Traffic shaping based on an exponential token bucket for quantitative QoS: implementation and experiments on DiffServ routers

E. Vayias\textsuperscript{a},* J. Soldatos\textsuperscript{b}, G. Kormentzas\textsuperscript{c}

\textsuperscript{a}Intracom SA, 19.5km Markopoulou Ave, Paiania 19002, Greece
\textsuperscript{b}Athens Information Technology, 19.5km Markopoulou Ave, Paiania 19002, Greece
\textsuperscript{c}University of Aegean, Karlovassi, Samos 83200, Greece

Abstract

Support for quantitative QoS guarantees and associated Per-Hop Behaviours in a Differentiated Services architecture hinges on the specification of appropriate traffic control mechanisms. However, recent and past studies of Internet traffic have shown that it is rather hard to find a global and representative traffic model, which at the same time remains simple enough to facilitate the implementation of such mechanisms. In some previous works, we have demonstrated that suitable traffic shaping of packet flows that are subsequently multiplexed at a network node can permit the application of the M/G/1 model for performance modelling and control of the aggregate traffic stream. Using some handy control laws derived from the M/G/1 model, quantitative QoS guarantees can be provisioned end-to-end, across a series of multiplexers. In this paper we present our experiences from embedding this traffic-shaping scheme, which we call exponential token bucket filtering (etbf), in routers based on the Linux operating system. Initially, we describe how the traffic-shaping scheme can fit in a Differentiated Services architecture as a mechanism applied on the traffic transmitted from the edge routers to the core of a DiffServ domain, so that traffic multiplexed at the core routers conforms to the M/G/1 model. The Linux traffic control architecture is then presented, which enables the implementation of a DiffServ framework as well as how this shaping scheme is implemented within this architecture. Finally, results from experiments in a lab environment consisting of such Linux edge routers are presented. These results are in-line with findings from experiments and simulations in our previous works and show that the performance at the core multiplexing nodes can be controlled according to the M/G/1 model.

Keywords: Traffic Control; QoS; Differentiated services; Linux; Experiments; Token bucket; Shaping

1. Introduction

Defining Per-Hop Behaviours that can guarantee quantitative QoS (even in a statistical sense) in a Differentiated Services architecture still remains an open issue. As a result, sensitive traffic that requires specific (even probabilistic) loss and/or delay bounds is usually treated by the Expedited Forwarding (EF) PHB in a DiffServ environment, which merely gives to the aggregated stream of sensitive traffic absolute priority over any other class. This way it is not possible to support more than one class that demands quantitative and/or stringent QoS guarantees. The research community is still investigating the development of traffic control mechanisms enabling quantitative QoS control [1].

Developing such mechanisms is not a trivial task, mainly due to the complexity and multi-level heterogeneity of the Internet (i.e. various link layers, routing protocols and algorithms, protocol versions and implementations, etc.). Furthermore, it is rather hard to find a global and representative model for Internet traffic, which at the same time remains simple enough to enable effective and easy to implement traffic control mechanisms. The problem of Internet traffic modelling is further aggravated due to the constantly changing nature of Internet traffic [2,3].

In order to cope with the complexities of packet-level traffic control and yet provide mechanisms for controllable and quantitative QoS, a framework for packet level traffic control that relies on the M/G/1 queuing model has been proposed and verified. The core of this framework is an algorithm for shaping of traffic streams through packet spacing [4]. This algorithm is quite general, since it can be
applied to virtually any kind of packet-based traffic (non-
stationary, non-Markovian, heterogeneous, long-range
dependent, etc.) in order to produce streams with specific
multiplexing behaviour at network nodes. Packet spacing is
based on a non-linear law, which leads in a tight upper
bound to the queue-length distribution at a multiplexer fed
by many independent shaped streams. This upper bound
provides a foundation for statistical yet quantitative QoS
guarantees, which is one of the virtues of the framework. It
should also be noted that the packet spacing law enforces a
specific effective rate to the traffic stream. The enforced
effective rate features the convenient additivity property,
allowing easy derivation of traffic control and accounting
functions, such as policing, admission control, bandwidth
allocation and charging.

The theoretical background and the traffic control laws
of this framework have been presented in previous works
[5–7]. They have also been studied experimentally and
verified by a rich set of simulation and experimental results
based on artificially generated traffic following diverse
models (exponential, uniform, heavy-tailed, heterogeneous
traffic mix, etc.) [7]. Furthermore, experiments and simu-
lations with real traffic traces have investigated the
implications of applying this kind of traffic shaping in
operational networks. In particular, the effect of the delay
introduced by the packet spacing law, as well as strategies for
assigning effective rates to the various traffic streams have
been studied in [8].

Having this set of sound theoretical and experimental
results at hand, this paper focuses on the implementation
of the proposed shaping law and its accompanying ‘M/G/1’-
based traffic control functions in actual network devices
(routers) and specifies how these results can be incorpo-
rated into a DiffServ architecture. These are complemented
by a set of experiments conducted in a testbed with routers, in
which these traffic control schemes were implemented. The
purpose of these experiments is not to exhaustively analyse
and validate the proposed traffic control scheme (this has
already been done in [6–8]), but rather to demonstrate that
the scheme can be incorporated in a real implementation
without sacrificing its properties. In terms of implemen-
tation platform, Linux was selected mainly due to the fact
that it provides a rich and extensible traffic control
architecture giving ample room for embedding new traffic
control mechanisms. Thanks to their traffic control
capabilities, Linux routers can be appropriately configured
to act as either edge or core routers in a DiffServ framework.

The rest of the paper is structured as follows. Section 2
briefly reviews the theoretical underpinnings of our frame-
work and illustrates the traffic control scheme for quantitative
QoS. Section 3 elaborates on the application of this ‘M/G/1’-
based traffic control scheme in the scope of a DiffServ
network. This application is particularly important given that it
drives the real implementation described in this paper. Section
4 presents the main principles of traffic control in Linux routers
and emphasizes on the embedded implementation of the traffic
control scheme. Section 5 reports experimental results
obtained on a DiffServ network testbed with Linux routers
incorporating this scheme. Finally, Section 6 draws our main
conclusions regarding the implementation of the traffic control
laws, as well as their operation in a DiffServ network domain.
Note that the presentation of our framework in Section 2 is
done for completeness reasons. Readers interested in an in-
depth coverage and experimental analysis and validation
should consult [5–8].

