Traffic Management

Karlsson, G., Roberts, J., Stavrakakis, I. (Eds), Alves, A., Avallone, S., Boavida, F., D'Antonio, S., Esposito, M., Fodor, V., Gargiulo, M., Harju, J., Koucheryavy, Y., Li, F., Marsh, I., Mas Ivars, I., Moltchanov, D., Monteiro, E., Panagakis, A., Pescapè, A., Quadros, G., Romano, S., Ventre, G.

1 Introduction

Not written yet.

2 Traffic theory and flow aware networking

We argue here that traffic theory should be increasingly used to guide the design of the future multiservice Internet. By traffic theory we mean the application of mathematical modeling to explain the traffic-performance relation linking network capacity, traffic demand and realized performance. Traffic theoretic considerations lead us to argue that an effective QoS network architecture must be flow aware.

Traffic theory is fundamental to the design of the telephone network. The traffic-performance relation here is typified by the Erlang loss formula which gives the probability of call blocking, E, when a certain volume of traffic, a, is offered to a given number of circuits, n:

$$E(n, a) = \frac{a^n/n!}{\sum_{i \le n} a^i/i!}.$$

The formula relies essentially only on the reasonable assumption that telephone calls arrive as a stationary Poisson process. It demonstrates the remarkable fact that, given this assumption, performance depends only on a simple measure of the offered traffic, a, equal to the product of the call arrival rate and the average call duration.

We claim that it is possible to derive similar traffic-performance relations for the Internet, even if these cannot always be expressed as concisely as the Erlang formula. Deriving such relations allows us to understand what kinds of performance guarantees are feasible and what kinds of traffic control are necessary.

It has been suggested that Internet traffic is far too complicated to be modeled using the techniques developed for the telephone network or for computer systems [1]. While we must agree that the modeling tools cannot ignore real traffic characteristics and that new traffic theory does need to be developed, we seek to show here that traditional techniques and classical results do have their application and can shed light on the impact of possible networking evolutions.

Folowing a brief discussion on Internet traffic characteristics we outline elements of a traffic theory for the two main types of demand: streaming and elastic. We conclude with the vision of a future flow aware network architecture built on the lessons of this theory.

2.1 Statistical characterization of traffic

Traffic in the Internet results from the uncoordinated actions of a very large population of users and must be described in statistical terms. It is important to be able describe this traffic succinctly in a manner which is useful for network engineering.

The relative traffic proportions of TCP and UDP has varied little over at least the last five years and tend to be the same throughout the Internet. More than 90% of bytes are in TCP connections. New streaming applications are certainly gaining in popularity but the extra UDP traffic is offset by increases in regular data transfers using TCP. The applications driving these document transfers is evolving, however, with the notable impact over recent years first of the Web and then of peer to peer applications.

For traffic engineering purposes it is not necessary to identify all the different applications comprising Internet traffic. It is generally sufficient to distinguish just three fundamentally different types of traffic: elastic traffic, streaming traffic and control traffic. Elastic traffic corresponds to the transfer of documents under the control of TCP and is so-named because the rate of transfer can vary in response to evolving network load. Streaming traffic results from audio and video applications which generate flows of packets having an intrinsic rate which must be preserved by limiting packet delay and loss. Control traffic derives from a variety of signalling and network control protocols. While the efficient handling of control traffic is clearly vital for the correct operation of the network, its relatively small volume makes it a somewhat minor consideration for traffic management purposes.

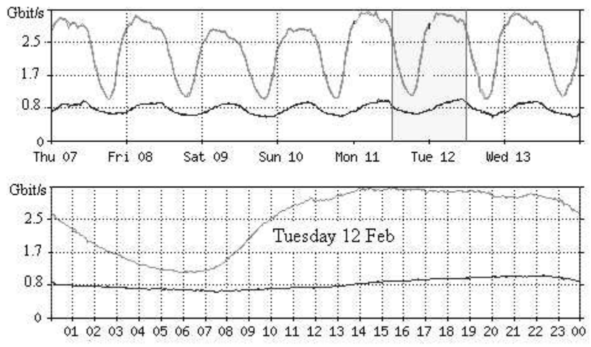


Fig. 1. Traffi c on an OC192 backbone link.

Observations of traffic volume on network links typically reveal intensity levels (in bits/sec) averaged over periods of 5 to 10 minutes which are relatively predictable from day to day (see Figure 2.1). It is possible to detect a busy period during which the traffic intensity is roughly constant. This suggests that Internet traffic, like telephone traffic, can be modelled as a stationary stochastic process. Busy hour performance is evaluated through expected values pertaining to the corresponding stationary process.

The traffic process can be described in terms of the characteristics of a number of objects, including packets, bursts, flows, sessions and connections. The preferred choice for modeling purposes depends on the object to which traffic controls are applied. Conversely, in designing traffic controls it is necessary to bear in mind the facility of characterizing the implied traffic object. Traffic characterization proves most convenient at flow level.

A flow is defined for present purposes as the unidirectional succession of packets relating to one instance of an application (sometimes referred to as a microflow). For practical purposes, the packets belonging to a given flow have the same identifier (e.g., source and destination addresses and port numbers) and occur with a maximum separation of a few seconds. Packet level characteristics of elastic flows are mainly induced by the transport protocol and its interactions with the network. Streaming flows, on the other hand, have intrinsic (generally variable) rate characteristics that must be preserved as the flow traverses the network.

Flows are frequently emitted successively and in parallel in what are loosely termed sessions. A session corresponds to a continuous period of activity during which a user generates a set of elastic or streaming flows. For dial-up customers, the session can be defined to correspond to the modem connection time but, in general, a session is not materialized by any specific network control functions.

The arrival process of flows in a backbone link typically results from the superposition of a large number of independent sessions and has somewhat complex correlation behaviour. However, observations confirm the predictable property that session arrivals in the busy period can be assimilated to a Poisson process.

The size of elastic flows (i.e., the size of the documents transferred) is extremely variable and has a so-called heavy-tailed distribution: most documents are small (a few kilobytes) but the number which are very long tend to contribute the majority of traffic. The precise distribution clearly depends on the underlying mix of applications (e.g., mail has very different characteristics to MP3 files) and is likely to change in time as network usage evolves. It is therefore highly desirable to implement traffic controls such that performance is largely insensitive to the precise document size characteristics.

The duration of streaming flows also typically has a heavy-tailed distribution. Furthermore, the packet arrival process within a variable rate streaming flow is often self-similar [2, Chapter 12]. As for elastic flows, it proves very difficult to precisely describe these characteristics. It is thus again important to design traffic controls which make performance largely insensitive to them.

2.2 Traffic theory for elastic traffic

Exploiting the tolerance of document transfers to rate variations implies the use of closed-loop control to adjust the rate at which sources emit. In this section we assume closed-loop control is applied end-to-end on a flow-by-flow basis using TCP.

TCP realizes closed loop control by implementing an additive increase, multiplicative decrease congestion avoidance algorithm: the rate increases linearly in the absence of packet loss but is halved whenever loss occurs. This behavior causes each flow to adjust its average sending rate to a value depending on the capacity and the current set of competing flows on the links of its path. Available bandwidth is shared in roughly fair proportions between all flows in progress.

A simple model of TCP results in the following well-known relationship between flow throughput B and packet loss rate p :

$$B(p) = \frac{\text{constant}}{\text{RTT}\sqrt{p}}.$$

where RTT is the flow round trip time (see [3] for a more accurate formula). This formula illustrates that, if we assume the loss rate is the same for all flows, bandwidth is shared in inverse proportion to the round trip time of the contending flows.

To estimate the loss rate one might be tempted to deduce the (self-similar, multifractal) characteristics of the packet arrival process and apply queuing theory to derive the probability of buffer overflow. This would be an error, however, since the closed-loop control of TCP makes the arrival process dependent on the current and past congestion status of the buffer. This dependence is captured in the above throughput formula.

The formula can alternatively be interpreted as relating p to the realized throughput B. Since B actually depends on the set of flows in progress (each receiving a certain share of available bandwidth), we deduce that packet scale performance is mainly determined by flow level traffic dynamics. It can, in particular, deteriorate rapidly as the number of flows sharing a link increases.

Consider the following fluid model of an isolated bottleneck link where flows arrive according to a Poisson process. Assume that all flows using the link receive an equal share of bandwidth ignoring, therefore, the impact of different round trip times. We further assume that rate shares are adjusted immediately as new flows begin and existing flows cease.

The number of flows in progress in this model is a random process which behaves like the number of customers in a so-called processor sharing queue [4]. Let the link bandwidth be C bits/s, the flow arrival rate λ flows/s and the mean flow size σ bits. The distribution of the number of flows in progress is geometric:

$$\Pr[n \text{ flows}] = (1-\rho)\rho^n,$$

where $\rho = \lambda\sigma/C$ is the link utilization. The expected response time $R(s)$ of a flow of size s is:

$$R(s) = \frac{s}{C(1-\rho)}.$$

From the last expression we deduce that the measure of throughput $s/R(s)$ is independent of flow size and equal to $C(1-\rho)$.

It is well known that the above results are true for any flow size distribution. It is further shown in [5] that they also apply with the following relaxation of the Poisson flow arrivals assumption: sessions arrive as a Poisson process and generate a finite succession of flows interspersed by think times; the number of flows in the session, flow sizes and think times are generally distributed and can be correlated. Statistical bandwidth sharing performance thus depends essentially only on link capacity and offered traffic.

Notice that the above model predicts excellent performance for a high capacity link with utilization not too close to 100%. In practice, flows handled by such links are limited in rate elsewhere (in the access network, by a modem,...). The backbone link is then practically transparent with respect to perceived performance.

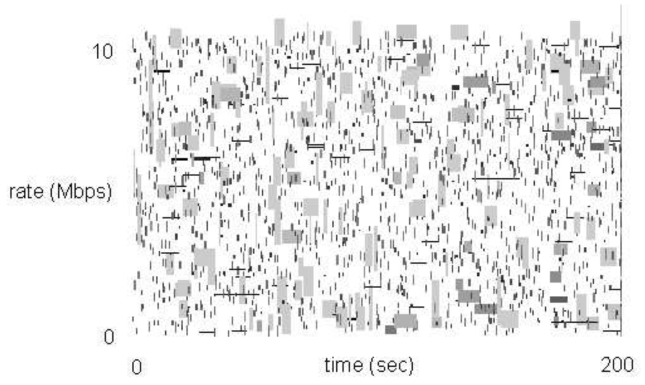


Fig. 2. Depiction of flow throughput for 90% link load.

The above model of ideally fair bandwidth sharing usefully illustrates two interesting points which turn out to be more generally true. First, performance depends primarily on expected traffic demand (in bits/second) and only marginally on parameters describing the precise traffic process (distributions, correlation). Second, backbone performance tends to be excellent as long as expected demand is somewhat less than available capacity. The latter point is illustrated in Figure 2.

The figure depicts the throughput of flows traversing a bottleneck link of capacity 10 Mbps under a load of 90% (i.e., flow arrival rate $\times$ average flow size = 9 Mbps), as evaluated by ns2 simulations. Flows are represented as rectangles whose left and right coordinates correspond to the flow start and stop time and whose height represents the average throughput. The flow throughput is also represented by the shade of the rectangle: the lightest grey corresponds to an average rate greater than 400 Kbps, black corresponds to less than 20 Kbps. In Figure 2, despite the relatively high load of 90%, most flows attain a high rate, as the model predicts.

In overload, when expected demand exceeds link capacity, the processor sharing queue is unstable: the number of flows in progress increases indefinitely as flows take longer and longer to complete while new flows continue to arrive. Figure 3 shows a rectangle plot similar to Figure 2 for an offered load equal to 140% of link capacity. The rectangles tend to become black lines since the number of competing flows increases steadily as the simulation progresses.

In practice, instability is controlled by users abandoning transfers, interrupting sessions or simply choosing not to use the network at all in busy periods. The end result is that link capacity is inefficiently used while perceived throughput performance becomes unacceptable, especially for long transfers [6]. An effective overload control would be to implement some form of proactive admission control: a new flow would be rejected whenever the bandwidth it would receive falls below a certain threshold. Figure 4 is the rectangle plot of the bottleneck link under 140% load with the application of admission control. Flow throughput is maintained at around 100 Kbps represented by the medium grey colour of the rectangles.

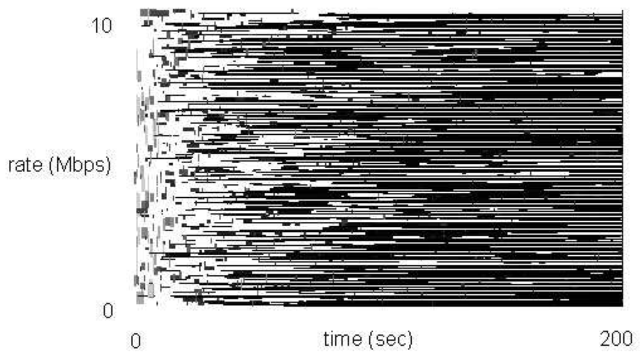


Fig. 3. Depiction of flow throughput for 140% link load.

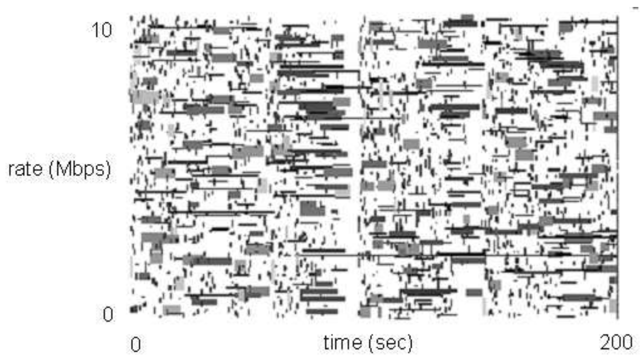


Fig. 4. Depiction of flow throughput with admission control.

The above traffic theory does not lead to an explicit traffic-performance relation showing how a provider can meet precise throughput guarantees. In fact, consideration of the statistical nature of traffic, the fairness bias due to different round trip times and the impact of slow-start on the throughput of short flows suggests that such guarantees are unrealizable. A more reasonable objective for the provider would be to ensure a link is capable of meeting a minimal throughput objective for a hypothetical very large flow. This throughput is equal to $C(1-\rho)$, even when sharing is not perfectly fair.

2.3 Traffic theory for streaming traffic

We assume that streaming traffic is subject to open-loop control: an arriving flow is assumed to have certain traffic characteristics; the network performs admission control, only accepting the flow if quality of service can be maintained; admitted flows are policed to ensure their traffic characteristics are indeed as assumed.

The effectiveness of open-loop control depends on how accurately performance can be predicted given the characteristics of audio and video flows. To discuss multiplexing options we first make the simplifying assumption that flows have unambiguously defined rates like fluids. It is useful then to distinguish two forms of statistical multiplexing: bufferless multiplexing and buffered multiplexing.

In the fluid model, statistical multiplexing is possible without buffering if the combined input rate is maintained below link capacity. As all excess traffic is lost, the overall loss rate is simply the ratio of expected excess traffic to expected offered traffic, i.e., $\mathrm{E}[(\Lambda_t - C)^+]/\mathrm{E}[\Lambda_t]$ where Λ_t is the input rate process and C is the link capacity. It is important to notice that this loss rate only depends on the stationary distribution of the combined input rate but not on its time dependent properties.

The level of link utilization compatible with a given loss rate can be increased by providing a buffer to absorb some of the input rate excess. However, the loss rate realized with a given buffer size and link capacity then depends in a complicated way on the nature of the offered traffic. In particular, loss and delay performance turn out to be very difficult to predict when the input process is self-similar (see [2, p. 540]). This is a significant observation in that it implies buffered multiplexing leads to extreme difficulty in controlling quality of service. Even though some applications may be tolerant of quite long packet delays it does not appear feasible to exploit this tolerance. Simple errors may lead to delays that are more than twice the objective or buffer overflow rates that are ten times the acceptable loss rate, for example.

An alternative to meeting QoS requirements by controlled statistical multiplexing is to guarantee deterministic delay bounds for flows whose rate is controlled at the network ingress. Network provisioning and resource allocation then relies on results from the so-called network calculus [7]. The disadvantage with this approach is that it typically leads to considerable overprovisioning since the bounds are only ever attained in unreasonably pessimistic worst-case traffic configurations. Actual delays can be orders of magnitude smaller.

Bufferless statistical multiplexing has clear advantages with respect to the facility with which quality of service can be controlled. It is also efficient when the peak rate of an individual flow is small compared to the link rate because high utilization is compatible with negligible loss

Packet queuing occurs even with so-called bufferless multiplexing due to the coincidence of arrivals from independent inputs. While we assumed above that rates were well defined, it is necessary in practice to account for the fact that packets in any flow arrive in bursts and packets from different flows arrive asynchronously. Fortunately, it turns out that the accumulation of jitter does not constitute a serious problem, as long as flows are correctly spaced at the network ingress [8].

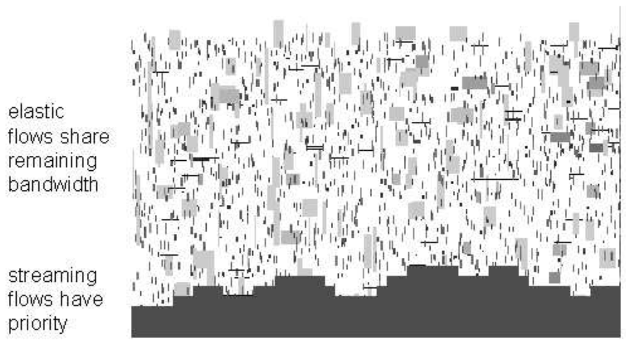

elastic flows share remaining bandwidth
streaming flows have priority

Fig. 5. Bandwidth sharing between streaming and elastic fbws.

Though we have discussed traffic theory for elastic and streaming traffic separately, integration of both types of flow on the same links has considerable advantages. By giving priority to streaming flows, they effectively see a link with very low utilization yielding extremely low packet loss and delay. Elastic flows naturally benefit from the bandwidth which would be unused if dedicated bandwidth were reserved for streaming traffic and thus gain greater throughput. This kind of sharing is depicted in Figure 5.

It is generally accepted that admission control must be employed for streaming flows to guarantee their low packet loss and delay requirements. Among the large number of schemes which have been proposed in the literature, our preference is clearly for a form of measurement-based control where the only traffic descriptor is the flow peak rate and the available rate is estimated in real time. A particularly simple scheme is proposed by Gibbens et al. [9]. In an integrated network with a majority of elastic traffic, it may not even be necessary to explicitly monitor the level of streaming traffic.

2.4 A flow aware traffic management framework

The above considerations on traffic theory for elastic and streaming flows lead us to question the effectiveness of the classical QoS solutions of resource reservation and class of service differentiation [10]. In this section we outline a possible alternative flow aware network architecture.

It appears necessary to distinguish two classes of service, namely streaming and elastic. Streaming packets must be given priority in order to avoid undue delay and loss. Bufferless multiplexing must be used to allow controlled performance of streaming flows. Packet delay and loss are then as small as they can be providing the best quality of service for all applications. Bufferless multiplexing is particularly efficient under the realistic conditions that the flow peak rate is a small fraction of link capacity and the majority of traffic using the link is elastic.

Elastic flows are assumed to fairly share the residual bandwidth left by the priority streaming flows. From results of the processor sharing model introduced above, we deduce that the network will be virtually transparent to flow throughput as long as overall link load is not too close to one. This observation has been confirmed by NS simulations of an integrated system [11].

Per flow admission control is necessary to preserve performance in case demand exceeds link capacity. We advocate applying admission control similarly to both streaming and elastic flows. If elastic traffic is in the majority (at least 90% at present), the admission decision could be based simply on a measure of the bandwidth currently available to a new elastic flow. If relative streaming traffic volume increases, an additional criterion using an estimation of the current overall streaming traffic rate could be applied as envisaged in [9].

Implementation of flow aware networking obviously requires a reliable means of identifying individual flows. A flow identifier could be derived from the usual microflow 5-tuple of IPv4. A more flexible solution would be to use the flow label field of the IPv6 packet header allowing the user to freely define what he considers to constitute a 'flow' (e.g., all the elements of a given Web page).

Flow identification for admission control would be performed 'on the fly' by comparing the packet flow identifier to a list of flows in progress on a controlled link. If the packet corresponds to a new flow, and the admission criteria are not satisfied, the packet would be discarded. This is a congestion signal to be interpreted by the user, as in the probe-based admission schemes discussed in Section XX. If admissible, the new flow identifier is added to the list. It is purged from the list when no packet is observed in a certain timeout interval.

Admission control preserves the efficiency of links whose demand exceeds capacity. Rather than rejecting excess flows, a more satisfactory solution would be to choose an alternative route. This constitutes a form of adaptive routing and would considerably improve network robustness and efficiency compared to that currently offered by Internet routing protocols.

Admission control can be applied selectively depending on a class of service associated with the flow. Regular flows would be rejected at the onset of congestion, premium flows only if congestion even then degrades to some higher degree. This constitutes a form of service differentiation with respect to accessibility.

Note finally that all admitted flows have adequate quality of service and are therefore subject to charging. A simple charging scheme is appropriate based on byte counting without any need to distinguish different service classes: streaming flows experience negligible packet loss and delay while elastic flows are guaranteed a higher overall throughput.

3 An IP Service Model for the Support of Traffic Classes

3.1 Introduction

The project that led to the development of the IP service model presented in this text started with the development of a metric for evaluating the quality of service in packet switched networks. Such a metric, presented in [12] and hereafter named QoS metric, is aimed at measuring quantifiable QoS characteristics [13] in communication systems, as throughput, transit delay or packets loss. It is especially tailored to the intermediary layers of communication systems, namely the network layer. During the metric development some ideas arose and were subsequently refined, progressively leading to a novel IP service model.

The central idea behind the proposed model is still to treat the traffic using the classical best effort approach, but with the traffic divided into several classes instead of a single class. This corresponds to a shift from a single-class best-effort paradigm to a multiple-class best-effort paradigm. In order to do this, a strategy which dynamically redistributes the communication resources is adopted, allowing classes for which degradation does not cause a significant impact to absorb the major part of congestion, thereby relieving the remaining classes.

Classes are characterized by traffic volumes, which can be potentially very different, and can be treated better or worse inside the communication system according to their needs and to the available resources. This means that classes may suffer better or worse loss levels inside the network and their packets may experience bigger or smaller transit delays. The model's capacity to differentiate traffic is achieved by controlling the transit delay and losses suffered by packets belonging to different classes, through the dynamic distribution of processing and memory resources. The bandwidth used by the traffic is not directly controlled, because the characteristics of the model make it unnecessary.

3.2 Service Model

The model proposal discussed here follows the *differentiated services* architecture [14] and considers that traffic is classified into classes according to its QoS needs. The central idea is still to treat the traffic using the *best effort* approach, but with the traffic divided into several classes instead of a single one. Thus, traffic is still treated *as well as possible*, but that now means different things according to the considered class.

In order to do that, a strategy is adopted which dynamically redistributes the communications resources, allowing classes for which degradation does not have a significant impact to absorb the major part of it, thereby relieving the remaining ones.

