Scheduling Time-Sensitive IP Traffic

Pedro Sousa, Paulo Carvalho, and Vasco Freitas

Universidade do Minho, Departamento de Informática,
4710-057 Braga, Portugal
email: {pns,paulo,vf}@uminho.pt

Abstract. This article presents an hybrid priority queuing model based scheduler for real-time traffic differentiation. The proposed scheduler is designed as a mechanism to provide queuing delay differentiation among real-time traffic classes. The novel characteristic of the presented scheduler is the capability to simultaneously try to achieve an upper time limit for queuing delays and, under heavy load conditions, deny class starvation by providing an expectable differentiation schema for excess queuing delays. The attractiveness of the proposed scheduler is its hybrid differentiation capabilities based on a simple queue selection procedure. Additionally, the enhanced differentiation behavior of the scheduler is also highlighted as three distinct configuration modes are possible.

1 Introduction

The advent of Quality of Service (QoS) in the Internet [1] is being fostered by an increasing need to provide adequate network services to a vast range of QoS-demanding applications. In the presence of distinct applications and traffic profiles, service providers need to differentiate customers so that an efficient and cost-effective network resource management can be achieved. In addition to admission control [2], reservation protocols [3], resource management solutions [4, 5], or even when these are not present, acceptable QoS conditions can be obtained in the presence of an appropriated delay differentiation mechanism. In fact, delay differentiation can be extremely useful to integrate real-time applications and other delay sensitive applications. From the applications' perspective there are two crucial aspects for the integration of real-time applications in IP networks: the ability to satisfy end-to-end delay requirements and the capability to absorb excess delays. The former aspect is related to real-time applications, such as voice over IP and other interactive applications which are highly delay sensitive [6]. In this context, the deployment of scheduling mechanisms providing queuing delay bounds plays a crucial role in the integration of real-time traffic in IP networks. The latter aspect is related to mechanisms used by real-time applications in order to smooth excess delays [7, 8] or to adapt to network conditions [9, 10]. This means that in addition to end-to-end delay bounds it is useful to provide expectable differentiation mechanisms to handle excess queuing delays inside the network. In this context, the use of rigid admission control procedures and resource reservations protocols (e.g. in the IntServ architecture [11]) play a relevant role. These solutions, suffering from well known lack of scalability and flexibility, led to lighter and easier to deploy solutions (e.g. DiffServ [12]). As expected, relaxing QoS-guarantees in the network raises

additional problems of integration of real-time applications mainly due to the absence of rigid admission control in the core routers. Even existing admission control at network edges the network core may cause feasibility problems to schedulers due to: *i)* the transient effect caused by dynamic flow aggregation; *ii)* traffic distortion and packet clustering caused by cascade queuing effects; *iii)* possible shaping inaccuracies at edge routers and *iv)* path changes caused by a route flip. In addition, delay-oriented scheduling mechanisms may show feasibility problems as regards their configuration parameters. Service starvation for low priority classes may also occur in some schedulers using strict priority schemes for class differentiation [13]. In conclusion, to contribute for a QoS-capable Internet it is important to develop scheduling mechanisms able to react to particular congestion situations in order to achieve expectable differentiation behaviors among real-time traffic classes. In this context, this work proposes a new hybrid scheduling mechanism with the following characteristics: *i) easy configuration*- it is possible to control the *expectable queuing delay* and *congestion queuing delay* of each traffic class through simple parameter configuration; *ii) simplicity*- the hybrid behavior of the differentiation mechanisms is obtained resorting to simple queue selection procedures, and not using additional node state information (e.g. arrival class packet counters, long-term average delay counters, etc.); *iii) enhanced differentiation capability*- the scheduler allows three configuration modes each one involving distinct differentiation semantics (mixed configuration modes are also possible); *iv) unfeasible scheduling regions control*- the differentiation mechanism also contributes to the control of unfeasible working regions of scheduling mechanisms by providing controllable and expectable differentiation schema for delay deviations. The proposed scheduler can be used in distinct operational scenarios independently of the network model. Nevertheless, and due to its delay-oriented nature, a *rate-oriented* mechanism might be used to allocate/differentiate bandwidth between elastic and real-time traffic, and within the latter, the proposed mechanism provides for delay differentiation.

The paper is organized as follows: Section 2 presents related work. Section 3 details the proposed queuing model. Section 4 presents simulation results of the scheduler behavior including: single-node differentiation, flow granularity, results verification and mixed configurations. Section 5 presents the conclusions of the work.