2. Packet level traffic control based on the M/G/1 model

2.1. Overview of M/G/1 modeling results

In [5,6], it was demonstrated that the M/G/1 queuing
model is governed by equations that can be handy for traffic
control of packet flows. The most important of these results
are briefly reviewed. A multiplexer with service rate C and a
buffer size large enough to be considered infinite is
considered. It is assumed that this multiplexer serves a
packet stream of M/G type, i.e. with exponentially
distributed packet arrivals with mean arrival rate \( \lambda \) and
generally distributed packet sizes with \( V(s) = E(e^{-sv}) \)
the moment generating function of the packet size
(denoted by \( v \)). The multiplexer is then a M/G/1 server
and its queue-length distribution is asymptotically governed
by the dominant (i.e. less negative) solution, \( q_0 \), of the M/G/
1 characteristic equation

\[
q_0 = \sup \{ q : \frac{\lambda}{C} (1 - V(q)) \},
\]

i.e. a Chernoff bound holds in the form

\[
\gamma e^{q_0x} \leq W_c(x) \leq e^{q_0x},
\]

where \( W_c(x) = \Pr\{ \text{queue length} > x \} \) is the queue-length
Complimentary Probability Distribution Function (CPDF);
and \( \gamma \) some positive number \( \in (0,1) \). It is noteworthy that
other works [9–12] provide more accurate and tight bounds.
However, we use Eq. (2) mainly due to its generality (i.e. no
particular assumptions about incoming traffic and service
discipline are made) and the fact that it yields very
convenient asymptotic delay formulas. In [5], it was shown that \( \ln W_c(x) \) becomes more tightly bounded by
\( q_0x \), the more the load of the multiplexer approximates 1, i.e.
\( q_0 \) is the asymptotic slope in logarithmic scale of the queue
length CPDF (see Fig. 1).

This asymptotic slope can be used as a probabilistic yet
quantitative QoS metric for both delay and packet loss. It
determines the QoS level seen by the packet stream at a
node’s queue. Consider that \( q \) is expressed as a ratio:

\[
q = \frac{\ln p}{V_b}
\]

If we let \( V_b = B \), the capacity of the queue, then \( q \) is
the slope required to bound the packet loss ratio by
constraint. Similarly, if we set $V_{\text{s}} = CT_D$, then $q$ expresses a probabilistic delay constraint: if the log-scale CPDF slope is $q$, then the queuing delay in the multiplexer is bounded by $T_D$ with a probability $Pr\{\text{queuing-delay} > T_D\} = p$. As an example, consider a multiplexer featuring a capacity of 100 Mbits/s and having a queue of $10^6$ bits size. Then a QoS metric denoting a delay exceeding the value of $10^6$ bits/100 Mbps = 10 ms with probability 0.1% (which is also the probability of dropping a packet) corresponds to $q = \ln 10^{-3}/10^6 = -6.91 \times 10^{-6}$.

For a desired $q$, one may calculate the required service rate $C(q)$ of the multiplexer, so that a specific CPDF slope (and corresponding QoS level) is maintained. Solving (1) for $C$, gives

$$C(q) = r \frac{1 - V(q)}{q \bar{v}}. \tag{3}$$

where $\bar{v}$ is the average packet size and $r = \lambda \bar{v}$ is the mean rate of the stream.

This service rate $C(q)$ is also called the Effective Rate (ER) $f(q)$ of the $M/G/1$ stream for the specific QoS setting $q$. In [4,5], it is proven that the effective rate of a stream in an $M/G/1$ system (calculated by (3)) is summable, i.e. the total rate required for two or more independent streams to maintain a specific QoS figure $q$ is equal to the sum of the rates required by the individual streams for the same QoS, independently of their packet size distributions. This expresses the additivity property of the effective rates, which allows for an easy calculation of the required resources.

2.2. A traffic-shaping scheme for enforcing $M/G/1$ model on a multiplexer

Traffic shaping through packet spacing can be employed in order to produce streams of desired ER, given a target $q$, through suitable spacing of the data units of the stream (shaping). Assuming that packets are transmitted batch-wise (i.e. with an arbitrarily high peak rate), packet spacing hinges on controlling the (statistics of the) inter-packet distance. This control is based on (3), where the specific case of an $M/D/1$ system is considered for reasons outlined in [5,6]. In the case of $M/D/1$, Eq. (3), which is the basic control law, becomes:

$$f = r \frac{1 - e^{-q\bar{v}}}{q \bar{v}}. \tag{4}$$

This equation determines the relation between the mean rate $r = \lambda \bar{v}$ and the effective rate $f$ (actually the load $\rho = rf$) for a specific $q$. Recalling that $r = \lambda \bar{v}$; the mean inter-packet distance $\bar{s} = 1/\lambda$ corresponding to (4) is:

$$\bar{s} = r \frac{1 - e^{-q\bar{v}}}{f} \tag{5}$$

Thus, the spacing law that has to be applied on a packet stream with arbitrary packet size distribution, towards producing a stream of mean rate $r$, which maintains the performance defined by $q$ when serviced with rate $f$, is to enforce after each packet of size $\nu$, a distance $s$ between the current and the following packet, where $s$ is calculated by:

$$s = \frac{1 - e^{-q\nu}}{f} \tag{6}$$

This packet spacing law is presented in [4] in a credit-based version, where some credit thresholds have been introduced to allow for some tolerance in packet positioning. These thresholds minimize shaping delay, since a source that remains in a 'silence' state for some time is rewarded by accumulating credits, so that it can burst more aggressively when data for transmission become available. It also alleviates the need for re-shaping after each multiplexing stage. Using the notion of credits, a stream of packets as shaped for a multiplexing quality $q$ and an effective rate $f$ could be defined, if the associated quantity:

$$c \equiv \left(\sum_{n} \frac{1 - e^{-q(nT)}}{qf}\right) - T, \tag{7}$$

named credit, remains between specific bounds $[e^{-n}~c^+]$ for any arbitrary time window of duration $T \gg (c^+ - c^-)$; in the above formula, the summation index, $n$, includes all the packets within the considered time window. A shaped stream, according to the above definition, remains shaped even after passing through a network with a variable (but bounded) delay, provided that we are allowed to enlarge the credit bounds by such maximum delay.

The solution to the bandwidth allocation problem for a multiplexer, which serves $N$ independent streams shaped according to the presented packet spacing algorithm and conforming to the same bound for their queue length distribution is given by the sum of the effective rates of the individual shaped streams (i.e. $C = \sum_{i=1}^{N} f_i$).

