Efficient Reservation-Based QoS Architecture

Rui Prior, Susana Sargento, Pedro Brandão, and Sérgio Crisóstomo

DCC & LIACC, Faculty of Sciences, University of Porto
Rua do Campo Alegre, 823, 4150-180 Porto, Portugal
{rprior,ssargento,pbrandao,slc}@ncc.up.pt

Abstract. This paper describes a new architecture that provides end-to-end QoS support, and analyses its performance in terms of QoS guarantees and scalability of the solution. This architecture introduces a scalable per-flow signalling model, using several techniques and algorithms developed in order to minimise the computational complexity. A label switching mechanism and an efficient timer implementation were developed with the goal of reducing the signalling processing overhead at each router. The underlying architecture is based on DiffServ and the resource reservation is performed for aggregates of flows at both core and access networks. The performance results presented in this paper show that this architecture is able to support both IntServ service models in high speed networks, minimizing the processing load in each network element.

1 Introduction

The Internet nowadays only supports best effort service. Since this kind of service cannot be mapped to the diversity of the applications and users requirements, several techniques have been proposed in order to introduce Quality of Service (QoS) support and service differentiation in the Internet.

The IETF proposed two main QoS architectures. The Integrated Services (IntServ) architecture [1] uses per-flow reservation, through the Resource ReSerVation Protocol (RSVP) [2], and provides strict QoS guarantees and efficient resource usage. However, it has several scalability problems, concerning the per-flow scheduling, classification and reservation procedures. The Differentiated Services (DiffServ) architecture [3] does not suffer from scalability problems: there are no per-flow resource reservations, flows are aggregated in classes according to specific characteristics, and services have a different treatment according to their class. However, without admission control mechanisms to limit the number of flows in the network, all flows belonging to a class may be degraded.

With the objective of benefiting from the virtues of both IntServ and DiffServ and mitigating its problems, several architectures have been proposed in the literature. However, none of these architectures ensures simultaneously the strict and differentiated QoS support and the maximization of the usage of network resources without scalability concerns. For example, the SCORE architecture [4] and its associated Dynamic Packet State (DPS), which consists on carrying the state information in the header of every packet, keeping the stateless character

of the network, imposes the same scheduling mechanisms in all routers and its computational complexity is still high. In the "Egress Admission Control" [5] architecture, only the egress routers perform resource management and admission control based on passive monitoring, but this architecture is not able to assure strict QoS guarantees. The probing schemes [6,7] have also the advantage of no network control required, but the flow setup times can be high, they introduce themselves congestion in the network, the measurements are imprecise, and they also suffer from a resource stealing problem. In [8] a framework is proposed for the operation of IntServ over DiffServ networks, where an entire DiffServ domain emulates a single network element in the end-to-end path, avoiding signalling processing inside the domain. Without the support of end-to-end signalling in the DiffServ network, the resource allocation is not optimised and admission control is imprecise. Finally, the aggregation of per-flow reservations, where the RSVP is extended to allow RSVP signalling messages to be hidden inside an aggregate [9] benefits from the fact that the signalling messages are only exchanged when the aggregate's bandwidth needs to be updated. However, aggregation implies a tradeoff: with high aggregation, more flows are rejected and the utilization decreases; with small aggregation the decrease in utilization is neglegible but the number of signalling messages remains high.

In this paper we propose a new architecture that provides end-to-end QoS support without the problems of the previously mentioned ones. More specifically, our proposed model does not impose a complex scheduling mechanism, supports both soft and strict QoS guarantees, optimises the resource allocation, and does not suffer from resource stealing problems. Moreover, it achieves the same QoS guarantees as the aggregation model without a trade-off between signalling and utilization. The developed architecture is based on scalable per-flow signalling and resource reservation for aggregates of flows at both transit (core) and access networks. Several techniques and algorithms have been developed aiming at the minimization of the computational complexity and, therefore, the improvement of the signalling scalability. More specifically, a label switching mechanism was developed with the goal of avoiding expensive lookups in flow reservation tables. A scalable implementation of expiration timers for soft reservations, with a complexity that is low and independent from the number of flows, was also developed. In terms of QoS guarantees, this paper shows that our architecture is able to support strict and soft QoS guarantees to each flow, irrespectively of the behavior of the other flows in the same and in different classes, and with increased scalability.

This paper is organized as follows. Section 2 presents a brief overview of the system architecture, the service differentiation and traffic control techniques, the label switching mechanism and the signalling protocol. In section 3 the performance results of the developed end-to-end QoS architecture are shown and analysed. In section 4, some considerations on performance and scalability are presented and, finally, section 5 presents the most important conclusions, and describes the future work to be performed and the extensions to be applied to the architecture.

