Software Techniques for Multiservice IP Networks

Giorgio Calarco

D.E.I.S. - University of Bologna, Viale Risorgimento, 2 - 40136 Bologna (ITALY)
giorgio.calarco@unibo.it

Abstract. The interest of the scientific and commercial telecommunications community for the use of software routers running in general purpose (PC) hardware, as an alternative to the traditional special purpose hardware routers, has grown quickly in the last few years. This is due to the high level of flexibility of this solution: the support for new protocols and network architectures and services, in fact, is easily obtained by re-programming the router itself. On the other end, the low cost of PCs and the continuous progress in their performance are making the design of a software packet switch, based on standard hardware platform, more and more attractive. In recent years, several proposals emerged, and a very promising architecture is the Click Modular Router, which is not only easily extensible, but also very effective: on a multiprocessor hardware, its maximum loss-free forwarding rate for IP routing reaches 1,250,000 64-byte packets per second. It allows the simple design and rapid implementation of new services: we first used it for the development of flow-based classifiers of real time traffic. Then, we introduced a traffic measurement mechanism and exploited several dynamical allocation schemes of the inter-domain link bandwidth, to maximize the bandwidth usage efficiency. Finally, we implemented a congestion and admission control technique, to preserve the bandwidth usage efficiency and avoid network congestion. These new functions were evaluated in terms of latency, jitter and bandwidth usage efficiency.

Keywords. Open router, Quality of Service, Real-time traffic.

Contents

Chapter I. A General Overview

The quality of a communication service is usually related to the users' expectation that a functionality will be furnished when necessary and in a reliable way. At the same time, under a technical point of view, the quality of a multimedia session is usually measured in terms of delay, losses and errors occurring in the data transfer. For example, a common application such as VoIP is not so helpful if affected by high delay (which makes a conversation rather arduous), by losses (which discontinue the interaction) or by errors (which disturb the voice quality). Fundamentally, the quality of a network service is strictly related to the traffic amount inside the network, which depends on the load and profile of the traffic traversing the infrastructure. A common way to obtain a requested quality for an application is given by network over-dimension. This solution, even if simple, does not take into account its economical cost and waste of resources. For this reason, in the last few years an intense research activity has focused on models for service differentiation and techniques for resource management in IP networks. A variety of architectures have been proposed: ATM, RSVP, the Integrated Services and the Differentiated Services represent different solutions concerning similar necessities.

Since Internet traffic is commonly the result of a wide aggregate of different communications, classifying and distinguishing this traffic is the first step in order to offer different performance for the different applications' needs. A possible classification of the Internet traffic distinguishes it in "elastic traffic flows" and "streaming traffic flows". For our purposes, a flow is identified as a unidirectional sequence of packets close in time and having a common identifier (for instance, a tuple like source and destination address and a port number). Elastic traffic is based on closed-loop control, as commonly happens during a TCP connection, and is named "elastic" since is able to react to changing network load: each elastic flow essentially varies its transmitting rate depending on packet loss probability. Streaming traffic instead is normally emitted from multimedia applications, which generate packets at a rate that should be conserved traversing the network, by avoiding losses and delay dispersion. A possible solution to meet these requirements could rely on fixing a deterministic delay limit for the incoming flows. The main drawback of this solution is that it usually brings to bandwidth over-provisioning, since the limit should be dimensioned for a worst-case situation. Thus, if link usage efficiency is considered a key factor, dynamic bandwidth allocation mechanisms should be introduced at the edge of the network, to preserve delay bounds on the fly. At the same time, efficiency can be maximized if both the two traffic categories are considered together: for instance, if streaming flows are high prioritized, they normally traverse the network with low delay and losses; elastic flows, instead, can take benefit from the bandwidth that is temporarily not utilized from multimedia flows. The feasible dynamic bandwidth allocation schemes are many; in the following chapters, we have experienced two of them: the first uses a threshold-based approach and it was then

compared with a new algorithm, which uses the logarithmic function for the same purpose. Both the systems have then been provided with an optional congestion avoidance control, which provides expedited bandwidth increases, to promptly solve short-term bandwidth increases. The congestion control mechanism can also be designed to allow flow deletion, when the bandwidth update cannot be performed. This process can be driven using different policies (e.g. we can cancel the largest flow, the smallest, the oldest, etc., or choose one randomly).

Another important mandatory capability for a reliable network functioning is the restriction of the incoming flows of traffic: this approach, commonly named "admission control", should be implemented at the edges of the network to preserve performance, in case bandwidth request exceeds link throughput. Obviously, this can be accomplished only if each individual flow is identified, classified and inserted in a proper list of admitted flows and if, at the same time, bandwidth usage measurement is performed. When the incoming packets of a new flow reach the edge router of a network, this should be able to control if they satisfy the admission criteria or discard them. At the same time, if an application does not generate packets for a long amount of time, its corresponding flow identifier should be properly removed from the list. Thus, this approach is helpful to preserve the bandwidth usage efficiency and avoid network congestion. It can be easily realized that the admission control service could be applied smoothly, assigning different policies to different service classes (e.g. regular flows could be rejected before than premium flows). This mechanism assures that all the admitted flows take advantage of adequate quality guarantees. Additionally, the kind of service provided to each individual user from the network administrator should be regularly arranged by a contract. This accordance, usually named Service Level Agreement (SLA), fixes the maximum amount of the network resources that such a user can rely on. This mechanism is not only interesting under an economical perspective, but is also helpful for the network link dimensioning.

A few years ago, the introduction of a hierarchical two-tier architecture into the Differentiated Services model, which is interesting for scalability aspects, has been proposed for end-to-end QoS support. This model requires some functions to be implemented in the edge routers, such as packet assignment to service classes on the basis of explicit signaling or classification mechanisms, and class management in the core routers. The Differentiated Services architecture considered in the following chapters supports a scalable solution to QoS in IP networks being it based on few fundamental concepts and components: the identification of the packet QoS class through a code point and the differentiated treatment of that packet within a DiffServ node. More recently, an intense research activity on models for service differentiation has been developed and lately it has focused on the techniques for resource management. Among these, the scheme based on the Bandwidth Broker concept can be considered as suitable to cope with the Differentiated Services model. Although this approach is fairly centralized, it can be made scalable through hierarchical organization of its functions. In fact, this scheme separates the intra- and inter-domain resource management problems apart: while the first one is delegated to the bandwidth broker inside the domain, the latter is provided by the agreements of the bandwidth brokers belonging to adjacent domains. At the same time, the edge router must be able to solve the classes of traffic designated for a privileged treatment, and to manage efficiently the resources arranged by its bandwidth broker.

Concerning this, leading routers' producers nowadays offer proprietary solutions, which are often complex or impossible to improve and adapt. Instead, there is a growing attention of the international scientific and industrial community for open and standard platforms supporting free software. To this end, the Click Modular Router, realized at the MIT a few years ago, is a flexible and modular framework for the simple design and the rapid implementation of new services. It offers excellent performance (in particular if executed with Symmetric Multi Processing architectures), quite impressive when compared with most of other existing software routers (the Linux kernel, BSD, Scout, CrossBow, etc.). It is easily extensible: for this main reason, we have preferred it to other similar frameworks. We first used it for the development of flow-based protocol and statistical classifiers of real time traffic; then, we introduced the traffic measurement and exploited the dynamic allocation of the inter-domain link bandwidth; finally, we implemented the congestion and admission control mechanisms previously depicted. These new functions were evaluated in terms of latency, efficiency and number of requests for the bandwidth broker.

Chapter II. Performance Measurements of PC-based Routers

1 Introduction

The interest of the scientific and commercial telecommunications community for the use of software routers running in general purpose (PC) hardware, as an alternative to the traditional special purpose hardware routers, is risen quickly in the last few years. Open routing approaches have been developed with the aim to use standard hardware platforms to support free and open software [1]. In recent years, several proposals have been made, most of which are suitable for the Linux environment and assume standard PCs as hardware platforms. This is due to the high level of flexibility and extensibility of this solution: the support for new protocols and network architectures and services, in fact, is easily obtained by re-programming the router itself. In addition, the diffusion of multiprocessor systems due to the progress in the semiconductor technologies allows software routers to obtain high performance if supported by multiprocessor PC hardware. Of course, in order to achieve a good use of the potentiality offered by multiprocessor architectures, the distribution of the tasks among the CPUs, and the parallel execution of the different operations, requires to be performed with some care. Moreover, different CPU scheduling techniques, that is, different approaches in the assignment of the tasks to the different CPUs, affect the router performance. This chapter addresses the problem of performance evaluation of routers implemented using open source software running on a general purpose PC. It evidences the need of the availability of flexible platforms to easily and timely implement new router functionalities in relation to the service scenario evolution. At the same time, the chapter presents measurements performed on a PC router implementation to evidence the performance bottlenecks of present-day PCs when used as routers and the main effects of the operating environment that influence packet forwarding. In fact, even if PC hardware limitations are well known in general, at the time of these writing limited contributions have been given to the knowledge of performance and bottlenecks of a PC when used as a router. The chapter is organized as follows: in section 2 basic concepts related to Personal Computers architectures are withdrawn, together with a very brief description of the MIT Click Modular Router; section 3 describes the performance tests conducted using a typical Desktop PC platform and analyzes in dept the bottlenecks of this kind of architecture; section 4 shows how is possible to use a multiprocessor PC platform combined with the SMP Click Modular Router to build very fast software routers.

2 PC Router Architecture

A software router can be characterized as a general purpose PC, governed by an application able to move data (packets) among different devices (the network interface cards). The hardware architecture of a modern PC includes at least the following elements, outlined in figure 1: one or several CPUs; a main memory bank; a shared I/O bus which interconnects various devices; a bridge chipset implementing the connection among CPUs, memory and I/O bus. A particular example of an I/O bus is the Peripheral Component Interconnect (PCI) bus. In the past years the PCI bus specifications and performance remained relatively stable over different PC generations and this situation should persist for further years. On the contrary, impressive improvements have concerned the CPU-Memory subsystem.

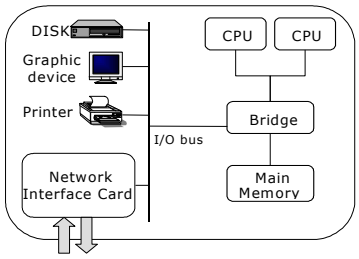

Fig. 1. A common PC architecture

The routing process generally involves all the subsystems summarized in Fig.1 and can be divided into three distinct steps. At first, the datagram is transferred from the input network interface card to memory using the I/O bus; in a second phase, the CPU reads the necessary data from memory and uses them to take the routing decision; then, the packet is sent to the output NIC, accessing again the I/O bus. Thus, the I/O bus subsystem, besides being the slowest path traversed by data, is also utilized twice. In the near future, if hardware evolution will follow the actual trends, the PCI bus will probably represent the bottleneck of a PC system. For this reason, if in the past many works had converged about the issue of the fair sharing of the CPU and memory resources [3] [4], more recently the management of the I/O bus resources has become a more crucial problem [5]. Furthermore, being it a single common communication line at the disposal of several devices (this architecture is named "Multiple Bus Master") poses some problems. In fact, access conflicts must be avoided, and thus an arbiter should manage the admission of the devices to the shared bus; moreover, the policy adopted by the arbiter should be carefully chosen to obtain a fair share of the bus resource, but the PCI specifications do not establish any constraint to obtain this result. A round-robin scheduling technique is usually adopted for the admission of the devices to the PCI bus. This choice is directed to minimize the access latency: the bus arbitration protocol is access-based instead of being time-slotted. The devices plugged into the PCI bus can communicate with software running on the host processor(s) using two distinct techniques: Programmed I/O and Direct Memory Access (DMA). The second mechanism uses shared data structures that can be manipulated by both

the processor and devices: its main advantage is evident with large amounts of data to transfer to or from main memory, since it does not lock the CPU during the relocation. Most modern NICs support DMA as the favorite method for moving packets and offer an integrated DMA controller to manage all DMA associated operations. These interfaces can request the bus arbiter to access the I/O bus, become the bus master and move the data from/to memory independently from the CPU activity.

2.1 The Click Modular Router

The functions of packet forwarding can be implemented by the operating system kernel itself, or by other software layers, like in the case of the Click Modular Router, a Linux-based software framework developed at the MIT [2]. It permits the design of PC routers or other packet processors offering an extensive library of simple modular components. The principal advantages of such a modular system are flexibility and extensibility: the designer can easily create various services simply connecting the basic modules, called elements. An IETF RFC-1812 router was proved quite simple to realize, since only 16 click elements must be employed, as described in [2].

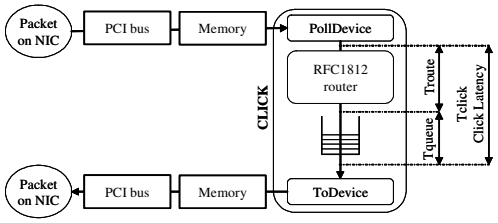

Fig. 2. Hardware/software elements traversed by a packet during forwarding

Even if performance is not the main goal of the project, Click can be dynamically linked as a module to the Linux kernel, taking advantage of the kernel-space execution priority and substituting the standard kernel networking functions. Moreover, some network cards can be managed with polling drivers, instead of using the more expensive interrupt-based technique [3], eliminating the livelock problem and improving the forwarding rate. The standard library also furnishes an effective support for measurements: the *SetCycleCount* and *CycleCountAccum* modules can be used to collect the average number of CPU-cycles required for a packet to traverse the elements comprised between them. Inserted at the sections outlined in figure 2, they permitted the latency evaluations (entirely done inside the Click configuration) presented in the rest of the chapter. This measurement support, which is based on the Intel Pentium RDTSC register, offers an excellent timing accuracy (using a 1.6 GHz CPU the resolution is 0.625 ns). In addition, the new functionalities we added to these elements permit a good insight into the PC based routers' functioning limitations, which cannot be so easily evaluated within the Linux kernel networking structures or in any way appreciated if measuring performance outside the router.

3 Performance Evaluation of a PCI Desktop Platform

In order to execute performance tests, the trial configuration outlined in Figure 3 was created to simulate functions of a real network environment. The test-bed consists of four PC-based systems, plugged in a Gigabit Ethernet layer2-switch. The edge router is equipped with a 1.6 GHz Pentium IV CPU and implements the RFC1812-compliant router, with a basic FIFO output queuing scheme. The other three PCs have got a 1Ghz–Pentium III processor. Every PC is equipped with the Intel PRO1000 network cards. On our router, two of them were plugged on the 32-bit/33 MHz PCI bus, even if they would be ready for the more advanced 64-bit/133 MHz PCI-X buses. The choice of this NIC permits to use the polling-based driver already developed by the MIT for these adapters. Like most of recent network cards, it disposes of an on-board FIFO buffer to store datagrams received from the wire or waiting to be transmitted. In addition, it contains dedicated registers maintaining statistics about its internal state. All the PCs mount the Linux kernel 2.4.18 and Click release 1.3. Two PCs work as traffic generators for injecting two distinct flows of UDP traffic at a constant rate into the input port of the router. Another PC collects the packets coming from the router.

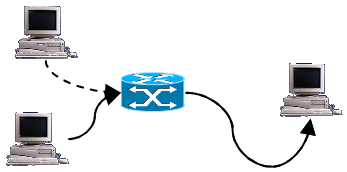

Fig. 3. The testbed layout

3.1 Performance evaluation with no packet losses

The results presented in the following are the average of 20 successive 120-seconds runs, as proposed in the RFC1242-2544, at the maximum forwarding rate and avoiding packet losses. Figure 4 shows the throughput and the bus occupancy of the router as a function of the packet length. Only with long datagrams, the maximum theoretical bus capacity of 1.056 Gbit/s is approached (the bus is traversed twice for each packet). Another important figure of interest for a router is packet delay (see Table 1). As well, the latency is not dependent on packet length: this behavior is expected, since only data (i.e. packet) pointers are passed through the Click modules, avoiding time-expensive data-touches.

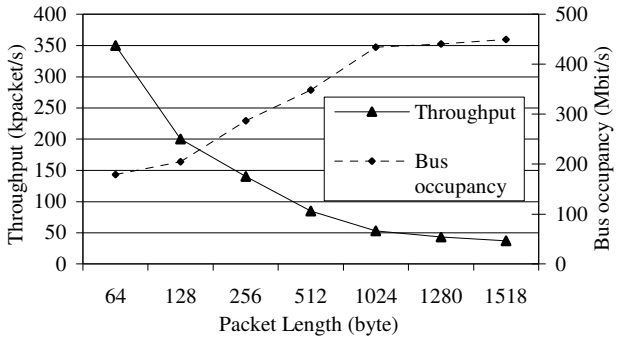

Fig. 4. Loss-free maximum forwarding rate and corresponding bus occupancy vs. packet length for a Click-based RFC-1812 router (20 120-seconds runs)

Table 1. Average packet latency in microseconds for a RFC1812-based Click router for different values of packet rate in kpacket/s (columns) and length in byte (rows), evaluated at the highest loss-free rates

	350	*200*	*140*	*85*	*53*	*43*	*37*
64	8,43	4,63	3,83	3,23	2,49	2,43	2,42
128		4,68	3,82	3,20	2,46	2,40	2,39
256			3,78	3,14	2,43	2,34	2,35
512				2,99	2,37	2,32	2,30
1024					2,50	2,39	2,42
1280						2,48	2,45
1518							2,49

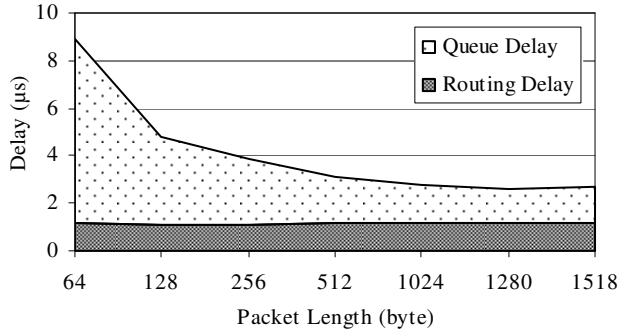

Fig. 5. Average delay vs. packet lenght for a Click-based RFC-1812 router (20 120-seconds runs)

Evaluations reported in [6] and summarized in Fig. 5 show how the variable contribution to the global delay reported in Table 1 is due to the output queue term,

while the routing delay alone requires constantly about 1.2 μs. Table 1 summarizes the Click latency for various packet sizes and rates. It makes clear the strict relation between packet delay and rate.

3.2 Forwarding delay analysis

More thorough analyses were done releasing that restriction with the aim to investigate system bottlenecks. To this end, additional tests were first performed using 64-byte short packets and increasing the input rate with respect to values of Table 1. The forwarding rate no longer increases and the packet latency remains the same, while the input interface reports "FIFO Errors" events through its internal registers. This denotes that the NIC-to-memory DMA transfer of packets is failing, that is the CPU-Memory subsystem is not capable of releasing DMA descriptors to the NIC fast enough to handle all the incoming packets. Results reported in [7], where a much slower CPU was utilized to achieve the same throughput, lead to conclude that the memory subsystem, rather than the CPU, represents the slowing factor. When using 1518-byte packets, an increase of the input rate produces very different consequences. In order to perform in depth investigation of different contributions to the routing delay, a timing diagram of the forwarding activities inside the router is presented in figure 6. The arrow at the left side of the diagram represents a packet stored on the NIC FIFO buffer and ready to be transferred to memory. The time spent for a packet forward can be split in three stages. During the first stage, the datagram is moved from the NIC towards memory using the I/O bus: the NIC fetches a free receive DMA descriptor (this takes a time T_{poll_dma}) and then the NIC-to-memory transaction on the bus occurs (T_{bus}). At the second stage, Click takes the routing decision: it polls the packet from memory by the PollDevice module (T_{poll}), elaborates it and queue it (T_{route}); then, ToDevice pulls it from the queue towards the transmit DMA buffer (T_{pull}). Finally, the packet is sent (accessing the I/O bus again) to the output NIC: this fetches a transmit DMA descriptor (T_{poll_dma}) and moves the packet from memory to its own internal FIFO (T_{bus}). At the end, before the next packet arrival, the routing process remains inactive for a time T_{idle}.

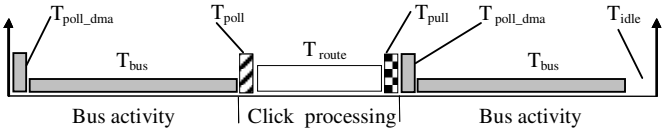

Fig. 6. Timing diagram of packet forwarding. The striped square locates the PollDevice activity, while the dotted one when the ToDevice element pulls packets from the queue

Using 1518-byte packets, the time requested for a complete forward cycle ($T_{forward}$) was first estimated, summing all the preceding contributions:

$$T_{forward} = T_{route} + 2T_{bus} + T_{poll} + T_{pull} + 2T_{poll\_dma} \approx 25020ns \ . \quad (1)$$

This corresponds to a theoretical maximum forwarding rate of 39968 packet/s, with no more margin in time (T_{idle} becomes zero) before the next packet arrival. The time required for all the bus accesses can be evaluated as:

$$2T_{bus} + 2T_{poll\_dma} \approx (23120 + 240)ns = 23360ns \ , \qquad (2)$$

which is the 93.3% of the global cycle duration. Experimental evidence confirms this estimation: packet losses suddenly grow at 39800 packet/s. This rate is also an edge for the average queue latency as shown in figure 7. This reveals that the router is congested: the input interface persists accepting the incoming packets, but these are not as much promptly extracted from the Click queue toward the output interface. This unbalanced functioning was deeper examined using the Click's accounting mechanisms: the ToDevice module, that pulls packets from the queue and puts them in the transmit DMA buffer, was frequently idle. ToDevice normally remains idle when the DMA buffer is full [8], thus the only possible cause of this kind of performance is the PCI bus overload (a memory overload could also be possible but should affect the receiving stage, too: however, such event was not encountered). To further investigate this aspect, the *CycleCountAccum* module was customized to collect each per-packet delay within an internal vector, instead of exporting its usual averaged estimates. This was possible since *SetCycleCount* appends the CPU internal cycle count obtained from the RDTSC register in a special packet annotation field. In combination with *CycleCountAccum*, it allows to measure how many cycles it takes for each packet to pass from one section to another.

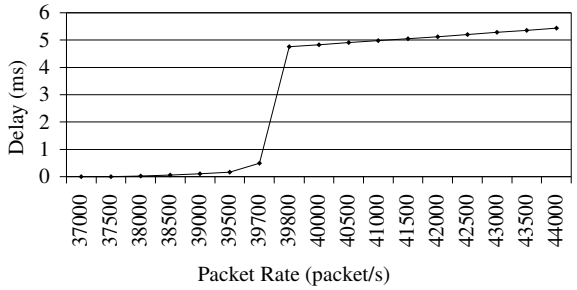

Fig. 7. Average output queue delay (T_{queue}) vs. packet rate (1518-byte packets)

Figure 8 sketches an example of the delay distribution obtained using 1518-byte packets and an input flow of 37000 packet/s. Under the same boundary conditions, the global average latency reported in Table 1 was 2.49 μs (this comprises a routing delay of 1.2 μs). Thus, surprisingly, the residual average queue latency (1.29 μs) is the result of a very scattered delay distribution.

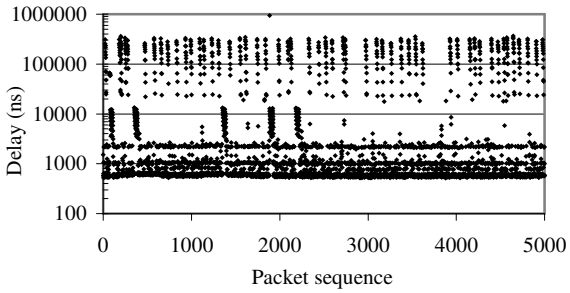

Fig. 8. Packet delay (T_{queue}) inside the output queue (1518-byte packets at a rate of 37000 packet/s)

Moreover, it's possible to observe how some delay states prevail, forming three dense horizontal lines in the graph, located approximately at 600, 1200, 2400 ns. High delay spikes (vertical lines) up to 300μs are also evident and will be discussed in Subsect. 3.4. Further increases of the flow rate gradually drag the average packet latency up to higher values and when the threshold rate of 39800 packet/s is reached, the packet latency is consistently positioned around a central value of 4.8 ms, as outlined in figure 9 and coherently with the graph in figure 7.