2.3. Controlling QoS parameters end-to-end

The above mechanisms can be directly used for establishing and controlling specific 'per-hop-behaviours', according to the DiffServ terminology. However, end users...
are interested in the end-to-end QoS for traffic crossing a considerable number of network nodes and domains. The two main QoS parameters of interest are the transfer delay and the data (packet) losses.

Each ‘per-hop-behaviour’ can be defined by a specific slope (in log scale) of the queue-size tail distribution (CPDF), expressed by \( q \). Towards computing the transfer delay, we suppose that a bandwidth \( C \) is serving this queue. Note that this bandwidth is tuned to the sum of the effective rates of the multiplexed streams. If \( d_{\text{min}} \) represents the minimum (constant) delay due to packet processing at a node, the total delay, \( d \), that a packet is experiencing when crossing the node is bounded on the right by the line \( qCd \) and on the left by a vertical line at \( d_{\text{min}} \) as shown in Fig. 2a. The transfer delay through a series of \( n \) nodes with similar performance is, respectively, bounded as shown in Fig. 2b, i.e. by the curve \( (c_n) \), corresponding to the \( n \)th convolution of the single-node delay p.d.f., and by the vertical line at \( nd_{\text{min}} \). This presupposes that the independence assumption for the traffic crossing the different nodes holds. The analytical expression of the curve \( (c_n) \) is given by the formula (see [7]):

\[
(c_n) : \ln \text{CPDF}_{q}(d) = \ln A_n(qCd),
\]

where \( A_n(z) \equiv e^z \sum_{j=1}^{n} \frac{(-z)^j}{(j-1)!} \) (8)

When traffic crosses nodes featuring different \( q \) values or different service rates, the end-to-end delay can in principle be derived in a similar manner, i.e. by convolving the delay p.d.f.s at successive nodes. However, the analytical expression of the result is not that straightforward if the nodes feature different performance characteristics. It becomes apparent that having similar delay characteristics at all nodes for a particular service class facilitates the management of the end-to-end QoS.

As far as end-to-end data loss is concerned, the respective end-to-end figure for a traffic flow crossing \( n \) nodes is (after the independence assumption)

\[
P_{\text{loss}} = 1 - (1 - p_{\text{loss}})^n \equiv np_{\text{loss}}
\]

3. Applying M/G/1 traffic control in a DiffServ framework

3.1. Definition of Per-Hop-Behaviours

As explained in Section 2.1, the asymptotic slope \( q \) of the queue length log-scale CPDF in a M/G/1 multiplexer can express a probabilistic quantitative QoS requirement for either loss or delay. Therefore, in a Differentiated Services architecture, a Per-Hop Behaviour for quantitative QoS can be defined by a specific value of \( q \) provided that the aggregate traffic stream offered at a multiplexer is of the ‘M/G’-type. This can be achieved by applying the shaping law of Section 2.2 on the individual flows aggregated in such a PHB, using the same \( q \) value with the PHB definition and effective rates \( f_i \) such that \( \sum f_i \leq C_{PHB} \), where \( C_{PHB} \) is the rate allocated to this PHB in a multiplexer (see Fig. 3). We can even define a set of different PHBs for different \( q \) values, e.g. PHB\((q_1)\), PHB\((q_2)\),...,PHB\((q_N)\), thus supporting more than one class of service that requires quantitative QoS guarantees. We will now present the traffic management mechanisms required in order to support such PHBs in a DiffServ network.

3.2. Traffic management at the backbone links

An important advantage of the DiffServ framework is the simple traffic management functionality required at the core of a DiffServ domain, i.e. when traffic is transmitted over backbone links within the domain. The only required functionality is the classification of a packet to a PHB according to the packet’s DSCP. Thus, a traffic aggregate is formed for each PHB, possibly served by a separate queue per PHB. An appropriate queue scheduling discipline should then guarantee the QoS required by each PHB.

For a PHB defined by a value of \( q \), say PHB\((q_i)\), a separate queue is needed at each link to serve the traffic aggregate transmitted out to the link. This queue will share the link bandwidth with other PHBs (either defined by other \( q \) values or other PHB types, such as EF, AF, etc.). We assume that a class-based scheduling discipline will control...
sharing of link bandwidth among the PHBs. For the PHB\((q_i)\) mentioned above, this discipline should conform to the following requirements:

1. Traffic aggregate of each PHB\((q_i)\) is served by a separate FIFO (First-In-First-Out) queue.
2. A minimum guaranteed portion \(C_i\) of the link’s bandwidth \(C\) must be allocated to the PHB\((q_i)\) queue.
3. The flows multiplexed in this queue should have been shaped by the algorithm presented in Section 2.2 for the specific value of the parameter \(q_i\). The sum of effective rates applied when shaping these flows should be lower or equal to \(C_i\).
4. If the class-based scheduling discipline of the link is based on a priority scheme, then the order of priority of the PHB\((q_i)\) queues should be in increasing order of \(q_i\), i.e. if \(q_1 < q_2 < \cdots < q_N\), then the order of the queues should be: PHB\((q_1)\), PHB\((q_2)\), ..., PHB\((q_N)\). This is because lower \(q_i\) values correspond to higher QoS level.

In general, if other PHBs are also served by a link then the order of queue priorities is suggested to be the following: first, the tight QoS PHBs (e.g. EF), then the PHB\((q_i)\) queues as above, then the qualitative and relative QoS PHBs (e.g. AF) and finally the Best-Effort service queue. The higher the QoS level of the PHB, the higher should be the priority of the relevant queue (see Fig. 4).

The above requirements minimize the inter-twinning with the resources of other PHBs, as long as effective rates \(f_i\) of the shaped flows are such that \(\sum f_i \leq C_i\).

### 3.3. Traffic management at the edge

The DiffServ architecture defines that at the edges of a domain, i.e. when traffic enters or leaves the domain, the required traffic management mechanisms can be implemented by a suitably configured ‘traffic conditioning component’. Such a component may consist of a classifier, a metre, a marker and a shaper or dropper in any suitable combination depending on the requirements of the PHBs as well as the direction of the traffic (entering or leaving the domain). Fig. 5 illustrates the general structure of a DiffServ traffic-conditioning component, also showing the particular parameters and techniques employed for incorporating our shaping scheme in each functional block of the component.