2 Related Work

Recently, there has been considerable research focusing on the use of priority queuing models for IP traffic differentiation. Some of these queuing models can be found in [14], where brief mathematical explanations are also given. The work presented in [15–18] focuses on the use of Relative Differentiation, where a multiplicative time dependent model is used to achieve proportional differentiation behavior of a network node. Additionally [19, 20] study some possible adaptive behaviors which can be applied to the previous models. In [21, 22] an overview of different delay differentiation schemas including proportional, additive and an hybrid upper-time queuing model are presented. The latter model allows the coexistence of the proportional model along with an unique upper time bounded traffic class. These contributions include end-to-end differentiation analysis as well as individual flow behaviors study. Although the work in

the area focuses mainly on the Relative Differentiation approach, such as proportional differentiation, the present proposal focuses on an hybrid queuing model of traffic differentiation. Some different schemas, such as EDD [23] (also denoted as EDF), also try to limit queuing packets delays but they are more suitable for scenarios of strong per node admission control procedures in order to ensure the necessary feasibility conditions [24] for their operations. Instead the mechanism proposed here is more adequate for scenarios where admission control procedures are more relaxed and operate at network edges devices. In this context, in [25] the EDD schema is modified in order to differentiate the probability of queuing delay violations under a congested network. Furthermore, is fundamental to differentiate the relative value of such violations, i.e. under general class congestion ensure that the excess queuing delays of high priority classes are not higher than the obtained by lower ones. Additionally, the present mechanism is based on a simple queuing selection procedure which does not use additional class state information, reducing the differentiation node complexity.

3 An Hybrid Priority Queuing Model

3.1 Model Construction

The Upper Time Limit (UTL) model belongs to the class of Priority Queuing models [14] where each queue is ruled by a priority function that varies over time (Time-Dependent Priorities). The nature of the priority function and its configuration parameters define the behavior of the service assigned to each queue. This study considers N classes $Class_{i(0 \leq i \leq N-1)}$ having $Class_0$ the highest priority. The UTL model is a more rigid schema than the additive and proportional models as it imposes a finite queuing delay. The main idea is to define a boundary (reflected by U_i) for the packet queuing time (Eq. (1)). In this model, the lower the boundary, the higher the priority function slope will be. When $t - t_{0_i} \geq U_i$, i.e. on or over the limit, the server is *forced*[1] to dispatch the packet awaiting service[2]. This model protects the high priority classes aiming that packets remain in queue for a maximum value U_i with $U_i < U_{i+1}$ (Fig. 1(a)).

$$p_i^a(t) = \begin{cases} \dfrac{t - t_{0_i}}{U_i - t + t_{0_i}} & \text{if } t < t_{0_i} + U_i \\ \infty & \text{if } t \geq t_{0_i} + U_i \end{cases} \quad (1)$$

Fig. 2 illustrates two examples of the model behavior for three CBR and Exponential sources[3] contending for a common 100Mbps capacity link and for specific U_i parameters. Within this model some additional considerations can be made regarding its behavior: *i)* as seen in Fig. 2, and as expected, under heavy load conditions the *upper time* parameters of the classes can be violated; *ii)* under a violation it is not possible to control the spread of excess queuing delay; *iii)* starvation of low priority classes may occur (e.g. $Class_C$ between 100-150 server transmission times). This shows that the model is not able to distinguish excess queuing delays in the traffic classes. The reason for this behavior is that when a class violates the respective upper time limit, the

[1] Obviously when congestion occurs packets can be dropped or the waiting time limit exceeded.
[2] t_{0_i} is the arrival time of the heading packet of $Class_i$.
[3] The results were obtained implementing the native UTL model in the *Network Simulator-2*.

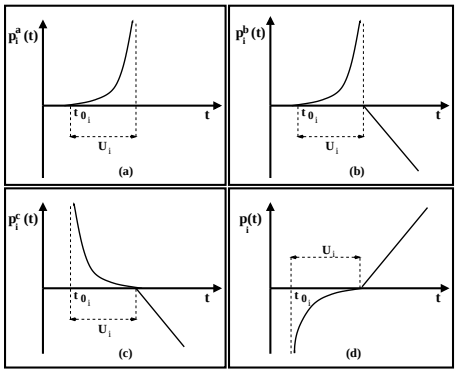

Fig. 1. a) Native UTL model b) c) d) Different stages leading to the hybrid queuing model.