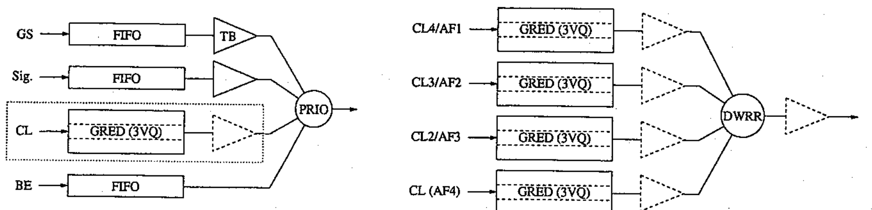
2 System Architecture

The developed architecture, described with more detail in [10], combines the strict end-to-end QoS guarantees of a signalling based approach with per-flow reservations subject to admission control, both in terms of bounded delay and minimal loss, with the efficiency and scalability provided by flow aggregation and by several mechanisms and algorithms developed.

The underlying architecture of the proposed model is strongly based on DiffServ (with which it may coexist) with the addition of signalling based reservations subject to admission control. The network is partitioned into domains, consisting of core and edge nodes. In addition, access domains have also access nodes. Individual flows are aggregated according to service classes, mapped to DiffServ compatible PHBs (Per-Hop Behaviors), and aggregate classification is performed based on the DS field of the packet header.

Besides best effort, our model provides two additional service classes: the Guaranteed Service (GS) class that is characterized by hard QoS assurance in terms of both delivery guarantee and maximum delay, based on the same principles as the EF (Expedited Forwarding) PHB in DiffServ; and the Controlled Load (CL) classes that emulate the behavior of a lightly loaded best effort network, based on the AF (Assured Forwarding) PHB. The simplest queuing model for the routers is depicted in figure 1. There are up to 4 different controlled load service classes using DSCPs from other AF classes, provided these are not used by DiffServ. In this case, the CL queuing block is replaced by the one shown in figure 1-b. Reservations for traffic flows using the GS class are characterized by a token bucket. Reservations for traffic flows using CL classes are characterized by three average rate water-marks: packets exceeding the first two water-marks will receive a degraded service in terms of drop probability; packets exceeding the third water-mark will be dropped. Admission control (described with detail in [10]) is performed at every node along the flow path, using different algorithms for GS and CL.

As can be seen in figure 1, the highest priority queue, corresponding to the GS traffic class, must be subject to a token-bucket type traffic shaper. The signalling/routing traffic, though not subject to admission control, must be shaped in order to prevent starvation of the CL class. The CL class may also be shaped,



a) Single CL class b) Multiple CL classes

Fig. 1. Queuing model

but this is only needed if the network administrator wants to make sure that some bandwidth always remains for best effort traffic. Contrary to the GS shaper, these ones are work-conserving.

All nodes in the architecture perform signalling and support the previously described queuing model. The access nodes perform per-flow policing for the CL class and per-flow ingress shaping for the GS class. Edge nodes perform aggregate policing and DSCP remarking. Core nodes perform no policing.

Probably the most scalability-limiting task for the core routers is the lookup of the stored flow information, based on the 5-tuple parameters that specify the flow, usually implemented using hash tables. In order to efficiently access the reservation structures we developed a label switching mechanism which allows direct access to these structures without any need for hash lookups. These labels are 32 bit values whose meaning is externally opaque, but internally may be an index to a table of reservation structures or the memory address of the reservation structure. Three label fields are stored in this structure: B, T and F. These label fields hold, respectively, the label to be used upstream (backwards), the label for the router itself[1], and the label to be used downstream (forwards), and are installed at reservation setup time.

The label switching mechanism has also strong advantages in all per-flow processing, like policing performed at the access routers. The labels may also be used to improve route change detection: a mismatch between the next hop assigned by the routing tables with the one stored in the reservation structure of the flow means that the route has changed. In order to profit from these advantages on per-flow processing, all packets would need to carry the label information. In [10], we present several proposals for the labels' insertion in the packet headers, both in IPv4 and IPv6. Notice that, in spite of these advantages, the labels are not used for packet classification (except maybe at the access routers), since it is performed on an aggregate basis, using just the DS field of the IP header.

The signalling protocol works on a hop by hop basis, providing unidirectional, soft state, sender initiated reservations. Although we have chosen to implement it as an extension to the RSVP protocol, it is much more scalable, since (1) the access to the flows' information is direct using the labels, (2) timers for the expiration of soft reservations are implemented in a very efficient way, and (3) it uses simple reservation identification in order to decrease the length of the refresh and explicit tear down messages. Since RSVP is meant to perform receiver initiated reservations, we had to extend it by adding three new message types: SResv (Sender Reservation), used to establish, refresh and modify reservations; SResvStat (Sender Reservation Status), used for reservation confirmation and error reporting; and SResvTear (Sender Reservation Tear Down), used to explicitly terminate a reservation. A detailed description of the signalling protocol is available in [10].

