Submission to:

Workshop on Resource Allocation Problems in Multimedia Systems
Held in conjunction with the 17th IEEE Real-Time Systems Symposium
December 3, 1996, Washington, DC

Paper Title:

Design and Implementation of a flexible traffic controller for ATM connections in a Multi-Tasks Operating System

Abstract

The emphasis in this paper is on the traffic control at an ATM end source. The purpose of locally controlling a source is the improvement of the TCP performances over ATM. We argue the need of a policing mechanism less stringent than the Generic Cell Rate Algorithm proposed by the ITU. A more flexible and liberal method which is suitable for providing performance guarantees in high speed networks is proposed in this paper. We present the results obtained by the implementation of this method in a multi-tasks environment. We show how we can obtain reasonable bandwidth utilization while remaining conform to the traffic contract limitations. Moreover, we present how this method achieves simplicity of implementation as well as flexibility in the allocation of bandwidth to different connections.

Corresponding Authors:

Name : **LAMTI Leila.**
E-mail : **Leila.Lamti@enst-bretagne.fr.**

Name : **AFIFI Hossam.**
E-mail : **Hossam.Afifi@enst-bretagne.fr.**

Affiliation: **Ecole Nationale Supérieure des Télécommunications - Département RSM.**

Address: **2, Rue de la châtaigneraie , BP 78, 35512 Cesson Sévigné, FRANCE .**

Phone Number: **(+33) 99.12.70.46**
New phone number after October 19: **(+33) 2.99.12.70.46**

Fax: **(+33) 99.12.70.30**
New fax number after October 19: **(+33) 2.99.12.70.30**

Design and Implementation of a flexible traffic controller for ATM connections in a Multi-Tasks Operating System

Abstract

The emphasis in this paper is on the traffic control at an ATM end source. The purpose of locally controlling a source is the improvement of the TCP performances over ATM. We argue the need of a policing mechanism less stringent than the Generic Cell Rate Algorithm proposed by the ITU. A more flexible and liberal method which is suitable for providing performance guarantees in high speed networks is proposed in this paper. We present the results obtained by the implementation of this method in a multi-tasks environment. We show how we can obtain reasonable bandwidth utilization while remaining conform to the traffic contract limitations. Moreover, we present how this method achieves simplicity of implementation as well as flexibility in the allocation of bandwidth to different connections.

1 Introduction:

High speed ATM networks in both local and wide environments start supporting per-connection end-to-end performance guarantees expressed in terms of delay, throughput and loss rate bounds [13]. Delivering this QoS to users such as computer applications or video displays (through VOD) raises however number of problems that still have to be considered.

These problems are scattered in different protocol layers ranging from the application level down to the hardware interface through the conventionnal stack of protocols. They are also caused in a different plane, by the operating system and the interaction of task scheduling with flow control.

Problems found in the protocol stack have different nature. In the application layer we face ergonomic factors like the choice of the best bandwidth for a Web session. Resource sharing among users is also an important issue in the presence of a service guaranteed network. These problems are beyond the scope of this paper. Related work can be found in [5].

Interaction of the transport layer flow control and the ATM one is also an important subject. Several contributions have for example proved that TCP protocol flow control is incompatible with ATM rate based flow controls like CBR or VBR. Proposals for new transport protocols and adjustments to existing ones can be found in [4]. Although network layer does not affect directly traffic management through specific flow control algorithms, we find in some situations a contradiction between the need to offer a guaranteed QoS and the inherent role of multiplexing usually found in this layer. A typical example of such a case can be encountered in the IP over ATM solution where TCP and UDP traffic from different applications is multiplexed on virtual channels according to their destination. In such a solution, many problems of performance guarantees can be encountered. In fact, at the IP level, packets from different connections will interact with each other; without proper control, these interactions may adversely affect the QoS guarantees desired by each source.

