Enhancing Delay Differentiation Semantics of Class-based IP Networks

Pedro Sousa, Paulo Carvalho and Vasco Freitas

Universidade do Minho, Dept. de Informática, 4710-057 Braga, Portugal
{pns,pmc,vf}@di.uminho.pt

Abstract. This paper presents a time-sensitive scheduler oriented to delay differentiation in class-based networks, studying its behaviour from a single node to an end-to-end perspective. The novel feature of this scheduler is that it tries to bound the queuing delay per class and, simultaneously, to control the excess queuing delays in order to avoid class starvation. The study analyses the scheduler behaviour for heterogeneous class-load distributions and distinct timescales proving the robustness of the mechanism. Discussion on its operational feasibility conditions is carried out and configuration guidelines for its use are provided. In addition, the paper proposes a new queue selection procedure in order to improve its performance in high speed networks.

1 Introduction

In class-based networks [1], where scalability and flexibility is achieved relaxing QoS-guarantees in the network, the integration of time sensitive traffic is difficult mainly due to the reduced traffic control carried out at the core routers. Thus, the deployment of scheduling mechanisms providing queuing delay differentiation among traffic classes plays a relevant role in the integration of real-time traffic in IP networks. In this context, the work presented in [2, 3], focusing on the use of Relative Differentiation, suggests a multiplicative time dependent model used to achieve proportional differentiation behaviour of a network node. In [4, 5] an overview of different delay differentiation models including proportional, additive and an hybrid upper-time queuing model are presented. This hybrid model allows the coexistence of the proportional model with an unique upper time bounded traffic class. Some different schemas, such as EDD [6], 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 that the necessary feasibility conditions [7]. Instead, the mechanism discussed in this paper is more adequate for scenarios where admission control procedures are more relaxed and operate in network edges devices. Related to this aspect, [8] proposes a modified EDD schema in order to differentiate the probability of queuing delay violations under a congested network. In our opinion, it is also 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 smaller than the obtained by low priority classes. In this way, the present work focuses on the hybrid scheduling mechanism proposed in [9], discussing its operational feasibility conditions and providing its configuration guidelines. Moreover, new studies for heterogeneous class load distributions and for finer-grain timescales are carried out. This work also extends the study of the scheduler behaviour from a node to an end-to-end perspective, debating also important implementation issues of the scheduler. In this paper, Section 2 presents the model definition upon which the hybrid scheduler is based. Section 3 presents simulation results illustrating the scheduler differentiation behaviour for (i) a single-node and (ii) end-to-end. Section 4 focuses on implementation issues of the scheduler. Section 5 presents the conclusions of the work.

2 Hybrid Priority Queuing Model

2.1 Model Construction

This section overviews the development of the hybrid PQ model oriented to handle multi-class delay differentiation. Therefore, lets consider N traffic classes, $Class_i$ with $0 \leq i \leq N-1$, where $Class_0$ has the highest priority. The proposed queuing model has evolved from the Upper Time Limit model [10], where a time boundary U_i is defined for the packet queuing time of class i. However, under congestion[1], this time can be exceeded resulting in an unbounded queuing delay, and consequently, class starvation of lower priority classes may occur. 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). The proposed model underlying idea is to allow congested classes to be differentiated avoiding the priority function to assume an infinity value in that region. To achieve this, the excess queuing delay, i.e. the difference between the total[2] and upper time delay ($(t - t_{0_i}) - U_i$) is multiplied by a scale parameter C_i. The resulting hybrid priority queuing model is then configured with two distinct sets of parameters: *Upper time differentiation parameters* $(U_0, ..., U_{N-1})$ and *Congestion differentiation parameters* $(C_0, ..., C_{N-1})$. The final priority function is given by (1) [9], with $\delta_t = t - t_{0_i}$ and $0 \leq i \leq N-1$. Based on Eq. (1), the scheduler selects the traffic class with the higher priority value, $p_i(t)$, and forwards the heading packet from such class. Using this mechanism, the total delay[3], 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*, $d_i^{\bullet}$, as expressed by Eq. (2).

$$p_i(t) = \begin{cases} \dfrac{\delta_t - U_i}{\delta_t} & \text{if } \delta_t < U_i \\ (\delta_t - U_i) * C_i & \text{if } \delta_t \geq U_i \end{cases} \quad (1) \qquad\qquad d_i = d_i^{\circ} + d_i^{\bullet} \quad (2)$$

[1] We use the term *congestion* 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 configuration parameters.

