than lower classes. Price differentiation, capacity provisioning, and strict prioritization are some mechanisms that can be used to achieve Relative Differentiation. The RED with In and Out packets (RIO) scheme [8] is an example of relative service differentiation in terms of packet loss.

iii. *Proportional DiffServ*: A refinement of the Relative DiffServ model, Proportional DiffServ [14] aims at achieving two goals :

   - *Predictability*: Consistent differentiation independent of class loads. Higher classes should always be better, or at least no worse, than lower classes, based on some performance metric.
   - *Controllability*: Ability to externally adjust the quality spacing between classes. Network operators must have *tuning knobs* to adjust the quality spacing between classes, based on pricing and policy.

   The Proportional DiffServ model controls some chosen class performance metric, e.g., average delay, loss rate, or throughput, proportional to the differentiation parameters which are determined by the network operator[2]. For example, if p_i is the chosen performance metric for class i, for $1 \leq i \leq N$ where N is the number of classes, and c_i is the corresponding quality

[1]The Expedited Forwarding PHB earlier definition presented in [21] was not precise, and may label an EF conformant service scheme as non-conformant. RFC 3246 [12] introduced a formal definition.

[2]Differentiation parameters are like weights, and are meant to specify the quality *distance* needed between classes.

differentiation parameter, then the proportional differentiation model attempts to achieve

$$\frac{p_i}{p_j} = \frac{c_i}{c_j}$$

over all pairs of classes, i and j, $1 \leq i, j \leq N$. Hence, even though actual service levels vary with class loads, service ratios are always maintained[3].

I.3 Proportional DiffServ: Previous Work

Following the introduction of the Proportional DiffServ model in [14], several proposals were introduced in order to proportionally control one or more performance metrics. This section will examine previous work in proportional control of a single performance metric.

- *Delay Differentiation: the BPR and WTP schedulers*

  Reference [16] proposes two schedulers to achieve *proportional delay differentiation.* Given that $\overline{w_i}$ is the mean queueing delay of a class i and W_i is the *Delay Differentiation Parameter* for class i such that $(W_1 > W_2 > ...W_N > 0)$, proportional delay differentiation was defined to satisfy $\frac{\overline{w_i}}{\overline{w_j}} = \frac{W_i}{W_j}$ over all pairs of classes. Two schedulers were proposed for the implementation of delay differentiation:

  i. *Backlog Proportional Rate (BPR) Scheduler* : Here, the class service rates are dynamically adjusted so that they are proportional to the backlog suffered by that class. The rationale behind this scheme is that, if a class has received a smaller amount of service than it *deserves* (based on its delay differentiation parameter and current load) in a recent time interval, then its queue will be proportionally larger. Serving this class will reduce its backlog, and therefore will help in reducing the unfairness in (average) delay for that class. Formally, the BPR proportionality constraint for N classes is:

  $$\frac{r_i(t)}{r_j(t)} = \frac{c_i}{c_j} \cdot \frac{q_i(t)}{q_j(t)}$$

  where $r_i(t)$ is the service rate for class i at time t, $q_i(t)$ is the length of the queue i at time t, and c_i is the *Scheduler Differentiation Parameter* for class i. This equation should hold for all pairs of classes, $1 \leq i, j \leq N$. Although this scheme is simple and requires very little complexity to implement, it has the drawback that, since a small relative

[3] Since in real systems one cannot expect the differentiation ratio to always hold, reference [6] suggested allowing some deviation from the target ratio, namely, $\frac{p_i}{p_j} = \frac{c_i}{c_j} \pm \Delta$, for all i and j.

backlog produces small service rates, queues that are just emptying or just filling up can experience larger delays.

ii. *Waiting Time Priority (WTP) Scheduler* : This is a priority scheduler for which the amount of time that a packet has been waiting in a queue, proportional to its differentiation parameter, determines its priority for service. Formally, $p_i(t) = w_i(t) \cdot c_i$, where $w_i(t)$ is the queuing delay of class i at time t. The class priorities have to be recalculated with every packet departure for all backlogged classes. This approach has the limitation that, in case of large bursts arriving in a higher class, short term starvation can occur for lower classes. This scheme was introduced and modeled in [23].

Recently, [17] studied the problem of allowing users to dynamically select their class of service such that the absolute mean delay requirements are satisfied. This paper also considered the class provisioning problem in order to satisfy the required proportional delay differentiation.

- *Loss Differentiation: Proportional Loss Rate Droppers*

In reference [15], the authors extended the proportional differentiation model developed for delay (above) to loss differentiation. Specifically, loss differentiation in the proportional model can be achieved if:

$$\frac{l_i}{l_j} = \frac{L_i}{L_j} \tag{1}$$

over all classes, where l_i is the packet loss rate for class i, and L_i is the Loss Differentiation Parameter for class i. A packet dropping mechanism must be developed to implement this.

The authors proposed a new dropping mechanism that is designed for proportional loss differentiation. The Proportional Loss Rate Dropper (PLR) is realized by interpreting equation (1) as having the normalized loss rate (l_i/L_i) being equal for all classes. The class from which to drop a packet is selected as the class which has the minimum normalized loss rate of l_i/L_i . This will cause equality of the normalized loss ratio over all classes. The authors suggested two versions of PLR dropper, depending on the length of time over which the loss rate l_i is measured. The *PLR(∞)* dropper maintains l_i as a long term fraction of dropped packets. The *PLR(M)* dropper computes l_i as the fraction of dropped packets in class i over the last M arrivals.

- *Throughput Differentiation: The MulTCP approach*

The issue of throughput proportional differentiation between classes, or flows, has not been widely discussed in the literature. Reference [11], which predates the formal introduction of the concept of proportional differentiation, proposed to implement bandwidth proportional differentiation between TCP flows using weighted proportional fairness, where the weight of each flow is related to the price paid by the user. A connection with a weight of C behaves like an aggregate of C TCP connections. Elements of the TCP control algorithm (linear increase phase, segment loss response, slow start phase) are modified so that the congestion window behaves as if it represents C TCP connections. Available bandwidth is thus distributed according to the different values of C chosen by the users. The problem with this approach, however, is that bandwidth assigned to TCP flows takes into account the window size only, and does not consider other important factors such as propagation delays.

An account of recent advances in proportional differentiated services is presented in [7].

This paper proposes a scheduler based on the Proportional DiffServ model. This model, as explained earlier, overcomes the difficulties associated with other DiffServ models, such as relative differentiation, and provides consistent differentiation and the ability to control *spacing* between classes. Although the Proportional DiffServ schemes mentioned so far handle a single performance metric, such as average delay or loss rate, separately, it is nonetheless desirable that an effective QoS solution should be able to jointly involve more than one significant performance metric. If each of the metrics is handled separately, it is difficult to predict how controlling one metric may affect the others. Therefore, we propose to combine the handling of multiple metrics. In particular, we propose a combined delay and throughput proportional differentiation. Delay differentiation is always exercised by the scheme. However, throughput differentiation is implemented only during overload periods, and if certain conditions are satisfied, where the classes are offered bandwidth shares consistent with their throughput proportional differentiation parameters. Otherwise, service rates are equal to packet arrival rates.

The paper is organized as follows. The next section will describe other proposals on combined proportional differentiation. We also describe the basis of our approach to achieve combined proportional differentiation of delay and throughput. Although we are not the first to propose combined proportional differentiation, our approach is different as will be explained in Section II. In Section III, we present our scheme, which includes a queue management scheme, as well as a packet scheduling mechanism. Throughout the development of the scheme, we present some feasibility bounds under the assumption of Poisson packet arrivals. Numerical examples based on a simulation model are presented in Section IV, and they involve more realistic packet arrival processes. Section V concludes the paper with a few remarks.

II Combined Proportional Differentiation

II.1 Previous Work

A few schemes for combined proportional differentiation involving different performance measures were proposed in the literature, e.g., [27, 28, 26, 6]. In [27] a single step joint scheduling and buffering management scheme was proposed for both average delay and loss rate proportional differentiation. The scheme also guarantees absolute bounds on the delay and loss rates. An exact scheme was formulated as a nonlinear optimization problem, which had two objectives: a primary objective of minimizing the dropped traffic, and a secondary objective of maintaining the current allocated rates. Due to the complexity of the exact approach, a heuristic approximation was then introduced. Reference [28] also introduced a heuristic for achieving combined delay and loss proportional differentiation. The heuristic is based on a modification of the Distance Based Priority (DBP) scheduler and the (m, k) model introduced in [19]. The (m, k) model makes sure that at least m packets out of every k packets meet their constraints, and the DBP scheduler assigns higher priority to flows/classes closer to a failing state in the (m, k) model, i.e., states in which fewer than m out of k packets meet their constraint. Two such schedulers, one for delay and one for loss, were used jointly in [28]. In [26] a probabilistic longest queue mechanism was introduced for combined loss and delay proportional differentiation. In this scheme, the queue size of class i, q_i, is measured, and class i is assigned a proportional weight, c_i. A packet is served, or dropped from class i with probability $f_i/\sum_j f_j$, where f_i is q_i/c_i for serving, and $\sqrt{q_i/c_i}$ for dropping. This increases the probability of serving a packet from class i if the queue size, and hence the delay increases, and/or when this class is assigned a small loss differentiation parameter. In [6] the authors, in addition to introducing a unified approach for several time-based heuristics for proportional delay differentiation, and a resetting mechanism of the dropper and scheduler, they also introduced a combined packet delay and loss ratio proportional differentiation scheme.

It is to be noted that an exact proportional differentiation based on loss ratios requires measuring the actual loss ratios, which can be very small. This is especially true when combined loss and delay differentiation is to be achieved, as will be shown in Theorem 1 below.

II.2 Little's Result and Proportional Differentiation

The proposed scheme in this paper has its basis in the fundamental result, first proven by J. D. C. Little, and known as Little's Result, e.g., see [22]. Consider any general queueing system, identified by the index i, with mean packet delay $\overline{w_i}$, mean number of packets in the system $\overline{q_i}$, and throughput s_i. Little's result states that at steady state

$$\overline{q}_i = s_i \cdot \overline{w}_i \quad . \tag{2}$$

This result makes no assumptions about the nature of arrival or departure processes in the queuing system. This result implies that if two of the performance metrics involved in equation (2) are changed, then the third is determined from this equation. Taking multiple classes into account, a straightforward application of the above gives:

$$\frac{\overline{q}_i}{\overline{q}_j} = \frac{s_i}{s_j} \cdot \frac{\overline{w}_i}{\overline{w}_j} \tag{3}$$

for every pair of classes, i and j. *Differentiation Parameters* can be associated with the performance metrics of each class, so that the problem of achieving delay and throughput proportional differentiation between classes can be reduced to a problem of enforcing equation (3).

