A scalable solution to provide Quality of Service in Next-Generation Networks

Gianmarco PANZA Senior Member IEEE [1], Sara GRILLI[2]
[1]*CEFRIEL, via Fucini 2, Milan, 20133, Italy*
Tel: +39 02 23954326, Fax: + 39 02 23954526, Email: gianmarco.panza@cefriel.it
[2] *CEFRIEL, via Fucini 2, Milan, 20133, Italy*
Tel: +39 02 23954204, Fax: +39 02 23954404, Email: sara.grilli@cefriel.it

Abstract: Next-Generation Networks (NGNs) will support Quality of Service (QoS) for multimedia traffic over an IP-based infrastructure. A scalable solution to provide also stringent guarantees is required. DiffServ architecture can offer different levels of service at low complexity, but it is not basically able to efficiently provide end-to-end absolute QoS for real-time traffic. Many research activities have addressed this issue, proposing either an absolute or a relative approach. While the former is complicated to implement in the global Internet, the latter is simpler and can be easily realized without arising scalability concerns because exploiting the Proportional Differentiation Model (PDM), in which the performance distance between classes is proportional to quality differentiation parameters that Network Service Provider (NSP) can configure.

This work aims to achieve absolute delay guarantees relying on a PDM that can be easily deployed in DiffServ architecture using a proportional scheduler, like Advanced Waiting Time Priority (AWTP). The key idea is to enhance the end-to-end delay differentiation provided by PDM, with a run-time class adaptation, which dynamically assigns the service class to critical traffic in order to fulfil its end-to-end delay requirements.

Simulation results have been collected to analyse the trade-off between the fast reaction to load changes and the system stability with different measurement processes, employed to decide for a class promotion or downgrade. The validity and good performance of our proposal have been demonstrated over various scenarios also in the critical case of network congestion.

Keywords: DiffServ, feedbacks, measurement process, Next-Generation Networks, PDM, QoS, scalability, SLA.

1. Introduction

Next-Generation Network (NGN) [1], is an IP network with Quality of Service (QoS) support and able to efficiently transport heterogeneous traffic. Network Service Provider (NSP) has to provide a differentiated treatment within its telecommunication infrastructure and fulfil the requirements of real-time traffic. As summarized in [2], ITU recommendations G.1010, Y.1541 and Y.1221 collect the requirements of real-time applications, from the standpoint of both the end user and NSP. Typically, these standards distinguish between interactive applications, like phone call, non-interactive applications, like video streaming, and best effort applications, like web browsing. The former needs that a certain set of quality parameters (i.e. end-to-end delay and loss) are guaranteed for all the packets or a given high percentage of them.
Different levels of service can be provided by IETF Differentiated Services (DiffServ) [3] in a scalable manner. This architecture focuses primarily on aggregates of flows in the core

routers, and differentiates between service classes rather than providing absolute per-flow guarantees. More specifically, while the access routers process packets on the basis of high traffic granularity, such as per-flow or per-organization, core routers do not maintain fine grain state, and process traffic based on a limited number of Per Hop Behaviours (PHBs) encoded in the packet header, namely in the DiffServ Code-Point (DSCP) field [4]. Diffserv is gaining popularity as the QoS paradigm for NGNs, primarily because it moves the complexity of providing QoS out of the core and into the edges of the network, where it may be feasible to maintain a restricted amount of per-flow state.

ITU Recommendation Y.1541 [2] maps the end-to-end requirements of an application to a given DiffServ Per Domain Behaviour (PDB) [5], which is actually associated to a PHB, hence to a service class. However, IETF has defined only a general framework and service functionality for a differentiated treatment over a limited number of traffic aggregates [3]. Therefore, DiffServ approach should be enhanced in order to offer also end-to-end per flow absolute quality guarantees to properly support real-time applications.

Several research efforts have already addressed the issue. There is also an active IETF Working Group (WG) focused on the matter, namely Next Step In Signalling (NSIS) WG [6]. It seeks to specify a general purpose signalling protocol over the Internet inspired to RSVP [7], also for resource reservation in the network. But, if such a protocol will be actually supported and used in NGNs is yet an open question.

Broadly speaking, the proposals available in literature can follow either an absolute or a relative approach for DiffServ [8]. The former aims to provide QoS on an absolute scale in each service class. While, the latter is only able to offer a service differentiation between classes, i.e. a class can grant a lower delay or loss than another in a qualitative manner. Absolute DiffServ has some drawbacks and appears complex to implement in the global Internet [8], whereas the Relative DiffServ is simpler and more suitable to be use with multimedia applications, which in some cases can even adapt to variations in network performance.

Many algorithms have been proposed [8] to realize Relative DiffServ, but the more promising ones, also for an efficient resource exploitation, are based on the Proportional Differentiation Model (PDM) [8], in which, the performance distance between classes is proportional to given differentiation parameters that NSP can specify.

This work[1] also adopts a PDM for a relative delay differentiation between classes, but augments it with a mechanism for a run-time class (re-)assignment. The proposed solution can dynamically adjust the DiffServ class of real-time traffic in order to address an end-to-end delay bound.

Different measurement processes to decide for the promotion and downgrade of traffic can be employed in Border Routers (BRs). Three main options were analysed and compared considering the two critical issues of performance stability and adaptation speed to network load dynamics.

The designed solution demonstrates effective in providing even absolute QoS guarantees, at low cost and without arising scalability concerns, because built on a DiffServ architecture where routers support a proportional scheduler like Waiting Time Priority (WTP [9], with additional functionality at the border of the network only. Furthermore, it inherits from PDM the efficient resource exploitation and the controllability of performance distance between classes.

