A MULTICONSTRAINED QOS AWARE SCHEDULER FOR CLASS-BASED IP NETWORKS

P. Sousa, P. Carvalho and V. Freitas
University of Minho, Department of Informatics, 4710-057 Braga, Portugal
email: {pns,pmc,vf}@di.uminho.pt

ABSTRACT

This article presents a modular scheduler with powerful semantics able to differentiate simultaneously multiple QoS metrics in class-based IP networks. In opposition to traditional scheduling mechanisms, this scheduler encompasses rate, loss and delay differentiation capabilities in a flexible way. This behaviour stems from new relative and mixed differentiation models able to bound QoS parameters on high sensitive traffic classes. The results show that using simple and intuitive configuration procedures the proposed architecture is able to provide enhanced QoS differentiation behavior in IP networks according to the users and applications needs. In this way, this proposal is an useful contribution to system designers and network engineers aiming at simple, intuitive, easy to configure and effective mechanisms to enhance QoS in IP networks.

Keywords

Quality of Service, Scheduling, Traffic Engineering, Queue Management, Resource Management.

1. INTRODUCTION

The growth and diversity of distributed applications has fostered the need for IP networks with quality of service (QoS) [1] differentiation capabilities in order to meet the users and applications demand. This need depends on the application nature and involved data, varying from relaxed to strict QoS constraints such as the ones required by human-interactive based applications, real-time distributed data processing or real-time multimedia transmission [2]. This type of applications is usually loss sensitive and has specific delay constraints that must be satisfied by the underlying network. To meet these requirements at the network level, it is fundamental to deploy traffic control mechanisms which are easy to implement and configure, while providing flexible QoS differentiation. Thus, the motivation of this work is to develop a clever scheduling mechanism which, despite resorting to simple and intuitive configuration tasks, is able to control simultaneously different QoS metrics in class-based IP Networks [3].

The scheduler proposed in this work is able to achieve independent control of delay, loss and rate differentiation through the use of two priority disciplines acting at distinct points of the scheduler architecture. The delay differentiation modules are based on theoretical schemes [4] and, in particular, proportional differentiation [5, 6, 7, 8] is considered as one of the possible options for delay differentiation. Other differentiation schemes are also supported [9, 10, 11, 12] by the scheduler, including an hybrid model specially devised for real-time differentiation. These delay models aggregate a packet drop mechanism in order to provide *(i)* loss differentiation or *(ii)* output rate differentiation with distinct work conserving behaviours or *(iii)* combination thereof. The present scheduling proposal can be viewed as a modular traffic control mechanism able to be configured with distinct semantics depending on each class QoS requirements, enhancing the scheduling QoS capabilities of a network node. The proposed model has been implemented and tested in the network simulator (*NS-2*).

The rest of the paper is organised as follows. Section 2 presents the scheduling architecture which integrates distinct differentiation modules and highlights the delay, loss and load control configuration models supported by the proposed architecture. Specific optimisations of the queue selection tasks are also discussed in Section 3. After that, Section 4 includes simulation results illustrating the viability of obtaining multiconstrained QoS differentiation semantics in class-based IP networks. Finally, Section 5 presents the conclusions of the work.

2. A MULTICONSTRAINED SCHEDULING ARCHITECTURE

To control multi-QoS metrics, the traffic scheduling architecture includes three distinct differentiation modules (see Fig. 1) [13]. The delay differentiation module acts as an output priority discipline. This module has to decide which queue should be attended in order to satisfy the delay semantics imposed by the supported delay models. The packet dropper acts as an input priority discipline and may rely on the loss module which includes a set of distinct loss differentiation models. In addition, under persistent congestion the packet dropper may also resort to the load control module which is able to control each class load inducing, over medium time scales, output rate differentiation.

The present scheduling proposal should not be understood as being oriented to a particular QoS model for IP networks. It should be viewed as a modular traffic control mechanism able to be configured with distinct semantics depending on the QoS requirements and corresponding control the network architecture requires. For instance, if no admission control is supported by the network then this scheduler may be adopted providing full differentiation capabilities using both the loss (or the load control) and delay modules. In opposition, if admission control is supported [14] (e.g. at network edges) hindering congestion in the network core, then the loss (or the load control) module may be disabled, relaxed or sporadically used to correct small loss or rate violations affecting high priority traffic classes. The same reasoning is valid for generic QoS traffic classes oriented to support distinct applications' requirements.

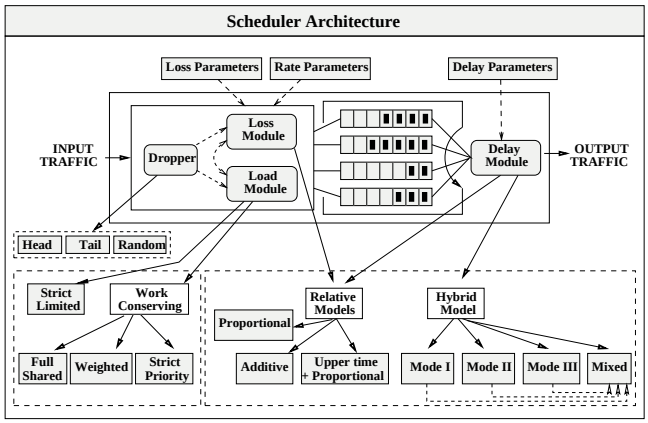


Figure 1. The scheduler architecture implemented in *NS-2*.

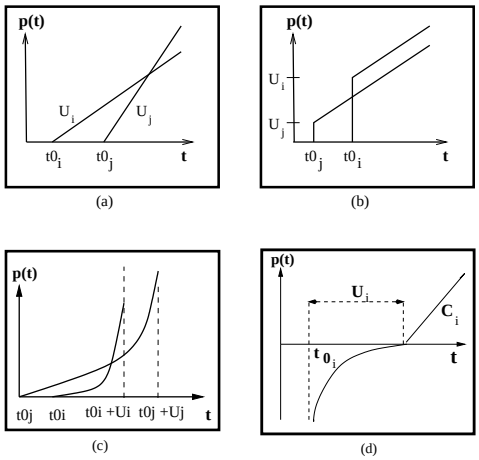


Figure 2. (a) Proportional model (b) Additive model (c) Upper time limit model (d) Hybrid model.

Some of them, due to their time sensitive nature, may be distinguished mainly by delay differentiation mechanisms, whereas others may rely on rate oriented differentiation mechanisms or Active Queue Management (AQM) schemes, such Random Early Detection (RED), to rule packet drop. Moreover, it is also possible that classes assuring both rate and delay guarantees (e.g. EF PHB in Differentiated Services [15]) may justify the use of strict delay differentiation mechanisms along with schemes protecting bandwidth resources or assuring low packet loss ratios on high priority classes.

In conclusion, each one of the modules included the scheduling architecture presented in Fig. 1 has its own viability and applicability in the context of traffic differentiation as it may be used in a stand-alone perspective inducing the corresponding QoS differentiation semantics. In addition, if they are organised as in Fig. 1 then it is possible to obtain multiple QoS metric differentiation in a flexible way, which may be very useful to define enriched differentiation semantics in class-based IP networks. In the next sections the emphasis is given on the main objectives, definition and configuration modes of the differentiation modules integrating the architecture presented in Fig. 1.