II.3 Limitations and Choice of metrics

Combined proportional differentiation has its limitations with regard to the metrics that can be combined. Although it is desirable to be able to combine metrics such as delay, loss rate, and throughput in a proportional service differentiation scheme, with no, or very little dependence on the system state, there are certain limitations which are imposed by the following theorem, which is proven in Appendix A:

Theorem 1: *It is not possible to achieve combined proportional differentiation in the delay and loss metrics, independent of actual values of packet loss ratios.*

The contrapositive of the above theorem, which must also be true, means that any combined delay and loss proportional differentiation scheme must take into account the actual values of packet loss ratios, and all such schemes discussed in Subsection II.1 do so.

The problem with measuring packet loss rates is that such losses in modern high-performance networks are typically very small. Therefore, packet loss ratios are difficult to measure with accuracy, unless they are measured over a very long time interval, which considerably reduces their utility. Moreover, since a loss rate is a fraction of the offered load, a user can achieve a high throughput by injecting high offered traffic in the network, while still incurring the same loss rate, hence defeating the purpose of loss control.

Given the above, and since it is easier to measure throughput, queue size, and packet delays, we have chosen to implement *combined delay and throughput differentiation.* That is, since for every pair of classes i and j, equation (3) must hold, control of any two of the three ratios in this equation will result in a proportional control of the third. Moreover, controlling the mean delay of a class involves greater complexity, since router bookkeeping and delay measurement has to be done for each packet passing through the router. Also, accuracy of time measurement can vary widely between systems, as it depends on clock granularity. Therefore, overruling active delay control leads to the choice of controlling $\overline{q_i}$ and s_i as control knobs for achieving combined proportional differentiation of delay and throughput, and without actually measuring the delay. The strategies for exercising this control will be explained in the next section.

It should be pointed out that delay proportional differentiation is always exercised, while throughput control is exercised only when the system is overloaded, and the total offered load exceeds the service rate. Otherwise, a class throughput will be equal to its input rate.

III Strategies for combined Throughput and Delay Differentiation

This section discusses our strategies for controlling the queue and throughput proportions, hence achieving combined proportional differentiation. It also establishes bounds on the feasible proportions of average delays and throughput values under the assumption of Poisson arrivals[4]. In Section III.1 we present the notation and symbols used in this paper. In Section III.2 we present a packet scheduler with algorithms for separately controlling the throughput and queue ratios, while in Section III.3 we show how to integrate them. Section III.4 shows how the queue manager handles packet arrivals to a full buffer.

III.1 Notations

We consider an output port of a core router in a DiffServ system, implementing a PHB that will be described in the next section. This router's port has a shared buffer with size B, and serves

[4]Considering arrival processes other than Poisson will cause the process of obtaining bounds to be much more involved, and in most cases intractable. This is why all feasibility bounds obtained in the literature for proportional differentiation assumed the same Poisson packet arrival process [16, 25].

N classes, $1, 2, \dots, N$. The marking and classification is implemented at DiffServ ingress routers, and is beyond the scope of this paper. Packets in class i have a service time given by the random variable, b_i, with first and second moments given by $\overline{b_i}$ and $\overline{b_i^2}$, respectively. The mean arrival rate from class i is given by λ_i, and the offered load from this class is $\rho_i = \lambda_i \overline{b_i}$. Class i's throughput is given by s_i, where $s_i \leq \lambda_i$. Corresponding to s_i, the class i carried load is given by $\sigma_i = s_i \overline{b_i}$. The number of class i packets in the router's buffer is given by the random variable q_i, and its mean value is given by $\overline{q_i}$. The mean waiting time of packets belonging to this class is given by $\overline{w_i}$. Finally, the throughput and delay proportional differentiation parameters are given by S_i and W_i, respectively.

It is to be noted that we assume that moments of b_i, λ_i and ρ_i to be steady state values, and are characteristics of class i traffic. However, s_i is a measured value over a moving window, as will be explained next. If the window is sufficiently large, the measured throughput approaches the actual class steady state throughput. However, since our moving window is finite, there is always the possibility of incurring an error in s_i. As it is very well known, there is a tradeoff between short and long window sizes. We have therefore chosen our window size to be on the order of the time to serve all packets queued in the router's buffer, B. By mean waiting time, we refer to the waiting time averaged over the window size, while the instantaneous queue size, q_i is in fact used to represent the mean queue size. As will be shown below, our scheme attempts to maintain the ratio between the queue sizes of the different classes as close as possible to the target queue ratio, and hence the target ratio between mean queue sizes. The above symbols are summarized in Table 1.

III.2 Mechanisms for Departure - The Packet Scheduler

Serving a packet from a class, i, $1 \leq i \leq N$, changes the throughput ratios as well as the queue length ratios (hence, delay ratios) of associated classes. Therefore, the packet scheduler must be designed to control these ratios properly. In this section, we define two scheduling mechanisms to control each of these ratios separately. The choice of which of the two mechanisms to use depends on the system load, as will be described later. The mechanisms will be described for a pair of classes, i and j, but is in fact used for all other pairs, and will be shown in the pseudocodes of Figures (1) and (2).

- **Controlling s_i/s_j: The throughput control mechanism**

  To control the throughput of all classes proportionally, a moving window averaging mechanism

Table 1: List of symbols used in the paper

Symbol	Definition
$i,\ j$	class number indices, with $1 \leq i, j \leq N$
$\overline{b}_i$	mean service time for packets from class i
$\overline{b^2}_i$	2^{nd} moment of service time for packets from class i
λ_i	mean arrival rate from class i in packets/sec
ρ_i	offered load (Erlangs) from class $i = \lambda_i \overline{b}_i$
s_i	throughput of class i in packets/sec measured over a moving window
σ_i	carried load from class i in Erlangs $= s_i \overline{b}_i$
$\overline{w}_i$	mean packet delay for class i packets
q_i	random variable representing the number of packets from class i in the buffer
$\overline{q}_i$	mean number of packets from class i in the buffer
S_i	throughput weight for class i
W_i	mean delay weight for class i
B	buffer size

is used[5]. Throughput data for departing packets is collected from each class over a moving (or sliding) window. A window of size M time units slides over time, and is used to measure the throughput of packets from class i, $1 \leq i \leq N$, using the equation

$$s_i = \frac{\text{time spent on serving class } i \text{ packets during } M}{M} \quad .$$

We have chosen the moving window size to be equal to the average time it takes to serve all packets in the buffer, which is neither too long, nor too short.

When the throughput mechanism chooses a class to serve a packet from, it will base its choice of the class on minimizing the difference between the throughput ratios of all classes in the system and the target ratios, using min-max optimality. That is, packet departures are scheduled from that class which results in a maximum deviation in throughput ratios from the target, after service, which is minimal. For example, consider three classes, 1, 2 and 3,

[5]During the mechanism development, both the moving window and jumping (discrete) window approaches were considered. It was found that the discrete window mechanism offered abrupt, sharp changes of throughput periodic measurements, which led to inadequate correction of the desired throughput ratios. We have, therefore, adopted the moving window mechanism.

and assume that only packets from classes 1 and 2 are queued. Let
$|\frac{s_1}{s_2} - \frac{S_1}{S_2}|_1 > |\frac{s_1}{s_3} - \frac{S_1}{S_3}|_1$ after serving a class 1 packet.
Also, let
$|\frac{s_2}{s_1} - \frac{S_2}{S_1}|_2 > |\frac{s_2}{s_3} - \frac{S_2}{S_3}|_2$ after serving a class 1 packet.
However, if $|\frac{s_1}{s_2} - \frac{S_1}{S_2}|_1 < |\frac{s_2}{s_1} - \frac{S_2}{S_1}|_2$, then a class 1 packet is served; otherwise, a class 2 packet is served.

This strategy, when used exclusively, will eventually fulfill the requested throughput weights of all classes, especially if the window size over which the throughput is measured is long enough. This also requires that the input traffic satisfies certain conditions. The satisfaction of the throughput ratios is governed by the following theorem, whose proof is given in Appendix B:

Theorem 2: *Under steady state operation, the necessary and sufficient conditions for the throughput ratios between two classes, 1 and 2, to be satisfied, i.e.,* $\frac{s_1}{s_2} = \frac{S_1}{S_2}$ *depend on the offered load, and are as follows:*
Case 1: When $\rho_1 + \rho_2 \leq 1$, *then* λ_1/λ_2 *must equal* S_1/S_2
Case 2: When $\rho_1 + \rho_2 \geq 1$:

2a: if $\rho_1/\rho_2 > S_1/S_2$, *then* $\lambda_2 \geq S_2/(S_1\bar{b}_1 + S_2\bar{b}_2)$

2b: if $\rho_1/\rho_2 < S_1/S_2$, *then* $\lambda_1 \geq S_1/(S_1\bar{b}_1 + S_2\bar{b}_2)$

When the offered traffic is less than the system capacity, it may not be possible to satisfy the throughput ratios, as indicated by Case 1 above. All incoming packets are served, hence throughput differentiation is not possible unless the offered traffic satisfies the desired throughput ratios as in Theorem 2 (we assume a work conserving system). However, when the input traffic exceeds the server capacity, the system is considered overloaded and packets may have to be dropped if the overload condition persists for a time that is sufficiently long to fill the buffer at the router (Case 2 in the theorem). Even with traffic shaping at the ingress routers, such a case may arise, especially if route pinning in DiffServ domains is not used, and it might be possible to guarantee throughput differentiation subject to the above conditions.

For a system with N classes, where $N > 2$, the above theorem can still be applied to a target class while taking this class as Class 1, and the rest of the classes are aggregated in Class 2, and therefore λ_2, ρ_2, s_2 and S_2 will be equal to the sum of all such classes λ's, ρ's, s's and S's.

- **Controlling $\overline{q_i}/\overline{q_j}$: The queue control mechanism**

  In addition to controlling the throughput ratios, ratios of the mean queue sizes of different classes need to be also controlled in order to control the mean class delays proportionally in accordance with equation (3). As indicated earlier, our control of mean queue ratios is implemented by minimizing the difference between the instantaneous queues and the target queue ratios. This, in fact, is more effective in controlling the mean queue sizes than controlling them over a time window, as it reduces variations around the mean. The scheduler therefore controls the queue lengths, also using the min-max approach, but applied to the queue sizes. That is, a packet is served from that class which results in a maximum deviation of the target queue ratios, after service, which is minimal. This strategy, when exclusively used, will control the queue lengths (and hence delays) proportionally, and together with the throughput ratio control strategy, will control the mean delay ratio.