A classifier ‘selects’ packets either according to multiple fields of the IP header (multi-field—MF classifier) or simply according to their DSCP (behaviour aggregate—BA classifier). The metre measures some parameters of

---

**Fig. 3.** Multiplexing of shaped streams in a DiffServ network.

**Fig. 4.** Priority order of PHBs in a scheduling discipline.
the selected traffic flow. The marker changes the DSCP if necessary. The shaper/dropper buffers/drops packets not conforming to a required profile in order to create a flow, which is conforming to this profile.

Nodes transmitting traffic towards a DiffServ backbone will shape the transmitted flows (individually or by PHB aggregate) according to the algorithm in Section 2.2 for certain values of effective rate \( f \), \( q \) and threshold \( c_{max} \). Nodes at the ingress of a DiffServ domain will monitor the conformance of a flow to the shaping parameters agreed with the traffic source.

The algorithm of Section 2.2 can be incorporated in such a traffic conditioning component, either as a shaping algorithm or as a conformance monitoring (policing) algorithm. Re-marking of non-conforming packets can be implemented as well, using the marker functional block. Fig. 5 shows the incorporation of the shaping algorithm in a DiffServ traffic-conditioning component.

4. Implementation in Linux-based routers

4.1. Linux routing and traffic control functionality

The Linux operating system has the capability to route IP packets between the network interfaces of a Linux system and features a fairly rich set of traffic control functionality. The Routing and Traffic Control subsystem within Linux is easily configurable as well as extendable and lends itself to the implementation of a QoS architecture [13].

Packet processing by the Linux kernel is highlighted in Fig. 6. It consists of three main blocks. The input classification and de-multiplexing block separates traffic destined for the local system from traffic that should be forwarded out one of the system’s network interfaces. The latter is passed on to the forwarding block, which selects an output interface based on the system’s routing table. Each interface has an output queuing and traffic
control block, which can classify packets to one of several queues configured for this interface (either BA or MF classification) and manages how the link bandwidth is shared among these queues according to the QoS policy.

The operation of these blocks can be configured in order to implement a particular QoS architecture and policy using some Traffic Control configuration tools like the \texttt{tc} tool for the output queuing and traffic control block and the \texttt{route}, \texttt{ip} and \texttt{ipchains} tools for the other blocks.

The output queuing and traffic control block is the one actually implementing the entire Traffic Control functionality of Linux. Ref.\cite{14} gives an extensive overview of this functionality while \cite{15} presents how this functionality can implement a DiffServ QoS architecture. We now present the basic components of this block.

Each network device has a Queuing Discipline associated with it, which controls how packets enqueued on that device are treated according to a specific algorithm such as FIFO, Token Bucket, Weighted Fair Queuing, Random Early Drop, etc.

A queuing discipline may just consist of a single queue (classless discipline) or it may consist of different Classes of packets among which the bandwidth is shared according to the queuing discipline (class-based discipline).

Filters will be used to distinguish among packets of different classes. Note that multiple filters may map to the same class and, in general, filters can be combined arbitrarily with queuing disciplines and classes as long as the queuing discipline is class-based.

Each network interface is associated with a root-queuing discipline, which has access to the entire link bandwidth. In a class-based discipline, each class will be associated with another queuing discipline to take care of how packets of this class are enqueued and transmitted. That queuing discipline can be arbitrarily chosen from the set of available queuing disciplines, and it may well have classes, which in turn use queuing disciplines, etc. In this way, a hierarchy of queuing disciplines and classes can be built, with filters controlling how packets of a class-based discipline are classified to its classes. Fig. 7 shows an example of such a hierarchy.

Several well-known queuing disciplines have been implemented in the Linux kernel. Table 1 presents the main ones.

<table>
<thead>
<tr>
<th>Name</th>
<th>Algorithm</th>
<th>Class-Based</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{Fifo}</td>
<td>FIFO</td>
<td>No</td>
<td>First-In-First-Out queue</td>
</tr>
<tr>
<td>\texttt{Red}</td>
<td>Random Early Drop</td>
<td>No</td>
<td>FIFO queue with Random Early Drop \cite{16}</td>
</tr>
<tr>
<td>\texttt{Tbf}</td>
<td>Token Bucket Filter</td>
<td>No</td>
<td>FIFO queue shaping traffic according to a token bucket \cite{17}</td>
</tr>
<tr>
<td>\texttt{Prio}</td>
<td>Strict Priority</td>
<td>Yes</td>
<td>Strict priority among the classes</td>
</tr>
<tr>
<td>\texttt{Cbq}</td>
<td>Class-Based Queuing</td>
<td>Yes</td>
<td>Supports hierarchical link-sharing as in \cite{18}</td>
</tr>
<tr>
<td>\texttt{Csz}</td>
<td>Clark-Schenker-Zhang</td>
<td>Yes</td>
<td>Implements the CSZ scheduler \cite{17}</td>
</tr>
<tr>
<td>\texttt{Dsmark}</td>
<td>DS marking</td>
<td>Yes</td>
<td>Supports DSCP marking</td>
</tr>
</tbody>
</table>

Fig. 7. An example of a class-based queuing discipline.
the ‘linear’ token bucket) where:

\[ \bar{v} = \frac{1 - e^{-qv}}{q} \text{ tokens required} \]

The token rate \( f \) corresponds to the effective rate of Section 2.2 and the depth \( c_{\text{max}} \) to the maximum credit threshold. Fig. 8 depicts this correspondence.

The exponential rate relation between the actual and the adapted packet size, we have called our algorithm exponential token bucket filter (etbf), in analogy to the token bucket filter (tbf) already implemented in the Linux kernel. Therefore, we implemented etbf as a classless queuing discipline, just like the tbf discipline.

\[ c_{\text{max}} = f \cdot \frac{1 - e^{-qv \cdot mpu}}{q}, \]

meaning that \( f \) packets of size \( mpu \) can be transmitted at once if we reached the credit threshold.

5. Experimental results and evaluation

Using \texttt{tc} to configure the etbf and other Linux queuing disciplines, we may implement Per-Hop-Behaviors (PHB) targeting particular values of the QoS parameter \( q \). The configuration of core routers is straightforward, since it is a matter of activating a scheduling discipline (subject to requirements set in Section 3.1). Configuring leaf routers is, however, much more interesting. Leaf routers apply the etbf shaping algorithm to micro and aggregate flows, so that a particular QoS point \( q \) and aggregate effective rate are achieved. Based on our theoretical findings, the bandwidth assigned to the queues of the core nodes should be equal to the sum of the effective rates of the multiplexed flows, in order to guarantee QoS for the aggregate stream at multiplexing points. Measuring experimentally the QoS (in terms of buffer length CPDF) and auditing it against the target value of (CPDF slope) \( q \) allows for evaluating the effectiveness of our M/G/1-based control scheme on real network elements.