2.1 Delay Differentiation Module

This section presents four delay differentiation models included in the proposed scheduler. Lets consider N classes denoted as $Class_{i(0\leq i\leq N-1)}$ where C_0 is the highest priority class.

2.1.1 Proportional Model

Assume that $p_i(t)$ is the priority function associated with the queue i and U_i the corresponding differentiation parameter. In the proportional model this function is given by (1) , with t_{0_i} denoting the arrival time of the heading packet of queue i and $U_0 > U_1 > ... > U_{N-1}$. The behaviour of (1) for two packets belonging to distinct classes is depicted in Fig. 2(a) with U_i as the slope of the priority function. Under heavy load conditions, it is expected that (2) is valid for all classes ($0 \leq i, j < N$) where $\bar{d}_i, \bar{d}_j$ are the mean queuing delays of the classes i and j, i.e. the proportional delay relations are ruled by the U_i parameters:

$$p_i(t) = (t - t_{0_i}) * U_i \tag{1}$$

$$\frac{U_i}{U_j} \approx \frac{\bar{d}_j}{\bar{d}_i} \tag{2}$$

2.1.2 Additive Model

The additive model differentiates queues by an additive constant as expressed by (3), with $U_0 > U_1 > ... > U_{N-1}$ (Fig. 2(b)). The focus of this model is on the possibility of achieving additive differentiation in class delays, as expressed by (4), denoting that high priority classes may have a delay gain over low priority classes similar to the difference between the differentiation parameters:

$$p_i(t) = (t - t_{0_i}) + U_i \tag{3}$$

$$[\bar{d}_i - \bar{d}_j] \approx [U_j - U_i] \qquad (i > j) \tag{4}$$

2.1.3 Upper Time Limit Model

This model tries to impose a finite queuing delay, reflected in U_i (see (5)) and, the lower the boundary time is, the higher the priority function slope will be. At the limit ($(t - t_{0_i}) \geq U_i$), the server is *forced* to dispatch the packet waiting service (see Fig. 2(c)). However, when congestion occurs or the load of high priority classes becomes very high, the time limit may be exceeded:

$$p_i(t) = \begin{cases} \frac{(t-t_{0_i})}{U_i - t + t_{0_i}} & if \quad t < t_{0_i} + U_i \\ \infty \quad if \quad t \geq t_{0_i} + U_i \end{cases} \tag{5}$$

This model protects high priority classes, assuming that packets remain queued for a maximum value U_i, with $U_0 < U_1 < ... < U_{N-1}$. This allows to establish delay bounds on the highest priority class and, simultaneously, achieve proportional differentiation between the other classes. For instance, $Class_1$ can be *protected* by a realistic upper time limit, and $Class_2$ and $Class_3$ by *virtual* limits (e.g. $U_2, U_3 \gg U_1$). Proportionality between $Class_2$ and $Class_3$ is obtained as explained by (1).

2.1.4 An Hybrid Delay Differentiation Model

This model is useful to distinguish real-time traffic with distinct sensibilities to queuing delays and excess delays. In this model, the priority function, $p_i(t)$, varies from negative infinity to zero. A

$p_i(t) = 0$ means that the queuing time of the packet matches the upper time parameter. Consequently, if the packet is still in queue, the scheduler switches to a new working region of positive values, where the priority behaviour is ruled by a *congestion parameter* which determines the slope of the priority function. It is to be noted that in this context the term *congestion* is used in a relaxed way as it may reflect heavy load conditions in the server; heavy load conditions in $Class_i$ impairing the expected upper time limit or feasibility problems in the differentiation parameters. The final priority function is given by (6) and Fig. 2(d) illustrates its behaviour:

$$p_i(t) = \begin{cases} \frac{\delta_t - U_i}{\delta_t} & \text{if } \delta_t < \mathrm{U}_i \\ (\delta_t - U_i) * C_i & \text{if } \delta_t \geq \mathrm{U}_i \end{cases} \quad (6)$$

with $\delta_t = t - t_{0_i}$ and $0 \leq i \leq N-1$.

$$d_i = d_i^{\circ} + d_i^{\bullet} \quad (7)$$

The total delay (d_i) affecting $Class_i$ can be divided in two components: one induced by the priority function when it assumes negative values ($t < t_{0_i} + U_i$), which we call *upper time delay*, d_i°, and the other one when the function assumes positive values, which we call *congestion delay*, $d_i^{\bullet}$ (see (7)). The magnitude of d_i° is controlled by U_i whereas C_i controls the magnitude of $d_i^{\bullet}$. This means that fundamental differentiation relations among classes, i.e. $d_0 \leq d_1 \leq ... \leq d_{N-1}$, can be achieved through different combinations of d_i° and $d_i^{\bullet}$, and consequently by different combinations of U_i and C_i. Fig. 3 illustrates a distinct behaviour of this model (configuration I, II and III) depending on the relations between the *upper time* and *congestion* delays for two generic classes i and j with $i < j$. Mixed configuration modes are also possible to be used by the hybrid delay differentiation model.

In configuration mode I identical upper time limits are configured for the two traffic classes. This means that both classes share the same priority function as the packets stay in queue for a time limit below the configured U_i parameter. In this configuration the traffic classes are differentiated by C_i which means that in case of congestion the priority function for the higher priority class assumes higher values than for the other. This configuration mode may be appropriated for real-time classes with the same upper time limit for queuing delay and distinct capabilities to compensate possible delay violations. The expected behavior of this model is that under feasible conditions the specified upper time limits for both classes are achieved, i.e. $d_i = d_i^{\circ} = d_j = d_j^{\circ} < U_i$ or $< U_j$. However, if the server becomes overloaded and the upper time limit delays of the classes are violated the excess delays of the classes are differentiated in a proportional way.

In configuration mode II the traffic classes are distinct with respect to the upper time differentiation parameters and the congestion differentiation parameters. As result, the priority function associated with higher priority classes has a larger increase than the lower ones for both negative and positive values of $p_i(t)$. This means that under congestion both congestion delay and upper time delay associated with high priority classes should be lower than the ones associated with the other classes. This configuration is appropriate to differentiate high delay sensitive applications with low capacity to compensate excess queuing delays.

The configuration mode III differentiates traffic classes only by U_i parameters. This configuration is used to distinguish a class by its maximum queuing delay limit but, in case of violation, the classes share the same priority behavior for the excess queuing delays, i.e. it is assumed that the classes have similar capacity to compensate excess queuing delays.

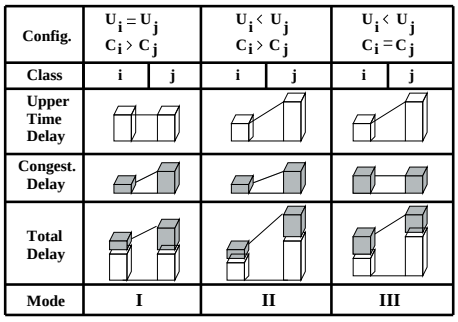


Figure 3. Configuration modes of the hybrid delay model.