The model's capacity to differentiate traffic is achieved by controlling the transit delay and losses suffered by packets of the different classes (which is the goal of the dynamic distribution of resources). The bandwidth used by the traffic is not controlled. Given the characteristics of the model, this kind of control is not necessary (and not possible either).

In fact, as one of the main goals of the model is to avoid complexity, traffic specifications and explicit resource reservations are not considered. As a result, it is not possible to anticipate the bandwidth used by the different classes and, consequently, it does not make sense to base a strategy of differentiation on the active control of this QoS characteristic. Classes are characterized by very different volumes, which can be treated better or worse inside the communication system. This means that classes may suffer better or worse loss levels inside the network and their packets may experience bigger or smaller transit delays. Consequently, this model exerts its action controlling these two characteristics.

The proposed IP service model is based on the following three main components:

– network elements, comprising the resources and mechanisms that implement the *multiple-class-best-effort* service;
– dynamic QoS-aware routing, which accounts for the routing of traffic taking into account its QoS needs;
– communications system management, which accounts for the dimensioning and operability of the communication system, including traffic admission control functions, the main goal of which is to avoid scenarios of extreme high load.

The model presented here proposes the evolution of the *single-class-best-effort* paradigm, currently used in the Internet, into a *multiple-class-best-effort* paradigm. One of the most important challenges for building such a paradigm, given that the framework of this proposal is the IETF DS model, is the definition of a PHB able to rule the behaviour of network elements, named D3 PHB.

In general terms, traffic is divided into classes according to their sensitivity to transit delay and packets loss degradation. As a result of network elements behaviour, traffic with higher sensitivity to degradation is protected at the expense of less sensitive traffic. Thus, the idea is to dynamically and asymmetrically redistribute the degradation among the different classes, protecting some classes at the expense of others – hence the name D3, which stands for *Dynamic Degradation Distribution*.

The strategy for this effect is built on measuring continuously the quality of service given to each traffic class and, according to the obtained measures, adjusting the mechanisms responsible for packet processing dynamically. The question is which criterion should be used for a degradation distribution among classes that is considered reasonable or sensible?

The QoS metric mentioned above is particularly adequate for this purpose, as shown below. The metric's main principle is to evaluate the impact of the variations of QoS characteristics (not the QoS characteristics variations themselves) in one or the other direction with respect to the normal range of values. The metric defines degradations and superfluity zones which, in turn, define how the impact varies with QoS characteristics variations. It defines such zones for each QoS characteristic (e.g. transit delay or packets loss) and for each traffic flow or traffic class. Through the aggregation of finer measurements (e.g. the ones corresponding to a given QoS characteristic of a given flow) it is possible to construct broader measures (e.g. measurements corresponding to a given traffic class, constituted by a combination of a given set of QoS characteristics).

The D3 PHB was first presented in [15], from which Figure 6 has been extracted. In this figure (upper left corner), an example is presented of three classes with different sensitivity to transit delay degradation – high, medium and low, respectively.

The network elements' main goal is to guarantee that the impact of the transit delay and packet loss degradation on the applications is the same for all the three classes (to facilitate the presentation let us consider for now transit delay only). Therefore, network elements must control the transit delay suffered by packets of the three different classes in such a way that the congestion indexes related to this QoS characteristic are the same for all classes. Considering again Figure 6 as a reference, suppose that for a certain load level, which happens in the instant of time t1, this impact is evaluated by the index value CI_{t1}. Then, the transit delays suffered by the packets of each class are, from the most to the least sensitive, d1, d2 and d3, respectively.

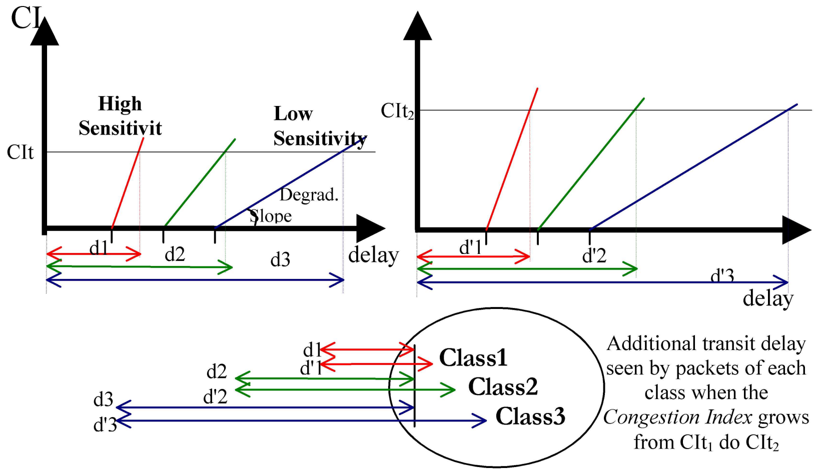


Fig. 6. Congestion indexes for three traffi c classes with different sensitivity to delay degradation.

In a more formal manner, the above mentioned goal means that the following equation holds for any time interval $[t_i, t_{i+1}]$:

$$\mathrm{CI_{avg}}{}_{Delay}^{Class\_1}[\mathrm{t_i}, \mathrm{t_{i+1}}] = \mathrm{CI_{avg}}{}_{Delay}^{Class\_2}[\mathrm{t_i}, \mathrm{t_{i+1}}] = \ldots = \mathrm{CI_{avg}}{}_{Delay}^{Class\_n}[\mathrm{t_i}, \mathrm{t_{i+1}}] = \mathrm{CI_{avg}}{}_{Delay}[\mathrm{t_i}, \mathrm{t_{i+1}}] \ . \qquad (1)$$

Exactly the same reasoning should be made if packet losses, instead of transit delay, are used as the QoS characteristic for evaluating the degradation of quality of service. In this case the formula which applies to any time interval $[t_i, t_{i+1}]$ is the following:

$$\mathrm{CI_{avg}}{}_{Loss}^{Class\_1}[\mathrm{t_i}, \mathrm{t_{i+1}}] = \mathrm{CI_{avg}}{}_{Loss}^{Class\_2}[\mathrm{t_i}, \mathrm{t_{i+1}}] = \ldots = \mathrm{CI_{avg}}{}_{Loss}^{Class\_n}[\mathrm{t_i}, \mathrm{t_{i+1}}] = \mathrm{CI_{avg}}{}_{Loss}[\mathrm{t_i}, \mathrm{t_{i+1}}] \ . \qquad (2)$$

Thus, equations 1 and 2 establish the formal criterion which rules the network elements' behaviour in respect to the way IP packets are treated. Through its simultaneous application, network elements control, in an integrated way, the transit delay and loss level suffered by the different classes.

To understand the way traffic from classes more sensitive to degradation is, in fact, protected from less sensitive traffic, let us consider again Figure 7. Its upper left corner represents a load situation that corresponds, as seen before, to a congestion index equal to CI_{t1}.

Suppose that at a given instant of time, t2, the load level to which the network element is submitted, rises. The impact of the degradation felt by the different applications (which generate traffic for the different classes) will also rise. The mechanisms of the network element will adjust themselves taking into account the criterion for resource distribution, that is, the values of such an impact (the congestion indexes) must be equal for all the classes. As can be seen in Figure 7 (right side), this corresponds to transit delays from the most to the least sensitive class of d'1, d'2 and d'3, respectively.

It is possible to see clearly in the figure that the value of (d'1-d1) is lower than the value of (d'2-d2) which, in turn, is lower than the value of (d'3-d3). Thus, the increase in transit delay suffered by packets of the different classes, when the load grows, is lower for classes that are more sensitive to transit delay degradation and greater for less sensitive classes. Hence, the degradation that happened at t2 was asymmetrically distributed among the different classes, the major part of it being absorbed by the ones less sensitive to degradation. Summing up, more sensitive classes are in fact protected at the expense of less sensitive classes, which is one of the most important goals of this proposal.

Finally, it is important to say that the control of the way in which some classes are protected at the expense of others, or even of the degradation part effectively absorbed by less sensitive classes, is materialized through the definition of the degradation sensitivity for each class (or, which is the same, through the definition of the size of each degradation zone).

3.3 An implementation of the D3 per-hop behaviour

It order to test the ideas put forward in the proposed IP service model, it was decided to build a prototype of the D3 PHB. This section describes this prototype.

The packet scheduler The basic idea of the work described in this section was the integration of the characteristics of the *work-conserving* (WC) and *non-work-conserving* (NWC) disciplines in one single mechanism, in order to obtain a new packet scheduler, which was simple but very functional – considering the characteristics of the PHB it would have to support – and able to effectively overcome the difficulties revealed by the experiments referred to in the previous sub-section.

Such a scheduler was first described in [16]. Figure 7 – extracted from that paper – presents the logical diagram of the scheduler. This figure shows the scheduler organised into two modules that reflect two stages of the controller's development.

Taking DEQUEUE_TIME as the instant of time after which the controller may process the next packet in a certain queue, XDELAY as the period of time that must pass between the processing of two consecutive packets in a certain queue (that is to say, the minimum time that a packet must wait in its queue), and TEMPO as the current system time, this is, concisely, how the scheduler works:

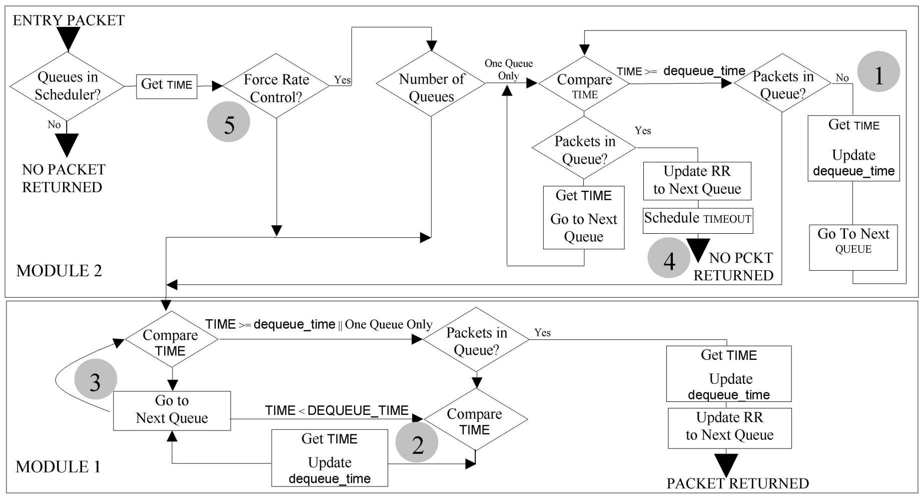


Fig. 7. Logical diagram of the scheduler implemented at LCT.

- it sequentially visits (*round-robin*) each IP queue, meaning, each class;
- on each visit it compares TEMPO against the value of the variable DEQUEUE_TIME that characterises the queue; if the first is greater than the second, the packet at the head of the queue is processed;
- on each visit it updates, if necessary, the queue's DEQUEUE_TIME (which is done by adding the value of the variable XDELAY[1] to TEMPO);
- for the most important class (*the reference class*), XDELAY is zero, which means that the scheduler behaves as a WC one. For all the other classes XDELAY will be greater than zero and will reflect the relative importance of each class. In this way, in these cases, the controller behaves as a NWC controller.

The packet dropper Having developed the packet scheduler, the next challenge became the conception of an integrated solution to control the loss of packets. In this text it has been referred to as the *packet dropper* (as opposed to the *packet scheduler*). In reality, the construction of the model demands far more that one packet dropper. It demands an active queue management strategy that allows not only an intelligent drop of the packets to be eliminated, but also an effective control of the loss level suffered by the different classes (without this, the operational principle of equality of the congestion indexes related to losses cannot be accomplished).

The queue management system was first presented in [17]. In general terms, the present system involves the storing of packets in queues and their discarding when necessary. It is composed of two essential modules – the *queue length management module* (QLMM) and the *packet drop management module* (PDMM).The network element's general architecture, including the queue management system and the packet scheduler, is presented in Figure 8.

[1] X_DELAY means the time, in μs, that the packet must wait in its queue.

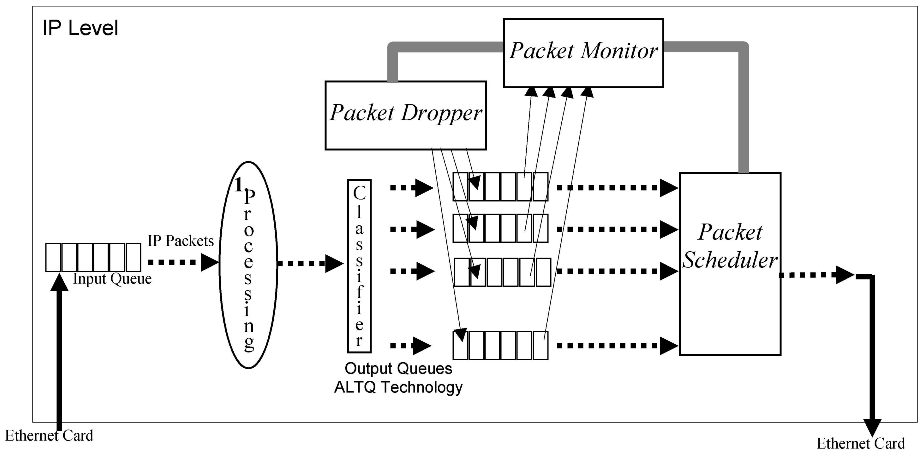


Fig. 8. Network element's general architecture [15].

3.4 Tests made to the D3 PHB implementation

The tests made to the D3 implementation can be subdivided into three main groups: robustness tests, performance tests and functional tests. During the development of the work presented here, tests pertaining to each of these groups were carried out.

The tests were carried out using both the QoStat tool [18] and netiQ's *Chariot* tool [19]. QoStat is a graphical user interface tool with the following characteristics:

- Graphical and numerical real-time visualisation of all values pertaining to the provision of quality of service by network elements, such as transit delay, number of processed packets, number of dropped packets per unit of time, queues' length, and used bandwidth.
- On the fly modification of all QoS-related operational parameters of network elements, such as maximum queue lengths, WFQ/ALTQ queue weights, virtual and physical queue limits, sensitivity of classes to delay degradation, and sensitivity of classes to loss degradation.

The tests made to the packet scheduler whose objective was to perform a first evaluation of the differentiation capability of the prototype, clearly showed the effectiveness of the scheduler. The tests' description and their results are presented in [16].

One example is the test carried out over a small isolated network composed of two source hosts independently connected through a router (with the new scheduler installed) to a destination host. All the hosts were connected through 100 Mbps Fast Ethernet interfaces and configured with queues with a maximum length of 50 packets. Two independent UDP flows composed of 1400 bytes packets were generated at the

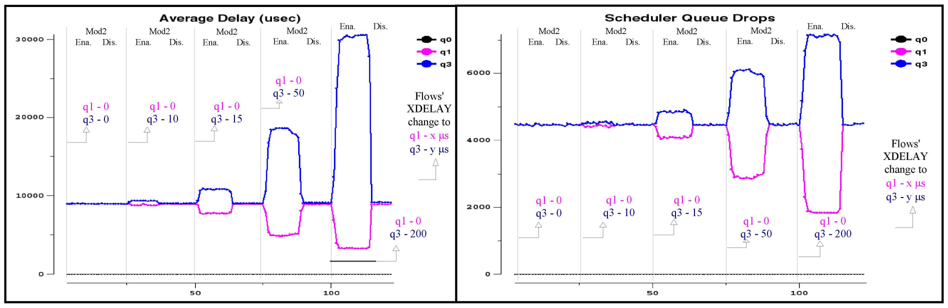


Fig. 9. left) Average IP transit delay (over 1-sec intervals); right) Number of packets sent per second

maximum possible rate, using the two different source hosts. They were associated to two different classes.

The general test strategy was the following: to set the X_DELAY associated with class 1[2] to zero, to vary the X_DELAY associated with class 3 (*qostat* tool was used to dynamically change those values, at the end of each 25 seconds time interval[3], from 0, to 10, 15, 50 and 200). Moreover, each 25 sec interval was divided into two parts. During the first part the scheduler module 2 (see Figure 7)[4] was activated; in the second part module 2 was deactivated.

Figure 9 shows the results of such a test. Its left part shows the average transit delay of packets at IP level, measured over 1-second time intervals. The right part the number of packets sent by the output interface per 1-second interval.

It is apparent that for the referred conditions the scheduler is able to differentiate traffic. It is also evident the importance of the scheduler module 2. Without the action of that module the scheduler is not able to differentiate traffic.

Tests made to the packet dropper, or, more precisely, on the active queue management system, were carried out in two distinct phases. In the first phase, the queue length management module (QLMM) was tested. The second phase tested the packet dropper management module (PDMM). These tests are presented in [15, 17] and they will be summed-up here.

Reference [16] presents tests that pertain to a scenario where the sensitivity to transit delay degradation is kept constant and the sensitivity to loss degradation varies over time. A variety of tests were performed. The results of one of these tests (using two classes) are shown in Figure 10 (fixed DSLOPE$_{losses}$[5], equal to 20^o for one of the flows and to 70^o for the other).

---

[2] Whose queue is named q1 in the fi gures.

[3] We did not use class 2 in the tests. As we are only using traffi c of classes 1 and 3 the measured values of class 0 are always null.

[4] This module guarantees that no packet can be processed, under any circumstances, before its DEQUEUE_TIME has come, even if there is only one class with outstanding packets.

[5] DSLOPE (*Degradation Slope*)

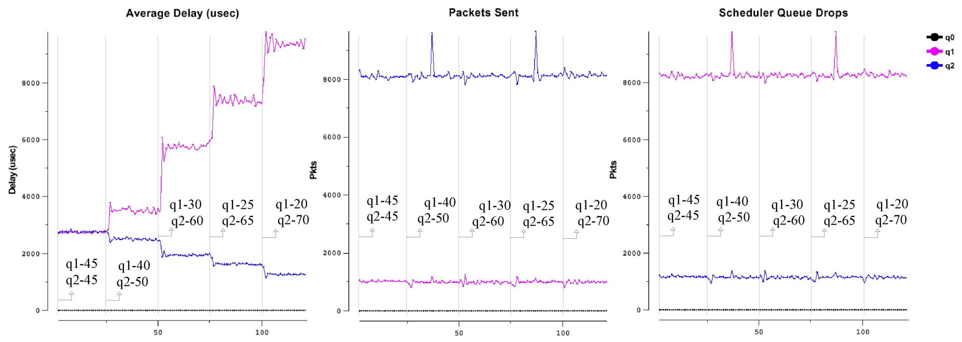


Fig. 10. Tests results in the following scenario: fi xed DSLOPElosses, equal to 20^o for the q2 fbw and to 70^o for the q1 fbw; delay sensitivity varied over time.

It is possible to observe that, under these conditions, the behaviour is as expected. The number of packets that get processed per unit of time, which is a complement of the number of dropped packets per unit of time, is constant and consistent with the respective loss class sensitivity, DSLOPE$_{losses}$. This happened in spite of the fact that transit delay was made to vary over time.

3.5 Conclusions and further work

One of the main objectives of the work presented in this paper was the conception of an IP service model based on the so called multiple-class-best-effort (mc-be) paradigm. This model is intended to deal with traffic classes taking into account their specific QoS needs, while still treating them *as well as possible* (without leaving aside the actual IP technology).

The applications do not have to specify the characteristics of the traffic they generate, nor explicitly request the QoS levels they need. Under heavy load conditions in the communication system (i.e. when performance degradation is likely to happen), they receive a better or worse treatment according to the traffic class they have chosen. This reflects the sensitivity to degradation pre-configured for each class, namely, the traffic sensitivity to delay degradation and the traffic sensitivity to the degradation of the level of packet losses.

The work presented here has strategically focused on the network element. This was achieved through the conception of a Per Hop Behaviour (PHB) capable of supporting the referred to model (the D3 PHB – *Dynamic Degradation Distribution*), and by the construction of the respective prototype.

The tests carried out on the prototype showed a very promising behaviour with regard to its capacity for differentiating the treatment applied to the classes in a coherent and controllable way. However, in order to characterise and evaluate the presented model in a more precise manner, it is clear that the prototype must be further tested, using network load conditions closer to those which actually happen inside IP networks. The development of other components of the model also fits into future work, namely, the development of the QoS-based routing component and the traffic admission control component which, as referred to above, are currently being developed at the LCT-UC [20–22].

Meanwhile, there is a set of contributions that stand out, as results of the work presented in this text: (1) a metric for the evaluation of QoS in IP networks; (2) an IP service model that supports the mc-be paradigm; (3) a PHB that supports the model, specifically considering the network element – the D3 PHB; (4) an implementation of such a PHB model involving the building of a packet scheduler and a queuing management system, which work together towards the D3 purposes.

4 Probe–based admission control in IP networks

4.1 Introduction

Today’s new applications on the Internet require a better and more predictable service quality than what is possible with the available best–effort service. Audio-visual applications can handle limited packet loss and delay variation without affecting the perceived quality. Interactive communication in addition requires stringent delay requirements. For example, IP telephony requires roughly speaking a maximum of 150 ms one–way delay that needs to be kept during the whole call.

The question of whether to provide the required service quality by over–provisioning network resources, or by admission control and reservation schemes, has been discussed extensively in the last years. In [23], Breslau and Shenker compare network performance and cost with over–provisioning and reservation. Considering non–elastic applications, their analysis shows that the amount of incremental capacity needed to obtain the same performance with a best–effort network as with a reservation–capable network diverges as capacity increases. Reservation retains significant advantages in some cases over over–provisioning, no matter how inexpensive the capacity becomes. Consequently, efficient reservation schemes can play an important role in the future Internet.

The IETF has proposed two different approaches to provide quality of service guarantees: Integrated Services (IntServ) [24] and Differentiated Services (DiffServ) [25]. IntServ provides three classes of service to the users: The guaranteed service (GS) offers transmission without packet loss and bounded end-to-end delays by assuring a fixed amount of capacity for the traffic flows [26]; the controlled load service (CLS) provides a service similar to a best–effort service in a lightly loaded network by preventing network congestion [27]; and, finally, the best–effort service lacks any kind of QoS assurances.

In the IntServ architecture, GS and CLS flows have to request admission from the network using the resource reservation protocol RSVP [28]. RSVP provides unidirectional per–flow resource reservations. When a sender wants to start a new flow, it sends a *path* message to the receiver. The message traverses all the routers in the path to the receiver, which replies with a *resv* message indicating the resources needed at every hop. This *resv* message can be denied by any router in the path, depending on the availability of resources. When the sender receives the *resv* message, the network has reserved the required resources along the transmission path and the flow is admitted. IntServ routers thus need to keep per–flow states and must process per–flow reservation requests, which can create an unmanageable processing load in the case of many simultaneous flows. Consequently, the IntServ architecture provides excellent quality in the GS class, and tight performance bounds in the CLS class, but has known scalability limitations.

The second approach for providing QoS in the Internet, the DiffServ architecture, puts much less burden on the routers, thus providing much better scalability. DiffServ uses an approach referred to as class of service (CoS), by mapping multiple flows into two default classes. Applications or ingress nodes mark packets with a DiffServ code point(DSCP) according to their QoS requirements. This DSCP is then mapped into different per–hop behaviors (PHB) at each router on the path, like expedited forwarding [29], or assured forwarding [30]. The routers additionally provide a set of priority classes with associated queues and scheduling mechanisms, and they schedule packets based on the per–hop behavior.

