9

Design and implementation of a flexible traffic controller for ATM connections[1]

L. LAMTI, H. AFIFI, M. HAMDI
Ecole Nationale Supérieure des Télécommunications de Bretagne, RSM Department
2, Rue de la Châtaigneraie, BP 78, 35512 Cesson Sévigné, France
Phone: +33 02 99 12 70 46 / Fax : 33 02 99 12 70 30
E-mail:{Leila.Lamti, Hossam.Afifi, Maher.Hamdi}@enst-bretagne.fr

Abstract

In ATM networks, the congestion control paradigm is built on two basic principles. The first states that the adequacy of a service delivered to a particular user should not depend on the detailed behaviour of other users. The second is that network should provide enough feedback to users to effectively utilize the available resources. Traffic management is concerned with controlling the source behaviours to ensure they get their desired Quality of Service (QoS). It includes the Connection Acceptance Control (CAC) and the Usage Parameter Control (UPC). In order to avoid data loss at the UPC, we propose to implement a traffic controller at the source edge. With its two components, - a traffic regulator and a connection scheduler-, this controller shapes the input traffic so that it becomes amenable to the policing mechanisms. The purpose of locally controlling a source is the improvement of the TCP performance over ATM networks. We argue that the policing mechanism used in the regulation component should be less stringent than the Generic Cell Rate Algorithm (GCRA) proposed by the ITU. The control method proposed in this paper is more flexible and is suitable for providing performance guarantees in high speed networks. Moreover, a quantitative means of defining a packet-to-cell parameters conversion is discussed. Its purpose is to ensure the efficiency of the control mechanism when implemented at the packet level. We present the results obtained by the implementation of this controller in a multi-tasks environment. We show how we can obtain reasonable bandwidth utilization while remaining conform to the traffic contract limitations. Finally, we present how this method achieves simplicity of implementation as well as flexibility in the allocation of bandwidth to different connections.

Keywords

ATM traffic contract, Traffic shaping, Leaky Bucket algorithm, Packet-to-cell parameter conversion.

[1] This work was partially funded by a CNET-Lannion, France Telecom Grant No:94PE7301.

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 (Zhang and Ferrari, 1994). Delivering this Quality of Service (QoS) to users in computer applications such as desktop video services raises however a certain 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 conventional stack of protocols. They are also caused by the operating system and the interaction of task scheduling with flow control.

Problems found in the protocol stack are of different kinds. In the application layer, we face ergonomic choices like the necessary bandwidth needed for a Web session. Resource sharing among users is also an important issue in a guaranteed service network. These problems are beyond the scope of this paper. Related work can be found in (Garrett, 1996).

Interaction of the transport layer flow control and the ATM one is still an open issue. 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 (Elloumi, et al., 1995). Although network layer does not directly affect 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 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 guarantee 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, congestion control mechanisms implemented within ATM networks rely on admission control, resource reservation and scheduling schemes (Radhakrishnan and Raghavan, 1994). 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 users may attempt to exceed QoS limits specified at the connection establishment within the traffic contract. In fact, it is to be noted that users (e.g. workstations) have a physical access offering a bandwidth much larger than what they may choose in terms of traffic contract parameters. So, it is an important task to control at the user side that no violations are committed before data is sent to the network. The need of traffic enforcement and shaping at the edges of the network becomes consequently important. We propose in this paper a solution to implement traffic shaping at the user's side in order to check that their flow's characteristics remain conform to the traffic contract.

We also present the principle of a packet-to-cell parameter conversion used in our implementation in order to take into account the fluctuations resulting from the fragmentation of a TCP packet into ATM cells.

The rest of this paper is organized as follows:

In Section 2, we delineate the TCP over ATM environment and its requirements. We show in this section what must be changed in TCP in order 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 introduced.

In Section 3, we first describe the controller's architecture. We then present the Generic Cell Rate Algorithm scheme and show its negative impact on a source traffic if implemented in a multi-tasks environment. 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. We present a solution using a credit-based scheme to implement the traffic shaping. A quantitative means of defining a packet-to-cell parameter 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 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. 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

2.1 Requirements:

Referring to the IP over ATM working group recommendations (IP-over-ATM WG, 1995), "ATM networks should provide a class of services which strives to cooperate with existing TCP congestion avoidance mechanisms, thereby, explicitly providing support not only for directly-attached and aware endstations, but also for endstations on LANs that are using current TCP implementations".