Full SResv messages include flow identification, reservation quantification, a LABEL_SETUP object (used to install the label), an identifier of the service

[1] The T label may be implicit.

class and a reservation expiration timeout value. The last two are conveyed by a SRESV_PARMS object. Upon receiving an initial SResv message, the request is subject to admission control; if accepted, the router updates the resource reservation of the flow's class, creates an entry in the reservation structure for the flow, stores the label at the B field for this reservation, and forwards the SResv message to the next router after changing the LABEL_SETUP to the reservation entry assigned to this flow. If the flow cannot be accepted (anywhere in the path), a SResvStat message is sent towards the sender reporting the error. This message already makes use of the labels in order to access to the flow reservation structure. When the SResv reaches the destination, all routers along the path have reserved resources for the new flow and all labels required for backward message processing are installed in the reservation state. The receiver acknowledges the successful reservation by sending a SResvStat message towards the sender, making use of the labels already installed in the opposite direction. The LABEL object in this message is used to access the memory structure for this reservation and the LABEL_SETUP object is stored in the F field of each node. Each node switches the LABEL to the one installed at the B field and forwards the message to the next node, until the sender is reached. Notice that the SResvStat message will also trigger the commitment of the resource reservation to both the policing and the queueing modules at the routers if the reservation succeeded, or the removal from the admission control module if it failed.

The reservations are soft state: if no SResvTear message is received and the reservation is not refreshed, the associated timer expires and it is removed. The basic implementation concept for timers is a sorted event queue: the processor waits until the first timer value in the list expires, dequeues it, performs the appropriate processing, then goes on waiting for the next timer value to expire. While dequeuing an event is trivial, inserting an event with a random expiration time is a very expensive operation, highly dependent on the total number of events queued. Contrasting to the complexity of generic timers, fixed delay timers are very simple and efficient to implement (a single FIFO queue). Trying to achieve some sort of balance between the two types, we have created an algorithm which has trivial timer queuing and a low and constant cost timer dequeuing, providing eight possible timer delays in a base 2 logarithmic scale, providing a range of 1:128. The implementation is based on eight different queues, each of which has an associated fixed delay. Internally, therefore, these queues are served using a FIFO discipline. Enqueuing an event is a simple matter of adding it to the tail of the corresponding queue, which is trivial. Dequeuing an event means choosing one of the eight possible queues (the one whose timers expires first) and taking the first event from that queue.

Having a good range of reservation expiration timer values means that short-lived flows will not remain stale for long times whenever something unusual occurs (such as an application lockup or premature termination, or an undetected route change) but longer-lived flows will not generate too much signalling traffic just to refresh the reservation. Figure 2 shows the relative weight of the refresh SResv messages in the total signalling traffic for flows with lifespans varying

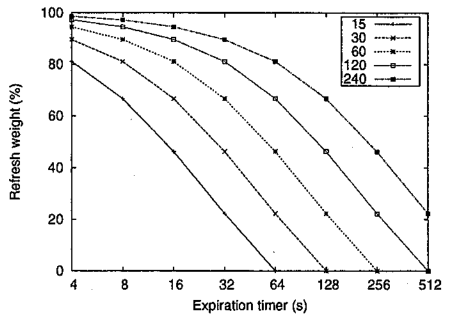


Fig. 2. Relative weight of refresh messages

from 15 s to 240 s using the eight possible different reservation timer values. The base timer is 4 s, and refresh messages are sent at a rate that is 4 times larger than the expiration timer rate to ensure that the reservation is correctly refreshed even in the presence of some signalling traffic losses. As can be seen, the weight of the refresh messages in the overall signalling traffic may vary from 0 to 98,6%, increases with the lifespan of the flows and decreases with the timer duration. Applications should use timer values proportional to the expected flow lifespan, representing a good tradeoff between signalling traffic and fast recovery from faults. When the lifespan cannot be estimated *a priori*, the application may use a short timer at first and increase it with refresh messages.

3 Performance Results

The architecture has been implemented using the ns-2 simulator, an extension of the Nortel DiffServ implementation and Marc Greis' RSVP patch. The DiffServ extensions implemented include, among others, the possibility of aggregate classification at the edge, the inclusion of multiple flows per node pair, dynamic modification of meter parameters for policing, and the configuration of the token bucket traffic shaper in priority mode in order to handle non-conformant aggregate GS traffic. These extensions are publicly available [11]. In this section we present the performance results of the end-to-end QoS architecture. These results mainly address the QoS guarantees achieved with the proposed model. Though very important in our model, processing efficiency measurement is out of the scope of this paper, since the ns-2 simulator is not suited to the evaluation of processing delays.