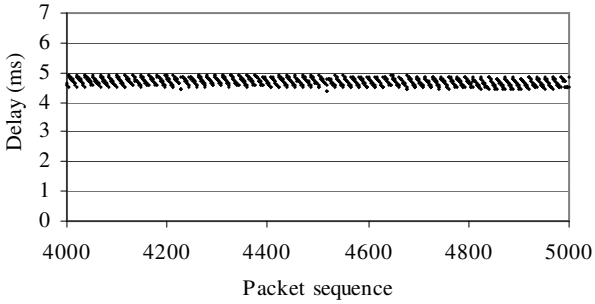

Fig. 9. Packet delay (T_{queue}) inside the output queue (1518-byte packets at a rate of 39800 packet/s)

3.3 Burst effects

A fraction of an horizontal delay line extracted from figure 8 is enlarged in figure 10, showing a surprising behavior: packets flow out of the Click queue with decreasing delays, instead of increasing ones, as it would be normally expected. An additional analysis was necessary to understand why this happens: the timing diagram in figure 11 shows the activities required inside the router to forward a two-packet burst, already stored inside the NIC internal FIFO. Let us call the two packets A and B respectively. At first, the NIC retrieves a couple of free receive DMA descriptor; then,

A and B are moved through the PCI bus to be placed in the receive DMA buffer. Next, the PollDevice element polls both of them (it can manage bursts up to 8 packets long) and pushes them inside the Click configuration. At this point, they are routed and stored in the output queue. From here, ToDevice pulls them (it can manage bursts up to 16 packets long) to the transmit DMA buffer, to be sent to the output NIC via the PCI bus.

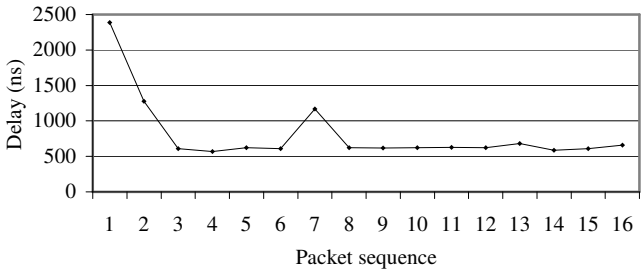

Fig. 10. Magnification of the packet delay graph reported in figure 8

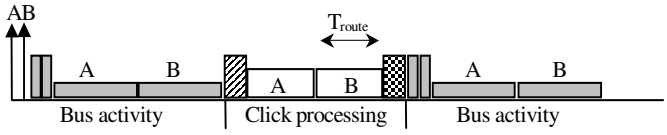

Fig. 11. Timing diagram of a two-packet burst forwarding

Since ToDevice pulls both A and B as a single burst, they respectively collect the delays in the output queue expressed by equation (3).

$$T_{queue\_A} = T_{route} \approx 1200ns, T_{queue\_B} \approx 600ns \ . \quad (3)$$

In fact, A must wait for B to be routed (this takes a time T_{route}) before being pulled from the queue. Instead, T_{queue_B} should in theory be zero, but a lower bound for packet delay is always experimentally measured for any datagram traversing the output queue. Equation (3) can be generalized as follows:

$$\begin{cases} T_{queue\_i} \approx (N-i)\cdot T_{route}, i < N \\ T_{queue\_i} \approx 600\,ns, i = N \end{cases} \quad (4)$$

with i = 1,…, N , being N the burst length (i.e. the number of packets admitted by PollDevice). This model is consistent with the experimental graph showed in Fig. 8 and justifies delay values up to 8400ns (PollDevice can poll up to 8 packets in each burst). Thus, the horizontal delay lines depicted in figure 8 are due to packet bursts stored in the input NIC FIFO.

3.4 Effects of bus sharing

Previous discussions do not explain either the presence of high delay spikes in figure 8 or packet bursts formation. When the routing task does not use the bus, other devices can obtain to move their own data. Two time windows permit this to happen: before a new packet arrival (T_{idle}) or during the Click processing stage. In the first case, the bus is kept locked and the input NIC can not transfer any packet to memory till the extraneous bus activity ends up: the whole routing process is shifted in time, while the input NIC stores incoming packets in its FIFO, forming new packets bursts. The magnification of a delay spike is showed in figure 12, as an example of what happens if the extraneous bus activity arises during the Click stage. This performance is justified in figure 13: when Click has finished its processing, the packet cannot be extracted from the output queue, since the bus is locked and ToDevice cannot start its pulling stage. Thus, the datagram hangs inside the output queue for the whole duration of the noise (T_{noise}), acquiring a delay that has no upper bound.

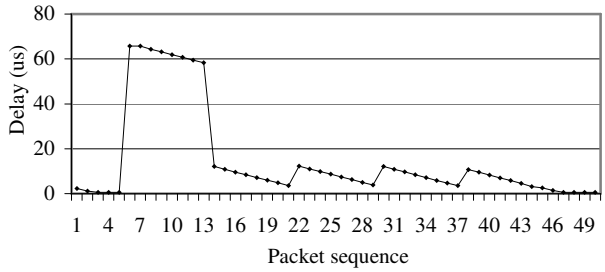

Fig. 12. Enlargement of a packet delay spike extracted from figure 8

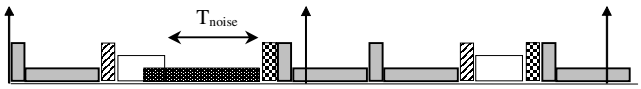

Fig. 13. Effects of an extraneous bus access (the black square) during the forwarding process

This kind of bus activity can form packet bursts, too: as the bus is kept locked, the input NIC receives new packets. It can be noticed that the collected extra latency does not apparently affect the subsequent routing cycles (as the packet latency is measured at the queue level): the queue delay rapidly decreases, since the pull stage is again straightly following the routing phase. Detailed bus monitoring would allow a more precise knowledge of bus-sharing effects but unfortunately the Linux kernel does not actually offer adequate tools yet.

4 Performance Evaluation of a Multiprocessor PCI-X Platform

The previous section clearly showed how the main performance problems of a "desktop" PC-based router are represented by both the CPU computing power (when short packets are injected at high rates) and the standard PCI bus throughput. To overcome these limits, a different test-bed was set up. In particular, to increment the computing power a dual processor platform was chosen, while the PCI-X bus was selected as a shared structure to plug the network interfaces. Multiprocessor systems can achieve high processing power thanks to a high level of parallelism between a certain number of CPUs. Given the great availability of multiprocessor systems, SMP Click [9] was developed with the aim of improving the Click software routers performance. Obviously, to really take advantage of a multiprocessor system, making a good use of the parallelism, great attention to the CPU task assignment techniques is necessary.

4.1 SMP Click Schedulable Elements

The main aim of SMP Click is to run Click configurations on multiprocessor PC architectures, through the parallel execution of packet processing tasks among different CPUs. SMP Click creates a different thread for each CPU available in the system. A CPU thread can schedule only a few elements, the so-called schedulable elements, of a Click configuration. All the schedulable elements in the configuration are placed in a task array as tasks. In addition, each CPU has its private work list. Each task is then assigned to the work list of one of the CPUs, in order to be processed by this CPU in a round robin order. Click schedulable elements are *PollDevice*, *ToDevice* and all the *PullToPush* elements. When a *PollDevice* is scheduled by a CPU, it examines the network device for new incoming packets, then pushes them to the downstream element: the whole *push* path, that is, the path starting from the *PollDevice*, and ending with a *Queue* element, is demanded to the CPU which schedules the *PollDevice*. In the same way, the path starting with a *Queue* element and ending with a *ToDevice*, called *pull* path, is demanded to the CPU which schedules the *ToDevice* element, and therefore starts the execution of a *pull* processing.

4.2 CPU Scheduling Management Techniques

SMP Click supports static and adaptive CPU scheduling. When the static CPU scheduling is used, all the schedulable elements are statically distributed between the work lists; in this way each of them is always processed by the same CPU. If adaptive CPU scheduling is used, on the other hand, Click maintains the average processing cost for each schedulable element, in terms of consumed CPU cycles, and the work list of each CPU is periodically updated according to these costs, in order to keep the load among the different CPUs balanced. The work list updating interval is a parameter which can be set in the configuration file. Then, a global scheduler runs through the task array again, by assigning each element to the CPU work list with the

lowest load, in order to achieve a good load balancing. It should be noticed that the adaptive scheduling algorithm takes only into account the load balancing among the CPUs, in order to avoid CPUs idle times: no attention is paid to other important aspects, like cache misses and cache coherency maintenance mechanisms [10], [11], which negatively affect the performance of multiprocessor systems. When packets, buffers or other data structures are handled by more than a single CPU, multiple CPU private caches can have a copy of a given memory location. As soon as one CPU modifies shared data in its private cache, the copies in the caches of the other CPUs must be invalidated. The mechanisms, which have to be used in order to ensure the consistence of the caches content (cache coherency protocols), introduce an overhead which degrades router performance. Instead, if a given data structure is always managed by the same CPU, this structure is present in its private cache only. This enforces cache affinity, by reducing the processing time, so increasing router performance.
By taking into account these considerations, a few simple rules can be followed for a good use of parallelism. For example, every mutable data structure, such as a queue, should be managed by as few CPUs as possible. In addition, any single packet, or packets belonging to the same flow, should be processed by a single CPU during its forwarding path. As a consequence of this last suggestion, if we consider that it is likely that a packet coming from a network interface leaves the router from another interface, the *PollDevice* and the *ToDevice* of the same interface should not be scheduled by the same CPU.

4.3 Performance Analysis

The following investigations were conducted to strictly respect the RFC1242-2544 specifications proposed by the IETF Benchmarking Methodology Working Group (BMWG). For instance, throughput is specified as the maximum forwarding rate at which none of the offered frames are dropped, and results are obtained as the average of 20 successive 120-second runs. The testing configuration depicted in Fig. 12 consists of five PC-based systems running the Linux kernel 2.4.18 and Click release 1.3.

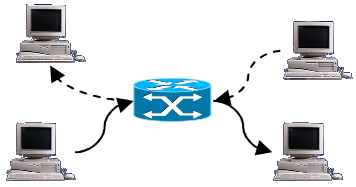

Fig. 12. The server-based test-bed layout

One PC implements a RFC1812-compliant router (with a basic FIFO output queuing scheme), two are packet generators, and two are packet receivers. During every test, each of the two sources injects, by means of the *FastUDPSource* element, a constant-rate balanced flow of 64-byte UDP packets into the input ports of the router, while receivers collect the traffic coming from the router. The edge router is equipped with a dual Intel Xeon 2.80 GHz CPU (with the HyperThreading technology disabled), an

Intel E7501 chipset motherboard with a 533 MHz FSB and four distinct PCI-X slots. The router has four Intel 82544EI Gigabit Ethernet controllers directly connected by copper point-to-point links to the packet generators and receivers. Two of them are plugged to the 64-bit/133 MHz PCI-X bus, while the two others are attached onto the slower 64-bit/100 MHz PCI-X slots. The choice of this NIC allows to use the polling-based driver extensions developed by the MIT for these adapters. Like most of recent network cards, this NIC disposes of an on-board FIFO buffer to store datagrams received from the wire or waiting to be transmitted. In addition, it contains dedicated registers maintaining statistics about its internal state. The MLFFR (Maximum Loss-Free Forwarding Rate) versus input rate for a Click-based router with SMP disabled (the 1-CPU case) is 737,000 64-byte packets per second (Fig. 13). More thorough analyses were done, increasing the input rate further with the aim to investigate system bottlenecks: the forwarding rate no longer increases, while the input interfaces report "FIFO Errors" events through their internal registers (Fig. 14). This denotes that the NIC-to-memory DMA transfer of packets is failing, that is the system is not capable of releasing DMA descriptors to the NIC fast enough to handle all the incoming packets, causing the FIFO overflow. This evidence shows how the main performance problem is the CPU computing limitation. To overcome this limit, SMP Click allows the usage of multiple CPUs (with the adaptive and static scheduling techniques described so far), with automatic or user-defined parallel execution of packet processing tasks to maximize performance, exposing the advantage of multi-processor architectures. In order to illustrate how the hardware/software choice affects the performance, in Table 2 we further considered the MLFFR values obtained with different configurations.

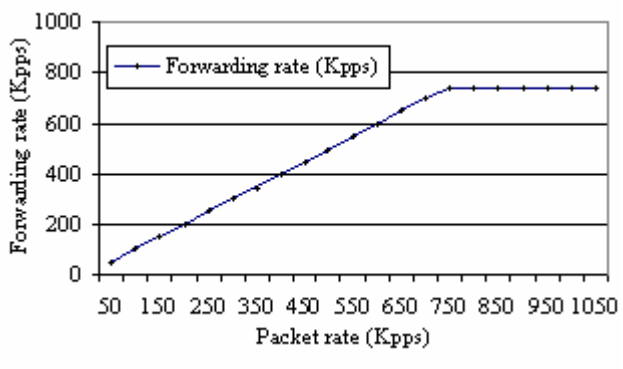

Fig. 13. Forwarding rate vs. input rate for a 1-CPU Click-based RFC-1812 router using 64-byte packets (20 120-second runs)

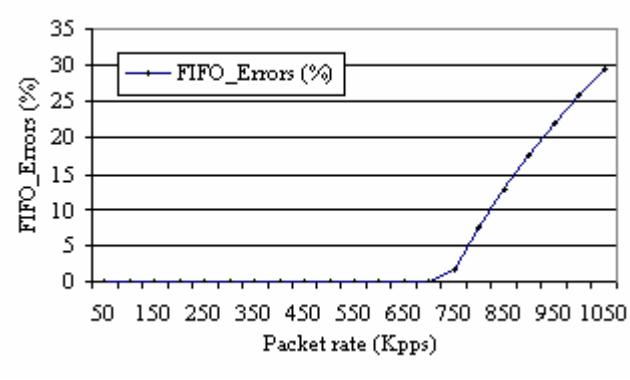

Fig. 14. FIFO errors vs. input rate for a 1-CPU Click-based router using 64-byte packets

Table 2. MLFFR for different router configurations using 64-byte packets

Platform	*MLFFR (packet/s)*
C1	350,000
C2	737,000
C3	750,000
C4	1,250,000

C1 consists of the 1.6 GHz Pentium III platform with 32bit/33MHz bus described in Section 3, while C2-C3-C4 is the dual-Xeon platform described at the beginning of this section, with Click SMP support disabled, and enabled with adaptive scheduling and static scheduling, respectively. Comparison between C1 and C2 shows how the technological improvement of FSB (CPU-memory) and PCI-X buses influences the forwarding rate: C2 CPU is less than twice as fast as C1 CPU, but the MLFFR is more than twice. C3 performs somewhat better than C2, but it seems that in these conditions (i.e. two distinct flows traversing the router) adaptive scheduling algorithm cannot take so much advantage by the presence of an additional CPU . Figures 15 and 15b show how different is the router behavior for C3 in comparison with C2. In fact, despite maximum throughput reaches absolute higher values for C3 (around 1,000 Kpps), packet losses (i.e. FIFO Errors) arise at the NIC level, indicating that all the available computing power is not efficiently utilized.
Since the adaptive scheduling does not take into account the cost of cache misses, but only tries to reach a balanced load among the available processors, if two schedulable elements handling the same packet are assigned onto different CPUs, the cost of processing increases, leading to unsatisfactory performance.

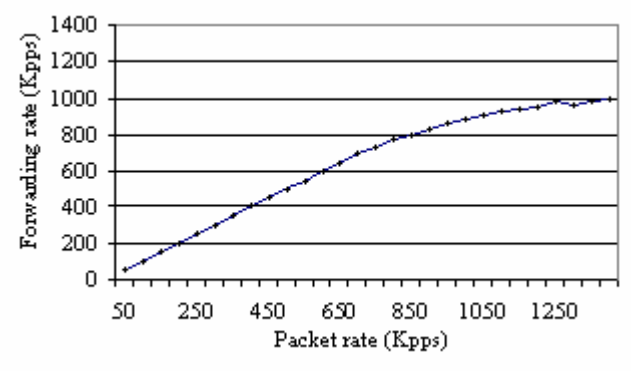

Fig. 15. Forwarding rate vs. input rate for a 2-CPU Click router with adaptive scheduling, using 64-byte packets (20 120-second runs)

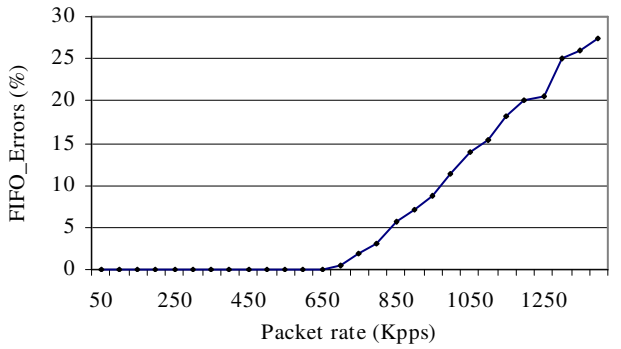

Fig. 15b. FIFO Errors rate vs. input rate for a 2-CPU Click router with adaptive scheduling, using 64-byte packets (20 120-second runs)

On the contrary, the static scheduling technique offers the network manager the possibility to improve router behavior starting from the knowledge of traffic statistics. For instance, in our experimental layout, which fundamentally emulates a border router, most of the traffic is exchanged between the inner and outer networks (i.e. different network interfaces). In these circumstances, the *PollDevice* and *ToDevice* elements referring to the same interface rarely handle the same packets and thus can be allocated onto different CPUs, while elements mostly involved in packet forwarding are assigned onto the same CPUs, preventing the expensive cache misses. With these guidelines in mind, Fig. 16 depicts how traffic flows are assigned onto the available processors.

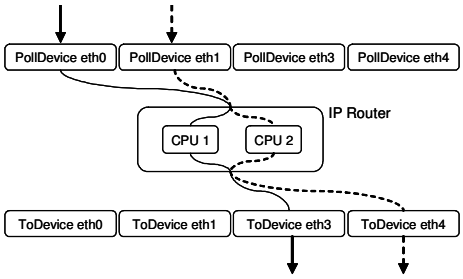

Fig. 16. Assignment of traffic flows to the CPUs, evidencing the parallel processing of forwarding activities

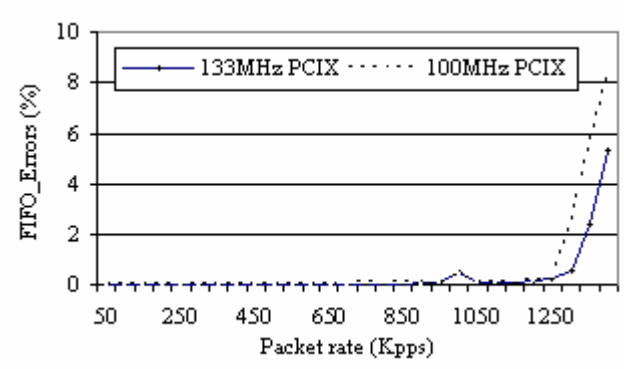

Fig. 17. FIFO errors events for the two distinct network interfaces vs. input rate for a SMP Click router using the "static scheduling" technique

This makes evident the parallel processing of the incoming packets, while numerical performance results are reported in Table 2 (row C4), showing how this technique can take full advantage of the available power computing. A further point to mention is how differences between the two PCI-X buses affect the performance. Figure 17 highlights how the main throughput limitation (using the static scheduling configuration) is due to the card connected onto the 100MHz/PCI-X bus, where "FIFO errors" events appear first, persuading us that a hardware platform equipped with four identical 64-bit/133MHz PCI-X slots should exploit all the available computing power, and could probably reach a maximum loss-free forwarding rate of 1,350,000 64-byte packet/s. Figure 18 resumes the throughput and bus occupancy of the router as a function of the packet length: it can be noticed how the limit NIC throughput (1 Gbit/s) is reached (the bus is traversed twice for each packet) when the packet length is over 256 byte.

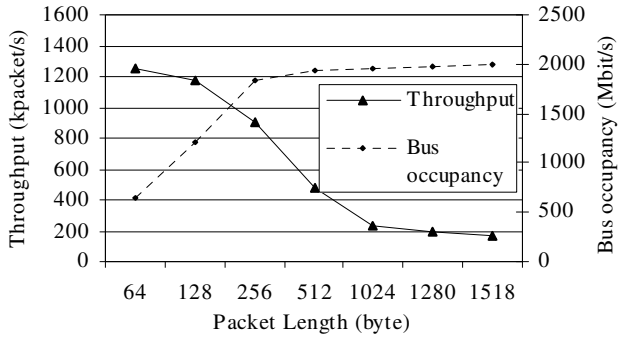

Fig. 18. Maximum loss-free forwarding rate and corresponding bus occupancy vs. packet length for a 2-CPU Click router with static scheduling

5 Conclusions

In this chapter measurements performed on a PC-based router have been presented and discussed. Investigation of system bottlenecks lead to the conclusion that the PCI bus represents the major cause of throughput limitation. The effects on this performance figure of different system parameters and bus sharing occurrences have been thoroughly evaluated also in terms of forwarding delay. The main conclusion is that the time-sharing performed by the PC bus does not guarantee delay fairness and leads to unbounded forwarding delays. The bus arbiter logics is in fact implemented having in mind the usage of a PC as an end user system, while, when using it as a network node, different arbiter programming should be made to assure fair sharing of the bus resource.
Then, it has been demonstrated how the usage of modern PCIX buses combined with the insertion of software routers able to parallelize the tasks among different processors allows to overcome these limits and permits to take full advantage of the potentiality offered by multiprocessor architectures.

Chapter References

1. Keshav S., Sharma R.: Issues and trends in router design. IEEE Communication Magazine, vol.36, n.5, pp.144-151, May 1998
2. E.Kohler, R.Morris, B.Chen, J.Jannotti, M.F.Kaashoek: The Click modular router. ACM Trans. Computer Systems Vol. 18, August 2000
3. J.C.Mogul, K.K.Ramakrishnan: Eliminating receive livelock in an interrupt-driven kernel. ACM Trans. Computer Systems Volume 15, August 1997
4. A. Bavier, S.Karlin, L.Peterson, X. Qie: Scheduling Computations on a Software-Based Router. ACM SIGMETRICS Volume 29, June 2001

5. Oscar- Ivàn Lepe-Aldama, Jorge Garcia: I/O Bus Usage Control in PC-based software routers. IFIP 2002, LNCS Volume 2345 pp.1135-1140, January 2002
6. G.Calarco, C.Raffaelli: An open modular router with QoS capabilities. HSNMC 2003, Lecture Notes in Computer Science, Volume 2720 pp.146-155, July 2003
7. E.Kohler, R.Morris, B.Chen: Programming language optimizations for modular router configurations. ACM SIGPLAN Notices, Volume 37, October 2002
8. E.Kohler, the ICSI Center for Internet Research, private communication, Available: https://amsterdam.lcs.mit.edu/pipermail/click/ , July 2003
9. B. Chen, R. Morris: Flexible Control of Parallelism in a Multiprocessor PC Router. Proceedings of the 2001 USENIX Annual Technical Conference (USENIX '01), June 2001, Boston, Massachusetts
10. J. Archibald, J.L. Baer: Cache Coherence Protocols: Evaluation Using a Multiprocessor Simulation Model. ACM Trans. Computer Systems Vol. 4, No. 4, November 1986
11. M. S. Papamarcos, J. H. Patel: A Low-Overhead Coherence Solution for Multiprocessors with Private Cache Memories. The 11th Intl. Symposium on Computer Architecture, pp. 348-354, June 1984

Chapter III. Introducing Quality of Service Capabilities within Open Routers

1 Introduction

Today users ask forever increasing bandwidth and especially require services with quality of service guarantees. Being the network built on a set of different sub-networks, service access control is necessary through procedures coordinated among different domains and quality of service requires to be managed during information. To this end new routing and queuing techniques must be investigated and new functionalities to manage different application flows must be introduced within the network [1], [2]. As regards the international scientific community, the IETF has defined models for quality of service management in the Internet and in particular the Differentiated Services model, which is interesting for scalability aspects. This model requires some functions to be implemented in the edge routers, such as packet assignment to service classes on the basis of explicit signalling or classification mechanisms, and class management in the core routers. Solutions for Differentiated Services implementation are available by main router manufacturers as proprietary solutions that are difficult to modify and optimize. Recently, open routing approaches have been developed with the aim to use standard hardware platforms to support free and open software [1].