It is becoming more common that large ATM networks provide virtual channels for the IP layer services in addition to native ATM services. In fact, the ATM alternative as a new support for computer communications has demonstrated some trials for the implementation of new transport protocols over native ATM (Cole, et al., 1995). The main motivation behind these trials is that TCP/IP fails to exploit ATM benefits (high bandwidth, QoS guarantees..). Some of these proposals consider also the use of a dual stack (either TCP/IP or native ATM transport protocol) (Almesberger, 1996) that starts a connection using conventional TCP/IP and switches if possible to an ATM native connection.

This paper presents a different approach that keeps the current TCP/IP protocol stack. We introduce however some necessary modifications to deal with native ATM connections. The advantages of this solution and additional protocol design details can be found in (Afifi, et al., 1996).

2.2 The self-regulation mechanism

In ATM networks, a call setup is always associated with a set of parameters specifying a service rate (even if no service rate is required by the application). If the call is accepted, the network guarantees the bandwidth. It also verifies the non-violation, by the user, of the connection parameters. This constitutes a traffic contract used to apply a control mechanism called *Usage Parameter Control* (UPC). ATM cells for a given connection are checked. Conforming cells are routed towards their destination. Others are usually discarded.

TCP protocol implements a specific congestion control mechanism designed for packet networks with no bandwidth or delay guarantees (Jacobson, 1988). This algorithm has been progressively improved (Matthis, et al., 1996) and is perfectly designed for communications needing a tight closed loop flow control. TCP/IP can be used for ATM networks with the help of the "Classical IP over ATM" framework (Laubach, 1994). In this case, it is recommended to use a best-effort connection. It is possible to offer a global guaranteed rate to all of the TCP/IP traffic (with Classical IP over ATM) going from a given source to a destination by pre-establishing the necessary virtual circuit with QoS guarantees. It is however not possible to establish separate TCP/IP connections each with a different guaranteed traffic contract. The reason comes from 1) TCP/IP flow control that acts separately for each connection. Connection starting late (a situation where start time arrives when other connections are already in an advanced Slow-Start region) or subject to losses will negatively affect the corresponding throughput 2) The global pre-established virtual circuit controlling the multiplexed IP traffic that is not capable of dealing with separate connections.

We propose to adapt the TCP/IP environment to deal with ATM requirements. By forcing self-regulation for each source, we avoid the loss of non-conforming packets through the ATM network and the unfairness resulting from the separate, independent closed loop flow control.

3 THE CONTROLLER'S ARCHITECTURE

As shown in (Figure 1), we have divided the traffic controller into two components: a traffic regulator and a source scheduler. The purpose of such a decomposition is to offer an independent regulation for each connection according to its traffic parameters. Thus, we immunize the bandwidth allocation from the effect of end-to-end delay variation among all the sources. The regulator shapes the input traffic on each connection into the desired traffic pattern. The regulation is performed by a flexible traffic shaper using a credit-based scheme. The regulators achieve the control before handing the input traffic to the Scheduler. The Scheduler is composed of real-time and non-real-time queues (Operating system Streams deal with pointers only, no copies are made within the kernel). A connection is considered to be real-time if it negotiates a CBR or VBR traffic contract.

The UBR connections are considered non-real-time. The most common First Come First Served (FCFS) discipline will be used for both queues. The role of the Scheduler appears only when the host's load prevents from fullfilling all the required QoS guarantees.

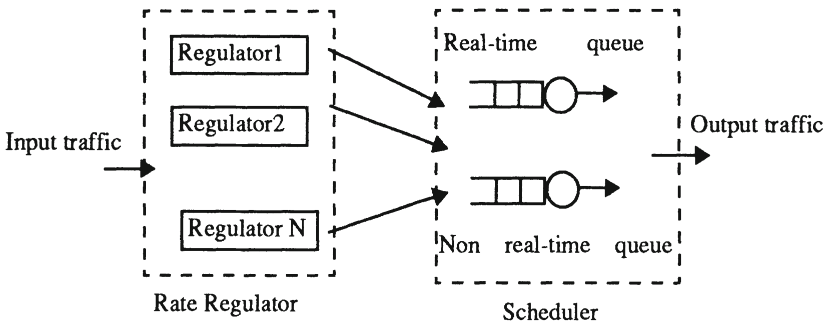

Figure 1 The controller's architecture.

