# Improving the Quality and Reliability of Traffic Differentiation in IP Networks*

Pedro Sousa
Department of Informatics
University of Minho
4710-057 Braga, Portugal
pns@di.uminho.pt

Paulo Carvalho
Department of Informatics
University of Minho
4710-057 Braga, Portugal
pmc@di.uminho.pt

Vasco Freitas
Department of Informatics
University of Minho
4710-057 Braga, Portugal
vf@di.uminho.pt

**Abstract** – *This article presents a modular scheduling architecture for multi-QoS metric differentiation in class-based IP networks. The rationale of the supported differentiation modules is presented, highlighting the distinct differentiation semantics that might be used to control the delay, loss and rate metrics associated with the traffic classes. The devised modules resort to several relative and hybrid differentiation models to bound QoS metrics on high priority classes. In the proposed scheduling architecture, the differentiation modules may act jointly in order to control simultaneosly multiple QoS metrics. 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.*

**Keywords:** Quality and Reliability, Resource Management, Scheduling, Traffic Engineering, Quality of Service.

## 1 Introduction

The growth and diversity of distributed applications has fostered the need for IP networks with quality of service (QoS) 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. 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. Accomplishing this objective will *(i)* allow an intuitive and easily programmable human-network interface, *(ii)* enhance the IP differentiation semantics with multi-constrained QoS models and *(iii)* increase the applications' overall performance. In this context, and in opposition to traditional scheduling mechanisms, the proposed scheduler encompasses rate, loss and delay differentiation capabilities in a flexible way, resorting to new relative and hybrid differentiation models. This scheduler provides independent control of delay, loss and rate differentiation through the use of two priority disciplines acting at distinct points of the proposed scheduler architecture. The delay differentiation modules are based on theoretical schemes such as proportional differentiation [1, 2, 3, 4]. Other differentiation schemes [5, 6, 7, 8] are also supported by the scheduler, including an hybrid model specially devised for real-time traffic differentiation. These delay models aggregate a packet drop mechanism able to provide *(i)* loss differentiation *(ii)* rate allocation with distinct work-conserving behavior or *(iii)* combination thereof. The present scheduling proposal, being a modular and QoS aware traffic control mechanism, 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. The scheduler architecture has been implemented and tested in the network simulator (*NS-2*).

## 2 A Scheduling Architecture

To control multi-QoS metrics, the traffic scheduling architecture includes three distinct differentiation modules (see Fig. 1). 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 drop module acts as an input priority discipline and includes a set of distinct loss differentiation models. In addition, the packet drop module is able to control each class load inducing, over medium time scales, output rate differentiation. In the next sections the emphasis is given on the main objectives, definition and configuration modes of the differentiation modules.

*0-7803-8566-7/04/$20.00 © 2004 IEEE.

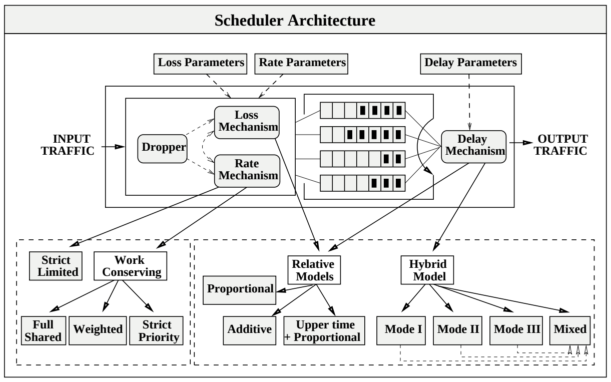


Figure 1: The scheduler architecture implemented in *NS-2*.

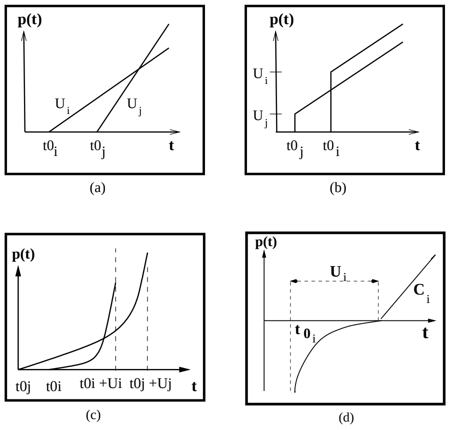


Figure 2: (a) Proportional model (b) Additive model (c) Upper time limit model (d) Hybrid model.