[2] t_{0_i} is the arrival time of the heading packet of $Class_i$.

[3] In the remaining of this paper, d_i is also used to denote the average queuing delay of $Class_i$ for a given measurement interval.

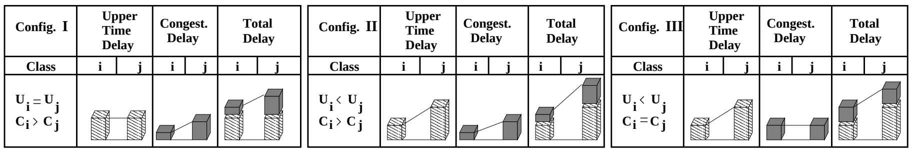

Fig. 1. Distinct combinations of configuration parameters.

2.2 Parameter Configuration

Fig. 1 illustrates distinct behaviour of the hybrid queuing model resulting from three configuration modes, obeying to the fundamental Relative Differentiation rule, i.e. $d_0 \leq d_1 \leq ... \leq d_{N-1}$. For each configuration, the relations between the *upper time delay* and *congestion delay* are presented for two generic classes[4] i and j with $i < j$.

Configuration Mode I: In this configuration identical upper time parameters U_i and U_j are configured for classes i and j, which are then differentiated by congestion parameters C_i and C_j. This operation mode is appropriated for real-time classes with the same upper time limit for queuing delay and distinct capabilities to absorb possible delay violations. The expected behaviour 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 Eq. (3) and is tuned in a proportional mode depending on the *congestion parameters*.

Configuration Mode II: In this configuration the traffic classes are distinct as regards both U_i, U_j and C_i, C_j parameters. This operation mode is appropriate to differentiate high delay sensitive applications with low capacity to absorb excess queuing delays. Again, if the server becomes overloaded and under *upper time limits* violations, the delay differentiation is given by Eq. (4).

Configuration Mode III: In this mode classes are only differentiated by U_i, U_j 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 behaviour for the excess queuing delays meaning that they have similar capacities to absorb delay violations. The delay differentiation achieved by this model is given by Eq. (5).

$$d_j - d_i \approx \underbrace{d_i^\bullet \cdot \left(\frac{C_i}{C_j} - 1\right)}_{congestion\ part} \quad (3) \qquad d_j - d_i \approx \underbrace{U_j - U_i}_{upper\ time\ part} + \underbrace{d_i^\bullet \cdot \left(\frac{C_i}{C_j} - 1\right)}_{congestion\ part} \quad (4)$$

$$d_j - d_i = (d_j^\circ - d_i^\circ) \approx \underbrace{U_j - U_i}_{upper\ time\ part} \quad (5)$$

Parameters Feasibility: It is common to find research work dealing with scheduling mechanisms assuming a set of configuration parameters without explaining or discussing the criteria and the feasibility problems within the choice of such parameters. From an administrative perspective and in order to setup realistic configurations on the

[4] The queuing model can be applied to more complex scenarios including a larger set of traffic classes with mixed configuration modes.

network nodes it is crucial to understand some basic aspects of queuing theory. Conservation law is the basic law to follow (see Eq.(6)). Its semantics demonstrates that the average queuing delay of a generic work-conserving queuing discipline cannot be lower that the queuing delay of the aggregate traffic in a First Come First Serve (FCFS) mechanism. When dealing with several traffic classes, an additional set of feasibility conditions is given by Eq. (7) (presented in [11] and also referred in [2]), where $\varPhi$ represents a set of $2^n - 2$ non-empty proper subsets of $\{1, 2..., n\}$ for a mechanism with n distinct classes, with λ_i denoting the arrival rate of $Class_i$, d_i the class average delay, ℓ_i the server utilisation by $Class_i$ and $d(\sum_{i\in\phi} \lambda_i)$ the average queuing delay suffered by the aggregate traffic in a FCFS server.

$$\sum_{i=1}^{N} \ell_i * d_i \geq \ell * d_{FCFS} \qquad (6) \qquad\qquad \sum_{i\in\phi} \lambda_i \cdot d_i \geq \left(\sum_{i\in\phi} \lambda_i\right) \cdot d\left(\sum_{i\in\phi} \lambda_i\right), \forall\phi \in \varPhi \qquad (7)$$

$$\{d_1^f = d_1 \pm \varDelta_1, d_2^f = d_2 \pm \varDelta_2, ..., d_n^f = d_n \pm \varDelta_n\} \qquad (8)$$