The simulated scenario is depicted in figure 3. It includes 1 transit and 5 access domains. Each terminal in the access domains simulates a set of terminals. The reason for having more than one access domain connected to an edge node of the access and transit domains is to check that correct aggregate policing is performed at the entry of the domain. The bandwidth of the connections in the transit domain, and in the interconnections between the transit and the access domains, is 10 Mbps. The propagation delay is 2 ms in the transit domain

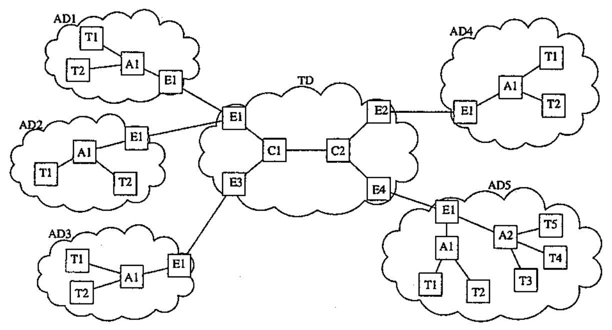


Fig. 3. Simulated scenario

connections and 1 ms in the interconnections between the access and the transit domain.

In this scenario we consider the coexistence of GS, CL and BE classes. At each referred connection, the bandwidth assigned to the signalling traffic is 1 Mbps. Note that, although this seems very high, the unused signalling bandwidth is used for BE traffic. The bandwidth assigned to the GS class is 3 Mbps, while for CL it is 4 Mbps. The remaining bandwidth is used for BE traffic. The bandwidth reserved for the GS and CL classes and left unused is also used for BE.

Each terminal of the access domains on the left side generates a set of flows belonging to the GS, CL and BE classes. Each source may generate traffic to all destinations; the destination of each flow is randomly chosen in the set of the terminals in the right side access domains. The traffic belonging to each class is a mixture of different types of flows.

All simulations presented in this paper are run for 5,400 simulation seconds, and data for the first 1,800 seconds is discarded. All values presented are an average of, at least, 5 simulation runs with different random seeds. The next sub-sections present the results of these experiments.

3.1 End-to-End QoS Guarantees

In this set of experiments we evaluate the end-to-end QoS guarantees of both GS and CL classes for different amounts of offered traffic in each class. In these experiments the set of flows is distributed in the following way (table1): (1) traffic in the GS class is composed by CBR (Constant Bit Rate) flows (Voice and Video256) and on-off exponential (Exp1gs) flows; (2) traffic in the CL class is composed by on-off exponential (Exp1cl) and Pareto (Pareto1cl) flows; and (3) traffic in the BE class is composed by on-off Pareto (Pareto1be) and FTP (Ftpbe) flows. The flows belonging to the BE class are active for the overall duration of the simulations (there are 3 FTP and 2 Pareto flows per source), while the flows in the other classes are initiated according to a Poisson process with a certain mean time interval between calls (MTBC), and each flow has

an average duration (Avg dur.) exponentially distributed. The characteristics of these flows are summarized in table 1.

The largest Mean Offered Load (MOL) in the GS and CL classes is, in terms of average traffic rates, about 20% higher than the bandwidth assigned to those classes, which, due to different mixes of flow types, translates, in terms of requested reserved rates (ROL - Requested Offered Load), in excess figures of 26% (GS) and 42% (CL). The values presented in the table correspond to this maximum offered load, which we will denote as a load factor of 1. For lower amounts of offered traffic, the mean time between flow generation events is increased in the inverse proportion of the offered load factor.

For GS flows, the reservation rate (Resv rate) represents the rate of the token bucket and the reservation burst (Resv burst) represents its depth. The reservation parameters provide a small amount of slack to compensate for numerical errors in floating point calculations. For CL flows, Low RR (Reservation Rate), Resv rate and High RR represent the three rate water-marks used for drop precedence selection and packet dropping at the policer. Admission control for the CL class in these simulations is parameter-based, with the utilization limits for the three rate water-marks set to 0.7, 1.0 and 1.7 times the bandwidth assigned to this class. The sum of the rates in each water-mark for all flows in the class must not exceed the respective utilization limits. Measurement-based admission control is a topic for further implementation. Notice that both scheduling and policing are performed on a per-class basis (except at the access routers).