The main aim of this new router design strategy is the definition of flexible and modular design environment and tools that allow fast router design and modification in order to meet user and context needs. The focus is on the edge router where users are required to register as willing to generate real time traffic: this information is stored at the edge router as Service Level Agreements (SLA) and then used for on line authentication. The main target of the Quality of service (QoS) function is to recognize real time flows without explicit user signalling on the basis of the protocol used or, as an alternative, on the basis of statistical analysis of user traffic. In this chapter a new approach to end-to-end QoS is proposed to allow QoS unaware users to access network QoS capabilities in a “plug and play” fashion. The basic idea behind it is a DSCP marking of the traffic based on a content-oriented micro flow classification. The flow classification is made by the edge routers by looking at the traffic aggregate generated within the stub network. The proposed classification process is developed for real time traffic and considers both protocol and statistical analysis of stub network traffic. It is implemented in two distinct open frameworks: the Linux Traffic Control and the Click Modular Router.

The chapter is organized as follows. In section 2 interactive multimedia traffic characteristics are analysed; section 3 introduces the Linux Traffic Control and the real time classifier implementation, describes the corresponding test bed and illustrates the obtained performance, particularly in terms of packet latency and jitter; section 4 focuses on the Click Modular Router, how the real time flow classification

was implemented using this different framework and which performance were obtained.

2 Differentiated Services Model and Interactive Multimedia Traffic

Quality of Service (QoS) is the capability of a network to forward packets in different ways by grouping them into traffic categories called classes. Several different solutions to the QoS problem have been devised: ATM (Asynchronous Transfer Mode), RSVP (Resource ReSerVation Protocol) [3], the Integrated Services [4] and the Differentiated Services [5] architectures are examples of complementary and interoperable approaches addressing different needs. The Differentiated Services architecture, that is considered here, supports a scalable solution to QoS in IP networks being it based on few fundamental concepts and components: the identification of the packet QoS class through a code point and the differentiated treatment of that packet within a DiffServ node as Per Hop Behaviour (PHB). Two main PHBs have been standardised so far: the Expedited Forwarding PHB[6] - for the support of services requiring time guarantees - and the Assured Forwarding PHB[7] - for packet treatment according to three types of drop precedence. PHBs are identified through a 6 bits label, called Differentiated Services Code Point (DSCP) which is placed into the DiffServ Field of the IP header. In order to permit the end-to-end QoS management a hierarchical, two-tier [8] architecture was also proposed. This model defines the inter- and intra-domain resource allocation, needed to achieve the end-to-end QoS support. The approach requires the interaction between the RSVP signalling in the stub networks and the bandwidth brokers within the DiffServ domains [9]. So the user application should be RSVP capable in order to take benefit from traffic differentiation.

2.1 Interactive Multimedia Traffic

From the overall performance point of view interactive multimedia applications are more resistant to packet loss than to high end-to-end delay or jitter when they are transmitted across IP networks. TCP flow control mechanisms assure the correctness of TCP streams but the delay introduced by the retransmission of lost packets creates a bigger damage than the loss itself, if this is reasonable small (i.e. 10%) [10]. Typically, these applications are based on the UDP protocol [11]. The impossibility for current best effort IP networks to assure a better service to real-time applications has lead to the development of special protocols. The main contributors on this direction are the IETF and ITU. To address the previous problems, the IETF Audio-Video Transport and Multiparty Multimedia Session Control working groups have developed RTP/RTCP [12] protocols for the transport of real-time content, and RTSP [13] protocol optimised for multimedia streaming. The benefits introduced by these protocols, together with the need for a common standard base, have given RTP the role of standard protocol for the transport of real-time contents over the Internet. The

RTP protocol is being used by the most common interactive multimedia applications covering both the commercial and the scientific community as shown in table 1. This means that the use of RTP by an application is a sufficient condition to classify the transmitted data as real-time data. The key concept, behind our classification and marking scheme, is to try to recognise RTP as the protocol used above layer four, typically above UDP.

Table 3. Most commonly used interactive multimedia tools

	RTP/RTCP	RTSP	RVSP	Audio/Video
Netmeeting	Yes	No	Yes	AV
Vic	Yes	No	No	V
Rat	Yes	No	No	A
Real Server	Yes (live)	Yes	No	AV

As regards specific delay requirements, the ITU [14] studies the transmission delay constraints for PSTN. Three different classes of delay that satisfy most of the applications have been identified for connections with adequately controlled echo [14]. In order to keep the end-to-end delay as low as possible, it is better to transmit the audio stream as a bigger number of small packets, instead of a smaller number of big packets [15]. There are different reasons that justify this choice. Smaller packets are more unlikely to be fragmented or dropped due to buffer management problems and moreover the loss of one packet introduces a very limited source of noise at the receiver side.

The packet size depends also on audio and protocol aspects. The audio part depends in its turn on the codec frame size and bandwidth while the protocol one is related to the use of different headers (i.e. IP, UDP, RTP). In the case of very limited transmission bandwidth (i.e. analog dial-up modems) the transmission of different audio frame within the same IP packet is required in order to limit the protocol overhead [16]. Given a set of codecs, it is possible to estimate the typical mean packet frequency and size for audio conference applications over IP networks in order to keep the end-to-end delay in the range of few hundreds of milliseconds as specified in ITU-T G.114 to assure acceptable quality to delay-aware users [14]. The lower bound for this packet frequency can be considered about 10 packets per second. This value can be drawn from Table 2 where the values of packet size and frequency are shown for G.723.1 codec, assuming 24 byte audio frames generated every 20 ms.

These considerations about delay, with minor differences, can be applied also for interactive videoconference applications and videophone connections. Both packet size and frequency values are considered in the classification process.

Table 4. Flow rate and packet sizes for G.723.1 codec with 24 byte frames every 20 ms

IP Packets per second	Coding Delay (msec)	Audio payload size (byte)
1	1000	1200
5	200	240
10	100	120

3 Using the Linux Traffic Control to implement a QoS Router

This section describes the design, implementation and testing of a Linux test-bed supporting flow-based classification functions for multiservice traffic. Protocol and statistical analysis of application flows is performed in the edge routers to provide EF treatment to multimedia traffic without any user signalling. These functions take advantage of the Linux Traffic Control utilities and implement SLA management and traffic statistics collection. Sample measurement performed on the test bed shows the effectiveness and feasibility of the proposed solution.

3.1 Test-bed Layout and Configuration

In order to perform quality of service trials, a local test bed was designed to emulate functions of a real network environment that offers real time services with quality of service guarantees. Figure 1 shows the test bed layout, which consists of five Intel i810-board systems connected through a Layer2-switch and equipped with a 1Ghz– Pentium III processor and a 256MB bank of RAM. An Internet link is also provided for geographical connection and testing. The edge router (called A*lfa*) is based on the popular Linux operating system. In detail, a 2.2.19 version of this kernel was used as a developing platform and partially modified to satisfy our aims. It represents the core of the test bed, since it performs the quality of service functions. To this end, it takes advantage of classification, SLA, and bandwidth management utilities, eventually by the interaction with a bandwidth broker (called *DeisBB*), connected through the switch.

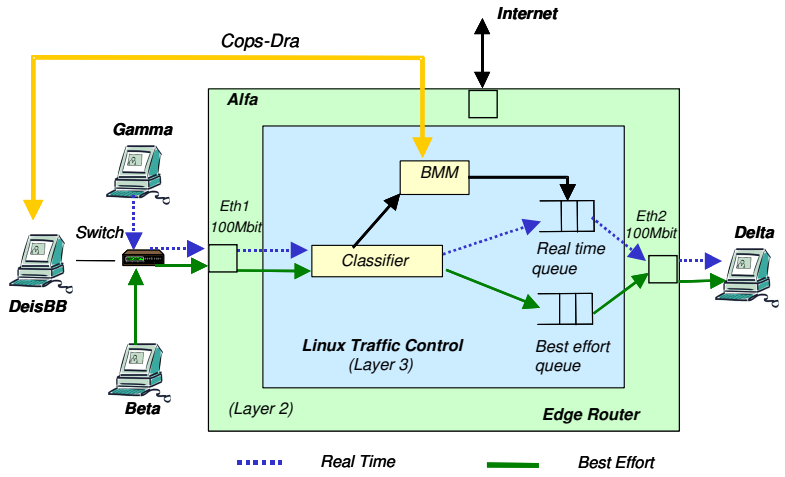

Fig. 5. Functional diagram of the test layout

It is worth noting that the output link of the router is represented by a 10 Mbit/s network card, so that it can be easily saturated; the input interface instead is a 100 Mbit/s card. The other three computers (called Beta, Gamma and Delta) are dual-boot

systems having Linux and Windows 2000 installed; they are exclusively utilised for traffic generation and analysis. In particular, Rude, a traffic generator [17] was installed on Beta and Gamma and used for injecting three distinct flows of traffic into the input port of the router. Specifically, these are a real time flow, a non real time flow (both at 64 Kbit/s), and a best effort flow (at 16 Mbit/s, thus sufficient to saturate alone the output port of the router). Access to the router by Beta and Gamma is obtained through the 10/100 Mbit/s Ethernet switch. A traffic receiver, Crude [17], is set up on Delta: it collects information about the packets coming from the output interface of the router, helping us to verify if the real time flow had been correctly treated. Other applications were also useful for generating the real time flow and evaluating how the system can significantly improve the quality of the communication under a human perspective. Examples of these are the popular Microsoft "NetMeeting", GnomeMeeting and RAT. The traffic control can be configured via the "tc" command, a user-space application which interacts with the Linux kernel to create various objects as queues, classes, SLAs, etc and to initialise them. A graphical front-end interface was also released for easiness of use. The Linux Traffic Control queuing discipline chosen for service differentiation is here based on the CBQ (Class Based Queuing) algorithm. Its configuration assumes two classes, with 1 Mbit/s and 9 Mbit/s rates, respectively, each with a 100 packet-long FIFO buffer attached. Figure 2 shows the basic software architecture of the system.

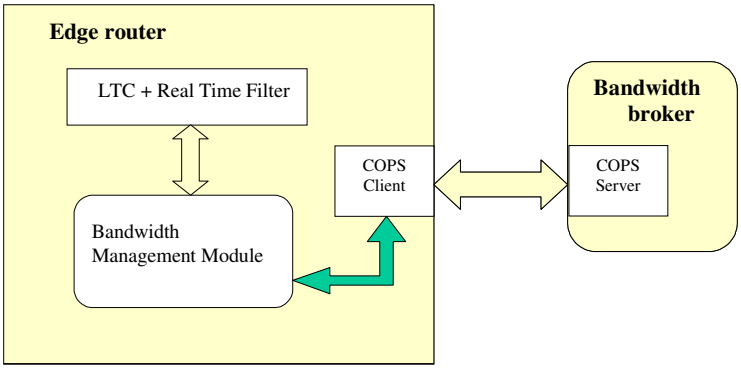

Fig. 6. Software architecture

The Bandwidth Management Module (BMM), is a Unix bash script which interacts with both the Linux Traffic Control and the Bandwidth Broker through the COPS client/server protocol (Common Open Policy Service) [18]. By measuring the real time traffic flows, the BMM decides if the utilised bandwidth is adequate or not. If a bandwidth increase is necessary, it interacts with the bandwidth broker trying to obtain additional bandwidth. The BB keeps the value of the residual bandwidth continuously updated.

3.2 The Real Time Classifier

In this section the approach to allow QoS unaware users of multimedia tools to take benefit of network QoS is described. The new functionality is introduced into the edge router in relation to the network scenario of figure 3, although it can be even introduced within user equipments. The new feature consists in a classifier that, according to a given set of SLAs, performs both protocol and statistical analysis on the traffic incoming from the stub network. The new functionality and its prototype implementation are called Real Time Classifier (RTC). RTC is designed for interactive multimedia applications and, at this moment, is able to recognize and mark that kind of traffic. In terms of DiffServ PHB, RTC marks the traffic recognized as belonging to real-time multimedia streams as EF, setting the IP packet DSCP field. The number of packets necessary used for classification can be chosen independently for each classification algorithm with the aim to optimise the classification delay and failure rate trade-off.

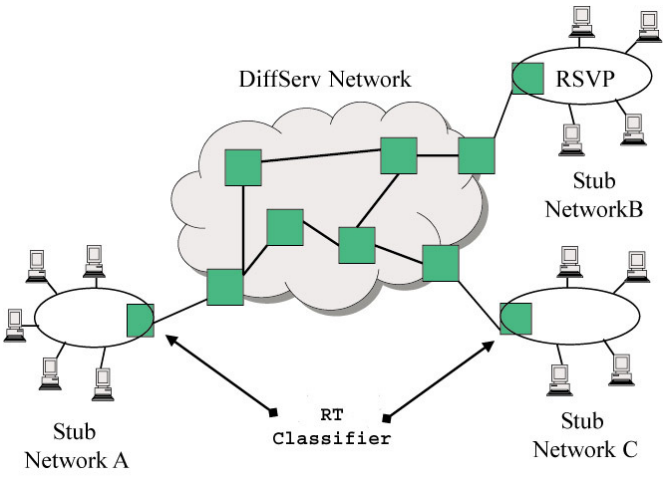

Fig. 7. RTC classifier functionality on the DiffServ Architecture

RTC is actually composed by the following four logical units:

- Classifier: performs traffic classification on protocol and statistical basis;
- Marker: marks the packets according to the classifier policy;
- Meter: meters the incoming traffic;
- Control Unit: manages the SLAs and performs supervision of other units actions.

The control unit has in charge the management of SLA's and the supervision of the whole classification process. For our purposes a 7th-tuple as shown in the table 3 defines a SLA.

Table 5. SLA Format

ID	IP	Mask	BW	Shared	DSCP	Policy

The ID parameter is the unique SLA identifier. The fields IP and MASK are used to identify the host/network belonging to the SLA. The BW parameter is the bandwidth allowed for the considered SLA. The policy parameter can take different values according with the policy adopted for the out-profile traffic of the considered SLA. In particular, at the moment the following values are allowed: OK when the out-profile traffic is forwarded as in-profile and no actions is taken; DROP when the out-profile traffic is discarded. Finally, "shared" is used to specify the degree of fairness to among flows belonging to the same SLA. The value of the field ranges between 0 and 100, and represents the percentage of the bandwidth used on a FCFS basis, with zero meaning that all the bandwidth is equally split between the flows and 100 meaning all bandwidth used on a FCFS basis. RTC control unit has in charge the supervision of whole classification, marking and policing process as a filter between the input interface and the scheduler.

Let consider two user A, and B, using a videoconferencing tool (i.e. NetMeeting) across a DiffServ capable network. Suppose to have a RTC capable router as Edge Router. For each packet coming from the stub network, the RTC control unit performs the following algorithm:

```
while (packet from stub network is received)
{
 if (exists SLA entry for Sender)
 {
   if (packet flow is already classified)
         Mark(Flow_DSCP);
   else Classify();
   if (Meter(packet,SLA)==out-profile)
         Policy(SLA policy);
 }
 else
 {
   Mark(Best Effort);
 }
 Forward();
}
```

The "Classify ()" procedure tries a classification of the incoming packet using both RTP protocol header and statistical information. Once sufficient information has been collected, the flow is classified and its socket tuple is recorded in the hashing table of classified flows. All its subsequent packets are recognized and marked accordingly by matching the value of the tuple. The RTC classifier is the first functional unit encountered by the traffic entering the Differentiated Services domain from a stub network. RTC is integrated in the Linux Kernel QoS framework named Linux Traffic Control, so once classified and marked a packet can be forwarded using a Linux QoS scheduler. RTC classifies traffic using both statistical and protocol analysis. The protocol analysis is based on the RTP header characteristics; in particular there are a few parameters within the RTP header keeping constant their values during the whole session. The RTP header has a minimum length of 96 bits and 42 of them remain constants during the whole session. The protocol-based classification algorithm can be tuned changing the size of the population used to classify. This tuning affects both

the precision of the results and the time needed to obtain them. The presence of the RTP protocol in the analysed flow is considered a sufficient, but not necessary, condition to classify the flow as an interactive multimedia stream.

For this reason, the statistical classification algorithm adds classification capability in the case of real time applications that are not RTP compliant. It takes into account the flow rate and the packet size as main parameters for the classification process. As described in section 2.1, the flow rate depends on the codec used and bandwidth. Typically each IP packet contains one single audio frame. On the other hand when the introduced overhead becomes an issue two or more audio frames are grouped in one IP packet. The number of audio frames per packet is kept as small as possible in relation to the line bandwidth. For example, the codec G.723.1 [19] produces audio frames of 30 ms (33 audio frames per second) and they are generally transmitted one per IP packet if the link bandwidth is enough, but considering a 14.4 kbps modem, they are grouped in three audio frames per IP packet in order to satisfy the bandwidth constraints (table 4).

Table 6. Flow rate and coding delay

	Modem 14.4	LAN
Packet size (Byte)	100 (3 audio frames)	52 (1 audio frame)
Packet/s	11	33
Bit/s	8800	13728
Overhead	28%	52%
Coding Delay	90 ms	30 ms

Taking into account all the parameters, a flow is supposed to be interactive multimedia flow if the number of packets per second is greater than 10 packets per second. This is the lower bound to keep the delay in the constraints defined by G.114 [19]. In this scenario, in order to obtain an acceptable delay for end users, an application has to encode the audio using more than 10 packets per second.

Finally RTC marker unit writes the DSCP value in the DSFIELD of the classified packets according to the classification results. RTC meter unit, in conjunction with the policy unit, performs the enforcement of the traffic profiles.

3.3 Experimental Evaluation

The RTC tool has been implemented hacking a Linux kernel (release 2.2.19) and the command TC included in the package iproute2 [20] in order to realize a new filter (called RTC) of the Linux Traffic Control. Several tests have been done in order to evaluate classification effectiveness and the amount of resources, in terms of memory and time per classified flow, required for the classification process. Three different traffic flows have been generated by Rude [17] in order to saturate the 10 Mbit/s router output link: a real time flow and a non real time flow, both at 64 Kbit/s, and a best effort flow at 16 Mbit/s. Access to the router by Beta and Gamma is obtained through a 100 Mbit/s Ethernet switch. The Linux Traffic Control queuing discipline for service differentiation is here based on the CBQ (Class Based Queuing). Its configuration assumes two classes, with 1 Mbit/s and 9 Mbit/s rates, respectively,

each with 100 packet FIFO queues attached. The main performance figures of interest are the bandwidth used by each flow, the packet loss rate and the time jitter. Jitter is defined with reference to figure 4 as J_tx – J_rx.

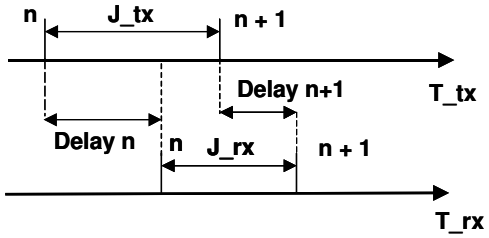

Fig. 8. Main quantities for jitter evaluation

Figure 5 shows the bandwidth usage for real time and non real time traffic during congestion. It is evident that the real time traffic, after the classification process has recognized this kind of traffic, obtains the required bandwidth of 64 Kbit/s even if saturation is present. The bandwidth used by non-real time traffic is on the other hand not stable. Some losses are present for real time traffic due to the layer 2 transmission buffer overflow, where both EF and BE traffics are considered in the same way .

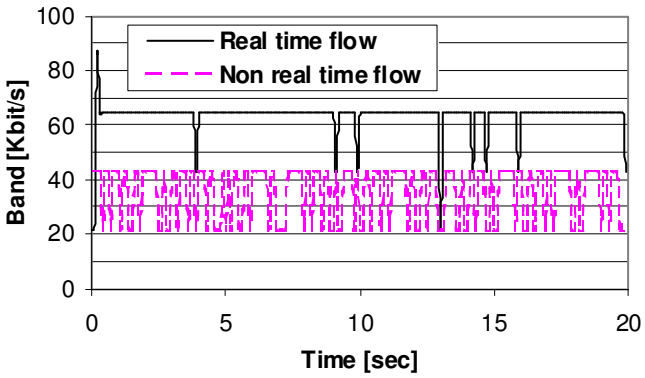

Fig. 9. Link bandwidth usage as a function of time for real time and non real time 64 Kbit/s flows in the presence of a 16 Mbit/s best effort flow over a 10 Mbit/s link

This losses can be eliminated by reducing the upstream CBQ queue size for the BE class, thus limiting the BE traffic offered to the transmission queue. The packet losses in time are represented in figure 6 where is evident the different behaviour of the two flows. The packet loss rate for real time traffic has been calculated to be lower than 2 %. The analysis of transmission delay is presented in figure 7 and 8. Figure 7 shows the percentage of packets at different delay for the to 64 Kbit/s flows. The real time flow has a delay limited at 5 ms, much lower than the delay for the non real time flow.

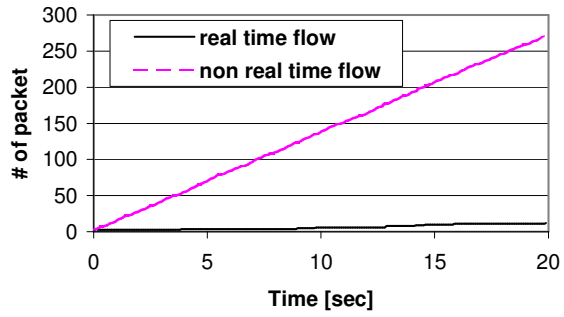

Fig. 10. Packets lost during test for real time and non real time traffic, both at 64 Kbit/s, in the presence of a 16 Mbit/s best effort flow over a 10 Mbit/s link

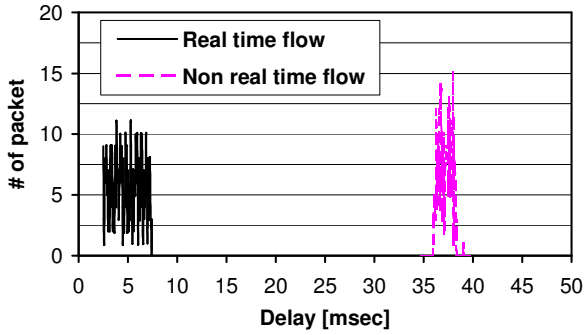

Fig. 11. Distribution of delay for real time and non real time traffic

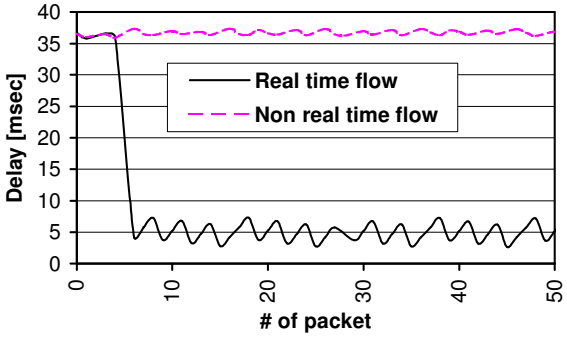

Fig. 12. Packet delay during test for real time and non real time traffic

Figure 8 evidences the effect of the RTC after the time necessary for recognizing the real time flow. At the beginning the two flows are dealt in the same way, then,

after less then 10 ms, when the classification procedure is completed, the different behaviour in terms of delay is evident. Figure 9 shows the temporal jitter as previously defined for real time and non real time traffic. The values of the jitter for real time traffic are acceptable being within 5 ms after the flow has been classified. As regards the jitter for non real time traffic, it is limited only because almost all-non real time traffic is lost and the small percentage of packets that enters the queue typically finds it full.