Algorithm 1 Pseudocode of the loss differentiation module

```
[Event: class_i packet arrival]: A_i++
[Event: class_i packet drop]: drop_i++
[Event: Δt period elapsed]: drop_i = 0, A_i = 0
[Event: buffer overflow on a class_j packet arrival]:
for all class_i do
  if ((length(class_i) > 0 or i==j) and (A_i > 0)) then
    /* evaluation of the class priority using distinct models*/
    priority_i = evaluate(p_i(t))
  end if
end for
class_drop = (class with the lowest priority_i value)
if (class_drop == j) then
  drop(arrived packet)
else
  drop(tail of class_drop queue)
  enqueue(arrived packet)
end if
```

2.2 Loss Differentiation Module

The loss differentiation module can also be configured according to relative models similar to those used for delay differentiation. This means that this module is able to provide proportional, additive and bounded loss ratios among the traffic classes using the loss parameters $(L_0, ..., L_{N-1})$.

In order to enable packet loss differentiation among distinct traffic classes, lets consider that $drop_{i,\Delta t}$ measures the number of packet drops in Δt and $A_{i,\Delta t}$ measures the number of packet arrivals of $Class_i$ in the same time interval. This means that A_i and $drop_i$ counters are reinitialized periodically each Δt. This makes the loss module more reactive to class load oscillations. If these counters hold cumulative values during the differentiation process, the loss differentiation module will not sense transient congestion periods of the network properly. Using this reasoning, $l_{i,\Delta t}$ denotes the packet loss ratio experienced by $Class_i$ over the time period used to evaluate $drop_{i,\Delta t}$ and $A_{i,\Delta t}$ variables as expressed by (8). To simplify the notation, from now on we suppress the index Δt, i.e. $drop_i$, A_i and l_i represent variables in Δt:

$$l_i = \frac{drop_i}{A_i} \quad (8)$$

The goal of the loss differentiation module is the provision of distinct loss differentiation semantics among the traffic classes contending for an output link. For that purpose, the use of common tail drop based mechanisms is no longer suitable to induce loss differentiation as they do not take into account the relative priority of

the classes. In opposition, it will be necessary that, under buffer overflow, the decision of dropping a packet attends the priorities and the current loss ratio of each class. With this purpose, whenever a packet arrives at the differentiation node and, simultaneously, no buffering resources are available, the drop module should be able to discard a previously enqueued packet from a specific traffic class, accepting the newly arrived packet in the corresponding queue. The drop module may also discard a group of packets, if a single drop is not sufficient to provide the buffer resources required to store the incoming packet. Using this mechanism it is possible to tune packet loss among traffic classes according to a predefined differentiation model. In the presented architecture, where distinct traffic classes have distinct queues, the drop decision occurs whenever the aggregate backlog is higher than a given threshold. This means that the node buffering resources are shared by all traffic classes and, as they are mapped to independent queues, the corresponding queue sizes may vary dynamically during the node operation. This coupled operation mode is different from traditional AQM techniques, such as RIO-coupled [16], where the dropping decision is centered on a particular traffic class.

With this purpose the loss differentiation module has to evaluate, for each traffic class, a priority value reflecting the likelihood of packet dropping, i.e. the traffic class with the lowest priority value is the one selected for packet discarding. Algorithm 1 presents the pseudocode for the loss differentiation module, which behavior is ruled by the priority function $p_i(t)$. Relaxed versions of this algorithm are possible such as estimating $p_i(t)$ only at the end of each Δt period, i.e. the candidate class for packet drop is kept unchanged during the following Δt. Despite being less accurate and reactive, this variant has a lower processing overhead given that, in the case of buffer overflow, it is not necessary to compute the priority values of all traffic classes. The loss differentiation module also follows some of the models previously explained for delay differentiation. In this case, instead of packet queuing times, i.e. $t - t_{0_i}$, the priority functions will use the current packet loss ratio of the classes, l_i, to decide from which class a packet is selected for dropping. In this case, the proportional loss differentiation is ruled by (9), the additive loss differentiation by (10) and the upper bound loss model, defined by (11), allows to bound the packet loss ratios on high priority classes:

$$p_i(t) = (l_i) * L_i \tag{9}$$

$$p_i(t) = (l_i) + L_i \tag{10}$$

$$p_i(t) = \begin{cases} \frac{(l_i)}{L_i - (l_i)} & if \;\; l_i < L_i \\ \infty & otherwise \end{cases} \tag{11}$$

2.3 Load Control Module

This section focuses on one of the roles of the packet drop mechanism associated with the scheduler. The mechanism is able to induce output rate differentiation among multiple traffic classes by controlling the corresponding loads. Consider that the traffic arriving at a network node, to be forwarded to a specific output link, is classified in N distinct traffic classes contributing with individual loads $R\_in_i(t)$ with $0 \leq i \leq N-1$. From queuing theory, the server associated with the corresponding output link enters in an unbalanced state, $\rho > 1$, where $\rho = \lambda \cdot \bar{S}$ with λ as the arrival rate and $\bar{S}$ the average service time. This means that the total traffic class load at the input exceeds the output capacity of the link, C. This situation, illustrated in (12), leads to packet loss and to different levels of throughput share depending on the service discipline, class load and buffering resources:

$$C < \sum_{i=0}^{N-1} R\_in_i(t) \tag{12}$$

$$C \geq \sum_{i=0}^{N-1} min(R\_in_i(t), R\_max_i) \tag{13}$$

$$C \geq \sum_{i=0}^{N-1} R\_max_i \tag{14}$$

The first step in the load control module design assures that (12) is not satisfied, i.e. the total arriving load does not exceed the output capacity of the server. Thus, to each $Class_i$ is assigned a value, R_max_i, which is the maximum input rate to be submitted to the server. If $R\_in_i(t)$ measures $Class_i$ input load at time t then (13) is valid and assures that the server is always under a balanced state ($\rho \leq 1$). This means that, assuming enough buffering resources, the server is able to forward all traffic, i.e. on average, the R_max_i will also represent the output rate share obtained by the $Class_i$. Assuming N distinct classes, it is clear that the sum of R_max_i values should not exceed the output capacity of the server, as denoted by (14). $R\_in_i(t)$ is estimated resorting to an adaptive exponential weighted moving average, (15), where l_i^k is the length of the k^{th} packet of $Class_i$ and $\Delta t_i^k = t_{0_i}^k - t_{0_i}^{k-1}$ is the inter packet arrival time. The parameter T acts as a reference value which should have a similar order of magnitude of the time period for which the estimation module is expected to provide average rate information. With different objectives a similar equation is presented in [17], highlighting the convergence ability of the estimation. In addition, the dropping mechanism was conceived so that the unused share of bandwidth of $Class_i$ is assigned to a variable $credit_i(t)$ (see (16)) representing the amount of bandwidth provided by $Class_i$ to the differentiation node for subsequent distribution. The sum of all $credit_i(t)$ values is represented by $Credits(t)$, where the boolean variable, $cong_i$, is true if $R\_in_i(t) \geq R\_max_i$:

$$R\_est_i = (1 - exp^{-\frac{\Delta t_i^k}{T}}) \cdot \frac{l_i^k}{\Delta t_i^k} + exp^{-\frac{\Delta t_i^k}{T}} \cdot R\_est_i^{old} \tag{15}$$