Fig. 9 depicts the physical configuration of the experimental network, which supported our performance evaluation experiments. The network includes two Linux routers connected to two other (sub)networks: an experimental IP over ATM network (through the atm0 network interface) and an Ethernet-based Local Area Network (through the eth0 interface).

In both Linux systems, we implemented and configured the etbf queuing discipline. These systems were configured as leaf routers, which shaped packet flows transmitted to the IP over ATM network. In this setup, the multiplexing point is not an IP router interface, rather an interface of the ASX-200BX ATM switch. The ATM interface was preferred over another IP router as a multiplexing point, due to the fact that our commercial ATM equipment provided for: (a) increased accuracy and fine granularity in configuring the bandwidth of the interface and (b) opportunities for monitoring the queue size of the interface. Moreover, by looping-back some output ports, a single ATM switch can emulate experiments with several
multiplexing stages. The experimental network also includes two measurement devices, namely the ATM BLTG (Burst-Level Traffic Generator) traffic generator [19] and its dual ATM BLTA (Burst-Level Traffic Analyzer) [20]. This network can be appropriately configured to support a wide range of experiments featuring two etbf enabled leaf nodes shaping flows for a given value of the QoS parameter \( q \).

5.1. Traffic generation and configuration of queuing disciplines

A key prerequisite to conducting the experiments was to generate IP packets following a desired packet size distribution. To this end, we used the \( \text{tg} \) Linux application. This application reads a parameter file, which defines the characteristics of the packet flows to be generated. These characteristics include: (a) the transport layer protocol (i.e. TCP or UDP), (b) the destination’s IP address and port, (c) the packet interarrival time distribution and (d) the (application layer) packet size distribution. A thorough description of the structure and content of \( \text{tg} \) files is given in [21].

In our experiments, we used the UDP protocol. UDP enables to tune the packet interarrival and packet size distributions. \( \text{tg} \) packet flows were filtered and directed to etbf queues and accordingly shaped based on the etbf algorithm. Filtering was performed based on the destination IP address. Using \( \text{tg} \), it is possible to generate flows resembling micro-flows (e.g. a VoIP session), as well as aggregate flows (e.g. an aggregate of multiple TCP flows). Moreover, packet flows stemming from more than one \( \text{tg} \) process can be directed to the same destination IP address and port, which allows a single etbf queuing discipline to shape traffic flows from multiple \( \text{tg} \) programs.

Towards implementing several instances of the etbf discipline and therefore many shaped flows in each of the Linux routers, we created a dsmark queuing discipline, as a root discipline for interface atm0. Under dsmark, we created another prio queuing discipline, which implements a priority service strategy coordinating several queuing disciplines. Furthermore, we attached a pfifo

Fig. 9. Experimental setup for evaluating the etbf implementation.

Fig. 10. Configuration of Queuing disciplines in an interface of the Linux routers.
queuing discipline to the queue with the highest priority, while we attached etbf queuing disciplines to other queues. We also created filters for the prio discipline, so that packet flows to destinations 147.102.39.101/udp/5015 up to 147.102.39.101/udp/5015 were directed to queues 0xf to 0xf respectively. These configurations imply that packet flows will be shaped by the etbf disciplines, while the 0x10 (pfifo) queue serves other packet flows (see Fig. 10).

The maximum number of queues that can be configured within a specific prio queuing discipline is 16, which implies that up to 15 independent rows can be shaped from each Linux router. Under realistic conditions, packet flows shaped by leaf Linux routers stem from other terminal PCs or routers as shown in Fig. 11(a). In the scope of our experiments, however, we ran the traffic generation programs (i.e. the programs generating packet flows) in linuxpc and linuxpc2, as shown in Fig. 11(b). This slight deviation provides simplicity and allows for better control of the experiment. Moreover, this configuration avoids multiplexing of tg flows in the incoming port of the router, which may cause distortions in the traffic profiles of the source flows.

5.2. Impact of the credit limit $c_{\text{max}}$

The first experiment concerned the impact of the credit value $c_{\text{max}}$, which is introduced in order to allow for some relaxation in tight packet spacing and thus reduce shaping delay. Specifically, in each Linux router, we configured 15 etbf queuing disciplines featuring the following parameters: Effective Rate = 20,000 bytes/s, $q = \ln 10^{-3}/2500$ bytes, mps (maximum packet size) = 1436 bytes and the credit limit $f$ taking values 1, 4, 5 and 10.

In every Linux router, we instigated traffic generation of 15 tg packet flows featuring exponentially distributed packet sizes with mean, minimum and maximum values of 1000, 10 and 1400 bytes, respectively. Traffic generation occurred at an appropriate rate, so that the shaper was always busy and maximum exploitation of the effective rate allocated to etbf queues was achieved. Hence, each packet flow was directed to an etbf queue for shaping, resulting in a total of 30 flows for both routers. These 30 flows were directed to an output port the ATM switching node. The output port rate was set to $C=12,810$ cells/s, which corresponds to the sum of the effective rates of the etbf queues. In calculating this cell rate, we had to make calculations relating to protocol conversions. In particular, etbf effective rates apply to data link layer PDUs (protocol data units), i.e. to IP packets encapsulated in AAL-5 PDUs. An application level packet of 1000 bytes becomes 1032 bytes as soon as we add the overhead of UDP, IP and AAL-5 headers. Hence, an effective rate of 2000 bytes/s corresponds to $20,000/1032 = 19.38$ packets/s. At the ATM layer, an AAL-5 packet of 1032 bytes corresponds to $1032/48 = 22$ cells, and therefore the 2000 bytes/s effective rate corresponds to $19.38 \times 22 = 427$ cells/s. As a result, the effective rate corresponding to 30 flows is $30 \times 427 = 12,810$ cells/s.

Fig. 12 depicts the results of the experiments corresponding to each one of the credit values. The figure shows that the curves corresponding to four different values of the credit limit almost coincide. Therefore, the credit limit parameter only marginally affects the QoS, as also discussed in simulation results in [8]. Note, however, that anyway all curves achieve the target QoS specified by the target CPDF slope $q$, which was used to configure the etbf shaping.