The remainder of the paper is organized as follows. In Sect. 2, the state-of-the-art about achieving QoS in DiffServ architecture with a PDM, as well as standards issued by this

[1] This work has been carried out within the framework of the IST OPTIMIX project, partially supported by the European Commission under the contract FP7 n°INFSO-ICT-214625.

work, are reported. In Sect. 3, the proposed solution from an architectural point of view and the employed measurement processes are presented. Sect. 4 describes the investigated simulation scenarios, while Sect. 5 discusses the collected results. In Sect. 6, a comparison with the related works is made. Finally, Sect. 7 outlines the main conclusions and future developments.

2. QoS in DiffServ and relevant standards

Proportional Differentiation Model (PDM) [8] yet allows for controllability, consistency, efficient resource exploitation and scalability. Ref. [8] explains how it is possible to use packet forwarding mechanisms to implement such a model by a joint scheduling-dropping method to enforce proportional average delay and packet loss. Specifically, Waiting Time Priority (WTP) scheduler [9] can approximate the proportional delay differentiation model also in short timescales, because in this scheduler the priority of a packet increases proportionally to its waiting time according to quality factors assigned to queues. Other works based on a PDM are available in literature and proposing other scheduling algorithms, like dynamic Weighted Fair Queuing (WFQ) [10].

Although the adoption of Relative DiffServ architecture based on a PDM appears to be the most promising solution for service differentiation, it is also necessary to provide absolute QoS guarantees to the real-time traffic. Various approaches have been proposed in literature for achieving absolute service bounds still relying on a proportional model. They can be divided into two groups: single-hop and end-to-end approaches.

In the former, the absolute quality requirement can be addressed with a joint packet scheduling and dropping algorithms. In literature there are a lot of examples of this approach, in which traditional schedulers, like WTP, are coupled with a dropping mechanism [11], or when new schedulers are defined (for example Deadline fair sharing [12]) for the purpose. In the category of single hop approaches there are also some methods that try to predict the delay of the backlogged traffic and use this delay to update the service rate of each class [13], and, additionally, the amount of traffic to be dropped [14].

In the latter, various strategies proposing a run-time change of service class for the real-time traffic are available. In the first solution, the selection of the initial service class and its change along the connection life-time of the issued traffic are based on the application requirements and triggered at the user side [15], as needed. The second solution integrates with the admission control process, using probe packets for measuring the packet loss rate and the presence of congestion [11]. The third one instead, chooses to adaptively select the service class depending on the current performance. Either the end-to-end delay [16] or the loss [17] of the real-time traffic is monitored and if the related Service Level Agreement (SLA) is not met the traffic is moved to a higher priority class, otherwise it can be moved to a lower one. More specifically, in Ref. [16], the ingress router sends a probe packet for each flow in order to monitor the end- to-end delay; when a router along the path receives the probe packet, it calculates the average per-hop delay of the corresponding flow and adds it to the value already written in the packet. Once the egress router has received the probe packet it sends it back to the ingress router, which evaluates the resulting end-to-end average delay. If the requirement on the delay is not met, the traffic is promoted to a better service class. While, if the end-to-end delay of the probe is below the bound and a lower class can sustain the traffic, the traffic is downgraded there. However, in this last and more promising proposal the adaptation is based on the average delay of each class as the reference QoS parameter, which is not often the more relevant in practice. A given (high) percentile of the end-to-end delay is of greater interest for real-time applications. Moreover, the average delay is calculated as the sum of average per-hop delays of each router along the path of the issued flow, which in their turn are actually only an estimate of the average per-hop delay, being calculated on all the packets transmitted on the concerned interface at

each router. Furthermore, burst of traffic and class change could entail inconsistency in supporting a PDM. Finally, functionality is added in core routers possibly compromising the scalability of the architecture.
The solution proposed in this work to achieve absolute QoS guarantees using a PDM is still based on a dynamic change of the service class for the real-time traffic with the aim of addressing the related SLA. However newly created probe packets are not injected into the network, avoiding bandwidth overhead, and additional functionality with respect to the base DiffServ architecture [3] is introduced only in Border Routers (BRs). Moreover, the 99th percentile of the end-to-end delay for the real-time traffic is considered as the reference QoS parameter, which gives a better picture, more reliable and consistent with the application requirements, of the granted service.
For the delay evaluation, time synchronization between BRs is needed. However, due to the transfer of legacy voice traffic of fixed and mobile users in NGNs, sharing accurate timing information by network nodes in a reliable and cost effective manner is mandatory, and several solutions have been already proposed in literature. Noticeably, IEEE 1588 [18] realizes a scalable synchronization scheme with sub-microseconds accuracy by the exchange of messages between network devices in the same manner as Network Timing Protocol (NTP) [19]. Furthermore, one of the recent major standardization trends in carrier-scale Ethernet is to deliver accurate timing and synchronization information for real-time audio/video streaming and circuit emulation for legacy services [20][21].