On the other hand, the congestion control mechanism implemented within ATM networks rely on admission control, resource reservation and scheduling schemes [11]. Resource reservation schemes manage the allocation of the resources at each of the nodes so that per-node QoS requirements can be met for each connection. With only these control mechanisms, QoS requirements can't be satisfied. This is due to the fact that the users may attempt to exceed the QoS limits specified at the connection establishment within the traffic contract. In fact, it is to be noted that since users (e.g. workstations) have a physical access offering a bandwidth much larger than what they probably will "buy" in terms of traffic contract parameters, it is an important task to control at the user side that no violations are committed before data can be sent to the network. The need of traffic enforcement and shaping at the edges of the network become consequently important. We propose in this paper a solution to the implementation of traffic shaping at the user's side in order to check that their flow's characteristics remain conform to the declared values throughout the duration of the connection. The rest of this paper is organized as follows:

In Section 2, we delineate the TCP over ATM environment and its requirements. A comparision between the window-based and the credit-based approaches is given. We show in this section the necessary TCP extension to enhance the application performances in terms of traffic enforcement. A solution consisting in a controller offering both traffic shaping and connection scheduling is then presented.

In Section 3, we first describe the controller's architecture. We then present the GCRA scheme and show its negative impact on a source traffic if implemented in a multi-tasks environment as the regulator's algorithm. We discuss the need of a less stringent policing mechanism which allows short term burstiness and show how it can result in a considerable performance improvement while policing a source traffic. A qualitative means of defining a packet-to-cell conversion is discussed. Its purpose is to ensure the efficiency of the control mechanism when implemented at the packet level. The principles of such a conversion and its effect on the credit calculation are finally described.

We present in Section 4 the controller's implementation and discuss its ability to smooth the burstiness of a source input traffic while respecting the bounds of the traffic contract. Moreover, we show how it can be considered equivalent to a leaky bucket. A comparison is made in this section between the performances obtained with the GCRA and with the proposed controller. Finally, Section 5 summarizes and concludes this paper.

2 The TCP over ATM environment:

Refering to the IP over ATM working group recommendations [14], ATM networks should provide a class of service which strives to cooperate with existing TCP congestion avoidance mechanism, thereby, explicitely providing support not only for directly-attached and aware endstations, but also for endstations on LANs that are using current TCP implementations. ATM alternative as a new support for computer communications has demonstrated some trials for the implementation of new transport protocols over native ATM [3], [6] and [8]. The main reason for these trials is that current TCP/IP fails to exploit ATM benefits (high bandwidth, QoS guarantees ..). A dual stack using either TCP/IP or native ATM transport protocol stacks was also considered. A different approach consists in keeping current TCP and IP protocols and internally modify them to deal with native ATM connections [1]. The advantage of this solution is that it necessitates a minimal number of modifications in existing applications. Users will see a gradual and smooth migration of their applications to a full native ATM stack. The proposed modifications concern the TCP congestion control and avoidance mechanisms. The main characteristics of these mechanisms are briefly outlined below.

2.1 The window-based versus the rate-based flow control:

The TCP congestion control mechanism is window-based. Two algorithms are applied in order to implement a congestion control at the TCP level. With the *Slow-Start Algorithm,* the window size doubles at each transmision cycle[1]. The *Congestion Avoidance Algorithm* makes the window size increase by one segment at every transmission cycle. The transition from the *Slow-Start* phase to the *Congestion Avoidance* one happens when the actual size reaches the current threshold. When the maximum window size is reached, the transmission continues without further increasing the window size [2]. The source considers the loss of a segment as a congestion indicator and reacts reducing the transmission window size to one segment, the threshold is set to half the current window size and the *Slow-Start* phase is entered. Several studies [4], [9] have demonstrated that using TCP with its current congestion control mechanisms would always lead to an under-utilization of the high speed network channels. The aim of this paper is not the proposal of a new TCP-like protocol for high speed networks, but the presentation of modifications introduced on TCP in order to support a traffic control mechanism. Additionnally, our solution provides a traffic shaping at the source edges (per-application).

1. The transmission cycle designates the sequence of events corresponding to the transmission of a number of segments equal to the current window size and to the reception of the corresponding ACK(s).