As illustrated, Eq. (7) is an evolution of Eq. (6) now applied to all possible combinations involving the traffic classes. The previous equation highlights the problems which may occur in the differentiation node setup: depending on the class traffic loads, the configuration parameters can become unfeasible, i.e. for a set of target delays, $\{d_i\}$, Eq.(7) and Eq. (6) may be invalid. These problems are common to all differentiation models that deal with multiple traffic classes. For unfeasible configurations of the differentiation mechanism Eq. (7) and Eq. (6) auto-adjust the class delays in order to obtain a set of valid feasible equations. This means that for a given set of arrival load values $\{\lambda_i\}$, the corresponding feasible delays, $\{d_i^f\}$ are evaluated and effectively achieved by the differentiation mechanism. Therefore, deviations from the initial target delays $\{d_i\}$ are expected to occur, and consequently, the obtained values can be expressed by (8), where $\varDelta_i$ represents the delay deviation of $Class_i$. One of the contributions of the proposed mechanism is the ability to control such deviations, in fact the *Congestion Differentiation Parameters* have the semantic power to establish maximum relations for the spread of $\{\varDelta_i\}$ values. In other words, they supply an extra control instrument to bound the spread of the deviations introduced by particular operational conditions of the differentiation nodes. Despite this improvement in the scheduler behaviour, it is also useful to have realistic target delays, i.e. values achieved by the differentiation node in some generic and expectable load conditions, otherwise there is the risk of having the differentiation mechanism always in *congested mode* and the deviations (or *congestion delays*) permanently above the reference upper time limits (i.e. $\varDelta_i \gg d_i$). For this purpose, it is useful to provide simple heuristics to help, for instance, the administrator, with acceptable parameter configurations. The idea is not to achieve precise configuration values due to the previously mentioned feasibility problems. In fact, the variability and characteristics of the traffic arrivals and the service times heterogeneity hinder the use of queuing models such as M/D/1, M/M/1, M/G/1, G/M/1 to obtain precise values[5] for the parameters' configuration. Despite that, and from the assumption of some valid range for class loads and service times, each of the mentioned models may constitute

[5] The equations of such models provide average values which means that considerable deviations can still be observed.

an acceptable reference for the configuration parameters. For example, if one simplifies the assumptions of the single-node study of Sec. 3.1 as: Poisson traffic arrivals, deterministic service times (for an average packet size of 500 bytes) for the same overall load (around 95% in this case) and then apply equation $d = \frac{\ell * \bar{S}}{2*(1-\ell)}$[6] then a value of $380 \mu sec$ is obtained. Using this value and taking into account Eq. (6), a set of reference delay limits can be obtained. In fact, an indication for the values selected for the U_i parameters in the single node study was obtained using this heuristic. Note that the use of M/D/1 model is a very optimistic approach as in practice network traffic has a mixed nature and the service times are likely non-deterministic. Therefore, it is expected that higher values are observed which, in this case, is also an objective in order to the total scheduler working region. Similar reference values can be obtained for specific scenarios using other queuing models. Nevertheless, considerable deviations on queuing delays are always expected and the proposed scheduling mechanism has an important role in the control of such unfeasible working regions of the scheduler.

2.3 End-to-End Considerations

This section extends the single node conclusions with considerations about the end-to-end behaviour of the scheduler as applications and users are ultimately interested in the overall service provided by network. This aspect is studied in section 3.2 where an end-to-end analysis is carried out. Another important end-to-end issue is the advertisement strategy (if any) of the network capabilities to the users/applications. Assuming a domain differentiation path, where M network nodes include the proposed scheduler, estimations of both end-to-end delay, $U_{advt,i}$, and maximum $C_{advt,i}$, for class i and for a given network path, can be announced as per Eq. (9). These metrics can be manipulated by a QoS-capable routing mechanism [12] or delivered to edge routers to help controlling the access to the differentiation domain.

$$U_{advt,i} = \sum_{m=0}^{M-1} U_i^m \; ; \; C_{advt,i} = max_{0 \leq m \leq M-1}(C_i^m) \tag{9}$$

3 Single Node and End-to-end Performance Evaluation

The proposed scheduler was implemented and tested in the *network simulator* (NSv2) following the simulation layout of Fig. 2. The testbed includes Pareto on-off (with $\alpha = 1.2.$), exponential and isochronous traffic sources which are mapped to classes A, B and C contending for a common link. Each class contributes evenly to the overall load (in the long term), and generates mean packet lengths of 500 bytes uniformly distributed over the interval $[250, 750]$. Similar queuing resources were allocated for all classes. In the tests, $Class_A$ has the highest priority as Pareto traffic is the more demanding on the differentiation algorithm. The study focuses on a single-node (dashed shape in the figure) and on an end-to-end perspective (along the four differentiation nodes).