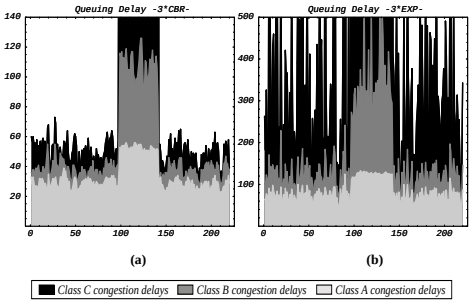

Fig. 2. a) Three CBR classes $(U_A, U_B, U_C) = (30\mu s, 40\mu s, 50\mu s)$ b) Three exponential classes $(U_A, U_B, U_C) = (100\mu s, 150\mu s, 250\mu s)$.

priority function assumes an infinity value and from that instant the system acts as a strict priority model. Furthermore, it is common that under a high delay violation of a class the other classes also become overloaded due to starvation and, as consequence, all the priority functions assume an infinite value (i.e. a cascade effect). This means that it is useless to use the priority values to differentiate classes as the decision is based again on static assumptions (as to serve the highest priority class first). These cascade effects are prominent in high speed networks dealing with high traffic loads and strict bounds for queuing delays. In order to overcome these problems, an additional mechanism allowing class differentiation in such scenarios is needed. The first modification introduced in the original model was to activate an additional function for $t \geq t_{0_i} + U_i$. The idea was to allow congested classes to be differentiated instead of using infinity values hindering the initial objective. Moreover, this additional function should assume priority values allowing the scheduler to differentiate the congested situation from the normal priority function behavior for $t < t_{0_i} + U_i$. This is achieved using a multiplicative function for $t \geq t_{0_i} + U_i$ as presented in Eq. (2) (see Fig. 1(b)). This function, as explained later, will allow the *proportional differentiation* of excess queuing delays between traffic classes. The excess queuing delay (the difference of the total and upper time delay, i.e. $t - t_{0_i} - U_i$) is multiplied by a scale parameter C_i which guides the priority function behavior. In this region the priority function assumes negative values.

$$p_i^b(t) = \begin{cases} \dfrac{t - t_{0_i}}{U_i - t + t_{0_i}} & \text{if } t < t_{0_i} + U_i \\ -(t - t_{0_i} - U_i) * C_i & \text{if } t \geq t_{0_i} + U_i \end{cases} \quad (2)$$

The drawback of Eq. (2) is the complexity of the queue selection procedure. In fact, it will be necessary to compute all $p_i^b(t)$ values; verify if a negative value exists; if so, select the lowest one, otherwise select the highest positive priority value. To simplify the selection procedure Eq. (3) was applied to Eq. (2) resulting in Eq. (4). In this modified

function the selection procedure consists of a simple selection of the queue with the lowest $p_i^c(t)$ value as the one to be served next by the scheduler (see Fig. 1(c)).

$$p_i^c(t) = \frac{1}{p_i^b(t)} \quad for \quad t < t_{0_i} + U_i \tag{3}$$

$$p_i^c(t) = \begin{cases} \frac{U_i - t + t_{0_i}}{t - t_{0_i}} & \text{if } t < t_{0_i} + U_i \\ -(t - t_{0_i} - U_i) * C_i & \text{if } t \geq t_{0_i} + U_i \end{cases} \tag{4}$$

In order to maintain the normal semantics of priority queuing, where the class with the highest priority function is selected first, the symmetric function (5) is used as the normalized priority queuing function. This model is then configured with two distinct sets of parameters: *Upper time* $(U_0, .., U_{N-1})$ and *Congestion*[4] parameters $(C_0, .., C_{N-1})$. The final priority function is given by (6) and Fig. 1(d) illustrates its behavior.

$$p_i(t) = -p_i^c(t) \tag{5}$$

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

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

In order to define the selection task, let C be the set of all (i, p_i) pairs in the system, where p_i is the priority to serve $Class_i$, i.e. $C = \{(i, p_i) \mid 0 \leq i \leq N - 1\}$. The selector Sel, defined according Eq. (7) and (8), determines the index of the class to be served, i.e. taking the maximum priority value, p_{max}, the corresponding minimum i is chosen. The total delay[5], d_i, affecting $Class_i$ can be divided in two components: one induced by priority function when it assumes negative values, i.e. $t < t_{0_i} + U_i$, which we call *upper time delay*, d_i°, and other when the function assumes positive values, which we call *congestion delay*, $d_i^\bullet$ (see Eq. (9)). The magnitude of d_i° is controlled by the *upper time parameter*, 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 parameters U_i and C_i. The next section discusses this aspect.

$$p_{max} = max\{y \mid (x, y) \in C\} \tag{7}$$

$$Sel = min\{x \mid (x, y) \in C \wedge y = p_{max}\} \tag{8}$$

$$d_i = d_i^\circ + d_i^\bullet \tag{9}$$

3.2 Parameter Configuration Modes

Fig. 3(a) presents three distinct behaviors of the hybrid queuing model (configuration mode I, II and III). For each configuration, the relations between the *upper time delay*

[4] We use the *congestion* term 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.