The drawback of the DiffServ scheme is that as it does not contain admission control. The service classes may be overloaded and all the flows belonging to that class may suffer increased packet loss. To handle overload situations, DiffServ relies on service level agreements (SLA) between DiffServ domains, which establish the policy criteria, and define the traffic profiles. Traffic is policed and smoothed at ingress points according to the SLA. Traffic that is out of profile (i.e. above the upper bounds of capacity usage stated in the SLA) at an ingress point has no guarantees and can be either dropped, over charged, or downgraded to a lower QoS class. Compared to the IntServ solution, DiffServ improves scalability at the cost of a less predictable service to user flows. Moreover, DiffServ eliminates the possibility to change the service requirements dynamically by the end user, since it would require signing a new SLA. Thus providing of quality of service is almost static.

Both IETF schemes provide end–to–end QoS with different approaches and thus with different advantages and drawbacks. Recent efforts focus on combining both schemes, like RSVP aggregation [31], the RSVP DCLASS object [32], or the proposal of the integrated services over specific link layer working group (ISSLL) to provide IntServ over DiffServ networks [33], that builds on RSVP as signaling protocol but uses DiffServ to actually share the resources among the flows.

4.2 Per–hop Measurement Based Admission Control Schemes

Recently, a set of measurement–based admission control schemes has appeared in the literature. These schemes follow the ideas of IntServ, with connection admission control algorithms to limit network load, but without the need of per–flow states and exact traffic descriptors. They use some worst–case traffic descriptor, like the peak rate, to describe flows trying to enter the network, and then to base the acceptance decision in each hop on real–time measurements of the individual or aggregate flows.

All these algorithms focus on provisioning resources at a single network node and follow some admission policy, like complete partitioning or complete sharing. The complete partitioning scheme assumes a fixed partition of the link capacity for the different classes of connections. Each partition corresponds to a range of declared peak rates, and the partitions cover together the full range of allowed peak rates without overlap.

A new flow is admitted only if there is enough capacity in its class partition. This provides a fair distribution of the blocking probability amongst the different traffic classes, but it risks lowering the total throughput if some classes are lightly loaded while others are overloaded. The complete sharing scheme, on the contrary, makes no difference among flows. A new flow is admitted if there is capacity for it, which may lead to a dominance of flows with smaller peak rate. To perform the actual admission control, measurement–based schemes use RSVP signaling.

The idea of measurement based admission control is further simplified in [34]. In this proposal the edge routers decide about the admission of a new flow. Edge routers passively monitor the aggregate traffic on transmission paths, and accept new flows based on these measurements.

An overview of several MBAC schemes is presented in [35]. This overview reveals that all the considered algorithms have similar performance, independently of their algorithmic complexity. While measurement–based admission control schemes require limited capabilities from the routers and source nodes, compared to traditional admission control or reservation schemes, like RSVP, they show a set of drawbacks: Not all proposed algorithms can select the target loss rate freely, flows with longer transmission paths experience higher blocking probabilities than flows with short paths, and flows with low capacity requirements are favored over those with high capacity needs.

4.3 Endpoint Admission Control Schemes

In the recent years a new family of admission control solutions has been proposed to provide admission control for controlled–load like services, with very little or no support from routers. These proposals share the common idea of endpoint admission control: A host sends probe packets before starting a new session and decides about the flow admission based on statistics of probe packet loss [36, 37], explicit congestion notification (ECN) marks [38–40], delay or delay variation [41–43]. The admission decision is thus moved to the edge nodes, and it is made for the entire path from the source to the destination, rather than per–hop. Consequently, the service class does not require explicit support from the routers, other than one of the various scheduling mechanisms supplied by DiffServ, and possibly the capability of marking packets.

In most of the schemes the accuracy of the probe process requires the transmission of a large number of probe packets to provide measurements with good confidence. Furthermore, the schemes require a high multiplexing level on the links to make sure that the load variations are small compared to the average load.

A detailed comparison of the different endpoint admission control proposals is given in [44], showing that the performance of the different admission control algorithms is quite similar, and thus the complexity of the schemes may be the most important design consideration. The following sections briefly summarize the three main sets of proposals.

Admission control based on probe loss statistics In the proposal from Karlsson *et al.* [36, 37, 45, 46] the call admission is decided based on the experienced packet loss during a short probe phase, ensuring that the loss ratio of accepted flows is bounded.

Delay and delay jitter are limited by using small buffers in the network. Probe packets and data packets of accepted flows are transmitted with low and high priority respectively, to protect accepted flows from the load of the probe streams. The probing is done at the peak rate of the connection and the flow is accepted if the probe packet loss rate is below a predefined threshold. This procedure ensures that the packet loss of accepted flows is always below the threshold value.

Admission control based on ECN marks The congestion level in the network in the proposal from F. Kelly *et al.* [39] and T. Kelly [40] is determined by the number of probe packets received with ECN marks by the end host. In this case, probe packets are transmitted together with data packets. To avoid network overload caused by the probes themselves the probing is done incrementally in probe rounds that last approximately one RTT, up to the peak rate of the incoming call. ECN–enabled routers on the transmission path set the ECN congestion experienced bit when the router detects congestion, e.g., when the buffer content exceeds a threshold or after a packet loss [38]. The call is accepted if the number of marked packets is below a predefined value. This proposal suggests that by running appropriate end–system response to the ECN marks, a low delay and loss network can be achieved.

The call admission control is coupled with a pricing scheme [47]. In this case users are allowed to send as much data as they wish, but they pay for the congestion they create (the packets that are marked). In this congestion pricing scheme the probing protocol estimates the price of a call, that is compared with the amount the end–system is willing to pay. The scheme does not provide connections with hard guarantees of service; it merely allows connections to infer whether it is acceptable to enter the network or not.

Admission control based on delay variations Bianchi *et al.* [43, 48] (first version published in [42]) propose to use measurements on the variation of packet inter–arrival time to decide about call admission, based on the fact that a non–negligible delay jitter can be observed even for accepted loads well under the link capacity. The admission control is designed to support IP telephony, thus it considers low and constant bit rate flows. The probe packets are sent at a lower priority than data packets. The probing phase consists of the consecutive transmission of a number of probe packets with a fixed inter–departure time. A maximum tolerance on the delay jitter of the received probe packets is set at the receiving node, and the flow is rejected immediately if the condition fails for one probe packet. The maximum tolerance on the delay jitter and the number of probe packets transmitted regulates the maximum level of accepted load on the network links. This maximum load is selected in a way such that the packet loss probability and end–to–end delay requirements for the accepted calls are met.

4.4 PBAC: Probe–Based Admission Control

As an example, we discuss the endpoint admission control procedure based on packet loss statistics in detail [36, 37, 45, 46]. This solution offers a reliable upper bound on the packet loss for the accepted flows, while it limits the delay and delay jitter by the use of small buffers in the routers.

The admission control is done by measuring the loss ratio of probe packets sent at the peak rate of the flow and transmitted with low priority at the routers. The scheduling system of the routers consequently has to differentiate data packets from probe packets. To achieve this, two different approaches are possible. In the first one there are two queues, one with high priority for data and one with low priority for probe packets (see Figure 11). In the second approach, there is just one queue with a discard threshold for the probes. Considering the double–queue solution the size of the high priority buffer for the data packets is selected to ensure a low maximum queuing delay and an acceptable packet loss probability, i.e., to provide packet scale buffering [49]. The buffer for the probe packets on the other hand can accommodate one packet at a time, to ensure an over–estimation of the data packet loss. The threshold–queue can be designed to provide similar performance, as it is shown in [45], and the choice between the two approaches can be left as a decision for the router designer.

Figure 12 shows the phases of the PBAC session establishment scheme. When a host wishes to set up a new flow, it starts by sending a constant bit rate probe at the maximum rate the data flow will require. The probing time is chosen by the sender from a range of values defined in the service contract. This range forces new flows to probe for a sufficient time to obtain a sufficiently accurate measurement, while it prohibits them from performing unnecessarily long probes. The probe packet size should be small enough so that there are sufficient number of packets in the probing period to perform the acceptance decision. When the host sends the probe packets, it includes the peak bit rate and the length of the probe, as well as a packet and flow sequence number in the data field of each packet. With this information the end host can perform an early rejection, based on the expected number of packets that it should receive not to surpass the target loss probability. The probe contains a flow identifier to allow the end host to distinguish probes for different sessions. Since one sender could open more than one session simultaneously, the IP address in the probes is not enough to differentiate them. When the probe process finishes, the sender starts a timer with a value over two times the expected round trip time. This timer goes off in case the sender does not receive an acknowledgement to the probe. The timer allows the sender to infer that none of the probe packets went through or the acknowledgement packet with the acceptance decision from the receiver got lost. The sender assumes the first scenario and backs off for a long period of time, still waiting for a possible late acknowledgement. In case a late acknowledgement arrives the sender acts accordingly and cancels the backoff process.

Upon receiving the first probe packet for a flow, the end host starts counting the number of received packets and the number of lost packets (by checking the sequence number of the packets it receives). When the probing period finishes and the end host receives the last probe packet, it compares the probe loss measured with the target loss and sends back an acknowledgement packet accepting or rejecting the incoming flow. This acknowledgement packet is sent back at high priority to minimize the risk of loss. If the decision is positive, the receiver starts a timer to control the arrival of data packets. The value of this timer should be slightly more than two RTTs. If this timer goes off, the host assumes that the acceptance packet has been lost and resends it.

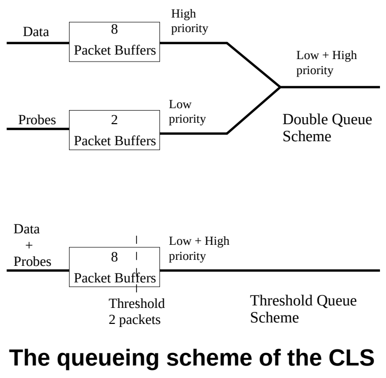


Fig. 11. The queueing system.

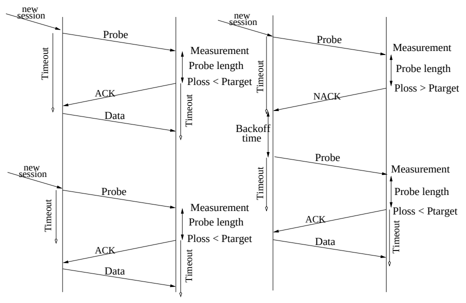


Fig. 12. The probing procedure.

Finally, when the sending host receives the acceptance decision, it starts sending data with high priority, or, in the case of a rejection, it backs off for a certain amount of time, before starting to probe again. In subsequent tries, the sender host can increase the probe length, up to the maximum level allowed, so that a higher accuracy on the measurement is achieved. There is a maximum number of retries that a host is allowed to perform before having to give up. The back off strategy and the number of retries affect the connection setup time for new sessions and should be carefully tuned to balance the acceptance probability with the expected setup delay.

The acceptance threshold is fixed for the service class and is the same for all sessions. The reason for this is that the QoS experienced by a flow is a function of the load from the flows already accepted in the class. Considering that this load depends on the highest acceptance threshold among all sessions, by having different thresholds all flows would degrade to the QoS required by the one with the less stringent requirements. The class definition also has to state the maximum data rate allowed to limit the size of the sessions that can be set up. Each data flow should not represent more than a small fraction of the service class capacity (in the order of 1%), to ensure that statistical multiplexing works well.

A critical aspect of end–to–end measurement based admission control schemes is that the admission procedure relies on common trust between the hosts and the network. If this trust is not present, security mechanisms have to protect the scheme. First, as end hosts decide about the admission decision based on measurements performed in the network, their behavior has to be monitored to avoid resource misuse. Second, as information on the call acceptance has to be transmitted from the receiving node to the source, intruder attacks on the transmission path altering this information have to be avoided. The specific security mechanisms of the general end–to–end measurement based admission control schemes can be addressed in different ways that are out of the scope of this overview, but a simple cryptographic scheme has been proposed in [50].

Application to Multicast A solution to extend the idea of end-to-end measurement based admission control for multicast communication is presented in [46]. The proposed scheme builds on the unicast PBAC process. The admission control procedure assumes a sender–based multicast routing protocol with a root node (*rendez-vous* point) implemented. The root node of the multicast tree takes active part in the probing process of the admission control, while the rest of the routers only need to have the priority–based queueing system to differentiate probes and data, as in the original unicast PBAC scheme.

In order to adapt the admission control for multicast communication two multicast groups are created: one for the probe process and one for the data session itself. Senders first probe the path until the root node of the multicast tree, and start to send data if accepted by this node. The probe from the sender is continuously sent to the root node, and is forwarded along the multicast tree of the probe group whenever receivers have joined this group.

Receivers trying to join the multicast data group first join the probe group to perform the admission control. They receive the probe packets sent by the sender node and forwarded by the root node, and decide about the admission based on the packet loss ratio. If the call is accepted the receiver leaves the probe group and joins the data group. Consequently, receivers have to know the addresses of both the probe and the data multicast group to take part in the multicast communication.

The unicast PBAC scheme is thus extended for multicast operation without additional requirements on the routers. The procedure to join a multicast group is receiver initiated to allow dynamic group membership. The scheme is defined to support many simultaneous or non–simultaneous senders, and it is well suited to multicast sessions with a single multimedia stream or with several layered streams.

4.5 Summary

This chapter presents an overview on probe–based admission control schemes. These solutions provide call admission control for CLS–like services. The admission control process is based on actively probing the transmission path from the sender to the receiver and deciding about the call acceptance based on end–to–end packet loss, packet marking on delay jitter statistics. As only the end nodes, sender and receiver, take active part in the admission control process, these mechanisms are able to provide per–flow QoS guarantees in the current stateless Internet architecture.

5 A component-based approach to QoS monitoring

The offering of Quality of Service (QoS) based communication services faces several challenges. Among these, the provisioning of an open and formalized framework for the collection and interchange of monitoring and performance data is one of the most important issues to be solved. Consider, for example, scenarios where multiple providers are teaming (intentionally or not) for the construction of a complex service to be sold to a final user, such as in the case of the creation of a Virtual Private Network infrastructure spanning multiple network operators and architectures. In this case, failure to provide certain required levels in the quality parameters should be met with an immediate attribution of responsibility across the different entities involved in the end-to-end provisioning of the service.

The same is also true in cases apparently much simpler, such as, for example, where a user is requiring a video streaming service across a single operator network infrastructure. In these situations there is also a need for mechanisms to measure the received quality of service across all of the elements involved in the service provisioning chain: the server system, the network infrastructure, the client terminal and the user application.

In described scenarios, the service and its delivery quality are negotiated through a contract, named Service Level Agreement (SLA), between a user and a service provider. Such a service provider is intended as an entity capable to assemble service contents as well as to engineer network and server side resources. The subscription of a Service Level Agreement implies two aspects, only apparently un-related: first, the auditing of the actual satisfaction of current SLA with the service provider; second, the dynamic re-negotiation of the service level agreements themselves.

As far as the first aspect, we can reasonably forecast that as soon as communication services with QoS or other service-related guarantees (e.g. service availability) are available, and as soon as users start to pay for them, it will be required to verify whether or not the conditions specified in the SLA are actually met by the provider. With reference to the second aspect, indeed, re-negotiation of QoS has been always accepted as an important service in performance guaranteed communications, strongly connected to critical problems such as the efficiency of network resource allocation, the end-to-end application level performance, and the reduction of communication costs. For example we might consider a scenario where the quality of service received by a distributed application can be seen as influenced by several factors: the network performance, the server load and the client computational capability. Since those factors can be varying in time, it is logical to allow applications to modify Service Level Agreements on the basis of the QoS achievable and perceivable at the application layer. We therefore believe that the possibility to modify the existing QoS based agreements between the service provider and the final user will assume an important role in Premium IP networks. Such networks provide users with a portfolio of services thanks to their intrinsic capability to perform a service creation process while relying on a QoS-enabled infrastructure. In order to allow the SLA audit and re-negotiation a framework for the monitoring of the received Quality of Service is necessary.

In this document, we propose a novel approach to the collection and distribution of performance data. The idea which paves the ground to our proposal is mainly based on the definition of an information document, that we called *Service Level Indication* (SLI). The SLI-based monitoring framework is quite simple in its formulation; nonetheless it brings in a number of issues, related to its practical implementation, to its deployment in real-life scenarios, and to its scalability in complex and heterogeneous network infrastructures. Some of these issues will be highlighted in the following, where we will also sketch some possible guidelines for deployment, together with some pointers to potential innovative approaches to this complex task.

The document is organized as follows. The reference framework where this work has to be positioned is presented in Section 5.1 . In Section 5.2 we introduce QoS monitoring issues in SLA-based infrastructures. In Section 5.3 we illustrate the data export process. Section 5.4 explains some implementation issues related to the proposed framework. Finally, Section 5.5 provides some concluding remarks to the presented work.

5.1 Reference Framework

This section introduces the general architecture proposed for the dynamic creation, provisioning and monitoring of QoS based communication services on top of Premium IP networks [51][52]. Such an architecture includes key functional blocks at the user-provider interface, within the service provider domain and between the service provider and the network provider. The combined role of these blocks is to manage user's access to the service, to present the portfolio of available services, to appropriately configure and manage the QoS-aware network elements available in the underlying network infrastructure, and to produce monitoring documents on the basis of measurement data.

Main components of the proposed architecture are the following: (i) **Resource Mediator(s)**: it has to manage the available resources, by configuring the involved nodes. Each service can concern different domains and then different Resource Mediators. Now, the Resource Mediator also has to gather basic monitoring data and export it; (ii) **Service Mediator(s)**: it is in charge of creating the service as required from the user, using the resources made available by one or more Resource Mediators. It has to map the SLA from the Access Mediator into the associated Service Level Specification (SLS) [4] to be instantiated in cooperation with the Resource Mediator(s); (iii) **Access Mediator(s)**: it is the entity that allows the users to input their requests to the system. It adds value for the user, in terms of presenting a wider selection of services, ensuring the lowest cost, and offering a harmonised interface: the Access Mediator presents to the user the currently available services.

5.2 A Monitoring Document: the Service Level Indication

Computer networks are evolving to support services with diverse performance requirements. To provide QoS guarantees to these services and assure that the agreed QoS is sustained, it is not sufficient to just commit resources since QoS degradation is often unavoidable. Any fault or weakening of the performance of a network element may result in the degradation of the contracted QoS. Thus, QoS monitoring is required to track the ongoing QoS, compare the monitored QoS against the expected performance, detect possible QoS degradation, and then tune network resources accordingly to sustain the delivered QoS. In SLA-based networks it becomes of primary importance the availability of mechanisms for the monitoring of service performance parameters related to a specified service instance. This capability is of interest both to the end-users, as the entities that 'use' the service, and to the service providers, as the entities that create, configure and deliver the service. QoS monitoring information should be provided by the network to the user application, by collecting and appropriately combining performance measures in a document which is linked to the SLA itself and which is conceived following the same philosophy that inspired the SLA design: i) clear differentiation of user-level, service-level and network-level issues; ii) definition of lean and mean interfaces between neighbouring roles/components; iii) definition of rules/protocols to appropriately combine and export information available at different levels of the architecture.

In Premium IP networks, the service provisioning is the result of an agreement between the user and the service provider, and it is regulated by a contract. The SLA is the document resulting from the negotiation process and establishes the kind of service and its delivery quality. The service definition stated in the SLA is understood from both the user and the service provider, and it represents the service expectation which the user can refer to. Such SLA is not useful to give a technical description of the service, functional to its deployment. Therefore, a new and more technical document is needed. The Service Level Specification document derives from the SLA and provides a set of technical parameters with the corresponding semantics, so that the service may be appropriately modelled and processed, possibly in an automated fashion. In order to evaluate the service conformance to specifications reported in SLA and SLS documents, we introduce a new kind of document, the Service Level Indication. By mirroring the hierarchical structure of the proposed architecture, it is possible to distinguish among three kinds of SLIs: (i) *Template SLI*, which provides a general template for the creation of documents containing monitoring data associated to a specific service; (ii) *Technical SLI*, which contains detailed information about the resource utilization and/or a technical report based on the SLS requirements. This document, which pertains to the same level of abstraction as the SLS, is built by the Resource Mediator; (iii) *User SLI*, i.e. the final document forwarded to the user and containing, in a friendly fashion, information about the service conformance to the negotiated SLA. The User SLI is created by the Service Mediator on the basis of the SLS, the Template SLI and the Technical SLI.

The service monitoring has to be finalized to the delivery of one or more SLI documents. In the SLI issue, multiple entities are involved, as network elements, content servers, and user terminals. Involving all these elements has a cost: it is due to the usage of both computational and network resources, needed for information analysis and distribution. This cost depends on both the number of elements involved and the information granularity. From this point of view, monitoring may be under all perspectives considered as a service, for which ad hoc defined pricing policies have to be specified and instantiated. More precisely, drawing inspiration from the concept of *metadata*, we might hazard a definition of monitoring as a *metaservice*, i.e. a ‘service about a service’. This definition is mainly due to the fact that a monitoring service cannot exist on its own: monitoring is strictly linked to a pre-existing service category, for which it provides some value-added information. Therefore we will not consider a standalone monitoring service, but we will rather look at it as an optional clause of a traditional service, thus taking it into account in the SLA negotiation phase.

5.3 Data Export

In the context of SLA-based services the following innovative aspect has to be considered: in order to allow users, service providers and network operators to have information about QoS parameters and network performance the need arises to export data collected by measuring devices. To this purpose, the concept of data model has to be introduced. Such model describes how information is represented in monitoring reports.

As stated in [53], the model used for exporting measurement data has to be flexible with respect to the flow attributes contained inside reports.

Since the service and its quality are perceived in a different fashion depending on involved actors (end user, service provider, network operator), there is a need to define a number of documents, each pertaining to a specific layer of the architecture, suitable to report information about currently offered service level. As far as data reports, we have defined a set of new objects aiming at indicating whether measured data, related to a specific service instance, is in accordance with the QoS level specified in the SLA.

With reference to our architecture, it is possible to identify the components responsible for the creation of each of the monitoring documents (Figure 13).

Such documents are then exchanged among the components as described in the following, where we choose to adopt a bottom-up approach:

1. at the request of the Service Mediator, the Resource Mediator builds the Technical SLI document on the basis of data collected by the measuring devices. The fields it contains are directly derived from those belonging to the SLS and are filled with the actual values reached by the running service. The resulting document is sent to the Service Mediator;
2. thanks to the Technical SLI received from the Resource Mediator, the Service Mediator is capable to evaluate the service quality conformance with respect to the requests formulated through the related SLS. It can be interested in such information both for its own business and in order to gather data for the creation of a complete report in case a user requests one;
3. at the user's request, the Service Mediator, exploiting data contained in a Technical SLI, produces a further report indicating the QoS level as it is perceived by the end user. The document it is going to prepare is derived from a service specific template (the so-called SLI Template), which provides an abstraction for the measurement results in the same way as the SLA Template does with respect to the service parameters. Such a document, hereby called User SLI, is ready for delivery to the end user;
4. the Access Mediator receives the User SLI from the Service Mediator, puts it in a format that is compliant with both the user's preferences and the user's terminal capabilities and forwards it to the end user.

