Time Slotted Moving Window - a New Policing Mechanism for High Speed Networks

Marcio Faerman *Célio Vinicius N. de Albuquerque*

Otto Carlos M. B. Duarte

Grupo de Teleinformática e Automação - GTA
COPPE/EE - Prog. de Eng. Elétrica/Dept. de Eletrônica
Universidade Federal do Rio de Janeiro
CP 68504 - CEP 21945-970 Rio de Janeiro, RJ
E-mail: {faerman,celio,otto}@coe.ufrj.br

Abstract

This paper describes a window-based policing mechanism, called Time Slotted Moving Window. Its key idea consists in aggregating the arrival events computation in order to minimize the processing and buffering requirements. Its main characteristics are simplicity, low implementation complexity, memory saving and use of a single low precision timer. Its performance behavior ranges from jumping window to sliding window results. Simulation and implementation results considering geometric and bursty traffics are obtained for comparison purpose. The comparative analysis concentrates on the number of timers and their precision, the maximum burst duration, the required memory and the violation probability. Furthermore, the mechanisms were implemented in UNIX and DOS environment using a high performance implementation architecture. The results confirm proposed mechanism efficiency.

1 Introduction

Broadband Integrated Services Digital Network (B-ISDN) aims to be a general purpose network that supports real time, interactive, broadcast, connection oriented and no connection oriented services provided to constant or variable bit rate traffics. The task of this network is to meet not only the present applications requirements but also the future applications ones, that were not conceived yet. Under these circumstances, the flexibility must be privileged to fulfill this great challenge.

B-ISDN must support efficiently the highest number of users and enforce the transport of the contracted traffic. Quality of Service (QoS) parameters as cell loss rate, delay, jitter delay, etc., must be defined, monitored and enforced by the network.

In the last years a great progress in B-ISDN research has taken place, however many open issues remain waiting a satisfactory solution. Undoubtedly traffic management of B-ISDN is one of the most complex problems to be solved.

To guarantee a B-ISDN good performance is necessary a set of traffic control mechanisms. There are several kinds of control such as connection admission control, priority control, congestion control, and usage parameter control.

Connection admission control is related to the conditional admission of a new connection by the network. The admission decision is made based on the availability of enough resources to meet the required quality of service without affecting the quality of service of other established connections. Concerning priority control, different actions are taken by the network to process the flow of a connection that contains cells of different priorities, indicated by the cell loss priority bit. Furthermore, the congestion control is associated to the state of network elements in such a way that a traffic overload does not affect the quality of service contracted by the established connections.

This work focuses on the usage parameter control that is related to the actions taken by the network, during the information transfer phase in order to monitor and enforce the user traffic. The main objective of this control is the protection of network resources against abusive traffic sources. These traffic sources violate the parameters established during connection admission phase and can also affect the quality of service of other connections. The mechanisms used in this control have the function of detecting a possible parameter contract violation. These mechanisms are also known as policing mechanisms.

Several mechanisms have been proposed. Among the most mentioned mechanisms in literature there are the leaky bucket [2,3], its variants [4], and window-based mechanisms. Among these last ones it can be pointed out the jumping window, the sliding window and the exponentially weighted moving average [2,5,6].

Among policing mechanisms one that presents simultaneously advantages referent to the traffic memorization, the mean bit rate control in addition to the restriction of the burst length, is the sliding window mechanism. However this mechanism has been criticized because of its implementation complexity [2,5]. Dittmann *et al* [6] proposes an implementation scheme that requires a complexity level not so critic.

This paper proposes and analyzes the time slotted moving window mechanism, a window-based policing mechanism that requires an implementation complexity inferior to sliding window mechanism complexity and that is so restrictive as this in relation to burst duration. This mechanism also has the advantage of being extremely flexible. According to user and network contract, the time slotted moving window can be configured ranging from jumping window simplicity to the wide set of advantages offered by sliding window.

In section 2 the jumping window and sliding window mechanisms are described along with their worst case traffics. Section 3 proposes the time slotted moving window mechanism and identifies its worst case traffic. Section 4 describes the simulation models and the mechanisms implementation procedures. Section 5 presents the numeric results obtained in simulation and implementation and realizes a comparative analysis of the mechanisms. Finally the conclusions are presented in section 6.

2 Window-Based Policing Mechanisms

Window-based mechanisms limit to a particular value N (maximum number of tokens), the number of cells submitted to the network during a time interval T. This time interval is called **window**. The

differences among window-based mechanisms reside in the way of how window progresses along the time and how the token value is considered.

The traffic fraction that exceeds token limit, in a particular window, is called violator traffic and its composing cells are simply discarded or marked. In this situation, if marked cells do not cause congestion to network they are transferred, possibly overtaxing users. This work considers that only non-violator traffic is submitted to the network.

Jumping Window

The Jumping Window (JW) mechanism considers windows of fixed time periods equal to T, juxtaposed along the time. Therefore, a window starts immediately after the end of the previous one. During a window, only N cells can be submitted to the network. If N cells have already been received by the network during a window, the succeeding cells will be considered violators and under this circumstance they will be dropped or marked.