2.2 The rate-based mechanism:

In ATM networks, at call setup time, a user requesting a rate controlled connection specifies a desired average service rate. If the call is accepted, the network guarantees bandwidth. The specification of the connection parameters constitutes a traffic contract used by the ATM switches to apply a control mechanism called *Usage Parameter Control* (UPC). The ATM cells are checked at each node. If a cell is conforming to the contract, it is routed towards its destination. If it is not, it may be tagged or destroyed. Since TCP congestion control mechanism is independant of these contracts, non conforming cells may be sent. This incoherence results in a performance degradation of the transmission rate. We propose a solution in which connections are policed using a modified version of the *Generic Cell Rate Algorithm GCRA* proposed by the ITU. It is a rate-based policing which can provide a reasonable compromise between the bandwidth utilization and the performance guarantees. Moreover, the traffic control applied to a given data flow affects all connections multiplexed into a unique stream. Hence, flow control has to be applied before multiplexing. This consideration has been taken into account in our design of the controller described in the following section.

3 The controller's architecture:

Conceptually, we have divided the controller into two components: a rate regulator and a scheduler (Figure 1). The regulator shapes the input traffic on each connection into the desired traffic pattern. The rate regulator consists in a set of sub-regulators corresponding to each of the connections traversing the TCP layer. The regulation is done via a leaky bucket controlled window coupled with a modified version of the GCRA. The regulators achieve the control before handing the input traffic to the scheduler.

The scheduler is thought to have a real-time and a non-real-time queues. A connection is considered to be real-time if it negotiates a CBR traffic contract. The VBR connections are considered non-real-time. The most common First Come First Served (FCFS) discipline will be used for both queues.

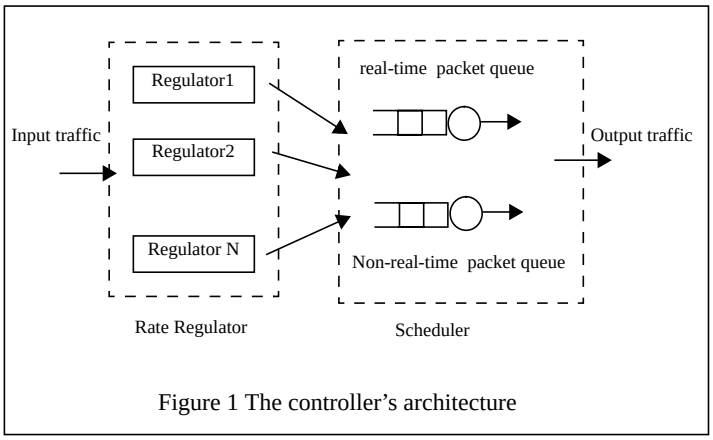

Figure 1 The controller's architecture

3.1 The regulator's design:

3.1.1 Using the GCRA algorithm:

The GCRA is defined with 2 parameters: the Increment (I) and the Limit (L). It can be a virtual scheduling algorithm or a continous-state leaky bucket algorithm [12]. The virtual scheduling algorithm updates a Theoretical Arrival Time TAT, which is the "nominal" arrival time of the packet assuming that the active source sends equally spaced packets. If the actual arrival time of a packet is not "too" early relative to the TAT, in particular if the actual arrival time is after TAT-L, then the packet is conforming; otherwise, it is non conforming. We have used this version of the GCRA as a principle in the design of the regulator's traffic shaping. It achieves the control by assigning each packet an Eligibilty Time (ET) upon its arrival and holding the packet till that time (ET) before giving it to the

scheduler. For our implementation model, we have adapted the DARPA formula [13] concerning the modelisation of a service discipline for high speed networks switches to our environment characteristics. We have in fact taken into account the principle of the GCRA algorithm (the virtual scheduling algorithm) to adapt this formula. If we consider a regulator based on the $(Xmin, Xave, I)$ set of parameters (as explained below), the eligibility time of the k^{th} packet, designated by $ET(k)$, is defined according to the eligibility times of the preceeding packets of the same connection by:

$ET(k) = -I$ if k<0

$ET(1) = AT(1)$

$$ET(k) = Max\left(ET(k-1) + Xmin, I + ET\left(k - \frac{I}{Xave}\right)\right) \quad (1)$$

With:

X_{min} = The minimum packet inter-arrival time.

X_{ave} = The minimum average packet inter-arrival time.

I = The averaging interval over which X_{ave} is computed.

The X_{min} parameter is computed according to the PCR, CDVT, SRC or (and) BT parameters depending on the type of the traffic contract. The X_{ave} parameter is calculated according to an image of the packet arrival in the past. With such an expression, we adapt the control to the source's behaviour.