3.1 The regulator's design

The GCRA scheme

Data flow generated by a source is examined within the UNI/NNI according to the Generic Cell Rate Algorithm (GCRA) (Sathaye, 1996). An equivalent traffic shaping must be performed at the source edge. We first present the principle of this algorithm and then explain how it can be considered too stringent if applied as a traffic shaping scheme.

The GCRA is defined with 2 parameters: the Increment (*I*) and the Limit (*L*). It can be a virtual scheduling algorithm or a continuous-state leaky bucket algorithm (Sathaye, 1996). 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. The proposed regulator achieves the control by assigning each packet an Eligibility Time upon its arrival (*ET*) and by holding (the packet is actually blocked in a queue and hence prevents any further activity in the application) the packet till that time (*ET*). For our implementation model, we have used results proposed by a study in (Zhang and Ferrari, 1994) concerning a service discipline model for high speed networks switches.

Let (X_{min}) and (X_{ave}) be the minimum packet inter-arrival time and the minimum average packet inter-arrival time respectively. If we consider a regulator based on the (X_{min}, X_{ave}, I) set of parameters, the eligibility time of the *k*th packet, $ET(k)$, is defined according to the eligibility times of the preceding packets of the same connection by:

$$ET(k) = -1 \quad \text{if} \quad k \le 0$$

$$ET(k) = Max\left(ET(k-1) + X_{min}, I + ET\left(k - \frac{I}{X_{ave}}\right), AT(k)\right) \qquad (1)$$

Where:
I = The averaging interval over which (X_{ave}) is computed.
$AT(k)$ = Arrival time of packet (k).

The (X_{ave}) is the minimum average packet inter-arriuval time. It is computed according to an image of the packet arrival in the past. With the (X_{ave}) parameter, we enable the end-system to describe its future flow in greater details than just the peak rate (within the (X_{min})). With (1), we can smooth the source flow to the $(Xmin, Xave, I)$ model and so, enhance resource utilization. This method chooses the maximum last cycle and the current busy period for averaging the current (*ET*). The (X_{min}) parameter is computed according to the (*PCR*), (*CDVT*), (*SCR*) and (*BT*) parameters included in the traffic contract. Details on the computation of (X_{min}) are presented in the following sections.

Limitations of the GCRA in a multi-tasks operating system and motivation for a new scheme

The first regulator's implementation relied on the GCRA algorithm. We have noticed that it could hardly generate the required throughput and remarked a semi-permanent under-utilization of the reserved bandwidth leading to a performance degradation independently of the network behaviour. This conclusion was jointly made in (Radhakrishnan, et al., 1996). 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 degrades the overall performances (Radhakrishnan, et al., 1996).

Moreover, the GCRA introduces access delays which can become prohibitive for the connections. Since each packet must wait until the arrival of its eligibility time, we had to use kernel blocking function. In the context of a multi-tasks operating system used for this implementation, we have noted that the granularity of the blocking function at the kernel level was inaccurate in certain situations. So, time was lost at each blocked packet arrival. Moreover, at each non-conforming packet, the connection passes into a *sleeping state* (due to the blocking function) 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 between the time the connection is ready to run and the time it is really scheduled as confirmed in (Khanna, et al., 1992). This delay becomes prohibitive for the transmission rate if the blocking frequency increases. The performance of this method was measured in different scenarii. The results are shown is Section 4.3.

In order to remedy to these problems and to reduce the access delays, a second approach was adopted. We have designed a traffic shaper conformant to a Leaky Bucket (LB) over long durations, allowing however periodic short burstiness.

The proposed scheme: A credit-based approach

The equivalence of the virtual scheduling and the leaky bucket algorithm was demonstrated in (Sathaye, 1996). 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 continuous rate of 1 unit of content per time-unit and increases by *(I)* for each conforming packet arrival. 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 naturally blocked (since no packets are accepted) until the beginning of the next period. As mentioned before, this approach may introduce a burst effect within *(TT)* if packets inter-arrival time is very small. This effect can be reduced by reducing *(TT)*.

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 in terms of the known ATM-Forum traffic management specifications. As the regulator is periodic, the attribution of the credit amount will be done at the beginning of each new period.