$$credit_i(t) = \begin{cases} R\_max_i - R\_in_i(t) & if\ !(cong_i) \\ 0 & if\ (cong_i) \end{cases}$$

$$Credits(t) = \sum_{i=0}^{N-1} credit_i(t) \tag{16}$$

Within this work conserving behaviour, (17) determines the server operating under a balanced state. The function $limit_i(t)$ defines the maximum throughput share for each class. If the traffic class exceeds its R_max_i then $limit_i$ will increase R_max_i of a value given by a credit distribution function, $dist(t)$ (see Tab. 1). The dropping mechanism associated with (17) is now ruled by (18) assuring a reactive response to load oscillations and redirecting the unused bandwidth to the congested classes:

$$limit_i(t) = \begin{cases} R\_max_i & if\ !(cong_i) \\ R\_max_i + dist(Credits(t)) & if\ (cong_i) \end{cases}$$

$$C \geq \sum_{i=0}^{N-1} min(R\_in_i(t), limit_i(t)) \tag{17}$$

	Differentiation Equation
a	$limit_i(t) = R\_max_i + \frac{Credits(t)}{\sum congested}$
b	$limit_i(t) = R\_max_i + \frac{excess_i}{\sum_{j=0}^{N-1} excess_j} \cdot Credits(t)$
c	$limit_0(t) = R\_max_i + \min(excess_0, Credits(t))$ $limit_i(t) = R\_max_i + \min(excess_i, Credits(t) -$ $- \sum_{j=0}^{i-1}(limit_j(t) - R\_max_j))(i > 0)$

Table 1. Distribution modes of server credits: (a) Full Shared, (b) Weighted, (c) Strict Priority.

$$drop\_prob_i(t) = 1 - \frac{limit_i(t)}{R\_in_i(t)} \ if\ (R\_in_i(t) > limit_i(t)) \quad (18)$$

It is to be noted that relaxed versions of the load control module are possible. For instance, the estimation process may operate only during specific probing periods after which the $drop\_prob_i$ is computed and kept unchanged till the next period. Another possible variant consists of computing $limit_i$ and $drop\_prob_i$ values only for specific time intervals, despite the class rate estimation being continuously updated.

In Table 1, three distinct equations ruling the credits distribution modes are presented. The first is called *full shared* and distributes the available resources among the traffic classes evenly. The second is named *weighted* and allocates higher credit shares to traffic classes with higher excess rates. In this context, excess rate is used to denote the difference $R\_in_i - R\_max_i$, if $R\_in_i > R\_max_i$, and is represented by $excess_i$. The third distribution mode is called *strict priority* and allocates credits to traffic classes according to their priority, i.e. server credits are first allocated to high priority classes resorting to a recursive equation in which $limit_i$ assigned to $Class_i$ depends on the other $limit_j$ of low priority classes.

3. QUEUE SELECTION OPTIMISATION

The delay differentiation mechanisms presented in Section 2.1 achieve the expected differentiation behaviour through the evaluation of priority values, $p_i(t)$ for each traffic class, whenever the node has a packet for transmission. Although being based on simple arithmetic operations, the processing time required to compute $p_i(t)$ may become significant when the output capacity of the server increases, which in turn leaves less CPU time for queue selection procedures.

In this context, any improvement in the queue selection procedures will represent an overall gain for the performance of the models when implemented in a real network. In this work, the influence of the selection procedures is measured using (19), where γ expresses the service degradation ratio of the differentiation model. In this formula, ρ denotes the server utilisation whereas $\rho^{\circlearrowleft}$ denotes the server utilisation taking into account the processing overhead induced by queue selection procedures:

$$\gamma = \left(\frac{\rho^{\circlearrowleft} - \rho}{\rho} \right) \quad (19)$$

According to the queuing fundamentals and attending to (19), the following relation is obtained: $\gamma = (\frac{\lambda \cdot (\bar{S} + t^{\circlearrowleft}) - \lambda \cdot \bar{S}}{\lambda \cdot S})$, where $t^{\circlearrowleft}$ represents the amount of time required to compute the next queue to be served, which is a platform-dependent factor. As consequence, this relation can be written as $\gamma = (\frac{t^{\circlearrowleft}}{\bar{S}})$, meaning that the service degradation is given by the ratio between the queue selection time and the average service time. In conclusion, (20) can also be used to compute the service degradation ratio, with k representing the average packet size and C_l the output link capacity:

$$\gamma = \left(\frac{t^{\circlearrowleft}}{k} \cdot C_l \right) \quad (20)$$

From (20) it is possible to argue that for a fixed output capacity and $t^{\circlearrowleft}$ value, γ depends highly on the packet size. This means that the higher the packet size is, the lower the service degradation will be. In contrast, for lower packet sizes the influence of the time wasted in queue selection will be higher, therefore, the service degradation increases. Due to the high capacity of current computational systems, low values for the service degradation ratio γ are expected. Nevertheless, even these small deviations may significant influence in the system behaviour. In fact, for working regions where $\rho > 0.6$ even small increases in the server utilisation may lead to considerable increases as concerns queuing delays and the number of customers in the system. For this reason, the overhead induced by queue selection procedures should be reduced as much as possible. The next section presents an algorithm to improve the queue selection tasks.

3.1 Advanced Transmission Time Algorithm

Algorithm 2 Advanced transmission time algorithm.

$tpacket \leftarrow \frac{sizeof(packet)}{C_l}$
$< ..sendpacket... >$
$Sel \leftarrow AdvPriority([C_0, C_{N-1}], t + tpacket)$ — $t^{\leftrightarrows}$
$while\ (serverbusy)\ null$
$Sel \leftarrow HigherPriority([C_0, C_{Sel}])$ — $t^{\circlearrowleft}$

The algorithm presented in this section is based on the inspection of specific data fields of the packet header that has been selected for transmission. Using this approach, it is possible to determine the packet length and evaluate the expected packet transmission time ($tpacket$). This means that the next queue selection (i.e. after busy period) should occur at the time instant $t + tpacket$. This previous knowledge allows the selection, during the busy period, of the next class to be served. This is done by single round robin of the traffic classes, evaluating for each one the corresponding $p_i(t)$ value as it was computed at $t + tpacket$. Algorithm 2 illustrates this behaviour. The keypoint of this strategy is that a substantial part of the queue selection procedures is done during the busy period ($t^{\leftrightarrows}$) reducing the time of the selection procedures after the busy period ($t^{\circlearrowleft}$) which is effectively responsible for service degradation.

The last line of Algorithm 2 is only required when a higher priority class is empty, the $AdvPriority$ function is called and, meanwhile, a new packet arrives for that class. In this case, it is necessary to select the highest priority value for the interval $[Class_0, Class_{Sel}]$, where C_{Sel} is the selected class during the previous busy period. However, as referred before, the differentiation mechanisms are designed mainly for heavy load traffic condi-

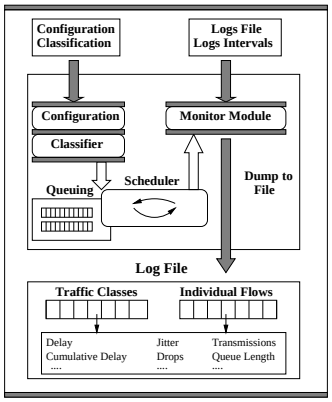


Figure 4. Experimental testbed implemented in *NS-2*.