3. Absolute QoS provisioning

The objective of this work is to provide absolute QoS for real-time traffic in an IP network with a PDM in DiffServ architecture. To achieve this, critical traffic is moved run-time to the lower service class yet addressing its delay requirement as by SLA.
The chosen reference quality parameter for the real-time traffic is the 99th percentile of the end-to-end delay (though, the solution can be easily generalize to a whatever percentile), since the real-time traffic has a very stringent requirement on the end-to-end delay experienced by a high percentage of its packets (e.g. about 200 ms for voice). Moreover, it is worthwhile to underline that having peaks on the end-to-end delay badly affects the jitter and therefore, the packet loss (if a packet does not arrive at the receiver soon enough to be played out it is actually discarded).
The end-to-end delay can be monitored either at the egress of the network (i.e. at the egress BR) or directly at the receiver terminal (as available). The figures of the 99th percentile of the end-to-end delay are sent as feedback to the ingress BR of the network and elaborated by using one out of three measurement processes, either a threshold-based method, moving average or low pass filter. If the calculation results in an exceed of the bound on the 99th percentile of the end-to-end delay for the issued traffic, the ingress BR remarks the DS-field of the packets of the given flow(s) promoting it (them) to the closest upper class of service. If in this new class the requirement on the delay is not respected yet, the flow(s) can be promoted to the subsequent better class, and so on up to the best class of service (or to the highest allowed, as specified by SLA). While if the end-to-end delay bound is granted by the closest lower class, the flow(s) can be moved back there (up to the initially assigned class, as defined by SLA).
The two critical issues that must be considered when deciding which method of promotion or downgrade is the most suitable are the system stability and reaction speed to network load changes.
From one hand, the system must be fast in order to promote the critical traffic as soon as the related requirement on the 99th percentile of the end-to-end delay is not addressed anymore in the currently assigned class or downgrade the flow if such requirement is met also in the closest lower class. From the other hand, the system must be stable enough to avoid

frequent changes of the assigned class because this could lead to extremely variable performance (e.g. high jitter), also to other flows, not to consider the complexity induced in the ingress BRs that should re-configure too often their marking policy. It is clear that the best solution should be sought in a good trade-off between the two issues, in order to have a robust and reliable system in providing absolute QOS guarantees in a scalable manner for real-time traffic.

3.1 – Measurement Processes

The evaluation of the 99th percentile of the end-to-end delay for the real-time traffic is to be performed run-time. Our proposal is to store the delay experienced by N_{udp} consecutive packets of the concerned traffic and then to pick up the sample D_{99}^{udp} that better corresponds to the 99th percentile (typically, the real-time traffic is over UDP). For example, if N_{udp} is 100, the second greatest one is picked up. The delay of a packet can be easily calculated if the network nodes are synchronized (refer to Sect. 5.4 for details).
When the feedback of the delay reaches the issued ingress BR, it is elaborated in order to decide for promotion. Three measurement processes have been considered.

- **Moving average**: the ingress BR collects and stores K_{up} consecutive values of D_{99}^{udp}; when a new value of the 99th percentile is received, the oldest one of the K_{up} consecutive values is discarded. On this data set the ingress BR calculates the moving average *MA*, that is,

$$MA = \sum_{i=0}^{i=K_{up}} D_{99}^{udp} \Big/ K_{up} \qquad (1)$$

  If *MA* is below the threshold *D* specified in the issued SLA nothing happens. Otherwise, the SLA is not satisfied and the given traffic needs to be promoted to the closest upper class.
- **Threshold**: the ingress BR checks if the 99th percentile of the end-to-end delay of the traffic is greater than the threshold D for K_{up} consecutive times. In the case, the traffic needs to be promoted.
- **Low pass filter**: the ingress BR uses the last calculated output of the averaged value of the 99th percentile of the end-to-end delay of the traffic and the current (last received) one for calculating the new output of the low pass filter *LPF*, that is:

$$LPF = LPF_{old}^{udp} * P_{up} + D_{99}^{udp} * (1 - P_{up}) \qquad (2)$$

  If the result is greater than the threshold *D*, the traffic needs to be promoted.

The decision for the downgrade of the issued real-time traffic is based on the monitoring of the end-to-end delay of the whole traffic D_{99} (in fact, a portion of it big enough to provide reliable evaluations) for the lower class of service closest to the currently assigned. The ingress BR collects the feedbacks about the delays sent by the egress BR (actually, from all the egress BRs, but only one is related to the given traffic) and elaborates them according to one of the following measurement processes.

- **Moving average**: the ingress BR collects and stores K_{down} consecutive values of $D_{99\ for}$ the closest lower class; when a new value of the 99th percentile is received; the oldest one of the K_{down} consecutive values is discarded. Using this data set the ingress BR calculates the moving average *MA*, that is,

$$MA = \sum_{i=0}^{i=K_{down}} D_{99} \Big/ K_{down} \qquad (3)$$

If *MA* is below the threshold D specified in the issued SLA, the closest lower class can address the traffic delay requirement and the given traffic can be moved into that class. Otherwise, the traffic remains in the current class.

- **Threshold**: the ingress BR checks if the 99th percentile of the end-to-end delay of the closest lower class is below the threshold *D* for K_{down} consecutive times. In the case, the traffic can be downgraded to that class.
- **Low pass filter**: the ingress BR uses the last output of the averaged value of the 99th percentile of the end-to-end delay and the current (last received) one of the closest lower class for calculating the new output of the low pass filter *LPF*, that is:

$$LPF = LPF_{old} * P_{down} + D_{99} * (1 - P_{down}) \qquad (4)$$

If the result is below the threshold *D*, the traffic can be downgraded.

The monitoring of the end-to-end delay for each class happens independently from the promotion of the issued traffic. This is reasonable because the movement of a small percentage of traffic (i.e. only the traffic with absolute QOS requirements) between classes in a core network with high speed links has no impact on the overall network behaviour. However, the target closest lower class never decreases its performance after the downgrade in our PDM implementation.

In Table 1, the configuration parameters with suitable values for both the promotion and the downgrade processes are reported.