The credit calculation

The conformance to a specific leaky bucket *LB(I,L)* can be controlled by maintaining an image of the credit counter. According to the traffic management specifications document (Sathaye, 1996), *N(t)*, the number of packets that can be emitted with spacing no less than *(I)* and still be in conformance with *LB(I,L)* over any closed interval of length *(t)* is bounded by:

$$N(t) \leq min\left(\left\lfloor 1 + \frac{t+L}{I} \right\rfloor, \left\lfloor 1 + \frac{t}{I} \right\rfloor\right) \qquad (2)$$

In our approach, 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. We consider that the rate of a connection is defined over an averaging interval *(TT)*. It is determined by dividing the total number of bytes transmitted during this interval. 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. It is blocked when its token bucket is empty until the beginning of the next period. The value of *(TT)* determines the allowed burstiness at a source. The larger this interval is, the higher the allowed burstiness will be. 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:

$$1)\ If\quad TT \geq MBS \times T \quad then \quad Credit = \left\lfloor 1 + \frac{TT + BT}{Ts} \right\rfloor \tag{3}$$

$$2)\ If\quad TT \geq MBS \times T \quad then \quad Credit = \left\lfloor 1 + \frac{TT}{T} \right\rfloor \tag{4}$$

Where:

BT = Burst Tolerance (in seconds).

$Ts = 1/(SCR)$ where *(SCR)* designates the Sustainable Cell Rate of the source.

MBS = Maximum Burst Size (in cells).

$T = 1/(PCR)$ where *(PCR)* is the Peak Cell Rate of the connection.

Equations (3) and (4) are conform to the ATM traffic management specifications (Sathaye, 1996).

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 (3) and (4) to calculate the credit of such connections. The definition of the *(SCR)* and the *(BT)* parameters is explained in the following section.

The packet-to-cell parameter conversion:

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 we implement the controller in the TCP module, we have thought to adapt our control and the parameters used in the credit calculation to the control of the burstiness caused by the fragmentation of a TCP packet into ATM cells. The motivation is based on the fact that if a non conforming cell is dropped at the UNI , the whole packet is damaged and can not be reassembled by the destination host. We propose a solution which is based on the choice of the *(BT)* parameter. This parameter must be chosen so that fluctuations within the cellular traffic can be absorbed inside the TCP packet. Let *(r)* designate the TCP traffic rate (in bit/s) and *(b)* the burst tolerance (in bit) both corresponding to a packet scale leaky bucket parameters. Consider periodical arrival of TCP packets, i.e. one packet of variable length is generated each period (τ). Packet scale conformance to a leaky bucket of parameters *(r)* and *(b)* is equivalent to :

$$A(m) \leq rm\tau + b \tag{5}$$

Where *A(m)* is the amount of data generated during *(m)* consecutive packets. At the fluid scale, let *N(s,t)* be the amount of data generated in the interval *[s,t]*. Conformance to a GCRA controller operating at the UPC involves *N(s,t)*. The GCRA can then detect violations even if the source remains conform to (5). This means that packet scale conformance does not necessarily ensure cell-scale conformance if the same parameters of the leaky bucket are used. To avoid this situation, the choice of parameter *(BT)* used in the GCRA becomes important. The analysis that follows determines the *(BT)* and *(SCR)* parameters in function of *(r)* and *(b)* values.

Let *R(t)* be the instantaneous TCP rate expressed in bits/s. The quantity of data emitted within an interval *[s,t]* is defined by:

$$N(s,t) = \int_s^t R(t)dt \tag{6}$$

If *(n)* designates the smallest integer verifying $s <= n\tau$ and *(m)* the highest integer that verifies $m\tau <= t$, then we have:

$$s \le n\tau \le m\tau \le t \tag{7}$$

$$n\tau - s \le \tau \tag{8}$$

$$t - m\tau \le \tau \tag{9}$$

And (6) becomes:

$$N(s,t) \le \int_s^{n\tau} R(t)dt + \int_{n\tau}^{m\tau} R(t)dt + \int_{m\tau}^{t} R(t)dt$$

The expression of the middle designates *A(m-n)* and is bounded by (5), we have:

$$N(s,t) \le b + r\tau(m-n) + \int_s^{n\tau} R(t)dt + \int_{m\tau}^{t} R(t)dt$$

If *(p)* designates the peak rate of the source in bits/s, we have:

$$\int_s^{n\tau} (R(t)-p)dt \le 0 \quad \text{and} \quad \int_{m\tau}^{t} (R(t)-p)dt \le 0$$

which is equivalent to:

$$\int_s^{n\tau} R(t)dt \le p(n\tau - s) \quad \text{and} \quad \int_{m\tau}^{t} R(t)dt \le p(t - m\tau)$$