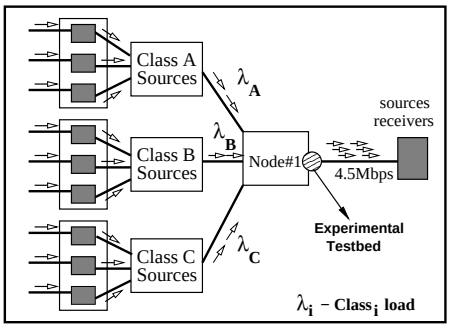


Figure 5. Simulation scenario.

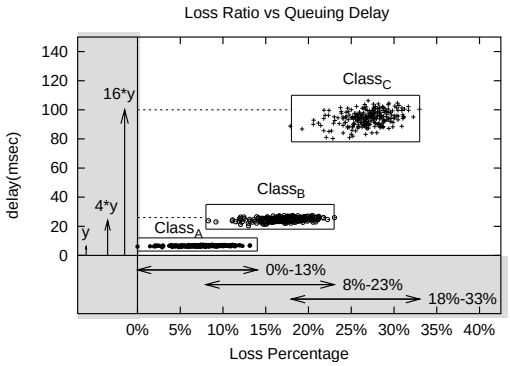


Figure 6. Additive loss and proportional delay differentiation models.

tions, which means that the probability of having an empty queue during the busy period is very low. So, a simple notification flag can be used to notify this specific event. This means that a server using Algorithm 2 will reduce the service degradation induced by queue selection tasks. At limit, the server might achieve the following performance: $t^{\circlearrowleft} \approx 0$, $\rho^{\circlearrowleft} \approx \rho$ and $\gamma \approx 0\%$, i.e. a performance similar to the obtained by the theoretical model.

Recall that Algorithm 2 still has complexity $O(n)$. In fact, in the worst case, two complete loops inspecting the classes' heading packets lead to $O(n) + O(n) = O(n)$. However, in this case, the first loop is performed during the busy period meaning that it does not affect the server utilisation and the second, considering high load conditions, is unlikely to happen. This means that, for heavy load conditions, a probabilistic analysis of the part of the algorithm performed after the busy period shows that its complexity is analogous to $O(1)$ since, in practice, the queue selection decision was already performed during the busy period. This solution can easily be implemented avoiding the use of specific hardware. In our opinion, even if the test platform or the operating assumptions vary leading to higher service degradation values, the use of Algorithm 2 will always be an added value as regards reducing service degradation.

4. RESULTS AND DISCUSSIONS

The scheduling architecture presented in Fig. 1 aggregating all the previously explained differentiation mechanisms was implemented in the network simulator (*NS-2*). Specific queues and monitors were also developed in order to collect results from the tests. Fig. 4 shows the implemented architecture associated with the output link of a differentiation node. At Otcl level, the input and output priority disciplines are selected, the differentiation parameters of the queues/classes are defined and classification data is provided, i.e. $(packet_{flowid}, queue_{id})$ pairs. At the same level, the state information granularity to be logged at scheduling time is indicated. In the architecture core, the monitor module logs periodically state information about flows/classes for subsequent analysis.

This section illustrates that the proposed scheduling architecture is able to decouple the rate, loss and delays differentiation behaviour, i.e. the differentiation mechanisms can act jointly but, simultaneously, can provide independent QoS metric differentiation. Due to the high number of possible differentiation schemes this section only covers illustrative examples of specific configuration modes. The selected examples were taken from a scenario where three classes ($Class_A$, $Class_B$ and $Class_C$) contend for a $4.5Mbps$ capacity link, with packet lengths of 500 bytes uniformly distributed over the interval $[250, 750]$ (see Fig. 5). In the selected examples the simulation period is $120s$ with a QoS metric evaluation interval of $1s$ and with the overall class load above the link capacity to force heavy load conditions and packet loss. The scheduler was tested successfully for distinct traffic sources such as CBR, exponential, on-off, pareto and combinations thereof.

4.1 Loss vs. Delay Differentiation

This section presents distinct differentiation examples with the loss and delay differentiation modules acting together. First, an example involving relative loss and delay differentiation models is presented. After that, the delay and loss differentiation modules were configured with the upper time and upper bound differentiation models, respectively. Using that configuration, three examples illustrate the enhanced differentiation semantics that can be obtained by such configuration modes. Finally, the last section includes one example of a relative loss differentiation model acting together with the hybrid delay model which is able to control the spread among the classes' excess delays.

4.1.1 Additive Loss and Proportional Delay Models

This example illustrates a scenario where the delay differentiation module is configured to follow the proportional model whereas the loss differentiation is ruled by the additive model. In this con-

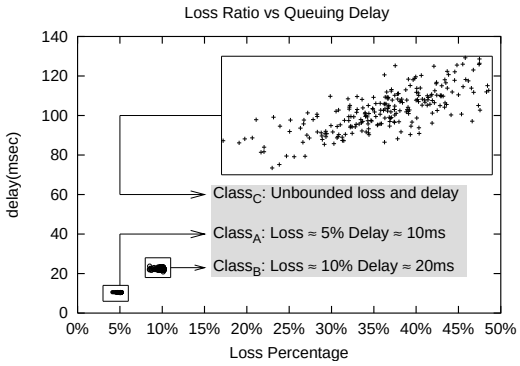


Figure 7. Upper bound loss and upper time delay differentiation models.

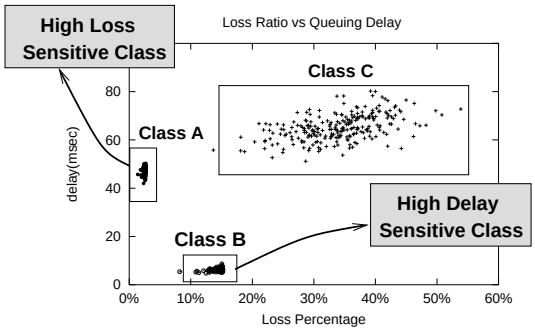


Figure 8. Upper bound loss and upper time delay differentiation models (with $Class_A$ as the the high loss sensitive class and $Class_B$ as the high delay sensitive class).

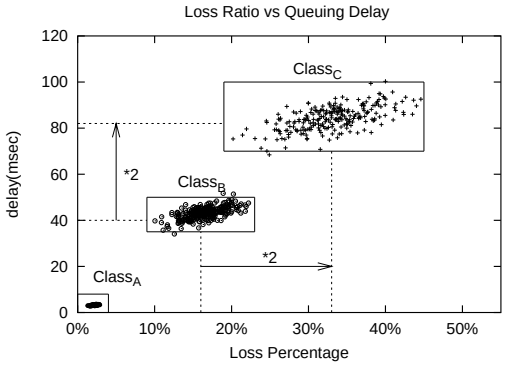


Figure 9. Mixed behaviour of the upper bound loss and upper time delay differentiation models.

text, the results of Fig. 6 report to a configuration with $(U_A, U_B$,-$U_C) = (16, 4, 1)$ and $(L_A, L_B, L_C) = (0.2, 0.1, 0)$. This means that proportional relations among the queuing delays are expected, however the spread among the loss ratios should follow the additive model, with the highest priority class having a loss gain of approximately 10% over $Class_B$ and a gain of 20% over $Class_A$, as ruled by the differences among the loss parameters. The analysis of the results presented in Fig. 6 clearly corroborates such loss and delay differentiation semantics. As observed, the proportional relations among the classes' queuing delays follow the proportional relations defined by the delay parameters and the differences between the classes' packet loss ratios are close to the differences defined by the corresponding loss parameters, showing the viability of the coexistence of both models.