4. Simulation scenarios

The simulation analysis has been carried out in a network scenario with four aggregated sources of traffic and three routers, (see Figure 1), which deploy a PDM scheduler on Output links of 100 Mbit/s (the other links are set to a higher capacity in order not to have an impact on the collected results).

The first router can classify and mark the incoming IP packets, as ingress BR in DiffServ (DS) architecture. While all routers route the packets and schedule them according to the assigned class of service. Indeed, at each router the traffic, coming from source(s) or from the uplink router, is divided into four queues by looking at the DS-field [4], each one corresponding to a different class.

Advanced Waiting Time Priority (AWTP) [9] is the deployed proportional delay differentiation scheduler. It derives from WTP, in which the priority that is assigned to a packet increases proportionally with its waiting time and linearly depends on the quality factor assigned to the associated queue (i.e. to the related class of service). The highest priority packet is served first. Ref. [8] explains how WTP can well support a PDM when the network load is heavy (indeed, in low load conditions the network performance is not an issue). AWTP enhances WTP considering in the priority calculation also the transmission time of the packets on the head of the queues to achieve a delay differentiation more consistent with the mutual ratio of the assigned quality factors. The factors 1, 2, 3 and 4 are assigned to four queues, respectively. According to a PDM, the delay experienced in the first queue should be about twice the delay experienced in the second queue and three times the delay in the third queue; while, the delay experienced in the second queue should be about twice the delay experienced in the fourth queue, and so on. Buffers are big enough to avoid losses.

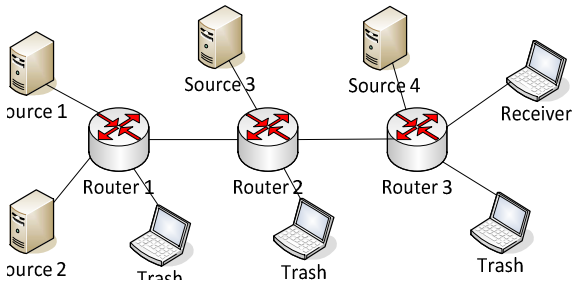

Figure 1: Reference network scenario used for simulations.

A router sends about 50% of the traffic to a node that acts as a trash, and the remaining downlink. At the receiver, the measurements are made on the real-time traffic generated by *Source 1* and *Source 2*, which is about 4 Mbit/s (the 15-20% of an aggregate).
For each class of service, an aggregate is composed of different types of traffic as taken by real backbone traces [22][23]:

- 1 TCP flow aggregate, with an average traffic rate of 16 Mbit/s,
- 17 MPEG4 video flows, each one with an average traffic rate of 280 kbit/s,
- 3 ON/OFF G.729 audio flows, each one with an average traffic of 10 kbit/s.

This leads to about 80 Mbit/s on average of overall traffic in each router output link, as ordinary network conditions, in a simulation period of 60 s. During two time intervals additional traffic is injected in every link to create a higher load, up to almost 100Mbit/s. This further traffic is artificially synthesized and is characterized by a constant packet inter-generation time of 1 ms and a constant packet size of 4 kbits between 10 and 25 s. whereas, by a constant packet inter-generation time of 2 ms and a constant packet size of 7 kbits between 40 and 55 s.
The threshold D for the 99^{th} percentile of the end-to-end delay as specified by SLA is set to 10 ms.
In a first analysis, the transmission delay of the feedbacks from the receiver side to the ingress BR is neglected. Next, different feedback transmission delays, a higher number of routers to be crossed and an enlarged performance distance between the service classes are considered.

5. Simulation results

The analysis was carried out by OPNET [24] as a well-known and reliable simulation tool. A wide variety of real traces were used, played starting from an instant of time randomly selected in the first 1 s of simulation, and the results were collected after the end of the initial transitory period. Furthermore, the scalar figures are the result of an averaging process performed over values gathered with several simulation runs, using each a different seed properly selected [25].
A large set of scenarios and design options were investigated. The relevant data and analysis are reported in the following subsections, as a compromise between clarity, completeness and available room.

5.1 – Comparison between the Measurement Processes

First, the results for the promotion of real-time traffic with either the three measurement processes are analysed. The aim is to evaluate how fast the system reacts to a load increase, while the downgrade is done employing the threshold-based method, being the most conservative one (as intuitive and later verified).
The reported graphs depict the DSCP assigned to the packets of the issued traffic. It changes from the initial value 1, when the said traffic is in the worst class, till the value 4,

Table 1: Configuration parameters and their values.

Promotion	Downgrade
$K_{up}=2$	$K_{down}=20$
$N_{udp}=10$	$N=100$
$P_{up}=0.9$	$P_{down}=0.9$

Table 2: Reaction time with the different measurement processes.

Promotion method	***1st load increase [s]***	***2nd load increase [s]***
Threshold	2	8
Moving average	1.5	3.5
Low pass filter	2.2	12

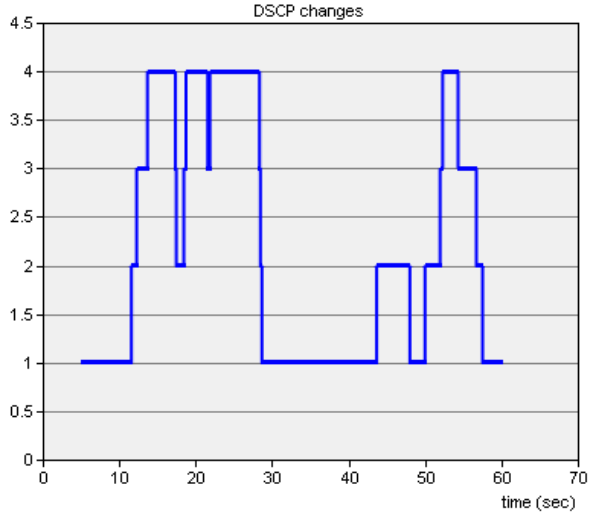

Figure 2: Promotion by moving average with $K_{up}=2$, downgrade by threshold with $K_{down}=20$.