  There is one special case that deserves attention, since application of the above min-max approach may result in an anomalous behavior, namely, when one or more of the queues are empty. In this case, it can be observed that when at least one of the queues, e.g., q_1, is 0, then considering q_1/q_2 may give a different service decision than by considering q_2/q_1. The frequency of this anomalous behavior increases as the input traffic decreases, until, under sparse incoming traffic, the delay ratios are reversed from the ideal. We have dealt with this case by applying a simple, deterministic heuristic in such cases, i.e., in which we serve the class with the smallest delay weight W_i.

  There are certain constraints to the delay control using min-max optimality. The actual achievable class delay ratios are strongly dependant on actual class loads, and also on whether the system is saturated or not. These bounds on the achievable delay ratios are stated below, in Theorem 3 and Theorem 4 (for unsaturated and saturated systems, respectively)[6] under Poisson arrival processes. For a unsaturated system we have:

[6]These two bounds are also consistent with equation (4) in [25], with our result being the limit on the mean delay ratio when the Waiting Time Priority Scheduler is used, and when the dynamic priority control parameters for class 1 is much higher (respectively lower) than that for Class 2.

Theorem 3: *When for classes 1 and 2, $\sigma_1+\sigma_2 < 1$, and under Poisson arrivals and general service times, the achievable delay ratio between the two classes is such that:*

$$\frac{(1-\sigma_1-\sigma_2)(\overline{b}_0+\overline{b}_1(1-\sigma_1))}{\overline{b}_0+\overline{b}_2(1-\sigma_1)(1-\sigma_1-\sigma_2)} \leq \frac{\overline{w}_1}{\overline{w}_2} \leq \frac{\overline{b}_0+\overline{b}_2(1-\sigma_1)(1-\sigma_1-\sigma_2)}{(1-\sigma_1-\sigma_2)(\overline{b}_0+\overline{b}_1(1-\sigma_1))}$$

where $\overline{b}_0$ is the residual service time as seen by an arrival, and is given by

$$\overline{b}_0 = \sum_{i=1}^{2} \sigma_i \frac{\overline{b^2}_i}{2\overline{b}_i} \tag{4}$$

We further introduce the following bounds for a saturated system, i.e., $(\sigma_1+\sigma_2 = 1)$:

Theorem 4: *When for two classes 1 and 2, $\sigma_1+\sigma_2 = 1$, i.e., the system is saturated, and under Poisson arrivals and general service times, the achievable delay ratio between the two classes is such that:*

$$\frac{(\overline{b}_0-\sigma_1\overline{b}_1+\overline{b}_1)\sigma_2\overline{b}_1}{[(B-1)(1-\sigma_1)\overline{b}_1-(\overline{b}_0-\sigma_1\overline{b}_1)\sigma_1]\overline{b}_2} \leq \frac{\overline{w_1}}{\overline{w_2}} \leq \frac{[(B-1)(1-\sigma_1)\overline{b}_1-(\overline{b}_0-\sigma_1\overline{b}_1)\sigma_1]\overline{b}_2}{(\overline{b}_0-\sigma_1\overline{b}_1+\overline{b}_1)\sigma_2\overline{b}_1}$$

where $\overline{b}_0$ is the residual service time seen by an arrival and given by equation (4).

The proofs of Theorems 3 and 4 are given in Appendices C and D, respectively.

When more than two classes are involved, theorems 3 and 4 can still be applied by considering a target class as Class 1, and the rest of the classes are aggregated as Class 2. The mean delay values of the aggregated class are expressed using the conservation law in [23].

The above two mechanisms, namely throughput and delay control, are combined and used to determine which packet to serve. The selection of which mechanism to use, is determined by the system load, namely, heavy and light. If the load is heavy, i.e. the queue is full or almost full, we control the throughput ratio. However, when the load is light, we control the queue ratio. The definition of the load levels will be given in the next section.

The pseudocode for this algorithm is given in Figure 1. For all pseudocodes, and to aid in implementing the min-max strategy, we define a set of pairwise parameter ratio offsets for Class i as:

$$\Delta(x)_i = \frac{x_i}{x_{(i+1) \bmod N}} - \frac{X_i}{X_{(i+1) \bmod N}} \tag{5}$$

for $1 \leq i \leq N$, where the argument x can take the value s, w and q for the throughput, the mean delay, or the queue length, respectively, while X corresponds to x's target weight, namely, S, W

and $S \cdot W$, respectively.

```
System::onServerIdle {
    if SYSTEM-STATUS = LIGHT {
        For each class $i$, and for all $j$, $q_j \neq 0$
            compute $\Delta(q)_i|_{q_j=q_j-1}$
        find class $j$ for which
            $\max_{0 \leq i < N}\{\Delta(q)_i|_{q_j=q_j-1}\} \leq \max_{0 \leq i < N}\{\Delta(q)_i|_{q_l=q_l-1}\}$, for $j \neq l$
        Serve a packet from class $j$
    }
    else if SYSTEM-STATUS = HEAVY {
        For each class $i$, and for all $j$, $q_j \neq 0$
            compute $\Delta(s)_i|_{q_j=q_j-1}$
        find class $j$ for which
            $\max_{0 \leq i < N}\{\Delta(s)_i|_{q_j=q_j-1}\} \leq \max_{0 \leq i < N}\{\Delta(s)_i|_{q_l=q_l-1}\}$, for $j \neq l$
        Serve a packet from class $j$
    }
}
```

Figure 1: Packet Scheduler

III.3 Modes of Operation for the Packet Scheduler

Under steady state operation, and for a sufficiently large value of the window size, M, the throughput control mechanism, if used exclusively (i.e., if packet scheduling is controlled by using this mechanism only), will satisfy certain throughput ratios, subject to the input constraints described earlier (Theorem 2). Given that the throughput ratios are satisfied, if the queue control mechanism is then used, it will provide the delay ratios required, subject to Theorems 3 and 4. Since our work aims at combined delay and throughput differentiation, when feasible, the system should employ both mechanisms in a complementary manner in order to satisfy both requirements. We define two modes of operation for the packet scheduler, namely, the *light* and *heavy* load modes[7].

[7]We use the terms light and heavy loads here in an informal context in order to define two different modes of operation.

i. *Light load state*: We define the system to be lightly loaded when the total input traffic measured over a given time window is less than or equal to the server capacity. In this case, each class obviously receives all the throughput it has requested. Since the system is work-conserving, throughput control is neither needed nor possible under this condition. However, the delays of the classes may be controlled in this stage by controlling the queue sizes. Therefore, when the system is lightly loaded, the queue control mechanism is invoked in order to choose the class to be served, hence satisfying the queue ratios, and consequently the delay ratios.

ii. *Heavy load state*: When the input traffic to the server is greater than the server capacity, measured over a given time window, some packets may have to be discarded (depending on the duration of this state, and the buffer capacity). In this case, the throughput values of all classes are controlled by invoking the throughput control mechanism which will maintain the necessary throughput ratios. The queue control mechanism is not used in this case[8].

When the number of arrivals within a predefined time frame is less than a predefined packet threshold, the system is considered lightly loaded; otherwise it is heavily loaded. Load levels are selected in Part 1 of the pseudocode of Figure 2.

III.4 Mechanisms for Arrival - The Queue Manager

To complete the mechanism, we must decide on how to handle packet arrivals. Notice that arrivals of packets from a class change the queue ratios (and hence delay ratios) of associated classes. However, since the system is work-conserving, queue control (by dropping packets to adjust queue ratios) is not a feasible option when available buffer space exists. Recall that in this state, the packet scheduler works to control queue ratios proportionally. However when the buffer is full, two queuing decisions are possible: 1) *an arriving packet may be dropped*, or 2) *it may be accepted by discarding an already-queued packet from one of the other classes.* This decision must be made with a view to satisfying the delay ratios. The queue manager performs this function. In this case, we use an active queue management scheme to achieve combined proportional differentiation. Deviations of the queue ratios from the target ratios are computed for each of the above-mentioned decisions. That decision is chosen for which deviation of the queue ratios from the target ratios is

[8]As will be shown in Section III.4, a queue manager process will have to be used to determine which packets should be discarded upon packet arrivals to a full buffer.

optimal, using the min-max optimality criterion described earlier. Part 2 of Figure 2 presents the pseudocode.

```
System::onPktArrival {
    /*Part 1: choosing Packet Scheduler mode*/
    if number of packet arrivals in window $\leq$ packet-threshold
        SYSTEM-STATUS = LIGHT
    else
        SYSTEM-STATUS = HEAVY
    /*Part 2: Queue Manager*/
    /* $j$ = class of incoming packet */
    if buffer = NOT-FULL
        accept incoming packet
    else {
        for class $i$
            compute $\Delta(q)_i|_{q_j=q_j}$     /* discard arriving packet */
            for class $k$, such that $q_k > 0$
                compute $\Delta(q)_i|_{q_j=q_j+1,q_k=q_k-1}$    /* accept arriving packet, and discard class $k$ packet */
        find the queuing decision for which:
            $\max_{0\leq i<N}\{\Delta(q)_i|_{q_j=q_j}, \Delta(q)_i|_{q_j=q_j+1,q_k=q_k-1}, \forall k \neq j\} \leq$
                $\max_{0\leq i<N}\{\Delta(q)_i|_{q_l=q_l}, \Delta(q)_i|_{q_l=q_l+1,q_k=q_k-1},\ \forall k \neq l\}$
        Apply the chosen queueing decision.
    }
}
```

Figure 2: Queue Manager

IV Numerical Results

In this section we provide several numerical examples to show the effectiveness of our algorithm, using a simulation model written in the C++ programming language. We consider both single hop, and multihop environments. We show that our mechanism can achieve delay control in all scenarios. Further, during heavy load periods, throughput levels can be controlled by limiting the throughput ratios to their target ratios. During light load periods, however, throughput is equal to the offered load.