[6] This formula provides the average queuing delay of the aggregate traffic in the M/D/1 model with $\bar{S}$ as the average service time.

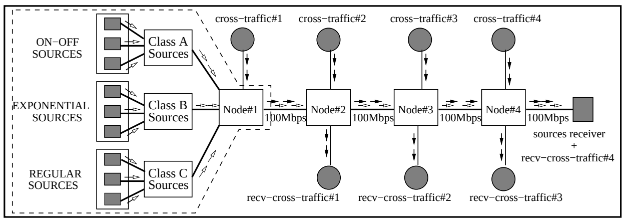

Fig. 2. Simulation scenario.

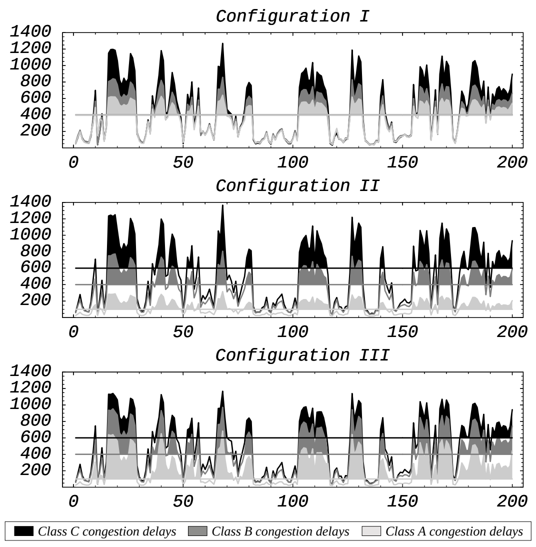

Fig. 3. (a) Delay differentiation for $(U_A, U_B, U_C) = (400\mu s, 400\mu s, 400\mu s)$ and $(C_A, C_B, C_C) = (4, 2, 1)$ (Conf. I) (b) $(U_A, U_B, U_C) = (100\mu s, 400\mu s, 600\mu s)$ and $(C_A, C_B, C_C) = (4, 2, 1)$ (Conf. II) (c) $(U_A, U_B, U_C) = (100\mu s, 400\mu s, 600\mu s)$ and $(C_A, C_B, C_C) = (1, 1, 1)$ (Conf. III).

3.1 Single-node Differentiation

Figs. 3(a)(b)(c) show three differentiation examples obtained using the hybrid scheduler from a single-node perspective, each one corresponding to a particular configuration mode. The x-axis represents the server packet transmission times with a plot granularity of $25ms$ (625 packet transmission times) and the y-axis represents the average queuing delays (in microseconds) over such intervals.

As plotted in Figure 3(a) 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. The proportional relation between the excess queuing delays may be easily visualised, as excess delays in $Class_C$ are approximately twice the delays in $Class_B$, which in turn double $Class_A$ delays. This agrees with the proportional relations defined for C_A, C_B, C_C. showing that this configuration mode is feasible. Fig. 3(b) plots the differentiation behaviour for configuration mode II. As the Figure shows, 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 $Class_C$. Fig. 3(c) illustrates the differentiation behaviour for configuration mode III. As the Figure shows, for congested periods there is an excess queuing delay in all congested classes following a fair distribution among classes. 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 $Class_C$. In addition, simulation results for different class load distributions and also for short measurement intervals were also obtained. Due to the impossibility of including graphic representation for different measurement time scales for all the configurations and for distinct class load distributions, three rules were defined as the basis for verifying the differentiation correctness of the configuration modes and their applicability to all simulation scenarios.