Figures 4 (a, b and c) present the delay, jitter and loss, respectively, of both GS and CL flows when the offered load factor of the GS flows is 1 and the offered load factor of the CL flows increases from 0.5 to 1. As can be seen in the figures, the average delay remains very low and almost constant for all flow types, except for the GS exponential flows. For all except these, the delay is mostly the sum of transmission and propagation delays. GS exponential flows suffer an additional, and potentially large, delay at the ingress shaper of the access router when they send at a rate larger than what they requested for long periods of time. It is the applications' fault, though, for transmitting non-conformant traffic. The fact that the delay for the other GS flows remains very low shows that they are not adversely affected. The delay for CL flows remains almost constant, independently of the offered traffic. Jitter values exhibit a similar behavior for GS flows. On the other hand, jitter for CL flows increases somewhat with the offered CL load, which is expected due to the increased multiplexing. Regarding losses, they are null for well behaved GS flows. In CL flows, packet losses increase

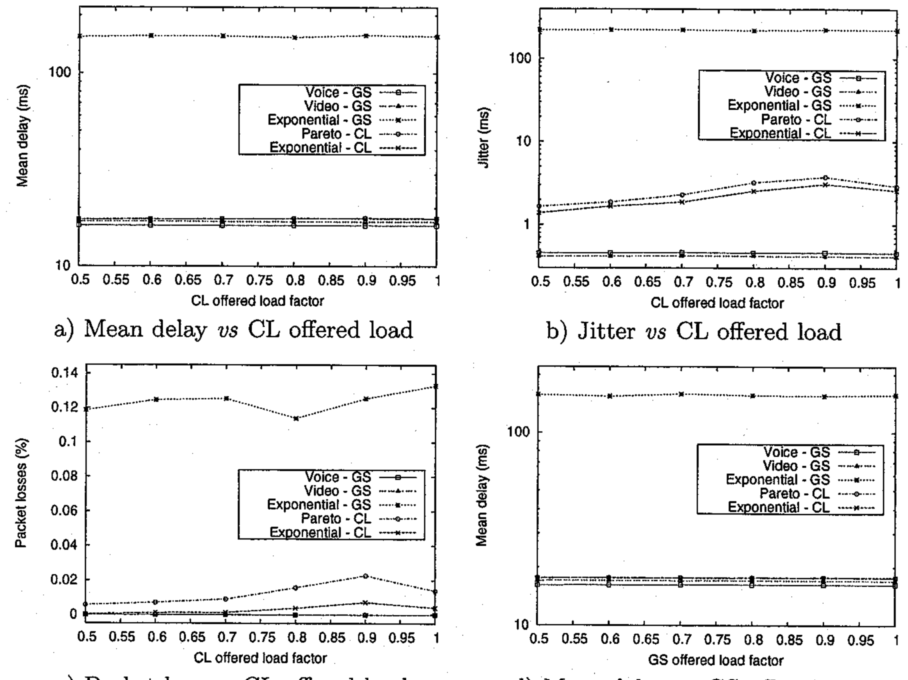
Table 1. Characteristics of the traffic flows

Class	Type	Peak rate (kbps)	On time (ms)	Off time (ms)	Avg. rate (kbps)	Pkt size (bytes)	Resv rate (kbps)	Resv burst (bytes)	Low RR (kbps)	High RR (kbps)	MTBC (s)	Avg dur. (s)	MOL (kbps)	ROL (kbps)
GS	Voice	48	-	-	48	80	48.048	81	-	-	45	120	768	769
	Video256	256	-	-	256	1000	256.256	1050	-	-	180	240	2048	2050
	Exp1gs	256	200	200	128	1000	160	5000	-	-	90	90	768	960
CL	Pareto1cl	256	200	200	128	1000	150	-	64	256	38	120	2425	2842
	Exp1cl	256	200	200	128	1000	150	-	64	256	38	120	2425	2842
											Simult. Flows			
BE	Ftpbe	-	-	-	-	1040	-	-	-	-	3 per src terminal		var.	N.A.
	Pareto1be	256	200	200	128	1000	-	-	-	-	2 per src terminal		2304	N.A.

with the offered load, but remain nevertheless very low (less than 0.03%). This means we should probably be more aggressive by reducing the requested rate water-marks for these flows. Losses for exponential GS flows are higher, though small (¡0.14%), and are due to buffer space limitation at the ingress shaper. At the core, the average utilization of the GS class is just below 2.5 Mbps (83%), and that of the CL class varies from 2.4 Mbps (60%) with a load factor of 0.5 to 3.1 Mbps (78%) with a load factor of 1, with a decreasing slope.