This mechanism is very simple and can be implemented with a timer and a counter. The juxtaposed windows are achieved by a timer on its auto-firing configuration that restarts the timer at the moment that it finishes its cycle. The counter must be initialized with token limit N and decremented for each cell submitted to the network (figure 1).

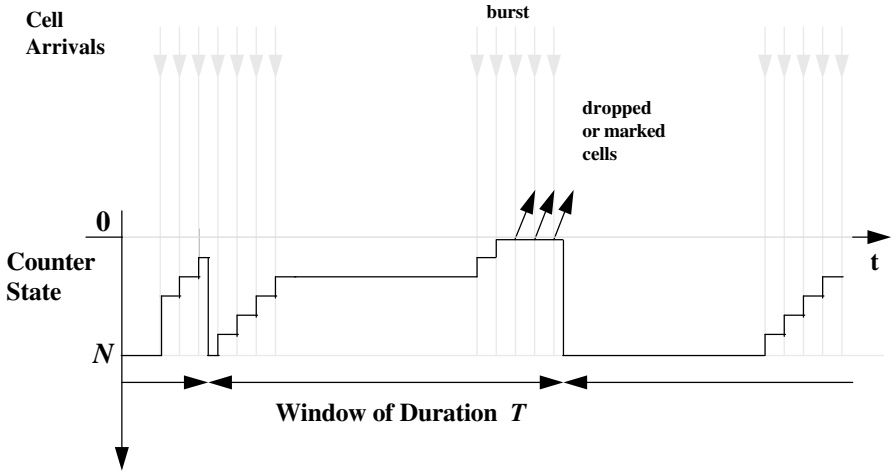

Figure 1: The Jumping Window Mechanism.

A traffic pattern used in policing mechanisms analysis is called worst case traffic. Worst case traffic is defined as the traffic that passes through the mechanism entirely exploiting the policed rate without suffering any kind of punishment in addition to having characteristics that difficult its transit in the network. The most severe condition for the network occurs during burst periods because cells may be buffered (causing delays) and even more, they can be lost as a consequence of memory overflow depending on network condition. Therefore, it can be concluded that worst case traffic has a burst/silence pattern where burst length is as long as possible.

If the source synchronizes to JW window then it can concatenate two bursts of length N at the end of a window interval and at the beginning of the next. A silence period of $2T$ - ($2N \times t_{cel}$) is required, where t_{cel} denotes the time interval between cell arrivals during a burst. Therefore, this pattern has a $2T$ period (figure 2).

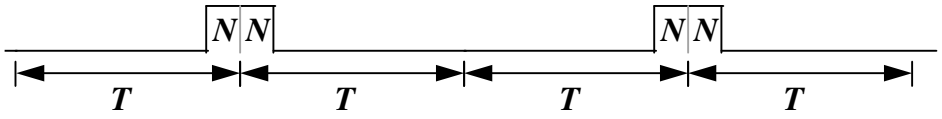

Figure 2: Worst case traffic for jumping window mechanism.

Sliding Window

The Sliding Window mechanism (SW) also limits to *N* the maximum number of cells submitted to the network during a window of length *T*. The difference consists on the way windows are considered. The beginning of a window is synchronized to the arrival instant of each cell and its end occurs after exactly *T* time units. Hence, several superposed windows are opened and closed along the time, all of them restricting the traffic to the maximum of *N* cells in any time interval of length *T*. The windows superposition yields an effect of traffic "memorization". Figure 3 shows this mechanism behavior.

Instead of considering several windows, another way of conceiving this mechanism is to consider only one window sliding along the time memorizing cell arrival instants. This memorization is done by a counter that is decremented (memorizing) at cell arrival instant and it is incremented (releasing from memory) after *T* time units.

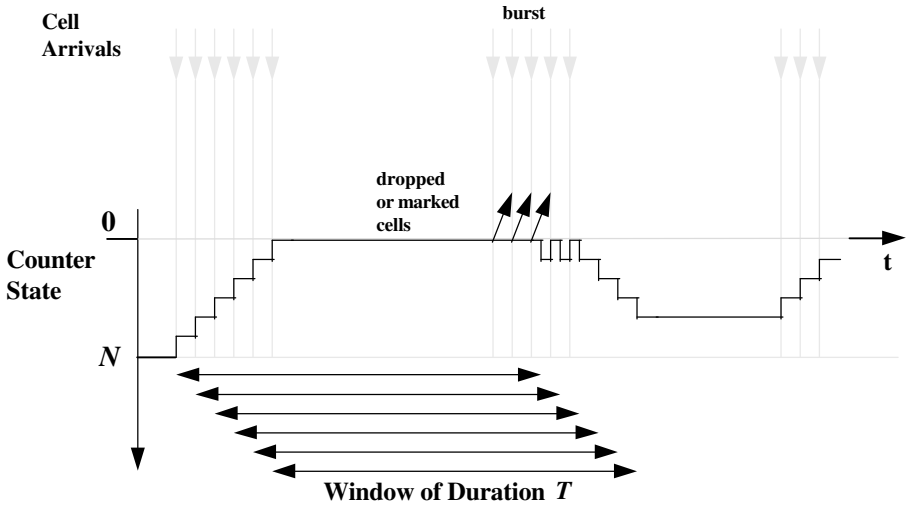

Figure 3 - The Sliding Window Mechanism.