## 2.1 Delay Differentiation Models

The delay differentiation module comprises three basic relative models configured through delay parameters $(U_0,...,U_{N-1})$ associated with $N$ distinct traffic classes, having $Class_0$ the highest priority. For each model, the queue selection procedure is based on a priority function, $p_i(t)$, on which $t_{0_i}$ is the arrival time of the heading packet of $Class_i$. The delay module encompasses: *(i)* a proportional model, where proportional queuing delay relations among the classes are ruled by proportional relations of $U_i$ parameters (see Fig. 2(a) and Eq. (1)); *(ii)* an additive model, where high priority classes have a delay gain over low priority classes similar to the difference among $U_i$ parameters (see Fig. 2(b) and Eq. (2)); *(iii)* an upper time model able to bound the delay (reflected by the $U_i$ parameter) on high priority classes (see Fig. 2(c) and Eq. (3)). Through appropriate configuration, this model may also be used to bound the delay on the highest priority class and, simultaneously, to achieve proportional differentiation among the remaining traffic classes.

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

$$p_i(t) = (t - t_{0_i}) + U_i \qquad (2)$$

$$p_i(t) = \begin{cases} \frac{(t-t_{0_i})}{U_i - t + t_{0_i}} & if \;\; t < t_{0_i} + U_i \\ \infty & otherwise \end{cases} \qquad (3)$$

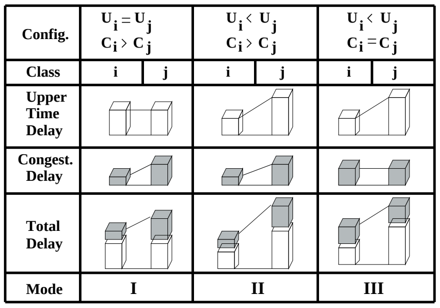


Figure 3: Configuration modes of the hybrid delay model.

An hybrid delay model specially devised for handling real-time traffic was also included in order to bound queuing delays (based on $U_i$) and, under congestion, differentiate the excess queuing delays using the congestion parameters $(C_0,...,C_{N-1})$ (see Fig. 2(d) and Eq. (4)). For instance, if an application has low capacity to absorb excess delays, the corresponding traffic class can be configured with a high congestion parameter. Three basic configurations are considered: Conf. I - similar $U_i$ and distinct $C_i$ parameters; Conf. II - distinct $U_i$ and $C_i$ parameters; Conf. III - distinct $U_i$ and similar $C_i$ parameters. Mixed configurations are also possible (Conf. I+II,II+III,I+III) depending on the semantics required (see Fig. 3).

$$p_i(t) = \begin{cases} \frac{\delta_t - U_i}{\delta_t} & \text{if } \delta_t < \mathrm{U}_i \\ (\delta_t - U_i) * C_i & \text{if } \delta_t \geq \mathrm{U}_i \end{cases} \qquad (4)$$

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

## 2.2 Loss Differentiation Models

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 $\Delta t$[1] and $A_{i,\Delta t}$ measures the number of packet arrivals of $Class_i$ in the same time interval. Using this reasoning, $l_{i,\Delta t}$ denotes the packet loss ratio experienced

[1]This means that $A_i$ and $drop_i$ counters are reinitialized periodically each $\Delta t$. 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 properly transient congestion periods of the network.

**Algorithm 1** Pseudocode of the loss differentiation module

```
[Event: $class_i$ packet arrival]: $A_i$++
[Event: $class_i$ packet drop]: $drop_i$++
[Event: $\Delta t$ 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
```

by $Class_i$ over the time period used to evaluate $drop_{i,\Delta t}$ and $A_{i,\Delta t}$ variables[2], as expressed by Eq. (5).

$$l_{i,\Delta t} = \frac{drop_{i,\Delta t}}{A_{i,\Delta t}} \qquad (5)$$

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[3] from a specific traffic class, accepting the newly arrived packet in the corresponding queue. 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 [9], 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 $\Delta t$ period, i.e. the candidate class for packet drop is kept unchanged during the following $\Delta t$. 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 Eq. (6), the additive loss differentiation by Eq. (7) and the upper bound loss model, defined by Eq. (8), allows to bound the packet loss ratios on high priority classes.

$$p_i(t) = (l_i) * L_i \qquad (6)$$

$$p_i(t) = (l_i) + L_i \qquad (7)$$

$$p_i(t) = \begin{cases} \frac{(l_i)}{L_i - (l_i)} & if \;\; l_i < L_i \\ \infty & otherwise \end{cases} \qquad (8)$$

## 2.3 Load Control and Rate Differentiation

Finally, the rate module, which is an alternative for ruling the packet drop behavior, allows to control each class load inducing, over medium time scales, output rate differentiation.