5.4 Implementation Issues

Upper Layers A Service Mediator can add to the service offer the monitoring option. If this is the case, two different strategies are possible. In the first scenario, the Service Mediator evaluates the resource usage for monitoring service as a fixed cost. Such a cost will be taken into account when the quotation is prepared. This strategy does not modify the business process related to the specific service since the user's request to monitor its own SLA uniquely implies a fixed add-on to the service quotation. Such an add-on depends upon the business strategies of both the Service Mediator and the Resource Mediator.

In the second scenario, the Service Mediator shows two different quotations to the user: the first one related to the service instance selected by the user, the second one

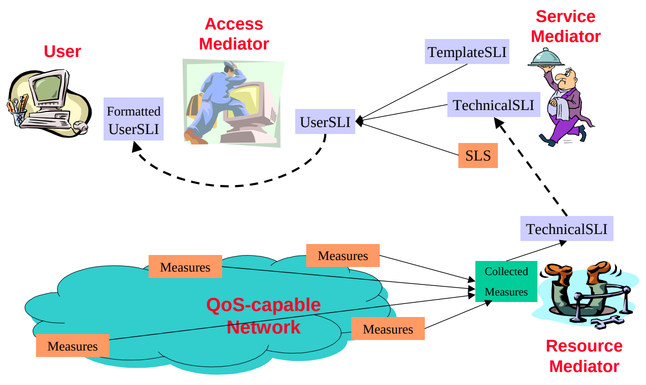


Fig. 13. Measurement data export.

regarding the monitoring activity. This solution introduces a scenario in which the negotiation of the SLA monitoring metaservice is considered. Interactions between the Access Mediator and the Service Mediator can be formalized through the same business process as that describing “traditional” services, such as VoD, VPN, and VoIP. Such a business process includes the following business transactions:

1. checking the service availability
2. submitting a quotation request
3. accepting a service quotation
4. issuing the purchase order

The definition of business processes can benefit from the availability of a standard proposal coming from the electronic business research community, named ebXML [54]. A detailed description of the application of the ebXML framework to the mediation architecture may be found in [55].

Lower Layers In Section 5.3 we have defined the roles of components in the creation of models to export measurement data. In particular, it is the responsibility of the Service Mediator to produce the User SLI intended to provide users with a high-level indication of the compliance (or not) of the delivered service to the negotiated SLA. The Resource Mediator, on the other hand, has to create the Technical SLI, which has the same fields as the corresponding SLS. These fields are filled by the Resource Mediator with the values resulting from the measurement activities.

At this point, the need arises to provide the Resource Mediator with monitoring data coming from the network devices so that it may be able to build the Technical SLI. In the proposed architecture the Resource Mediator is in charge of managing the whole underlying network for each domain (i.e. Autonomous System - AS) [56]. In particular, in the service configuration phase, it has to analyze the SLS, received from the Service Mediator, in order to select the subset of information related to its own domain. Such a subset is passed to the Network Controller, which translates it into a form that is compliant with the specific network architecture adopted (MPLS, Diffserv, etc.). The rest of the SLS is forwarded to the next peer Resource Mediator en-route towards the destination. If the user selects the monitoring option during the SLA negotiation, this choice does not modify the usual sequence of interactions concerning the service creation and configuration. In fact, once received from the Resource Mediator the subset of information contained in the SLS, the Network Controller translates it into a set of policy rules and, acting as a *Policy Decision Point* (PDP), sends policies to the underlying *Policy Enforcement Points* (PEPs), by exploiting the COPS protocol [57]. Upon the reception of a new policy, the PEP forwards it to the Device Controller, which produces configuration commands for the network device as well as rules enabling it to distinguish the traffic flow to be measured. Then, the Device Controller is able to appropriately configure the traffic control modules (e.g. allocation and configuration of queues, conditioners, markers, filters, etc.) and to perform the measurement activities.

Figure 14 depicts the policy framework components, distinguishing among the following abstraction layers: (i) **NI-DI**: Network Independent - Device Independent; (ii) **ND-DI**: Network Dependent - Device Independent; (iii) **ND-DD**: Network Dependent - Device Dependent.

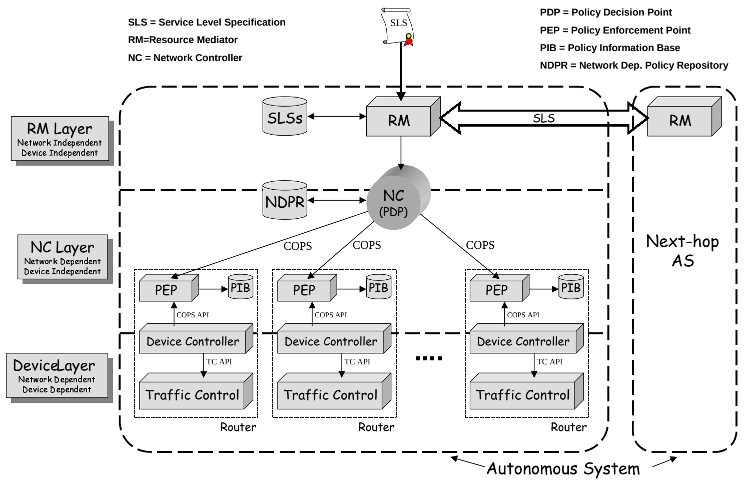


Fig. 14. Overview of the policy framework components.

Metering Approaches Several solutions are available in order to allow Device Controllers to make measurements on the traffic flows at controlled devices.

Our solution is based on the use of the SNMP protocol, which permits to obtain performance data without any traffic injection (passive measurement). In this case, an SNMP client interacts with an SNMP agent located at the device in order to obtain measures about throughput and packet loss on a device interface. The main advantage of such a solution is the capability to obtain information about every network device that supports the SNMP standard protocol. If involved devices are routers or switches made by Cisco, it is possible to exploit a particular Cisco IOS functionality, named NetFlow.

Exporting Data When the Device Controller, acting as a meter, obtains data about the controlled device, it has to forward it to the Resource Mediator. Using this raw data, the Resource Mediator can build the Technical SLI.

The Resource Mediator is able to create such an end-to-end document thanks to the interactions among the involved components, as described below:

1. The Device Controller sends the measurement results to the Policy Enforcement Point, e.g. using the COPS protocol [58]. This approach perfectly mirrors the policy-based architecture used at service configuration time.
2. Each PEP forwards received data to the Network Controller, which acts as a collector of measurement data from different measurement points;
3. Finally, the Resource Mediator, by gathering information from both its Network Controller and adjacent Resource Mediator, can build the Technical SLI. Such a document contains information about end-to-end performance metrics related to the traffic flow specified in the SLS.

5.5 Discussion and Conclusions

In this work, we have presented a component-based architecture for QoS measurement and monitoring. It is clear that the provisioning of such feature is particularly complex and critical, since it involves the coordination and orchestrated operation of a large number of elements, separately owned and managed along what we have called the provisioning chain from the service location to the end user. We therefore foresee a number of issues to be faced: for some of them we believe a solution can be already provided, while for others the discussion is still wide open. We briefly mention here the main facets of the general issue of QoS monitoring, focusing on the networking infrastructure. First of all, the collection of monitoring data from the network elements. This issue is clearly related to both technical and business aspects. As far as the first ones, the work ongoing in the area of policy based management of network elements is providing a technical framework in which the control and configuration of network nodes will be much more straightforward than that currently achievable through the traditional SNMP based approach. However, it is clear that for global communication infrastructures involving large number of nodes with a huge number of active connections we do have a problem of scalability with respect to the collection and delivery of performance data. In spite of this, we believe that there are features in the existing network architectures that might be exploited to reduce at least this problem. For example, in Diffserv based network architectures monitoring of Service Level Agreements can be performed usually per traffic classes and not per single traffic flows, and could be normally limited to the ingress and egress points of a domain. More detailed performance data collections (in terms of specific flows or network elements) could be triggered only in the presence of specific demands from the involved parties or in the case of anomalies. As far as the business aspects, i.e. those related to the business nature of the provisioning of communication services, we can mention here the one we believe is the most important:

trust. In global networks, large scale infrastructures will be managed by a multitude of different operators, each managing a separate network domain. Quality of Service will therefore be an issue involving a number of parties, each responsible only for the service provided in the domain that it directly manages. Such parties will be obliged, at the same time, to compete and to cooperate with peering entities. Can we foresee a scenario where such performance data will be openly (albeit in a controlled way) available? We believe that rather than being an obstacle to the deployment of a common framework for SLA monitoring, trust will be an important trigger for it, if not a prerequisite. In fact, we can expect that no operator will start charging for premium services involving infrastructures owned by others without a formal, standardized way for exchanging performance data about the communication services offered to and received from other operators.

A further issue is related to the devising of a common quality of service measurement framework. It is clear that performance data should be provided in a way that is independent of both the network architecture offering the service and the application service demanding it. Our proposal is a first attempt in this direction.

6 Deterministic delay guarantees under the GPS scheduling discipline

Under the GPS scheduling discipline traffic is treated as an infinitely divisible fluid. A GPS server that serves N sessions is characterized by N positive real numbers $\phi_1, ..., \phi_N$, referred to as weights. These weights affect the amount of service provided to the sessions (or, their bandwidth shares). More specifically, if $W_i(\tau, t)$ denotes the amount of session i traffic served in a time interval $(\tau, t]$ then the following relation will hold for any session i that is continuously backlogged in the interval $(\tau, t]$; session i is considered to be backlogged at time t if a positive amount of that session traffic is queued at time t.

$$\frac{W_i(\tau, t)}{W_j(\tau, t)} \geq \frac{\phi_i}{\phi_j}, j = 1, 2, ...N \tag{3}$$

The Generalized Processor Sharing (GPS) scheduling discipline has been widely considered to allocate bandwidth resources to multiplexed traffic streams. Its effectiveness and capabilities in guaranteeing a certain level of Quality of Service (QoS) to the supported streams in both a stochastic ([59–61]) and deterministic ([62–65]) sense have been investigated.

In this subchapter the single node case is considered and sessions (sources) are assumed to be (σ, ρ) leaky bucket constrained and to have a stringent delay requirement. Such sessions are referred to as QoS sensitive sessions in the sequel; the triplet (σ_i, ρ_i, D_i) is used in order to characterize the QoS sensitive session i, where σ_i, ρ_i and D_i represent the burstiness, the long term maximum mean arrival rate and the delay requirement of session i, respectively.

In the first part of the subchapter the problem of Call Admission Control (CAC) is considered. CAC plays a key role in provisioning network resources to meet the QoS requirements of traffic. It has the function of limiting the amount of traffic accessing the network and can have an important impact on network performance. A CAC algorithm is described which fully exploits the bandwidth sharing mechanism of GPS and determines the *optimal weights ϕ directly from the QoS requirements of the sessions*.

Nevertheless, the use of GPS scheduling discipline can lead to inefficient use of resources even if the optimal CAC scheme is employed. In the second part of the subchapter a service paradigm is described according to which each session is “represented” by two entities – referred to as session components – in the GPS server. Each session’s component is assigned a weight and it is served by the GPS server. The work provided by the GPS server to the components of a session is mapped back to the original session. It turns out that the proposed service scheme leads to resource utilization improvement.

Related Work The GPS scheduling discipline has been introduced in [62], [63] where bounds on the induced delay have been derived for single node and multiple nodes systems, respectively. These (loose) delay bounds have a simple form allowing for the solution of the inverse problem, that is, the determination of the weight assignment for sessions demanding specific delay bounds, which is central to the Call Admission Control (CAC) problem.

Tighter delay bounds have been derived in [65] and in [64] (also reported in [66]). These efforts have exploited the dependencies among the sessions -due to the complex bandwidth sharing mechanism of the GPS discipline- to derive tighter performance bounds. Such bounds could lead to a more effective CAC and better resource utilization. The inverse problem in these cases, though, is more difficult to solve. For example, the CAC procedure presented in [64] employs an exhaustive search having performance bound calculations as an intermediate step. Specifically, the maximum delay experienced by the sessions is determined for a weight assignment and the assignment is modified trying to maximize an objective function. While the search in [64] terminates after a finite number of steps, it does not guarantee that an acceptable assignment does not exist if not found.

The fairness oriented nature of GPS scheduling discipline, which follows directly from its definition, makes GPS an appealing choice for a best effort environment, or more precisely for an environment where fairness is the main concern. Some arguments on why fairness is not necessarily the main concern even in a best effort environment may be found in [67] where some state dependent scheduling policies aiming to decrease loss and / or delay jitter by “sacrificing” fairness are presented.

Modifications / extensions of the GPS scheduling discipline include ([68], [69], [70]). In [68] Service Curve Proportional Sharing (SCPS) has been proposed. It is a generalization of GPS, according to which the (constant) weighting factors used by the GPS are replaced with well defined varying weights. In [69] a less complex implementation is proposed for the special case of piecewise linear service curves. In [70] a modification of the GPS scheduling discipline, aiming to improve the QoS provided to adaptive sessions is presented, according to which each session is assigned two weights; one weight determines its guaranteed rate and the other weight determines its target rate.

6.1 Optimal CAC Algorithm

The optimal CAC algorithm for the GPS scheduling discipline has been derived in [71] by considering a mixed traffic environment in which the bandwidth resource controlled by the GPS server is assumed to be shared by a number of QoS sensitive streams and best effort traffic. This system will be referred to as a Best Effort Traffic Aware Generalized Processor Sharing (BETA-GPS) system.[6] The Best Effort Traffic Aware (BETA) GPS system is depicted in figure 15. The BETA-GPS server capacity C_G is assumed to be shared by N QoS sensitive sessions with descriptors $(\sigma_i, \rho_i, D_i), i = 1, \ldots, N$ and best effort traffic represented by an additional session. Each session is provided a buffer and the input links are considered to have infinite capacity. Generally, the task

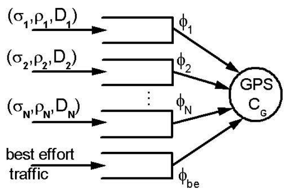


Fig. 15. The BETA-GPS system

of CAC is to determine whether the network can accept a new session without causing QoS requirement violations. In the case of a GPS scheduler it should also provide the server with the weight assignment which will be used in the actual service of the admitted calls. A CAC scheme for a GPS server is considered to be optimal if its incapability to admit a specific set of sessions implies that no ϕ assignment exists under which the server could serve this set of sessions (respecting all QoS requirements).

An optimal CAC scheme for the BETA-GPS system should seek to maximize the amount of service provided to the (traffic unlimited) best effort session under any arrival scenario and over any time horizon, while satisfying the QoS requirement of the (traffic limited) QoS sensitive sessions. That is, it should seek to maximize the normalized[7] weight assigned to the best effort traffic (ϕ_{be}), while satisfying the QoS requirement of QoS sensitive sessions. Obviously, maximizing the weight assigned to the best effort traffic is equivalent to minimizing the sum of weights assigned to the QoS sensitive sessions.

Optimal CAC Scheme for the BETA-GPS system In the sequel only the rationale of the optimal CAC algorithm is described and a numerical example is provided; the detailed algorithm may be found in [71].

The CAC problem for a GPS system is simplified in view of the following Theorem (Theorem 3, [62]): If the input link speed of any session i exceeds the GPS service

[6] In a system where only QoS sensitive sessions are present the existence of an extra session may be assumed and the presented algorithm be applied (see [71]).

[7] Without loss of generality, it is assumed that $\sum_{i=1}^{N} \phi_i + \phi_{be} = 1$

rate, then for every session i, the maximum delay D_i^* and the maximum backlog Q_i^* are achieved (not necessarily at the same time) when every session is greedy starting at time zero, the beginning of a system busy period. It is noted that a GPS system busy period is defined to be a maximal time interval during which at least one session is backlogged at any time instant in the interval. A session i is characterized as greedy starting at time τ if it sends the maximum allowable amount of traffic starting at time τ and an all-greedy GPS system is defined as a system in which all the sessions are greedy starting at time 0, the beginning of a system busy period.

The aforementioned theorem implies that if the server can guarantee an upper bound on a session' s delay under the all greedy system assumption this bound would be valid under any (leaky bucket constrained) arrival pattern. Thus, in the framework of the CAC problem it is sufficient to examine only all greedy systems.

It [71] it has been shown that a given acceptable ϕ assignment is converted to the optimal one if each QoS sensitive session's busy period is expanded as much as its QoS would permit, starting from the set of QoS sensitive sessions that empty their backlog first in order.[8] This implies that in order to determine the optimal ϕ assignment it is sufficient to allocate to the QoS sensitive sessions such weights that their QoS would be violated if their busy periods were expanded.

In order to determine these optimal weights the CAC algorithm of [71] emulates the all greedy system and examines the QoS sensitive sessions at all time instants coinciding with either the delay bound or the backlog clearing time of some session; these time instants are referred to as checkpoints in [71]. For each QoS session the algorithm computes two quantities (two potential weights); the minimum weight that is necessary for the QoS requirements of the session to be met *up* to the specific time instant and the weight that is necessary for the QoS requirements of the session to be met *after* the specific time instant, denoted as $\phi_i^-(\tau_j)$ and $\phi_i^+(\tau_j)$ at the checkpoint τ_j ,respectively. If $\phi_i^-(\tau_j) \geq \phi_i^+(\tau_j)$ session i is assigned a weight $\phi_i = \phi_i^-(\tau_j)$, since this is the minimum weight that could be assigned; else the specific session is examined again at the next checkpoint. In order to apply the aforementioned procedure the algorithm keeps track of the bandwidth that is available to the still backlogged sessions (a greedy (σ_i, ρ_i) constrained session requires a rate equal to ρ_i after it empties its backlog, and thus, the additional (compared to ρ_i) bandwidth that the session utilizes until it empties its backlog is shared among the still backlogged sessions in proportion to their weight).

Next a numerical example is provided, where the optimal CAC scheme for the BETA-GPS system is compared with the effective bandwidth-based CAC scheme. Deterministic effective bandwidth ([72]) can be used in a straightforward way to give a simple and elegant CAC scheme for the GPS scheduler. A similar approach is followed in [61] for the deterministic part of their analysis. The deterministic effective bandwidth of a (σ_i, ρ_i, D_i) session is given by $w_i^{eff} = \max\{\rho_i, \frac{\sigma_i}{D_i}\}$. It is easy to see that the requirements of the QoS sensitive sessions are satisfied if they are assigned weights such that $\phi_i C_G = w_i^{eff}$ ($\frac{\phi_i}{\phi_j} = \frac{w_i^{eff}}{w_j^{eff}}, \forall i, j \in QoS$).

[8] A ϕ assignment is characterized as acceptable if it is feasible (that is $\sum_{i=1}^{N} \phi_i < 1$)and delivers the required QoS to each of the supported QoS sensitive sessions; a ϕ assignment is characterized as more effi cient than another if the sum of ϕ's$\sum_{i=1}^{N} \phi_i$ under the former assignment is smaller than that under the latter.

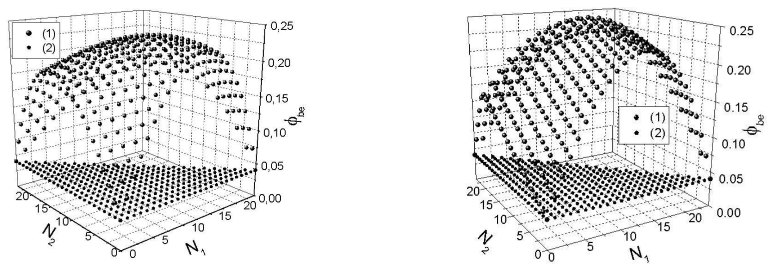
Two traffic mixes are considered which are denoted as *Case 1*and *Case 2* in Table 1 where the parameters of the sessions for each case are provided. All quantities are considered normalized with respect to the link capacity C. In order to compare the optimal

Table 1. Sessions under investigation

Case 1	s_1	s_2		s_3
Case 2	s_1		s_2	s_3
σ_i	0.04	0.16	0.04	0.64
ρ_i	0.01	0.01	0.04	0.01
D_i	1	4	4	16
w_i^{eff}	0.04	0.04	0.04	0.04

CAC algorithm with the effective bandwidth-based CAC scheme the following scenario is considered. The effective bandwidth-based CAC scheme admits the maximum number of sessions under the constraint that a nonzero weight remains to be assigned to best effort traffic. From Table 1 it can be seen that the effective bandwidth of each QoS sensitive session is 1/25 of the server's capacity (which is considered to be equal to the link capacity ($C_G = C$)), implying that for the BETA-GPS system at most 24 QoS sensitive sessions can be admitted under the effective bandwidth-based CAC scheme. This means that $N_1 + N_2 + N_3 = 24$ must hold and that the best effort traffic is assigned weight equal to 0.04 for each such triplet (N_1, N_2 and N_3 denote the number of admitted sessions of type s_1,s_2 and s_3 respectively).

For each triplet (N_1, N_2, N_3), $N_1 + N_2 + N_3 = 24$, the weight assigned to the best effort traffic by the optimal CAC scheme is computed. The results are illustrated in figure 16.



Case 1 *Case 2*

Fig. 16. Weight assigned to the best effort traffi c according to the (1) optimal CAC (2) effective bandwidth-based CAC scheme, both under the constraint $N_1 + N_2 + N_3 = 24$. The minimum guaranteed rate to the best effort traffi c is $\phi_{be} C_G$

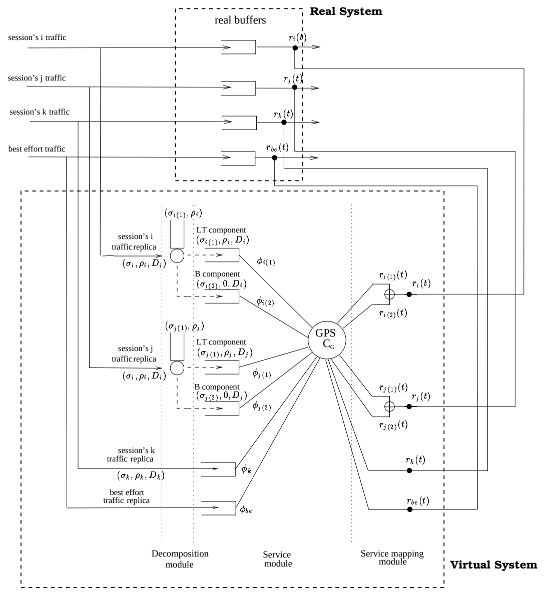

real buffers
Real System
session's i traffic
session's j traffic
session's k traffic
best effort traffic
$r_i(t)$
$r_j(t)$
$r_k(t)$
$r_{be}(t)$
$(\sigma_{i(1)}, \rho_i)$
session's i traffic replica
(σ_i, ρ_i, D_i)
LT component $(\sigma_{i(1)}, \rho_i, D_i)$
B component $(\sigma_{i(2)}, 0, D_i)$
$\phi_{i(1)}$
$\phi_{i(2)}$
$(\sigma_{j(1)}, \rho_j)$
session's j traffic replica
(σ_i, ρ_i, D_i)
LT component $(\sigma_{j(1)}, \rho_j, D_j)$
B component $(\sigma_{j(2)}, 0, D_j)$
$\phi_{j(1)}$
$\phi_{j(2)}$
session's k traffic replica
(σ_k, ρ_k, D_k)
ϕ_k
best effort traffic replica
ϕ_{be}
GPS C_G
$r_{i(1)}(t)$
$r_i(t)$
$r_{i(2)}(t)$
$r_{j(1)}(t)$
$r_j(t)$
$r_{j(2)}(t)$
$r_k(t)$
$r_{be}(t)$
Decomposition module
Service module
Service mapping module
Virtual System

Fig. 17. The DS-system

6.2 Decomposition-based Service (DS-) Scheme

According to the Decomposition-based Service (DS-) scheme, which is depicted in figure 17, sessions are not served directly by the GPS scheduler. The GPS scheduler is employed by a Virtual System which is provided with an exact replica of each session's traffic. In the Virtual System some of the sessions (replicas) are decomposed into two components, by engaging properly dimensioned leaky buckets, one for each session. These two components are buffered in separate virtual buffers and are assigned weights in proportion to which they are served by the GPS scheduler. The real traffic of a session is served at any time instant at a rate equal to the sum of the service rates of its components in the Virtual System.