5.3. Impact of packet size distributions and number of multiplexed flows

In a next experiment, we studied the impact of different packet size distributions, as well as of the number of multiplexed flows. These two factors are directly related with the theory supporting our traffic control framework. Producing an aggregate stream of shaped flows that features exponential interarrival times (thus of M/G-type) can be achieved when there is significant randomisation in the packet sizes or conversely when a very large number of packet flows is multiplexed. In our earlier work [7], we showed that few flows following different packet size distributions could result in adequate randomisation in the interarrival times of the aggregate stream.
In this experiment, we examined a similar multiplexing scenario on our network testbed with Linux routers. We considered a small to medium number of packet flows (i.e. 15 and 30 flows, respectively) and various packet size distributions some of them featuring a strong or weak deterministic component (i.e. a high probability for one particular packet size or a small interval of packet sizes). Specifically, we used the packet size distributions in Table 2, listed in order of increasing deterministic behaviour.

Each Linux router was configured with 15 etbf disciplines with the following parameters: Effective Rate $Z_{35,000}$ bytes/s, $q = \ln 10^{-3}/2500$ bytes, credit limit $f = 4$ for packet size mps $Z_{1436}$ bytes.

Moreover, $tg$ was configured to produce 15 packet flows in each one of the routers. Packets were distributed according to the distributions shown in Table 2. Appropriate packet generation rates were used to allow for a full exploitation of the etbf effective rate. Each one of the $tg$ flows was directed in one of the etbf disciplines for shaping and then fed to the output port of the ATM switching node, where we audited the QoS offered at this multiplexing point. Accordingly, we performed the same experiment based on 15 shaped packet flows, which were shaped in linuxpc2 and accordingly directed to the same output port. In both experiments, the output rate was set to the sum of the effective rates of the multiplexed flows. This output rate was calculated as explained in the previous paragraph.

Fig. 13 depicts the curves derived from the experiments with the 15 packet flows and the packet size distributions illustrated in Table 2. The curve corresponding to the exponential distribution features the best similarity to the theory. Curves corresponding to distributions with deterministic components resemble results observed in deterministic systems [22]. These curves present a change of slope around the packet sizes that occur with the highest probability (i.e. sizes around 1300–1400 bytes). In the case of constant packet size (for all 15 shaped flows), the system behaves like a deterministic periodical system $nD/D/1$, rather than a Poisson $M/G/1$ system. Except for this case, other distributions and packet sizes even with strong deterministic components render results close to the theoretical ones.

Table 2

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Comment / Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Exponential distribution</strong>, Mean value $= 1000$, min $= 10$, max $= 1400$</td>
<td>Distribution with adequate randomisation, where the maximum value 1400 corresponds to a common value for the MTU at the application layer</td>
</tr>
<tr>
<td><strong>Uniform distribution</strong>, interval [1250,1350]</td>
<td>Distribution with some randomisation, however within a small range of packet sizes</td>
</tr>
<tr>
<td><strong>Hybrid distribution</strong>, 20% uniform distributed packet sizes in the range [11,1379] and 80% packets with constant size 1380</td>
<td>Distribution with a strong deterministic component, as well as with a random component. A similar distribution is derived from analysing packets from multiple TCP sessions$^a$</td>
</tr>
<tr>
<td><strong>Hybrid Distribution</strong>, 10% uniform distributed packet sizes in the range [11, 1379] and 90% constant size packets of 1380</td>
<td>Distribution similar to the previous one, featuring however an even stronger deterministic component</td>
</tr>
<tr>
<td><strong>Uniform Distribution</strong>, range [1280, 1320]</td>
<td>Distribution within a limited range of sizes resulting in almost deterministic behaviour</td>
</tr>
<tr>
<td><strong>Constant packet size</strong> $= 1300$</td>
<td>Deterministic packet size</td>
</tr>
</tbody>
</table>

$^a$ A TCP session transferring X bytes will practically generate $[X/MTU]$ packets with MTU size and one packet with uniformly distributed size in the interval [0,MTU-1]. The total flow of such session will produce several packets with MTU length, as well as some packet with sizes in the range [0,MTU-1] with almost equal probability for each size value. Thus, few and large (compared to the MTU) TCP sessions will increase the percentage of packets with MTU length, while many and small sessions reduce the same percentage.
Smaller randomisation in the packet sizes results in a lower observed CPDF curve, which means that our traffic control framework acts in a conservative fashion, the more deterministic the packet size. The positive aspect, however, is that in no case, the scheme becomes extremely conservative with regard to resource provisioning.

Fig. 14 depicts experimental curves corresponding to the CPDF of the output queue of the ATM switch corresponding to three of the packet size distributions depicted in Table 2, this time, however, for different numbers of multiplexed flows (i.e. 15 and 30 flows). The number of multiplexed flows from 15 to 30 has a significant impact only in the case of constant packet size distributions.

Fig. 15 illustrates the CPDF of the interarrival times for the aggregate stream composed by shaping flows with packets following a hybrid size distribution with 90% constant-size packets and 10% packets following a uniform distribution. The observed interarrival times do not follow an exponential distribution. This discrepancy explains the difference between the CPDF of the queue length and the expected theoretical curve.

5.4. QoS measurements for shaped traffic flows emulating Voice over IP

In another scenario, we also studied the case of multiplexing shaped packet flows that emulate Voice over IP sessions. According to voice compression algorithm ITU-T G.723.1, a digital voice signal can be compressed in a way that its bandwidth is reduced to 6.3 Kbps. This means that a data unit containing 30 ms of G.723.1-based compressed voice has length of 189 bits (or 24 bytes). Considering also the minimum header of RTP protocol (i.e. 12 bytes), the average length of the data unit given to UDP will be 36 bytes. Taking flows with uniform packet length distribution from 35 to 40 bytes, we covered a slight variance in the above packet length. Given that a voice source alternates between ‘talk’ and ‘silence’, we modelled...
the packet generation procedure of a voice flow with a two-state Markov chain. Each state had an average duration of 5 s. For the ‘talk’ state, there was a constant packet generation rate equal to one packet every 0.03 s, while for the ‘silence’ state, the source was inactive.

The average rate of such a packet flow at the Data Link Layer (with UDP, IP and AAL5 headers equal to 36 bytes) is \( \frac{1}{2}(36 + 36) \text{ bytes/0.03 s} = 1233 \text{ bytes/s} \), while the maximum rate (in the ‘talk’ state) is \( \frac{38 + 36}{0.03 \text{ s}} = 2467 \text{ bytes/s} \).