Consider that 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 ($\rho > 1$)[4] when the total class load at the input exceeds the output capacity of the link, $C$. This situation, illustrated in Eq. (9), leads to packet loss and to different levels of throughput share depending on the service discipline, class load and buffering resources.

$$\sum_{i=0}^{N-1} R\_in_i(t) > C \qquad (9)$$

$$\sum_{i=0}^{N-1} min(R\_in_i(t), R\_max_i) \leq C \qquad (10)$$

$$\sum_{i=0}^{N-1} R\_max_i \leq C \qquad (11)$$

The first step in the mechanism design assures that Eq. (9) 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 the input load of

[2]To simplify the notation, from now on we suppress the index $\Delta t$, i.e. $drop_i$, $A_i$ and $l_i$ represent variables in $\Delta t$.

[3]Or a group of packets, if a single drop is not sufficient to provide the buffer resources required to store the incoming packet.

[4]$\rho = \lambda \cdot \bar{S}$, $\lambda$ is the arrival rate and $\bar{S}$ the average service time.

$Class_i$ at time $t$ then Eq. (10) is valid, assuring that the server is always under a balanced state ($\rho \leq 1$)[5]. 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 Eq. (11). $R\_in_i(t)$ is estimated resorting to an adaptive exponential weighted moving average, Eq. (12), 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 packet inter arrival time. The parameter $T$ acts as a reference value and should have a similar order of magnitude of the time period for which the estimation module is expected to provide average rate information. 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 Eq. (13)) 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 $Credits(t)$[6].

$$R\_est_i = (1 - 2^{-\frac{\Delta t_i^k}{T}}) \cdot \frac{l_i^k}{\Delta t_i^k} + 2^{-\frac{\Delta t_i^k}{T}} \cdot R\_est_i^{old} \qquad (12)$$

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

$$Credits(t) = \sum_{i=0}^{N-1} credit_i(t) \qquad (13)$$

Within this work-conserving behavior, Eq. (14) 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 specific credit distribution function, $dist(t)$. The dropping mechanism associated with Eq. (14) is now ruled by Eq. (15) in order to assure a reactive response to load oscillations and redirect the unused bandwidth to the congested classes[7].

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

$$\sum_{i=0}^{N-1} min(R\_in_i(t), limit_i(t)) \leq C \qquad (14)$$

$$drop\_prob_i(t) = \begin{cases} 1 - \frac{limit_i(t)}{R\_in_i(t)} & if\ (R\_in_i(t) > limit_i(t)) \\ 0 & otherwise \end{cases} \qquad (15)$$

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[8]. The third 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.

Table 1: Distribution modes of server credits.

| Mode | Differentiation Equation |
|---|---|
| Full Shared | $limit_i(t) = R\_max_i + \frac{Credits(t)}{\sum congested}$ |
| Weighted | $limit_i(t) = R\_max_i + \frac{excess_i}{\sum_{j=0}^{N-1} excess_j} \cdot Credits(t)$ |
| Strict Priority | $limit_0(t) = R\_max_i + \min(excess_0, Credits(t))$ $limit_i(t) = R\_max_i + \min(excess_i, Credits(t) - - \sum_{j=0}^{i-1}(limit_j(t) - R\_max_j))(i > 0)$ |

## 3 Simulation Results

The following examples illustrate how the proposed scheduling architecture (Fig. 1) operates, showing its ability to decouple rate, loss and delay differentiation behavior, i.e. the differentiation mechanisms might act jointly but, simultaneously, might provide independent QoS metric differentiation[9].

### 3.1 Rate vs. Delay Differentiation

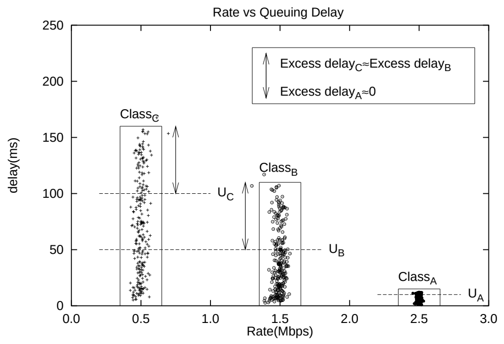


Figure 4: Rate differentiation with hybrid delay model (Conf. II+III) for ($R\_max_A$,$R\_max_B$,$R\_max_C$) = (2.5$Mbps$,1.5$Mbps$,0.5$Mbps$), ($U_A$,$U_B$,$U_C$) = (10$ms$,-50$ms$,100$ms$) and ($C_A$,$C_B$,$C_C$) = (20,1,1).