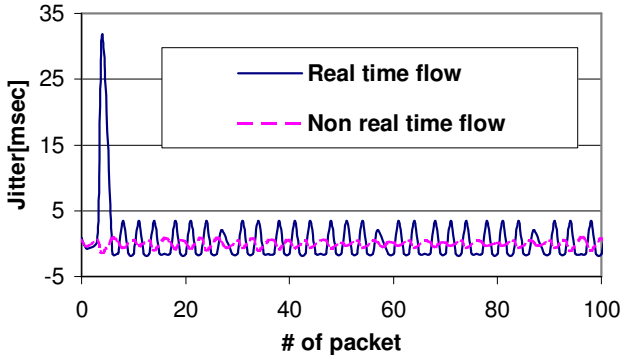

Fig. 13. Time Jitter for real time and non real time traffic

A temporal analysis focused only on real time traffic has been also performed in the presence of best effort traffic at 16 Mbit/s with the insertion at a given time of the RTC function. In figure 10 the time behaviour for the bandwidth used by real time traffic is considered, showing the transition from a situation with not guaranteed bandwidth to a stable one when the RTC function is active.

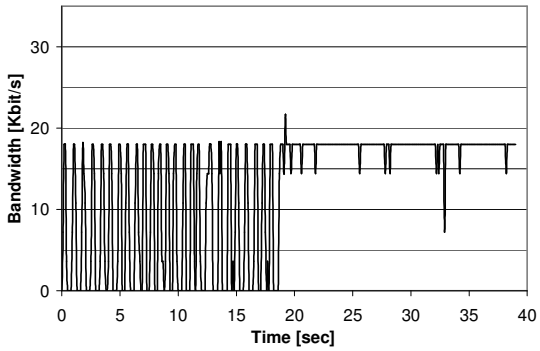

Fig. 14. Effect of the introduction of the RTC function on the behaviour of real time traffic

4 Using the Click Modular Router to implement a QoS Router

The large number of hardware producers, low costs of PCs, and the continuous progress in performance are important factors which are making the design of a packet switch based on a free and open-source platform more attractive [21][22]. Another important demand is nowadays the modularity of the software structure, which would help network manufacturers' revision and design of new functionalities in a router, according to different types of needs. For instance, edge routers, contrary to core routers, usually need more specialized tasks, like packet filtering and classification for Quality of Service. This section describes the design, implementation, and testing of an edge router supporting flow-based classification functions for real time traffic. Protocol and statistical analysis of application flows is performed to provide EF treatment to multimedia traffic without any user signalling. These functions take advantage of the Click Modular Router [23], which is assumed as a starting point to develop flow-based classification of real time services and to demonstrate a viable design procedure to support new multi-service router functionalities Sample measurement shows the effectiveness and feasibility of the proposed approach as a step beyond in the field of open routing design.

4.1 Real-time Flow Classification

The classification procedures require modification of the reference Click router diagram and new output queue management modules. The new functionalities have been evaluated in terms of latency and classification effectiveness. Comparisons of performance are presented to show the processing overhead introduced by classification. In the following, the approach to allow QoS unaware users of multimedia tools to take benefit of network QoS is described. The new service is introduced into the edge router and consists in a classifier that, according to a given set of SLAs, performs both protocol and statistical analysis on the traffic incoming from the stub network. The new functionality and its prototype implementation are called *Real Time Classifier* (RTC). Figure 11 depicts the modules inserted in the Click framework to realize the RTC. Comparing it with the RFC1812 router described in [24], the RTC replaces here the simple FIFO-based output queuing scheme, being it inserted between the basic router and the output device interface.

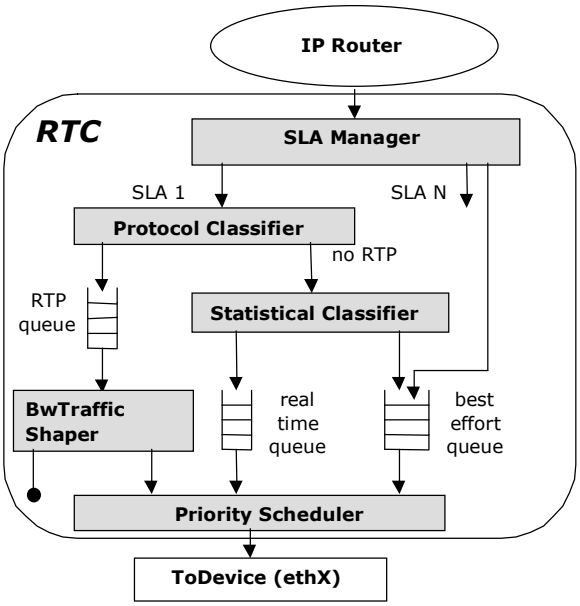

Fig. 15. The RTC internal structure

The RTC is designed for interactive multimedia applications and is able to recognize and mark that kind of traffic. In terms of DiffServ PHB, RTC marks the traffic recognized as belonging to real-time multimedia streams as EF, setting the IP packet DSCP field. The number of packets required for classification can be chosen independently for each classification algorithm with the aim to optimize the classification delay and failure rate trade-off. With reference to figure 11, the SLA Manager module is dedicated to identify the traffic belonging to a certain SLA (see table 3). The SLA Manager is implemented using an existing Click element, *IPClassifier*, which performs a pattern-based filtering to examine the source IP address and checking if it pertains to a specific SLA. If so, the packet is passed to the Protocol Classifier, which is as a new Click compound element designed and added to the library; otherwise, it is pushed into the BE (low-prioritized) output queue.

The Protocol Classifier, depicted in Figure 12, is able to filter the RTP-marked traffic flows. The IETF RFC1889 [25] establishes that the RTP packet must always contain a 32-bit field called SSRC (Synchronization SouRCe identifier), which is kept constant and distinct for each single flow. The *GetSSRC* element extracts these useful bits from the RTP header and passes them to the *RTPClassifier*, which is able to hook each distinct RTP flow if obtaining N occurrences of its SSRC value in a T-seconds interval of time. Once a flow is identified, an internal table keeps the flow in the "classified" state, deleting it only after 30s of inactivity. The *SetIPDSCP* element is then involved in marking the TOS field of the IP header. The presence of the RTP protocol in the analyzed flow is considered a sufficient, but not necessary, condition to classify it as a multimedia stream. For this reason, the Statistical Classifier does an additional analysis. Figure 12 also depicts the structure of this other new compound

element. The statistical classification algorithm adds classification capability in the case of real time applications, which are not RTP compliant. It takes into account the flow rate and the packet size as main parameters for the classification process. Typically, these parameters depend on the source encoding used and bandwidth, as mentioned in section 2: for instance, a flow can be considered an interactive multimedia flow if the packet rate is greater than 15 packets/s and the packet size is less than 200 bytes. Once sufficient information is collected, the flow is classified as real-time. Obviously, since no SSRC field is present in this case, miscellaneous flows cannot be distinguished from each other. The *SplitFirst(J)* block is destined to redirect (in the BE queue) the leading J packets: this mechanism should guarantee the cascading modules to receive stable data. *CheckAverageLength(K)*, instead, is able to monitor the mean size of the last 15 received packets: if greater than K bytes, the current packet is passed to the BE queue.

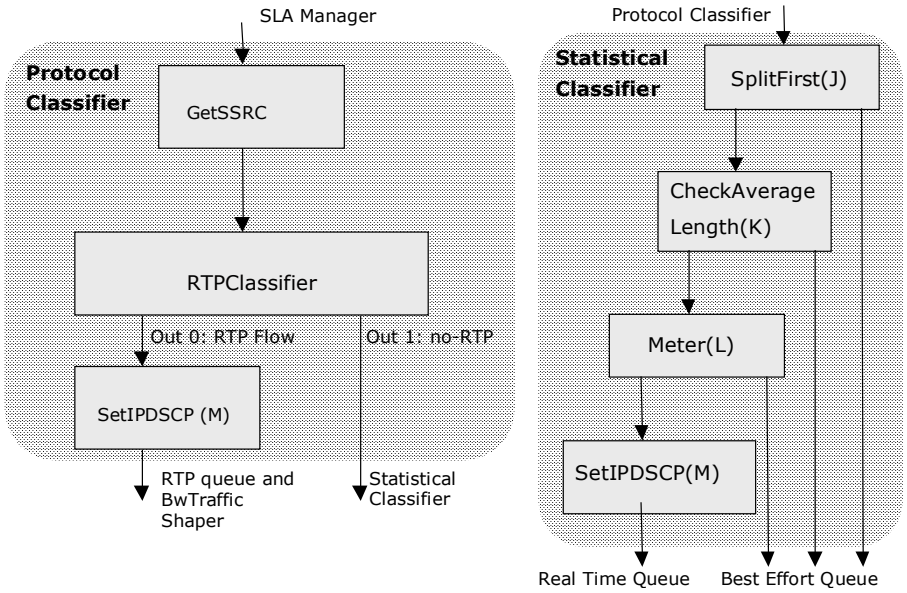

Fig. 16. Protocol based and statistical classification

Otherwise, it is pushed in the *Meter(L)* element, which measures the packet rate and classifies the real-time flow, when that is greater than L packets/s. Accordingly to the SLA "BW" and "Policy" parameters, another new block was designed to manage the out-of-profile traffic with more flexibility, the *BwTrafficShaper*, similar to the standard BandwidthShaper, but with a secondary output dedicated to the out-of-profile traffic. While the in-profile traffic is shaped as required, it allows to apply various policies to the out-of-profile one: besides being discarded, this can be redirected elsewhere (for instance, to the BE queue).

4.2 Test-bed Layout and Configuration

In order to perform quality of service trials, a testing configuration was designed to emulate functions of a real network environment and to offer real time services with quality of service. Figure 13 shows the test-bed layout, which consists of four PC-based systems, connected through a Gigabit Ethernet layer2-switch. The edge router (*Alpha*) is equipped with a 1.6 GHz Pentium IV Processor. The other PCs (*Beta*, *Gamma*, and *Delta*) are exclusively utilized for traffic generation and analysis and have an on-board 1Ghz–Pentium III processor. Every PC is equipped with the Intel PRO1000XT-Server network adapters. On our router, we plugged two of them on the 32-bit/33 MHz PCI bus, even if they would be ready to work with the more advanced 64-bit/133 MHz PCI-X bus. The choice of this NIC was led by the necessity of using the same polling-based driver already developed by the MIT for the Intel PRO1000 family cards.

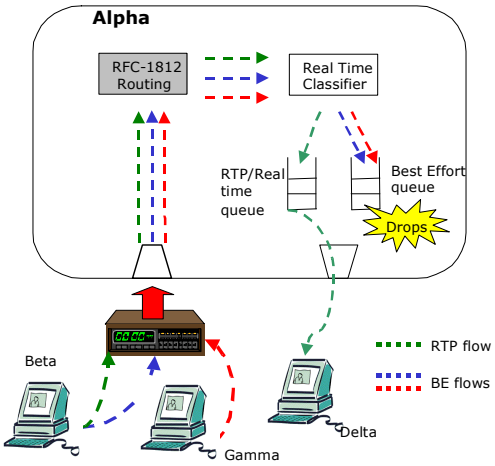

Fig. 17. Functional diagram of the test-bed layout

All the PCs were installed with the 2.4.9 version of the Linux operating system and Click release 1.2.4 (with the Intel Pro1000 4.3.15 driver added). The edge router performs the quality of service functions. To this end, it offers RTP-based and statistical classification of multimedia traffic, and SLA management. When necessary, the output link of the router was also tightened to work at 10 Mbit/s, so that it can be easily saturated. *RUDE*, a traffic generator, was installed on Beta and Gamma, which were used for injecting three distinct flows of traffic into the input port of the router. Specifically, these are a real time flow, a non real-time flow, and a best effort flow (at 16 Mbit/s, thus sufficient to saturate alone the output port of the router). Access to the router by Beta and Gamma is obtained through the Gigabit Ethernet switch. A traffic receiver, *CRUDE*, is set up on Delta: it collects information about the packets coming from the output interface of the router, helping us to verify if the real time flow was correctly treated. Other applications, the popular Microsoft "NetMeeting" or RAT, were also useful for generating the real time flows and evaluating how the system can significantly improve the quality of the

communication under a human perspective. Other more general measurements were done using Click installed on *Beta* and *Delta* as a traffic generator and collector.

4.3 Experimental Evaluation

Figure 14 shows performance offered by the described hardware platform. The router can manage a maximum loss-free forwarding rate of 120,000 packet/s with the interrupt-based drivers and 370,000 packet/s using the polling technique. The main performance figures of interest for a QoS capable router, however, are the packet delay and the time jitter. Figure 15 depicts the average delay time of an RFC-1812 polling-based router, at different input packet rates with 60 s runs of traffic.

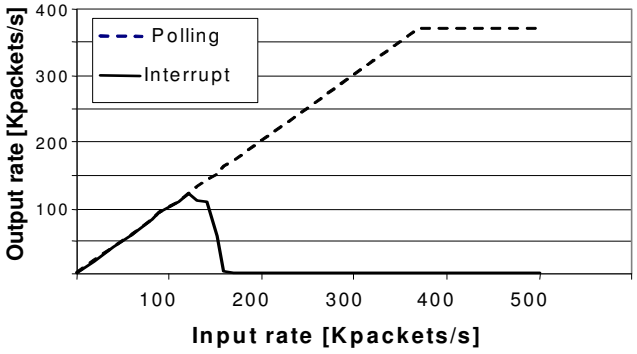

Fig. 18. Forwarding rate as a function of input rate for a Click-based RFC-1812 router using interrupt and polling techniques (64-byte packets)

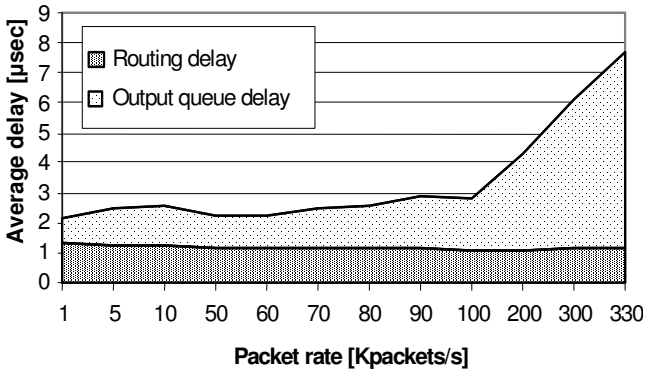

Fig. 19. Average delay time for a Click-based RFC-1812 router as a function of input rate (64-byte packets and 60 seconds runs)

The main contribution to the global delay is due to the output queue term (a FIFO with 100 elements), while the routing process alone requires about 1.2 µs. Under

these conditions, no packet loss is observed. It is to mention that 10 elements FIFOs were also tested: the global delay is much smaller (about 2-3 μs), but packets drops are observed when coming up to the maximum input rate. More detailed analyses were done for the QoS capable router, after the insertion of the Real Time Classifier. In this case, its output link was tightened to 10Mbit/s, inserting a *BandwidthShaper* before the *ToDevice* block (which interfaces the Click environment to the NIC). The *RUDE*s installed on Beta and Gamma were used for injecting three distinct flows of traffic into the input port of the router. Specifically, a real time flow, a non real-time flow (at 512 Kbit/s), and a best effort flow (at 16 Mbit/s, thus sufficient to saturate alone the output port of the router). Table 5 illustrates the average delay time for the RTP and the best-effort traffic, and the maximum output queues length reached during the runs (the output queues are 100 elements FIFOs). As expected, the RTP traffic is forwarded with a much smaller delay (≈32 μs) than best effort traffic (typically 5 ms).

Table 7. Average delay time and maximum output queues length for a RTP 512 Kbit/s flow in the presence of a 16 Mbit/s best effort flow over a 10 Mbit/s link

Traffic	Average IP Routing Delay	Average RTC Delay	Average Click Delay	Max Queue Length
RTP	1.19 μs	30.75 μs	31.94 μs	13
Best Effort	1.19 μs	> 5000 μs	> 5000 μs	Overflow

With the same contour conditions, an additional evaluation was done about the end-to-end delay of the RTP traffic. This is feasible since RUDE marks any packet with a sequential time-stamp, making possible to determine the total delay of the packets, due to the path from the generator to the receiver. This measurement is influenced by all the delay factors inside the test-bed (the traffic generator and collector inner delays, the switch latency, the Click delay, and the NIC-to-Click transit time). Figure 16 shows how the end-to-end delay for the RTP flow is about 120 μs. It is determined in the presence of the implemented functions and compared with a wired generator-to-receiver connection. The latency introduced by all the functions inserted in the Click environment is denoted with T_{click} and is given by

$$T_{click} = T_{routing} + T_{rtc}, \quad (5)$$

where $T_{routing}$ is the delay introduced by the RFC1812 router part, and T_{rtc} is the delay due to the RTP classifier blocks. The values of these contributions were measured through the Click support. The end-to-end lag can be resumed as

$$T_{delay\_with\_click} = T_{gen} + 2*T_{sw} + 2*T_{nic} + T_{click} + T_{coll}, \quad (6)$$

where T_{gen} is the time spent by the traffic generator to put the packets on the wire, T_{sw} is the switch latency (2,5 μsec), T_{nic} is the NIC-to-Click transfer time, and T_{coll} is the time spent by the traffic collector to retrieve the packets from the wire. The direct wire connection instead is

$$T_{delay\_with\_wire} = T_{gen} + T_{sw} + T_{coll} \ . \quad (7)$$

Thus, knowing that T_{click} = 31,94 μsec, $T_{delay\_with\_click}$ =123 μsec, $T_{delay\_with\_wire}$= 58,2 μsec, it is possible to estimate the value of

$$T_{router} \equiv T_{nic} + T_{click} + T_{nic} = 62{,}3 \text{ μsec} \; ; \; T_{nic} \approx 15{,}18 \text{ μsec}. \quad (8)$$

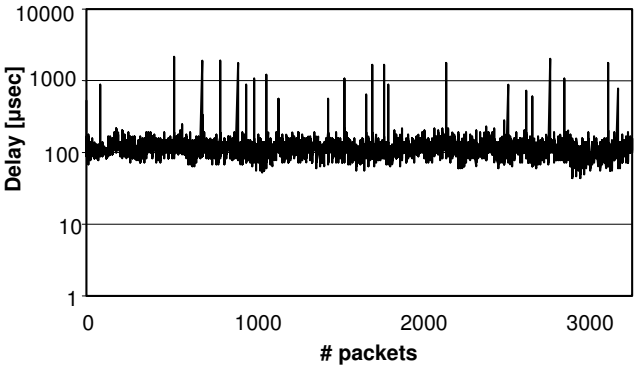

Fig. 16. End-to-end delay for a RTP 512 Kbit/s flow in the presence of a 16 Mbit/s best effort flow over a 10 Mbit/s link

A point to mention regards the graphed data in figure 16: it is possible to notice a few high delay peaks in the picture. The same spikes were also observable in the case of the wire connection delay and are not due to an odd behaviour of the router, but to some extraneous synchronism activity on the traffic generator and collector systems. An additional analysis of the transmission delay (inside the Click environment) is presented in figure 17 and 18.

Figure 17 shows the distribution of the delay for the RTP and BE flows, while figure 18 depicts the temporal jitter for the same kinds of traffic. Most of the values of the jitter for real-time traffic are acceptable being within 100 μs after the flow has been classified. As regards the jitter for non real time traffic, it is limited to some hundreds of microseconds only because almost all non-real time traffic is dropped and the small percentage of packets that enters the queue typically finds it full.

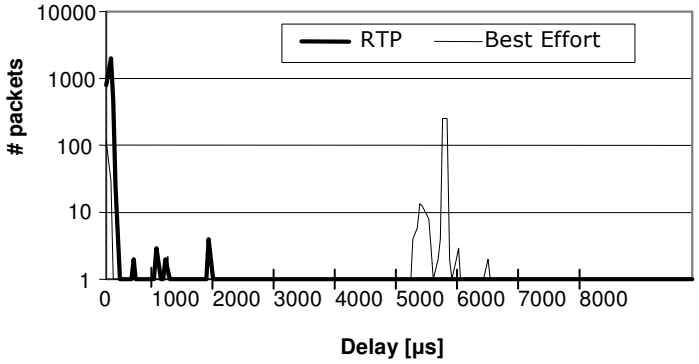

Fig. 17. Delay time distribution for a RTP 512 Kbit/s flow in the presence of a 16 Mbit/s best effort flow over a 10 Mbit/s link

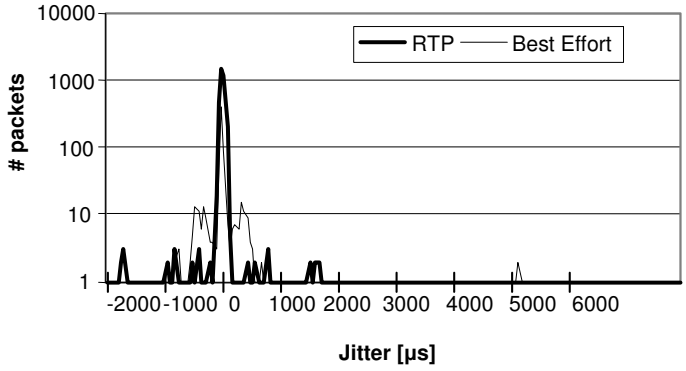

Fig. 18. Jitter distribution for a RTP 512 Kbit/s flow in the presence of a 16 Mbit/s best effort flow over a 10 Mbit/s link

5 Conclusions

In this chapter the design, implementation and testing of a flow based real-time classifier called RTC was described. RTC uses different methodology to perform its function, based on protocol analysis and traffic patterns. Being it flow-oriented, it can perform functions such as bandwidth usage measurement and thus can be fruitfully used in a dynamic bandwidth management scheme. The results show effectiveness of RTC in recognizing real time flows and in guaranteeing bandwidth as limited delays for these kinds of applications. The RTC was implemented in both the Linux kernel environment and Click Modular Router.

Chapter References

1. Keshav S., Sharma R.: Issues and trends in router design, IEEE Communication Magazine, vol.36, n.5, pp.144-151, May 1998
2. Xipeng Xiao, Ni L.M: Internet QoS: a big picture, IEEE Network , Volume: 13 Issue: 2 , March/April 1999 Page(s): 8 –18
3. R. Braden Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin: Resource ReSerVation Protocol RSVP) Version 1 Functional Specification, Request For Comment 2205, IETF, 1997
4. R. Braden, D. Clark, S. Shenker: Integrated Services in the Internet Architecture: an Overview, Request for Comments 1633, Internet Engineering Task Force, June 1994
5. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss: An Architecture for Differentiated Service, Request For Comment 2475, IETF, Dec. 1998
6. V. Jacobson, K. Nichols, K. Poduri : An Expedited Forwarding PHB, Request for Comments 2598, Internet Engineering Task Force, June 1999
7. J. Heinanen, F. Baker, W. Weiss, J. Wroclawski: Assured Forwarding PHB Group, Request for Comments 2597, Internet Engineering Task Force, June 1999
8. K. Nichols, V. Jacobson, L. Zhang. A Two-bit Differentiated Services Architecture for the Internet. Request for Comments 2638, Internet Engineering Task Force,. July 1999
9. Andreas Terzis, Jun Ogawa, Sonia Tsui, Lan Wang, Lixia Zhang: A Prototype Implementation of the Two-Tier Architecture for Differentiated Services, RTAS99, Vancouver, Canada, 1999
10. D. Su and J. Srivastava, and Jey-Hsin Yao. Investigating factors influencing {QoS} of Internet phone. In Proc. of IEEE International Conference on Multimedia Computing and System, pages 308-313,June 1999
11. J. Postel. User Datagram Protocol. Request for Comments 768, IETF, Aug 1980
12. H. Schulzrinne, and S. Casner, and R. Frederick, and V. Jacobson. RTP:a Transport Protocol for Real-Time Applications. Request for Comments 1889, Internet Engineering Task Force, Jan. 1996
13. H. Schulzrinne, and A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP). Request for Comments 2326, IETF, Apr. 1998
14. International Telecommunication Union (ITU). Transmission Systems and Media, General Recommendation on the Transmission Quality for an Entire International Telephone Connection; One-Way Transmission Time. Recommendation G.114, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Mar. 1993
15. J.C. Bolot, A.V. Garcia: Control Mechanisms for Packet Audio in the Internet, INFOCOM, San Francisco, California, Mar. 1996
16. Opens H.323 group. Codec Bandwidth and Latency Calculations. http://www.openh323.org/bandwidth.html, June 2000
17. http://www.atm.tut.fi/rude/
18. R. Mameli, S. Salsano: Use of COPS for Intserv Operations over Diffserv: Architectural Issues, Protocol Design and Test-bed implementation, ICC 2001, Helsinky
19. http://www.itu.int
20. ftp://ftp.sunet.se/pub/Linux/ip-routing/
21. Click Modular Router, http://www.pdos.lcs.mit.edu/click/, MIT, Cambridge, MA
22. E.Kohler, R.Morris, B.Chen, J.Jannotti, M.F.Kaashoek: The Click modular router. ACM Trans. Computer Systems 18, August 2000
23. J.C.Mogul, K.K.Ramakrishnan: Eliminating receive livelock in an interrupt-driven kernel. ACM Trans. Computer Systems 15, August 1997
24. E.Kohler, R.Morris, B.Chen: Programming language optimizations for modular router configurations. ACM SIGPLAN Notices, Volume 37, October 2002
25. http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1889.html - RTP, Real Time Protocol

Chapter IV. Design and Implementation of Adaptive Algorithms for Inter-domain Dynamic Bandwidth Allocation

1 Introduction

The evolution of the Internet to support different quality of service classes, especially needed for real time services, requires models and techniques for network engineering and resource management, which should be suitable for large communication infrastructures and meet efficiency and scalability requirements. An intense research activity on models for service differentiation [1],[2], [3] has been developed in the last few years and, more recently, on the techniques for resource management [4], [5], [6], [9], [10]. Among these, the scheme based on the Bandwidth Broker concept has been considered as suitable to cope with the Differentiated Services model proposed for QoS support [4]. Although this approach is fairly centralized it can be made scalable through hierarchical organization of functions as proposed in [6]. This proposal split the resource management problem into intra-domain and inter-domain functions with different administrative scope: intra-domain resource management is referred to the bandwidth broker of the domain and is typically controlled by a single organization, while inter-domain resource management involves interactions between bandwidth brokers of different organizations and, on the basis of the proposal, it is achieved by bilateral agreements between adjacent domains. The aim is to optimize the usage of the inter-domain link and, at the same time, enhancing system scalability through the reduction of the number of requests issued to the bandwidth brokers of the interworking domains.