We have first implemented the regulator using these expressions. We have remarked that this method induces a connection performance degradation independently of the network behaviour. This constatation was made in [10]. In fact, it has been shown that a stringent input rate control may unnecessarily increase the end-to-end delay by a significant amount which degradates the whole performances [10]. The GCRA introduces access delays which can become prohibitive for the connections. On the other hand, with such an approach, especially in the context of multi-tasking non real-time OS, we have noted that the granularity of the process blocking function at the kernel level may be inaccurate in certain situations and time is lost at each blocked packet. Moreover, because of the kernel scheduling, the considered process passes into the *sleeping state* (at each blocking*)* waiting for a timer to expire. At the arrival of the eligibility time (expiration of the timer), it moves to the *runnable state.* Because of the context-switching, paging and swapping, a delay is introduced when a process is ready to run again [16]. This leads to a delay between the real activation and the effective one [7]. This delay becomes prohibitive for the transmission rate if the blocking frequency increases. We have led an experiment to determine the performances of this method. The results are shown is Section 4.3.

In order to remedy to these problems, and with an aim at reducing the access delays, we have developped a traffic shaper which performs like a Leaky Bucket over long durations, but allows short burstiness. The improvement that we are searching for is a better control of periodical burstiness and the increase of the statistical multiplexing gain at the network access.

3.1.2 Using the credit-based approach:

The equivalence of the virtual scheduling algorithm and the leaky bucket was demonstrated in [12]. We have adapted the definition of the leaky bucket to our implementation environment in order to make it less stringent and more flexible.

The original version of the leaky bucket algorithm *LB(I,L)* can be viewed as a finite-capacity bucket whose real-valued content drains out at a continous rate of 1 unit of content per time-unit and whose content is increased by the increment I for each conforming packet. If at a packet arrival, the content of the bucket is less than or equal to the limit value L, then the packet is conforming, otherwise, the packet is not conforming. The capacity of the bucket is upper bounded by *L+I.* Our approach is less stringent because it attributes an amount of tokens to the connection for a certain period *TT.* The credit-counter is at its maximum value at the beginning of the period. Each accepted packet consumes a token. When the bucket becomes empty, the whole connection is then blocked (no packets are accepted) until the beginning of the next period. So, we limit the number of packets that can be sent over an interval TT but we don't control the spacing between 2 successive packets. A source can emit the maximum number of packets over a duration less than TT but can't exceed the maximum amount of allowed packets.

A credit amount is attributed to each connection. Its value depends on the source traffic type (CBR, VBR, ABR....). In the next section, we will present the formula used to derive the credit expression for each connection type. As the regulator is periodic, the attribution of the credit amount will be done at the beginning of each new period. On the other hand, the blocking time is expressed by:

$$B(k) = TT - Ta(k) \tag{2}$$

Where:

$B(k)$ = The blocking time of the k^{th} packet.
TT = The period of the controller.
$Ta(k)$ = The arrival time of the k^{th} packet.

3.1.2.1. The credit calculation:

The conformance to a specific leacky bucket LB(T,τ) can be controlled by maintaining an image of the credit counter. We can notice using the traffic management specifications document [12] that over any closed interval of length t, the number of cells, N(t), that can be emitted with spacing no less than T and still be in conformance with LB(T,τ) is bounded by:

$$N(t) \leq min\left(\left\lfloor 1 + \frac{t+\tau}{T} \right\rfloor, \left\lfloor 1 + \frac{t}{T} \right\rfloor\right) \tag{3}$$

We consider that the rate of a connection is defined over an averaging interval TT. The rate is determined by dividing the total number of bytes transmitted during this interval. The value of TT determines the allowed burstiness at a source. The larger this interval is, the higher the allowed burstinees is. We have estimated TT by experimentation trials. A one second TT value was found to be optimal for the current platform. A more generic method is currently being studied for TT estimation. Relying on these considerations, the calculation of the credit for each connection is made according to these formula :

For VBR sources:

* *If TT >=MBS*T :*

$$Credit = \left\lfloor 1 + \frac{TT + BT}{Ts} \right\rfloor Size \tag{4}$$

* *If TT < MBS*T :*

$$Credit = \left\lfloor 1 + \frac{TT}{T} \right\rfloor Size \tag{5}$$

with:

BT : Burst Tolerance.
$Ts = 1/(SCR)$ where SCR designates the Sustainable Cell Rate of the source.
$Size$: An ATM cell size (53 bytes).
MBS : Maximum Burst Size.
$T = 1/(PCR)$ where PCR is the Peak Cell Rate of the connection.