4.1.2 Upper Bound Loss and Upper Time Delay Models

Three examples illustrate the coupling of the upper bound loss and upper time delay differentiation models. As explained, these models might be used to bound the loss and delay QoS metrics on high priority classes, meaning that the differentiation node will try first to satisfy the QoS requirements imposed by the highest priority class and only if they are met the differentiation mechanisms will try to satisfy the QoS constraints of the other classes.

The first example, presented in Fig. 7, corresponds to a loss configuration which tries to bound the packet loss ratios of the classes A, B and C to values of 5%, 10% and 20%, respectively. The delay configuration used in this example intends to bound the queuing delays of the classes to values of $10ms$, $20ms$ and $30ms$. The results presented in Fig. 7 clearly show three distinct areas with a high plot density. In the first one, which corresponds to the performance of $Class_A$, the plots are extremely concentrated in a small area denoting a packet loss ratio of 5% and a queuing delay of $10ms$, exactly the loss and delay bounds imposed by the L_A and U_A parameters. In the second plotting area, which illustrates the results of $Class_B$, the plots fall within an area denoting loss ratios of 10% and queuing delays around $20ms$, as ruled by the L_B and U_B parameters of $Class_B$. In opposition, the third plotting area, relative to $Class_C$, exhibits a higher plot dispersion meaning that this class has unbounded QoS metrics. This example clearly show that it is possible to bound the delays and losses on high priority classes simultaneously. In this case, the differentiation node is able to meet the requirements of $Class_A$ and $Class_B$ but due to the congestion level of the node and the classes' parameters it cannot fulfil the QoS bounds of $Class_C$, the lowest priority class.

In the differentiation examples presented so far the higher priority class, $Class_A$, experiences better, or at the least similar, delay and loss performance than lower priority classes. However, using the proposed differentiation modules, it is possible to define differentiation schemes where a class having the higher priority as regards to one of the QoS metrics, may have a lower priority in the context of other metric. The results presented in Fig. 8 where obtained from a simulation scenario were $Class_A$ is the highest loss sensitive class, configured with a L_A parameter of 2.5%, but, simultaneously, is able to suffer queuing delays in the order of $50ms$. In opposition, $Class_B$ is considered as a high delay sensitive class, with a delay bound of $5ms$, but with a L_B parameter of 15%. These assumptions lead to the differentiation results presented in Fig. 8 where $Class_A$ and $Class_B$ plots are now positioned in two distinct regions obeying to the loss and delay constraints imposed by the corresponding parameters. This example shows the versatility of the proposed scheduling architecture in the attainment of enhanced differentiation semantics, not obliging that a single class has the better performance in all of the supported differentiation modules.

The last example included in the section illustrates a mixed differentiation behaviour involving the upper bound loss and upper time delay models. In this case, $Class_B$ and $Class_C$ loss and de-

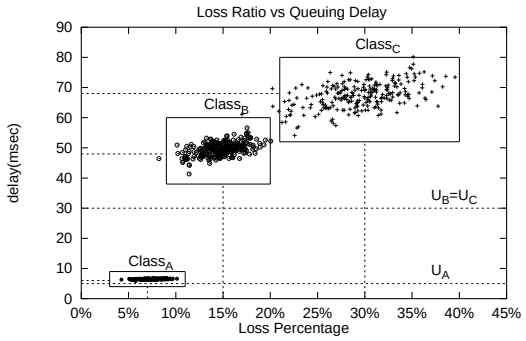
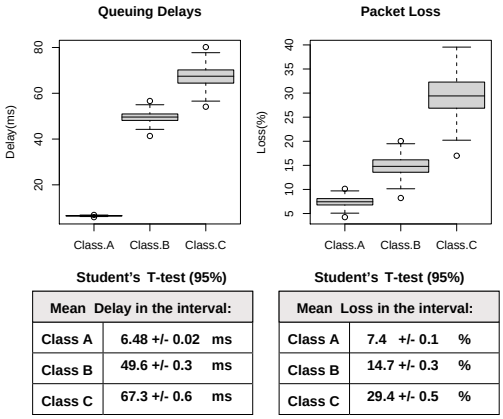


Figure 10. Proportional loss and hybrid delay differentiation models (Configuration I+II).



Student's T-test (95%)

Mean Delay in the interval:		
Class A	6.48 +/- 0.02	ms
Class B	49.6 +/- 0.3	ms
Class C	67.3 +/- 0.6	ms

Student's T-test (95%)

Mean Loss in the interval:		
Class A	7.4 +/- 0.1	%
Class B	14.7 +/- 0.3	%
Class C	29.4 +/- 0.5	%

Figure 11. Box-Whisker plots and Student's t-Test for the mean values of delay and loss metrics (of Fig. 10).

lay parameters were configured with very high values which, for this specific simulation scenario, are unlikely to occur. As previously explained, under such configuration, it is expected that the upper bound loss and the upper time delay models, while protecting the loss and delay of the highest priority class, induce proportional relations among the low priority classes, $Class_B$ and $Class_C$, accordingly to the proportional relations between the loss and delay parameters. The results presented in Fig. 9 illustrate such mixed differentiation semantics for a configuration $(L_A, L_B, L_C) = (0.05, 0.4, 0.8)$ and $(U_A, U_B, U_C) = (5ms, 80ms, 160ms)$. As observed, $Class_A$ is highly protected, having packet loss ratios and average queuing delays never exceeding 5% and $5ms$, respectively. Meanwhile, the differentiation semantics between $Class_B$ and $Class_C$ follows a proportional relation. In fact, the packet loss ratios and the queuing delays experienced by $Class_C$ are roughly two times higher than the obtained by $Class_B$, i.e. following the relation between L_B and L_C parameters, and between U_B and U_C parameters, respectively.

4.1.3 Proportional Loss and Hybrid Delay Models

In the example included in this section the classes are configured to have proportional loss differentiation with $(L_A, L_B, L_C) = (4, 2, 1)$. They are also configured for an hybrid delay differentiation with $(U_A, U_B, U_C) = (5ms, 30ms, 30ms)$ and $(C_A, C_B, C_C) = (40, 2, 1)$, i.e. Conf. I+II. This means that proportional packet loss is expected and, due to a very high C_A parameter, $Class_A$ should have queuing delays close to $5ms$. In addition, the congestion delays of $Class_C$, i.e. the difference between the obtained delays and the target delay of $30ms$, should be twice the congestion delay of $Class_B$, which has a similar delay target of $30ms$, but a congestion parameter two times higher than $Class_C$. This behavior is illustrated in Fig. 10. An alternative results analysis is made in Fig. 11 which shows the Box-Whisker plots of each QoS metric along with student's T-test, with a confidence interval of 95% for the mean values of delay and loss, corroborating the expected differentiation behavior.