Regarding the behavior of this mechanism, the number of cells submitted to the network, during every time interval of length *T,* is limited to the maximum number of tokens *N*. Therefore, sliding window mechanism has the advantage of restricting the policed rate to the value of *N*/*T*, during **every** interval of length *T*. The worst case traffic is given by a burst of length *N*, followed by a silence period of $T - (N \times t_{cel})$, as illustrated in figure 4.

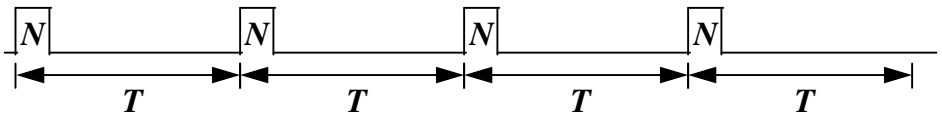

Figure 4: Worst case traffic for sliding window mechanism.

The mechanism SW also presents the advantage of memorizing the traffic behavior during a particular interval. However, its implementation complexity is considerably high taking into account

the need of storing the cell arrival instants, firing a timer for each cell arrival in addition to computing the number of arrivals up to *N* cells. This can be extremely expensive for long duration windows. It is possible to simplify the scheme of storing arrival instants by associating a bit for each position of *T* where the cell can arrive. Moreover, for high values of *N* this scheme still demands high implementation cost.

3 The Time Slotted Moving Window Mechanism

The Time Slotted Moving Window (TSMW) proposed in this section looks for a trade-off between jumping window simplicity and sliding window efficiency regarding the traffic memorization, the accurate policing of mean rate and the restriction to burst length.

One of the main factors that leads the sliding window mechanism to require a high implementation complexity is the storage of cells arrival times. On the other hand, the juxtaposed windows of jumping window mechanism can be simply implemented with an auto-firing timer.

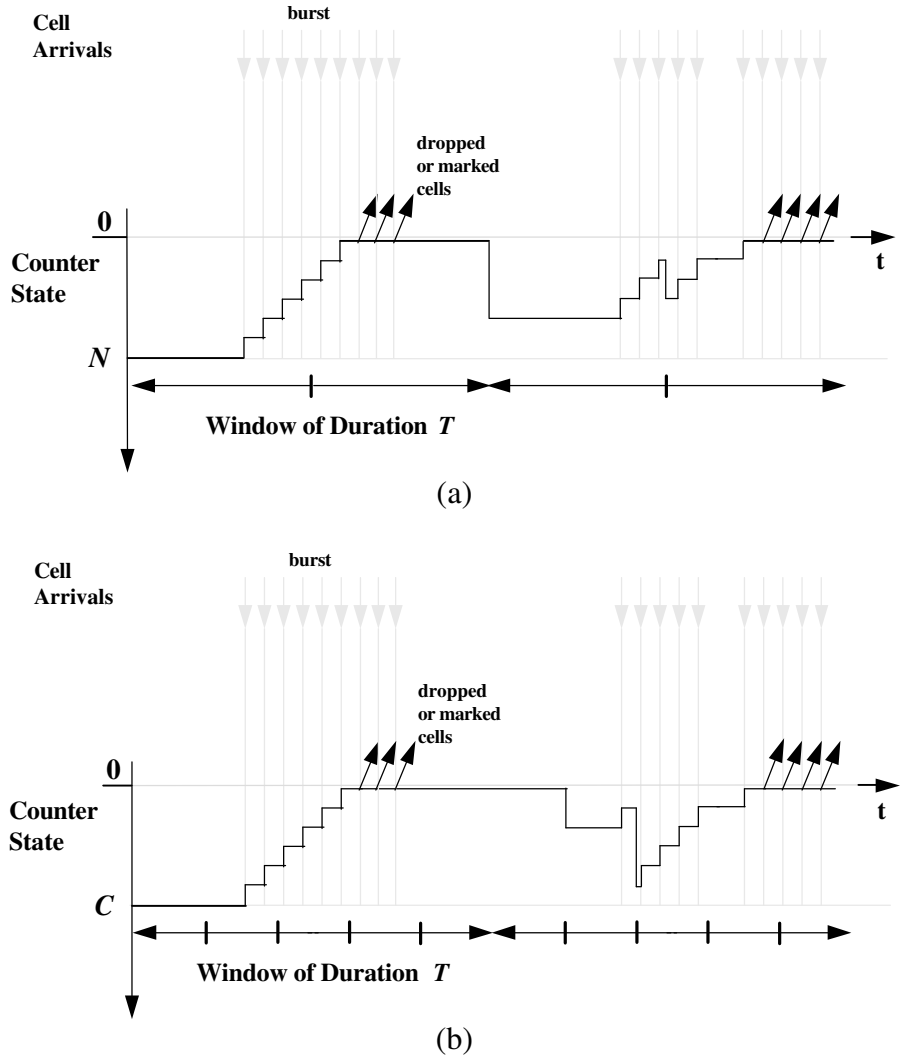

Figure 5: The Slotted Moving Window Mechanism with a) 2 slots and b) 5 slots.