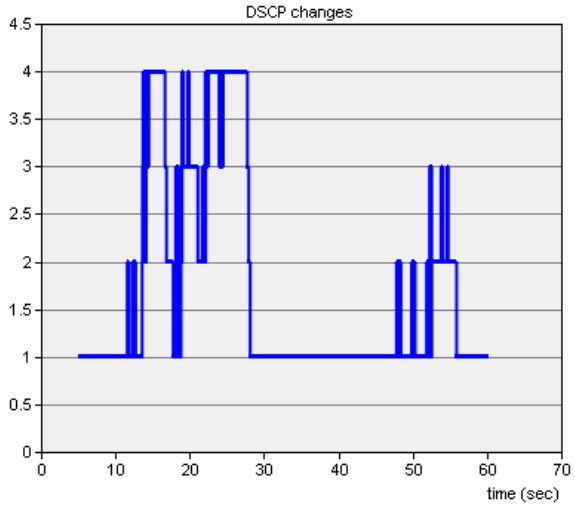

Figure 3: Promotion by moving average with $K_{up}=2$, downgrade by moving average with $K_{down}=20$.

when the issued traffic is in the best class.

In this set of simulations, the configuration parameters were as in Table 1, for a good compromise between stability and fast adaptation. It is worthwhile to point out that the number of samples of the 99th percentile of the end-to-end delay of the issued traffic that are considered for a promotion, is K_{up} equals 2 only (this avoids negative memory effects on the decision process).

The system can be considered stable (actually, consistent and efficient as well) when the real-time traffic remains in the lowest class (the one initially assigned at minimum) addressing its delay requirement, while the traffic load does not change on average.

The reaction time (time passed from the load increase and the first re-assignment to the subsequent upper class) at each traffic addition is inserted in Table 2.

The moving average (see also Table 2) shows the best reaction time in particular at the second load increase, while the low pass filter appears the slowest method.

Such a conclusion still holds when analysing the results collected with different configuration parameters for the measurement processes, which still allow for the system stability.

Once fixed the promotion method of the critical traffic to moving average with $K_{up}=2$, the different methods for the downgrade are analysed. Again, how fast the system reacts to traffic reduction and its stability (i.e. change of class with load dynamics) are to be evaluated.

Noticeably, the threshold-based method is very stable (Figure 2). When the network load increases at 10 s, the real-time traffic changes class, till it arrives in the best one. It remains in that class till the network load reduces slightly at 15 s and then it is promoted again and definitely, into the best class where it stays till the network congestion is finished at 25 s. Considering the second network load increase at 40 s, the issued traffic is first promoted, but then downgraded after some seconds because of a reduction in the network load, while

it is promoted when the load increases again at 50 s. The downgrades start at 55 s till the issued traffic returns to the first class. Reducing K_{down} to 10, the reaction time is shortened, but the drawback is that there are more changes of classes.
With the moving average oscillations appear every time there is a promotion (Figure 3). This is due to the fact that the calculation of the moving average made with 20 samples of the 99th percentile of the end-to-end delay can include a lot of them still below the threshold *D*. As a consequence, while the delay of the issued traffic is above threshold in the current class and hence a promotion is triggered, the moving average evaluated for the previous class can result in a period of time below threshold (the network load is variable in nature) and the issued traffic is then downgraded. This situation could repeat for some time after the first promotion. Indeed, by reducing the value of K_{down}, while yet keeping the system stable, a slightly lower number of class changes happen, but not enough to outperform the threshold-based method.
The low pass filter shows a similar behaviour to the moving average, though with fewer oscillations. The filter considers only a sample of the 99th percentile of the end-to-end delay of the lower class closest to the currently assigned, weighted according to the value of the filter pole, which is a configuration parameter and can be set properly. However the real-time traffic still changes frequently class, possibly jeopardizing the stability of the system, in particular when there is congestion (e.g. in the first load increase).
In the end, the downgrade decision for the real-time traffic in order to have a fairly stable system but also fast adaptation to load reduction and dynamics can be better made using the threshold method with K_{down} set to a value between 10 and 20, according to the desired response time of the system.

5.2 –Delay in feedback transmission and further investigations

After having identified the best measurement process for class adaptation (i.e. the moving average with K_{up} equals 2 and threshold with K_{up} equals 20 for the promotion and downgrade, respectively), further analysis takes into account the feedback transmission delay. Indeed, the feedbacks are generated at the receiver side by either the user terminal directly or the egress BR, and then sent to the issued ingress BR (for example, using a subscription/notification method [26] or some standard protocol over IP, preferably connectionless considering that they carry time-sensitive information, like UDP or ICMP [27]) across the network.
The performance stability and fast adaptation of the system are investigated with either 20, 50 or 100 ms as feedback delay, looking also at the peaks in the end-to-end delay of the real-time traffic.
Interestingly, with the shortest delay, the reaction time is quite the same as in the former case of no delay (refer also to Table 2).
Comparing the three analysed cases (Table 3), it is apparent that the system has very similar behaviour at the first load increase, while the performance is different at the second one. This is due to the fact that the introduced latency affects enough the system behaviour, being the processed values of the feedback quite different when the second load increase happens, leading to significantly diverse reaction time, and performance in the end.
As also confirmed by further analysis, the impact of the feedback transmission delay on the system adaptation strongly depends on the profile of the monitored traffic, but a short latency is always fairly negligible. In addition, it is worthwhile to point out that the feedbacks are control traffic and hence, according to best practices of network configuration, delivered with high priority. As a consequence, a latency of 20 ms is a realistic value even for large core networks.