IV.1 Class Traiffc Models: The Modulated Markov Poisson (MMPP) process

The MMPP model, explained in [1], is a well known distribution commonly used to model voice and telephony traffic. In addition, it was recently used to model Internet traffic [20]. In this paper, MMPP is used as a general model that can encompass several types of traffic sources. An MMPP consists of a number of states, where the duration of each state, k, is exponentially distributed with rate μ_k. Transitions between states take place between adjacent states, and are memoryless and independent. When the process is in state k, packets are generated according to a Poisson process with rate λ_k. This process can be used to model a number of other processes, for example, an aggregation of N independent and identically distributed ON-OFF modulated Poisson sources. In this section, we assume three such ON-OFF modulated Poisson processes, each representing traffic generation from one of the three different classes, respectively. The rates of the Poisson process during the ON and OFF states determine the traffic intensity, and the mean duration of the ON and OFF states can control the burstiness of the traffic generated by each packet class. Each stream is controlled independently. Figure 3 shows the MMPP state diagram modeling a single packet class.

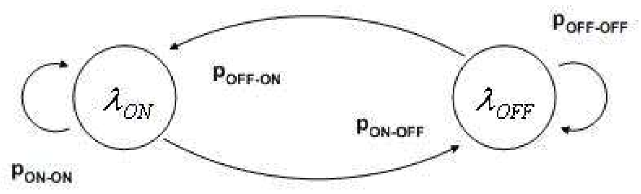


Figure 3: State diagram for a Markov Modulated Poisson Process

Consider a single server, corresponding to the output port of a router, which is fed by three such packet streams. In simulation experiments, all packets from all streams are assumed to require a transmission time, which is identically and independently distributed according to an exponential distribution with a unity mean. The buffer at the server can accommodate a maximum of 100 packets, and the sliding window size (M), which is used for throughput measurement, is 100 time

Table 2: Heavy load: throughput and delay ratios

	Throughput ratios			Delay ratios		
	s_0/s_1	s_1/s_2	s_2/s_0	$\overline{w_0}/\overline{w_1}$	$\overline{w_1}/\overline{w_2}$	$\overline{w_2}/\overline{w_0}$
Target ratios	1.0	2.0	0.5	2.0	1.0	0.5
Measured ratios	1.003	1.96	0.507	2.048	0.968	0.504

units. The mean ON and OFF periods are exponentially distributed, and their mean values are used to control the burstiness of the incoming traffic. In addition, a hysteresis process was used to determine the system status, which is used in the process in the packet scheduler shown in Figure 1. An upper threshold level of 120 packet arrivals within 100 time units was used to switch the system status from LIGHT to HEAVY if this threshold is exceeded. On the other hand, a lower threshold of 80 packet arrivals within a window of 100 time units was used to switch the system status from HEAVY to LIGHT if the number of arrivals falls below this level. Each simulation experiment was run for 300,000 time units.

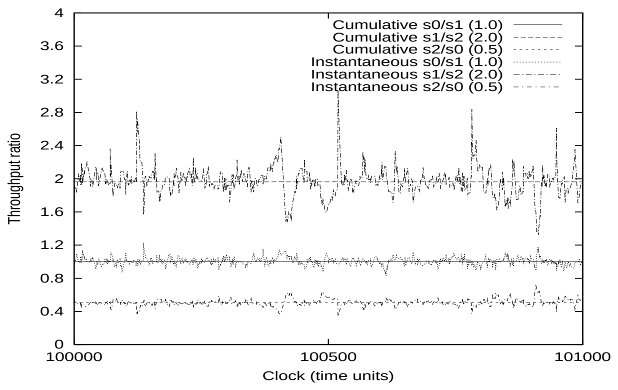
The results presented in all following sections, except Section IV.6 (which presents results for a Pareto modulated source) use an MMPP source.

IV.2 Results for heavy, light and mixed traffic

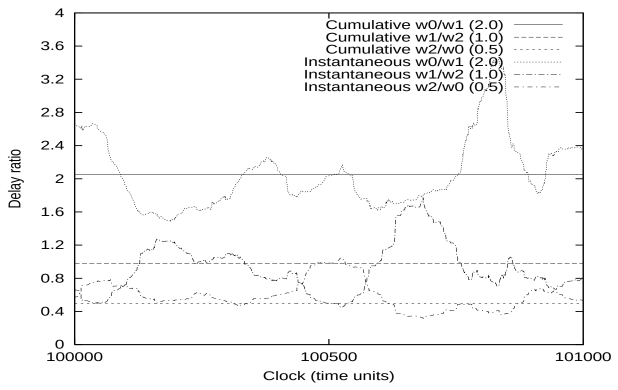
1) Heavy Traffic:

The first scenario we consider is the heavy traffic case, where the system is almost always in the heavily loaded state. This is the case in which our scheme should be able to achieve full control over the throughput and mean delay ratios. The traffic from the three streams is adjusted such that their offered loads are 1, 1 and 2 Erlangs, respectively, and is Poisson, i.e., the OFF period duration is zero. The throughput weights are 2, 2 and 1, while the mean delay weights are 2, 1 and 1, respectively. In Table 2 we show both the target and achieved ratios for throughput and delay, measured over the entire simulation run.

Since the offered load satisfies the condition for Case 2 in Theorem 2, then it is possible to control the throughput ratios. This is shown in Table 2, where the measured throughput ratios are very close to the target ratio, with a maximum error of 2%. The achieved delay ratios are also very close to the target values, with a maximum error of 3%. In Figure 4(a) we plot the instantaneous and cumulative throughput ratios over a time window with a duration of 1000 packet transmission times. The instantaneous ratios are measured every one time window. The cumulative measures at time t correspond to ratios measured over the time interval $(0, t]$. The instantaneous and cumulative mean delay ratios, measured over the same interval, and using the same sliding window, are also shown in Figure 4(b). In the figures, the target values are shown between parentheses in the legend.



(a)



(b)

Figure 4: Heavy traffic: (a) Throughput ratios; (b) Delay ratios

Note that, for both delay and throughput, the cumulative values are constant and are very close to the target, and the maximum deviations of instantaneous ratios from the target are limited, and are usually small[9]. Such fluctuations occur due to the lack of packets from one or more classes at a particular instant. However, these fluctuations are corrected as soon as packets from the under-served class arrive. For example, in Figure 4(b), and in the time interval [10300,10400], there is a lack of Class 2 packets. Therefore, packets are served from classes 0 and 1, hence resulting in an increasing s_1/s_2 ratio, and a decreasing s_2/s_0 ratio. The s_0/s_1 ratio, however, is practically unaffected. Towards the end of this period, packets from Class 2 arrive, and are served, hence resulting in correcting these ratios. It is to be noticed that because of the relation in equation (3), an increase in s_1/s_2 can result in a corresponding decrease in the $\overline{w_1}/\overline{w_2}$ ratio. The slope of the delay instantaneous ratios graph is small most of the time.

2) Light Traffic

The second scenario is with light traffic, where the system is almost always lightly loaded. The three classes generate Poisson traffic at rates of 0.1, 0.3 and 0.2 Erlangs, respectively, for a total of 0.6 Erlangs. In this case, the queue size never exceeds 20 packets, and no packet losses are ever

[9]This is in comparison to fluctuations produced by the WTP and BPR algorithms in [15].

encountered. The individual class throughput values are equal to the offered load, and therefore cannot be controlled. However, setting the target mean delay weights to 1, 1 and 1.25, our scheme was able to control the mean delay ratios, as shown in Table 3. The error in the mean delay ratios is less than 3%.

Table 3: Light load: delay ratios

Delay Ratio	Target	Measured
$\overline{w_0}/\overline{w_1}$	1.0	0.976
$\overline{w_1}/\overline{w_2}$	0.8	0.812
$\overline{w_2}/\overline{w_0}$	1.25	1.261

The graphs of the cumulative mean delay ratios, during a 1000 time unit window, and measured in a manner similar to the above, were almost flat and close to the target, as shown in Figure 5. We note that the instantaneous delay ratios graph has a much steeper slope than in the heavy load case (Figure 4(b)). This is due to the fact that the idle-system case, when queues are momentarily empty, occurs much more frequently when the system is lightly loaded.

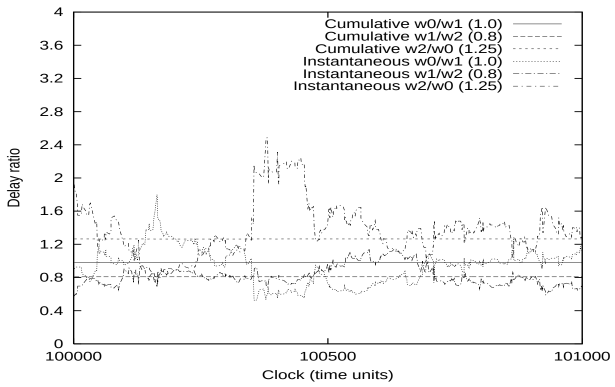


Figure 5: Light Load: Delay ratios

3) Mixed Load:

The third scenario shows the effectiveness of our scheme in mixed loads, when the system alternates between periods of heavy load and light load states, which are generated using the MMPP process. The three classes offer average loads of 0.4, 0.4 and 0.85 Erlang during the heavy load

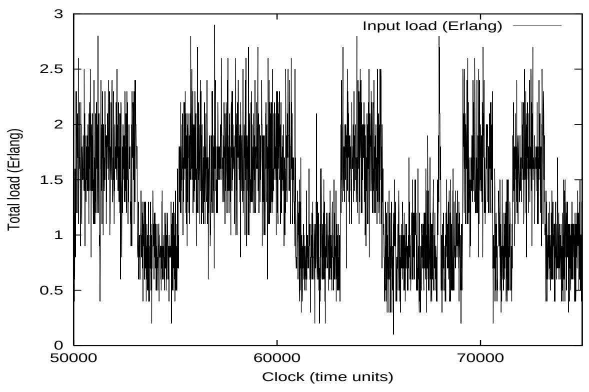


Figure 6: Mixed load: Snapshot of input load

periods, and average loads of 0.2, 0.2 and 0.425 Erlang during the light load periods, respectively. The durations of the heavy and light load periods were exponentially distributed with a mean of 2,000 time units. We set the target delay weights to 1.0, 1.0 and 1.25, which are to be enforced over the entire simulation run. The target throughput weights are set to 1.2, 1.0 and 1.4, and are enforceable under periods of heavy load only. Under light load conditions, throughput will be equal to the offered load. Figure 6 shows a snapshot of the input load in the period of 50,000 to 75,000 time units.