In this mechanism the window of duration *T* is subdivided in *S* periods of equal time named slots. Similarly to JW and SW, at each cell arrival a counter, if positive, is decremented. The counter is incremented by the number of cells that arrived during a slot s_i, *T* time units after the

beginning of this slot s_i. Hence, the time slotted moving window is based on the idea of "superposed windows that jump, from slot to slot" instead of sliding.

It can be observed that for the number of slots S equal to one this mechanism is equivalent to jumping window. When S tends towards infinite, the duration of each slot tends towards zero and this mechanism approximates to sliding window mechanism.

The worst case traffic is generated by bursts of cell length N. The bursts are synchronized in an alternate way with the slots. The end of a burst synchronizes to the end of a slot, while the next burst starts at the same instant of the beginning of the slot that it occupies. This pattern is repeated periodically. Figure 6 shows a comparison among the approached mechanisms. When the burst duration is shorter than slot duration, the minimum silence period between bursts has a duration of S - 1 slots. In this situation, as each slot lasts T/S time units, the duration of the minimum silence period can be expressed as:

$$\frac{(S-1)\times T}{S}.$$

As well as sliding window, this mechanism limits the burst duration to N cells while the maximum burst duration of jumping window is N cells. However, the mechanism does not succeed to maintain sliding window characteristic where in every time interval equal to T, only N cells can be submitted to the network. As in jumping window the source can transfer up to $2N$ cells in a T time period but they are spaced by a period of time that is at least equal to one slot, in the worst case this silence period is $T/2$ for two slots.

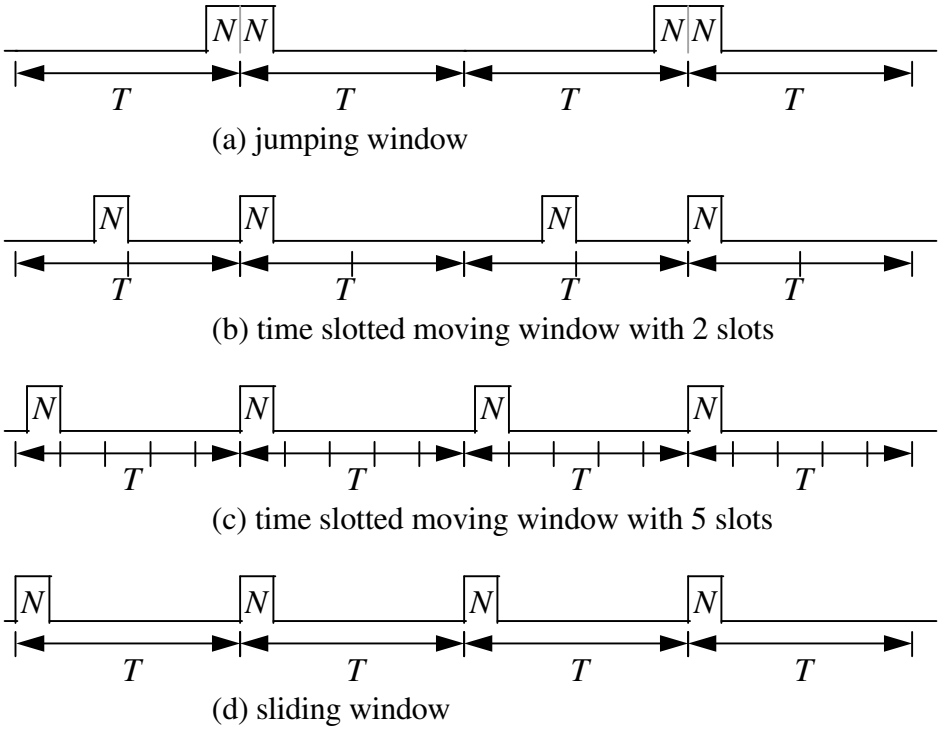

Figure 6: Comparison of worst case traffic for time slotted moving window mechanism.

For an increase in the number of slots, the minimum interval between the bursts also increases. However, when S tends towards infinite the slot duration goes to zero and each burst cell arrives in a different slot. In this situation, the worst case traffic pattern of time slotted moving window tends towards worst case traffic pattern of sliding window.

The Quality of Service Manager is responsible to configure the number of slots of this mechanism, according to QoS established requirements.

4 Simulation and Implementation

This section presents structured descriptions of the simulation and the implementation of the policing mechanisms considered. To simulate the policing mechanisms it was used the software *Queueing Networks Analysis Package* - QNAP that is based on queueing networks with a confidence level of 0.95. In this way, each policing mechanism is modeled by network elements as described as follows.

The jumping window was modeled by a queue with constant service time equal to T to generate juxtaposed windows. The toll queue verifies if each cell generated by the source is violator, in accordance with the counter value of the mechanism (figure 7).