The Virtual System is necessary (that is, the decomposition can not be applied directly on the original sessions) so that the real traffic of each session remains in a common buffer in order to avoid "packet reordering"; that is, to ensure that each session's traffic leaves the system in the order it arrived. The Virtual System may be considered to consist of three modules (see figure 17): (a) the decomposition module responsible for the decomposition of the replicas of the sessions (b) the service module responsible for determining the service provided to each session's components, and (c) the service mapping module which maps the service provided to the sessions components back to the original sessions.

Each session $i \sim (\sigma_i, \rho_i, D_i)$ may be considered as the superposition (aggregation) of two component sessions $i_{(1)} \sim (\sigma_{i(1)}, \rho_i, D_i)$ and $i_{(2)} \sim (\sigma_{i(2)}, 0, D_i)$, $\sigma_{i(2)} = \sigma_i -$

$\sigma_{i(1)}$, which will be referred to as the Long Term (LT) and the Bursty (B) component of session i, respectively.

For the decomposition of session i a $(\sigma_{i(1)}, \rho_i)$ leaky bucket, with $\sigma_i > \sigma_{i(1)} > 0$, is employed. Session i traffic (replica) traverses the $(\sigma_{i(1)}, \rho_i)$ leaky bucket; the part of the traffic that finds tokens is considered to belong to the LT-component of session i and the rest of session's traffic is considered to belong to the B-component of the session. Both components of session i have t he same delay constraint D_i as the original session i.

It is noted that not all the sessions are necessarily decomposed into two components; only the sessions that fulfill the criteria described in section 6.2 are decomposed (and $\sigma_{i(2)}$, $\sigma_{i(1)} = \sigma_i - \sigma_{i(2)}$, are determined). Sessions which are not decomposed are represented by only one component (session traffic replica) in the service module of the Virtual System.

CAC for the DS-system The following Claim, whose proof may be found in [73] shows that in order to develop a CAC scheme for the DS-system it suffices to develop a CAC scheme for the Virtual System, since a schedulable traffic mix for the Virtual System is also schedulable for the DS-system whose central element is the Virtual System itself.

Claim. If the components of a session are served by the service module of the Virtual System in such a way that their QoS requirements are satisfied and the real session is served at any time instant at a rate equal to the sum of the service rates of its components, then the QoS requirements of the session are also satisfied.

Let $\mathcal{T}$ denote the original traffic mix requesting service by the DS-system. Let $\mathcal{T}'$ denote the traffic mix served by the service module of the Virtual System; $\mathcal{T}'$ is obtained from $\mathcal{T}$ by replacing sessions whose replicas are decomposed in the decomposition module of the Virtual System by their components; the service module of the Virtual System serving $\mathcal{T}'$ is equivalent to (does not differ in any way from) the BETA-GPS system serving $\mathcal{T}'$ (see Figures 15 and 17). Thus, the Virtual System may be considered as a BETA-GPS system with a tunable input (under the constraints that the envelopes of the sessions components have a predefined form and the sum of the envelopes of each session components is equal to the envelope of the original session). Consequently, the CAC scheme for the Virtual System (or equivalently the DS-system) may be considered as a CAC scheme for a system similar to the BETA-GPS system with an extra degree of freedom (an extra dimension in the design space), which is the ability to determine sessions components. The CAC scheme for the DS-system that has been derived in

Table 2. Sessions parameters

session id i	1	2	3
σ_i	10.95	5.45	10.95
ρ_i	0.05-1	0.05-1	0.05-1
D_i	12	24	36

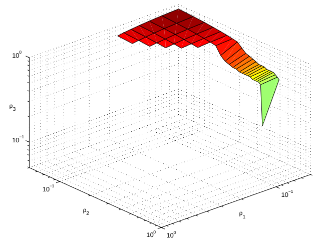


Fig. 18. BETA-GPS system employing the optimal CAC scheme for the BETA-GPS

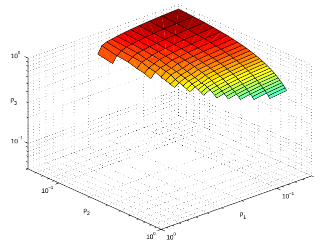


Fig. 19. DS-system employing the D_CAC scheme

[73] and is referred to as the D_CAC scheme (Decomposition-based Call Admission Control scheme) is a generalization of the optimal CAC scheme for the BETA-GPS, in the same way as the DS-system is a generalization of the BETA-GPS system. It may be considered as consisting of two distinct functions: (a) one responsible for determining which sessions of the original traffic mix will be decomposed in the Virtual System and the exact form of their components, and (b) one responsible for determining the weight assignment for the sessions components. The optimal CAC scheme for the BETA-GPS is employed for the determination of the weight assignment of the session components. The other function, which is responsible for determining the session components is such that a session replica is decomposed in the Virtual System only if (and in such a way that) this decomposition: (a) leads to better (or at least not worse) resource utilization, and (b) the session is assigned a total weight (sum of the weights of the session components) no greater than if it were not decomposed.

These conditions imply that the decomposition procedure is such that it leads to a traffic mix $\mathcal{T}'$ – starting from a traffic mix $\mathcal{T}$ – for which the optimal CAC scheme for the BETA-GPS returns total weights for the components of any session that are typically smaller and never larger than those identified by applying the optimal CAC scheme for the BETA-GPS to the original traffic mix $\mathcal{T}$. This allows a typically greater and never smaller weight to be assigned to the best effort traffic, for the traffic mixes that are schedulable in the BETA-GPS system. In addition, some traffic mixes that are not schedulable by the GPS in the BETA-GPS system, can be admitted in the DS-system (whose service module is a GPS server), and thus, the transformation of $\mathcal{T}$ to $\mathcal{T}'$ may be considered to lead to an *indirect* expansion of the schedulability region of GPS.

The details of the D_CAC scheme may be found in [73]. In the sequel a numerical example is provided that demonstrates the improvement in resource utilization that can be achieved by employing the DS-system. The considered traffic mix consists of three sessions whose parameters are shown in Table 2. σ_i and D_i are kept constant while ρ_i takes values between 0.05 and 1 (the step is equal to 0.02). The link capacity is assumed to be equal to 1. In figure 18 the maximum long term rate of session 3 (ρ_3), such that the traffic mix is schedulable, is depicted as a function of ρ_1 and ρ_2 for the BETA-GPS system employing the optimal CAC scheme for the BETA-GPS . In figure 19 the corresponding plot is given for the case of the DS-system employing the D_CAC scheme.

7 MPEG Traffic Modeling at the Frame and GoP Levels Using GBAR and D-BMAP Processes

7.1 Introduction

Modern broadband telecommunications networks are intended to deliver traffic generated by multimedia applications. Video-on-Demand (VoD) is an example of such applications. Traffic generated by VoD servers must be delivered to destinations using a certain network technology. To provide an adequate quality of service (QoS), some quality control mechanisms have to be implemented in the network.

It is well known that uncompressed video information can easily throttle the available bandwidth. In order to achieve efficient transmission of such traffic, video information must be compressed and encoded in accordance with one of the existing compression algorithms. Today, one of the most used algorithms is MPEG.

Modeling of variable bit rate (VBR) videotraffic has become an important issue since it provides the starting point for both theoretical analysis and engineering design. The source models of different types of videotraffic are needed to design and study the network performance and also to predict the QoS that a particular video application may experience at different levels of network congestion [74, 75].

A number of video source models have been proposed in literature. Recently, most attention has been paid to MPEG traffic modeling using discrete- and continuous-time Markov chains [76, 75]. Such models produce an excellent approximation of both the histogram of frame sizes and the autocorrelation function (ACF) of the empirical data but they are computationally inefficient because the construction of these Markov modulated processes from empirical data involves the so-called inverse eigenvalue problem [76, 75]. This drawback restricts their use in simulation studies where it is necessary to produce the model *”on the fly”*.

Firstly, we propose a novel frame level two-step MPEG modeling algorithm that emulates the behavior of a single MPEG-1 elementary video stream. The proposed algorithm captures well the distribution of frame sizes and the ACF at frame level.

Then, based on the special case of the D-BMAP process, we model the smoothed traffic from the single MPEG source. Based on the statistical analysis of MPEG traffic at the group of pictures (GoP) level we propose to limit the state space of the modulating Markov chain of the D-BMAP process such that it now employs only four states. The limited state space allows us to decrease the complexity of the algorithm while the necessary accuracy is retained. Our D-BMAP model is simple enough, captures both the histogram of relative frequencies and the ACF of a single smoothed MPEG traffic source, allows an analytical evaluation of the queuing systems and can be used in simulation studies.

It should be noted that in modeling MPEG traffic we focus on MPEG data, ignoring auxiliary information: the sequence header, packet headers etc.

The rest of the section is organized as follows. In the next subsection we briefly outline MPEG traffic. Then we consider our modeling methodology. In Section 7.3 we present a GBAR(1) MPEG traffic model. Section 7.4 provides an MPEG traffic methodology based on the D-BMAP process. Conclusions are drawn in the last section.

7.2 MPEG Traffic

MPEG traffic is characterized by high peak rates and drastic short-term rate changes. These features cause losses within node buffers or remarking within traffic conditioners. To deal with these problems to some extent we assume that MPEG traffic is smoothed at the group of pictures (GoP) level as follows:

$$K^m(h) = \frac{1}{m} \sum_{i=(h-1)m+1}^{hm} X(i), m = 1, l_{GoP}, ..., N, h = \frac{n}{m}, \frac{n}{m} - 1, ..., 1, \quad (4)$$

where *N* is the number of frames, l_{GoP} is the length of the GoP, *X(n)* denotes the sizes of individual frames and $K^m(h)$ denotes the sizes of the smoothed sequence. In our study GoP has (12,3,2) structure [77] and we can only use those smoothing patterns whose length is divisible by 12. We set *m*=12.

The outlined approach does not require any computational resources during the transmission of video, provides a deterministic delay and efficiently employs the frame structure of MPEG traffic. This approach can thus be used in VoD systems.

7.3 VoD Traffic Models

The GBAR(1) modeling algorithm has a two-step structure. Heyman in [78] shows that the model of MPEG traffic at frame level must incorporate three strongly auto- and cross-correlated processes. These are *I*-frame process, *P*-frame process and *B*-frame process. Therefore, we should take into account the correlation structure of empirical data. At the first step of the proposed algorithm we use the property that *I*-frames are coded autonomously without reference to preceding or following frames, and, therefore we can state that the sizes of *I*-frames are independent of the sizes of *P*- and *B*-frames. Based on this we propose to model the *I*-frame generation process as an independent stochastic process. In order to accomplish this we use the GBAR(1) process. The sequence of sizes of *I*-frames that is obtained during the first step is used together with intra-GoP correlation information as initial data for the second step of the algorithm.

Before discussing the details of the algorithm we point out the necessary prerequisites. As has been shown in [79] the distribution of intra-frame sizes at the output of an H.261 codec can be approximated by a negative-binomial distribution and its continuous equivalent - the gamma distribution. The intra-frame coding scheme utilized by H.261 codecs is almost identical to that used for *I*-frame coding in the MPEG standard. Therefore, we can expect that the gamma distribution will provide a good approximation for the *I*-frame sizes. To prove this we compared an empirical *I*-frame size distribution and the gamma distribution for the pattern *“Star Wars”* by the quantile-quantile statistical analysis. We used the following parameter values: shape parameter was set to 3.0 and scale parameter was set to 2.6*E*-5. These parameters were derived directly from the empirical distribution. We have noticed in [80] that such an approach gives a good approximation for the empirical data.

The property that allows us to proceed from the *I*-frame generation process to an MPEG aggregate frame generation process is a strong dependence between *I*-frame sizes and *B*- and *P*-frame sizes within the same GoP. This property was discovered by

Lombardo *et al.* [81, 82], clearly explained in [76] and later were called ”intra-GoP correlation”.

***I*-frame Process** In order to represent the *I*-frame generation process we propose to use the GBAR(1) process. It was originally presented by Heyman [78], where it was used as the approximation model for the frame size distribution of H.261 codec. The main distinctive feature of the process appears in the geometrical distribution of its ACF. This property allows us to model the ACF of the empirical data that exhibits some sort of short range dependence (SRD) behavior. Moreover, the marginal distribution of the frame size sequence is a gamma distribution.

Let $G(\beta, \lambda)$ be a random variable with a gamma distribution with shape parameter β and scale parameter λ, and let $B(p, q)$ be a random variable with a beta distribution with parameters p and q. The GBAR(1) process is based on two well-known results: the sum of independent random variables $G(\alpha, \lambda)$ and $G(\beta, \lambda)$ is a $G(\alpha + \beta, \lambda)$ random variable, and the product of independent random variables $B(\alpha, \beta - \alpha)$ and $G(\beta, \lambda)$ is a $G(\alpha, \lambda)$ random variable.

Thus, if we denote $G(\beta, \lambda) = X_{n-1}$, $A_n = B(\alpha, \beta - \alpha)$ and $B_n = G(\beta - \alpha)$, α and if they are mutually independent, then $X_n = A_n X_{n-1} + B_n$ is a stationary stochastic process $\{X_n, n = 0, 1, ...\}$ with marginal distribution $G(\beta, \alpha)$. The ACF is geometrical and given by $r_k = (\alpha/\beta)^k$. Parameters β and λ can be directly estimated from empirical data. Assume that the ACF is above zero for sufficiently large lag *k*: $r_k = p^k$ then the following equation holds $\alpha = p\beta$ and α is obtained.

Random variables with gamma or beta distributions generate non-integer values because they are continuous. Since the number of bits in the frame is a discrete random variable we round the values obtained from this process to the nearest integer.

Approximation of Intra-GoP Correlation In order to approximate the intra-GoP correlation structure and to obtain sizes of *P*- and *B*-frames, the algorithm proposed in [76] can be used. The algorithm is intended to clarify the dependency between the mean value and the standard deviation of *I*-frames and the sizes of appropriate *P*- or *B*-frames. These dependencies are determined as follows:

$$M[X] = f^M(K_I), \sigma[X] = f^\sigma(K_I), X \in \{P, B\}, \tag{5}$$

where $M[X]$ is the mean value of the appropriate frame (*P* or *B*), $\sigma[X]$ is the standard deviation of the *P*- or *B*-frames, and K_i is the size of the I-frame. Results are shown in [76, 80–82] where it was shown that these dependencies can be approximated by straight lines.

The mean value and the standard deviation given by (5) serve as initial parameters for certain probability distributions. These probability distributions will allow us to obtain *B*- and *P*-frame sizes holding the *I*-frame size constant (or, more precisely, an interval of *I*-frame sizes). Note that in this case, the output values will vary even for constant *I*-frame size. This property of the model emulates the behavior of the real codecs.

Approximation of *B*- and *P*- frame sizes It has been shown that histograms of the *B*- and *P*-frame sizes corresponding to a certain *I*-frame size can be approximated by a gamma distribution [81, 82]. The ACF of the *P*- and *B*-frame generation processes holding the *I*-frame size constant have the SRD property only. Since the GBAR process captures well all of the properties mentioned here we use this process as a model for *P*- and *B*-frame generation processes [80].

7.4 Traffic Modeling at the GoP Level

To model the smoothed traffic at the GoP level from the single MPEG source we propose to use the D-BMAP process. In each state of the modulating Markov chain of this process we can use an arbitrary distribution and the ACF of the process can be made to constitute a good approximation of the empirical ACF.

Consider general characteristics of the D-BMAP process. Let $\{W(b), n = 0, 1, ...\}$ be the D-BMAP arrival process. In accordance with D-BMAP, the number of arriving packets in each interval is modulated by an irreducible aperiodic discrete-time Markov chain (DTMC) with *M* states. Let *D* be the transition matrix of this process. Assume that the stationary distribution of the modulating DTMC is given by the vector $\overrightarrow{\pi}$. We define the D-BMAP process as a sequence of matrices $D(= k), k = 0, 1, ...$, each of which contains probabilities of transition from state to state with $k = 0, 1, 2, ...$ arrivals respectively [83]. Note that the number of arrivals in real models is always bounded.

Let the vector $\overrightarrow{G} = (G_1, G_2, ..., G_M)$ be the mean rate vector of the D-BMAP process. The input rate process of the D-BMAP $\{W(n), n = 0, 1, ...\}$ is defined by $\{G(n), n = 0, 1, ...\}$ with $G(n) = G_i$ while the Markov chain is in the state *i* at the time slot *n* [80]. The ACF of the rate process is given by [83]:

$$R^G(i) = \sum_{l, l \neq 1} \varphi_l \lambda_l^i,$$

$$\varphi_l = \overrightarrow{\pi} \left(\sum_{k=1}^{\infty} kD(= k) \right) \overrightarrow{g_l} \cdot \overrightarrow{h_l} \left(\sum_{k=1}^{\infty} kD(= k) \right) \overrightarrow{e}, i = 1, 2, ..., \qquad (6)$$

where λ_i is the i^{th} eigenvalue of D, $\overrightarrow{g_l}$ and $\overrightarrow{h_l}$ are the eigenvectors of *D* and $\overrightarrow{e}$ is a vector of ones.

Note that the ACF of the rate process consists of several geometric terms. Such behavior produces a good approximation of empirical ACFs that exhibit near geometrical sum decays [76, 84]. The number of geometrical terms composing the ACF depends on the number of eigenvalues which, in turn, depends on the number of states of the DTMC [85]. Thus, by varying the number of states of the modulating Markov chain we can vary the number of geometrical terms composing the ACF.

ACF Approximation In order to approximate the empirical ACF of MPEG traffic at the GoP level from the single MPEG source we use the method originally proposed in [76]. Particularly, we minimize the error of ACF approximation γ by varying the values of coefficients $(\lambda_i, \varphi_i), l = 1, 2, ..., K$ for each $K = 1, 2, ...$ in accordance with:

$$\gamma(k) = \frac{1}{i_0} \sum_{i=1}^{i_0} R^{emp}(i) - \frac{\sum_{i=1}^{K} \varphi_i \lambda_i}{R^{emp}(i)}, K = 1, 2, ..., \tag{7}$$

where *i* is the lag, i_0 is the lag when the empirical ACF reaches the confidential interval, and $R^{emp}(i)$ is the value of the ACF for lag *i*.

We note that we do not consider here those cases when $K > 3$. This is because with the increasing of the number of coefficients, which approximate the empirical ACF, the number of eigenvalues also increases and, therefore, the state space of the modulating Markov chain expands significantly. Thus, it is wise to keep the state space as small as possible and, from this point of view, $K = 2$ presents the best trade-off between the accuracy of the approximation of the empirical ACF and the simplicity of the modulating Markov chain. Note that with $K = 2$ the number of states of modulating Markov chain of the D-BMAP process may not exceed three.

At this stage the only approximation error is induced. This is the error of ACF approximation $\gamma(2)$ by two geometrically distributed terms.

Approximation by the Input Rate Process The construction of Markov modulated processes from empirical data involves a so-called inverse eigenvalue problem. It is known that the general solution of this problem does not exist. However, it is possible to solve such problems when some limitations on the form of the eigenvalues are set [76, 75].

Our limitation is that the eigenvalues should be located in the $(0, 1]$ fraction of the *X* axis. Note that one part of limitation $-1 \leq \lambda_l \leq 1, \forall l$ is already fulfilled since all eigenvalues of the one-step transition matrix of an irreducible aperiodic Markov chain are located in the $[-1, 1]$ fraction of *X* axis [85]. The second part $0 < \lambda_l \leq 1, \forall l$ should be fulfilled by the solution of the inverse eigenvalue problem.

We propose to construct the D-BMAP arrival process from two simple D-BMAP processes with a two-state modulating Markov chain. Let $\{W(n), n = 0, 1, ...\}$ be the D-BMAP arrival process, which models the smoothed traffic from an MPEG source, and $\{W_{emp}(n), n = 0, 1, ...\}$ be the empirical process of GoP sizes. $\{W^{[1]}(n), n = 0, 1, ...\}$ and $\{W^{[2]}(n), n = 0, 1, ...\}$ are two simple two-state D-BMAP processes (switched D-BMAP, SD-BMAP).

It is known that the rate process of simple an SD-BMAP can be statistically characterized by the triplet $(E[W], \varphi, \lambda)$ [86], where $E[W]$ is the mean arrival rate of the process, φ is the variance of the D-BMAP process and λ is the real eigenvalue of the modulating Markov chain which is not zero [85]. However, in order to define the rate process of the SD-BMAP, we should provide four parameters $(G_1, G_2, \alpha, \beta)$, where G_1 and G_2 are the mean arrival rates in state 1 and state 2 respectively, α is the probability of transition from state 1 to state 2 and β is the probability of transition from state 2 to state 1.

If we choose G_1 as the free variable [76] with constraint $G_1 < E[W^{[1]}]$ to satisfy $0 < \lambda \leq 1$ [86], we can obtain the other variables from the following equations [76, 86]:

$$G_2 = \frac{\varphi}{E[W] - G_1} + G_1, \alpha = \frac{(1-\lambda)(E[W] - G_1)}{G_2 - G_1}, \beta = \frac{(1-\lambda)(G_2 - E[W])}{G_2 - G_1} \quad (8)$$

Therefore, if we set $\lambda^{[1]} = \lambda_1$, $\lambda^{[2]} = \lambda_2$, $E[W] = E[W^{[1]}] + E[W^{[2]}]$, and choose $G_1^{[1]}$ and $G_1^{[2]}$ such that $G_1^{[1]} < E[W^{[1]}]$ and $G_1^{[2]} < E[W^{[2]}]$ we get both SD-BMAP arrival processes whose superposition gives us the D-BMAP process with the same ACF and mean arrival rate as the empirical data. The one-step transition matrix of the superposed process is given by the Kronecker product of composing processes $D = D^{[1]} \otimes D^{[2]}$.

It is clear from (8) that there is a degree of freedom when we choose the parameters $G_1^{[1]}$ and $G_1^{[2]}$. Moreover, the additional degree of freedom arises when we choose the values of $E[W^{[1]}]$ and $E[W^{[2]}]$. Thus, there is an infinite number of D-BMAP arrival processes with the same mean arrival rate $E[W^{emp}]$ which approximate the empirical ACF with error γ.