For CBR sources:

Since our controller is implemented at the packet layer, a CBR source is considered as a VBR one at the TCP level. This is due to the fluctuations generated by the decomposition of a TCP packet into ATM cells. So, even if the BT and the SCR parameters are not defined for such connections within the traffic contract, we define them for our credit calculation. We use (4) and (5) to calculate the credit of such connections. The definition of these parameters is explained in the following section.

3.1.2.2. The bursty generation effect at the TCP level:

The traffic shaping principles proposed by the credit-based approach conditions a ***cell*** input stream. The control is based on the traffic contract parameters expressed in terms of number of cells and their inter-arrival time throughout the life time of the connection. As our implementation takes place at the TCP layer, we have thought to adapt our control and the parameters used in the credit calculation to the control of the burtiness caused by the decomposition of a TCP packet into a set of ATM cells. A VBR traffic control must be done for all the connections. In order to apply (4) and (5), we have to define for each type of source the parameters SCR and BT. For VBR sources, these parameters are contract defined. For CBR sources, only the Peak Cell Rate (PCR) and the Cell Delay Variation Tolerance (CDVT) parameters are defined. We have assumed that BT corresponds to the CDVT parameter and so:

$$BT = PCR \times CDVT$$

To define the SCR parameter for this type of sources, we have used this expression proposed in the ATM forum standard document [12]:

$$MBS = 1 + \frac{BT}{Ts - T} \quad \text{where} \quad Ts = \frac{1}{SCR} \quad \text{and} \quad T = \frac{1}{PCR}$$

The MBS designates the maximum burst size which represents the maximum number of cells that can be emitted at the PCR rate without being out-of-band. As CBR sources can emit at the PCR rate for all the connection time, we have transformed this parameter into the maximum number of cells that can be emitted within the interval TT which designates the period of the controller. So we have :

$$Ts - T = \frac{BT}{MBS - 1}$$

which leads to:

$$Ts = T + \frac{BT}{MBS - 1}$$

If we replace MBS, Ts and T by their expressions as a function of SRC and PCR, we have:

$$\frac{1}{SCR} = \frac{1}{PCR} + \frac{BT}{TT \times PCR - 1}$$

Finally, we have:

$$SCR = \frac{PCR}{1 + \frac{BT}{TT - \frac{1}{PCR}}} \tag{6}$$

3.2 The scheduler's principle :

The role of the scheduler becomes important if the system load increases so greatly that the station can not satisfy the whole needs of the connections. In such a situation, the packets are put onto the network with a certain delay because of the multiplexing of the different data flows. The scheduler's purpose is to introduce a set of parameters assigning 2 priority levels to the connections. The packets coming from the different connections are organized into 2 queues : a real-time queue and a non-real-time queue. For CBR connections which support real-time applications requiring tightly constrained delay variations, we steer the packets towards the real-time queue. For the other connection types, we direct them to the non-real-time queue. The real-time queue has the highest priority. The scheduler services the packets using a non-preemptive FCFS discipline. The non-real-time packets are serviced only if

there are no real-time packets. We have choosen the FCFS discipline because of its simplicity of implementation which is an important advantage in the context of high speed environment. The scheduler performance under heavy load conditions is subject to outgoing research.

4 Implementation:

4.1 Environment:

As mentionned before, a multi-tasks environment supports our implementation. It consists in a SOLARIS operating system running over a SPARC 20 station. SOLARIS is a system V Release 4 OS offering the Streams environment. TCP is implemented as a module within the ATM communication architecture. A stream consists in a full duplex connection made between a device driver monitoring the ATM card and the stream head supervising the interface with the user (Figure 2). For each connection established at the application level, an specific data structure is interpreted as the traffic contract of the source. These information elements are transfered by the stream head which is represented by the Transport Layer Interface (TLI) system routines down to the TCP layer [15]. Since a TCP connection corresponds to a kernel process in the system context, each source can be considered as a unique TCP process We can then ensure the independency between the connections since we implement the control before the IP multiplexing.