Table 3: Reaction time with different feedback transmission delays.

Feedback delay [ms]	***1st load increase [s]***	***2nd load increase [s]***
20	2	4
50	2	10
100	2	6

Table 4: Relevant figures for the jitter evaluation.

Scenario	***Variance [ms²]***	***99th percentile [ms]***	***Peak-to-peak distance [ms]***
reference case (3 routers, 20 ms of feedback transmission delay)	9.23	10.45	17.49
reference case (3 routers, 50 ms of feedback transmission delay)	10.80	11.15	19.98
reference case (3 routers, 100 ms of feedback transmission delay)	11.87	11.64	20.61
6 routers (20 ms of feedback transmission delay)	32.70	17.00	26.90
Increased service distance between classes (20 ms of feedback transmission delay)	4.17	7.34	14.43

The designed solution to provide absolute QoS has been also evaluated in a network scenario composed of six routers (i.e. twice the number as in the former case). Accordingly, the delay threshold changes from 10 to 20 ms, as by SLA.

It happens that the system still quickly adapts to a load increase, but it is slightly more conservative than in the former case at a load decrease. Indeed, 6 s are needed for coming back to the initial (worse) class instead of 2 s. Actually, with a greater number of routers the delay at the receiver is also greater and a longer time is needed to update all the delay feedback values considered by the threshold-based method (set for the downgrade). However, it is to be highlighted that the downgrade is not critical for addressing the target performance.

Finally, the distance between the service classes in terms of relative QoS is augmented by assigning the quality factors 1, 4, 8 and 16 to the four queues at every router output interface, respectively. Therefore, the delay experienced in the first queue should be about four times the delay experienced in the second queue and eight times the delay in the third queue; while, the delay experienced in the second queue should be about twice the delay experienced in the third queue, and so on. The feedback transmission delay is set to the realistic value of 20 ms (see the first paragraph of this subsection).

In this new scenario, the system adapts quickly to the first load increase, showing the same behaviour as in the former case with quality factors equal 1, 2, 3 and 4, respectively for the

four queues. But, it is slightly slower at the second load increase, because it requires 5 s to react instead of 3.8 s. as also supported by other analysis, with another assignment of quality factors; the traffic is moved differently between the classes. Hence, it can happen that, when a load increase even of the same amount appears, the delay of the real-time traffic increases more or less quickly than in the former case.
Being the performance distance between the service classes enlarged, the best class(es) can provide a lower delay than before. As a consequence, also the range of values where end-to-end delay peaks mostly appear improves significantly. Indeed, a reduction from 18-20 to 12-14 ms is registered for the scenarios specified in the first and last rows of Table 4, respectively. Further results and analysis about the case of enlarged performance distance between service classes are included in the next subsection.

5.3 –Jitter discussion

In order to completely evaluate the designed system, a jitter analysis has been also carried out. The aim is to verify if the requirements of real-time multimedia applications for a proper user Quality of Experience (QoE) are satisfied.
In our proposal, jitter (i.e. delay-variations) can be introduce in the class change, with possible packet re-ordering in the case of promotion, when some packets could reach the egress BR faster than others previously transmitted. However, it is worthwhile to point out that by a buffering mechanism at the egress BR and a proper DSCP marking, the re-ordering phenomena can be avoided in the IP core network. Otherwise, the receiver (i.e. at RTP-level or audio/video decoder) will be in charge of it.
For the discussion on jitter, the statistical distribution, peak-to-peak distance, the 99^{th} percentile and variance of the end-to-end delay are considered for all the investigated scenarios.
Noticeably, an increase in the feedback transmission delay causes a slight worsening in the end-to-end delay, also in terms of jitter (see the first three rows of Table 4). However, as already pointed out such a delay should be small in practice. Therefore, its impact can be neglected.
As expected, the end-to-end delay and its variation increase with a higher number of routers to be crossed (consider for example, the first and fourth rows of Table 4). This clearly appears on the values of the 99^{th} percentile and variance, respectively. Nevertheless, the 99^{th} percentile and the peak-to-peak distance augment less then proportionally with the number of routers, and the former is indeed the reference performance parameter. This demonstrates that the design solution scales with the network size.
It is worthwhile to highlight that the achieved performance is slightly lower than the target (i.e. 10 ms) in the former three scenarios. This is simply due to the fact that for a period of time during the first load increase, the network is so congested that even the fourth (best) class is not able to fulfil the real-time traffic requirements, and this reflects on the delay figures. However, with an enlarged service distance between classes (compare the first and last rows of Table 4), the achieved performance, in terms of both 99^{th} percentile and delay-variation (i.e. variance and peak-to-peak distance) improves enough to fulfil the real-time traffic requirements. Indeed, in this case the fourth class is yet able to grant a delay lower than the reference threshold even under the worst load conditions. This also demonstrates the consistency of our proposal.

Last but not least, the maximum allowed jitter should be less than 100 ms for most real-time applications [28], and even less than 50 or 20 ms for high quality video [29], in terms of absolute difference between the delay of a packet and the average delay of all the packets of the considered application flow. Furthermore, depending on the specific application, a

bound should be respected by all packets or a high percentage of them (as for delay non-tolerant or tolerant applications, respectively). By analyzing the relevant figures about the jitter, we conclude that our solution can fulfil the QoS requirements of real-time traffic also in large networks and even when the real-time traffic needs to be moved up to the best (farthest) class in order to address the related SLA.