We also note that from the definition of the Kronecker product it follows that the eigenvalues of the matrix *D* which is the Kronecker product of matrices $D^{[1]}$ and $D^{[2]}$ with respective eigenvalues $\lambda_l^{[1]}, l = 1, 2, .., m$, and $\lambda_l^{[2]}, l = 1, 2, .., n$, has eigenvalues which are given by $\lambda_i = \lambda_l^{[1]} \lambda_l^{[2]}, i = 1, 2, .., n, .., nm$. Therefore, there is an additional error in the empirical ACF approximation, which is caused by only one eigenvalue $\lambda_l^{[1]} \lambda_l^{[2]}$ of the superposed process, since the one-step transition matrix of a two-state irreducible aperiodic Markov chain possesses two eigenvalues and one of them is always 1. Therefore, the error of the ACF approximation can be expressed precisely:

$$\gamma(2) = \frac{1}{i_0} \sum_{i=1}^{i_0} R^{emp}(i) - \frac{\sum_{l=1}^{3} \varphi_l \lambda_l^i}{R^{emp}(i)} \quad (9)$$

where $\lambda_3 = \lambda_1^{[1]} \lambda_1^{[2]}$.

Approximation of Relative Frequencies of GoP sizes Note that the above mentioned derivation of the D-BMAP process restricts us to the mean arrival rate and the ACF and does not take into account the histogram of relative frequencies of GoP sizes. To have assurance that both the histogram and ACF are matched we should assign the PDF of GoP sizes to each state of the 4-state modulating DTMC such that the whole PDF matches the histogram of GoP sizes.

Assume that the histogram of relative frequencies has *m* bins. Therefore, each PDF in each state of the modulating Markov chain should have not less than *m* bins. Since the stationary probabilities of the modulating Markov chain of the D-BMAP process are known for each PDF the following set of equations should hold:

$$\sum_{i=1}^{4} \left(f_j^i(\Delta)\pi_j^i\right) = f_j^{emp}, \sum_{j=1}^{m} \left(f_j^i(\Delta) j\Delta\right) = G^i, \sum_{j=1}^{m} f_j^i(\Delta) = 1, \quad (10)$$

where $j = 1, 2, .., m$, $i = 1, 2, 3, 4, m$ is the number of histogram bins, $f_j^{emp}(\Delta)$ and $f_j^i(\Delta)$ are the relative frequency and probability respectively corresponding to the j^{th}

bin in the i^{th} state of the modulating Markov chain, G^i is the mean arrival rate in state *i* and Δ is the length of histogram intervals.

Note that we have only $(4 \cdot 2 + m)$ equations while there are $4m$ unknowns. We also should note that in general, if the Markov chain has *M* states and there are m histogram bins the number of unknowns is M_m and we have only $2M + m$ equations. It is seen that with the increasing the number of states of the modulating Markov chain the complexity of the task increases rapidly. This is the additional reason why we should keep the state space of the modulating Markov chain as small as possible.

In order to get values of $f_j^i(\Delta), i = 1, 2, 3, 4, j = 1, 2, .., m,$ we propose to use the random search algorithm. In accordance with this algorithm we should firstly choose the necessary error of the approximation of histogram of relative frequencies η and then assign the PDF to each state of the 4-state modulating Markov chain to yield (10). Note that the time the algorithm takes to find the suitable solution depends on the error η.

7.5 Modeling Results

We have applied both algorithms [80, 84] to all traces from the MPEG trace archive [77] and found that it matches both the histogram of relative frequencies of GoP sizes and empirical ACF fairly well. In order to compare the model with empirical data we generated exactly the same amount of *I*-, *P*-, *B*- and GoP sizes as the empirical traces. Here we present a discussion of comparison studies between both models and *"Star Wars"* trace given in [80, 84].

In order to compare the distribution function of the models with corresponding histograms of relative frequencies (*I*-, *P*- and *B*-frame sizes for frame level and GoP sizes for GoP level) we have performed a chi-square statistical test. The test has shown that the statistical data of GoP sizes belong to the PDF of the D-BMAP model given a level of significance of 0,05 and statistical data of *I*-, *P*- and *B*-frame sizes belong to corresponding distribution functions of the GBAR(1) model given a level of significance of 0,1.

Considering the ACF behavior we note that up to lag 50-100 the empirical ACF and the ACF of the GBAR model are close to each other. Later, the ACF of the model quickly decays to zero. With the increasing of *p* parameter the correlation has a strong trend to rise for all lags. Thus, the model overestimates the ACF up to lag 80 and, consequently, is not able to give a good approximation for larger lags. At the frame level the D-BMAP model approximates the behavior of the empirical ACF of GoP sizes well for any lag.

One of the major shortcomings of the GBAR(1) model stems from the fact that the real data trace contains several frames with a very high number of bytes. The model can approximate this property in case of small values of *p* parameter, which will lead to underestimation of the ACF for large lags. There is a trade-off between two properties of the model: the distribution of frame sizes and the ACF. One possible solution is the careful choice of *p* values.

7.6 Conclusions

In this section we have considered the modeling of an MPEG-1 video elementary stream at the output of the codec. We propose two MPEG traffic models aimed at different MPEG logical levels: frame and group of pictures (GoP) levels.

On the frame level we proposed an extension of a modeling methodology proposed in [78] and [76]. We have used a two-step approach in order to model the video frames sequence. At the first step we approximate the I-frames generation process by a GBAR process. The second step consists of the approximation of frame sizes based on both the output of the GBAR process and intra-GoP correlations. The proposed algorithm provides a simple model of the MPEG source based on three cross-correlated processes. The algorithm captures well both the distribution of frame sizes and the ACF of empirical data. The GBAR model is fast, computationally efficient and captures well the SRD behavior of the ACF and the distribution of frame sizes. It can be used in simulation studies where there is a need to generate MPEG traffic *“on the fly“*.

The GBAR process needs only a few parameters, which can be estimated directly from the analysis of empirical data. This is a big advantage of the GBAR source model compared to other models of video traffic sources.

Our model at the GoP level is a refinement of the model originally proposed in [76]. The model of smoothed MPEG traffic at the GoP level from a single MPEG source is based on a special type of D-BMAP process. Based on the statistical analysis of MPEG traffic at the GoP level we limited the state space of the modulating Markov chain of the D-BMAP process. The limited state space allows us to decrease the complexity of the algorithm while the necessary accuracy of the approximation is retained. The D-BMAP model captures both the histogram of relative frequencies and the empirical ACF of a single smoothed MPEG traffic source. In addition, it is simple enough and can easily be used in simulation studies even when it is necessary to generate the model *“on the fly“*. The model is useful for both analytical evaluation of queuing systems and simulation studies.

8 An Open Architecture for Diffserv-enabled MPLS networks

8.1 Introduction

The Internet has quickly evolved into a very critical communications infrastructure, supporting significant economic, educational and social activities. Simultaneously, the delivery of Internet communication services has become very competitive and end-users are demanding very high quality services from their service providers. Consequently, performance optimization of large scale IP networks, especially public Internet backbones, has become an important problem. This problem is addressed by traffic engineering, which encompasses the application of technology and scientific principles to the measurement, characterization, modeling and control of Internet traffic. Enhancing the performance of an operational network, at both the traffic and resource levels, are major objectives of Internet traffic engineering; this is accomplished by addressing traffic oriented performance requirements, while utilizing network resources economically and reliably. Traffic engineering deals essentially with the selection of optimal paths that different flows should follow in order to optimize resource utilization and satisfy each flow's requirements. Historically, effective traffic engineering has been difficult to achieve in public IP networks, due to the limitations of legacy IGPs, which are adaptations of shortest path algorithms where costs are based on link metrics. This can easily lead to unfavorable scenarios in which some links become congested while others remain lightly loaded [87].

The development and introduction of Multi-Protocol Label Switching (MPLS) [88] has opened new possibilities to address some of the limitations of IP systems concerning traffic engineering. Although MPLS is a relatively simple technology (based on the classical label swapping paradigm), it enables the introduction of traffic engineering function in IP networks because of its support of explicit LSPs, which allow constraint-based routing (CBR) to be implemented efficiently. For this reason, MPLS currently appears as the best choice to implement traffic engineering in IP networks [89].

It is clear, however, that in modern IP networks, the need for supporting flows with different QoS requirements would demand additional mechanisms, such as those that perform policing on the incoming traffic, classify packets in different service classes and assure them the required quality of service. The introduction of appropriate packet scheduling disciplines and of architectures for differentiating services, such as Diffserv [90], can be used for this purpose. When MPLS is combined with Diffserv and explicit routing, we have a powerful architecture which allows both traffic engineering and quality of service provisioning. The interoperability between MPLS and Diffserv has already been the subject of studies from the Internet community and the resulting architecture is defined in RFC 3270 [91].

Believing in the benefits of a Diffserv-enabled MPLS architecture, this paper presents an experimental approach to the study of the interoperability between the Diffserv and MPLS paradigms. Few implementations currently exist of both architectures and most of them represent proprietary solutions, with device-specific contaminations. With respect to Diffserv, two different open source implementations have been considered, based respectively on the FreeBSD and Linux operating systems. The former is the ALTQ [92] package developed at the Sony Research Laboratories in Japan; the latter is the Traffic Control (TC) [93][94] module. Such modules basically make the fundamental traffic control components available, which are needed in order to realize the Diffserv paradigm: classifiers, schedulers, conditioners, markers, shapers, etc. As far as MPLS, a brand new package running under Linux has been utilized [95]. In order to gain experience with this platform and to evaluate the performance overhead due to pushing/popping the MPLS label some measurements have been performed, shown in Sect. 2.

The testbed set up at the University of Naples makes use of the Linux implementations of both Diffserv and MPLS to analyze four different scenarios (best effort, Diffserv, MPLS and Diffserv over MPLS), in order to appreciate the efficiency of Diffserv in traffic differentiation and to evaluate the interoperability between MPLS and Diffserv. Section 3 describes the experimental testbed, while Sect. 4 shows the results obtained from experimentations and draws a comparison among the four different scenarios.

We point out that the trials described in this paper are carried out by statically configuring each router, while a dynamic configuration is required to serve each incoming service request as it arrives (it is necessary to update policers and filters, create a new LSP if it is the case, etc.). Therefore, the next step is to develop an infrastructure for the dynamic configuration of routers. Some work in this direction has already been done, as described in Sect. 5. Conclusions and directions of future work are provided in Sect. 6.

8.2 MPLS: Protocol Overhead

This section presents an analysis of the overhead introduced by the MPLS encapsulation; for this purpose, two Linux PCs were put on a hub and a client-server application was used to measure the round-trip-time of packets.

We present here only the results obtained generating UDP traffic; more details can be found in [96]. Figure 20 compares the mean RTT (in microseconds) for each of the 10 trials carried out in the plain IP case and in the MPLS case, for a packet length of 1000 bytes. The first thing to note is that the mean RTT in the MPLS case is always greater than the one in the IP case; this means that the insertion and the extraction of a label introduce a certain overhead. Now we want to focus on how much greater this overhead is and how it changes with the varying of the packet size. The second graph of Figure 20 shows the percentage difference between the mean RTTs (calculated over all the 10 trials) in the two cases, for three packet lengths (10, 100 and 1000 bytes). This graph shows that the percentage difference decreases when the packets size increases. This behavior can be explained considering that, increasing the packet size, the transmission time (protocol independent) increases while the processing time (protocol dependent) remains the same. Thus the protocol dependent component is less and less important and the difference between the two RTT values tends to diminish.

These measures show that the introduction of MPLS comes with a certain cost, and thus this effect should be taken into account when we put QoS mechanisms beside MPLS. To this purpose, in the next sections we examine the performance relative to the four configurations (best-effort, Diffserv, MPLS and Diffserv-MPLS); first, we need to describe the experimental testbed.

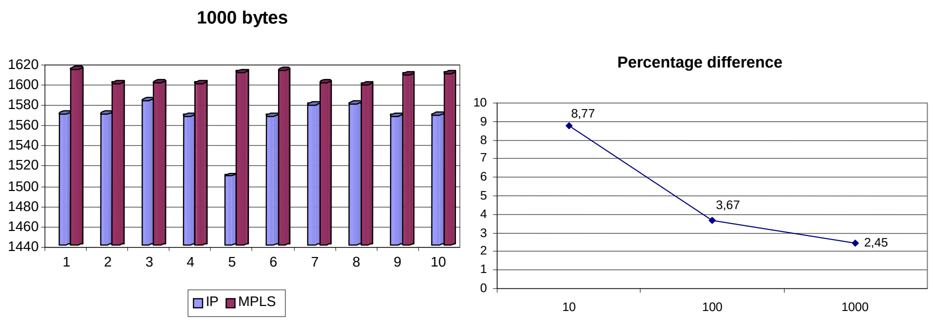


Fig. 20. Experimental results: round-trip-times.

8.3 The Experimental Testbed

In this section we present a test platform developed at the University of Naples with the intent of gaining experience from actual trials and experimentations. Such testbed, shown in Figure 21, is made of Linux routers and closely mimic (apart from the scale factor) an actual internetworking scenario. It is built of a number of interconnected LAN subnets, each realized by means of one or more cross-connections between pairs of routers. For these tests, routers A, B and C represent the QoS-enabled IP infrastructure whose performance we want to evaluate; host A acts as the traffic generator and also as a sink. The direct connection from host A to host B was added in order to allow a precise evaluation of transmission time for the packets: host A just sends data from one of its interfaces to the other and, provided that network routes have been appropriately configured, such data flows first to host B (thus crossing the QoS backbone) and then directly back to host A. Before going into the detailed performance analysis for the four different configurations (Best Effort, Diffserv, MPLS and Diffserv over MPLS) let us spend a few words about the traffic pattern we adopted for all of the experiments. We used Mtools, a traffic generator developed at University of Naples [97][98], which lets you choose packet dimensions, average transmission rates and traffic profiles (either deterministic or poisson), also giving a chance to set the Diffserv Code Point (DSCP) of the generated packets. An interesting feature of Mtools is the capability to reproduce the same realization of the packet generation random process, by setting the same generating seed for different trials. We fed the network with four poisson flows (generated by host A), each requiring a different Per Hop Behavior (EF, DE, AF11, AF12). The overall bit rate of the traffic injected into the network is about 3 Mbps. In order to appreciate the impact of scheduling (and, in the case of Diffserv, also policing) on the network, we configured all of the routers so to use a Class Based Queuing (CBQ) scheduler, with a maximum available bandwidth of 2.8 Mbps (as indicated in Figure 21). Since the injected traffic is more than the network can bear, packet losses will be experienced and we expect them to be uniformly distributed among the various classes of service only in the best effort and plain MPLS (i.e. MPLS with just one LSP carrying all the traffic) scenarios. More details on the generated traffic can be found in [99].

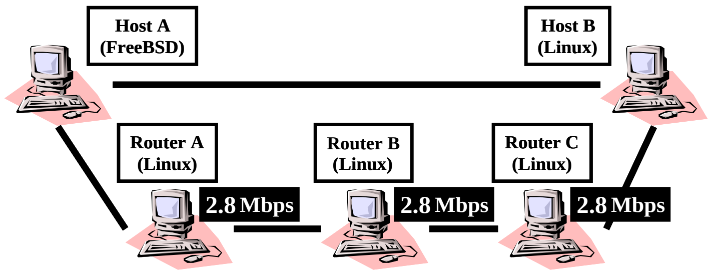


Fig. 21. Testbed used for the trials.

In the best effort case, we have just one CBQ class for all of the traffic flows (which are thus scheduled without taking into consideration the DSCP inside the IP header). All the available output bandwidth has been assigned to this class.

In the Diffserv scenario, instead, of the available 2.8 Mbps, 0.8 are assigned to the EF class, 1.0 to AF1 and 1.0 to default traffic (DE). Expedited Forwarding is served in a first in first out (FIFO) fashion, while the Assured Forwarding and default behavior queues are managed, respectively, with a Generalized Random Early Discard (GRED) and a Random Early Discard (RED) algorithm. Referring to [99] for further details, we only want to point out here that the ingress Diffserv edge router (router A), unlike the others (routers B and C), has been configured with a policer, whose function is to either drop or remark out-of-profile packets.

In a pure MPLS network, packets are assigned to the various Label Switched Paths (LSPs) based on information such as sender's and receiver's IP addresses. More specific fields, such as the DSCP (i.e the Type Of Service byte of the IP header), are completely ignored. In this scenario we will thus rely on a single LSP to carry all of the traffic flows that enter the MPLS cloud. This means that all packets will be marked with the same label and will experience the same treatment.

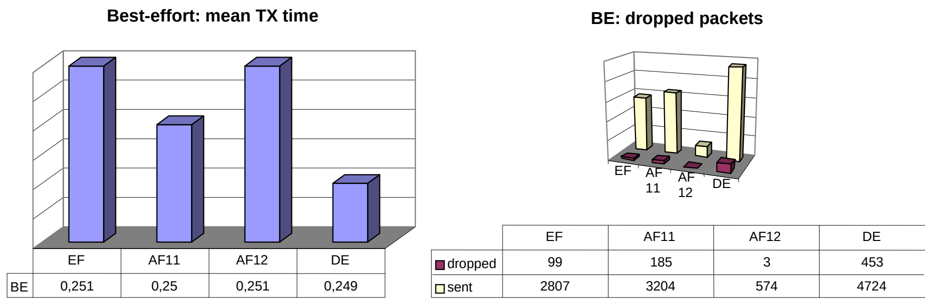
As opposed to the previous case, the Diffserv over MPLS scenario provides for MPLS support to Diffserv, that is to say packets are forwarded via MPLS label swapping, but different packet flows (as identified by the DSCP code) are treated in a different fashion. This is achieved by inserting additional information into the MPLS header. Here we will exploit one LSP for EF and DE traffic and a different one for the two AF1 flavors. In the first case, information about the Per Hop Behavior will be encoded in the EXP bits of the MPLS header, thus creating a so-called E-LSP [91]; on the other hand, for the Assured Forwarding flows (which have to be scheduled in the same fashion) the EXP bits carry information related to the drop precedence level (L-LSP). For this trial, again, router A has to perform policing at its ingress interface.

8.4 Experimental Results

In this section we will show, comment and compare the experimental results we obtained for the four aforementioned scenarios. As a preliminary consideration, please notice that transmission time is a useful indicator when evaluating performance, since it contains information related to two fundamental aspects: the time needed to process packets and the time spent waiting inside queues.

Best Effort Scenario We will start by analyzing the results obtained with the Best Effort network setup. Since no differentiation is provided in this case, we expect that all the flows receive more or less the same treatment. This is what actually happens, as witnessed by the first graph of Figure 22 and the table below, which show the mean transmission delay for every flow. From the second graph, notice that packet losses are directly proportional to the different number of sent packets for the flows.

Diffserv Scenario Let us now switch to Diffserv. What we expect now is that the requirements we specified for the single flows are actually met: EF packets should be forwarded much faster than the others, AF packets should be reliably delivered, while DE

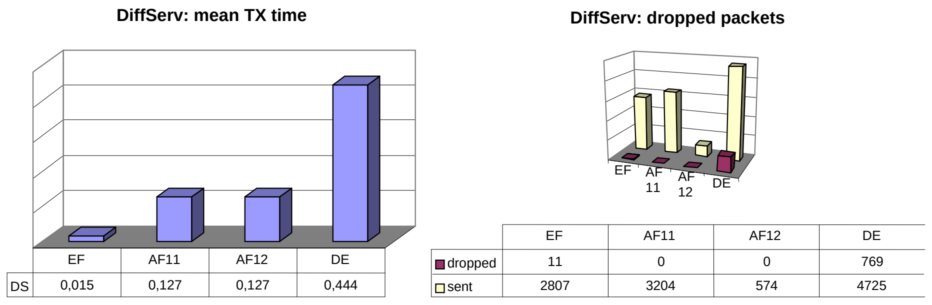


	EF	AF11	AF12	DE
BE	0,251	0,25	0,251	0,249

	EF	AF11	AF12	DE
■ dropped	99	185	3	453
□ sent	2807	3204	574	4724

Fig. 22. Best effort scenario

packets should be treated in a Best Effort fashion. The first graph of Figure 23 reports mean transmission times, while the second shows the number of dropped packets in the Diffserv case. The first comment that can be done in this case is that packets belonging to different flows are definitely treated in a different manner: transmission delays vary from one class to the other and this was exactly what we envisaged, since Diffserv is nothing but a strategy to differentiate packets based on their specific requirements. EF packets are those that endure the smallest delay; AF packets, in turn, experience a reliable delivery (as witnessed by the zero packets lost for both AF11 and AF12 classes). Finally, the DE class is the one which suffers from the highest delay and the greatest packet losses.

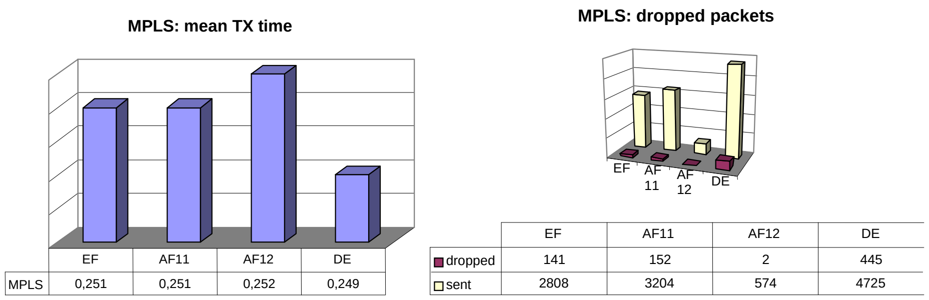


	EF	AF11	AF12	DE
DS	0,015	0,127	0,127	0,444

	EF	AF11	AF12	DE
■ dropped	11	0	0	769
□ sent	2807	3204	574	4725

Fig. 23. Diffserv scenario

MPLS Scenario In the MPLS scenario all the flows should be treated the same way, since no differentiation mechanism has been set up and just one LSP is in place. Figure 24 shows the mean transmission delay and the number of dropped packets for the MPLS configuration. As the reader will have noticed, such graphs are surprisingly similar to those we showed for the Best Effort case. With the enforced network setup the actual bottleneck is definitely represented by the output interface's bandwidth, hence the contribution due to packet processing is negligible when compared to the others: that is why we were not able to appreciate the performance difference between plain IP and MPLS we observed and measured in Sect. 2.

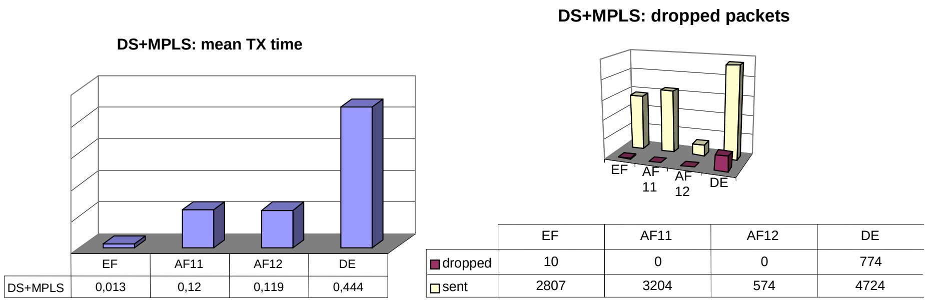


	EF	AF11	AF12	DE
MPLS	0,251	0,251	0,252	0,249

	EF	AF11	AF12	DE
■ dropped	141	152	2	445
□ sent	2808	3204	574	4725

Fig. 24. MPLS scenario

Diffserv over MPLS Scenario Let us finally take a look at the results obtained with the last configuration, where Diffserv is running over an MPLS backbone. Figure 25 shows how the presence of a Diffserv infrastructure meets the requirements for both EF and AF flows. By comparing the graphs in Figuer 23 with those in Figure 25 we notice that the traffic profiles are almost the same in the two scenarios. If you are wondering why MPLS should be used, since it adds little to network performance in the context described (i.e. one in which the routing tables dimensions are pretty small), remember that its major benefit is disclosed as soon as constraint-based routing techniques are taken into account. Stated in different terms, a comparison based solely on the delay performance figures is not fair, since a thorough analysis cannot avoid considering the fact that MPLS enables traffic engineering, by evading traditional (e.g. shortest path) forwarding, in favor of fully customized routing paradigms.