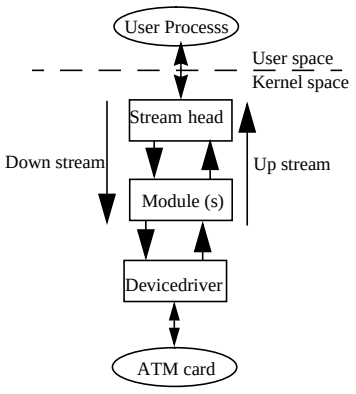

Figure 2 The streams architecture

On the other hand, since the implementation is supported by a multi-tasks OS, we have to take into account the properties of this environment to ensure good performances of our controller. Within the SOLARIS operating system, the processes are scheduled leading to costly context switches [7]. The TCP process is then subject to the scheduling rules that are applied within the kernel and its performances are consequently affected by the background load supported by the system. Our implementation has taken place at the kernel space.

4.2 The controller algorithm:

The steps of the controller algorithm are:

1) At the first packet arrival, we calculate the credit allocated for the connection according to its type and to its traffic contract parameters. The credit consumed is initialized to zero. A timestamp indicates the beginning of a period.

2) At each packet arrival:

2.1) If a new period begins, the credit consumed is reinitialized to zero and the available credit is set to the maximum value calculated in the step (1). A new timestamp is relieved indicating the beginning of another new period.

2.2) If sufficient credit is available for the packet, we consume an amount proportionnal to the packet size and the packet is passed down through the streams architecture.

2.3) If no suffisant credit is available, we block the current process till the arrival of the next period. The blocking function suspends the calling process for an amount of time expressed in terms of absolute clock ticks since the last system reboot. At the expiration of the timer, the process becomes runnable and the packet is sent down. A timestamp is relieved to indicate the beginning of a new period and the credit consumed is set to zero.

4.3 Results:

A series of experiments has been made on a client-server architecture. All clients are supposed to have very long data transfer to a server. In order to compare the behaviour of the sources in the two cases when they are controlled and uncontrolled, we have activated two client processes simultaneously : the first supporting the control and the second without any rate constraints. In regulated mode, the client specifies its traffic contract via an interface and the parameters choosen are taken into account by the controller. The results of these experiments are reassembled in (Figure 3). We remark that with the controller, the source has been limited to its allowed traffic rate. On the other hand, we notice an unsteady traffic if no control is made due to a lot of factors (system load, number of active connections,...). A best-effort capability is offered to users is this case.

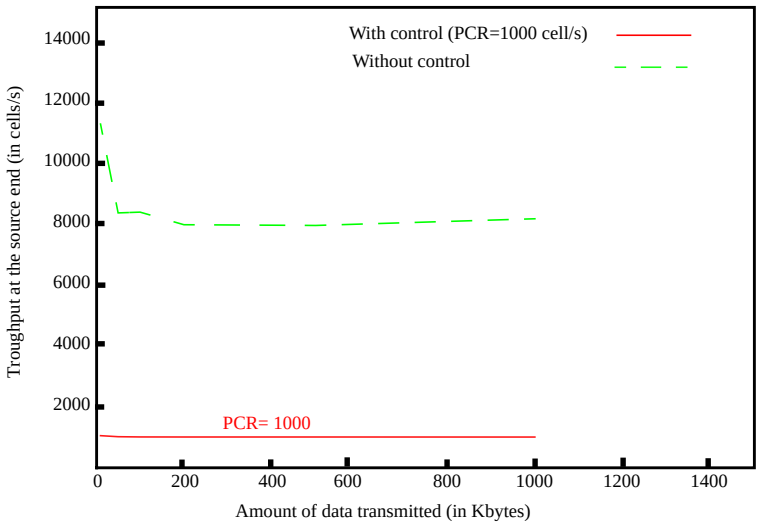

Figure 3 Controlled and uncontrolled behaviours

A second experiment is shown in (Figure 4). It consists in comparing the performances of the GCRA algorithm with those of our control algoritm. Since the GCRA is based on the control of the packet inter-arrival time, for each non conforming packet, the kernel blocking function is used to enforce the connection to reduce its transmission rate. We can notice in (Figure 4) that when only two sources are emitting simultaneously, the performances of the two algorithms are nearly equivalent. When we multiply the system load by increasing the number of simultaneous connections, we note a degradation of the perfomances with the GCRA approach. This is due to the kernel scheduling and to the context switching. The blocking frequency is higher with GCRA. This leads to an excessive access delay which induces a performance drop. Since the operating system is not a real-time one, the accuracy reached has limits that can be prohibitive in such cases.