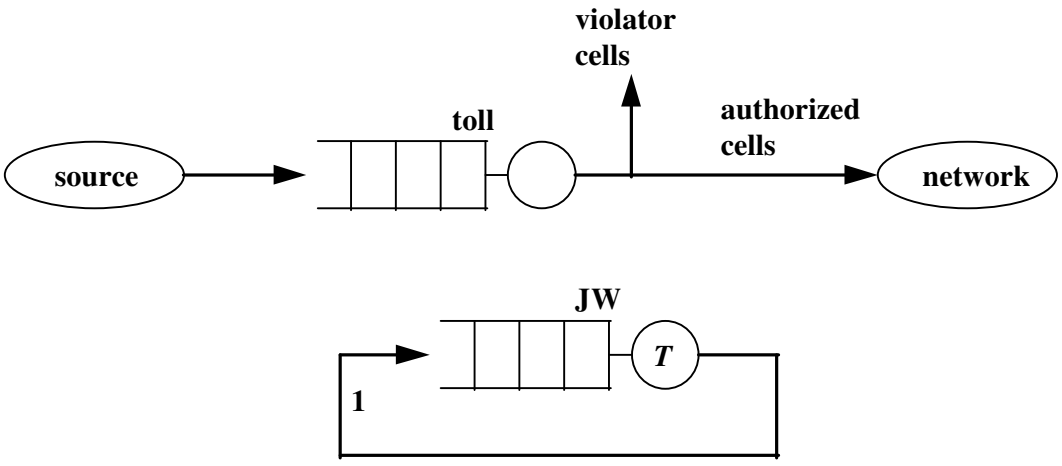

Figure 7: Simulation model for jumping window.

The sliding window was modeled (figure 8) as a loss system with multiple servers, where the service times are deterministic with duration T. The number of servers is equal to the counter upper bound N.

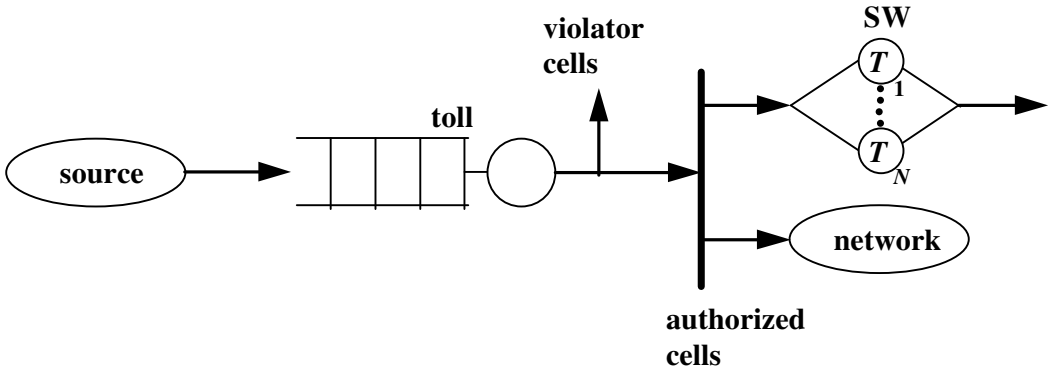

Figure 8: Simulation model for sliding window.

To simulate the time slotted moving window (figure 9) it was used a queue with a server of deterministic service time equal to *T/S*. It was employed an array FIFO for the mechanism control. At the end of each time interval *T/S* the mechanism counter is increased by the value of the array FIFO last position. At this time, the array is shifted of one position and the number of cells that passed through the mechanism during the last slot is stored in the first position of the array. The toll queue verifies the condition of the cell violation according to the counter value, as stated to the previous mechanisms.

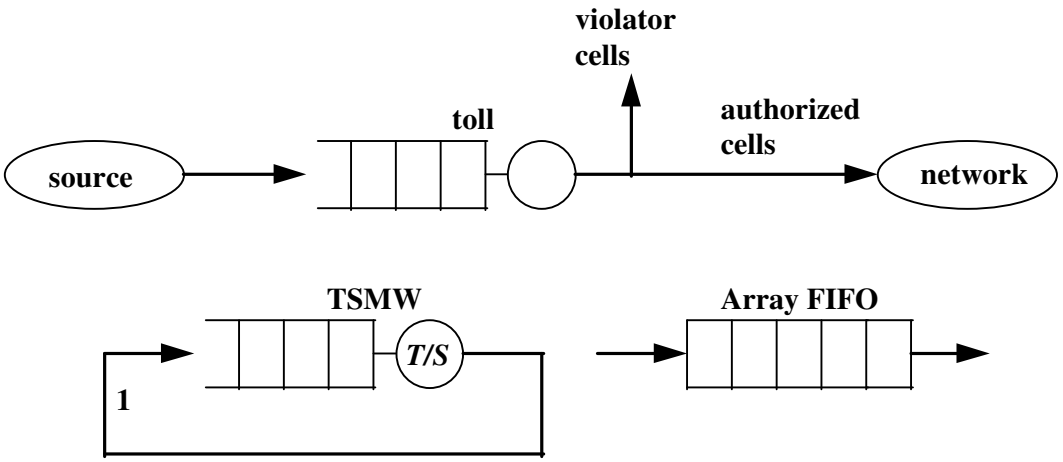

Figure 9: Simulation model for time slotted moving window.