[5] In the remaining of the paper, d_i is also used to denote the average queuing delay of $Class_i$.

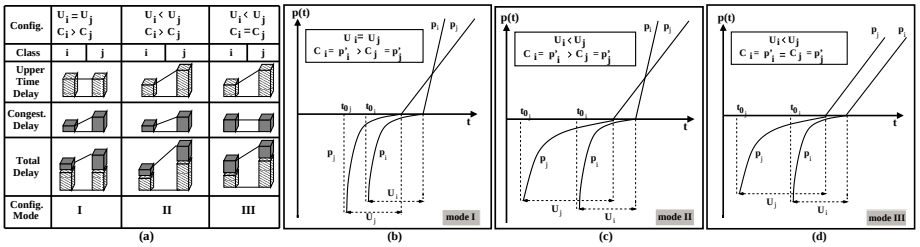

Fig. 3. (a) Combinations of U_i, C_i parameters (b)(c)(d) Representation of configuration modes.

and *congestion delay* are presented for two generic classes i and j with $i < j$. As shown, the *total delay* differentiation behavior for all configuration modes obeys the *Relative Differentiation* relation: $d_0 \leq d_1 \leq ... \leq d_{N-1}$.

Configuration I: In this configuration mode 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. Fig. 3(b) illustrates the priority function evaluation in this configuration mode for traffic classes i and j, with the assumption of heading packet time arrivals $t_{0_i} > t_{0_j}$. As shown, identical priority function shapes for both classes are obtained for $t < t_{0_i} + U_i$. On the other hand, in the positive priority function region, the function slope C_i is greater than function slope C_j eventually leading to the expected switch in priority values for these queues. This configuration mode may be appropriated for real-time classes with the same upper time limit for queuing delay and distinct capabilities to absorb 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 maximum difference between the queuing delays is given by (10)[6] and (11). Recall that within this configuration mode and due to similar *upper time* configurations, the queuing delay difference of *congested* classes is only influenced by the C_i parameters.

$$d_j - d_i = d_j^\bullet - d_i^\bullet \approx \frac{C_i}{C_j} \cdot d_i^\bullet - d_i^\bullet \tag{10}$$

$$d_j - d_i \approx \underbrace{d_i^\bullet \cdot \left(\frac{C_i}{C_j} - 1\right)}_{congestion\ part} \tag{11}$$

Configuration II: In this configuration mode the traffic classes are distinct as regards *upper time differentiation parameters* and *congestion differentiation parameters*. As result, the priority function associated with higher priority classes has a larger in-

[6] Note that for the proportional model $\frac{d_j^\bullet}{d_i^\bullet} \approx \frac{C_i}{C_j} \Rightarrow d_j^\bullet \approx d_i^\bullet \cdot \frac{C_i}{C_j}$.

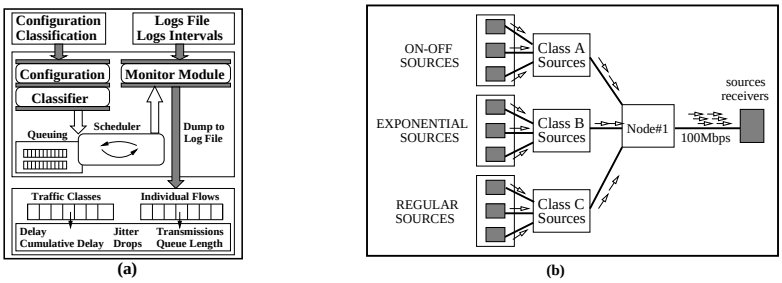

Fig. 4. (a) Components implemented in each output link (b) Simulation scenario.

crease 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 absorb excess queuing delays. Fig. 3(c) presents an example of the priority function evaluation in this configuration model for two classes. Again, if the server becomes overloaded and under *upper time limits* violations, the delay differentiation is given by (12) and (13). As presented by Eq. (13) the delay difference has two components: one resulting from the U_i parameters and the other from the C_i parameters, having these ones the same characteristics as in configuration mode I.

$$d_j - d_i = (d_j^{\circ} - d_i^{\circ}) + (d_j^{\bullet} - d_i^{\bullet}) \tag{12}$$

$$d_j - d_i \approx \underbrace{U_j - U_i}_{upper\ time\ part} + \underbrace{d_i^{\bullet} \cdot \left(\frac{C_i}{C_j} - 1\right)}_{congestion\ part} \tag{13}$$

Configuration III: This mode differentiates traffic classes only by U_i parameters. This configuration is used to distinguish a class by its maximum queuing delay limit and, in case of violation, the classes share the same priority behavior for the excess queuing delays (see Fig. 3(d)). The delay differentiation achieved by this model is presented in Eq. (14). As expected the differentiation is caused only by U_i as C_i and transitively the behavior of $p_i(t)$ are the same for both classes and for $t \geq t_{0_i} + U_i$.