In Table 4, we show the average delay ratios for the heavy and light load periods, as well as the overall delay ratios. The overall averages are reasonably close to the target ratios, and the light load period ratios also follow. Table 5 shows similar results for throughput. Note that very precise

Table 4: Bursty load: delay ratios

Delay Ratio	Target	Heavy period Avg.	Light period Avg.	Overall Avg.
$\overline{w_0}/\overline{w_1}$	1.0	0.78	0.93	0.84
$\overline{w_1}/\overline{w_2}$	0.8	0.73	0.77	0.83
$\overline{w_2}/\overline{w_0}$	1.25	1.74	1.38	1.42

control is achieved over the ratios in the heavy load phase. During the light load period, however, no throughput control is exercised, and all incoming traffic is served. Throughput ratios in this phase are the same as those for the incoming traffic. It is to be noted that the longer and more frequent the light load periods, the more deviation from the target the overall throughput ratio will exhibit.

Table 5: Bursty load: throughput ratios

Delay Ratio	Target (for heavy load)	Heavy period Avg.	Light period Avg.	Overall Avg.
s_0/s_1	1.2	1.19	0.99	1.11
s_1/s_2	0.71	0.71	0.47	0.59
s_2/s_0	1.16	1.17	2.1	1.50

It is important to point out that, for the lightly loaded system, our scheme reduces to a new implementation of *Proportional Delay Differentiation* which does not require actual delay measurements.

IV.3 Effect of offered load distribution

This section demonstrates our scheme's ability to handle variations in distribution of the load. Unlike the limitations of some schemes[10], this scheme works well under varying load conditions. To demonstrate this ability, the following scenarios are presented:

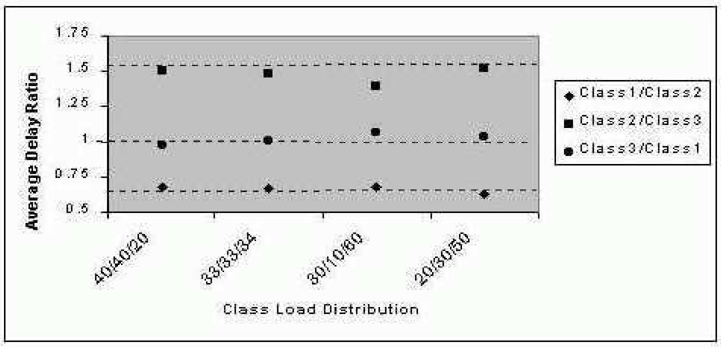


Figure 7: Unsaturated system: Delay ratios as functions of load distribution

- *Unsaturated system: Effect of class load distribution on delay*

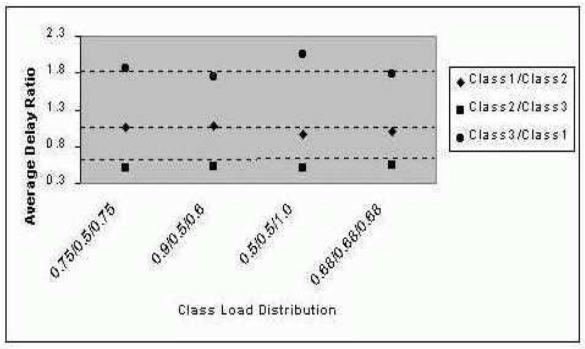
  Figure 7 shows the cumulative achieved delay ratios when the system is unsaturated, and

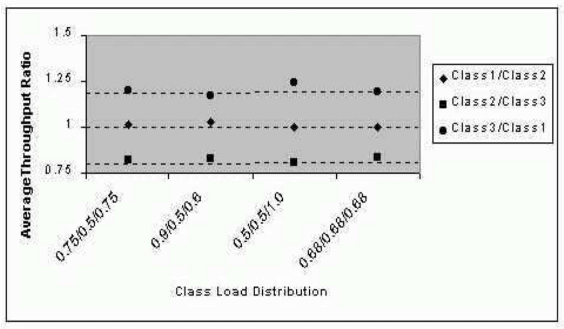
[10]Reference [13] presents results for the BPR proportional delay scheduler, in which uneven loads are shown to strongly and adversely affect the achieved ratios.

compares them to the target ratios (shown by dotted lines). All data is for a total load of 96% (unsaturated) and with delay weights of 1.0, 1.6, and 1.0. Throughput weights are irrelevant, since throughputs cannot be controlled in an unsaturated system. Different percentages of the total load among the different classes were considered, namely, 40/40/20, 33/33/34, 30/10/60, and 20/30/50. It is seen that delay correction is uniformly accurate, regardless of the distribution of the input traffic. The maximum deviation from the target is in the case when the input loads of the classes are distributed as 30/10/60, i.e., when the class loads show a large difference. Such a deviation from the ideal is, however, still limited to less than 7%.

- *Saturated system: Effect of class load distribution on delay and throughput*

  For this scenario, it is not very useful to take the same approach as in the unsaturated case, as explained below. Consider the following scenario: throughput weights of 1.0, 1.0 and 1.2 for the three classes. These numbers are chosen to generate fairly small ratios. Theorem 2 places restrictions on the achievable throughput ratios depending on the input load, and therefore an arbitrary combination of input loads cannot be expected to satisfy the throughput weights. Given these restrictions, it is not useful to take the approach taken for the unsaturated case, i.e., of choosing class input loads as arbitrary percentages of a total load. Instead, different class loads are chosen so as to keep the total input load as 2 Erlang, while maintaining each class load as at least 0.5 Erlang. Further, delay weights are taken as 1.0, 1.0, and 1.8.





(a) (b)

Figure 8: Saturated System (a) Delay ratios as functions of load distribution in Erlangs; and (b) Throughput ratios as functions of load distribution in Erlangs

The following 4 scenarios have been chosen to demonstrate combined delay and throughput differentiation:

- Class 1: 0.75 Erlang; Class 2: 0.5 Erlang; Class 3: 0.75 Erlang
- Class 1: 0.9 Erlang; Class 2: 0.5 Erlang; Class 3: 0.6 Erlang
- Class 1: 0.5 Erlang; Class 2: 0.5 Erlang; Class 3: 1.0 Erlang
- Class 1: 0.667 Erlang; Class 2: 0.667 Erlang; Class 3: 0.667 Erlang

Delay and throughput differentiation results are presented in Figure 8(a) and Figure 8(b) respectively. The graphs show that the combined delay and throughput differentiation is uniformly achievable in all these cases. As before, the maximum deviation from the ideal is in the case where the class loads are 0.5, 0.5, and 1.0 Erlangs, respectively, which is the case with the most diverse loads. Note that if both weight sets are taken together, Class 3 requires both high delay and high throughput. This, however, is satisfied by our scheme.

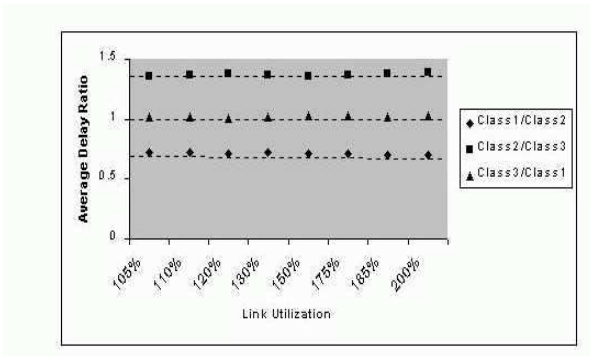
IV.4 Effect of varying the offered load

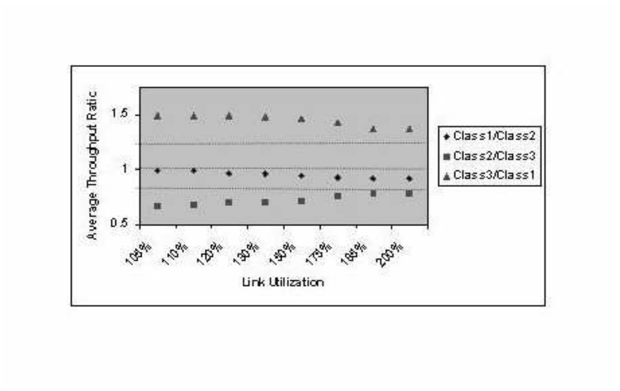
This section presents results for delay and throughput differentiation with varying loads conditions. Typically[11], delay ratios get closer to the target ratios as the load increases and the system saturates, with moderate loads (75%-90%) unable to offer close correction.

Similar to the previous section, two cases, saturated and unsaturated, are considered. However, since in the unsaturated case only the delay is controllable, we present results for the saturated case only, where our algorithm can exercise both delay and throughput differentiation. We have experimented with several load levels exceeding 100%, and ranging between 105% and 200%. We have also used three classes, and set the delay weights to 1.0, 1.4, and 1.0, while the throughput weights were set to 1.0, 1.0 and 1.2. In all cases, the individual class loads are distributed as 1:1:1.5 of the total system load, respectively. The ON and OFF periods have average durations of 10 and 0.1 time units, respectively. Figure 9 shows the results of the delay and throughput differentiation in parts (a) and (b), respectively.

The graphs show that the correction in delay and throughput improves as the input load increases. It is to be noted that the delay remains well corrected over all loads. Throughput ratios are observed to be better corrected as the system load increases.

[11]Proportional delay differentiation schemes presented in [13] show up to 30% variation from ideal, under moderate loads of 75% utilization.





(a) (b)

Figure 9: Saturated System (a)Delay ratios as functions of utilization and (b)Throughput ratios as functions of utilization

IV.5 Validating the delay and throughput bounds

This section demonstrates the validity of the delay and throughput bounds stated in Theorems 2, 3 and 4 in Section III. Each of the following sections considers a scenario, derives related bounds, and shows results that demonstrate the applicability of these bounds.

i. *Theorem 2: Bounds for throughput ratios*

   Theorem 2 states that, in case the system is unsaturated, the throughput levels are uncontrollable. For saturated systems, however, the achievable throughput ratios are as stated. Consider a saturated system serving 2 classes having respective throughput weights of 5 and 1, and consider two scenarios. In the first scenario, the two classes have input loads of 0.7 Erlang and 0.6 Erlang, respectively. Then, Theorem 2 states that, in order for these throughput ratios to be achieved, Class 1 must have a minimum arrival rate of 0.833 Erlang, while Class 2 would retain its arrival rate of 0.6 Erlang. Table 6 shows that, as expected, Scenario 1, which has an input rate of 0.7 Erlang for Class 1, is unable to satisfy the throughput ratios. However, in the second scenario, the input load of Class 1 is increased to 0.9 Erlang, while the input load of Class 2 is unchanged. In this case, the desired throughput ratio can be achieved as shown in the table.

ii. *Theorem 3: Bounds on delay ratios in an unsaturated system*

   If the system is unsaturated, i.e., if $\sigma_1 + \sigma_2 < 1$, then the achievable delay ratios are as stated in Theorem 3. Consider two classes with input loads of 0.2 and 0.3 Erlangs, respectively.