The policing mechanisms implementation made use of a development environment of high performance communication systems, with quality of service guarantees proposed in [7-10]. This environment defines optimized modules of memory management, timer management, task scheduling, in addition to data structures for the interfaces among the modules, the application, the protocols of the communication system and of the medium access. The policing mechanisms were implemented in C language in UNIX and DOS environment.

The jumping window mechanism is implemented according to the following procedure. After connection establishment and of parameter configuration, the JW creates a auto-firing timer responsible for token updating at each *T* time unit. Concurrently, for each cell offered to JW, it tests the token variable and if the result is positive, the mechanism decreases the number of tokens and delivers the cell to be transmitted, multiplexed with the other connections, otherwise the cell is considered violator, and in this case it was chosen to discard it.

Sliding window mechanism can be implemented with several timers and by a token counter. For each cell submitted to the network if the token counter is positive the mechanism decreases the counter and fires a timer with duration of *T* time units. At each timer time-out (end of window) the token variable is incremented. This scheme will be referenced in this work as continuous sliding window. This implementation has the inconvenient of requiring a large and variable number of timers, implying in a higher utilization of the memory and of the CPU, in addition to raising difficulties to an efficient resource allocation for the connection.

Dittman *et al* [6] propose another scheme of implementation residing on a pre-allocation of a shift register where each bit registers a cell arrival in the respective time unit. This scheme will be referenced as discretized sliding window. To allow an arrival of just one cell per time unit, it is necessary that the time unit has the duration equal to t_{cel} - the time required to transmit a cell - for example 2.7 μs for 155 Mb/s.

The time slotted moving window proposed in this work was implemented with an array FIFO allocated at connection establishment according to the parameters of the traffic and of the mechanism. Different from other window-based mechanisms, in addition to the parameters T and N, this mechanism defines the parameter S that indicates the number of slots in each window of duration T. This parameter is extremely important in the definition of memory and CPU utilization. The greater is the number of slots S, the greater is the amount of memory required for the allocation of the array FIFO, the greater is the CPU utilization and the closer this mechanism is to SW. Otherwise, the smaller is the number of slots, the smaller is the necessary amount of memory, and the closer this mechanism is to JW. When connection establishment phase is completed, the array FIFO is allocated with size $S \times$ nb, where nb is the number of bits necessary to represent the maximum number of cells that can be transmitted during a slot. Analytically the amount of bits necessary to represent the array FIFO is expressed by:

$$M = S \times nb;$$

where:

$$nb = \left\lfloor \log_2 \left\lceil T \Big/ \left(S \times t_{cel} \right) \right\rceil \right\rfloor + 1 .$$

The timer is responsible for the token updating at each interval of T/S duration. This updating is related to the token increment by the number of cells that arrived during the slot s_i started T time units before this updating. This value is stored in the last position the array FIFO.

5 Comparison of the Mechanisms

This section presents quantitative and qualitative results obtained in simulation and implementation comparing considered mechanisms. The comparison concentrates on the number of timers and their precision, the maximum burst duration, the required memory and the violation probability.

In order to compute violation probability (p_{viol}) of the mechanisms in both simulation and implementation, two traffic patterns were used - a geometric traffic where the mean of the interval between cell arrivals is equal to five time units and a burst/silence traffic. In the last one, burst length was given by a geometric distribution with mean 10, where the burst bit rate is equal to one cell/time unit and the silence period was given by a geometric distribution with a mean of 40 time units. Therefore, this traffic has a burstiness factor (maximum cell rate divided by mean cell rate) equal to 5 and such as the previous traffic, its mean bit rate is equal to 1/5 cells/time unit. The values of $T = 50$ and $N = 10$ were chosen for all mechanisms, representing a mean bit rate proportional to 1/5 cell/time unit.

Figure 10 shows the influence of the configured number of slots on the violation probability behavior of time slotted moving window mechanism, submitted to a geometric traffic, while figure 11 shows the mechanism behavior for burst/silence traffic. The vertical axis of the graphics shows the ratio R that is defined as the ratio of TSMW violation probability to SW violation probability. The lower bound of the graphics is the ratio of JW violation probability (TSMW with $S = 1$) to SW violation probability, while the upper bound is equal to one representing ratio R to SW itself. In both graphics it is possible to verify the similarity between the simulation and implementation results, being simulation results more precise than implementation ones, as it was simulated hundred times more traffic samples than the amount of samples used in implementation analysis. It can be observed that for a number of slots S equal to one, the violation probability of time slotted moving window mechanism is the same of the jumping window mechanism and as long as number of slots increases, the violation probability tends quickly towards violation probability of sliding window mechanism

(R = 1). In the case of burst/silence traffic the p_{viol} tends more quickly - or since a lower number of slots - towards its final value (R = 1 or sliding window p_{viol}) than in geometric traffic case.