	EF	AF11	AF12	DE
DS+MPLS	0,013	0,12	0,119	0,444

	EF	AF11	AF12	DE
■ dropped	10	0	0	774
□ sent	2807	3204	574	4724

Fig. 25. DS+MPLS scenario

8.5 Dynamic Network Configuration

As already underlined, the network configurations of the four scenarios depicted in the previous section are static and manually enforced. In order to develop a general framework for the effective negotiation and delivery of services with quality assurance, it is necessary to realize an infrastructure for the dynamic and automatic configurations of network elements.

In this section, we briefly outline the building blocks of such an architecture. The "brain" of this architecture is a centralized element called *Network Controller* (NC), whose task is to configure network elements so as to satisfy the conditions included in the SLSs, which represent the contracts stipulated with the users (SLAs) in technical terms. NC can make use of two traffic engineering algorithms (one online and the other offline) in order to perform its task of admitting service requests and determining the paths their packets should follow. Once the NC has taken its decisions, it must communicate them to the network elements, so they can be enforced; these communications take place by means of the COPS (Common Open Policy Service) protocol. COPS defines the set of messages required to send the so-called *policies*, that is the actions to perform when certain conditions take place.

The set of policies that constitutes the current network configuration is stored in an LDAP directory; this enables to store information on a non-volatile support and also provides a mean for a quick recover of the configuration whenever a network element should fail and loose its configuration.

The part of the NC which is responsible of the transmission of policies is called, in the COPS jargon, PDP (Policy Decision Point). The PDP asks the LDAP server for the policies and sends them to the PEPs (Policy Enforcement Points), which are processes running on network elements. Each PEP, in its turn, passes these policies to the Device Controller (DC), which translates them in a set of command to actually configure the device. The DC is subdivided into three parts: the first one, which installs filters, markers, policers and packet scheduling disciplines; the second one, which deals with the creation of E-LSP [91], and the last one, which is responsible of mapping traffic on the appropriate LSP.

8.6 Conclusions and Future Work

In this paper we summarized the lessons we learned when applying an engineering approach to the study of state-of-the-art QoS-enabled network architectures. We presented a number of experiments aimed at investigating network performance in the presence of either Diffserv, or MPLS, or a hybrid Diffserv/MPLS infrastructure.

As to the future work, we are currently facing the issue of dynamically controlling such advanced networks, by applying to them a policy-based management paradigm. The research direction we are most interested in further exploring is related to the realization of a complex architecture for service assurance, capable to take the best out of the two technologies. A traffic-engineered Diffserv environment (TRADE) is the final target of this specific research: the major ingredients behind it are a Diffserv-capable network on top of a traffic-engineered MPLS backbone.

8.7 Acknowledgments

This work has been carried out partially under the financial support of the Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR) in the framework of the FIRB Project "Middleware for advanced services over large-scale, wired-wireless distributed systems (WEB-MINDS)"

9 Wide Area Measurements of Voice Over IP Quality

It is well known that the users of real time voice services are sensitive and susceptible to audio quality. If the quality deteriorates below an acceptable level or is too variable, users often abandon their calls and retry later. Since the Internet is increasingly being used to carry real time voice traffic, the quality provided has become, and will remain an important issue. The aim of this work is therefore to disclose the current quality of voice communication at end-points on the Internet.

It is intended that the results of this work will be useful to many different communities involved with real time voice communication. Within the next paragraph we list some potential groups to whom this work might have relevance. Firstly end users can determine which destinations are likely to yield sufficient quality. When deemed insufficient they can take preventative measures such as adding robustness, for example in the form of forward error correction to their conversations. Operators can use findings such as these to motivate upgrading links or adding QoS mechanisms where poor quality is being reported. Network regulators can use this kind of work to verify the quality level that was agreed upon, has indeed been deployed. Speech coder designers can utilise the data as input for a new class of codecs, of particular interest are designs which yield good quality in the case of bursty packet loss. Finally, researchers could use the raw data we gathered to investigate questions such as, "is the quality of real time audio communication on the Internet improving or deteriorating?".

The structure of the following sections are as follows: Section 9.1 begins with some background on the quality measures we have used in this work namely, loss, delay and jitter. Following on from the quality measures section 9.2 gives a description of the methodology used to ascertain the quality. In section 9.3 the results are presented, and due to space considerations we condense the results into one table showing the delay, loss and jitter values for the paths we measured. In section 9.4 the related work is given, comparing results obtained in this attempt with other researchers' work. This is considered important as it indicates whether quality has improved or deteriorated since those studies. Section 9.5 rounds off with some conclusions and a pointer to the data gathered.

9.1 What Do We Mean by Voice over IP Quality?

Ultimately, users judge the quality of voice transmissions. Organisations such as ETSI, ITU, TIA, RCR plus many others have detailed mechanisms to assess voice quality. For these organisations the particular focus is speech coders. Assigning quality "scores" involves replaying coded voice to both experienced and novice listeners and asking them to state the perceived quality. Judging the quality of voice data that has been transmitted across a wide area network is more difficult. The network inflicts its own impairment on the quality of the voice stream. By measuring the delay, jitter and loss of the incoming data stream at the receiver, we can provide some indication on how suitable the network is for real time voice communication.

Therefore the quality of VoIP sessions can often be quantified by network delay, packet loss and packet jitter. We emphasise that these three quantities are the major contributors to the perceived quality as far as the network is concerned. The G.114 ITU standard states the end-to-end one way delay should not exceed 150ms [100]. Delays over this value adversely effect the quality of the conversation. In an alternative study Cole and Rosenbluth state that users perceive a linear degradation in the quality up to 177ms [101]. Above this figure the degradation is also linear although markedly worse. As far as the packet loss is concerned using simple speech coding, e.g. A-law or μ-law, tests have shown that the mean packet loss should not exceed 10% before glitches due to lost packets seriously affect the perceived quality. Note that a loss rate such as this does not say anything about the distribution of the losses. As far as the authors are aware of, no results exist that state how jitter solely can affect the quality of voice communication. When de-jittering mechanisms are employed, the network *jitter* is typically transferred into application *delay*. The application must hold back a sufficient number of packets in order to ensure smooth, uninterrupted playback of speech. To summarise, we refer to the quality as a combination of delay, jitter and loss. It is important to mention we explicitly do not state how these values should be combined. The ITU E-model [102] is one approach but others exist, therefore we refer the interested reader to the references in this section as well as [103] and [104].

9.2 Simulating and Measuring Voice over IP Sessions

Our method to measure VoIP quality is to send pre-recorded calls between globally distributed sites. Through the modification of our own VoIP tool, Sics*o*phone, the intervening network paths are probed by a 70 second pre-recorded “test signal”. The goal of this work is therefore to report in what state the signal emerges after traversing the network paths available to us. Incidentally, we do not include the signalling phase (i.e. establishing communication with the remote device) in these measurements, rather we concentrate solely on the quality of the data transfer.

Nine sites have been carefully chosen with large variations in hops, geographic distances, time zones and connectivity to obtain a diverse selection of distributed sites. One limitation of the available sites was they were all located at academic institutions, which are typically associated with well provisioned networks. Their locations are shown in the map of Figure 26. The sites were connected as a full mesh allowing us, in theory, to measure the quality of 72 different Internet paths. In practice, some of the combinations were not usable due to certain ports being blocked, thus preventing the audio to be sent to some sites. There were four such cases. Bi-directional sessions were scheduled on an hourly basis between any two given end systems. Calls were only transferred once per hour due to load considerations on remote machines.

In Table 3 below we list the characteristics of the call we used to probe the Internet paths between those indicated on the map. Their locations, separation in hops and time

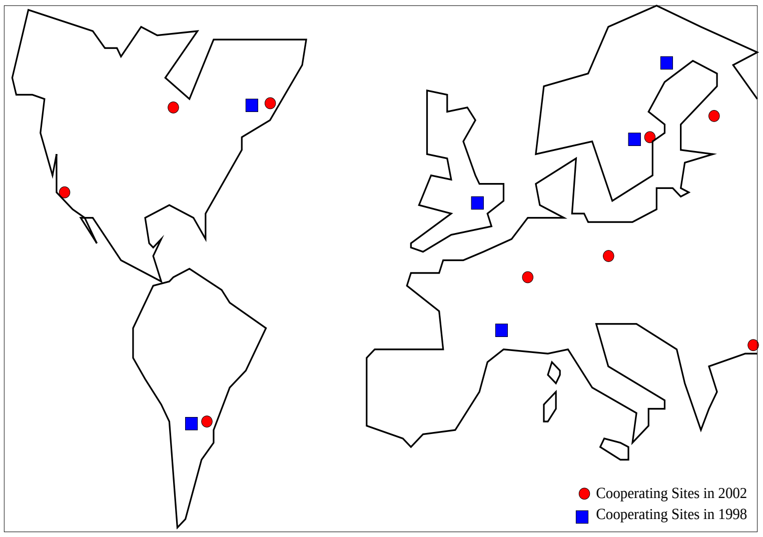


Fig. 26. The nine sites used in 2002 are shown with circles. The six depicted with squares show those that were available to us in 1998, three remained unchanged during the past four years.

zones are given in the results section. As stated the call is essentially a fixed length PCM coded file which can be sent between the sites, the length of the call and the payload size were arbitrarily chosen. Over a 15 week period we gathered just over 18,000 recorded sessions. The number of sessions between the nine sites is not evenly distributed due to outages at some sites, however we attempted to ensure an even number of measurements per site, in total nearly 33 million individual packets were received during this work.

A Networking Definition of Delay We refer to the delay as the *one way network* delay. One way delay is important in voice communication, particularly if it is not equal in each direction. Measuring the one way delay of network connections without the use of synchronised clocks is a non-trivial task. Hence many methods rely on round-trip measurements and halve the result, hence estimating the one way delay. We measured the network delay using the RTCP protocol which is part of the RTP standard [105]. A brief description follows. At given intervals the sender transmits a so called “report” containing the time the report was sent. On reception of this report the receiver records the current time. Therefore two times are recorded within the report. When returning the report to the sender, the receiver subtracts the time it put in the report, therefore accounting for the time it held the report. Using this information the sender can calculate the round-trip delay and importantly, discount the time spent processing the reports at the receiver. This can be done in both directions to see if any significant anomalies exist. We quote the network delay in the results section as they explicitly do not include any contribution from the end hosts. Therefore it is important to state the delay is *not* the end-to-end delay but the network delay. We chose not to include the delay contributed by the end system as it varies widely from operating system to operating system and

Test "signal"	
Call duration	70 seconds
Payload size	160 bytes
Packetisation time (ms)	20ms
Data rate	64kbits/sec
With silence suppression	2043 packets
Without silence suppression	3653 packets
Coding	8 bit PCM
Recorded call size	584480 bytes
Obtained data	
Number of hosts used (2003)	9
Number of traces obtained	18054
Number of data packets	32,771,021
Total data size (compressed)	411 Megabytes
Measurement duration	15 weeks

Table 3. The top half of the table gives details of the call used to measure the quality of links between the sites. The lower half provides information about the data which we gathered.

how the VoIP application itself is implemented. The delay incurred by an end system can vary from 20ms up to 1000ms, irrespective of the stream characteristics.

Jitter - An IETF Definition Jitter is the statistical variance of the packet interarrival time. The IETF in RFC 1889 define the jitter to be the mean deviation (the smoothed absolute value) of the packet spacing change between the sender and the receiver [105]. Sics$o$phone sends packets of identical size at constant intervals which implies that $S_j - S_i$ (the sending times of two consecutive packets) is constant. The difference of the packet spacing, denoted D, is used to calculate the interarrival jitter. According to the RFC, the interarrival jitter should be calculated continuously as each packet i is received. For one particular packet the interarrival jitter J_{i-1} for the previous packet $i-1$ is calculated thus:

$$J_i = J_{i-1} + (|D(i-1, i)| - J_{i-1})/16.$$

According to the RFC "the gain parameter 1/16 gives a good noise reduction ratio while maintaining a reasonable rate of convergence". As stated earlier buffering due to jitter adds to the delay of the application. This is therefore not visible in the results we present. The "real" time needed for de-jittering depends on how the original time spacing of the packets should be restored. For example if a single packet buffer is employed it would result in an extra 20ms (the packetisation time) being added to the total delay. Note that packets arriving with a spacing greater than 20ms should be discarded by the application as being too late for replay. Multiples of 20ms can thus be allocated for every packet held before playout in this simple example.

Counting Ones Losses in the Network We calculate the lost packets as is exactly defined in RFC 1889. It defines the number of lost packets as the expected number of packets subtracted by the number actually received. The loss is calculated using expected values so as to allow more significance for the number of packets received, for example 20 lost packets from 100 packets has a higher significance than 1 from 5. For simple measures the percentage of lost packets from the total number of packets expected is stated. From above we know that the losses in this work *do not* include those incurred by late arrivals, as knowledge of the buffer playout algorithm is needed, therefore our values are only the network loss. Detailed analysis of the loss patterns is not given in the results section, we simply state the percentages of single, double and triplicate losses.

9.3 Results

The results of 15 weeks of measurements are condensed into Figure 27. The table should be interpreted as an 11x11 matrix. The locations listed horizontally across the top of the table are the locations configured as receivers, and when listed vertically they are configured as senders. The values in the rightmost column and bottom row are the statistical means for all the connections *from* the host in the same row and *to* the host in the same column respectively. For example the last column of the first row (directly under Mean) the average delay to all destinations from Massachusetts is 112.8ms.

Each cell includes the delay, jitter, loss, number of hops and the time difference listed vertically in the cell and prefixed by the letters D, J, L, H and T for each of the connections. The units for each quantity is the delay in milliseconds, the jitter in milliseconds, the loss in percentage, the hops as reported by traceroute and time differences in hours. A ‘+’ indicates that the local time from a site is ahead of the one in the corresponding cell and behind for a ’-’. The values in parenthesis are the standard deviations. A NA signifies “Not Available” for this particular combination of hosts. The bottom rightmost cell contains the mean for all 18054 calls made, both to and from all the nine hosts involved. The most general observation is the quality of the paths is generally good. The average delay is just below the ITU’s G.114 recommendation for the end-to-end delay. Nevertheless at 136ms it does not leave much time for the end systems encode/decode and replay the voice stream. A small buffer would absorb the 4.1ms jitter and a loss rate of 1.8% is more than acceptable with PCM coding [103].

There are two clear groupings from these results, those within the EU and the US and those outside. The connections in Europe and the United States and between them are very good. The average delay between the US/EU hosts is 105ms, the jitter is 3.76ms and the loss 1.16%. Those outside fair less well. The Turkish site suffers from large delays, which is not surprising as the Turkish research network is connected via a satellite link to Belgium (using the Geant network). The jitter and loss figures however are low, 5.7ms and 4% respectively. The Argentinian site suffers from asymmetry problems. The quality when sending data to it is significantly worse than when receiving data from it. The delay is 1/3 higher, the jitter is more than twice as in the opposite direction and the loss is nearly four times higher than when sending to it. Unfortunately we could not perform a traceroute from the host in Buenos Aires due to not having root access with which to run a traceroute like command, so we cannot say how the route contributed to these values.

receiver sender	Massachusetts	Michigan	California	Belgium	Finland	Sweden	Germany	Turkey	Argentina	Mean
Massachusetts	*	D:38.0 (17.1) J:2.4 (1.7) L:0.1 (0.6) H:14 (+1) T:0	D:54.2 (15.8) J:2.4 (1.8) L:0.1 (0.9) H:19 T:-3	D:67.1 (15.5) J:3.6 (1.5) L:0.1 (0.8) H:11 T:+6	D:97.1 (2.6) J:2.5 (1.5) L:0.1 (0.8) H:15 T:+7	D:99.5 (8.5) J:3.2 (1.7) L:0.04 (0.2) H:21 T:+6	D:58.4 (5.0) J:4.5 (1.4) L:0.0 (0.0) H:17 (+3) T:+6	D:388.2 (43.2) J:10.4 (4.9) L:4.9 (4.7) H:20 T:+7	D:99.7 (4.9) J:19.9 (8.4) L:8.9 (7.2) H:25 T:+1	D:112.8 J:6.1 L:1.2 H:17
Michigan	D:36.4 (15.4) J:4.7 (0.8) L:0.0(0.2) H:14 (+1) T:0	*	D:40.4 (4.5) J:4.4 (0.8) L:0.2 (1.1) H:20 (+1) T:-3	D:63.5 (4.2) J:4.3 (0.7) L:0.0 (0.1) H:11 T:+6	D:88.2 (8.0) J:4.1 (0.7) L:0.1 (1.1) H:17 T:+7	D:86.7 (4.7) J:5.2 (0.6) L:0.1 (2.2) H:23 T:+6	D:63.6 (8.2) J:7.3 (1.9) L:0.2 (0.9) H:16 (+1) T:6	D:358.9 (44.9) J:5.6 (1.7) L:3.0 (1.9) H:20 T:7	D:112.1 (10.6) J:18.7 (7.9) L:6.5 (7.0) H:25 T:+1	D:106.2 J:6.8 L:1.3 H:18
California	D:54.5 (16.7) J:2.0 (1.0) L:0.1 (0.36) H:18 (+1) T:+3	D:40.6 (5.1) J:1.2 (0.6) L:0.1 (1.9) H:21 T:+3	*	D:81.0 (2.2) J:1.6 (0.8) L:0.2 (0.8) H:20 T:+9	D:106.0 (3.0) J:1.4 (0.8) L:0.6 (1.4) H:25 (+1) T:+10	D:108.0 (2.4) J:2.1 (0.9) L:0.2 (0.3) H:30 (+2) T:+9	D:81.5 (1.8) J:4.9 (1.5) L:2.8 (3.0) H:23 T:+9	D:386.9 (60.5) J:5.3 (1.7) L:4.4 (2.4) H:23 T:+10	D:123.9 (12.4) J:18.1 (9.9) L:8.9 (8.2) H:25 T:+4	D:122.2 J:4.6 L:2.2 H:23
Belgium	D:65.2 (10.1) J:1.6 (0.6) L:0.0 (0.0) H:16 T:-6	D:63.4 (3.3) J:0.6 (0.1) L:0.0 (0.0) H:17 T:-6	D:84.0 (1.3) J:0.9 (0.8) L:1.2 (1.0) H:23 T:-9	*	D:31.3 (0.6) J:0.9 (0.5) L:0.0 (0.0) H:17 T:+1	D:33.4 (0.2) J:1.6 (0.9) L:0.0(0.0) H:22 T:0	D:16.6 (10.4) J:3.4 (1.5) L:0.21 (0.7) H:13 T:0	D:341.1 (24.7) J:6.9 (2.0) L:3.8 (2.7) H:16 (+2) T:+1	D:136.5 (7.1) J:NA L:NA H:19 T:-5	D:96.4 J:2.0 L:0.6 H:17
Finland	D:97.8 (4.2) J:1.7 (0.8) L:0.0 (0.1) H:15 (+1) T:-7	D:86.8 (1.9) J:1.1 (0.6) L:0.0 (0.3) H:17 (+1) T:-7	D:109.9 (4.7) J:1.4 (0.8) L:0.7 (1.4) H:24 (+2) T:-10	D:30.7 (0.3) J:1.4 (0.6) L:0.1 (0.3) H:16 T:-1	*	D:13.6 (1.0) J:1.9 (0.9) L:0.0 (0.0) H:20 T:-1	D:26.8 (7.3) J:3.9 (1.1) L:0.0(0.0) H:20 (+1) T:-1	D:321.2 (39.3) J:3.4 (1.7) L:3.2 (1.7) H:17 (+2) T:0	D:161.5 (12.2) J:17.4 (8.2) L:7.5 (6.5) H:19 T:-6	D:106.3 J:4.1 L:1.4 H:18
Sweden	D:99.3 (8.8) J:3.0 (1.9) L:0.0 (0.0) H:22 (+1) T:-6	D:84.9(1.9) J:2.5 (2.0) L:0.03 (0.4) H:25 T:-6	D:105.6 (2.1) J:3.2 (1.96) L:0.1 (0.1) H:30 T:-9	D:33.3 (0.4) J:2.8 (1.6) L:0.1 (0.3) H:24 T:0	D:13.5 (0.5) J:2.4 (1.8) L:0.0 (0.01) H:21 T:+1	*	D:29.8 (12.8) J:4.8 (2.5) L:0.0 (0.0) H:25 T:0	D:322.2 (30.3) J:3.2 (1.49) L:2.9 (1.0) H:26 T:+1	D:165.6 (17.9) J:NA L:NA H:41 T:-5	D:107.8 J:2.8 L:0.4 H:26
Germany	D:63.5 (9.6) J:1.72 (0.7) L:0.0(0.0) H:15 T:-6	D:60.4 (0.5) J:0.7 (0.3) L:0.0 (0.0) H:16 T:-6	D:84.4 (1.0) J:1.8 (0.7) L:2.5 (1.9) H:22 T:-9	D:11.1 (0.2) J:0.8 (0.3) L:0.0 (0.0) H:12 T:0	D:27.8 (7.3) J:1.0 (0.5) L:0.0 (0.0) H:17 T:+1	D:29.2 (7.6) J:1.5 (0.6) L:0.0 (0.0) H:22 T:0	*	D:300.7 (39.7) J:4.8 (2.1) L:3.7 (2.5) H:16 T:+1	D:149.8 (15.6) J:NA L:NA H:18 T:-5	D:90.9 J:1.6 L:0.8 H:17
Turkey	D:379.1 (47.1) J:8.6 (0.7) L:8.1 (2.8) H:18 (+1) T:-7	D:387.9 (35.5) J:8.9 (1.2) L:8.0 (2.9) H:20 T:-7	D:410.9 (43.9) J:8.8 (2.5) L:7.6 (6.8) H:19 T:-10	D:330.2 (28.6) J:9.2 (2.0) L:7.10 (4.0) H:17 T:-1	D:318.9 (42.4) J:8.8 (0.6) L:7.8 (2.7) H:19 T:0	D:311.1 (8.3) J:9.1 (0.7) L:8.4 (3.1) H:25 T:-1	D:378.2 (49.3) J:10.7 (1.2) L:8.0 (3.1) H:16 T:-1	*	D:490.8 (26.0) J:NA L:NA H:18 T:-6	D:375.9 J:8.0 L:6.9 H:19
Argentina	D:117.0 (30.8) J:4.2 (2.0) L:0.5 (1.4) H:NA T:-1	D:146.7 (44.2) J:4.3 (2.3) L:0.5 (1.5) H:NA T:-1	D:152.0 (47.8) J:3.1 (2.4) L:0.6 (1.8) H:NA T:-4	D:NA J:4.2 (2.0) L:0.5 (1.4) H:NA T:+5	D:164.1 (27.2) J:3.9(2.2) L:0.5 (1.4) H:NA T:+6	D:160.9 (47.7) J:2.9 (0.8) L:0.0 (0.1) H:NA T:+5	D:180.5 (50.5) J:4.7 (1.5) L:0.1 (0.1) H:NA T:+5	D:NA J:6.0(1.2) L:5.8 (3.0) H:NA T:+6	*	D:115.2 J:4.2 L:1.1 H:NA
Mean	D:114.1 J:3.4 L:1.1 H:14	D:113.6 J:3.4 L:1.1 H:16	D:115.7 J:3.2 L:1.6 H:19	D:77.1 J:3.5 L:1.0 H:13	D:105.8 J:3.1 L:1.1 H:16	D:105.2 J:3.4 L:1.1 H:20	D:104.4 J:5.5 L:1.4 H:16	D:345.6 J:5.7 L:4.0 H:17	D:180.0 J:9.3 L:4.00 H:23	D:136.2 J:4.1 L:1.8 H:18

Fig. 27. A summary of 18000 VoIP sessions. The delay, jitter and loss for the nine sites. The delay and jitter are in milliseconds, the losses are in percentages. The number of hops and time zones (in hours) are also given. The means for each site and all sites are stated and standard deviations are in parenthesis.