Table 6: Validating bounds: Theorem 2

	Scenario 1	Scenario 2
Class 1 input load	0.7	0.9
Class 2 input load	0.6	0.6
Achieved s_1/s_2	2.29	5.39

Since the throughput levels are uncontrollable in an unsaturated system, the throughput levels of the two classes are also 0.2 and 0.3 Erlangs. In this case, according to Theorem 3, the achievable delay is bounded by $0.722 \leq (\overline{w_1}/\overline{w_2}) \leq 1.39$. We consider two scenarios here. In the first scenario, the delay weights of the two classes are 1.2 and 1.0 respectively, and the delays should satisfy these weights, such that $\overline{w_1}/\overline{w_2} = 1.2$. However, in the second scenario, the delay weights are increased to 2.0 and 1.0, and according to the bounds, the average delay figures can no longer satisfy these weights, i.e., $\overline{w_1}/\overline{w_2}$ cannot be 2.0. Table 7 shows these two scenarios, and verifies these conclusions.

Table 7: Validating bounds: Theorem 3

	Scenario 1	Scenario 2
Class 1 delay weight	1.2	2.0
Class 2 delay weight	1.0	1.0
Class 1 measured $\overline{w_1}$	2.658	2.86
Class 2 measured $\overline{w_2}$	2.226	2.13
Achieved $\overline{w_1}/\overline{w_2}$	1.194	1.342

iii. *Theorem 4: Bounds on delay ratios in a saturated system*

Consider two classes with throughput levels of 0.6 and 0.4 Erlang, respectively. Then, since the system is saturated, i.e., $\sigma_1 + \sigma_2 = 1$, the bounds on the achievable delay ratios are $0.014 \leq (\overline{w_1}/\overline{w_2}) \leq 70.3$. To show the applicability of these bounds, also two cases are considered. If the delay weights are taken as 50.0 and 1.0 (Scenario 1), they are within the bounds and the delay ratio can be satisfied. If the weights are 90, 1 (Scenario 2), the delay ratio will no longer be satisfied. Table 8 shows the obtained results for both scenarios.

Table 8: Validating bounds: Theorem 4

	Scenario 1	Scenario 2
Class 1 delay weight	50.0	90.0
Class 2 delay weight	1.0	1.0
Class 1 measured $\overline{w_1}$	146.311	140.890
Class 2 measured $\overline{w_2}$	3.325	2.056
Achieved $\overline{w_1}/\overline{w_2}$	44.003	68.526

It must be noted that this condition of precise saturation ($\sigma_1 + \sigma_2 = 1$) is difficult to achieve, due the bursty nature of arriving traffic and dropped packets.

IV.6 Results with Pareto Modulated Input Traffic

It was shown in [24] that Ethernet traffic exhibits self-similar characteristics. It was later shown by many researchers, e.g. [10], that Internet traffic, including web traffic exhibits the same characteristics. This nature of the Internet traffic is usually modeled using a heavy tailed distribution [18, 9]. The Pareto distribution is usually used for modeling burst sizes in web transfers, where the probability that a given burst size is larger than x bytes is $(1/x)^b$, where b is the shape parameter of the Pareto distribution, and $x \geq a$.

For testing the effectiveness of our scheme under such heavy tailed traffic, the ON and OFF periods (which are analogous to burst and idle lengths) are modeled using Pareto distributions. The interarrival times of the input traffic during the ON and OFF periods are still modeled as Poisson, as in the previous sections. The Pareto shape parameter, b, was taken as 5, and a was taken as 0.

Scenarios very similar to the ones in Section IV.2 (for light, heavy, and mixed loads) were employed. Under heavy load, the traffic from the three classes were 1, 1 and 2 Erlangs, respectively. The throughput weights were 2, 2, and 1, while the mean delay weights were 2, 1 and 1, respectively. The mean ON time is 100 time units, and the mean OFF time is 1 time unit. Table 9 shows both the target and achieved ratios for througput and delay. The achieved values are very close to the target values, and similar in accuracy to the results obtained for MMPP traffic.

The second scenario is under light traffic, where the system is almost always lightly loaded. The

Table 9: Heavy load behavior under Pareto distributed burst lengths: throughput and delay ratios

	Throughput ratios			Delay ratios		
	s_0/s_1	s_1/s_2	s_2/s_0	$\overline{w_0}/\overline{w_1}$	$\overline{w_1}/\overline{w_2}$	$\overline{w_2}/\overline{w_0}$
Target ratios	1.0	2.0	0.5	2.0	1.0	0.5
Measured ratios	1.004	1.94	0.513	2.033	0.992	0.495

three streams generate Poisson traffic at rates of 0.1, 0.3 and 0.2 Erlang, for a total of 0.6 Erlang. Mean ON times are 10 time units, and mean OFF times are 120 time units. The queue size does not exceed a maximum of 29 packets, and no packets are dropped. The individual stream throughput levels are equal to the offered load, and therefore cannot be controlled. However, setting the target mean delay weights to 1, 1 and 1.25, the scheme was able to control the mean delay ratios with a high degree of accuracy, as shown in Table 10.

Table 10: Light load under Pareto distributed burst lengths: delay ratios

Delay Ratio	Target	Measured
$\overline{w_0}/\overline{w_1}$	1.0	1.008
$\overline{w_1}/\overline{w_2}$	0.8	0.784
$\overline{w_2}/\overline{w_0}$	1.25	1.264

The final scenario is under mixed load, where the system alternates between periods of heavy and light load. The three classes offer average loads of 0.4, 0.4 and 0.85 Erlangs during the heavy load periods, and average loads of 0.2, 0.2 and 0.425 Erlangs under light load periods, respectively. The ON and OFF periods were generated with a Pareto distribution and have a mean of 2000 time units each. The target delay weights are set as 1.0, 1.0 and 1.25, which will be achievable over the entire simulation run. The target throughput weights are set to 1.2, 1.0 and 1.4, and can be enforced only during periods of heavy load. Table 11 shows the average delay ratios for the heavy and light load periods, as well as the overall delay ratios. The overall averages are reasonably close to the target ratios, as do the ratios during light load periods. The average ratios for the heavy period show somewhat larger deviations from the target.

Table 11: Bursty load with Pareto distributed burst lengths: delay ratios

Delay Ratio	Target	Heavy period Avg.	Light period Avg.	Overall Avg.
$\overline{w_0}/\overline{w_1}$	1.0	0.76	0.93	0.82
$\overline{w_1}/\overline{w_2}$	0.8	0.74	0.78	0.88
$\overline{w_2}/\overline{w_0}$	1.25	1.78	1.38	1.38

Table 12 shows the results for throughput. In the heavy load phase, throughput ratios are satisfied very accurately. During the light load duration, no throughput control is exercised, and all incoming traffic is served. Throughput ratios in this phase are the same as those for the incoming traffic.

Table 12: Bursty load with Pareto distributed burst lengths: throughput ratios

Delay Ratio	Target (for heavy load)	Heavy period Avg.	Light period Avg.	Overall Avg.
s_0/s_1	1.2	1.19	0.99	1.11
s_1/s_2	0.71	0.71	0.47	0.59
s_2/s_0	1.16	1.17	2.1	1.52

The instantaneous throughput and delay ratios under the above load conditions exhibit a behavior that is similar to that obtained with MMPP trarffic, and are not shown here due to lack of space. However, the results presented in this section show that the scheme is equally effective under Pareto distributed burst lengths.

IV.7 A Network Scenario: Proportional Differentiation over Multiple hops

The examples presented in this paper have dealt with our proposed scheme in a single server environment. The set of proposed strategies are meant to implement specific per hop behaviors (combined delay and throughput control) at a router. However, for an end-to-end connection to perceive the benefits of this scheme, this behavior will need to translate to effective performance on an end-to-end basis. This section considers the use of this scheme on an end-to-end network.

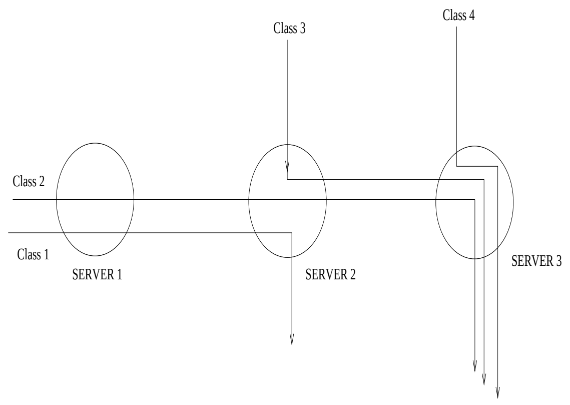


Figure 10: An example Multihop Network

Table 13: Throughput and Delay weights for the network

		Thruput Weight	Delay Weight
Server 1	Class 1	1.25	1.0
	Class 2	1.0	1.6
	Class 3	1.0	1.0
	Class 4	1.0	1.0
Server 2	Class 1	1.0	1.0
	Class 2	1.0	1.4
	Class 3	2.0	1.0
	Class 4	1.0	1.0
Server 3	Class 1	2.0	1.0
	Class 2	1.0	1.0
	Class 3	2.0	1.0
	Class 4	2.0	1.0

Table 14: Throughput and Average Delay values for the network

		Throughput	Avg. Delay
Server 1	Class 1	$s_1 = 0.558$	$\overline{w_1} = 77.24$
	Class 2	$s_2 = 0.439$	$\overline{w_2} = 122.62$
	Class 3	$s_3 =$ -	$\overline{w_3} =$ -
	Class 4	$s_4 =$ -	$\overline{w_4} =$ -
Server 2	Class 1	$s_1 = 0.268$	$\overline{w_1} = 111.0$
	Class 2	$s_2 = 0.255$	$\overline{w_2} = 142.38$
	Class 3	$s_3 = 0.474$	$\overline{w_3} = 85.82$
	Class 4	$s_4 =$ -	$\overline{w_4} =$ -
Server 3	Class 1	$s_1 =$ -	$\overline{w_1} =$ -
	Class 2	$s_2 = 0.186$	$\overline{w_2} = 114.38$
	Class 3	$s_3 = 0.342$	$\overline{w_3} = 112.52$
	Class 4	$s_4 = 0.469$	$\overline{w_4} = 98.74$