5.4 – Overhead evaluation and implementation issues

The final analysis regards the overhead that is introduced in the network when the proposed solution is implemented. Feedbacks can be transmitted by either UDP or ICMP [27] over IP. If the arrival time of a packet at the ingress BR is included in the packet itself, the issued egress BR can easily calculate the end-to-end delay experienced by such a packet (NGNs are synchronized, as already pointed out in Sect. 2). For the purpose, either a Hop-by-Hop or a Destination option of IPv6 [30] can be used.
The following design choices and assumptions can be made:

- the IPv6 option is 8 bytes long, containing a timestamp of 4 bytes;
- the feedback includes a flowID (20 bits) and an end-to-end delay value in the same format as for the timestamp (4bytes). Being the flowID the "Flow label" field of the IP header in the packets of the issued real-time traffic (a subset of bits could be reserved for identifying the ingress BR);
- a feedback is generated every 10 packets of the real-time traffic carrying a timestamp;
- in the worse condition, a timestamp is included in every packet of the real-time traffic (actually, it can be included less frequently if the related packet rate of the issued flow is high enough, also as a compromise between a fine delay evaluation and introduced overhead);
- the real-time applications generate about 20% of the overall network traffic.

As a result, the overhead introduced on the whole by feedbacks and inserted IPv6 options is only 0.75% of the overall network traffic (i.e. fairly negligible).

6. A comparison with the Related Works

This section provides a comparison between our solution with the one proposed in Ref. [16] (briefly described in Sect. 2), which appears to be the most performing alternative in literature addressing QoS by relying on a PDM over NGNs.
Ref. [16] presents a system with a very fast reaction time in both the promotion and the downgrade. Indeed, the real-time traffic is moved into an upper class within 1 s from an increase of the estimated average delay, demonstrating ability to follow very well the dynamics of the monitored QoS parameter. However it comes with several oscillations of the issued traffic between classes before the assignment of the class where it remains as far as a different load condition originates. While, our system achieves the target performance with considerably fewer oscillations, without possibly jeopardising the stability of the network.
Ref. [16] points out that when just few flows leave the network, the delay experienced by the still active flows can augment, sometimes causing a class change even if not needed. This dysfunctional behaviour is avoided in our solution because a more reliable and consistent decision mechanism for class change has been designed, a different scheduler (i.e. AWTP) employed and the 99^{th} percentile of delay used as reference QoS parameter instead of the average delay, which is also dramatically of less interest for real-time applications (for further details about this, see Sect. 3 where it is also mentioned that our proposal can be generalized to whatever percentile).

Moreover, Ref. [16] highlights a packet re-ordering phenomena during the adaptation process. This happens also in our solution, but the jitter evaluation provided in Subsect. 5.3 clarifies that it is of a few ms, and therefore can be easily recovered by buffering techniques. While the jitter introduced by the considered alternative system appears higher, though a quantitative analysis is missing.
Last but not least, our proposal introduces a lower bandwidth overhead (i.e. an IPv6 option is added to real-time traffic packets instead of creating an entire probe packet) and most importantly, additionally functionality are included in the BRs only, without compromising the scalability of the DiffServ model (refer also to Sect. 2 for more details).

7. Conclusions

This paper presents a solution to provide absolute QoS guarantees for real-time applications In Next-Generation Networks (NGNs), relying on a Proportional Differentiation Model (PDM) over DiffServ architecture. The proposal starts from the relative end-to-end delay differentiation between service classes offered by PDM, and enhances it with a mechanism for the dynamic change of the assigned service class in order to fulfil absolute delay requirements, as defined by Service Level Agreement (SLA). More specifically, if the currently assigned class is not able to address a given end-to-end delay bound, the real-time traffic is promoted to the closest upper class. While, if the closest lower class to the currently used, yet meets the issued end-to-end delay requirement, the real-time traffic can be downgraded there. For the purpose, feedbacks about the actual end-to-end delay are sent from either the user terminal directly or the egress Border Router (BR) to the ingress BR.
From an implementation standpoint, Advanced Waiting Time Priority (AWTP) scheduler is employed at each router output interface and the 99^{th} percentile of the end-to-end delay is taken as the reference QoS parameter (as issued by real-time applications). However, our system can use any other scheduling discipline able to provide a proportional delay differentiation and satisfy whatever absolute delay requirement. Furthermore the designed solution inherits the controllability from PDM, allowing each Network Service Provider (NSP) to properly configure the performance distance between service classes.
Three different measurement processes for both the promotion and the downgrade of real-time traffic have been described and evaluated for the best trade-off between fast reaction to network load changes and system stability. By analysing the collected simulation results, adopting a moving average over 2 delay feedback values for the promotion and a threshold-based method considering 20 feedback values for the downgrade, the system reacts quite fast to relevant load changes, preventing the real-time traffic from oscillating between service classes. Indeed, as detailed in a discussion about jitter, the introduced delay-variation does not badly affect the Quality of Experience (QoE) of the end-users.
The proposed solution has demonstrated consistent in a wide variety of scenarios with different feedback transmission delay, number of routers to cross or performance distance between service classes.
Last but not least, the designed system can provide both relative and absolute delay guarantees over NGNs in a scalable manner, because relying on a PDM over DiffServ architecture with additional functionality in the BRs only.
The described approach is easy to implement and at a low cost in terms of computational and bandwidth overheads. Therefore, NSPs can aim at new business models for offering multimedia real-time applications at high quality to their clients with fairly low investments. The designed system supports the realization of end-to-end solutions with limited interaction between NSPs. Indeed, SLAs can be established between NSPs allowing for the definition and application of different policies within each administrative domain.