Figure 4 (d) presents the delay of both GS and CL flows when the offered load of the GS flows increases from 0.5 to 1 and that of CL flows remains constant at 1. The exponential GS flows exhibit larger delays compared to the one of the other flows, as expected due to ingress shaping. Jitter and losses, not shown here due to space limitations, have values comparable to the ones presented in the previous experiments, though they do not vary with the GS offered load. At the core, the average utilization of the GS class varies from 1.7 Mbps (57%) with a load factor of 0.5 to 2.5 Mbps (83%) with a load factor of 1, while that of the CL class remains constant at 3.1 Mbps (78%).

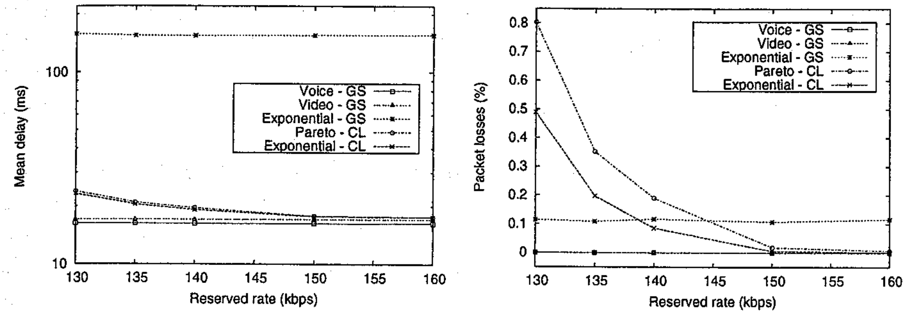
As was previously referred, we could be more aggressive on the requested rate of the CL traffic flows. In the next experiments we will analyse the effect, on the delay and packet losses of both GS and CL classes, of decreasing the requested rate. Figures 5 (a and b) show, respectively, the variation of the delay and packet



a) Mean delay *vs* CL offered load b) Jitter *vs* CL offered load

c) Packet loss *vs* CL offered load d) Mean delay *vs* GS offered load

Fig. 4. Loss, delay and jitter vs offered CL and GS load



a) Mean delay *vs* increasing reserved rate b) Packet loss *vs* increasing reserved rate

Fig. 5. Delay and packet losses with varying reserved rates for the CL flows

loss values with varying requested rates for CL flows. Here we have set the flow acceptance utilization limits of the three rate water-marks to 0.7, 1.0 and 2.0 times the bandwidth assigned to CL in order to ensure that flow admission would be performed based on the second rate water-mark, the varying factor in these experiments. Since the average rate for both types of CL flows used in this experiment is 128 kbps, we varied the requested rate from 130 kbps to 160 kbps, a little higher than the 150 kbps used in the previous experiments. As a result, the average utilization of the CL class at the core decreased from 3.5 Mbps (88%) to 3.0 Mbps (75%).

The delay for CBR GS flows remains constant, and is approximately equal to the sum of transmission and propagation delays. Exponential GS flows experience a much higher delay due to the ingress shaper. As expected, the delay for CL flows decreases with the increasing requested rate, since the number of accepted flows is lower. Jitter figures, though not shown, have a similar variation pattern. The most interesting results for this group are the loss figures. Packet loss in GS flows is not affected by the CL reservations, being null for conformant flows. CL flows, on the other hand, exhibit increasing losses with decreasing requested rates. With a requested rate of 130 kbps, which is only 1.6% higher than the average transmission rate, packet loss for exponential CL flows is just below 0.5%, while for the heavier tailed Pareto it is slightly above 0.8%. This shows that the architecture is also able to support soft QoS guarantees.

This set of experiments shows that our model, though being aggregation-based, is able to support both strict and soft QoS guarantees and achieves complete independence between traffic classes.

3.2 Independence between Flows

In this sub-section we evaluate the performance of the architecture in the presence of misbehaved flows, that is, flows that send at a rate much higher than the one they requested for considerable periods of time. Moreover, we also analyse the influence of misbehaved flows on well behaved ones. In order to protect the network from these flows, the access router performs per-flow ingress shaping for GS class flows. This shaper absorbs multiplexing jitter from the terminal and ensures that the traffic injected into the network does not exceed the reserved parameters by absorbing application bursts above the requested bucket (of 5 packets in this case), thus protecting the other GS flows.

In this experiment, the mean offered load (MOL) for the GS class is 23% larger than its assigned bandwidth (table 2). GS class includes three types of flows: (1) a CBR flow (Video64) that is considered a well behaved flow; (2) a on-off exponential flow (Exp1gs) with a burstiness of 50% (average busy and idle times of 200 ms) and a peak rate of 256 kbps, that is considered a nearly well behaved flow, since it sends at a rate a little higher than it is requesting; and (3) a on-off exponential flow (Exp2gs) with varying burstiness and peak rate that is considered a misbehaved flow, since it sends at a rate much larger than the one it is requesting for considerable periods of time. Its burstiness is variable, from 50% to 12.5%, varying its peak rate between 256 kbps (average busy and idle times of 200 ms) and 1024 kbps (average busy and idle times of 50 ms and 350 ms, respectively). Notice that the sum of the average idle and busy times remains constant (400 ms), as does the average rate. It is the high mismatch between the requested rate and the peak transmission rate that turns Exp2gs flows into misbehaved ones.