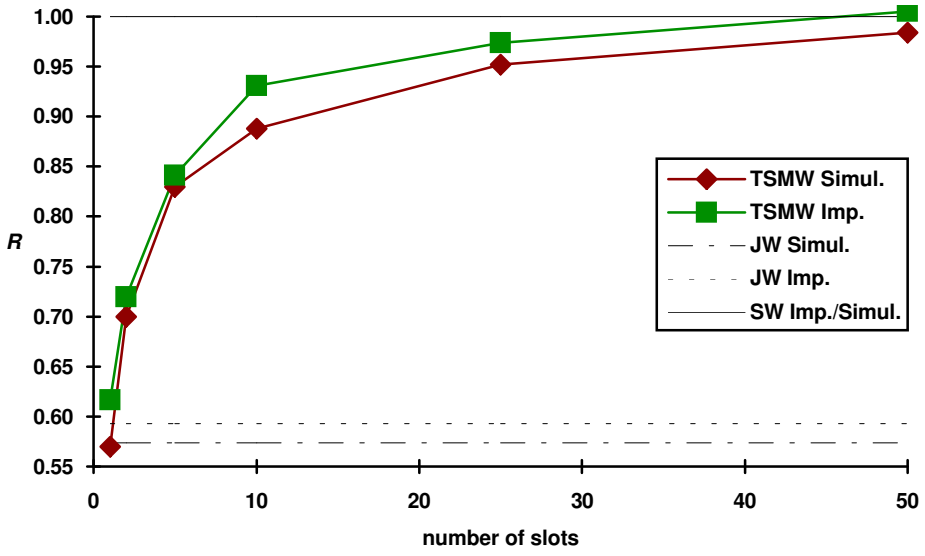

Figure 10: Influence of the number of slots on the behavior of TSMW (for geometric traffic).

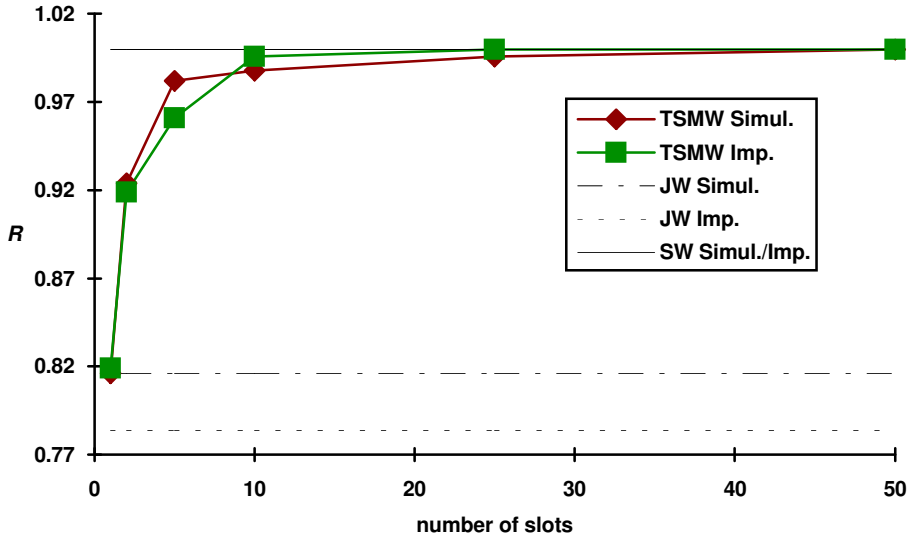

Figure 11: Influence of the number of slots on the behavior of TSMW (for burst/silence traffic).

Table 1 contains comparative information among the mechanisms. The notation $\{x\}$ indicates the number of bits necessary to represent x value. In relation to number of timers, it can be observed that continuous sliding window mechanism needs the highest value. It must be emphasized that the greater the number of timers, the greater is the CPU utilization. Table 1 shows jumping window characteristic of allowing maximum burst duration equal to $2N$.

An important issue to be considered is the required precision of timers. The continuous sliding window mechanism needs very high precision timers because they must be fired exactly at the

arrival of each cell, in other words, at any instant of time. Discretized sliding window is a little more feasible, however there is still a need for a considerable high precision, for example 2.7μs for a 155 Mb/s transmission bit rate. The jumping window mechanism needs the lowest precision timers, in other words, of the magnitude of window duration *T*. The time slotted moving window yields a timer utilization with all precision range - since precision used in JW to the one used in SW - according to the number of slots used.

Table 1: Comparison of the Mechanisms.