[5]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 $Class_i$.

[6]The boolean variable, $cong_i$, is true if $R\_in_i(t) \geq R\_max_i$.

[7]Relaxed versions of the rate differentiation module are possible. For instance, this mechanism may operate only during specific probing periods or the $limit_i$ and $drop\_prob_i$ values are only computed for specific time intervals, despite the class rate estimation being continuously updated.

[8]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 $excess_i$.

[9]Due to the high number of possible differentiation schemes only specific configuration modes are covered. The selected examples were taken from a scenario where three classes contend for a 4.5$Mbps$ link, with packet sizes uniformly distributed over $[250, 750]$ bytes. The scheduler was tested successfully for traffic sources such as CBR, exponential, pareto, for a simulation period of 120$s$ with a QoS metric evaluation interval of 1$s$ and for an overall load above the link capacity to force packet loss.

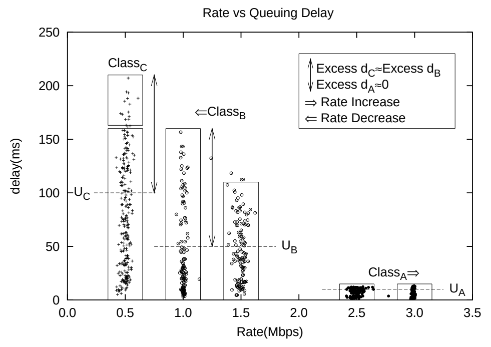


Figure 5: Strict priority rate with hybrid delay model (Conf. II+III) for ($R\_max_A$,$R\_max_B$,$R\_max_C$) = (2.5$Mbps$,1.5$Mbps$,0.5$Mbps$), ($U_A$,$U_B$,$U_C$) = (10$ms$,50$ms$,100$ms$) and ($C_A$,$C_B$,$C_C$) = (20,1,1).

*Strict Priority Rate Model with Hybrid Delay*- This example illustrates the operation of the hybrid delay differentiation model (Conf. II+III) and the rate differentiation module for a configuration with ($R\_max_A$,$R\_max_B$,$R\_max_C$) = (2.5$Mbps$,1.5$Mbps$,0.5$Mbps$), ($U_A$,$U_B$,$U_C$) = (10$ms$,50$ms$,100$ms$) 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 absorb excess delays despite having different upper time delays. Fig. 4 shows the average output rate and queuing delays obtained by the classes, clearly corroborating the expected differentiation behavior. Fig. 5 illustrates a similar delay differentiation, now with strict priority rate differentiation. Fig. 5 plots the differentiation behavior when $Class_B$ decreases its rate to 1$Mbps$. As shown, only $Class_A$, which has the highest priority, has assigned extra bandwidth (shift to the right side of the graph), exactly the 0.5$Mbps$ 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 0.5$Mbps$ 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.

## 3.2 Loss vs. Delay Differentiation

*Proportional Loss and Hybrid Delay*- In this example 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 5$ms$. In addition, the congestion delays of $Class_C$, i.e. the difference between the

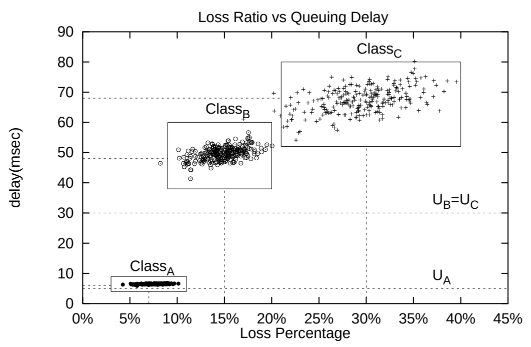
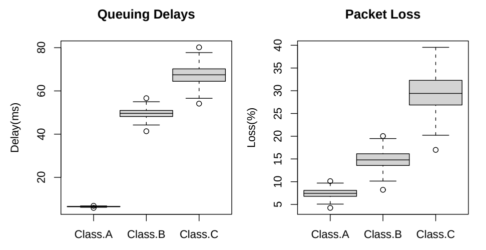


Figure 6: Proportional loss and hybrid delay model for (Conf. I+II), with ($U_A$,$U_B$,$U_C$) = (5$ms$,30$ms$,30$ms$), ($C_A$,$C_B$,$C_C$) = (40,2,1), ($L_A$,$L_B$,$L_C$) = (4,2,1).