So, we have:

$N(s,t) \leq p(n\tau - s) + p(t - m\tau) + r\tau(m-n) + b$

Using (8) and (9), we obtain:

$$N(s,t) \leq r(t-s) + 2\tau(p-r) + b \qquad (10)$$

With such an expression, we can conclude that the conformity at the packet level can be ensured if we choose:

$BT = b + 2\tau(p-r)$ and $SCR = r$

The parameter definition

In order to apply (3) and (4), 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 *(PCR)* and the Cell Delay Variation Tolerance *(CDVT)* parameters are defined. Since a CBR connection sends at the peak cell rate for all the connection time, then the *(BT)* parameter can be supposed equivalent to the *(CDVT)* one. Within the *(CDVT)* duration, we have a tolerance for cell clumping equal to :

$$BT = CDVT \times PCR$$

To define the *(SCR)* parameter for this type of sources, we have used this expression proposed in the ATM forum standard document (Sathaye, 1996):

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

The *(MBS)* parameter designates the maximum burst size. As CBR sources can emit at the *(PCR)* rate for all the connection time, we have transformed this parameter into the maximum number of bits that can be emitted within the interval *(TT)* (due to the tolerance admitted in the inter-arrival time). So we have:

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

If we replace *(MBS)*, *(Ts)* and *(T)* by their expressions as a function of *(SCR)* and *(PCR)*, we have:

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

Finally, we have:

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

To recapitulate:

* For CBR sources, let *(CDVT)* and *(PCR)* be the 2 parameters specified in the traffic contract at establishment time. Since we consider the connection as bursty at the packet level, we suppose that we have 2 auxiliary parameters considered as a Burst Tolerance *(BT(c))* and a Sustainable Cell Rate *(SCR(c))* defined as follows:

$$BT(c) = CDVT \times PCR \qquad \text{and} \qquad SCR(c) = \frac{PCR}{1 + \frac{BT(c) \times SCR}{TT - \frac{1}{PCR}}}$$

To implement the control at the packet level, we have choosen for the LB algorithm *(LB(1/SCR(p), BT(p)))* these parameters:

$$BT(p) = BT(c) - 2(p - SCR(p)) \quad \text{and} \qquad SCR(p) = SCR(c)$$

* For VBR sources, let *BT(c)* and *SCR(c)* be the traffic parameters choosen by the connection which will be used by the UNI to perform the control. Then, we choose to implement the control at the packet level with *LB(1/SCR(p), BT(p))* where *BT(p)* and *SCR(p)* are defined as follows:

$$BT(p) = BT(c) - 2(p - SCR(p)) \quad \text{and} \qquad SCR(p) = SCR(c)$$

For the other types of traffic (ABR and UBR), we have left the calculation for a further study.

3.2 The scheduler's principle

The role of the scheduler becomes important if the system load increases so greatly that the endstation 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 and VBR connections which support real-time applications requiring tightly constrained delay variations, we steer the packets towards the real-time queue. For the ABR and UBR connections, 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 chosen 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, a 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.

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 independence between the connections since we implement the control before the IP multiplexing. 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 process scheduling leads to costly context switches (Khanna, et al., 1992). The TCP process depends on scheduling rules that are applied within the kernel and its performances are consequently affected by the background load supported by the system.

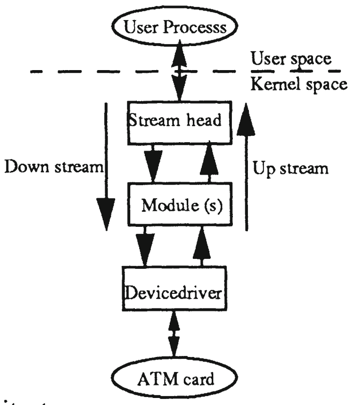

Figure 2 The streams architecture.

4.2 The controller's 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, to its traffic contract parameters and to the packet-to-cell parameter conversion principle. 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 to indicate the beginning of another new period.

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

2.3. If insufficient credit is available, we block the current process till the arrival of the next period. 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).