Table 15: Throughput and Delay ratios for the network

	Thrput Ratio	Target	Measured	Delay Ratio	Target	Measured
Server 1	s_1/s_2	1.25	1.27	$\overline{w_1}/\overline{w_2}$	0.625	0.629
	s_2/s_3	1.0	-	$\overline{w_2}/\overline{w_3}$	1.6	-
	s_3/s_4	1.0	-	$\overline{w_3}/\overline{w_4}$	1.0	-
	s_4/s_1	0.8	-	$\overline{w_4}/\overline{w_1}$	1.0	-
Server 2	s_1/s_2	1.0	1.04	$\overline{w_1}/\overline{w_2}$	0.714	0.779
	s_2/s_3	0.5	0.54	$\overline{w_2}/\overline{w_3}$	1.4	1.65
	s_3/s_4	2.0	-	$\overline{w_3}/\overline{w_4}$	1.0	-
	s_4/s_1	1.0	-	$\overline{w_4}/\overline{w_1}$	1.0	-
Server 3	s_1/s_2	2.0	-	$\overline{w_1}/\overline{w_2}$	1.0	-
	s_2/s_3	0.5	0.55	$\overline{w_2}/\overline{w_3}$	1.0	1.016
	s_3/s_4	1.0	0.73	$\overline{w_3}/\overline{w_4}$	1.0	1.139
	s_4/s_1	1.0	-	$\overline{w_4}/\overline{w_1}$	1.0	-

Figure 10 shows a simple end-to-end network with 3 servers, each of which has the same capability of the single server described in the previous sections. The class throughput weights and delay weights chosen by each server are shown in Table 13. The classes offer heterogeneous loads, with Class 1 offering an aggregate load of 0.7 Erlang, Class 2 offering 0.8 Erlang, Class 3 offering 0.9 Erlang, and Class 4 offering 0.7 Erlang. Table 14 shows the actual delay and throughput values after the traffic exits each of the servers. Note that, out of the 0.439 Erlang of Class 2 traffic that leaves Server 1 (and enters Server 2), only 0.255 Erlang leaves Server 2. This is because of the queue manager at Server 2, which drops arriving packets when faced with a full buffer. When the system is saturated, packets are dropped in order to maintain the desired queue ratios. Similar observations can be made regarding Class 2 and Class 3 traffic leaving Server 2 and entering Server 3. Table 15 shows the delay and throughput ratios achieved, for the entire network. The achieved delay and throughput ratios are very close to the target ones.

To see the efficacy of the scheme for an end-to-end topology, consider that a Source *Src*1 (at Server 1) is sending a stream of packets to a destination *Dst*1 (through servers 1, 2 and 3). The source requests a delay weight of 1.2, and sends Class 1 traffic. By setting the weight of Class 1 to 1.2 at every server, it is obvious that this can be achieved. In the case of throughput, it is sufficient to set the desired throughput weight at Server 3[12]. Regardless of the path taken by Src1's traffic,

[12]Setting the throughput weights at all servers is necessary to determine bandwidth sharing during heavy load periods.

Dst 1 will receive its weighted share of inputs into Server 2, provided that the input traffic into Server 3 is adequate to satisfy the throughput weights.

Based on the above discussion, it can be observed that the scheme can implement different PHB's at different servers. Thus end-to-end delay differentiation is achievable, while the throughput differentiation depends on the traffic and the throughput weights at the different servers.

V Conclusions

This paper investigated combined proportional differentiation between multiple performance metrics. It first showed that combined proportional differentiation between loss and mean delay cannot be achieved unless the actual loss values were taken into account. Since loss ratios are very small, and are very difficult to measure accurately, we have therefore presented an algorithm for combined proportional differentiation between mean delay and throughput. The algorithm is based on an implementation that enforces the throughput ratios, and then enforces the mean queue ratios. Based on the satisfaction of Little's result, the mean delay ratio will then be achieved. The paper presented several numerical examples which show the effectiveness of the algorithm, The algorithm was verified using traffic from sources with MMPP packet generation processes, and also with Pareto distributed ON and OFF periods. An example of a network environment was also shown. When throughput is uncontrollable, our algorithm can be regarded as a new, and a simple implementation of proportional delay differentiation, which does not require delay measurements. However, under heavy load, the algorithm is able to control both throughput and mean delay ratios properly.

Appendix A

Theorem 1: *It is not possible to achieve combined proportional differentiation in the delay and loss metrics, independent of actual values of packet loss ratios.*

Proof: Let l_i be the fraction of lost packets for class i, and λ_i be the mean arrival rate for class i. Then the throughput for class i, s_i can be expressed as $\lambda_i(1-l_i)$. Little's Result states that $\overline{q_i} = s_i\overline{w_i}$, which yields

$$\frac{\overline{q_i}}{\overline{q_j}} = \frac{s_i}{s_j} \cdot \frac{\overline{w_i}}{\overline{w_j}} = \frac{\lambda_i}{\lambda_j} \cdot \frac{(1-l_i)}{(1-l_j)} \cdot \frac{\overline{w_i}}{\overline{w_j}}$$

Let the target proportions be $\overline{w_i}/\overline{w_j} = W_i/W_j$ and $l_i/l_j = L_i/L_j$. Then the ratio $\overline{q_i}/\overline{q_j}$ must be expressible in terms of these proportions only. We prove the infeasibility of this using contradiction.

Assume these proportions can be simultaneously satisfied, then

$$\frac{\overline{q_i}}{\overline{q_j}} = \frac{\lambda_i}{\lambda_j} \cdot \frac{(1 - l_j \cdot L_i/L_j)}{1 - l_j} \cdot \frac{W_i}{W_j} \tag{6}$$

However, equation (6) depends on l_j in addition to L_i and L_j. This contradicts the assumption, proving that combined proportional delay and loss differentiation is only possible if the actual loss ratios are taken into account. □

Appendix B

Theorem 2: *The necessary and sufficient conditions for the throughput ratios between two classes, 1 and 2, to be satisfied depend on the offered load, and are as follows:*
Case 1: When $\rho_1 + \rho_2 \leq 1$, *then* λ_1/λ_2 *must equal* S_1/S_2
Case 2: When $\rho_1 + \rho_2 \geq 1$:

2a: if $\rho_1/\rho_2 > S_1/S_2$, *then* $\lambda_2 \geq S_2/(S_1\overline{b}_1 + S_2\overline{b}_2)$

2b: if $\rho_1/\rho_2 < S_1/S_2$, *then* $\lambda_1 \geq S_1/(S_1\overline{b}_1 + S_2\overline{b}_2)$

Proof: Assume that the window size is very large such that the class throughput measured over the window is the actual class throughput. We first prove sufficiency by direct proof.
Case 1: Since the system is work-conserving, $\rho_1 + \rho_2 \leq 1$, implies $s_1 = \lambda_1$, $s_2 = \lambda_2$. Then, it must be that $s_1/s_2 = \lambda_1/\lambda_2 = S_1/S_2$.
Case 2: $\rho_1 + \rho_2 \geq 1$ implies that some traffic will be discarded. Further,
Case 2a: $\lambda_1/\lambda_2 > S_1/S_2$ means that $s_1 < \lambda_1$, and that at least some traffic from class 1 must be dropped. In the worst case, all Class 2 traffic will be carried, i.e., $s_2 \leq \lambda_2$, and

$$s_2 = s_1 \cdot \frac{S_2}{S_1} \leq \lambda_2 \tag{7}$$

Since the system is saturated, then

$$s_1\overline{b}_1 + s_2\overline{b}_2 = 1 \tag{8}$$

Combining equations (7) and (8), we have

$$\lambda_2 \geq \frac{S_2}{S_1\overline{b}_1 + S_2\overline{b}_2}$$

which is the required sufficient condition.

Case 2b: The proof is similar to *Case 2a*.

To prove the converse, i.e., that this is the the necessary condition, we use contradiction.

Case 1 is obvious.

Case 2: Let $\rho_1 + \rho_2 \geq 1$. Further,

Case 2a: It is given that $\lambda_1/\lambda_2 > S_1/S_2$. Assume that the necessary condition is not satisfied, i.e., $\lambda_2 < S_2/(S_1\overline{b}_1 + S_2\overline{b}_2)$. Then, $\lambda_1/\lambda_2 > S_1/S_2$ means that a fraction, X_1, of the Class 1 offered load must be discarded until $(\lambda_1 - X_1)/\lambda_2 = S_1/S_2$, where $0 < X_1 < \lambda_1$. Then, both $(\lambda_1 - X_1)$ and λ_2 can be reduced proportionally with the S_1/S_2 ratio satisfied. This means that

$$\lambda_1 - X_1 = \lambda_2 \cdot \frac{S_1}{S_2}. \tag{9}$$

By the assumption of $\lambda_2 < S_2/(S_1\overline{b}_1 + S_2\overline{b}_2)$, equation (9) yields

$$\lambda_1 - X_1 < \frac{S_1}{S_1\overline{b}_1 + S_2\overline{b}_2}$$

The reduction in λ_1 by X_1 should affect the throughputs as well, to give $s_1 \leq \lambda_1 - X_1$, and $s_2 \leq \lambda_2$. This yields,

$$\overline{b}_1 s_1 + \overline{b}_2 s_2 \leq (\lambda_1 - X_1)\overline{b}_1 + \lambda_2\overline{b}_2 < \frac{S_1\overline{b}_1}{\overline{b}_1 S_1 + \overline{b}_2 S_2} + \frac{S_2\overline{b}_2}{\overline{b}_1 S_1 + \overline{b}_2 S_2} = 1$$

Since the system is work conserving, it must be $s_1 + s_2 = 1$. Therefore, this last equation is a contradiction.