At the same time the edge router must be equipped with new functionalities, in order to resolve the classes of traffic designated for a privileged treatment, and to manage efficiently the resources arranged by its bandwidth broker. Concerning this, leading routers' producers nowadays offer proprietary solutions, which are often complex or impossible to improve and adapt. Instead there is a growing attention of the international scientific and industrial community for open and standard platforms supporting free software. To this end, the Click router [7] is a flexible and modular framework for the simple design and the rapid implementation of new services. In the previous chapter, it was used for the development of flow-based protocol and statistical classifiers of real time traffic. New functionalities are now going to be presented to meter the aggregated traffic of a service class in order to dynamically modulate the inter-domain link bandwidth and accomplish its effective usage. These new functions have been evaluated in terms of latency, efficiency and number of requests for the bandwidth broker.

The chapter is organized as follows: in section 2 the general model for resource management is introduced; in section 3 a simplified threshold-based model for resource management is discussed, focusing on the problems of design parameters,

and measurement process of traffic; the effect of the measurements process will also be discussed and exploited to manage bandwidth allocation even in the presence of congestion and a combination of the original algorithm with a strategy for congestion management is proposed to overcome the instability problems that could arise in real operating environments; in section 4 a new bandwidth update algorithm based on the use of the logarithmic function is introduced, together with the performance analysis and design procedures presentation; the new algorithm is characterized by few design parameters whose main effects on performance are mostly independent of each other. Both sections 3 and 4 demonstrate how the bandwidth allocation schemes were implemented within the Click Modular Router environment: system performance measurements are presented and discussed. A comparison of the two different bandwidth allocation schemes is also presented in terms of time behaviour, efficiency and packet latency.

2 The System Model

The network model that has been considered is represented by multiple interconnected domains each equipped with a resource manager called Bandwidth Broker (BB) [4]. The BB is supposed to be responsible for call admission functions and resource management within the single domain and resource management on inter-domain links. This last is of particular interest in network design because it involves the interaction between different administrations to achieve a trade-off between performance and costs by optimizing the bandwidth allocated on inter-domain links in relation to the real link usage. A threshold-based mechanism is here considered based on the proposal presented in [5] and is briefly summarized as regards the interactions with the BB to increase/decrease the bandwidth allocated to inter-domain links. The mechanism is applied to a single class of service to which a given amount of bandwidth b_d is initially allocated by network manager. The basic operations are sketched in figure 1 and involve the edge routers (ERs) that interface with the inter-domain link and the bandwidth broker of each domain. The inter-domain link bandwidth is assumed shared among different service classes. Dynamic allocation of the link bandwidth to service classes is provided for each domain to achieve efficient bandwidth utilization. Bandwidth allocation management is performed in relation to the aggregate traffic of a single class that transits through adjacent domains. The following operations are performed as illustrated in figure 1:

- the originating edge router ER1 sends a request for bandwidth increase/decrease to the bandwidth broker BB1;
- BB1 forwards this request to BB2 that verify through a query to ER2 the bandwidth availability;
- BB2 notifies BB1 of the result;
- BB1 enables ER1 for its request.

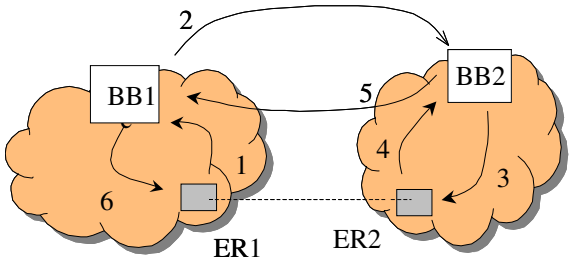

Fig. 20. Network model based on the bandwidth broker concept for inter-domain resource allocation

The interactions between ERs and BBs can be implemented using protocols like COPS [8]. With this scheme each BB manages a number of interactions that is related only to the ERs of its domain and to the BBs of directly connected domains, thus enhancing the scalability of the whole system. The mechanism is here applied to a single class of service, which is initially equipped with the amount of bandwidth allocated by the network manager. ERs accessing network links are responsible of monitoring bandwidth usage by the aggregate traffic of a class and of asking the related BBs for the necessary increase/decrease of allocated bandwidth; an increase/decrease request is assumed here always followed by a positive answer within a time T_{bb}.

3 Threshold-based Dynamic Bandwidth Allocation

3.1 The threshold-based mechanism

A well-known approach is based on a threshold-based system that behaves as explained in the following [6][8]. Let us introduce the following symbols:

- b_d the bandwidth currently allocated to the service class;
- b_o the bandwidth currently used by the aggregate traffic of the class;
- $t_u = b_d * l_{max}$ upper threshold;
- $t_l = b_d * l_{min}$ lower threshold.

The following assumption is made: the initial bandwidth is allocated by BBs on inter-domain links to an application session in relation to parameters specified during a call admission procedure. If call admission is not provided no initial bandwidth is allocated. No per flow information is kept in any case in the BBs. ERs accessing inter-domain links are responsible of monitoring link bandwidth usage by aggregate traffic and of asking the related BBs for the necessary increase/decrease of allocated bandwidth. A request for additional bandwidth is sent to the BB only when $b_o > t_u$ and

a request for release is sent to the BB when $b_o < t_l$. No requests are sent to the responsible BB if $t_l < b_o < t_u$. Bandwidth increments/decrements are performed by means of coefficients i and d, respectively, that update the allocated resources b_d on the basis of the following relationships: $b_d' = b_d * i$, in the case of bandwidth increase, and $b_d' = b_d * d$, in the case of bandwidth decrease. An increase/decrease request is always followed by a positive answer within a time T_{bb}. It should be noticed that this system has a main drawback consisting in the large number of parameters to set up for system design.

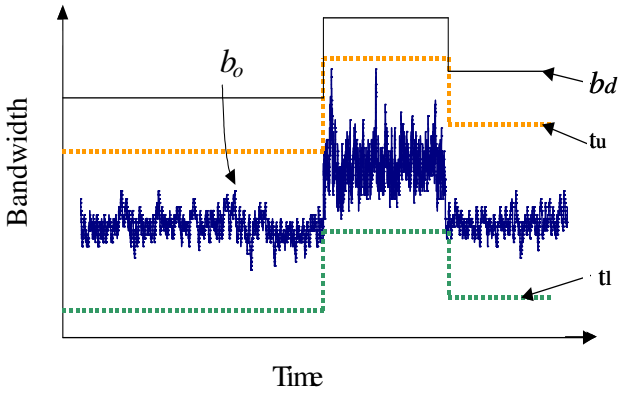

Fig. 21. Sketch of the threshold-based mechanism

Figure 2 gives a sketch of the threshold-based mechanism by neglecting the time required for bandwidth updating. The algorithm described requires the system to know the existence of a flow, as it can result from a classification process. The parameter b_o can be ideally assumed exactly known (ideal system) or be the result of a measurement procedure (measurement-based system) whose characteristics and effects will be analyzed in the following.

3.2 Bounds for design parameters

The choice of the correct parameter configuration of system described is crucial to achieve bandwidth utilization efficiency, scalability and system stability. In fact, under particular conditions of traffic patterns, bandwidth oscillations could arise as a consequence of wrong parameter configuration. These oscillations could be unavoidable for traffic patterns with rapid and consistent fluctuations. In figure 3 a configuration that causes bandwidth oscillations is presented. Starting from bandwidth assignment B, oscillation can arise after an increment if the new bandwidth $B' = i * B$ causes b_0 to be lower than $B' * l_{min}$; similarly, after a decrement, oscillation arises if the resulting bandwidth $B'' = d * B$ cause b_0 to be higher than $B'' * l_{max}$. Both these situations require a new bandwidth update in the opposite direction.
Under the hypothesis of slowly variable traffic behaviours during the update time, sufficient conditions can be obtained to avoid oscillating bandwidth updates, that are

$B \cdot l_{\max} > B' \cdot l_{\min} = (B \cdot i) \cdot l_{\min}$ and $B \cdot l_{\min} < B'' \cdot l_{\max} = (B \cdot d) \cdot l_{\max}$

that result in $l_{\max} > i \cdot l_{\min}$ and $l_{\max} > \frac{1}{d} \cdot l_{\min}$ respectively.

The two conditions can then be simultaneously expressed by:

$$l_{\max} = k \cdot l_{\min} \text{ with } k = \max(i, \frac{1}{d}) > 1.$$

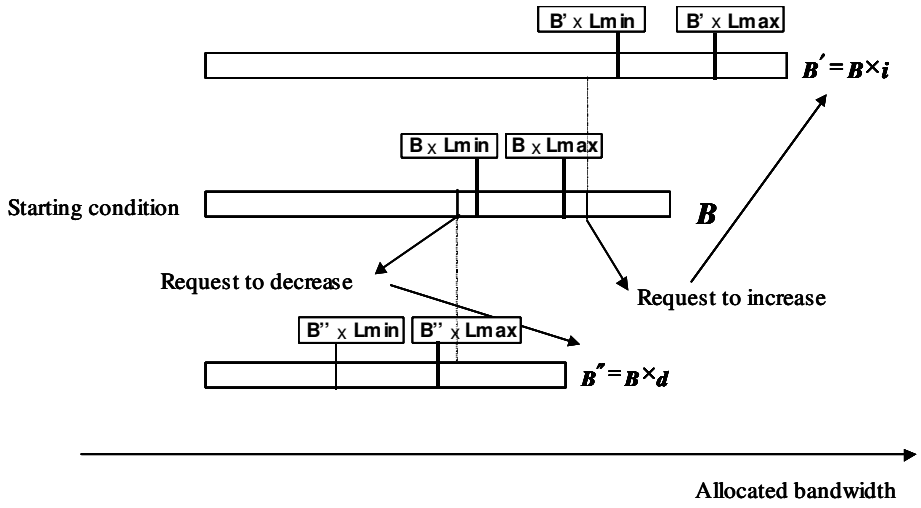

Fig. 22. Configurations of system parameters that give rise to bandwidth oscillations

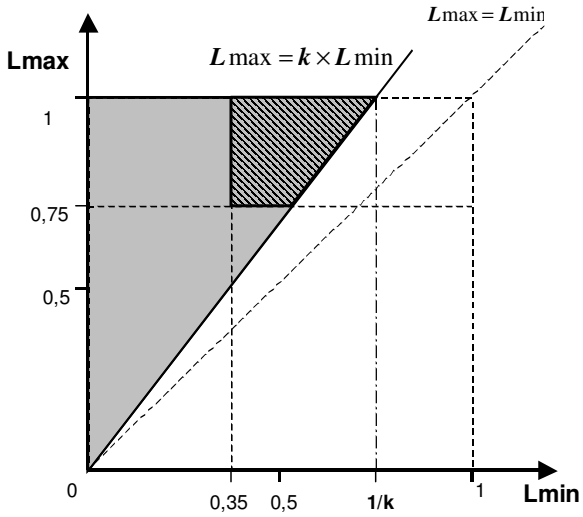

Fig. 23. Admissible configurations of system parameters

The previous relationships can be graphically represented as shown in figure 4, which gives the area of admissible configurations for l_{min} and l_{max} depending on the slope of the limiting line. The limiting line is determined by only one of the values d and i, the one which gives rise to the most requiring condition. Further delimitations of this area are consequences of considerations related to efficiency and congestion limitation, which will be discussed later in the paper; in the example they are fixed at 0.35 and 0.75 for l_{max} and l_{min} respectively. In practical implementations of the mechanism, the bandwidth is constrained to assume values on a discrete range, being it updated through fixed amount of bandwidth. In this case a further relationship between d and i can be obtained by observing that the value after the decrement (increment) that follows an increment (decrement) results to be equal to the starting value so that B=B'·d=B·i·d that turns in i·d=1 and B=B''·i=B·d·i that turns in i·d=1 again.

In order to gives a guideline for the choice of the parameters let us refer to bandwidth efficiency as the main performance target. To this end the driving parameter of the set up procedure is represented by l_{min}. The procedure starts from a reference value for l_{min} and tries to increase it, to improve bandwidth usage as far as other constraints, represented, for example, by the frequency of interactions with the bandwidth broker or by the percentage of traffic overlimit, are satisfied. At this point different strategies must be followed depending on the condition that is taken into account. If the percentage of traffic overlimit is considered, it is possible to further increase l_{min} by decreasing accordingly l_{max} . If the frequency of interactions is considered it is possible to optimize the efficiency by increasing l_{max} accordingly. As regards d and i, they should be both decreased to diminish the overlimit traffic while i should be increased and d decreased to reduce the number of interactions. The diagram of the empirical optimization procedure for efficiency is sketched in figure 5.

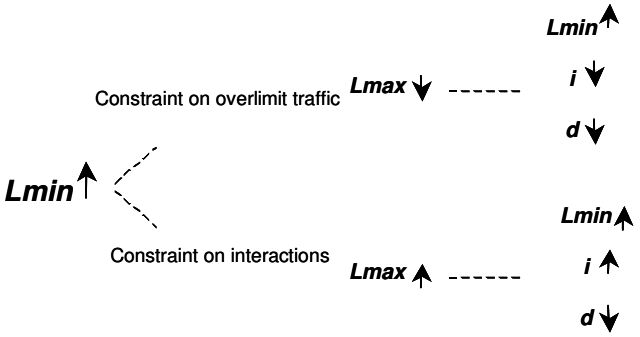

Fig. 24. Diagram of parameters set up in the efficiency optimization procedure

3.3 Effects of the measurements process

Previous considerations on the threshold based mechanism refer to the ideal situation of exact knowledge of current bandwidth usage. In practice the evaluation of bandwidth usage is the result of a measurement process that influences the values which are available for the algorithm. A simple time window mechanism is assumed here to evaluate the influence of the measurement process on the knowledge of the parameters. The measurement process assumes the time divided into interval of size T each in its turn further divided in intervals of size S. The average bandwidth is calculated over each S-interval and the measure updated each time the results is greater than the current value. The maximum over T of these values is maintained as bandwidth estimate until the end of one subsequent T interval as a maximum. This measurement process has the advantage to be very simple and to take into account possible correlation in traffic at the same time. It tends to capture the envelope of the traffic, depending on traffic behaviour and on the values of parameter T and S. Some evaluations of the measurement algorithm have been performed by simulation to investigate the effect of parameters S and T in relation to traffic characteristics. The results presented here a related to a traffic aggregate resulting from 60 Pareto sources with the following characteristics:

T_{ON}	T_{OFF}	Peak transmission rate
Pareto α =1.2, k=0.4	Pareto α =1.2, k=0.4	32 Kbit/s

thus obtaining a traffic aggregate with long range dependent characteristics. Figure 6 shows a comparison between the real behaviour, the simple algorithm with window T=1·S and the window based algorithm with T=10·S.

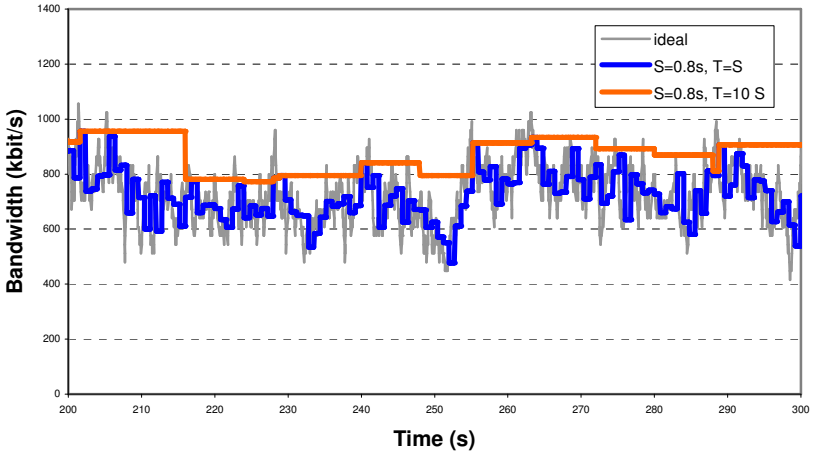

Fig. 25. Time behaviour of bandwidth for different configurations of the measurement process

The simple algorithm better follows the dynamics of the trace but is delayed and averaged on S thus typically giving an underestimate of the bandwidth usage. On the contrary, by extending the window T, a more stable value is produced by the measurement process although in excess with respect to the real requirements. In order to suitably choose the values of T and S the difference between the average of real and estimated behaviour has to be minimized while trying to maximize the fraction of time during which the estimate is greater than the real behaviour.

3.4 An algorithm for congestion management

Real time services require low delay within the node that can be assured by a suitable set up of the bandwidth allocated for the corresponding EF class in the Differentiated Services model. In any case congestion can temporarily arise due to delays in bandwidth estimate and update, that causes packets to be queued waiting for transmission resources. The introduction of l_{max} can reduce the occurrence of these events, but a trade off must be reached between system performance and efficiency. So the threshold-based mechanism can be fruitfully coupled with a congestion resolution mechanism based on the monitoring of the queue occupancy. The main aim of the algorithm is to limit the time to get through the system by setting the maximum acceptable time T_s to empty the queue. So the queue threshold S_q in bytes is given by $S_q=T_s*B_d/8$, being B_d the allocated bandwidth at time t. The threshold S_q varies as a function of B_d. As a consequence of the previous definition, congestion is defined as a system state with queue occupancy greater than S_q. When congestion arises, the measurement-based algorithm is excluded and the following operations take place:

- bandwidth update is performed such that the new threshold S_q' is greater than queue occupancy; to this end a margin M_S is introduced as a percentage of the

current queue occupancy Q such that $S_q' = (1+M_s/100)*Q = T_s*B_d'/8$ from which the new value of bandwidth that assure the target delay T_s can be obtained.

- after a bandwidth update, if the queue occupancy is still greater than the threshold (due to further arrivals during the update time) a new update is performed with the same rule;
- when congestion finishes, that is when the queue occupancy gets below the threshold, the normal algorithm is used after a guard time T_g: this is important to allow the measurement process to produce the new output value and thus avoiding unsafe oscillations. The value T_g should be at least two times S_q to obtain the stable value.

The packet transfer delay is thus bounded by $T_s+2\ T_{bb}$. For example for a target maximum packet transfer delay of 30 ms in the router with T_{bb}= 10 ms, T_s must be limited to 10 ms. On the other hand T_s should be large enough to avoid too frequent activation of the congestion control algorithm. In all other cases the threshold-based algorithm prevents congestion through l_{max}. The frequency of activation of the congestion management algorithm can be reduced through the adoption of a larger guard bandwidth corresponding to lower values of l_{max}.

3.5 Performance Analysis

Performance analysis has been developed by simulation with the aim to compare a system with static bandwidth allocation, a system with dynamic bandwidth allocation based on the exact knowledge of the real behaviour and the measurement based system with the congestion control algorithm. The following general hypothesis are made for the dynamic cases:

- the answer time for the bandwidth broker is assumed equal to 10 ms;
- the result of the answer is always positive;
- the edge router output link capacity is 10 Mbit/s.

Evaluations has been performed with two kinds of ON/OFF traffic whose characteristics are summarized below.

Table 8. Traffic configurations used in the simulations

Type	# sources	T_{ON}	T_{OFF}	Peak transmission rate
1	60	Pareto α=1.2, k=0.4	Pareto α=1.2, k=0.6	32Kbit/s
2	60	Pareto α=1.2, k=0.04	Pareto α=1.2, k=0.6	128Kbit/s

Static allocation – The static allocation is considered related to the sum of the peak bandwidths required by each flow of the aggregate. This results in a very low efficiency although the time transparency is good as shown in the table for the traffic types previously described.

Table 9. Efficiency and delay for static peak rate allocation

Type	Static bandwidth	Efficiency (%)	Average delay (ms)
1	1920000	40,77%	0,0187
2	7680000	9,29%	0,0175

Ideal system – Performance for a system that works on the basis of the ideal knowledge of the real behaviour are here reported for type 1 traffic in terms of efficiency, defined as the ratio between the average occupied bandwidth and the average available bandwidth, average delay and number of interaction with the bandwidth broker. In figure 7 the efficiency is shown to increase with the threshold l_{min}; at the same time the average delay increases as shown in figure 8 thus calling for a trade off between these two aspects. Also the number of interactions increases with l_{min} (figure 9).