Future works regards the analysis of the achieved performance with other schedulers still providing a proportional delay differentiation between classes (e.g. a dynamic version of WFQ) and the addressing of other QoS requirements still in absolute terms, such as packet loss. Finally, a subjective evaluation of the resulting QoE by end-users for different types and grades of real-time multimedia application when our solution is deployed should be of interest.

References

[1] ITU Y.2001, "General overview of NGN", December 2004

[2] D. Mustill and P.J. Willis, "Delivering QoS in the next generation network - a standards perspective", BT Technology Journal, Volume 23 n° 2, April 2005.

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An architecture for Differentiated Services – RFC2475," IETF DiffServ WG, 1998.

[4] K. Nicholas, S. Blake, F. Baker, D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers – RFC2474" IETF DiffServ WG, 1998.

[5] K. Nichols, B. Carpenter, "Definition of Differentiated Services Per Domain Behaviors and Rules for their Specification - RFC3086", IETF DiffServ WG, 2001

[6] IETF NSIS WG charter, url: datatracker.ietf.org/wg/nsis/charter/

[7] R. Braden, L. Zhang and others, "Resource ReSerVation Protocol (RSVP)", IETF Standard Track, September 1997.

[8] C. Dovrolis and P. Ramanathan "A Case for Relative Differentiated Services and the Proportional Differentiation Model" in IEE Network, Volume 13, Issue 5, September/October 1999.

[9] Yuan-Cheng Lai, Wei-Hsi Li, "A novel scheduler for proportional delay differentiation by considering packet transmission time," IEEE Communications Letters, vol. 7, No. 4, pp. 189-181, April 2003.

[10] A.Martino, G. Panza, A. Pattavina and F. Valente, "QoS guarantees on DiffServ architectures with GPS-like scheduling disciplines", in 2007 Proc. IEEE/GLOBECOM Conf., p. 2672 – 2677.

[11] Y. Chen, C. Qiao, M. Hamdi and D. Tsang "Proportional Differentiation: a scalable QoS approach", Communications Magazine, IEEE June 2003, Volume 41, Issue 6, pp 52-58

[12] D.L. Pham, S. Sugawara, T. Miki, "A new scheduler for real-time applications in differentiated services networks", in 2005 Proc. IEEE/INFOCOM Conf., p. 2841 - 2846 vol. 4.

[13] K. M. Lim, J. Paik, J. Ryoo and S. Joo "Prediction Error Adaptation of Input Traffic for Absolute and Proportional Delay Differentiated Services", QShine'06 The Third International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks, August 2006.

[14] R. Madadian and M.H.Y. Moghaddam, "Fuzzy controller design for proportional loss differentiation services", in 2009 Proc. Fuzzy Systems, 2009. FUZZ-IEEE 2009, p. 1308 – 1313.

[15] C. Dovrolis and P. Ramanathan, "Dynamic Class Selection: from Relative Differentiation to Absolute QoS", in 2001 Proc. IEEE/Network Protocols Conf., p. 120-128.

[16] T. Nandagopal et al., "Delay Differentiation and Adaptation in Core Stateless Networks," in 2000 Proc. INFOCOM, Conf., vol. 2, pp. 421–30.

[17] W. Wu et al., "Forwarding a Balance between Absolute and Relative: A New Differentiated Services Model", in 2001 Proc IEEE Wksp. High Perf. Switching and Routing, pp. 250–54.

[18] IEEE 1588-2002, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," 2002.

[19] D. L. Mills, "Internet Time Synchronization: The Network Time Protocol," IEEE Trans. Commun., vol. 39, no. 10, Oct. 1991, pp. 1482–93.

[20] G. M. Garner et al., "IEEE 802.1 AVB and its Application in Carrier-Grade Ethernet," IEEE Commun. Mag., vol. 45, no. 12, Dec. 2007, pp. 126–34.

[21] J. Ferrant et al., "Synchronous Ethernet: A Method to Transport Synchronization," IEEE Commun. Mag., vol. 46, no. 9, Sept. 2008, pp. 126–34.

[22] url: www-tkn.ee.tu-berlin.de/research/trace

[23] url: tracer.csl.sony.co.jp/mawi/

[24] OPNET Technology Inc. web site, url: www.opnet.com

[25] M. Umlauft and P. Reichl, "Getting Network Simulation Basics Right – A Note on Seed Setting Effects for the ns-2 Random Number Generator", Lecture Notes in Electrical Engineering, 2009, Volume 44, 215-228

[26] T. Zahariadis, A. Jari and others, "Efficient Streaming in Future Internet", to be published in FIA Internet Book 2010.

[27] A. Conta and S. Deering, "Internet Control Message Protocol (ICMPv6) for Internet Protocol Version 6 (IPv6) Specification – RFC 2463", IETF Standard Track, December 1998.

[28] N. Laoutaris, B. Van Houdt, I. Stavrakakis, “Optimization of Packet Video Receiver under Different Levels of Delay Jitter: an Analytical Approach,” Performance Evaluation, Volume 55, p. 251-275, 2004.
[29] Jim Anuskiewicz, “Measuring jitter accurately,” Lightwave Online Feature articles, April 2008, url: www.lightwaveonline.com/featured-articles/measuring-jitter-accurately-54886317.html.
[30] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification, RFC 2460”, IETF Standard Track, December 1998.