- *Property 1*: Fundamental Differentiation Rule: For a generic time interval $[t_x, t_{x+\Delta_t}]$ in which all classes *a*, *b*, *c* have packets waiting to be served[7], $d_a \leq d_b \leq d_c$ are the average queuing delays in $[t_x, t_{x+\Delta_t}]$, having $Class_a$ the highest priority.
- *Property 2*: Congested Differentiation Rule: For a generic time interval $[t_x, t_{x+\Delta_t}]$ in which all classes *a*, *b*, *c* have packets waiting to be served, with upper time limit violations, $d_a^{\bullet} \leq d_b^{\bullet} \leq d_c^{\bullet}$ are the average excess queuing delays in $[t_x, t_{x+\Delta_t}]$, having $Class_a$ the highest priority.
- *Property 3*: Uncongested Differentiation Rule: For a generic time interval $[t_x, t_{x+\Delta_t}]$ in which all classes *a*, *b*, *c* have packets waiting to be served, with no upper time limit violations, $d_a^{\circ} \leq d_b^{\circ} \leq d_c^{\circ}$ are the average queuing delays during $[t_x, t_{x+\Delta_t}]$, having $Class_a$ the highest priority.

New simulation studies were performed for scenarios with distinct combinations of class loads such as $(C_A load, C_B load, C_C load) = (50\%, 25\%, 25\%)$, $(25\%, 50\%, 25\%)$, $(25\%, 25\%, 50\%)$, $(40\%, 40\%, 20\%)$, $(60\%, 20\%, 20\%)$. In all these load scenarios the results[8] verified the three properties defined above. This shows the model ability to handle distinct class load distributions.

3.2 End-to-End Differentiation

This section analyses the behaviour of the scheduler in a network comprising multiple differentiation nodes in order to study its end-to-end performance. As for the single

[7] This condition was included due to the possibility that for a given interval or subinterval there is no traffic from a given class in the server which would cause a zero or lower average queuing delay than the obtained by higher priority classes.

[8] All results report to a sampling interval of 40 measurements per second, the same scale used in the single-node and end-to-end study, i.e. $\Delta_t = 25ms$. Different time scales were also used and similar conclusions are also valid for a more finer-grain sampling interval of $\Delta_t = 2.5ms$.

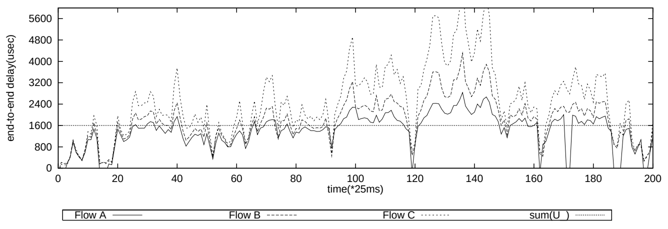

Fig. 4. End-to-end delay differentiation for $(U_A, U_B, U_C) = (400\mu s, 400\mu s, 400\mu s)$ and $(C_A, C_B, C_C) = (4, 2, 1)$ (Configuration I).

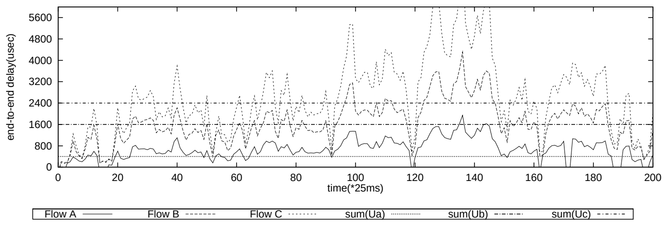

Fig. 5. End-to-end delay differentiation for $(U_A, U_B, U_C) = (100\mu s, 400\mu s, 600\mu s)$ and $(C_A, C_B, C_C) = (4, 2, 1)$ (Configuration II).

node study, congestion of the differentiation nodes along the path is achieved resorting to cross-traffic. Here the term *congestion* is used in a medium time scale perspective. In fact, due to the nature of the traffic sources, e.g. Pareto which has high load variability, there are small simulation periods where the nodes are not under heavy load. This situation was also noticed in the single-node case for time intervals where all classes had low delays. In the end-to-end analysis of the scheduler, if all nodes were under permanent congestion, i.e. even over all short time scales, then the conclusions for each configuration mode were almost similar as the ones presented for the single-node study. Nevertheless, for specific and maybe more realistic conditions (e.g. only a subset of the path nodes congested or short-time load oscillations in the server) different results can be observed. To assess the expectable end-to-end differentiation behaviour, several examples are presented based on the testbed of Fig. 2, where the end-to-end class queuing delays were measured (if $d_i^\star$ represents the end-to-end queuing delay of $Class_i$ then $d_i^\star = \sum_{j=0}^{M-1} d_i^j = \sum_{j=0}^{M-1}[d_i^{\circ,j} + d_i^{\bullet,j}]$).