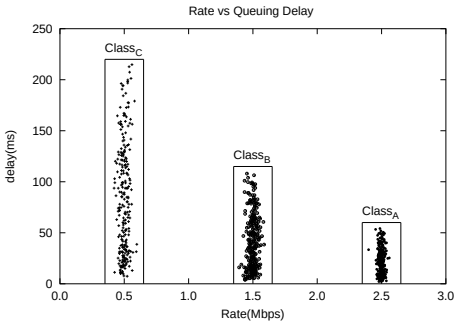


Figure 12. Load control and proportional delay differentiation models.

4.2 Output Rate Share vs. Delay Differentiation

This section includes two illustrative examples of the load control module operating jointly with the delay differentiation module. The selected examples show that is possible to control, on average, the maximum output rate shares obtained by the traffic classes and, simultaneously, differentiate the queuing delays. In addition the second example included in this section shows that the differentiation semantics can be maintained even after rate borrowing operations among the traffic classes.

4.2.1 Load Control and Proportional Delay Model

Figs. 12 illustrates the proportional delay differentiation behaviour, for $(U_A, U_B, U_C) = (4, 2, 1)$, coupled with the load control model with parameters $(R\_max_A, R\_max_B, R\_max_C) = (2.5Mbps, 1.5Mbps, 0.5Mbps)$. Fig. 12 proves that the configuration is feasible as the plots are vertically centred in the correspondent class target rate and, simultaneously, the delay spread of $Class_A$ is almost four times lower than the obtained by $Class_C$ and approximately half than the obtained by $Class_B$. This examples illustrates that is possible to control the average output rate share obtained by the classes and, simultaneously differentiate the queuing delays according to one of the delay models supported by the scheduling architecture.

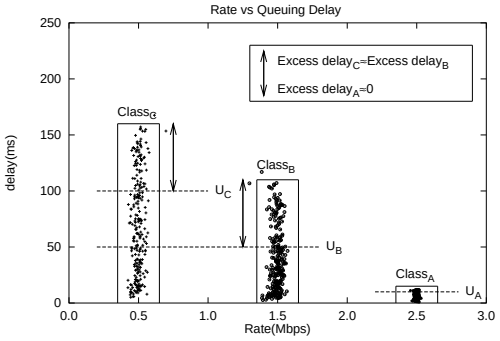


Figure 13. Load control with hybrid delay differentiation model (Configuration II+III).

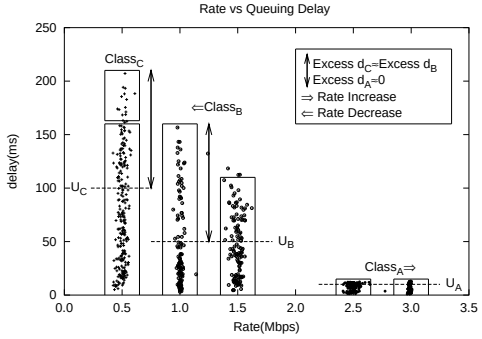


Figure 14. Strict priority load control with hybrid delay differentiation model (Configuration II+III).

4.2.2 Strict Priority Load Control and Hybrid Delay Model

The example of Fig. 13 illustrates the operation of the hybrid delay differentiation model (Conf. II+III) and the load control module for a configuration with (R_max_A,R_max_B,R_max_C) = ($2.5Mbps$,$1.5Mbps$,$0.5Mbps$), (U_A,U_B,U_C) = ($10ms$,$50ms$,-$100ms$) and (C_A,C_B,C_C) = (20,1,1). In the delay configuration, $Class_A$ is the highest protected class as regards both rate and delay violations and $Class_B$ and $Class_C$ have distinct U_i but similar C_i parameters, meaning that they have similar capacity to compensate excess delays despite having different upper time delays. Fig. 13 shows the average output rate and queuing delays obtained by the classes, clearly corroborating the expected differentiation behavior. Fig. 14 illustrates a similar delay differentiation, but now with the load control module operating under the strict priority distribution mode. Fig. 14 plots the differentiation behavior when $Class_B$ decreases its rate to $1Mbps$. As shown, only $Class_A$, which has the highest priority, has assigned extra bandwidth (shift to the right side of the graph), exactly the $0.5Mbps$ share provided by $Class_B$. As a consequence, a new delay distribution occurs at the server and both $Class_B$ and $Class_C$ delays increase. For $Class_C$, all plots are still centered on $0.5Mbps$ as this class does not receive any extra bandwidth. The increase in $Class_C$ excess delay is represented by a second box above the previous one. The magnitude of $Class_B$ and $Class_C$ excess delays is still similar even after the rate sharing, while $Class_A$ delay violations keep a low value due to its high C_A parameter.

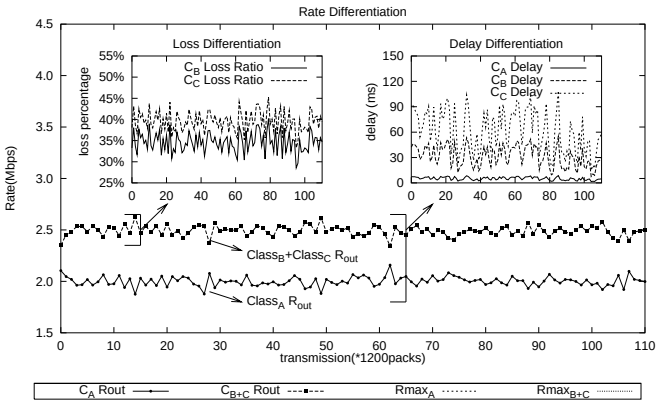
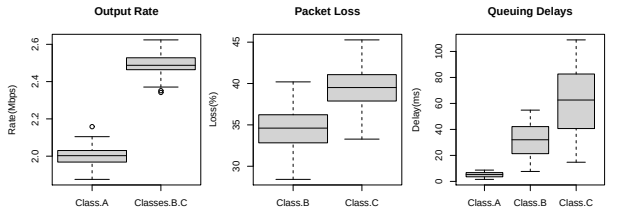


Figure 15. Load control with additive loss and upper time delay differentiation models.



Mean Rate in the interval:		Mean Loss in the interval:		Mean Delay in the interval:	
Class A	2.001 +/- 0.008 Mbps	Class B	34.6 +/- 0.4 %	Class A	5.17 +/- 0.32 ms
Class B+C	2.491 +/- 0.008 Mbps	Class C	39.5 +/- 0.4 %	Class B	31.5 +/- 2.2 ms
				Class C	62.1 +/- 4.2 ms

Figure 16. Box-Whisker plots and Student's t-Test for the mean values of delay, rate and loss metrics (of Fig. 15).

4.3 Output Rate Share vs. Loss vs. Delay Differentiation

The example included in this section illustrates the three differentiation modules acting together. In the selected scenario the load control module operates jointly with the additive loss and upper time delay models. The objective is to show that it is possible to obtain enhanced differentiation semantics induced by the delay, loss and load control module operating jointly and controlling each one of the corresponding QoS metrics.