Case 2b: Proof is similar to Case 2a. □

Appendix C

Theorem 3: *When for classes 1 and 2, $\sigma_1 + \sigma_2 < 1$, and under Poisson arrivals and general service times, the achievable delay ratio is such that:*

$$(1 - \sigma_1 - \sigma_2) \cdot \frac{\overline{b}_0 + \overline{b}_1(1 - \sigma_1)}{\overline{b}_0 + \overline{b}_2(1 - \sigma_1)(1 - \sigma_1 - \sigma_2)} \leq \frac{\overline{w}_1}{\overline{w}_2} \leq \frac{1}{1 - \sigma_1 - \sigma_2} \cdot \frac{\overline{b}_0 + \overline{b}_2(1 - \sigma_1)(1 - \sigma_1 - \sigma_2)}{\overline{b}_0 + \overline{b}_1(1 - \sigma_1)}$$

where $\overline{b}_0$ is the residual service time as seen by an arrival, and is given by

$$\overline{b}_0 = \sum_{i=1}^{2} \sigma_i \frac{\overline{b_i^{\,2}}}{2\overline{b}_i} \tag{10}$$

Proof: We assume that the buffer will never overflow. The minimum value of $\overline{w}_1/\overline{w}_2$ will be achieved when Class 1 has a non-preemptive, higher priority than Class 2. Stating the result given in [23] regarding the mean waiting time in the queue (excluding service) in classes 1 and 2, t_1 and t_2, we have

$$\begin{aligned} \bar{t}_1 &= \frac{\bar{b}_0}{(1-\sigma_1)} \qquad\qquad \text{and} \\ \bar{t}_2 &= \frac{\bar{b}_0}{(1-\sigma_1-\sigma_2)(1-\sigma_1)}. \end{aligned}$$

Taking into account that $\overline{w}_i = \bar{t}_i + \bar{b}_i$, then we have

$$\frac{\overline{w}_1}{\overline{w}_2} = (1-\sigma_1-\sigma_2) \cdot \frac{\bar{b}_0 + \bar{b}_1(1-\sigma_1)}{\bar{b}_0 + \bar{b}_2(1-\sigma_1)(1-\sigma_1-\sigma_2)}.$$

Conversely, $\overline{w}_1/\overline{w}_2$ will be maximum when Class 2 has a non-preemptive, higher priority than Class 1. In this case, and similar to the above, the lower bound is

$$\frac{\overline{w}_1}{\overline{w}_2} = \frac{1}{1-\sigma_1-\sigma_2} \cdot \frac{\bar{b}_0 + \bar{b}_2(1-\sigma_1)(1-\sigma_1-\sigma_2)}{\bar{b}_0 + \bar{b}_1(1-\sigma_1)}$$

□

Appendix D

Theorem 4: *When for two classes 1 and 2, $\sigma_1 + \sigma_2 = 1$, i.e., the system is saturated, and under Poisson arrivals and general service times, the achievable delay ratio between the two classes is such that:*

$$\frac{(\bar{b}_0 - \sigma_1\bar{b}_1 + \bar{b}_1)\sigma_2\bar{b}_1}{[(B-1)(1-\sigma_1)\bar{b}_1 - (\bar{b}_0 - \sigma_1\bar{b}_1 + \bar{b}_1)]\bar{b}_2} \leq \frac{\overline{w}_1}{\overline{w}_2} \leq \frac{[(B-1)(1-\sigma_2)\bar{b}_2 - (\bar{b}_0 - \sigma_2\bar{b}_2 + \bar{b}_2)]\bar{b}_1}{(\bar{b}_0 - \sigma_2\bar{b}_2 + \bar{b}_2)\sigma_1\bar{b}_2}$$

where $\bar{b}_0$ is the residual service time given by equation (4).

Proof: We prove this theorem by considering a packet that arrives when the buffer has only one empty location, and will therefore be accepted, and eventually served. Or, the packet arrives to find the queue full, and will remove a packet from the other class. This packet will also be served eventually. We also assume that the numbers of packets in the buffer are kept in accordance with the target ratio.

To obtain the lower bound, let class 1 have a higher non-preemptive priority over class 2. As such, the effect of class 2 packets is only through the residual service time of class 2 packets found in service upon arrival.

The mean waiting time of an arriving packet from class 1 which will be accepted in the buffer is given by

$$\begin{aligned}\overline{w_1} &= \overline{b}_0 + (\overline{q}_1 - \sigma_1)\overline{b}_1 + \overline{b}_1 \\ &= \frac{\overline{b}_0 - \sigma_1\overline{b}_1 + \overline{b}_1}{1-\sigma_1} \end{aligned} \quad (11)$$

Since the queue size changes by ± 1, then the distribution of the number of packets in the system seen by a departing packet is the same distribution seen by an arriving packet. This property was used in the above equation. Also, Little's result was used in order to relate the mean queue size to the mean waiting time. Notice that the mean number of class 1 packets which must be served ahead of the target packet is reduced by 1 if a class 1 packet is already in service when the target packet arrives; hence, the subtraction of the σ_1 term in the first equation.

Class 2 packets, which are assumed to have a lower priority, will be also affected by class 1 packets. Since the system is heavily loaded, then

$$\overline{q}_1 + \overline{q}_2 = B - 1$$

Therefore,

$$\overline{q}_2 = B - 1 - \overline{q}_1 = s_2\overline{w}_2$$

The last term is due to the application of Little's result. Hence, after simple algebraic manipulations while using the expression given in equation (11) and substituting $\overline{q}_1 = s_1\overline{w}_1$, we have

$$\begin{aligned}\overline{w}_2 &= \frac{B-1-\overline{q}_1}{s_2} \\ &= \frac{[(B-1)(1-\sigma_1)\overline{b}_1 - (\overline{b}_0 - \sigma_1\overline{b}_1 + \overline{b}_1)\sigma_1]\overline{b}_2}{\sigma_2(1-\sigma_1)\overline{b}_1} \end{aligned} \quad (12)$$

Dividing equation (11) by equation (12), we obtain the lower bound. The upper bound can similarly be obtained by assuming that Class 2 has a higher non-preemptive priority over Class 1. □

References

[1] A. Adas, "Traffic models in Broadband Networks", *IEEE Communications Magazine*, pp. 82-89, July 1997.

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," *RFC 2475,* Dec. 1998.

[3] R. Braden, D. Clark, and S. Shenkar, "Integrated Services in the Internet Architecture: an Overview," *RFC 1633,* June 1994.

[4] R. Braden, L. Zhang, F. Baker, and D.L. Black, "Resource ReSerVation Protocol (RSVP) Version 1, Functional Specification," *RFC 2205,* Sept. 1997.

[5] A. Charny et al., "Supplemental Information for the New Definition of the EF PHB (Expedited Forwarding Per-Hop Behavior)", Network Working Group Request for Comments, RFC 3247, March 2002.

[6] Y. Chen et al., "Proportional QoS Provision: A Uniform and Practical Solution", in the proceedings of the IEEE International Conference on Communications, ICC 2002, pp. 2363–2367.

[7] Y. Chen, C. Qiao, M. Hamdi and D. Tsang", "Proportional Differentiation: A Scalable QoS Approach", IEEE Communications, Vol. 41, No. 6, June 2003, pp. 52–58.

[8] D. Clark and W. Fang, "Explicit Allocation of Best Effort Packet Delivery Service", *IEEE/ACM Trans. on Networking*, vol. 6, pp. 362-373, Aug. 1998.

[9] D. Clark and W. Lehr, "Provisioning for Bursty Internet Traffic: Implications for Industry and Internet Structure", presented at *MIT ITC Workshop on Internet Quality of Service*, Nov. 1999

[10] M. E. Crovella and A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes", IEEE/ACM Trans. on Networking, Vol. 5, No. 6, December 1997, pp. 835–846.

[11] J. Crowcroft and P. Oechslin, "Differentiated end-to-end Internet Services using a Weighted Proportional Fair sharing TCP", in *ACM Computer Communication Review,* vol. 28, no. 3, pp. 53-67, July 1998.

[12] B. Davie et al., "An Expedited Forwarding PHB (Per-Hop Behavior)", Network Working Group Request for Comments, RFC 3246, March 2002.

[13] C. Dovrolis, D. Stiliadis and P. Ramanathan, "Proportional Differentiated Services: Delay Differentiation and Packet Scheduling," in *Proc. ACM SIGCOMM,* pp. 109-120, 1999

[14] C. Dovrolis and P. Ramanathan, "A Case for Relative Differentiated Services and the Proportional Differentiation Model", *IEEE Network,* vol. 13, no. 5, pp. 26-35, Sept. 1999.

[15] C. Dovrolis and P. Ramanathan, "Proportional Differentiated Services, Part II: Loss Rate Differention and Packet Dropping", in *Proc. IEEE/IFIP Intl. Workshop on Quality of Service,* pp. 52-61, June 2000.

[16] C. Dovrolis, D. Stiliadis and P. Ramanathan, "Proportional Differentiated Services: Delay Differentiation and Packet Scheduling", in *IEEE/ACM Transactions on Networking*, vol. 10, no. 1, pp. 12-26, Feb. 2002.

[17] C. Dovrolis and P. Ramanathan, "Dynamic Class Selection and Class Provisioning in Proprotional Differentiated Services", Computer Communications, vol. 26, 2003, pp. 204–21.

[18] S. Floyd and V. Paxson, "Difficulties in Simulating the Internet", in *IEEE/ACM Transactions on Networking*, vol. 9, no. 4, pp. 392-403, Aug. 2001.

[19] M. Hamdaoui and P. Ramanathan, "A Dynamic Priority Assignment Technique for Streams with (m, k)-Firm Guarantees", IEEE Trans. Computers, vol. 44, No. 12, Dec. 1995, pp. 1443-1451.

[20] D. P. Heyman and D. Lucantoni, "Modeling Multiple IP Traffic Streams with Rate Limits", IEEE/ACM Trans. on Networking, Vol. 11, No. 6, Dec. 2003, pp. 948–958.

[21] V. Jacobson, K. Nichols, and K. Poduri, "An Expedited Forwarding PHB", *RFC 2598,* July 1999.

[22] L. Kleinrock, "Queueing Systems, Vol. I: Theory", *John Wiley, New York,* 1975.

[23] L. Kleinrock, "Queueing Systems, Vol. II: Computer Applications", *John Wiley, New York,* 1976.

[24] W. E. Leland, M. S. Taqqu, W. Wilinger, and D. V. Wilson, "On the Self-Similar Nature Ethernet Traffic (Extended Version)", IEEE/ACM Trans. on Networking, Vol. 2, No. 1., Feb. 1994, pp. 1–15.

[25] M. Leung, J. Lui, and D. Yau, "Adaptive Proportional Delay Differentiated Services: Characterization and Performance Evaluation", in *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, pp. 801-817, Dec. 2001.

[26] J.-S. Li; H.-C. Lai, " Providing proportional differentiated services using PLQ", Globecom 2001, pp. 2280–2284.

[27] J. Liebeherr and N. Christin, "Rate Allocation and Buffer Management for Differentiated Services", Computer Networks, vol. 40, 2002, pp. 89–110.

[28] A. Striegel and G. Manimaran, "Packet Scheduling with Delay and Loss Differentiation," *Computer Communications,* vol. 25, no. 1, pp. 21-31, Jan. 2002.

[29] Z. Wang, "Internet QoS - Architecture and Mechanisms for Quality of Service," *Morgan-Kaufman Publishers,* 2001.