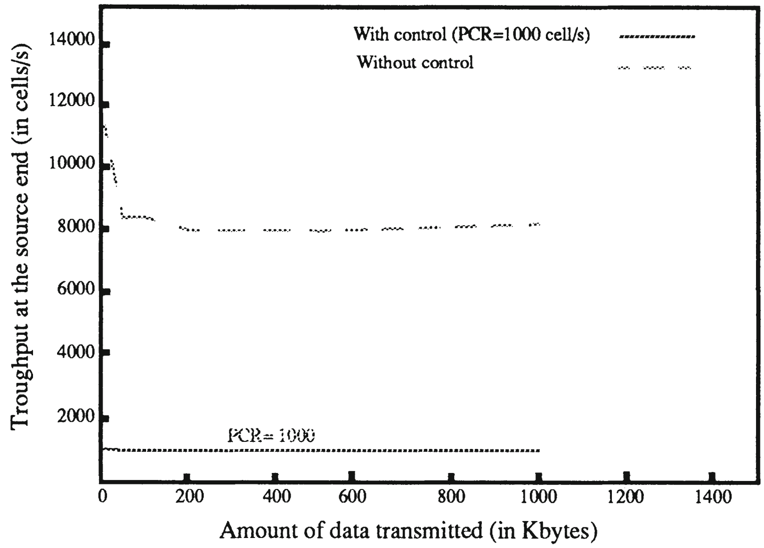

Figure 3 Controlled and uncontrolled behaviours.

A second experiment is shown in (Figure 4). It consists in comparing the performances of the GCRA algorithm with those of the proposed 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 2 sources are emitting simultaneously, the performances of the 2 algorithms are nearly equivalent. When we multiply the system load by increasing the number of simultaneous connections, we note a degradation of the performances 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 has limits which is prohibitive in such cases.

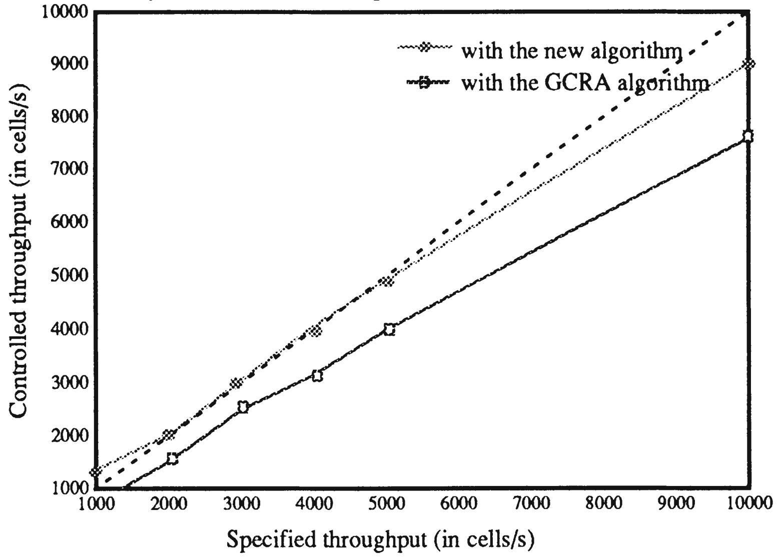

Figure 4 Comparision between GCRA and the new algorithm.

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.

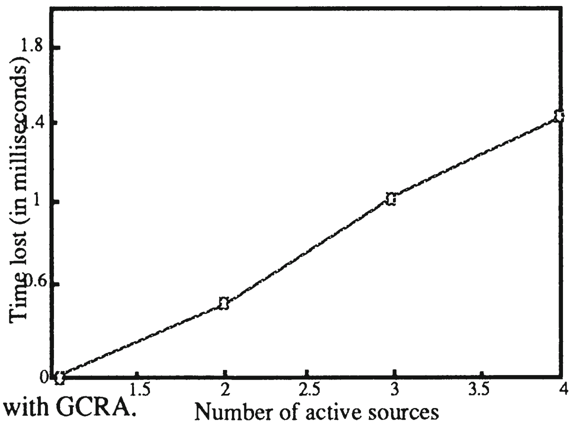

Figure 5 Time loss with GCRA.

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 simultaneously 4 sources having a bandwidth allocation of 5000 cell/s. Since the kernel blocking function becomes inaccurate, we remark a drop in the transmission rate with such a control.