We now turn our attention to results which are not related to any particular site. As far as loss is concerned the majority of losses are single losses. 78% of all the losses counted in all trace files were single losses whereas 13% were duplicate losses and only 4% triplicate losses. Generally the jitter is low relative to the delay of the link, approximately 3-4%. This is not totally unexpected as the loss rates are also low. With the exception of the Argentinian site, the sites did not exhibit large differences in asymmetry and were normally within 5% of each other in each direction. It is interesting to note that the number of hops could vary under the 15 week measurement period denoted by () in the hops field. Only very few ($< 0.001\%$) out of sequence packets were observed. Within [106] there are details of other tests, such as the effect of using silence suppression, differing payload sizes and daytime effects. In summary no significant differences were observed as these quantities were varied.

9.4 Related Work

Similar but less extensive measurements were performed in 1998 [107]. Only three of the hosts remain from four years ago so comparisons can only be made for these routes. An improvement, in the order of 5-10% has been observed for these routes. We should point out though, the number of sessions recorded four years ago numbered only tens per host, whereas on this occasion we performed hundreds of calls from each host. Bolot et. al. looked at consecutive loss for a FEC scheme [108]. They concluded that the number of consecutive losses is quite low and stated that most losses are one to five losses at 8am and between one to ten at 4pm. This is in broad agreement with the findings in this work, however we did not investigate the times during the day of the losses. Maxemchuk and Lo measured both loss and delay variation for intra-state connections within the USA and international links [109]. Their conclusion was the quality depends on the length of the connection and the time of day. We did not try different length of connections but saw much smaller variations (almost negligible) during a 24 hour cycle (see [106]). We attribute this to the small 64kbits per second VoIP session on well dimensioned academic networks. It is worthy to point out our loss rates were considerably less than Maxemchuks (3-4%). Dong Lin had similar conclusions [110], stating that in fact even calls within the USA could suffer from large jitter delays. Her results on packet loss also agree with those in [108], which is interesting, as the measurements were taken some four years later.

9.5 Conclusions

We have presented the results of 15 weeks of voice over IP measurements consisting of over 18000 recorded VoIP sessions. We conclude that the quality of VoIP is very good and in most cases is over the requirements as stated in many speech quality recommendations. Recall that all of the sites were at academic institutions which is an important factor when interpreting these results as most universities have well provisioned links, especially to other academic sites. Nevertheless, the loss, delay and jitter values are very low and from previous measurements the quality trend is improving. We can only attribute this to more capacity and better managed networks than those four years ago. However some caution should be expressed as the sample period was only 15 weeks, the bandwidth of the flows very small and only used once per hour. We do have however quite a large number of sample sessions. VoIP is dependent on the IP network infrastructure and not only on the geographic distance. This can be clearly seen in the differences between the Argentinian and Turkish hosts. We have found performing measurements on this scale is not an easy task. Different access mechanisms, firewalls, NATs and not having super-user permission complicates the work in obtaining measurements. Since it is not possible to envisage all the possible uses for this data we have made it available for further investigation at http://www.sics.se/˜ianm/COST263/cost263.html.

References

1. Willinger, W., Paxson, V.: Where mathematics meets the internet. Notices of the American Mathematical Society **45** (1998) 961–970
2. Park, K., Willinger, W.: Self-similar network traffi c and performance evaluation. Wiley (2000)
3. Padhye, J., Firoiu, V., Towsley, D., Kurose, J.: Modeling tcp throughput: a simple model and its empirical validation. In: Proceedings of SIGCOMM 98, ACM. (1998)
4. Kleinrock, L.: Queuing Theory - Vol 2. Wiley (1976)
5. Ben Fredj, S., Bonald, T., Proutire, A., Rgni, G., Roberts, J.: Statistical bandwidth sharing: a study of congestion at fbw level. Proceedings of Sigcomm 2001, Computer Communication Review **31** (2001) 111–122
6. Bonald, T., Roberts, J.: Congestion at fbw level and the impact of user behaviour. Computer Networks, to appear (2003)
7. Le Boudec, J.Y., Thiran, P.: A theory of deterministic queuing systems for the Internet. Volume 2050 of Lecture notes in computer science. Springer-Verlag (2001)
8. Bonald, T., Proutire, A., Roberts, J.: Statistical performance guarantees for streaming fbws using expedited forwarding. In: Proceedings of Infocom 2001. (2001) 1104–1112
9. Gibbens, R., Kelly, F., Key, P.: A decision theoretic approach to call admission control. IEEE JSAC **13** (1995) 1104–1114
10. Bonald, T., Oueslati, S., Roberts, J.: Ip traffi c and qos control: towards a fbw-aware architecture. In: Proceedings of World Telecom Conference, Paris (2002)
11. Benameur, N., Ben Fredj, S., Delcoign, F., Oueslati-Boulahia, S., Roberts, J.: Integrated admission control for streaming and elastic fbws. In: Proceedings of QofIS 2001, LNCS 2156, Springer (2001)
12. Quadros, G., Monteiro, E., Boavida, F.: A qos metric for packet networks. In: Proceedings of SPIE International Symposium on Voice, Video, and Data Communications, Hynes Convention Center, Boston, Massachusetts, USA (1998)
13. for Standardization, I.O.: Information technology – quality of service: Framework. International Standard 13236 (1998)
14. Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: Rfc 2475 – an architecture for differentiated service (1998)
15. Quadros, G., Alves, A., Monteiro, E., Boavida, F.: An approach to support traffi c classes in ip networks. In Crowcroft, J., Roberts, J., Smirnov, M.I., eds.: Lecture Notes in Computer Science. Volume 1922., Berlin, Germany, Springer-Verlag, Heidelberg (2000) 285–299
16. Quadros, G., Alves, A., Monteiro, E., Boavida, F.: An effective scheduler for ip routers. In: Proceedings of ISCC'2000, Antibes, France, Fifth IEEE Symposium on Computers and Communications (2000)

17. Quadros, G., Alves, A., Silva, J., Matos, H., Monteiro, E., Boavida, F.: A queue management system for differentiated-services ip routers. In Crowcroft, J., Roberts, J., Smirnov, M.I., eds.: Lecture Notes in Computer Science. Volume 1922., Berlin, Germany, Springer-Verlag, Heidelberg (2000) 14–27
18. Alves, A., Quadros, G., Monteiro, E., Boavida, F.: Qostat – a tool for the evaluation of qos-capable routers. In: Proceedings of SPIES's International Symposium on Voice, Video, and Data Communications, Boston (2000)
19. netIQ: Chariot – User Guide. (2001)
20. Oliveira, M., Brito, J., Melo, B., Quadros, G., Monteiro, E.: Encaminhamento com qualidade de serviço: Desafi os da implementação da estratégia qosr-lct (portuguese). In: Proceedings of CRC'2000, Third National Conference on Computer Networks – Technologies and Applications, Viseu, Portugal, FCCN (2000)
21. Oliveira, M., Melo, B., Quadros, G., Monteiro, E.: Quality of service routing in the differentiated services framework. In: Proceedings of SPIES's International Symposium on Voice, Video, and Data Communications, Boston (2000)
22. Lourenco, D., Oliveira, M., Quadros, G., Monteiro, E.: Defi nição do mecanismo de controlo de admissão para o modelo de serviços de lct-uc. In: Proceedings of CRC'2000, Third National Conference on Computer Networks – Technologies and Applications, Viseu, Portugal, FCCN (2000)
23. Breslau, L., Shenker, S.: Best–effort versus reservations: A simple comparative analysis. ACM Computer Communication Review **28** (1998) 3–16
24. Braden, R., Clark, S., Shenker, S.: Integrated services in the internet architecture. RFC 1633, IETF (1994)
25. Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: An architecture for differentiated services. RFC 2475, IETF (1998)
26. Shenker, S., Partridge, C., Guerin, R.: Specifi cation of guaranteed quality of service. RFC 2212, IETF (1997)
27. Wroclawski, J.: Specifi cation of the controlled-load network element service. RFC 2211, IETF (1997)
28. Braden, R., Zhang, L., Berson, S., Herzog, S., Jamin, S.: Resource ReSerVation protocol (RSVP). RFC 2205, IETF (1997)
29. Jacobson, V., Nichols, K., Poduri, K.: An expedited forwarding PHB. RFC 2598, IETF (1999)
30. Heinanen, J., Baker, F., Weiss, W., Wroclawski, J.: Assured forwarding PHB group. RFC 2597, IETF (1999)
31. Baker, F., Iturralde, C., Le Faucheur, F., Davie, B.: RSVP reservations aggregation. Internet Draft, IETF (2001) (Work in progress).
32. Bernet, Y.: Format of the RSVP DCLASS object. RFC 2996, IETF (2000)
33. 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, IETF (2000)
34. Cetinkaya, C., Knightly, E.W.: Egress admission control. In: Proc. of IEEE INFOCOM 2000, Tel Aviv, Israel (2000) 1471–1480
35. Breslau, L., Jamin, S., Shenker, S.: Comments on the performance of measurement–based admission control algorithms. In: Proc. of IEEE INFOCOM 2000, Tel Aviv, Israel (2000) 1233–1242
36. Karlsson, G.: Providing quality for internet video services. In: Proc. of CNIT/IEEE ITWoDC 98, Ischia, Italy (1998) 133–146
37. Fodor (née Elek), V., Karlsson, G., Rönngren, R.: Admission control based on end-to-end measurements. In: Proc. of IEEE INFOCOM 2000, Tel Aviv, Israel (2000) 623–630

38. Ramakrishnan, K., Floyd, S.: A proposal to add explicit congestion notifi cation (ECN) to IP. RFC 2481, IETF (1999)
39. Kelly, F.P., Key, P.B., Zachary, S.: Distributed admission control. IEEE Journal on Selected Areas in Communications **18** (2000) 2617–2628
40. Kelly, T.: An ECN probe–based conection acceptance control. ACM Computer Communication Review **31** (2001) 14–25
41. Bianchi, G., Capone, A., Petrioli, C.: Throughput analysis of end–to–end measurement-based admission control in IP. In: Proc. of IEEE INFOCOM 2000, Tel Aviv, Israel (2000) 1461–1470
42. Bianchi, G., Capone, A., Petrioli, C.: Packet management techniques for measurement based end-to-end admission control in IP networks. Journal of Communications and Networks **2** (2000) 147–156
43. Bianchi, G., Borgonovo, F., Capone, A., Petrioli, C.: Endpoint admission control with delay variation measurements for QoS in IP networks. ACM Computer Communication Review **32** (2002) 61–69
44. Breslau, L., Knightly, E.W., Shenker, S., Stoica, I., Zhang, H.: Endpoint admission control: Architectural issues and performance. In: Proc. of ACM SIGCOMM 2000, Stockholm, Sweden (2000) 57–69
45. Más Ivars, I., Karlsson, G.: PBAC: Probe–based admission control. In: Proc. of QoFIS 2001, Coimbra, Portugal (2001) 97–109
46. Más, I., Fodor, V., Karlsson, G.: Probe–based admission control for multicast. In: Proc. of IWQoS 02, Miami Beach, Florida (2002) 97–109
47. Gibbens, R.J., Kelly, F.P.: Distributed connection acceptance control for a connectionless network. In: Proc. of the 16th International Teletraffi c Congress, Edinburgh, Scotland (1999) 941–952
48. Bianchi, G., Borgonovo, F., Capone, A., Fratta, L., Petrioli, C.: PCP: an end-to-end measurement-based call admission control for real-time services over IP networks. In: Proc. of QoS–IP 2001, Rome, Italy (2001) 391–406
49. Roberts, J.W., Mocci, U., Virtamo, J., eds.: COST 242: Broadband Network Teletraffi c. Volume 1155 of Lecture notes in computer science. Springer–Verlag (1996)
50. Conte, M., Más, I., Fodor, V., Karlsson, G.: Policy enforcing for probe–based admission control. In: Proc. of NTS 16, Espoo, Finland (2002) 45–55
51. Ventre, G., et al.: Quality of Service control in Premium IP networks. Deliverable 2.1, IST Project CADENUS — IST 11017 (2001)
52. Smirnov, M., et al.: SLA Networks in Premium IP. Deliverable 1.1, IST Project CADENUS — IST 11017 (2001)
53. Quittek, J., Zseby, T., Claise, B.: Requirements for IP Flow Information Export. Internet Draft, IETF (2002) (Work in progress).
54. ebXML Technical Architecture Project Team: ebXML Technical Architecture Specifi cation v.1.0.4. Technical specifi cation, ebXML Consortium (2001)
55. D'Antonio, S., Fadini, B., Romano, S., Ventre, G.: Designing Service Negotiation Entities for the Electronic Marketplace. In: Proceedings of SEKE2002, Ischia, Napoli — Italy (2002)
56. Cremonese, P., et al.: A Framework for Policy-based Management of QoS-aware IP Networks. In: Proceedings of Networking2002, Pisa — Italy (2002)
57. Chan, K., et al.: COPS Usage for Policy Provisioning (COPS-PR). RFC 3084, IETF (2001)
58. Rawlins, D., et al.: Framework of COPS-PR Policy Usage Feedback. Internet Draft, IETF (2002) (Work in progress).
59. Zhang, Z., Towsley, D., Kurose, J.: Statistical analysis of the generalized processor sharing scheduling discipline. In Proceedings of ACM SIGCOMM (1994) 68–77

60. Zhang, Z., Liu, Z., Kurose, J., Towsley, D.: Call admission control schemes under generalized processor sharing scheduling. The Journal of Telecommunication Systems, Modeling, Analysis, Design, and Management **7** (1997)
61. Elwalid, A., Mitra, D.: Design of generalized processor sharing schedulers which statistically multiplex heterogeneous qos classes. In Proceedings of IEEE INFOCOM '99 (1999) 1220–1230
62. Parekh, A., Gallager, R.G.: A generalized processor sharing approach to flow control in integrated services networks: The single-node case. IEEE/ACM Transactions on Networking **1** (1993) 344–357
63. Parekh, A., Gallager, R.G.: A generalized processor sharing approach to flow control in integrated services networks: The multiple node case. IEEE/ACM Transactions on Networking **2** (1994) 137–150
64. Szabo, R., Barta, P., Nemeth, F., Biro, J.: Call admission control in generalized processor sharing (gps) schedulers using non-rate proportional weighting of sessions. In Proceedings of INFOCOM '00 (2000)
65. Zhang, Z., Liu, Z., Towsley, D.: Closed-form deterministic end-to-end performance bounds for the generalized processor sharing scheduling discipline. Journal of Combinatorial Optimization **1** (1998)
66. Georgiadis, L., Gu'erin, R., Peris, V., Sivarajan, K.: Efficient network qos provisioning based on per node traffic shaping. IEEE/ACM Transactions on Networking **4** (1996) 482–501
67. Duffield, N.G., Lakshman, T.V., Stiliadis, D.: On adaptive bandwidth sharing with rate guarantees. In Proceedings of INFOCOM '98 (1998) 1122–1130
68. Chang, C.S., Chen, K.C.: Service curve proportional sharing algorithm for service-guaranteed multiaccess in integrated-service distributed networks. In Proceedings of GLOBECOM '99 (1999) 1340 –1344
69. Stamoulis, A., Giannakis, G.: Deterministic time-varying packet fair queueing for integrated services networks. In Proceedings of GLOBECOM '00 (2000) 621–625
70. Toutain, F.: Decoupled generalized processor sharing: A fair queueing principle for adaptive multimedia applications. In Proceedings of INFOCOM '98 (1998) 291–298
71. Panagakis, A., Stavrakakis, I.: Optimal call admission control under generalized processor sharing scheduling. In Proceedings of IWQoS '01 (2001)
72. Boudec, J.Y.L.: Application of network calculus to guaranteed service networks. IEEE Transactions on Information Theory **44** (1998) 1087–1096
73. Panagakis, A., Stavrakakis, I.: Generalized processor sharing enhancement through session decomposition. In Proceedings of Net-Con '02 (2002)
74. Sisodia, G., Headley, M.: Statistical analysis and simulation study of vbr coded video source models in atm networks. In: Proceedings of UPC. (1998) 177–181
75. Li, S.Q., Hwang, C.L.: Queue response to input correlation functions: discrete spectral analysis. IEEE Trans. on Networking **1** (1997) 533–552
76. Lombardo, A., Morabito, G., Schembra, G.: An accurate and treatable markov model of mpeg video traffic. In: Proc. of IEEE INFOCOM 1998. (1998) 217–224
77. MPEG traces archive: (http://www-info3.informatik.uni-wuerzburg.de/mpeg/traces/)
78. Heyman, D.: The gbar source model for vbr videoconferences. IEEE Trans. on Networking **5** (1997) 554–560
79. Heyman, D., Tabatabai, A., Lakshman, T.: Statistical analysis and simulation study of video teleconference traffic in atm networks. IEEE Trans. on Circ. and Syst. for Video Tech. **2** (1992) 49–59
80. Koucheryavy, Y., Moltchanov, D., Harju, J.: A novel two-step mpeg traffic modeling algorithm based on a gbar process. In: Proc. of NET-CON. (2002) 293–304

81. Lombardo, A., Morabito, G., Palazzo, S., Schembra, G.: Intra-gop modeling of mpeg video traffi c. In: Proc. of IEEE ICC. Volume 1. (1998) 563–567
82. Lombardo, A., Morabito, G., Palazzo, S., Schembra, G.: A fast simulation of mpeg video traffi c. In: Proc. GLOBECOM. Volume 2. (1998) 702–707
83. Blondia, C., Casals, O.: Performance analysis of statistical multiplexing of vbr sources. In: Proc. IEEE INFOCOM. (1992) 828–838
84. Koucheryavy, Y., Moltchanov, D., Harju, J.: A top-down approach to vod traffc transmission over diffserv domain using the af phb class. In: Proc. of IEEE ICC, Alaska, USA (2003)
85. Meyer, C.: Matrix analysis and applied linear algebra. SIAM Publications (2000)
86. Hajek, B., Linhai, H.: On variations of queue response for inputs with the same mean and autocorrelation function. IEEE Trans. on Networking **6** (1998) 588–598
87. Awduche, D.: MPLS and Traffi c Engineering in IP networks. IEEE Communications Magazine **37** (1999) 42–47
88. Rosen, E., Viswanathan, A., Callon, R.: Multiprotocol label switching architecture. RFC 3031, IETF (2001)
89. Awduche, D., Chiu, A., Elwalid, A., Widjaja, I., Xiao, X.: Overview and Principles of Internet Traffi c Engineering. RFC 3272, IETF (2002)
90. Blake et al., S.: An architecture for differentiated services. RFC 2475, IETF (1998)
91. Le Faucheur et al., F.: Multi-protocol label switching (MPLS) support of differentiated services. RFC 3270, IETF (2002)
92. Cho, K.: Alternate Queuing (ALTQ) module, (http://www.csl.sony.co.jp/person/kjc/programs.html)
93. Almesberger, W.: Linux network traffi c control - implementation overview. White paper, EPFL ICA (2001)
94. Almesberger, W., Hadi Salim, J., Kuznetsov, A.: Differentiated services on linux. Internet Draft, IETF (1999) (Work in progress).
95. Leu, J.R.: MPLS for Linux, (http://sourceforge.net/projects/mpls-linux)
96. Avallone, S., Esposito, M., Pescapé, A., Romano, S., Ventre, G.: Measuring MPLS overhead. In: Proc. of ICCC2002, Mumbai, India (2002)
97. Avallone, S., D'Arienzo, M., Esposito, M., Pescapé, A., Romano, S., Ventre, G.: Mtools. IEEE Network **16** (2002) 3 Networking column.
98. Avallone, S., Esposito, M., Pescapé, A., Romano, S., Ventre, G.: Mtools: a one-way delay and round-trip time meter. In Mastorakis, N., Mladenov, V., eds.: Recent Advances in Computers, Computing and Communications. (2002)
99. Avallone, S., Esposito, M., Pescapé, A., Romano, S., Ventre, G.: An experimental analysis of Diffserv-MPLS interoperability. In: Proc. of ICT 2003, Papeete, French Polynesia (2003)
100. Recommendation G.114, I.T.: General Characteristics of International Telephone Connections and International Telephone Circuits: One-Way Transmission Time (1998)
101. Cole, R., Rosenbluth, J.: Voice over IP Performance Monitoring. ACM Computer Communication Review (2002)
102. ITU-T Recommendation G.107: The E-Model, a computational model for use in transmission planning (1998)
103. L.F.Sun, G.Wade, B., E.C.Ifeachor: Impact of Packet Loss Location on Perceived Speech Quality. In: Proceedings of 2nd IP-Telephony Workshop (IPTEL '01), Columbia University, New York (2001) 114–122
104. Kitawaki, N., Kurita, T., Itoh, K.: Effects of Delay on Speech Quality. NTT Review **3** (1991) 88–94
105. Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V.: RTP: A Transport Protocol for Real-Time Applications. RFC 1889, Internet Engineering Task Force (1996) http://www.rfc-editor.org/rfc/rfc1889.txt.
106. Li, F.: Measurements of Voice over IP Quality. Master's thesis, KTH, Royal Institute of Technology, Sweden (2002)

107. Hagsand, O., Hansson, K., Marsh, I.: Measuring Internet Telephone Quality: Where are we today ? In: Proceedings of the IEEE Conference on Global Communications (GLOBECOM), Rio, Brazil, IEEE (1999)
108. Bolot, J., Crepin, H., Garcia, A.: Analysis of audio packet loss in the internet. In: Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV). Lecture Notes in Computer Science, Durham, New Hampshire, Springer (1995) 163–174
109. Maxemchuk, N.F., Lo, S.: Measurement and interpretation of voice traffi c on the Internet. In: Conference Record of the International Conference on Communications (ICC), Montreal, Canada (1997)
110. Lin, D.: Real-time voice transmissions over the Internet. Master's thesis, Univ. of Illinois at Urbana-Champaign (1999)