$$d_j - d_i = (d_j^{\circ} - d_i^{\circ}) \approx \underbrace{U_j - U_i}_{upper\ time\ part} \tag{14}$$

4 Performance Evaluation

The differentiation mechanisms were implemented and tested in the *Network Simulator (NS-2)*. Specific queues and monitors were also developed in order to collect results from the tests. Fig. 4(a) shows the implemented architecture associated with the output link of a differentiation node. At Otcl level, the scheduler is selected, the differentiation parameters of the queues/classes are defined and classification data is provided, i.e.

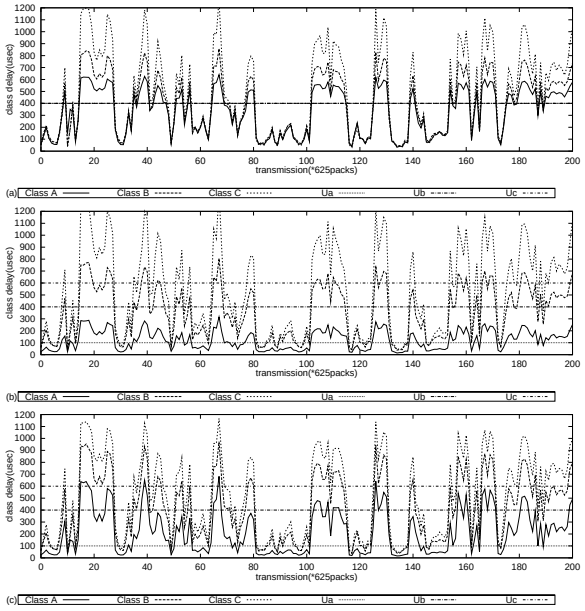

Fig. 5. Delay differentiation for a) $(U_A, U_B, U_C) = (400\mu s, 400\mu s, 400\mu s)$, $(C_A, C_B, C_C) = (4, 2, 1)$ (mode I) b) $(U_A, U_B, U_C) = (100\mu s, 400\mu s, 600\mu s)$, $(C_A, C_B, C_C) = (4, 2, 1)$ (mode II) c) $(U_A, U_B, U_C) = (100\mu s, 400\mu s, 600\mu s)$, $(C_A, C_B, C_C) = (1, 1, 1)$ (mode III).

$(packet_{flowid}, queue_{id})$ 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 and classes for subsequent analysis. Fig. 4(b) shows the simulation scenario used to test the scheduler behavior. In this example different traffic patterns were mixed converging to the differentiation nodes. The scenario includes *on-off*[7], exponential and isochronous traffic sources which are mapped to different classes (A, B and C) contending for a common link. Each class contributes evenly to the overall load, i.e., have similar long term rates, and generates mean packet lengths of 500 bytes uniformly distributed over $[250, 750]$. Similar queuing resources were allocated for all classes and $Class_A$ has the highest priority[8].

4.1 Single-node Differentiation

Fig. 5 shows three distinct differentiation examples obtained using the scheduler in configuration modes I, II and III. The results are presented graphically where the x-axis represents the server packet transmission times with a plot granularity of $25ms$.

Fig. 5(a) shows a configuration where all classes share an upper time limit of $400\mu s$, i.e. $(U_A, U_B, U_C) = (400\mu s, 400\mu s, 400\mu s)$, and distinct *Congestion Differentiation*

[7] On-off periods follow a Pareto distribution with $\alpha = 1.2$.

[8] The reason for this choice is that this class carrying Pareto related traffic causes high variability on queue lengths being more demanding on the differentiation algorithm.

Parameters, in this case $(C_A, C_B, C_C) = (4, 2, 1)$. As plotted, all classes have similar queuing delays in the non-congested scheduling region $(d_A, d_B, d_C \leq 400\mu s)$. However, in the congested regions the scheduler switches to proportional differentiation. As a result when the queuing delays are higher than $400\mu s$ the excess queuing delays on $Class_A$ are lower than those on $Class_B$, and both lower than $Class_C$ delays. Additionally, the proportional relation between the excess queuing delays may be easily visualized, as excess delays in $Class_C$ are approximately twice the delays in $Class_B$, which in turn double $Class_A$ delays. Consequently $Class_C$ excess delays are four times longer than in the highest priority $Class_A$. This satisfies the proportional relations defined by the *Congestion Differentiation Parameters*. These results show that the proposed behavior for the configuration mode I is feasible and can be used by classes sharing the same queuing delay constraints but with different capabilities to absorb excess queuing delays.