Figure 6 (a and b) depicts the packet loss ratio and the mean delay for all three types of flows with increasing burstiness values of the misbehaved (Exp2gs) flows. We may observe that the packet loss of both well behaved (Video64) flows and nearly well behaved (Exp1gs) flows is, respectively, 0 and just above 0.1%. The packet loss of the misbehaved (Exp2gs) flows reaches 7.1% when its burstiness reaches 12.5%. With such a burstiness, the peak rate of this type of flows is much larger than the reserved rate, and a large number of packets is lost. However, this misbehavior does not affect the previous flows. The mean delay of the well behaved flows is very small, and is mainly due to transmission and propagation delays. The nearly well behaved flows have a constant average delay in the order of 160 ms, which is significantly larger than that of the well behaved ones. Notice that this type of flow has a peak bandwidth approximately 100 kbps larger than the requested one, and therefore the packets will experience some delay (and small amounts of losses) at the ingress shaper of the access routers when the sources transmit at the peak rate for longer periods of time. As expected, the misbehaved flows have a delay that increases with their burstiness. With a burstiness of 12.5%, this delay can reach more than 400 ms. Jitter curves, though not shown, exhibit the same behavior as the delay curves. Notice that since all GS flows are aggregated and use the same queue, internally served in

Table 2. Characteristics of the GS traffic flows

Class	Type	Peak rate (kbps)	On time (ms)	Off time (ms)	Avg. rate (kbps)	Pkt size (bytes)	Resv rate (kbps)	Resv burst (bytes)	Low RR (kbps)	High RR (kbps)	MTBC (s)	Avg dur. (s)	MOL (kbps)	ROL (kbps)
GS	Video64gs	64	-	-	64	500	64.064	501	-	-	75	240	1229	1230
	Exp1gs	256	200	200	128	1000	160	5000	-	-	75	120	1229	1536
	Exp2gs	var.	var.	var.	128	1000	160	5000	-	-	75	120	1229	1536

a FIFO fashion, the queueing delay is shared by all GS flows. Therefore the large delays for nearly well behaved and misbehaved flows are inflicted at the ingress shaper. This reaction against misbehaved flows (in terms of large delays and losses) is meant to protect the other GS flows. This way, well behaved flows preserve a constant and small delay and no packet losses irrespectively of the burstiness of the misbehaved flows. It is the applications' fault for requesting inadequate reservations in face of the traffic to be transmitted.

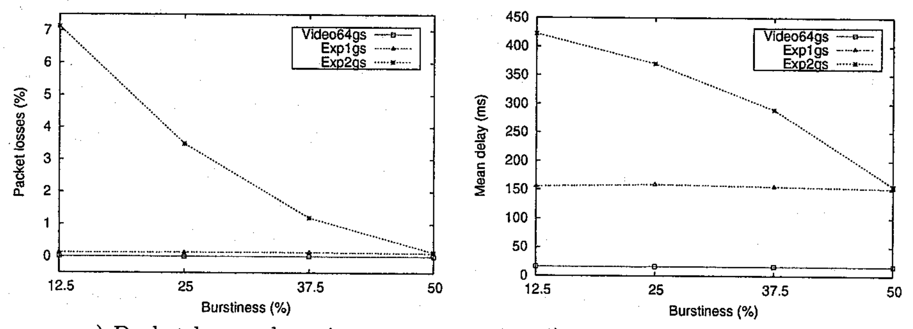
This experiment shows that the system reacts accordingly in the presence of misbehaved flows, keeping a complete independence between flows. These results are not unusual, and are to be expected in guaranteed service type classes. The main achievement of our model is to provide these guarantees in a scalable, aggregation-based architecture.

Due to space limitations, we do not present the same kind of experiments for the CL class. In terms of packet losses, the results are similar, though not providing absolute guarantees as in GS (packet losses are minimal, but not null). This protection is due to the (re)marking (and dropping) of the excess traffic at the access router. Forcing higher packet losses in excess traffic ensures that network congestion remains low, protecting well behaved flows. On the other hand, there is no delay penalty in misbehaved CL flows, since there is no shaping in this class. This means that the CL class is more appropriate for misbehaved flows with loose QoS requirements.

4 Considerations on Performance and Scalability

The previous section shows that the developed architecture is able to guarantee both soft and strict end-to-end QoS support and achieves independence between traffic classes. The performance in terms of QoS is, therefore, similar to that of the IntServ architecture.