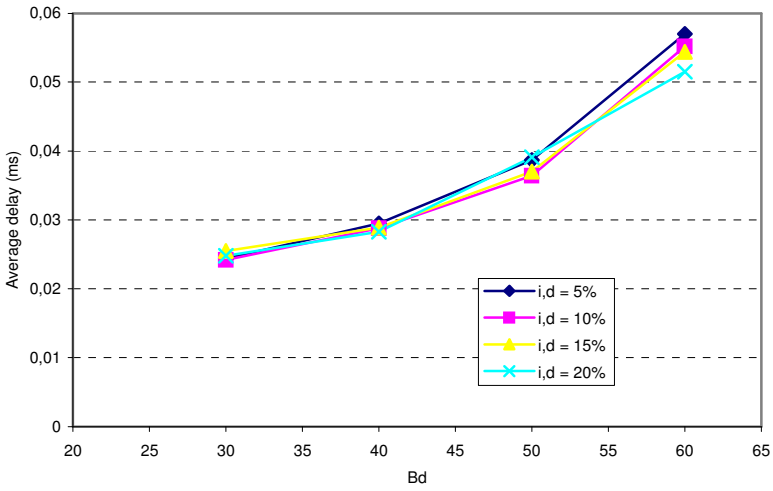

Fig. 26. Average delay for the dynamic allocation as a function of the lower threshold l_{min} with exact knowledge of bandwidth usage

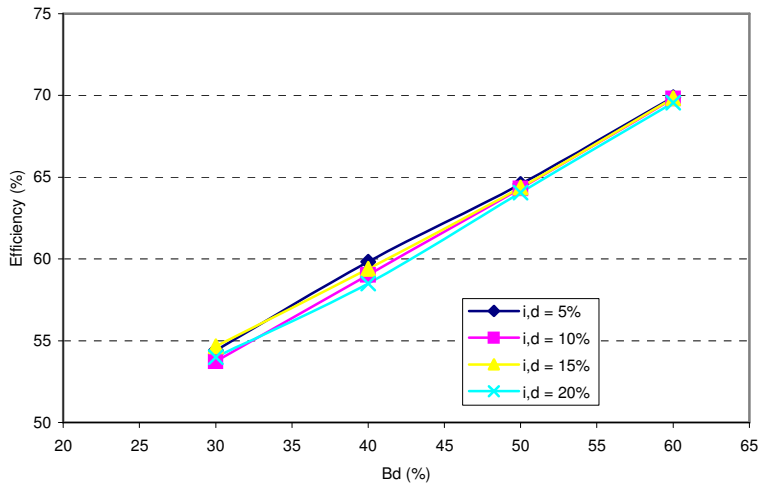

Fig. 27. Efficiency for the dynamic allocation as a function of the lower threshold l_{min} with exact knowledge of bandwidth usage

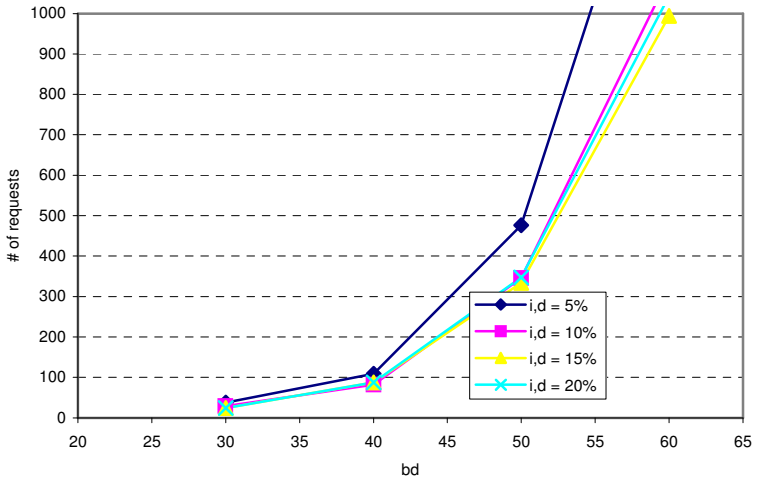

Fig. 28. Number of interactions as a function of the lower threshold l_{min} with exact knowledge of the bandwidth usage

Measurement based system without congestion control – The dynamic bandwidth allocation is here considered in relation to the time window measurement process described in section 2.3. A system with S=0.8s and T=10 S has been simulated and its time behaviour was shown in figure 6. It is evident the difference between the output of the measurement process and the real behaviour. In particular the dynamic of the measurement bandwidth is delayed and, in this case, always within the guard band of

the dynamic allocation algorithm. This causes the allocated bandwidth to be stable although the real bandwidth has a more dynamic behaviour. This is expected to lead to lower efficiency but at the same time to a lower number of interactions with the bandwidth broker while the delay is expected to be lower too.

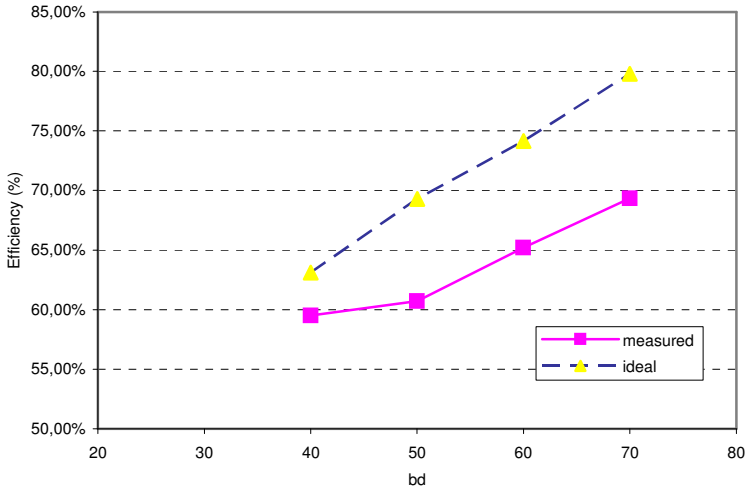

Fig. 29. Comparison of efficiency for the measurement based algorithm and the ideal approach with traffic type 1, with S=0.8s, T=10·S, i=d=10%, l_{max}=90%

In figure 10 a comparison in terms of efficiency with the ideal case is presented and shows the lower efficiency of the measurement based system that in any case is much higher than in the static case. On the other hand the delay is better as shown in figure 11 and the system can be considered more suitable for quality of service guarantees. Also the number of interactions with the bandwidth broker presented in figure 12 is much lower and stable thus assuring the scalability of the solution.

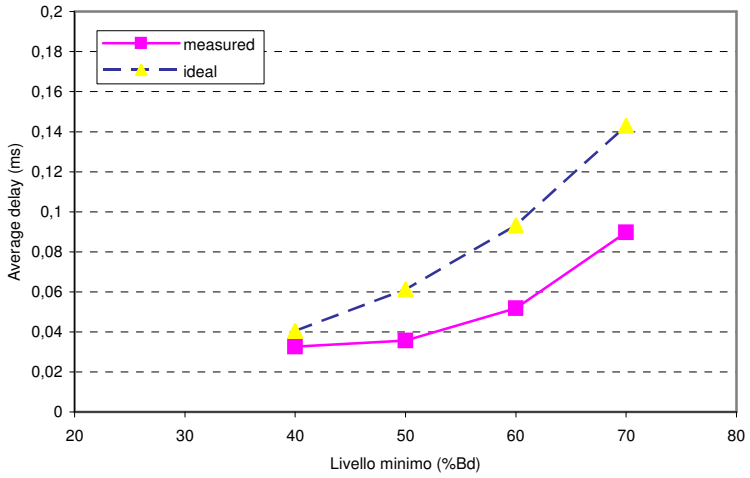

Fig. 30. Comparison of the average delay for the measurement based algorithm and the ideal approach with traffic type1, with S=0.8s, T=10·S, i=d=10%, l_{max}=90%

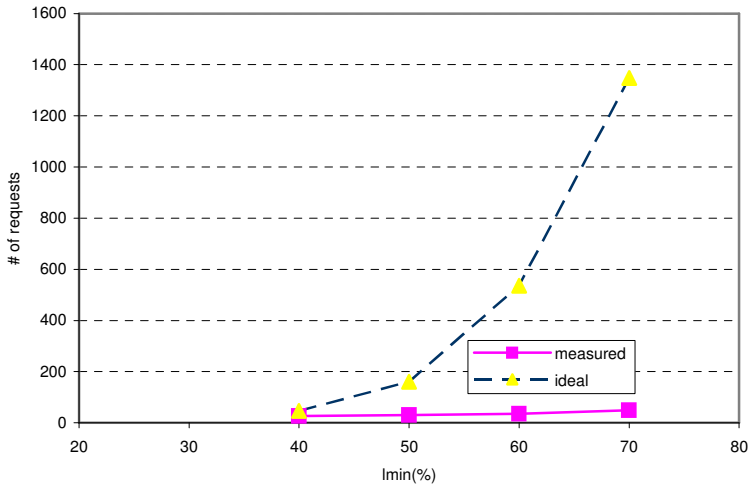

Fig. 31. Comparison of the number of interactions for the measurement based algorithm and the ideal approach with traffic type1, with S=0.8s, T=10·S, i=d=10%, l_{max}=90%

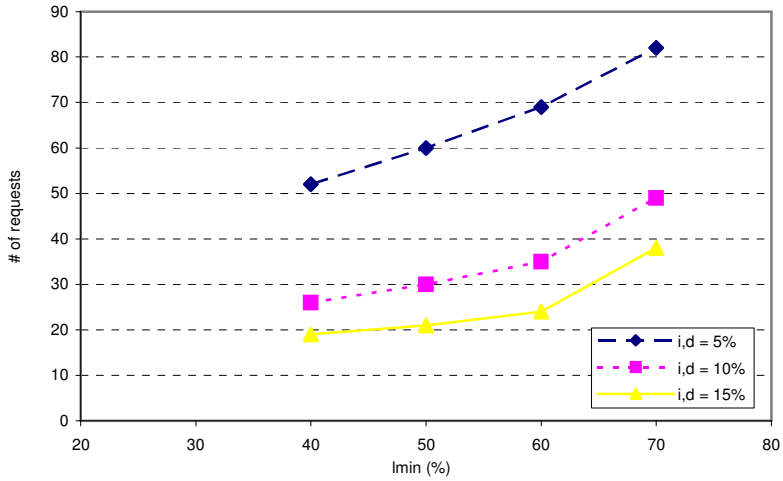

Fig. 32. Dependence of the number of requests on l_{min} for the measurement based algorithm with traffic type 1, S=0.8s, T=10·S, l_{max}=90%,, for different values of i=d

Measurement based system with congestion control - Finally the introduction of the congestion control algorithm in the measurement based system is evaluated. In table 3 results regarding the maximum delay are given and show that it is always less than 35ms, as expected, being T_{BB} =10 ms and T_s =15ms. The delay without congestion control is also reported.

Table 10. Maximum delay: comparisons for the measurement based system with and without congestion control. (i=d=10%, T_s=15ms, M_s=20%, T_g=1.6s)

Type	(S,T)	l_{min}%	l_{max}%	Max delay without control [ms]	Max delay with control [ms]
1	(0.8s, 10S)	70%	90%	61.8	34.5
2	(0.8s, 50S)	60%	80%	37.6	22.1

The following results are given for type 1 traffic. Figure 14 shows the efficiency as a function of the threshold and figure 15 shows the number of interactions. The efficiency increases with T_s and accordingly the number of interactions decreases. On the other hand T_s must be limited for QoS guarantees, being it involved in the expression of maximum delay. In figure 16 efficiency degradation for the system with QoS control is shown for T_s =15ms.

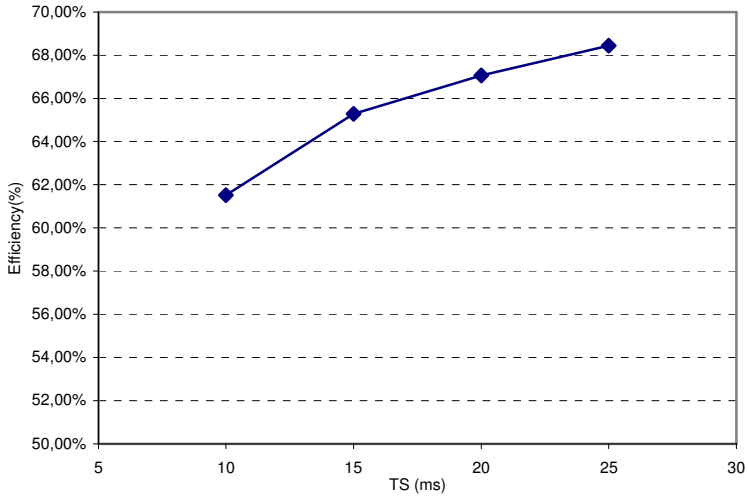

Fig. 33. Efficiency as a function of the lower threshold for type 1 traffic for the dynamic allocation algorithm with congestion control (S=0.8s, T=10·S, l_{max}=90%, l_{min}=70%, i=d=10%, M_S=20%, T_g =1.6s)

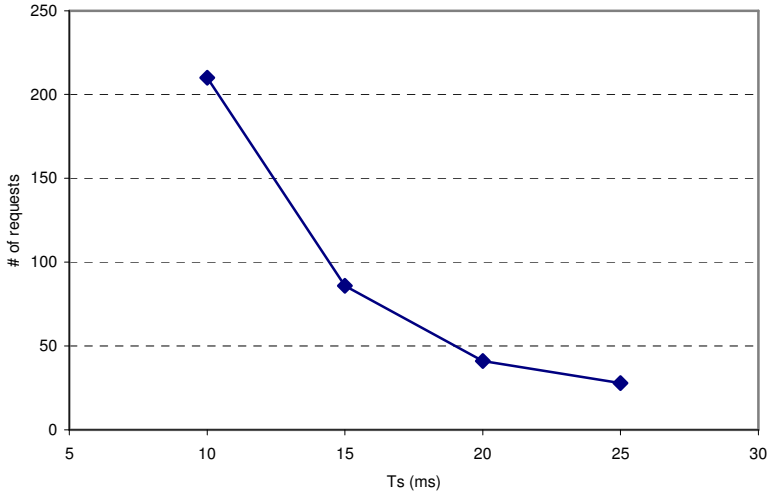

Fig. 34. Number of interactions as a function of the lower threshold for type 1 traffic for the dynamic allocation algorithm with congestion control (S=0.8s, T=10·S, l_{max}=90%, l_{min}=70%, i=d=10%, M_S=20%, T_g =1.6s)

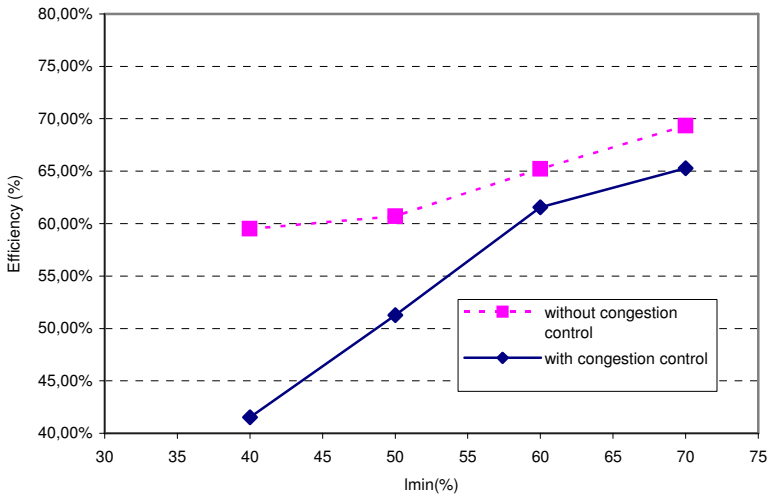

Fig. 35. Comparison in terms of efficiency as a function of the lower threshold for type 1 traffic for the dynamic allocation algorithm with congestion control (S=0.8s, T=10·S, l_{max}=90%, i=d=10%, T_S=15ms, M_S=20%, T_g =1.6s)

3.6 Dynamic Bandwidth Management Design and Implementation

In the previous chapter, the Click modular router was utilized for implementing flow-based classification of real time services. A set of new modules (the *Real Time Classifier*) was inserted into the edge router to change the simple FIFO-based output queuing scheme of a RFC1812-compliant router. Coherently with a given set of Service Level Agreements (SLA), it proved the possibility of achieving protocol and statistical analysis of the traffic incoming from the stub network, real time flows recognition and privileged treatment (i.e. limited delays). The RTC is designed for interactive multimedia applications and is able to recognize and mark that kind of traffic. In terms of DiffServ PHB, RTC also marks the traffic recognized as belonging to real-time multimedia streams as EF, setting the IP packet DSCP field. The former scheme is here enhanced with new functions, which implement the algorithms for the measurement and dynamic allocation of bandwidth exposed in the previous chapter. Figure 17 resumes the modules inserted into the Click environment to realize all the described functions:

- the *SLA Manager*, implemented using an existing Click element, is dedicated to select the traffic belonging to a set of existing SLAs;
- the *RTC* compound module contains the Protocol and the Statistical Classifiers: the first is able to hook and filter the RTP-marked traffic; the latter, analyzing the flow rate and the packet size, adds classification capability in the case of real time applications that are not RTP compliant (thus the presence of the RTP protocol is considered a sufficient but not necessary condition to classify a

multimedia stream);

- the *RoundRobinScheduler* is introduced to multiplex and balance the distinct traffic flows pertaining to different SLAs;
- the *PriorityScheduler* implements different policies for the real time (high-prioritized) and best effort (low-prioritized) packets;
- the *MeterShaper* module is designed to implement the threshold-based algorithm previously described for aggregated bandwidth measurement and management.

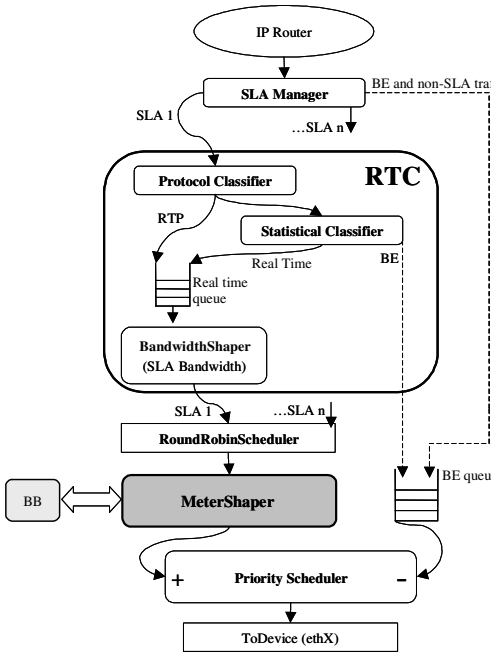

Fig. 36. The QoS router internal structure. The BB module represents the domain Bandwidth Broker (external to the Edge Router)

The internal structure of this new compound element, depicted in figure 18, shows the primitive modules composing it. The *BwTrafficShaper* is inherited from the BandwidthShaper C++ class (available in the Click standard libraries) and is enriched with a more flexible functionality: while the in-profile traffic is shaped as required, it branches out the out-of-profile packets to its secondary output. These can be discarded or, accordingly to a different policy, redirected to the best effort queue. The BwTrafficShaper is also interfaced with the *Meter* element, which implements the time-window measurement algorithm (metering the amount of traffic passing through the shaper) and modulates the allocated bandwidth (modifying the shaper parameters). It's important to remark here that Click elements are C++ objects and their methods are always called when a "trigger signal" appears (i.e. a packet arriving to their input ports): thus they are executed asynchronously. On the contrary, the Meter module must measure the used bandwidth on a strict synchronous basis and independently

from the real-time packets' arrivals: in fact the measurement process assumes the time divided into intervals of size S while the arrival of the real-time packets is not predictable. A time reference (i.e. clock signal) is thus supplied to the Meter using the *TimedSource* module as a synchronous source of dummy packets. These packets also contain the value of the S interval inside their payload, so that the Meter module is informed about the timing currently imposed. The MeterShaper element can be configured using several handlers: besides admitting the values of i, d, l_{min} and l_{max}, the "inc_parameter" and "dec_parameter" can be used to notify the Bandwidth Broker about the demand or release of bandwidth; the boolean "BB_response" reports the arrival of the BB answer. Others are available for performance monitoring and have been used extensively for the evaluations reported below: "allocated_bit_rate" and "current_bit_rate" export the value of the allocated and measured bandwidth, while "BB_connections_inc" e "BB_connections_dec" count the number of increments or decrements requests sent to the bandwidth broker.

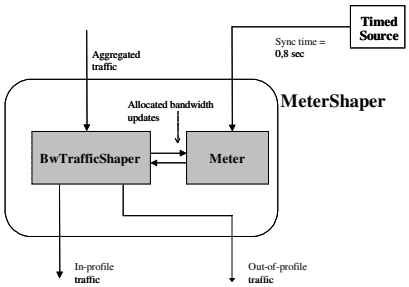

Fig. 37. The Meter Shaper element

3.7 Implementation issues

In order to perform quality of service trials, the testing configuration already described in Chapter III (§ 4.2) is used. Figure 19 recalls the test-bed layout, which consists of four PC-based systems, connected through a Gigabit Ethernet layer2-switch. The edge router (*Alpha*) performs the quality of service functions and its output link is tightened to work at 10 Mbit/s, so that it can be easily saturated. The other PCs (*Beta*, *Gamma*, and *Delta*) are exclusively utilized for traffic generation and analysis. All the PCs are installed with the 2.4.9 version of the Linux operating system and Click release 1.3.

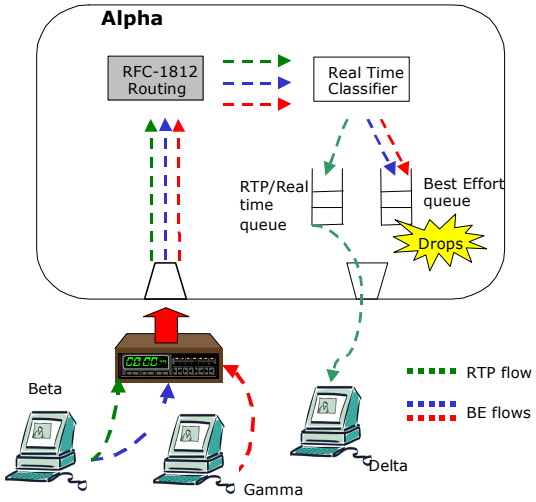

Fig. 38. Functional diagram of the test-bed layout

3.8 Performance measurements

Measurements were done using *Beta* and *Gamma* as traffic generators and *Delta* as a collector. Three distinct flows of traffic are injected into the input port of the router. Specifically, a real time flow, a non real-time flow, and a best effort flow (at 16 Mbit/s thus sufficient to saturate alone the output port of the router), have been used. Under these contour conditions, Table 4 illustrates the average delays for the real-time traffic, obtained by twenty 60-second runs (all output queues are 100 elements FIFOs). It is important to remark that the real time traffic rate is kept constant and small enough not to activate the bandwidth management algorithm. The multimedia traffic is forwarded with a much smaller delay (≈32 μs) than best effort traffic (typically 5 ms), and with no losses, showing the effectiveness of the introduced functions; the latency is independent from the packet size and rate used for the real-time flow itself.

Table 11. Average latency in microseconds for real-time traffic traversing the Click router. Rows correspond to the packet sizes used. Columns represent the multimedia flow rate. A 64-byte best-effort flow at 16Mb/s is also injected. The router output link is tightened to 10 Mbit/s

Delay (μs)	64 kb/s	128 kb/s	256 kb/s	512 kb/s	1024 kb/s
64 byte	31,51	31,33	31,05	30,99	30,84
128 byte	31,60	31,59	31,50	31,09	31,29
256 byte	31,85	31,69	31,61	31,54	31,16
512 byte	32,09	32,16	31,78	31,63	31,72

The delay contribution of all the QoS modules inserted in the Click environment has been evaluated: Table 5 reports the typical values registered during the trials. If no

best effort traffic is contemporary injected, conducted tests not reported here show how the delay for the multimedia flow is much lower (typically about 4 μs). On the contrary, packet length variations for the best effort traffic modify the real time packet latency, as reported in fig. 20.

Table 12. Delay contribution of the modules composing the QoS router. This measure is obtained with a real-time flow of 1024 kb/s (512-byte packets). A 64-byte best-effort flow at 16Mb/s is contemporary injected. The router output link is tightened to 10 Mbit/s

IP Router	1200 ns
SLA Manager	150 ns
Protocol Classifier	1320 ns
Statistical Classifier	1110 ns
RTP/Real Time Queue	27700 ns
BwTrafficShaper	60 ns
Round Robin Scheduler	60 ns
MeterShaper	60 ns
Priority Scheduler	60 ns

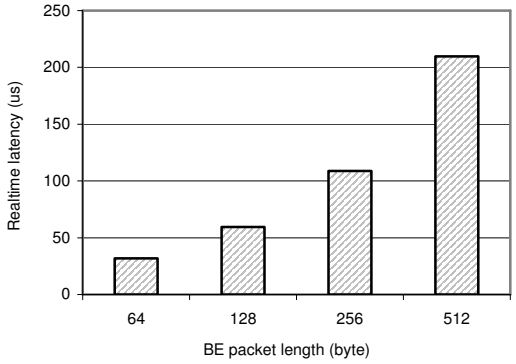

Fig. 39. Latency of the real time packets as a function of the best effort packets length

To investigate the proper behaviour of the MeterShaper module, the real-time traffic shape has then been varied. Measurements were obtained by soliciting the system through steps of bandwidth variations of the RTP flow, as shown in figure 21. At the same time, the best effort flow has been removed. The response time of the bandwidth broker is here neglected and the result of the answer is always positive. A good tracking of the measured bandwidth obtained by the allocation algorithm can be observed.