Fig. 5(b) plots the differentiation behavior for configuration mode II. In this case different upper time limits are assigned to each class, $(U_A, U_B, U_C) = (100\mu s, 400\mu s, 600\mu s)$. All classes are also differentiated by the *Congestion Differentiation Parameters* with $(C_A, C_B, C_C) = (4, 2, 1)$. As a result, as plotted, for congested periods there is an excess queuing delay in all congested classes following the proportional differentiation approach. For example, when the upper time of the highest class is violated ($d_A > 100\mu s$) the remaining queuing delay is approximately two times lower than the obtained by $Class_B$ (relative to its upper time of $400\mu s$). The same applies to relations between $Class_A$ and $Class_C$ and to $Class_B$ and $Class_C$. This configuration is useful when aiming of a fully differentiation schema, which means that highest classes have lower upper time delays and simultaneously are more sensitive to excess queuing delays, resulting in a higher *Congestion Differentiation Parameter*.

Fig. 5(c) illustrates the differentiation behavior for configuration mode III. In this case, different upper time limits are assigned to each class, $(U_A, U_B, U_C) = (100\mu s, 400\mu s, 600\mu s)$, but classes are not differentiated by the *Congestion Differentiation Parameters* as $(C_A, C_B, C_C) = (1, 1, 1)$. As a result, and as illustrated, for congested periods there is an excess queuing delay in all congested classes following a fair distribution. For example, when the upper time of the highest class is violated ($d_A > 100\mu s$) the remaining of the queuing delay is similar to the obtained by $Class_B$ (relative to its upper time of $400\mu s$). The same applies to relations between $Class_A$ and $Class_C$ and to $Class_B$ and $Class_C$. This configuration is useful for partial differentiation schemas, which means that higher priority classes have lower upper time delays but simultaneously have similar sensitivity to excess queuing delays as the lower priority classes.

4.2 Flow Granularity and Fairness

In this section the model behavior is studied at the flow level. The aim is to examine how the delay-oriented QoS offered to each class is extended to the flow level. Knowledge of the flow characteristics can be useful to check whether applications can also expect a fair delay differentiation as a complement to the class differentiation behavior of the model. Furthermore, it is expected that the scheduler provides per flow queuing delay consistency. In other words, it is expected that flows sharing a common traffic class at a given time also share identical average queuing delays. To check this, two flows

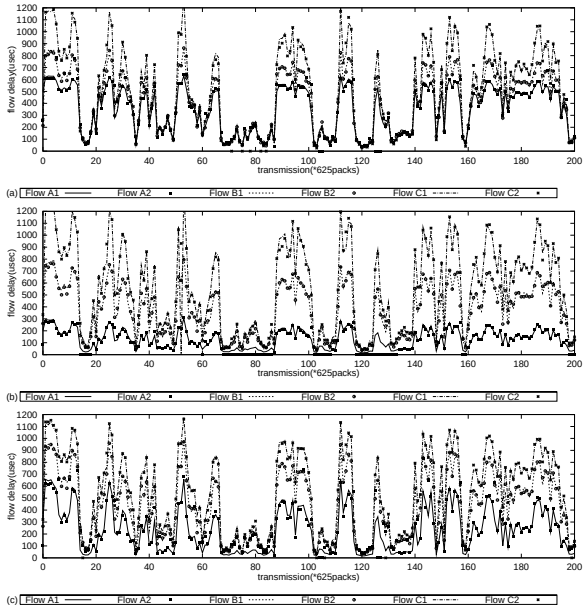

Fig. 6. Differentiation at Flow Level a) Configuration I b) Configuration II c) Configuration III.

of each traffic class were selected and their queuing delay behavior plotted together as shown in Fig. 6. It can be seen that the differentiation behavior is also valid at flow level, i.e. delay violations at flow level behave similarly to their traffic classes. Furthermore, the differentiation mechanism keeps per flow queuing delay consistent with their traffic classes having the same queuing delay. Inspection of Fig. 6 (a),(b) and (c) shows that the plots of delay per flow coincide within each class for all configurations.

4.3 Verification of Results

In this section we verify if the results obtained by simulation satisfy the delay difference equations devised in Sec. 3.2, representing the maximum queuing delay spread which may occur for each configuration mode under heavy load conditions. Fig. 7 plots the current delay differences and the maximum theoretical differences expressed by equations (11), (13) and (14), for each configuration mode. The latter are represented by lines, during the congested periods only, and the former represented by dots. Only the differences $d_B - d_A$ and $d_C - d_A$ are plotted. It can be seen that in congested periods, i.e. when the lines show up, the plots of delay differences follow the lines, thus confirming that the scheduler is performing appropriately. In fact, during the congestion periods the dots approximately follow the lines and for uncongested periods the observed differences are smaller. This corroborates equations (11), (13) and (14).