In the selected example it was assumed that $Class_A$ is used for loss and time sensitive traffic, being its bandwidth limited at network edges to $2Mbps$. $Class_B$ and $Class_C$ are used for low priority traffic and, depending on the network conditions, packet loss is likely to occur. In this context, the rate parameters are configured as $(R\_max_A, R\_max_{B+C}) = (2Mbps, 2.5Mbps)$. The additive model is used to guide loss differentiation between $Class_B$ and $Class_C$ with $(L_B, L_C) = (0.05, 0)$, meaning that $Class_B$ should experience a loss percentage which is 5% lower than the obtained by $Class_C$. Finally, the upper time model is used to limit the queuing delay of $Class_A$ to a maximum of $10ms$, with proportional relations between $Class_B$ and $Class_C$. As depicted in Fig. 15, the output rate share of $Class_{B+C}$ aggregates is close to $2.5Mbps$ whereas $Class_A$ share is around $2Mbps$. Moreover, the subfigures inside Fig. 15 show that the delay and loss differentiation also obey

to the configured parameters. The results are corroborated once again by the stand-alone metrics analysis of Fig. 16. This example proves that, despite being easily configurable, the scheduling architecture has a powerful differentiation semantics to improve QoS capability of network nodes.

5. CONCLUSIONS

This article presents a modular scheduler architecture providing enhanced rate, loss and delay differentiation behaviour. The proposed architecture integrates three distinct differentiation modules which may operate stand-alone or jointly. Each one of the devised differentiation modules supports distinct configuration modes inducing distinct differentiation semantics among the traffic classes. The supported configuration models might be used in a qualitative or quantitative differentiation perspective. In addition, some of the devised models allow to bound the QoS metrics on high priority classes and differentiate the other classes in a relative perspective, thus assuming an hybrid differentiation perspective.

The diversity of the configuration modes for the three QoS metrics turns the proposed scheduler in an useful component to be used in network scenarios aiming at QoS differentiation. The proposed scheduler allows to achieve independent QoS metrics differentiation behaviour, avoiding coupling effects which may affect other differentiation mechanisms. Due to the enhanced differentiation semantics, many combinations of rate, loss and delay differentiation behaviour are possible using a small set of simple and intuitive configuration parameters.

REFERENCES

[1] G. Armitage. *Quality of Service in IP Networks: Foundations for a Multi-Service Internet*. Macmillan Technical Publishing, April 2000.

[2] M. Baldi. End-to-End Delay Analysis of VideoConferencing over Packet-Switched Networks. *IEEE/ACM Transactions on Networking*, 8(4), August 2000.

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

[4] G. Bolch, S. Greiner, K. Trevedi, and H. Meer. *Queueing Networks and Markov Chains - Modeling and Performance Evaluation with Computer Science Applications*. Jhon Wiley and Sons INC., 1998.

[5] C. Dovrolis and P. Ramanathan. A Case for Relative Differentiated Services and the Proportional Differentiation Model. *IEEE Network Magazine*, 1999.

[6] C. Dovrolis and D. Stiliadis. Relative Differentiated Services in the Internet: Issues and Mechanisms. In *Proc. of ACM SIGMETRICS'99*, 1999.

[7] C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional Differentiated Services: Delay Differentiation and Packet scheduling. In *Proc. of ACM SIGCOMM'99*, 1999.

[8] C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional Differentiated Services: Delay Differentiation and Packet Scheduling. *IEEE/ACM Transactions on Networking*, 10(1), February 2002.

[9] P. Sousa, P. Carvalho, and V. Freitas. End-to-End Delay Differentiation of IP Traffic Aggregates using Priority Queueing Models. In *Proc. of the 2002 IEEE Workshop on High Performance Switching and Routing (HPSR2002)*, pages 178–182, Kobe, Japan, May 26-28 2002.

[10] P. Sousa, P. Carvalho, and V. Freitas. Tuning Delay Differentiation in IP Networks using Priority Queueing Models. In E. Gregori et al, editor, *Proc.* 2^{nd} *International IFIP-TC6 Networking Conference*, pages 709–720, Pisa, Italy, May 2002. LNCS 2345, Springer-Verlag.

[11] P. Sousa, P. Carvalho, and V. Freitas. Scheduling Time-Sensitive IP Traffic. In G. Goos et al, editor, *Proc. 6th IFIP/IEEE International Conference on Management of Multimedia Networks and Services, MMNS*, pages 368–380, Belfast, Northern Ireland, September 2003. LNCS 2839, Springer-Verlag.

[12] P. Sousa, P. Carvalho, and V. Freitas. Enhancing Delay Differentiation Semantics of Class-based IP Networks. In Z. Mammeri and P. Lorenz, editors, *Proc.* 7^{th} *IEEE International Conference on High Speed Networks and Multimedia Communications, HSNMC*, pages 26–37, Toulouse, Fance, June/July 2004. LNCS 3079, Springer-Verlag.

[13] P. Sousa, P. Carvalho, and V. Freitas. A Multi-constrained QoS Aware Scheduler for Class-based IP Networks. In *Proc. of Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, pages 279–282, School of Electrical, Electronic and Computer Engineering, The University of Newcastle-upon-Tyne, UK, July 2004. ISBN: 0-7017-0177-3.

[14] S. Lima, P. Carvalho, and V. Freitas. A Distributed Admission Control Model for CoS Networks using QoS and SLS Monotoring. In *Proceedings of ICC'2003 - IEEE International Conference on Communications*, May 2003.

[15] B. Davie, A. Charny, J. C. R. Bennet, K. Benson, J. Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis. An Expedited Forwarding PHB (Per-Hop Behavior). *RFC 3246*, March 2002.

[16] R. Makkar, J. Salim, N. Seddigh, B. Nandy, and J. Babiarz. Empirical Study of Buffer Management Scheme for DiffServ Assured Forwarding PHB. *Proceedings of International Conference on Computer Communications and Networks*, 2000.

[17] I. Stoica, S. Shenker, and H. Zhang. Core-stateless fair queueing: Achieving approximately bandwidth allocations in high speed networks. In *Proc. of ACM SIGCOMM'98*, pages 118–130, 1998.

Pedro Sousa graduated in Systems and Informatics Engineering at the University of Minho, Portugal, in 1995 and obtained his MSc degree in Computer Science in 1997. In 1996, he joined the Computer Communications Group of the Department of Informatics at University of Minho, where he is a Lecturer. He is currently finishing his PhD in the area of scheduling mechanisms for Internet traffic.

Paulo Carvalho graduated in 1991 and received his PhD degree in Computer Science from the University of Kent at Canterbury, United Kingdom, in 1997. He is currently Assistant Professor of Computer Communications, Department of Informatics, at the University of Minho, Portugal. His main research interests include broadband technologies, multiservice networks and protocols, traffic characterization and modelling, and mobile networks.

Vasco Freitas graduated in 1972 and obtained his MSc and PhD degrees in Control and Computer Communications at the University of Manchester, UK, in 1977 and 1980. From 1989 until 1994 he was Director of Networking at the Portuguese Foundation for Scientific Computing to establish the National University Data Network and has since been a Professor of Computer Communications at the Universidade do Minho, Portugal. He is currently on a temporary appointment at the University of Algarve, Portugal.