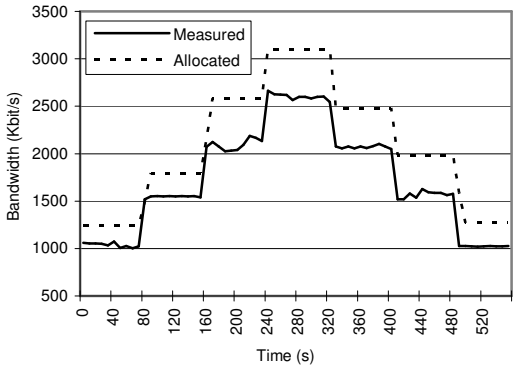

Fig. 40. Measured vs. allocated bandwidth with steps size of bandwidth variations of 512Kbit/s (S=0.8s, T=10·S, i = 1.2, d = 0.8, l_{max} = 90%, l_{min} = 70%)

In figure 22 the efficiency in the bandwidth usage is shown as a function of the bandwidth variations. Efficiency is here defined as the ratio of the average values of the measured and allocated bandwidth. Figure 23 illustrates how the frequency of requests to the bandwidth broker varies, depending on the step size. Figure 24 sketches how the bandwidth constraint introduced by the MeterShaper element influences the packets' latency. The delay suddenly rises up when the step size is 2048 kbit/s, since a ramp peak transmission rate of 8192 kbit/s is reached, thus close to the output link capacity (which is limited at 10Mbit/s).

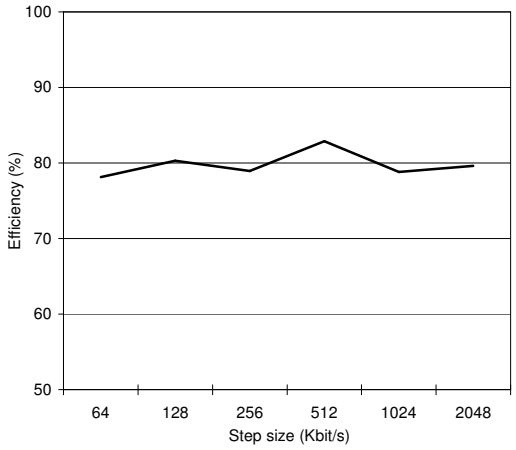

Fig. 41. Efficiency in the bandwidth usage, varying the step size (l_{max}=90%, l_{min}=70%, i=1.2, d=0.8)

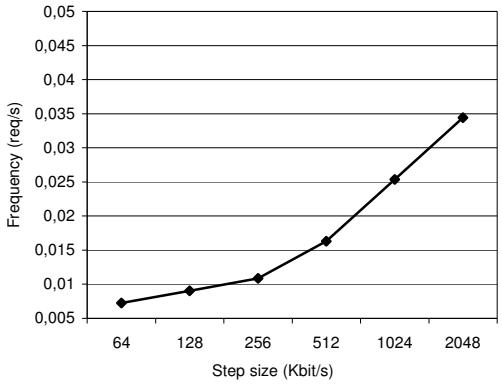

Fig. 42. Frequency of interactions with the bandwidth broker depending on the step size

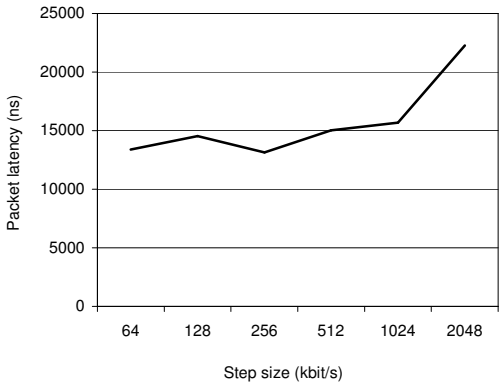

Fig. 43. Average packet latency for the RTP traffic as a function of the step size

4 A New Approach for Bandwidth Update: the Log-based Dynamic Allocation Scheme

4.1 The logarithmic-based algorithm

The choice of the correct parameters' configuration of the threshold-based system described above is crucial to achieve bandwidth utilization efficiency, scalability and system stability. In fact, under particular conditions of traffic patterns, bandwidth oscillations could arise as a consequence of approximate parameter configuration. On the other hand, the parameters of the threshold-based model are strictly related to each other and strongly dependent on the traffic behaviour, thus making the parameters set

up a very critical point. A new procedure is here proposed, where the bandwidth update is based on a logarithmic function that reduces the number of design parameters. It uses the output of a measurement-based process to know the value B_m of the bandwidth currently used by the aggregate traffic of the service class. So B_m replaces b_o of the threshold-based scheme described before. The traffic measurement is based on the same time-window measurement system already explained in the previous paragraph. The choice of the logarithmic function was suggested by the need to carefully increment the bandwidth, when necessary, in order to avoid sudden congestion and, at the same time, to rapidly decrement the bandwidth to promptly reduce its waste. The chosen function intrinsically meets these characteristics when used to calculate the bandwidth update ΔB as

$$\Delta B = K * \ln\left(\frac{B_m}{B_d * S}\right), \quad (9)$$

being B_m the measured bandwidth, B_d the allocated bandwidth, and having indicated with ln(x) the natural logarithm of x. The parameter K is the constant of the feedback system and S is a margin to avoid sudden congestion. So the new bandwidth value after the update is given by

$$B'_d = B_d + \Delta B . \quad (10)$$

In order to avoid too frequent requests to the bandwidth broker, the update procedure is applied only if the following condition holds:

$$\Delta B > \Delta B_{min} . \quad (11)$$

So ΔB_{min} must be suitably chosen to assure the scalability of the approach.

This system requires three parameters to be defined: K, S and ΔB_{min} whose influence on bandwidth efficiency, scalability and delay will be evaluated in the next paragraph.

4.2 Performance analysis

Performance analysis has been developed by simulation with the aim to prove the effectiveness of the logarithmic algorithm. The following general hypotheses are made:

- the answer time for the bandwidth broker (T_{BB}) is assumed equal to 10 ms;
- the result of the answer is always positive;
- the edge router output link capacity is 10 Mbit/s.

Evaluations have been performed with three kinds of ON/OFF traffic whose characteristics are summarized below in table 6. The aggregate traffic is considered as generated by the superposition of 60 sources. The instantaneous bandwidth used is given by the sum of contributions of sources that are in the ON state at the instant considered. The value of the bandwidth used during the ON period (peak transmission rate) is suitably modified to obtain the same average for all traffic types.

Table 13. Traffic configurations used in the simulations

Type	*# sources*	T_{ON}	T_{OFF}	*Peak transmission rate*
1	60	Pareto,α=1.2, k=0.4	Pareto,α=1.2, k=0.6	32Kbit/s
2	60	Exp λ=1	Exp λ=1	26 Kbit/s
3	60	Pareto,α=1.2, k=0.04	Pareto,α=1.2, k=0.6	128Kbit/s

The main performance figure evaluated is the bandwidth efficiency, defined as

$$\rho_b = \frac{1}{T}\int_T \frac{b_0(t)}{b_d(t)}dt \cong \frac{b_{0m}}{b_{dm}} \quad , \tag{12}$$

being b_{dm} and b_{om} the average values of available and occupied bandwidth. In the following evaluations K is chosen as a constant: its value mainly influences the number of interactions with the bandwidth broker while having practically no effects on efficiency and delay. It has been set equal to 600 Kbit/s to minimize the number of interactions for all kinds of traffic considered (figure 25).

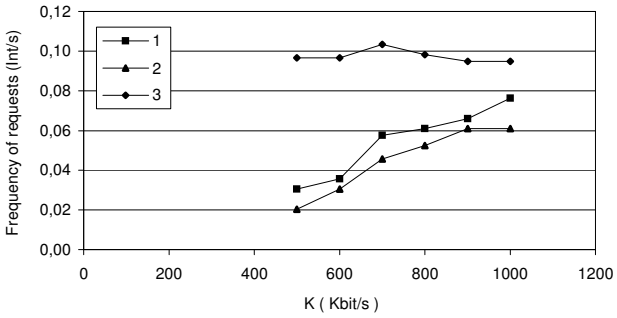

Fig. 44. Frequency of requests at the bandwidth broker as a function of K for the three kinds of traffic for ΔB_{min} =50 Kbit/s and S=10%

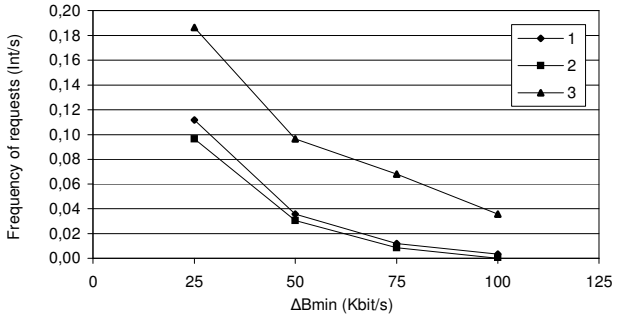

Fig. 45. Frequency of requests as a function of ΔB_{min} for the three kinds of traffic, K=600Kbit/s and S=10%

ΔB_{min} has been shown to sensibly influence the number of interactions. Results are presented in figure 26 for S=10% and can be used to choose a value for ΔB_{min} to meet the constraint on the maximum acceptable request frequency. In figure 27 ΔB_{min}=50 Kbit/s is considered, to show the influence of the margin S on average delay.

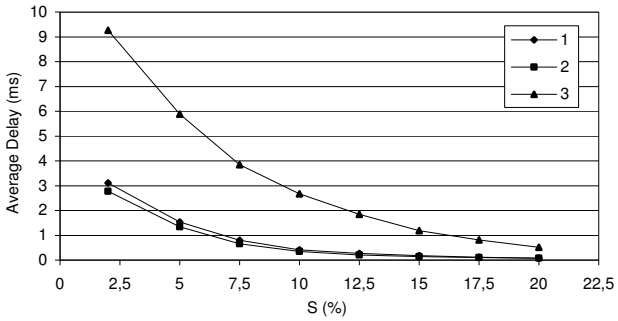

Fig. 46. Average delay as a function of the margin S for the three kinds of traffic, K=600Kbit/s and ΔB_{min} =50 Kbit/s

4.3 Implementation issues

In the previous chapter, we showed how the Click modular router was utilized for implementing flow-based classification of real time services. In paragraph §2 of this chapter, a new set of modules was then added to implement a threshold-based dynamic allocation algorithm, which was then evaluated in terms of efficiency and packet latency. Now the new proposed logarithmic algorithm was implemented on the same test bed based on the Click environment. The main difficulty was the unavailability of a function at this level that calculates the needed natural logarithmic. Different numerical methodologies have been used to achieve this target with satisfactory approximation for the whole range of argument values. Power series expression has been applied when argument was in the range near the unit where the method gives a very good approximation. Outside this range a calculation of the natural logarithm based on the expression of the base 10 logarithm as the sum of the characteristic and the mantissa and a following base conversion has been adopted. Figure 28 resumes the Click configuration used for the performance evaluations: compared to the scheme previously outlined in paragraph §2, the *LogMeterShaper* element still implements the time-window measurement algorithm (metering the amount of traffic passing through the shaper) but modulates the allocated bandwidth using the new logarithmic algorithm previously described. The *SLA Manager* is dedicated to select the traffic belonging to a set of existing Service Level Agreements, the *RTC* compound module contains the Protocol and the Statistical Classifiers to hook the real-time traffic, the *RoundRobinScheduler* is introduced to multiplex and balance the distinct traffic flows pertaining to different SLAs, while the *Priority*

Scheduler implements different policies for the real time (high-prioritized) and best effort (low-prioritized) packets.

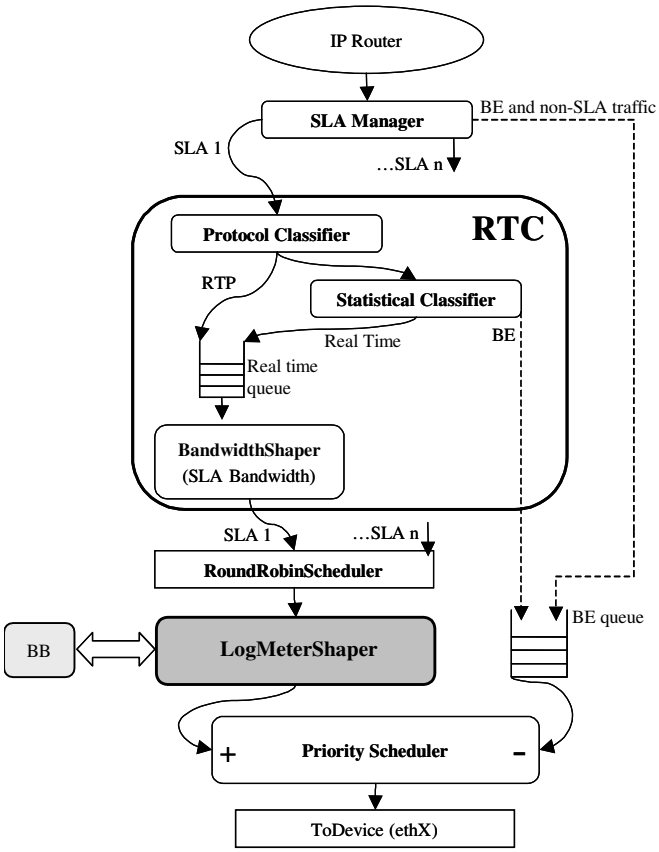

Fig. 47. The QoS router internal structure. The BB module represents the domain Bandwidth Broker (external to the Edge Router).

4.4 Performance measurements and comparison with the threshold-based algorithm

Measurements have been performed by soliciting the system through steps of bandwidth variation of an RTP flow as shown in figure 29. No best effort flow is contemporary injected, the response time of the bandwidth broker is here neglected and the result of the answer is assumed always positive. It can be seen a good tracking of the measured bandwidth obtained by the logarithmic bandwidth allocation when the step size is less or equal than 256 Kbit/s. Figure 30 compares the behaviour of the threshold-based and logarithmic algorithms with traffic variations of 512 Kbit/s.

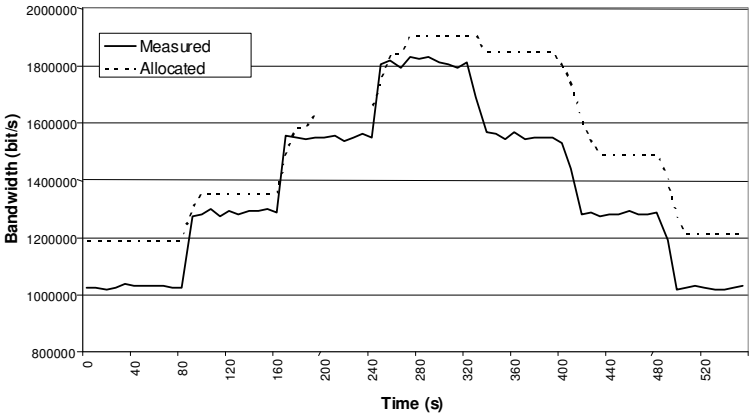

Fig. 48. Measured and allocated bandwidth in the test bed for the logarithmic algorithm with K=600Kbit/s, S=10% and ΔB_{min}=50 Kbit/s (with steps size of bandwidth variations of 256Kbit/s)

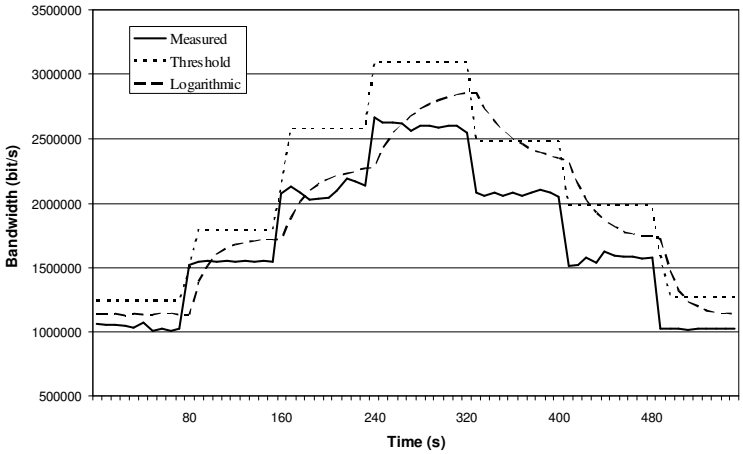

Fig. 49. Measured and allocated bandwidth in the test bed for the logarithmic and threshold-based algorithms with steps size of bandwidth variations of 512Kbit/s

It is evident that the two algorithms match the measured bandwidth differently. This causes different responses when the traffic behaviour presents sudden raisings. The threshold algorithm tends to bandwidth overprovision and this explains the lower efficiency that it presents. On the other hand the logarithmic algorithm is slower in the bandwidth update process and typically tends to loose more packets because the bandwidth is not sufficient. In figure 31 the efficiency in the bandwidth usage as a function of the bandwidth steps' size is shown in comparison with the same evaluation performed with traditional threshold algorithms (we should remember that

efficiency was defined as the ratio of the average values of the measured and allocated bandwidth).

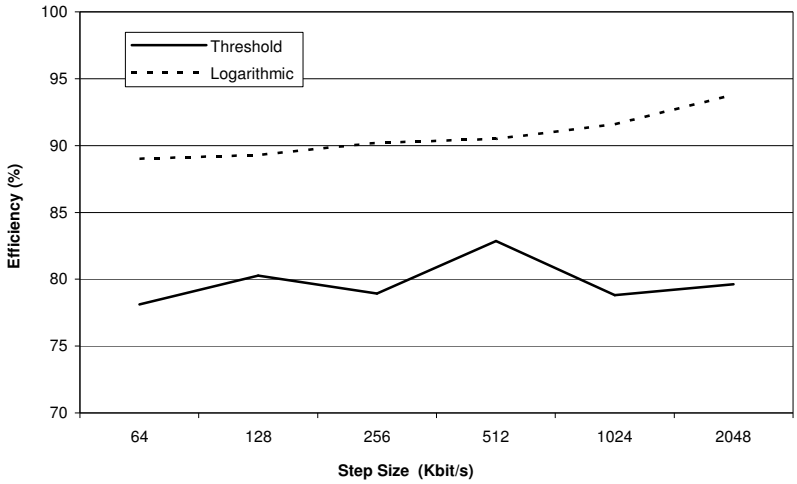

Fig. 50. Efficiency in the bandwidth usage for threshold based and logarithmic algorithms varying the step size for K=600Kbit/s, S=10% and ΔB_{min}=50 Kbit/s, as regards the logarithmic algorithm, and l_{min}=70%, l_{max}=90%,i=1.2, d=0.8, as regards the threshold algorithm

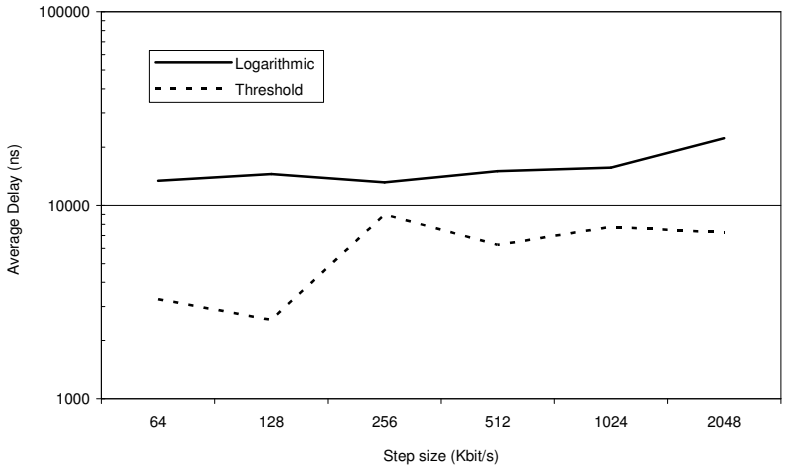

Fig. 51. Average packet latency (in nanoseconds) for threshold-based and logarithmic algorithms varying the step size and using the same configuration parameters of figure 8

In figure 32 the average packet latency is shown in comparison with the threshold–based scheme and sketches how the bandwidth constraint introduced by the *LogMeterShaper* element influences the packets' latency.

5 Conclusions

In this chapter resource management is considered for quality of service provisioning in IP network. A threshold based scheme is initially introduced and design guidelines for its parameters are given. Implementation issues are considered with particular reference to the measurement process that gives the information on which the dynamic bandwidth allocation algorithm works. Extension of the basic algorithm is proposed to deal with congestion situation that is suitable to be used with the threshold-based scheme to absorb transient overload conditions. Performance of the algorithms in terms of bandwidth usage efficiency, average and maximum delay and scalability (in terms of needed interactions with the bandwidth broken) are given in the presence of realistic internet traffic. Then, a new bandwidth allocation algorithm for Differentiated Services environment is proposed based on a bandwidth broker model. A logarithmic function is assumed to control bandwidth update. The proposed algorithm introduces a limited number of parameters that can be set up more easily then in the threshold-based system. The new design is less dependent on traffic characteristics, except for highly time variable traffic patterns.
Implementation issues are considered, with particular reference to the Click Modular Router, which offers a flexible environment for the development of new functionalities. Performance of the system in terms of bandwidth usage efficiency and latency are given. The main conclusion is that the logarithmic scheme provides a reduced transfer delay as required by real time traffic. The comparison between the two algorithms also shows how under heavy traffic conditions the packet loss could reach unacceptable values: the introduction of a congestion control mechanism also for the logarithmic function to promptly solve short-term contention in relation to delay constraints will be further discussed inside the next chapter.

Chapter References

12. Xipeng Xiao, Ni L.M: Internet QoS: a big picture. IEEE Network , Volume: 13 Issue: 2 , March/April 1999 Page(s): 8 –18
13. R. Braden, D. Clark, S. Sheneker: Integrated Services in the Internet Architecture: an Overview, IETF RFC 1633, June 1994
14. S. Blake et. Al.: An Architecture for Differentiated Services, RFC2475, December 1998.
15. K. Nichols, V. Jacobson, L. Zhang: A two-bit Differentiated Services Architecture for the Internet, IETF RFC 2638, June 1999
16. E. W. Knightly, N.B. Shroff: Admission Control for Statistical QoS: Theory and practice, IEEE Network, Vol. 13, No. 2, March/April 1999
17. A.Terzis, L. Wang, J. Ogava, L. Zhang: A Two-tier Resource Management Model for the Internet, IEEE Globecom 1999
18. E.Kohler, R.Morris, B.Chen, J.Jannotti, M.F.Kaashoek: The Click modular router. ACM Trans. Computer Systems 18, August 2000
19. R. Mameli, S. Salsano: Use of COPS for Intserv operations over Diffserv: Architectural issues, Protocol Design and Test-bed Implementation, ICC 2001, Helsinky

20. E. W. Fulp, D. S. Reeves: On line Dynamic Bandwidth Allocation, IEEE International Conference on Network Protocols, 1997
21. C.P.W. Kulatunga, P. Malone, M.O.Foghlu: Adaptive Measurement Based QoS Management in DiffServ Networks, First International Workshop on Inter-domain Performance and Simulation (IPS 2003, February 20-21, Salzburg (A)

Chapter V. Implicit Flow-based Admission and Congestion Control

1 Introduction

As described in the previous chapters, the scheme based on the Bandwidth Broker concept has been considered as suitable to cope with the Differentiated Services model proposed for QoS support. Although this approach is fairly centralized it can be made scalable through hierarchical organization of functions. This proposal splits the resource management problem into intra-domain and inter-domain functions with different administrative scopes: intra-domain resource management is within the competence of the bandwidth broker of the domain and is typically controlled by a single organization, while inter-domain resource management involves interactions between bandwidth brokers of different organizations and, on the basis of the proposal, it is achieved by bilateral agreements between adjacent domains. A critical discussion is ongoing on suitability of the proposed models to effectively support the convergence of different communication services such as voice, video and data [1]. New concepts for service differentiation are under consideration to better take into account the statistical nature of traffic as evidenced by practical measurements [2].
A still open issue is related to control signaling that impacts on user behavior and protocols to provide quality of service guarantees to legacy application programs. The strategy for graceful evolution towards fully operating QoS networks is to maintain the same user network interface to legacy application program while enhancing network intelligence to manage information flows with different requirements without the introduction of explicit signaling. This approach is referred as implicit service differentiation. The definition of the flow concept seems to easy this task [2] allowing the implementation of efficient algorithms to recognize information produced by different applications and to perform the needed QoS related functions within the network. In this chapter an implicit QoS model is proposed, suitable to manage, as an example, real time services, but that can be applied also to a larger variety of service classes. It takes advantage of a call admission procedure to limit the bandwidth requirements of a service class coupled with dynamic bandwidth management performed by the edge router within each class to optimize the bandwidth usage. No explicit signaling is required to legacy application program while the concept of flow is introduced in the edge router to implement implicit QoS concept. In order to achieve QoS targets the implicit admission control is coupled with dynamic bandwidth management algorithms, previously evaluated as stand alone procedures. The evaluation of the effects of the joint application of the admission control and bandwidth management procedures within the proposed QoS model is one of the main target of the chapter. The proposed model could cope with the differentiated service model implemented in the core, being its role played at the internetworking between edge and core networks. The role of the bandwidth broker is here played

within the core network for call and bandwidth management purposes. In any case users willing to obtain such enhanced service should declare this intention to the network and subscribe a contract in terms of Service Level Agreement (SLA) with the service provider [3]. The implementation of such a concept requires the addition of new functionalities in the edge router that interfaces peripheral QoS unaware network with the QoS enabled core network. The feasibility of the approach is proved with reference to two different service classes (i.e. best effort and real time) and their implementation in a modular software environment on a PC-based test bed. The sensitivity of the system to the main design parameters and the effectiveness of the QoS algorithms in terms of bandwidth usage efficiency and congestion limitation are obtained by practical measurements.
The chapter is organized as follows. In section 2 the general model for implicit QoS management is described with particular reference to the call admission procedures. In section 3 we recall the logarithmic-based dynamic bandwidth management algorithm, which is now modified in comparison with the description furnished in chapter §4, by adding a congestion control mechanism to promptly solve rapid rising of the used bandwidth. In section 4 system implementation and measurements are presented and discussed. In section 5 some conclusions of the work are drawn.