Fig. 4 illustrates the differentiation results for configuration mode I with $(U_A, U_B,$-$U_C) = (400\mu s, 400\mu s, 400\mu s)$ and $(C_A, C_B, C_C) = (4, 2, 1)$ for all nodes. If all nodes are under heavy load, the expected total end-to-end target delay is around $1600\mu s$. Additionally, it is expected that for high congestion periods the single node queuing delay violations follow a proportional spread according to C_i. From the data in Fig. 4, as for the single node study, this characteristic is also valid at end-to-end. Fig. 5 presents the differentiation results for configuration II with $(U_A, U_B, U_C) = (100\mu s, 400\mu s,$-$600\mu s)$ and $(C_A, C_B, C_C) = (4, 2, 1)$ for all nodes. If all nodes are under heavy load, the expected total end-to-end target delay is around $(400\mu s, 1600\mu s, 2400\mu s)$. Again,

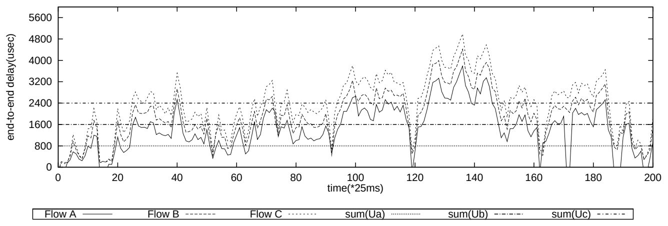

Fig. 6. End-to-end delay differentiation for $(U_A, U_B, U_C) = (200\mu s, 400\mu s, 600\mu s)$ and $(C_A, C_B, C_C) = (1, 1, 1)$ (Configuration III).

it is expected that for high congestion periods the single node queuing delay violations follow a proportional spread depending on C_i. From the data in Fig. 5 it is clear that when all nodes in the path are under heavy load conditions (e.g. x-axis interval $[120, 150]$), the spread among the classes' excess queuing delays is close to the ratio between the corresponding C_i parameters. Despite that, for time intervals where only a subset of the nodes are under heavy load conditions the gap between the excess end-to-end queuing delays is much lower than the obtained for full congested periods, as expected. Fig. 6 presents the differentiation results for configuration mode III with $(U_A, U_B, U_C) = (200\mu s, 400\mu s, 600\mu s)$ and $(C_A, C_B, C_C) = (1, 1, 1)$ for all nodes. Assuming again all nodes under heavy load, the expected total end-to-end target is around $(800\mu s, 1600\mu s, 2400\mu s)$. This configuration leads to an end-to-end differentiation behaviour slightly different from the single node study. As shown in Fig. 6, during the simulation example, the end-to-end excess queuing delays of the high priority class are slightly higher than the obtained by the lower priority class[9]. As pointed out before, this is caused by a partial congestion state of differentiation nodes. Let's consider a limit situation to illustrate such end-to-end behaviour with the assumption that only two of the four nodes are under heavy load conditions and with the same configuration parameters as in Fig. 6. Additionally, a delay close to zero is considered for all classes in the non-congested nodes and an excess queuing delay of $210\mu s$ for all classes in the congested nodes. These conditions lead to in an end-to-end queuing delay of $(820\mu s, 1220\mu s, 1620\mu s)$. Comparing these values with the corresponding end-to-end target queuing delays, i.e. $(800\mu s, 1600\mu s, 2400\mu s)$, only the high priority class has its upper time limit violated. Although the example of Fig. 6 assumes all nodes under long-term heavy load conditions which means that such limit situation is not applicable for the simulation scenario, the truth is that a similar reasoning can be made considering natural load oscillation along the transmission path. Configurations I and II may suffer similar deviations in such conditions but due to the spread of C_i parameters assumed in these configurations, the end-to-end differentiation behaviour is close to the single node conclusions for a large set of tested scenarios. For these reasons, configurations modes I and II are more appropriate for the required end-to-end delay differentiation under operational conditions other than the permanent congestion of network