Each voice flow was shaped according to an \texttt{etbf} discipline with parameters: Effective Rate = 1650 bytes/s, \( q = \ln 10^{-3}/500 \text{ bytes} \), credit limit factor \( f = 4 \) for packet length \( m\text{ps} = 74 \text{ bytes} \). We selected this value for \( q \) in order to have a more realistic QoS requirement. The selected value for 30 flows with 1650 bytes/s, corresponds to a delay of 500 bytes/(30 \times 1650 \text{ bytes/s}) = 10 ms with probability \( 10^{-3} \) per node. These 30 flows were multiplexed at an output port of an ATM switch with an output rate of \( 30 \times (1650 \text{ bytes/s}) / (74 \text{ bytes/pkt}) \times (74/48(\text{cells/pkt} = 1350 \text{ cells/s}) \).

Fig. 16 depicts the resulting queue-length CPDF, which is in-line with the theoretical results (target CPDF slope \( q \)).

It is important to note that while shaping the ‘VoIP’ flows, it was observed\(^1\) that the queues of the \texttt{etbf} disciplines did not add any significant delay. This means that assigning an effective rate of 1.65 KB/s, i.e. 34% more than the average rate (1.233 KB/s), an overall good performance can be achieved concerning the transfer delay (i.e. about 10 ms per node with probability \( 10^{-3} \) and minimal shaping delay). Shaping delay was minimized due to using the credit-based version of the shaping algorithm [4,7,8]. According to this version, when a source is ‘silent’, it is rewarded by accumulating credit, so as to allow it to burst more aggressively when there is data to transmit. Credits are accumulated in a way that respects the effective rate to be enforced. This is the way ‘talk’-‘silence’ sources can be

\(^1\) For this experiment scenario, we could only observe and not record the delay imposed by the queues of the \texttt{etbf} schemes.
accommodated with manageable shaping delay by the shaping algorithm.

5.5. End-to-end delay measurements in a cascaded multiplexers setting

This section discusses a set of delay measurements performed in a cascaded multiplexers setting loaded by shaped packet flows. For this set of measurements, we used the BLTG and BLTA experimental tools in a setting such as the one depicted in Fig. 9. BLTA can measure and record the end-to-end transfer delay of a received flow, generated by BLTG. For our experiments, BLTG generated a test CBR flow of a small rate, e.g. 1 cell/0.025 s. The distribution of the measured delay for such a flow corresponds to the combined distribution of the queue length of the cascaded multiplexers.

In the first of our experiments in such a setting, the test flow generated by BLTG, passed through the two multiplexers, and reached BLTA, where the delay experienced by each ATM packet was recorded. For this scenario (see Fig. 17a), the two multiplexers were loaded with 15 completely independent flows. Each such flow was shaped according to an \( ef \) discipline with parameters: Effective Rate = 35,000 bytes/s, \( q = \ln 10^{-7}/2500 \) bytes, credit limit factor \( f = 4 \) for packet length \( mps = 1436 \) bytes.

The second experiment (see Fig. 17b) referred to a more common case, where there is correlation between the loads of the cascaded multiplexers. In this case, the load of the second multiplexer, besides 10 independent shaped flows, included also five flows coming from the first multiplexer.

For both scenarios, two different packet length distributions for the shaped flows were used:

- An exponential distribution with average value 1000 bytes, minimum value 10 bytes and maximum value 1400 bytes.
- A hybrid distribution with constant packet length 1380 bytes with probability 80%, and uniform distributed length in the interval \([11,1379]\) bytes with probability 20%.

The output rate \( C \) of multiplexers’ ports A2 and A4 had a fixed value equal to the sum of the rates of 15 shaped flows and the constant rate of the test flow. For the exponential distribution case, we had

\[
C = 15 \times 35000/1000 \times [1000/48] + 40 = 11065 \text{ cells/s,}
\]

while for the hybrid distribution case we had

\[
C = 15 \times 35000/1243 \times [1243/48] + 40 = 11020 \text{ cells/s.}
\]

By dividing the measured delay values on BLTA with the appropriate \( C \) value, we can get the corresponding combined queue length value that a specific ATM packet encountered through the multiplexers. For the exponential

Fig. 17. Experimental settings for the delay measurements in two multiplexers in tandem.
distribution case. Fig. 18 depicts the resulted queue length CPDF for both the two cascaded multiplexers (measured with BLTA) as well as only for the second multiplexer (measured with a buffer sampling technique), for the case of exponentially distributed packet sizes. Fig. 19 depicts the corresponding CPDFs for the hybrid distribution case. The experimental curves are compared with their corresponding theoretical curves shown in Fig. 2.

Regarding the theoretic queue length CPDF for the case of the two cascaded multiplexers, it comes from Eq. (8) for \( n = 2 \) that

\[
W_{c,2}(x) = (-qx + 1)e^{qx}.
\]

Setting \( q = \ln 10^{-3}/2500 \) bytes and queue length \( x = 2500 \) bytes we have

\[
Pr_{n=2 \text{ multiplexers}} \{ \text{queue length} > 2500 \text{ bytes} \} = 7.91 \times 10^{-3}.
\]

On both figures, we observe that the results are always more conservative than the theoretical curves, however, with no considerable deviation. Moreover, although in the Fig.17b scenario, part of the flows offered to the second multiplexer are not re-shaped, there is no considerable influence on the observed result. These scenarios provide evidence that the proposed traffic control laws are
appropriate for controlling QoS parameters (such as delay) end-to-end across a series of nodes.

6. Conclusions

Several traffic control results targeting IP-based QoS in a DiffServ context have been proposed in the research literature. Most of these efforts emphasize on the qualitative differentiation of packet flows into different traffic classes. In this work, we have emphasized on a traffic control framework that can deliver probabilistic, yet quantitative QoS in IP packet flows. The theoretical foundations of this framework have been thoroughly illustrated in our earlier works and are only briefly discussed in this paper for the sake of completeness. At the heart of this framework lies a non-linear traffic-shaping algorithm that takes into account the packet size.

This control framework allows for establishing Per-Hop-Behaviours (PHB) at multiplexing points. PHBs are established on the basis of the traffic-shaping scheme: a PHB corresponds to a set of shaping parameters targeting a particular QoS level pertaining to the PHB. We have therefore shown that the proposed traffic control framework fits the traffic control components of DiffServ architecture, towards implementing quantitative QoS PHBs. While the concepts and the vision of the traffic control components are straightforward, applying them in a realistic network environment presents several challenges. These stem mainly from the fact that several assumptions might not be valid in an operational context. Therefore, we have put emphasis on validating our traffic control framework on a real network testbed. To this end, we have exploited the rich set of options for implementing traffic control functionality offered by the Linux operating system. In particular, we have extended the traffic control functionality of Linux kernel, incorporating the proposed shaping algorithm. A proof-of-concept implementation of the shaping algorithm has been carried out, enabling Linux routers to shape and forward flows.