2 Implicit Service Differentiation and Admission Control

This paragraph introduces a flow-aware QoS model for implicit service differentiation in the Internet to maintain end user best effort interface while differentiating services within the core network. The basic operations to access QoS enabled networking in the core network are performed by the edge router (ER) at the interface between the legacy networks and the QoS domain. The end user that expects to obtain service differentiation registers a SLA at the pertinent ingress edge router before the start of any communication sessions, with off line procedures that are out of the scope of this work. After then the edge router performs two main functions related to communication sessions:

- implicit call admission control for real time flows; call admission control acts only for real time flows that belong to a SLA while best effort traffic is considered as an aggregate; implicit admission control was considered in [1];
- dynamic bandwidth management within the service class with the aim of efficient link bandwidth usage and time constraints. The access link bandwidth is assumed to be shared among different service classes. Bandwidth allocation management is performed in relation to the aggregate traffic of each class that accesses the QoS domain.

A flow is here identified by packets closely spaced in time [4]. A communication session is a sequence of flows alternating with silence periods. If a silence longer than a time out is detected, the previously related flow is considered finished.
The model can take advantage of the Bandwidth Broker concept as regard bandwidth updates required by the dynamic procedure [5]. Its action is related to core operations

and in this case, signaling within the QoS domain could be implemented through protocols like COPS [6].

2.1 Implicit call admission

Real time flows are assumed here to adopt the RTP protocol. In any case the model can suit other approaches such as native UDP flow by applying the statistical analysis for real time flow recognition available in the classification process (see Chapter §3). Under this hypothesis, a new RTP flow is recognized by the ingress edge router through a flow oriented classification process on the basis of a predefined SLA and of the SSRC field of the RTP header. The SLA contains, among its parameters, the specification of quality of service requirements through a DSCP value and the IP source address, thus allowing its correspondence with incoming RTP flows. After a new flow is recognized, it is accepted if the following condition holds:

$$B_o < T_a \quad (\mathbf{13})$$

where Bo is the estimation of the bandwidth currently used by the service class and T_a is a design parameter called admission threshold.
An accepted flow is maintained active until one of the following conditions is verified:

- a time out expires indicating that the flow has been idle for a long time (some seconds) and probably has no more information to transmit;
- $B_o > T_d$, that indicates an aggregated bandwidth usage approaching the maximum allowed for the service class. T_d is called dropping threshold. When this condition is true a random choice is performed to drop one of the admitted flows. Other choices could be studied for system optimization.

The value B_o is typically the results of a measurement procedure [7].

2.2 Dynamic bandwidth management

A service class is initially equipped with the amount of bandwidth allocated by the network manager. ERs accessing network links are responsible of monitoring bandwidth usage by the aggregate traffic of a class and of asking the related BBs for the necessary increase/decrease of allocated bandwidth; an increase/decrease request is assumed here always followed by a positive answer within a time T_{bb}. The point is to decide when to generate the requests for the broker.
A well-known approach is based on the threshold-based system explained in Chapter 4. This system has a main drawback consisting in the large number of parameters to set up for system design. Two components characterize this system: the bandwidth increment/decrement mechanism and the calculation of the bandwidth update needed. The choice of the correct parameters' configuration of the system described above is crucial to achieve bandwidth utilization efficiency, scalability and system stability. In fact, under particular conditions of traffic patterns, bandwidth oscillations could arise

as a consequence of approximate parameter configuration (see § IV.3.2). Moreover the parameters of the threshold-based model are strictly related to each other and strongly dependent on the traffic behavior thus making the parameters set up a very critical point.

3 Combining Logarithmic Bandwidth Update and Congestion Control

In order to overcome the main limitations of the threshold-based system and to reduce the number of design parameters, in the previous chapter we adopted a logarithmic function to control the update events. It uses the output of a measurement-based process to know the value B_m of the bandwidth currently used by the aggregate traffic of the service class. So B_m replaces B_o of the threshold-based scheme described before. The choice of the logarithmic function was suggested by the need to carefully increment the bandwidth, when necessary, in order to follow the bandwidth variation as better as possible and, in case of bandwidth decrease, to promptly reduce bandwidth waste. In Chapter §4, we showed that the chosen function intrinsically meets these characteristics when used to calculate the bandwidth update ΔB as

$$\Delta B = K * \ln\left(\frac{B_m}{B_d * S}\right), \quad (\mathbf{14})$$

being B_m the measured bandwidth, B_d the allocated bandwidth, and having indicated with ln(x) the natural logarithm of x. The parameter K is the constant of the feedback system and S is a margin to avoid sudden congestion. So the new bandwidth value after the update is given by

$$B'_d = B_d + \Delta B . \quad (\mathbf{15})$$

In order to avoid too frequent requests to the bandwidth broker, the update procedure is applied only if the following condition is verified:

$$\Delta B > \Delta B_{min} . \quad (\mathbf{16})$$

This system requires three parameters to be defined: K, S and ΔB_{min}. Experimental evaluations and measurements also showed that under heavy traffic conditions the packet loss percentage can reach unacceptable values: the introduction of a congestion control mechanism combined with the logarithmic function is required to promptly solve short-term contention in relation to delay constraints. In fact, due to delays in bandwidth estimate and increase, congestion can temporarily arise, that causes packets to be queued waiting for transmission resources. The effect of parameter S is to reduce the occurrence of these events, but a trade off must be reached between system performance and efficiency. So, the mechanism can be fruitfully coupled with a congestion resolution mechanism based on the monitoring of the queue occupancy, as already done for the threshold-based algorithm. The main aim of the algorithm is to limit the time needed for a packet to pass through the system by setting the maximum acceptable time T_s to empty the queue. Thus, as sketched in figure 1, a queue

threshold S_q in packets is considered given by $S_q=T_s*B_a/L_p$,being B_a the allocated bandwidth and L_p the packet length in bit. When queue occupancy S(t) at a generic instant t overcomes S_q , the system enters a congestion state during which bandwidth is quickly updated to empty the queue within T_s. The new value B_a' is calculated as $B_a'=S(t)\cdot L_p/T_s$, that assures that the queue will be emptied within the time constraint. The system stays in the congestion time for at least a guard time T_g; after then, if queue occupancy is less than S_q the regular algorithm is applied. The packet transfer delay is thus bounded by $T_s+2\cdot T_{bb}$., being T_{bb} the time required for the Bandwidth Broker to answer.

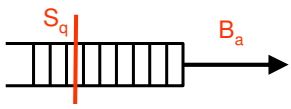

Fig. 52. The congestion control queue

For example, for a target maximum packet transfer delay of 30 ms in the router with T_{bb}= 10 ms, T_s must be limited to 10 ms. In all other cases the regular algorithm prevents congestion through S. The frequency of activation of the congestion management algorithm is related to the occurrence of the congestion state that can be reduced through the adoption of a larger guard bandwidth.

4 System Implementation and Measurements

In the previous chapters, the Click modular router was adopted for implementing flow-based classification of real time services. We recall here the basic functionalities introduced with the set of modules which were added and modified here to implement implicit admission and congestion control mechanisms. Figure 2 outlines the main components of the Click configuration:

- the *MeterShaper* element implements the time-window measurement algorithm (metering the amount of traffic passing through the shaper) that calculates the value B_o and updates the allocated bandwidth using the logarithmic algorithm previously described. Moreover, it implements the implicit admission control and the interaction with the bandwidth broker of the service domain;
- the *SLA Manager* is dedicated to select a flow as belonging to a set of existing Service Level Agreements. A time out is set to detect the end of a flow.
- the *RoundRobin Scheduler* performs multiplexing of distinct traffic flows pertaining to different SLAs;
- the *Priority Scheduler* implements different policies for the real time (high-prioritized) and best effort (low-prioritized) packets.

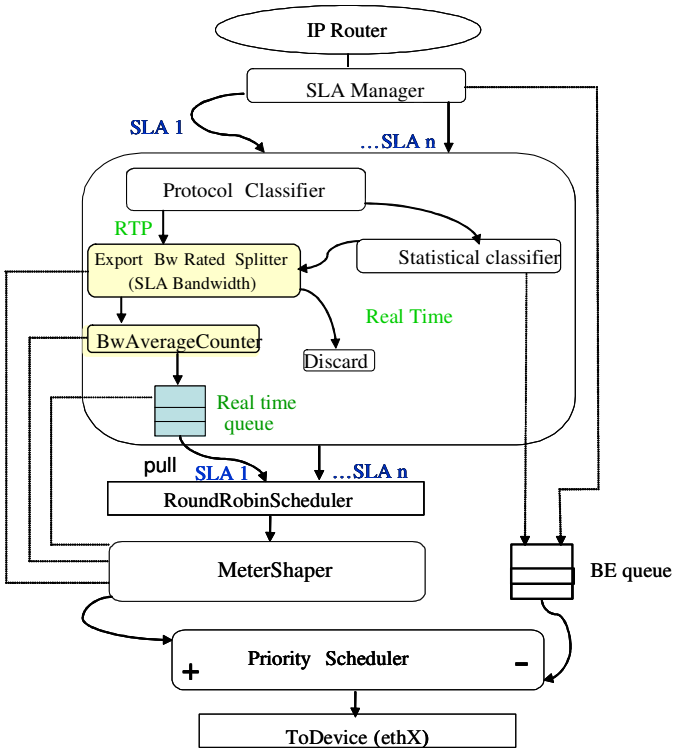

Fig. 53. QoS router internal structure

Other related functions are performed by *ExportBwRatedSplitter,* that limits the bandwidth of flows to the value specified in the corresponding SLAs, and *BwAverageCounter* that collects information about average bandwidth usage that will be processed by the *MeterShaper* element. In addition, the *MeterShaper* element is now able to lock the admission of new flows or to cancel an existing flow to implement the admission control policy previously described; finally, it monitors the state of each realtime output queue and, if congestion arises, updates the allocated bandwidth to respect the packet timing constraints.
Measurements have been performed by soliciting the system through steps of bandwidth variation of an RTP flow (figure 3). In any case, the effectiveness of this approach with other traffic types such as the superposition of Pareto sources has been verified by simulation in Chapter §4, leading to the same conditions. No best effort flow is contemporary injected, the response time of the bandwidth broker is here neglected and the result of its answer is assumed always positive. A good tracking of the measured bandwidth is obtained by the logarithmic bandwidth allocation when the step size is less or equal than 256 Kbit/s, as shown in figure 3.

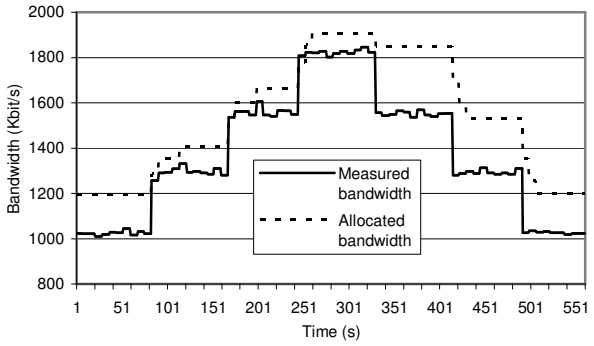

Fig. 3. Measured and allocated bandwidth in the test bed for the logarithmic algorithm with K=600Kbit/s, S=10% and ΔBmin =50 Kbit/s (bandwidth variations of 256Kbit/s)

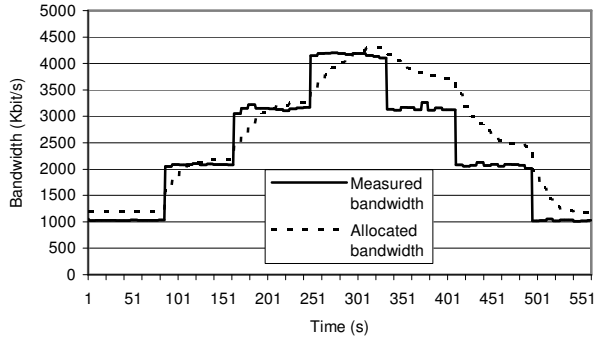

Fig. 4. Measured and allocated bandwidth in the test bed for the logarithmic algorithm with K=600Kbit/s, S=10% and ΔBmin =50 Kbit/s (bandwidth variations of 1024 Kbit/s)

When the bandwidth increase is wider, the system is no longer so prompt to follow sudden variations as reported in figure 4, for step size equal to 1024 Kbit/s. As a consequence the logarithmic algorithm typically tends to loose packets because the bandwidth is not sufficient. Figure 5 shows the improvement introduced by the congestion control algorithm with step size 1024 Kbit/s. Congestion control tends to bandwidth over provisioning, and at the same time decreases packet dropping. As a consequence efficiency reduction is expected for the largest bandwidth increases as shown in figure 6. In any case, the delay is maintained at controlled values as shown in figure 7, obtained with T_s=100 μs.

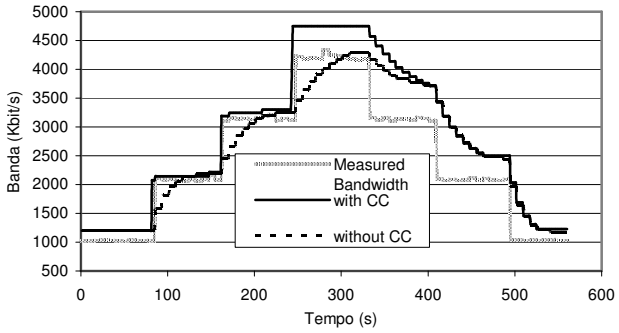

Fig. 5. Comparison of performance with and without the congestion control (CC) algorithm with K=600Kbit/s, S=10% and ΔBmin =50 Kbit/s (steps size of bandwidth variations of 1024 Kbit/s)

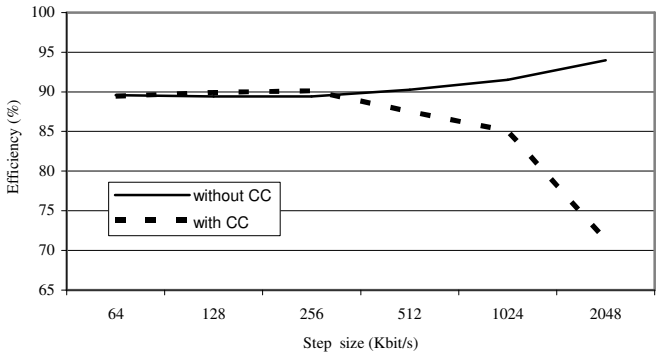

Fig. 6. Efficiency in the bandwidth usage with and without the congestion control (CC) varying the step size, with K=600Kbit/s, S=10% and ΔB_{min} =50 Kbit/s, T_s=100 μs

Figures 8-10 are related to implicit call admission control evaluation for 16 RTP flows. They have been obtained by injecting step traffic generated by 3 PCs with step size cyclically assuming the values 0, 8 16 packets/s with step duration uniformly distributed between 0 and 20 seconds and packet size of 256 bytes. Admission and dropping thresholds are given in terms of percentage of the bandwidth assigned to the real time class, which is established to 300 Kbit/s. Figure 8 depicts the timing behaviour of admitted and cancelled calls when the admission and dropping threshold are set to 90% and 95 %. The logarithmic algorithm is applied for dynamic bandwidth management with K=600Kbit/s, S=10% and ΔB_{min} =50 Kbit/s. Then, varying the admission threshold, the refused and dropped flows are plotted with dropping threshold set at 95 % (figures 9 and 10). It can be seen that the percentage of refused flows decreases as the admission threshold increases while the percentage of dropped flows increases with the admission threshold. This is due to the greater number of flows that are admitted and possibly cancelled after then.

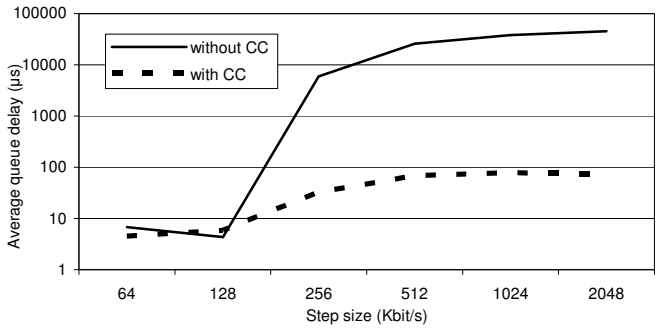

Fig. 7. Average packet latency (µs) with and without the congestion control (CC) algorithm and varying the step size, with K=600Kbit/s, S=10% and ΔB_{min} =50 Kbit/s, T_s=100 µs

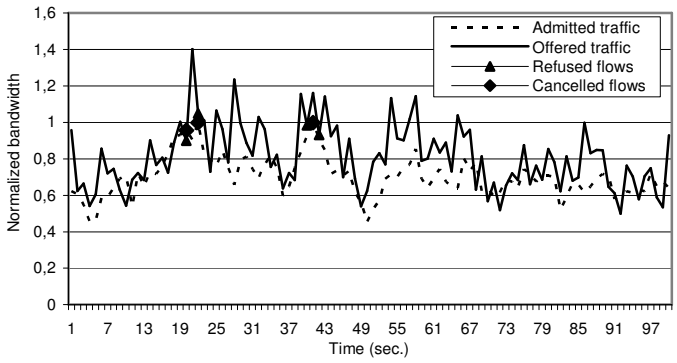

Fig. 8. Timing behaviour of bandwidth normalized to the available bandwidth Bd=300Kbit/s with admission threshold T_a =90% and dropping threshold T_d =95%

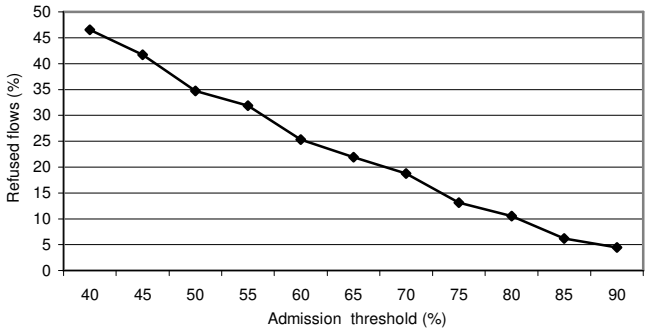

Fig. 9. Percentage of refused flows as a function of the admission threshold with dropping threshold T_d=95%

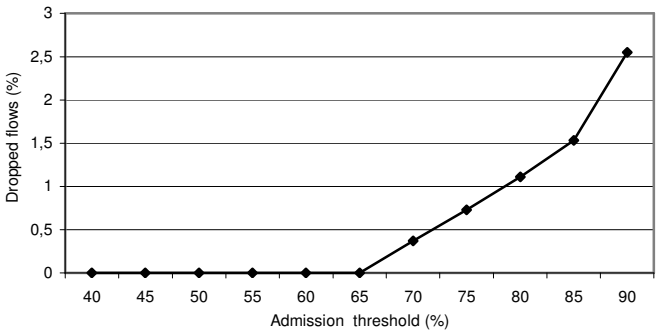

Fig. 10. Percentage of dropped flows as a function of the admission threshold with dropping threshold T_d=95%

5 Conclusions

In this chapter a model for implicit QoS support is introduced and implemented in a modular software router context. An implicit call admission procedure is designed for the edge router coupled with a dynamic bandwidth allocation algorithm that uses a logarithmic function. The proposed algorithm introduces a limited number of parameters that can be set up more easily then previously proposed systems The model achieves the target to limit the router transfer delay for real time traffic at an assigned value. The sensitivity of the system to the main model parameters is analyzed by measurements showing the choices for design optimization both in terms of usage efficiency and in terms of the number of successful flows. Moreover the field trial showed the feasibility of the introduction of new network concepts in a PC-based router for next generation Internet.

Chapter References

22. R. Mortier, I. Pratt, C. Clark, S. Crosby : Implicit Admission Control, IEEE Journal on Selected Areas in Communications, vol. 18, pp. 2629-2639, December 2000
23. A. Kortebi, S. Oueslati, J. W. Roberts: Cross-protect: implicit service differentiation and admission control, Workshop on High Performance Switching and Routing, Phoenix (USA), 2004
24. Dinesh C. Verma, “Service Level Agreements on IP Networks”, Proceedings of IEEE, Vol. 92, No 9, September 2004
25. J. W. Roberts: Internet Traffic, QoS and Pricing, Proceedings of IEEE, Vol. 92, No 9, September 2004
26. A.Terzis, L. Wang, J. Ogava, L. Zhang: A Two-tier Resource Management Model for the Internet, IEEE Globecom 1999

27. R. Mameli, S. Salsano: Use of COPS for Intserv operations over Diffserv: Architectural issues, Protocol Design and Test-bed Implementation, ICC 2001, Helsinky (SF), 2001
28. S. Jamin, P.B. Danzig, S.J. Shenker, L. Zang: A Measurement-based Admission Control Algorithm for Integrated Services Packet Networks, IEEE/ACM Transactions on Networking, Vol. 5 No 1, February 1997

Related publications

I. G.Calarco, C.Raffaelli, "Algorithms for Inter-domain Dynamic Bandwidth Allocation", IPS 2003, Salzburg (A), February 2003

II. G.Calarco, R.Maccaferri, G.Pau, C.Raffaelli, "Design and Implementation of a Test Bed for QoS Trials",QOS-IP2003, Lecture Notes in Computer Science Vol. 2601 pp.606-618, Milano (I), February 2003

III. G.Calarco, C.Raffaelli, "An Open Modular Router with QoS Capabilities",HSNMC2003, LNCS vol. 2720, Estoril (P), July 2003

IV. G.Calarco, C.Raffaelli, "Implementation of Dynamic bandwidth Allocation within Open Modular Router", ICN2004, Guadalupe (F), March 2004

V. G.Calarco, C.Raffaelli, "Design and Implementation of a New Adaptive Algorithm for Dynamic Bandwidth Allocation", HSNMC 2004, LNCS vol.3079, Toulose (F), July 2004

VI. G.Calarco, C.Raffaelli, “Implementation of implicit QoS control in a modular software router context”, QOS-IP2005, Lecture Notes in Computer Science Vol.3375 pp. 390-399, Catania (I)

VII. G.Calarco, C.Raffaelli, G.Schembra, G.Tusa, “Comparative Analysis of SMP Click Scheduling Techniques”, QOS-IP 2005, Lecture Notes in Computer Science Vol.3375, Catania (I)