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 7: Box-Whisker plots and Student's t-Test for the mean values of delay and loss metrics (of Fig. 6).

obtained delays and the target delay of 30$ms$, should be twice the congestion delay of $Class_B$, which has a similar delay target of 30$ms$, but a congestion parameter two times higher than $Class_C$. This behavior is illustrated in Fig. 6. Fig. 7 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.

## 3.3 Rate vs. Loss vs. Delay Differentiation

*Rate Differentiation with Additive Loss and Upper Time Delay*- This example illustrates the three differentiation modules acting together. We assume that $Class_A$ is used for loss and time sensitive traffic, being its bandwidth limited at network edges to 2$Mbps$. $Class_B$ 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 per-

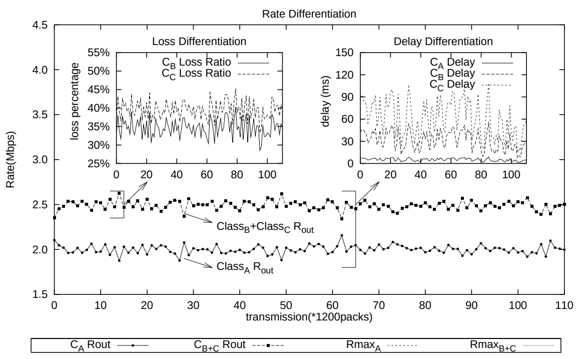
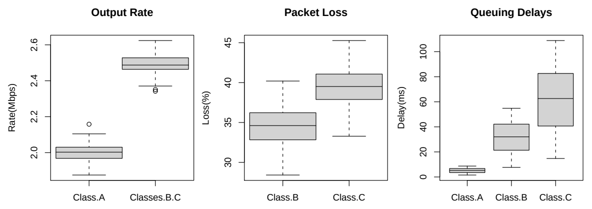


Figure 8: Rate differentiation with additive loss and upper time delay model with $(R\_max_A, R\_max_{B+C}) = (2Mbps, 2.5Mbps)$, $(L_B, L_C) = (0.05, 0)$, $(U_A, U_B, U_C) = (10ms, 100ms, 200ms)$.



| Student's T-test (95%) | | Student's T-test (95%) | | | Student's T-test (95%) | | |
|---|---|---|---|---|---|---|---|
| **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 9: Box-Whisker plots and Student's t-Test for the mean values of delay, rate and loss metrics (of Fig. 8).

centage 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. 8, 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. 8 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. 9. This example proves that, despite being easily configurable, the scheduling architecture has a powerful differentiation semantics to improve QoS capability of network nodes.

## 4 Conclusions

This article has presented a multi-constrained QoS scheduler to improve the quality and reliability of class-based IP nodes. The devised modular scheduling architecture includes three differentiation modules providing enhanced differentiation semantics for loss, delay and rate QoS metrics. The underlying concept of each differentiation module was presented, focusing on the distinct configurations supported by the proposed architecture. The robustness and efficiency of the differentiation mechanisms were also corroborated resorting to simulation. The results prove that the differentiation modules may operate jointly and are able to provide multi-QoS metric differentiation in class-based networks. The diversity of the supported QoS differentiation semantics, along with the simplicity and intuitive nature of the configuration tasks, turn the presented mechanisms into 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.

## References

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

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

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

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

[5] P. Sousa, P. Carvalho, and V. Freitas. End-to-end delay differentiation of IP traffic aggregates using priority queueing models. In *Proc. of IEEE HPSR 2002*, pages 178–182, Kobe, Japan, May 2002.

[6] P. Sousa, P. Carvalho, and V. Freitas. Tuning delay differentiation in IP networks using priority queueing models. In E. Gregori et al, editor, *Proc. of* $2^{nd}$ *IFIP-TC6 Networking Conference*, pages 709–720, Pisa, Italy, May 2002. LNCS 2345, Springer-Verlag.

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

[8] P. Sousa, P. Carvalho, and V. Freitas. Enhancing Delay Differentiation Semantics of Class-based IP Networks. In *Proc. of IEEE International Conference on High Speed Networks and Multimedia Communications (HSNMC)*, Toulouse, France, June 2004. LNCS 3079, Springer-Verlag.

[9] R. Makkar et al. Empirical Study of Buffer Management Scheme for DiffServ Assured Forwarding PHB. *Proc. of International Conference on Computer Communications and Networks*, 2000.