Concerning scalability, the only quantifiable measures performed are the ones of the algorithm developed to implement efficient expiration timers. We have shown that our timer implementation, though very light in terms of processing,



a) Packet loss *vs* burstiness   b) Mean delay *vs* burstiness

Fig. 6. Architecture performance in the presence of misbehaved flows

is flexible enough to allow for a significant reduction in signalling traffic for long-lived flows while, at the same time, avoiding stale reservations for long periods of time in presence of application or network problems for short-lived ones.

The impact on scalability of aggregate-based classification and scheduling, trivial admission control and label switching mechanisms is obvious, but can only be quantified in a practical implementation. This implementation is a topic for further work.

The heaviest task in our model is the per-flow policing and ingress shaping at the access routers. Although this is not a huge problem, since the number of flows at the access routers is usually small, its complexity is reduced to O(1) if the labels are introduced in the data packet headers.

As a conclusion, we may state that our model achieves end-to-end QoS guarantees with per-flow signalling without the scalability concerns of IntServ.

5 Conclusions and Future Work

In this paper a new QoS architecture that scalably supports end-to-end QoS with strict and soft guarantees was proposed, and performance results were presented. This architecture includes per-flow signalling with enhanced scalability and resource reservation for aggregates of flows at both core and access networks, with an underlying DiffServ architecture. In order to improve the signalling scalability, several algorithms and techniques were introduced to reduce the signalling messages' processing at core and edge nodes, namely the label switching mechanism and the efficient implementation of expiration timers in soft reservations. Moreover, all mechanisms related to packet classification and scheduling are performed on a per-aggregate basis, making its complexity independent of the number of flows, and the admission control decisions are based on aggregate parameters trivially computed. The results presented in this paper show that this architecture is able to support both IntServ service models in high speed networks, minimizing the processing load at each network element.

As future work, we plan to improve the simulation model with measurement-based admission control in the controlled load class, and with the processing of route changes. We also plan to compare the performance of our model with that of others providing similar services and also aiming at high scalability. We are particularly interested in models providing reservation aggregation [9] over a DiffServ infrastructure. The expected outcome from these tests will be an improved network resource utilization with our model, while assuring the same quality of service. The drawback is increased signalling traffic, which we expect will be compensated by our efficient signalling processing.

In order to evaluate and quantify the scalability of the solution as compared to others, we plan to develop a prototype implementation.

Other topics for further research include the introduction of accounting and charging models, security and privacy models, the integration with mobility and wireless scenarios, and the possibility of interaction with QoS routing protocols and multicast.

References

1. Braden, R., Clarck, D., Shenker, S.: Integrated Services in the Internet Architecture: an Overview. RFC 1633, Internet Engineering Task Force (1994)
2. Braden, R., Zhang, L., Berson, S., Herzog, S., Jamin, S.: Resource Reservation Protocol (RSVP) - Version 1 Functional Specification. RFC 2205, Internet Engineering Task Force (1997)
3. Blake, S., Blake, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: An Architecture for Differentiated Services. RFC 2475, Internet Engineering Task Force (1998)
4. Stoica, I.: Stateless Core: A Scalable Approach for Quality of Service in the Internet. PhD thesis, Carnegie Mellon University (2000)
5. Cetinkaya, C., Knightly, E.: Egress Admission Control. In: Proceedings of IEEE INFOCOM 2000. (2000)
6. Breslau, L., Knightly, E., Shenker, S., Stoica, I., Zhang, H.: Endpoint admission control: Architectural issues and performance. In: Proceedings of ACM SIGCOMM 2000. (2000)
7. Sargento, S., Valadas, R., Knightly, E.: Resource Stealing in Endpoint Controlled Multi-class Networks. In: Proceedings of IWDC 2001. (2001) Invited paper.
8. Bernet, Y., Ford, P., Yavatkar, R., Baker, F., Zhang, L., Speer, M., Braden, R., Davie, B., Wroclawski, J., Felstaine, E.: A Framework for Integrated Services Operation over Diffserv Networks . RFC 2998, Internet Engineering Task Force (2000)
9. Baker, F., Iturralde, C., Faucheur, F.L., Davie, B.: Aggregation of RSVP for IPv4 and IPv6 Reservations. RFC 3175, Internet Engineering Task Force (2001)
10. Prior, R., Sargento, S., Cris stomo, S., Brand o, P.: End-to-end Quality of Service with Scalable Reservations. INternational Conference on Telecommunication System, Modeling and Analysis (accepted for publication) (2003)
11. Prior, R.: ns DiffServ extensions. http://www.ncc.up.pt/ rprior/ns/index-en.html (2003)