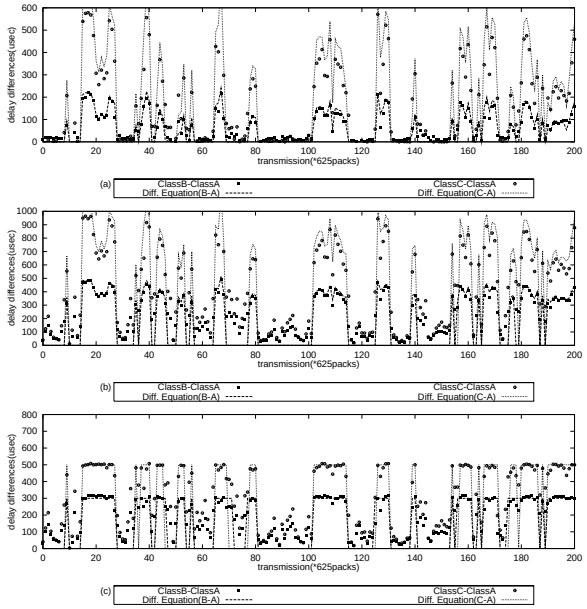

Fig. 7. Delay differences compared to maximum values given by a) Eq. (11) for configuration I b) Eq. (13) for configuration II c) Eq. (14) for configuration III.

4.4 Mixed Configurations

The discussion so far has focused on the scheduler operating under three specific configuration modes, however, it also supports mixed configurations, i.e. any subset of the traffic classes can be configured under a distinct mode. This point has major relevance in the scheduling area because it allows supporting a number of differentiation requirements. The example in Fig. 8(a) corresponds to an hybrid configuration involving configuration mode I and II. $Class_B$ and $Class_C$ are configured under configuration mode I, i.e. with the same U_i and different C_i parameters, whereas $Class_A$ and the other classes follow configuration mode II, i.e. different U_i and C_i parameters. As result, a mixed differentiation behavior is obtained. $Class_B$ and $Class_C$ share a common upper time constraint of $500\mu s$ and due to different congestion parameters, 2 and 1 respectively, suffer different delay violations. $Class_A$ is protected by a lower upper time constraint of $100\mu s$ and a much higher congestion parameter of 20 which causes very low violation values for this class. This example illustrates a possible configuration for a class with a behavior similar to EF PHB [26], consisting of a low latency and jitter[9] traffic class. The second example in Fig. 8(b) corresponds to mixed configuration of modes II and III. $Class_B$ and $Class_C$ are configured in mode III with upper time constraints of $500\mu s$ and $700\mu s$ and the same sensitivity to delay violations through identical congestion parameters. $Class_A$ and the other classes follow configuration mode II, as this class has different upper time and congestion parameters. Again,

[9] $Class_A$ delay oscillations are very low which means that jitter also assumes a low value.

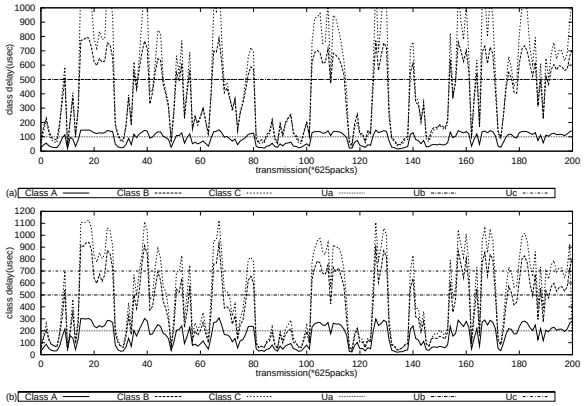

Fig. 8. a) $(U_A, U_B, U_C) = (100\mu s, 500\mu s, 500\mu s)$, $(C_A, C_B, C_C) = (20, 2, 1)$ (Conf. I+II) b) $(U_A, U_B, U_C) = (200\mu s, 500\mu s, 700\mu s)$, $(C_A, C_B, C_C) = (5, 1, 1)$ (Conf. II+III).

a different mixed behavior can be seen in Figure 8(b). When $Class_B$ and $Class_C$ are under congestion they suffer identical delay violations while $Class_A$ due to its higher congestion parameters achieves a delay violation which can be as lower as five times the obtained by the other classes. These examples illustrate the configuration capabilities of the scheduler which is able to achieve mixed differentiation behaviors.