[9] Note however that for all configurations modes, including III, the fundamental Relative Differentiation rule is preserved even in the end-to-end behaviour: $d_0^\star \leq d_1^\star \leq ... \leq d_{N-1}^\star$.

nodes. Nevertheless, this conclusion does not impair using configuration III for more specific network scenarios assuming that, as the congestion increases special attention has to be given to delay differentiation mechanisms, which, in the limit, are operating at their maximum differentiation capacity. Recall that the adaptive parameterization of differentiation nodes can be used to improve the end-to-end differentiation capabilities of the scheduler for other than permanent congestion situations. In fact, network monitoring schemes can be used in order to measure the delay differentiation achieved in a specific network path and, based on such information, readjust *on-the-fly* the nodes' configuration parameters in order to provide a better end-to-end delay differentiation.

4 Implementation Issues

Although based on simple arithmetic operations, the processing time required to compute p_i may become a bottleneck when the output capacity of the server increases, which in turn leads to low 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 selection procedures is measured using Eq. (10) where γ expresses the *service degradation ratio* of the differentiation model, ρ denotes the server utilisation[10] and $\rho^{\circlearrowright}$ denotes the server utilisation taking into account the processing overhead induced by queue selection procedures. From queuing theory along with Eq. (10), the following relation is obtained: $\gamma = (\frac{\lambda \cdot (\bar{S} + t^{\circlearrowright}) - \lambda \cdot \bar{S}}{\lambda \cdot \bar{S}})$, where $t^{\circlearrowright}$ represents the amount of time required to compute the next queue to be served[11]. As consequence, this relation can be presented as $\gamma = (\frac{t^{\circlearrowright}}{\bar{S}})$, i.e. γ is the ratio between the processing time and the service time. In conclusion, Eq. (11) can also be used to compute γ, for an average packet size k and an output link capacity C_l.

$$\gamma = \left(\frac{\rho^{\circlearrowright} - \rho}{\rho} \right) \qquad (10) \qquad\qquad \gamma = \left(\frac{t^{\circlearrowright}}{k} \cdot C_l \right) \qquad (11)$$

Eq. (11) shows that for a fixed output capacity and a specific $t^{\circlearrowright}$ value, γ depends highly on the packet size. Due to the high capacity of current computational systems, low values of γ are expected. Nevertheless, even these small deviations can have strong influence in the system behaviour. In fact, and as explained by queuing theory, for working regions where $\rho > 0.6$ even small increases in the server utilisation may lead to considerable increases in queuing delays and in the number of customers in the system. For this reason, the overhead induced by queue selection procedures should be reduced. In order to illustrate the queue selection constraints, a simple example is presented for a Linux PC with traffic differentiation capabilities and a 2GHz Intel processor. The aim is to illustrate the overhead induced by the queue selection operations when varying the output link capacity and the packet length. The value of γ also depends on the number of traffic classes to differentiate. The example presented in Fig. 7 assumes four traffic classes, packet lengths varying from 256 to 10240 bytes and output link capacities in the interval $[1Mbps, 5Gbps]$. As shown, the value of γ is almost irrelevant for an

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

[11] This is a platform-dependent factor.

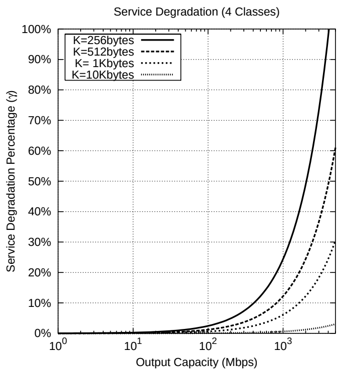

Fig. 7. Service degradation examples.

```
$tpacket \leftarrow \frac{sizeof(packet)}{C_l}$

$< ..sendpacket... >$

$Sel \leftarrow AdvPriority([C_0, C_{N-1}], t + tpacket)$        $t^{\leftrightarrows}$

$while\ (serverbusy)\ null$

$Sel \leftarrow HigherPriority([C_0, C_{Sel}])$        $t^{\circlearrowright}$
```

Fig. 8. Advanced transmission time algorithm.

output capacity below $100Mbps$, becoming more relevant for capacities in the range $[100Mbps, 1Gbps]$. For output capacities above $1Gbps$, γ increases steadily assuming more expressive values. Additionally, and as expected, for a specific output capacity the service degradation increases as packet size decreases. One of the solutions to improve the scheduler performance is to use proprietary hardware circuits to implement the queue selection procedures expressed in Eq. (1) to reduce the service degradation. A more generic and low cost solution can be achieved using the following algorithm:

Advanced Transmission Time Algorithm: The algorithm presented in Fig. 8 is based on the inspection of specific data fields of the packet selected for transmission. Using this mechanism it is possible to check the packet length and evaluate the expected packet transmission time (t_{packet}). This means that the next queue selection (i.e. after busy period) should occur $t + t_{packet}$. The previous knowledge of this time allows the selection, during the busy period, of the next class to be served. This is achieved by round robin the traffic classes, evaluating for each one the corresponding $p_i(t)$ value, as it was computed at $t + t_{packet}$. The key point of this strategy is that, a substantial part of the queue selection procedure is carried out during the busy period ($t^{\leftrightarrows}$) reducing the time of the selection procedures after the busy period ($t^{\circlearrowright}$), which is effectively responsible for service degradation. The last line of the algorithm is only required when a higher priority class is empty and $AdvPriority$ function is called and meanwhile a new packet arrives for that class. In such cases, it is necessary to select the highest priority value for the class interval $[Class_0, Class_{Sel}$[12]$]$. However, as referred before, the differentiation mechanisms are designed mainly for heavy load conditions, 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 this algorithm, under heavy load conditions and for similar parameters' assumptions as the ones for Fig. 7, will achieve $t^{\circlearrowright} \approx 0$, $\rho^{\circlearrowright} \approx \rho$ and $\gamma \approx 0\%$, i.e. a performance similar to the obtained by the theoretical model[13]. Recall that the

[12] The one selected during the previous busy period.

[13] Note that in Fig. 7 most of the service degradation percentages are below 100% meaning that, on average, the packet transmission time is sufficient to perform the queue selection tasks.

presented algorithm still has complexity $O(n)$. In fact, in the worst case, two complete loops inspecting the classes heading packets lead to $O(T(n)) = O(n) + O(n) = O(n)$. 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 been performed during the busy period. In our opinion, even if the platform or the assumptions vary leading to higher service degradation than in Fig. 7, the use of the presented algorithm will always be an added value as regards reducing service degradation.

5 Conclusions

This article presents an hybrid queuing model able to provide full delay differentiation of real-time traffic. Through a simple and flexible configuration, it is possible to control both the expectable queuing delay and the congestion queuing delay on a traffic class basis. After illustrating the scheduler behaviour for a single node, the study is extended focusing on the end-to-end delay differentiation capability. Specific implementation issues are also discussed and an low overhead queue selection algorithm is proposed.

References

1. S. Blake et al. An architecture for differentiated services. *RFC2475*, Dec. 1998.
2. C. Dovrolis et al. Proportional differentiated services: Delay differentiation and packet scheduling. In *Proc. of ACM SIGCOMM'99*, 1999.
3. C. Dovrolis et al. Proportional differentiated services: delay differentiation and packet scheduling. *IEEE/ACM Transactions on Networking*, 10(1), Feb. 2002.
4. 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 26-28 2002.
5. 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. LNCS 2345, Springer-Verlag, 2002.
6. 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.
7. V. Sivaraman. Statistical analysis of delay bound violations at earliest deadline first (EDF) scheduler. *Performance Evaluation*, 36(1):457–470, 1999.
8. Stefan Bodamer. A scheduling algorithm for relative delay differentiation. In *Proc. of the IEEE Conf. on High Performance Switching and Routing*, pages 357–364, June 2000.
9. Pedro Sousa, Paulo Carvalho, and Vasco Freitas. Scheduling Time-Sensitive IP Traffic. In G. Goos et al, editor, *Proc. 6th IFIP/IEEE International Conference, MMNS*, pages 368–380, Northern Ireland, Belfast, September 2003. LNCS 2839, Springer-Verlag.
10. G. Bolch et al. *Queueing Networks and Markov Chains - Modeling and Performance Evaluation with Computer Science Applications*. John Wiley and Sons INC., 1998.
11. E. Coffman and I. Mitrani. A characterization of waiting time performance realizable by single-server queues. *Operations Research*, 24, 1980.
12. X. Yuan. Heuristic algorithms for multiconstrained quality-of-service routing. *IEEE/ACM Transactions on Networking*, 10(2), April 2002.