We set up a small-scale network testbed, comprising (apart from the Linux routers) an ATM switching node, as well as traffic generation and analysis utilities. This testbed allowed for the configuration of PHBs, targeting several experimental scenarios. One particular scenario studied whether allowing small perturbations in the shaped streams can improve the QoS. These small perturbations can be expressed in the form of credits allowing slight deviation from the strict distances of packets within a flow, as the latter are enforced by the shaping law. These perturbations are useful in order to minimize the shaping delay experienced when shaping bursty sources (such as ‘talk-silence’ sources). Experiments showed that allowing such perturbation do not influence the observed QoS, while they assist in minimizing the shaping delay.

Other experiments studied the Poisson assumption required to apply the M/G/1 theory on shaped streams. Specifically, we studied the effect of aggregating different numbers of flows, as well as flows featuring a variety of packet size distributions. Experiments proved that even for a small number of flows, it is realistic to apply the proposed framework, as soon as the packet size of the multiplexed flows does not feature a very strong deterministic component. In particular, the traffic control laws deliver the expected QoS when multiplexing 15 flows generating packets according to various distributions. Deviations from the theoretically expected behaviour arise for traffic sources generating packets of either constant size or distributed within a very short range. In cases where the packet size is deterministic, the behaviour of the multiplexer resembles an nD/D/1 system rather than M/G/1.

Experiments based on traffic models that emulated ‘VoIP’ sessions have shown that the schemes can be used in order to guarantee QoS for such traffic.

Apart from experiments addressing a single multiplexing point, we investigated scenarios with more than one multiplexers in tandem. Specifically, we considered cascading nodes that supported the same PHBs, and proved that the end-to-end packet delay is distributed according to the convolution of the respective distributions of the individual nodes. These scenarios provide evidence that the proposed traffic control laws are appropriate for controlling QoS parameters (such as delay), end-to-end within a network domain. This is particularly important for supporting the wave of emerging business critical IP QoS applications (e.g. Voice-over-IP), which pose stringent QoS guarantees.

In summary, we have demonstrated that the proposed traffic control scheme can be implemented within a router’s traffic control architecture without sacrificing its properties and can be incorporated into a DiffServ framework, in order to support quantitative QoS PHBs. We showed how suitable PHBs can be defined and how the shaping algorithm can be applied at the edge nodes of a DiffServ domain to enforce such PHBs. With appropriate dimensioning of the resources guaranteed to these PHBs at the internal network nodes, quantitative QoS can be offered. Although our objective was not to study the traffic types that are more suitable for such QoS control, our experiments have shown that the traffic control scheme is applicable both to aggregate TCP traffic (resembling a uniform plus constant packet size distribution) as well as to ‘VoIP’-like traffic.

Applying the shaping algorithm to a number of flows flowing through a Linux-based router requires tight job scheduling from the operating system’s side. Due to the interrupt-driven nature of the Linux traffic control architecture and the scheduling quantum of the basic Linux kernel (1 ms), the combined packet rate of all shaped flows should be limited to 1000 packets/s. Our future work consists in porting our implementation to a platform which supports more fine-grained scheduling. Such a platform is the RTLinux kernel (http://www.rtlsacle). We are also investigating whether the shaping algorithm can be incorporated, not in the operating system kernel, but rather in the driver of a network interface.
adapter. To this respect, we will evaluate the porting of the algorithm to a network interface adapter with programmable packet scheduling capabilities that is being developed in the National Technical University of Athens.

The presented results have been derived in configurations with only one class of service. Another direction of our future work will consist in validating our approach in a multi-class setting in the sense of Fig. 4, i.e. in the presence of a tight QoS, high priority PHB, a best effort PHB and at least another PHB featuring a different value of $q$.

References


Evangelos K. Vayias (evag@intracom.gr) received the Dipl.-Eng. and PhD degrees, both from the department of electrical and computer engineering, National Technical University of Athens, Greece, in 1996 and 2000, respectively. He is now a project consultant with Intracom SA, network professional services department, specializing in next-generation IP-based network technologies. Prior to this, he was a research associate with the Telecommunications Laboratory of the National Technical University of Athens, where he worked in numerous research projects in the areas of resource management and performance evaluation of broadband networks, distributed object-oriented technologies for network management, interactive multimedia services, and grid computing. He also consulted for several public and private organizations in Greece. He is a member of the Technical Chamber of Greece and a Cisco Certified Internetwork Expert.

John K. Soldatos (jsol@ait.edu.gr) was born in Athens, Greece in 1973. He obtained his Dipl.-Eng. degree in 1996 and his PhD in 2000, both from the electrical and computer engineering department of the National Technical University of Athens (NTUA). He has had an active role in several EU co-funded research projects. Specifically, he has actively participated in EXPERT AC-094, WATT AC-235, IMPACT AC-324, Chameleon EP 20597, CATCH-2004 IST-1999-11103, and LION IST-1999-11387 projects, and is now with Athens Information Technology (AIT), involved in the CHIL-FP6-506909 project. He has also consulted in many ICT projects for leading Greek enterprises (INTRACOM S.A, PEGASUS S.A, IBM Hellas S.A, OTE S.A, TEMAGON S.A). Dr. Soldatos has extensively lectured in NTUA and AIT, and has given numerous invited lectures. As a result of his activities he has co-authored more than 45 papers published in international journals and conference proceedings. Since March 2003 he has been with AIT, where he is currently an assistant professor. His current research interests are in broadband traffic control, network management, grid computing, and ubiquitous and autonomic computing.

George Kormentzas (gkorm@aegean.gr) is currently a lecturer at the University of the Aegean, department of information and communication systems engineering. He received the Diploma in electrical and computer engineering and the PhD in computer science, both from the National Technical University of Athens (NTUA), Greece, in 1995 and 2000, respectively. From 2000 to 2002, he was a research associate with the Institute of Informatics & Telecommunications of the Greek National Center for Scientific Research ‘Demokritos,’ working in a number of EU funded research projects in the area of high-speed wired and wireless networks. His research interests are in the fields of traffic analysis, network control, resource management, and quality of service in broadband networks. He has more than 40 publications in books, journals, and international conference proceedings in the above areas. He is a member of IEEE, IFIP, and of the Technical Chamber of Greece.