5 Conclusions

This study proposes an hybrid queuing model to provide real-time traffic differentiation. The novel characteristic of the presented scheduler is the capability to simultaneously achieve an upper time limit for queuing delays and, under heavy load conditions, deny class starvation by providing an expectable differentiation schema for excess queuing delays. Through simple configuration is possible to control the expectable queuing delay and the congestion queuing delay of each traffic class. The hybrid behavior of the differentiation mechanism is obtained resorting to simple queue selection procedures, without using additional node state information. The scheduler allows three configuration modes each one involving distinct differentiation semantics. Mixed configurations modes are also possible, which improve the differentiation semantics of the proposed mechanism. Moreover, the differentiation achieved by the scheduler is fair at flow level, obeys the corresponding theoretical formulation and has reduced impact on the node complexity.

References

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

2. S. Lima and P. Carvalho and A. Santos and V. Freitas. A Distributed Admission Control Model for CoS Networks using QoS and SLS Monitoring. In *ICC'2003 - IEEE International Conference on Communications*, May 2003.
3. R. Braden et al. Resource reservation protocol (rsvp). *RFC2205*, Sep. 1997.
4. I. Stoica and H. Zhang. Providing guaranteed services without per flow management. In *Proc. of SIGCOMM'99*, 1999.
5. Z. Zhang et al. Decoupling qos control from core routers: A novel bandwidth broker architecture for scalable support of guaranteed services. In *Proc. of SIGCOMM'00*, 2000.
6. M. Baldi. End-to-end delay analysis of videoconferencing over packet-switched networks. *IEEE/ACM Trans. on Networking*, 8(4), Aug. 2000.
7. W. E. Naylor et al. Stream traffic communications in packet switched networks: Destinations and buffer considerations. *IEEE Transactions on Communications*, COM-30(12), 1982.
8. P. Sousa and V. Freitas. A framework for the development of tolerant real-time applications. *Computer Networks and ISDN Systems*, 30:1531–1541, Dec. 1998.
9. T. Nandagopal and N. Venkitaraman. Delay differentiation and adaptation in core stateless networks. In *Proc. of INFOCOM 2000*, Tel Aviv, Israel, 2000.
10. I. Busse et al. Dynamic QoS control of multimedia applications based on rtp. *Computer Communications*, 19(1):49–58, Jan 1996.
11. R. Braden et al. Integrated services in the Internet architecture: an overview. *RFC1633*, June 1994.
12. S. Blake et al. An architecture for differentiated services. *RFC2475*, Dec. 1998.
13. L. Kleinrock. *Queueing Systems, Volume 2*. John Wiley and Sons, 1976.
14. G. Bolch et al. *Queueing Networks and Markov Chains - Modeling and Performance Evaluation with Computer Science Applications*. John Wiley and Sons INC., 1998.
15. C. Dovrolis and P. Ramanathan. A case for relative differentiated services and the proportional differentiation model. *IEEE Network Magazine*, 1999.
16. C. Dovrolis and D. Stiliadis. Relative differentiated services in the Internet: Issues and mechanisms. In *Proc. ACM SIGMETRICS'99*, 1999.
17. C. Dovrolis et al. Proportional differentiated services: Delay differentiation and packet scheduling. In *Proc. of ACM SIGCOMM'99*, 1999.
18. C. Dovrolis et al. Proportional differentiated services: delay differentiation and packet scheduling. *IEEE/ACM Transactions on Networking*, 10(1), Feb. 2002.
19. L. Essafi, G. Bolch, and H. Meer. Dynamic priority scheduling for proportional delay differentiated services. Technical Report TR-14-01-03, Univ. Erlangen-Nuremberg, March 2001.
20. M. Leung, J. Lui, and D. Yan. Adaptive proportional differentiated services: Characterization and performance evaluation. *IEEE/ACM Transactions on Networking*, 9(6), Dec. 2001.
21. P. Sousa, P. Carvalho, and V. Freitas. End-to-end delay differentiation of IP traffic aggregates using priority queueing models. In *Proc. of the IEEE Workshop on High Performance Switching and Routing (HPSR2002)*, pages 178–182, Kobe, Japan, May 2002.
22. P. Sousa, P. Carvalho, and V. Freitas. Tunning 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. LNCS 2345, Springer-Verlag, 2002.
23. C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1):44–61, Jan 1973.
24. V. Sivaraman. Statistical analysis of delay bound violations at earliest deadline first (EDF) scheduler. *Performance Evaluation*, 36(1):457–470, 1999.
25. S. Bodamer. A scheduling algorithm for relative delay differentiation. In *Proc. of the IEEE Conf. on High Performance Switching and Routing*, pages 357–364, Heidelberg, June 2000.
26. B. Davie et al. An expedited forwarding PHB (per-hop behavior). *RFC3246*, March 2003.