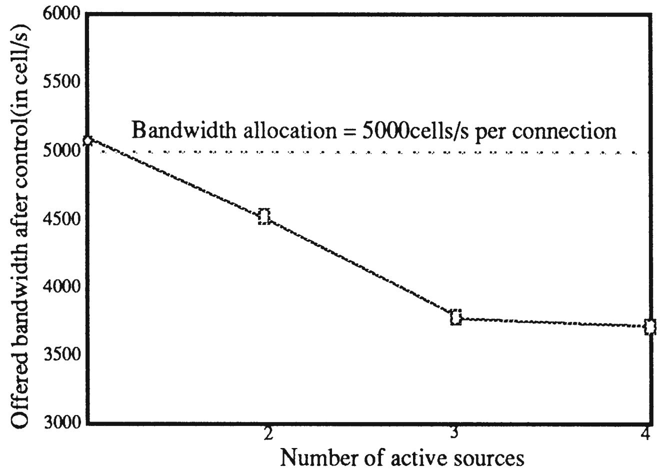

Figure 6 Bandwidth offered with GCRA.

5 CONCLUSION

In this paper, we have presented the design and the implementation of a traffic controller for applications using an IP over ATM environment. It achieves traffic shaping and 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 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 the proposed algorithm to the traffic control purpose in a multi-tasks environment. A quantitative means of making a packet-to-cell parameter 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 guaranteed 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 guaranteed network.

6 REFERENCES

• (Afifi, et al., 1996)
Afifi, H., Bonjour, D., Elloumi, O., (May 1996). *TCP over Non Existing IP For ATM Networks- JECN'7 Budapest.*
• (Almesberger, 1996)
Almesberger, W., (1996). *Arequipa: Design and Implementation, ftp://lrcwww.epfl.ch/pub/arequipa/aq_di-1.tar.gz, Technical Report 96/213. Nov 96*
• (Marsan, et al., 1995)
Marsan, A., Bianco, A., Lo Cigno, R., Munafo, M. (April 1995). *Shaping TCP Traffic in ATM Networks In proceedings ICT'95 Bali Indonesia.*
•(Cole, et al., 1995)
Cole, R.G., Shur, D.H., Villamizar, C., (April 1995). *IP over ATM* - A Framework Document , Internet-Draft.
•(Elloumi, et al., 1995)
Elloumi, O., Afifi, H., Rolin, P., Hamdi, M. (December 1995). *Issues in Improving TCP performance over ATM. In Proceedings of the IFIP ATM workshop on traffic management WATM'95 Paris.*
•(Garrett, 1996)
Garrett, A.W., (May-June 1996). *A Service Architecture for ATM : From Applications to Scheduling. IEEE Network, May-June 1996.*
•(Jacobson, 1988)
Jacobson, V., (1988). *Congestion Avoidance and Control. In the proceedings of SIGCOMM'88 Symposium, pages 314-32, Aug. 1988.*
•(Kandlur, et al., July 1995)
Kandlur, D.D., Saha, D., Willebeek-LeMair, M., (July 1995). *Protocol Architecture for Multimedia Applications over ATM networks. CCR Volume 25 Number 3 July, 1995.*
•(Khanna, et al., 1992)
Khanna, S., Sebrée, M., Zolnowsky, J., (1992). *Real-time Scheduling in SunOS 5.0. In Proceedings of Usenix 92-Winter 1992.*
•(Keshav and Hill, 1995)
Keshav, S., Hill, M., (1995). *Semantics and Implementation of a Native-Mode ATM Protocol Stack.*
• (Laubach, 1994)
Laubach, M., (1994). *Classical IP and ARP over ATM. RFC-1577, IETF 1994.*
• (Matthis, et al., 1996)
Mathis, M., Mahdavi, J., (1996). *Forward Acknowledgement: Refining TCP Congestion Control. ACM Sigcomm 1996 Proceedings 281-291.*
•(Perloff and Reiss, 1995)
Perloff, M., Reiss, K., (1995). *Improvements to TCP Performance in High Speed ATM Networks. Communications of the ACM. Feb 1995 Vol 38-2.*
•(Radhakrishnan, et al., 1996)

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

•(Radhakrishnan and Raghavan, 1994)

Radhakrishnan, S., Raghavan, S.V., (1994). *Network Support for Distributed Multimedia - Issues and Trends, SEACOMM'94.*

•(Sathaye, 1996)

Sathaye, S., (1996). *The ATM forum Traffic Management Specification,version 4.0 ATM Forum/95-0013R6 April 96.*

•(Zhang and Ferrari, 1994)

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

•(IP-over-ATM WG, 1995)

IP over ATM WG (1995). *Recommendations for the ATM Forum's Multiprotocol BOF.*