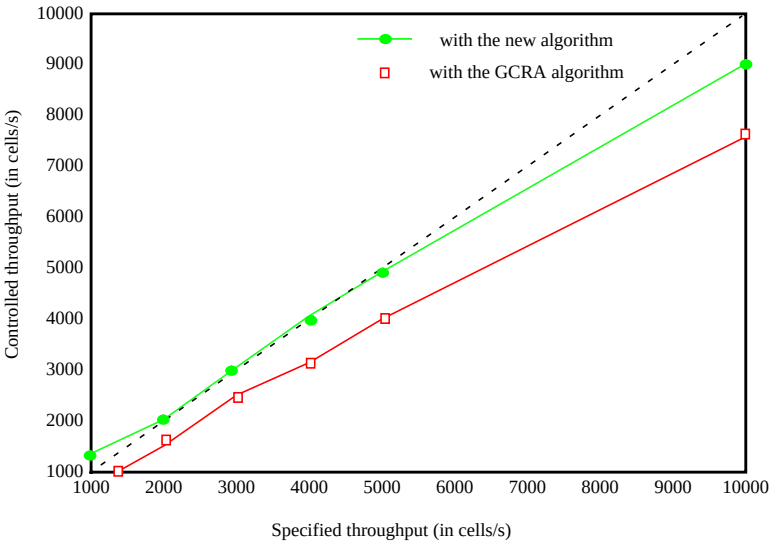

Figure 4 Comparison between the GCRA and the new algorithm

The curve in (Figure 5) shows the amount of time lost at each packet blocking depending on the number of active connections. For several connections, an amount of time which reaches the maximum value of 1.4 milliseconds is measured between the theoretical activation time of the process after a blocking caused by a non-conforming packet and the real activation time.

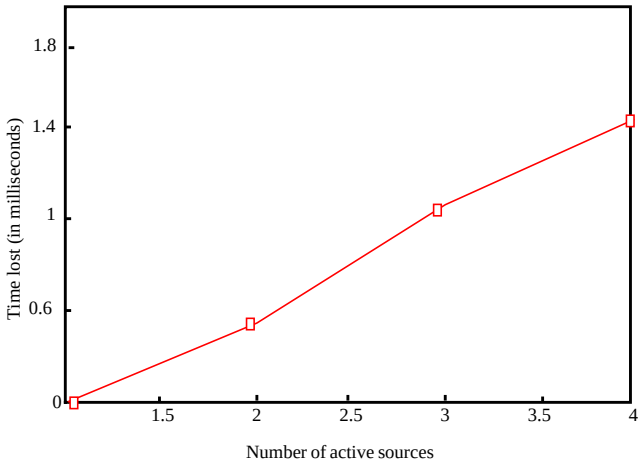

Figure 5 Time loss with GCRA function of connection number

So, we can notice in (Figure 6) the impact of the system load increase and the time loss on the bandwidth offered by the GCRA. We have activated simulataneously 4 sources having a bandwidth allocation of 5000 cell/s. Since the kernel blocking function becomes innacurate, we remark a drop in the transmission rate with that control. In fact, the frequency of process blocking is higher with the GCRA approach than with our controller's alogrithm.