Mechanism	JW	Continuous SW	Discretized SW	**TSMW**
Number of Timers	1	N	1	**1**
Maximum Burst Duration	$2N$	N	N	$\boldsymbol{N}$
Precision of Timers	low (policed rate}	high (non limited)	high (max. source rate)	**adjustable over all the range**
Required Memory for Timers	{Timer}	$N \times$ {Timer}	{Timer}	**{Timer}**
Required Memory for Counters and Parameters	$\{N\}$ + {Counter}	$\{N\}$ + {Counter}	$\{N\}$ + {Counter}	$\boldsymbol{\{N\}}$ **+ {Counter} +** $\boldsymbol{\{S\}}$
Control Memory	———	———	$\left\lceil \frac{T}{t_{cel}} \right\rceil$	$\boldsymbol{S \times \left\{ T / \left(S \times t_{cel} \right) \right\}}$

Another very important issue in implementation complexity analysis is the amount of memory necessary for each mechanism. The two variants of sliding window are comparable in relation to the necessary amount of memory. Jumping window is the one that requires the smallest amount of memory. In the case of time slotted moving window, the need is for a variable amount of memory that is, according to the number of slots, inferior to the amount required to sliding window. For example, the case of discretized moving window with $T = 60 \times t_{cel}$ and $N = 30$. In this case it requires 60 bits to store the traffic of a window interval. If time slotted moving window is used with 20 slots then at most 3 cells can arrive during each slot. These cells are represented by 2 bits. Hence, the total number of bits to represent the array FIFO is 40 bits. For a TSMW with 10 slots, 30 bits are required and with 5 slots, 20 bits. It can be observed from figure 11 that with 5 slots, the behavior of time slotted moving window is very close to sliding window performance.

6 Conclusion

This paper presented the time slotted moving window mechanism. The mechanism key idea resides in the division of window in equal time intervals. This discretization of the averaging interval of traffic allows to aggregate the processing of arrival events during each slot. The functionality and the complexity of this mechanism ranges from those of jumping window (proposed mechanism with $S = 1$) until those of sliding window, according to the used number of slots.

Among important qualities of the mechanism is possible to mention the traffic memorization, the accurate policing of mean bit rate and the restriction of the maximum burst length up to N cells, in addition to the low complexity of implementation and flexibility in the precision of used timer. The worst case traffic of this mechanism is characterized by the transfer of up to $2N$ cells to the

network during a *T* time period, but they are spaced by a silence period at least equal to one slot duration, that in the worst case is *T*/2 for two slots.

Simulation and implementation results obtained for geometric and burst/silence traffic patterns confirm that even with a small number of slots ($S = 5$) a functionality very close to sliding window performance can be obtained, with a low implementation complexity and with considerably saving of memory and CPU processing.

Considering the example mentioned in section 5, this mechanism, with only one timer of *T*/5 precision, limits the burst length up to 5 cells obtaining a memory saving of 66% in relation to discretized sliding window and achieving very close result to this mechanism. Regarding violation probability, the results for burst/silence traffic present a relative difference of just 4%.

7 Acknowledgments

The authors acknowledge the collaboration of Paris VI University professors, Bruno Baynat and Kim Thai that helped very much in the simulation work during CAPES/COFECUB mission.

8 References

[1] ITU-T Recommendation I.311, B-ISDN General Networks Aspects, Genebra, 1991.

[2] E. Rathgeb, "Modeling and Performance Comparision of Policing Mechanisms for ATM Networks", IEEE JSAC, pp.325-334, vol. 9, no. 3, Apr. 1991.

[3] M. Butto, E. Cavallero, A. Tonietti, "Effectiveness of the "Leaky Bucket" Policing Mechanism in ATM Networks", IEEE JSAC, pp. 335-342, vol. 9, no. 3, Apr. 1991.

[4] A. Berger,"Performance Analysis of a Rate Control Throttle Where Tokens and Jobs Queue", IEEE INFOCOM, pp. 30-38, 1990.

[5] J. Junior, J. Monteiro, "On The Efficiency of Policing Mechanisms for ATM Networks", IEEE ITS, pp.470-474, Aug. 1994.

[6] L. Dittman, S. Jacobsen, K. Moth, "Flow Enforcement Algorithms for ATM Networks", IEEE JSAC, pp. 343-349, vol 9. no. 3, Apr. 1991.

[7] C. V. Albuquerque, M. L. Lanza, O. C. Duarte, "Sistemas de Comunicação Abertos em Redes Locais: Uma Implementação e Medidas de Desempenho em Ambientes de Computadores Pessoais", XIII Congresso Sociedade Brasileira de Computação (SEMISH), pp. 327-329, Florianópolis, SC, Brazil, Sep. 1993.

[8] C. V. Albuquerque, M. D. Nunes, O. C. Duarte, "Performance Measurements in a Manufacturing Communication System", IEEE International Conference on Circuits and Systems ISCAS' 94, pp.3.69-3.72, London, England, May 1994.

[9] L. F. Baginsky e O. C. Duarte, "Um Modelo de Implementação de Alto Desempenho para Protocolos de Comunicações Estruturados em Camadas", Revista da Sociedade Brasileira de Computação", vol. 6, pp. 46-56, Apr. 1991.

[10] L. F. Baginsky e O. C. Duarte, "Um Modelo de Implementação de Alto Desempenho para Sistemas Abertos", IX Congresso da Sociedade Brasileira de Telecomunicações, São Paulo, Brazil, pp. 19.3.1-19.3.5, Sep. 1991.