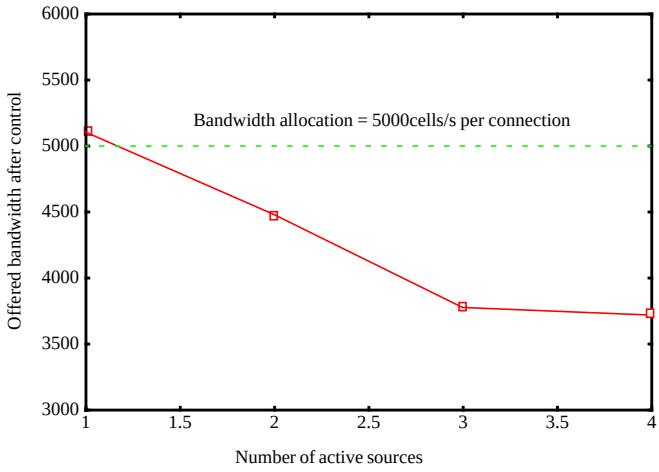

Figure 6 Bandwidth offered by the GCRA function of connection number

5 Conclusion:

In this paper, we have presented the design and the implementation of a traffic controller for applications using a IP over ATM environment. It improves the traffic shaping and the connection scheduling at the source edges. A comparative study between the performances of the GCRA as described by the ITU and those of the proposed controller's algorithm is also delineated. We show the results obtained by both algorithms when implemented in a multi-tasks environment. A performance comparison shows that the stringency of the GCRA leads to excessive delays before data can be sent to the network. We show the adaptability of our algorithm to the traffic control purpose in a multi-tasks environment. A qualitative means of making a packet-to-cell conversion is explained. Such a conversion is needed since the implementation is done at the TCP level. With this traffic shaping offered at the source edges, we can ensure guaranted QoS for applications. The traffic shaper and the switches will collaborate to provide the required performances and good network channels utilization. We will investigate in a future work on the

choice of the bandwidth allocation for each application in order to ensure ergonomic presentation (e.g WEB sessions) and to give the required performance guarantees. Resource sharing among users is also an important issue in the presence of a guaranted network.

6 References:

[1] H.Afifi, D.Bonjour, O.Elloumi : *TCP over Non Existing IP For ATM Networks- JECN'7 Budapest May 1996.*

[2] A.Ajmone Marsan, A.Bianco, R.Lo Cigno, M.Munafo : *Shaping TCP Traffic in ATM Networks - ICT'95 Bali Indonesia April 1995.*

[3] R.G. Cole, D.H. Shur, C.Villamizar : *IP over ATM* - A Framework Document , Internet-Draft.

[4] O.Elloumi, H.Afifi, P.Rolin, M.Hamdi : *Issues in Improving TCP performance over ATM - Proceedings of the IFIP ATM workshop on traffic management WATM'95 December 1995 - Paris.*

[5] M.W. Garrett : *A Service Architecture for ATM : From Applications to Scheduling - IEEE Network - May-June 1996.*

[6] D.D. Kandlur, D.Saha, M.Willebeek-LeMair : *Protocol Architecture for Multimedia Applications over ATM networks.* Computer communication review Volume 25 Number 3 July, 1995.

[7] S.Khanna , M.Sebrée , J.Zolnowsky : *real-time Scheduling in SunOS 5.0.* In Proceedings of Usenix 92-Winter 1992.

[8] S.Keshav, M.Hill : *Semantics and Implementation of a Native-Mode ATM Protocol Stack.*

[9] M.Perloff, K.Reiss : *Improvements to TCP Performance in High Speed ATM Networks.* Communications of the ACM. Feb 1995 Vol 38-2.

[10] S.Radhakrishnan, S.V.Raghavan, A.K.Agrawala : *A flexible Traffic Shaper for High Speed Networks: Design and Comparative Study with Leaky Bucket - Computer Networks and ISDN systems 28 (1996) 453-469.*

[11] S.Radhakrishnan, S.V.Raghavan : *Network Support for Distributed Multimedia* - Issues and Trends, SEACOMM'94.

[12] S.Sathaye : *The ATM forum Traffic Management Specification, version 4.0 ATM Forum/95-0013R6 April 96.*

[13] H.Zhang, D.Ferrari : *Rate-Controlled Service Disciplines. Journal of high speed networks 3 (1994), 389-412.*

[14] IP over ATM working Group : *Recommendations for the ATM Forum's Multiprotocol BOF*.

[15] SOLARIS Streams Programmer's Guide (2.4) - SunSoft.

[16] SunOS 5.3 System Services (Book) - Process Scheduler - Performance.