# Design of an ATM Shaping Multiplexer with Guaranteed Output Burstiness

*H. Jonathan Chao and Jun S. Hong*
Polytechnic University
Department of Electrical Engineering
6 Metrotech Center, Brooklyn, New York 11201
TEL: (718)260-3302, FAX: (718)260-3074
E-mail: chao@antioch.poly.edu, jhong@mobius.poly.edu

August 28, 1996

**Abstract**

In order to utilize network resources effectively while providing satisfactory quality of service (QoS) to all users on the ATM network, it is necessary to control users' traffic at the network edges. ATM traffic may be controlled at the User-Network Interface (UNI) and Inter-Network Interface (INI) based on the traffic contracts. We propose an ATM shaping multiplexer along with its implementation architecture for all ATM traffic connections multiplexed in an output stream. These connections are to be strictly regulated by the so-called Generic Cell Rate Algorithm (GCRA). When ATM streams conforming to the GCRA are multiplexed together, contention can disturb the streams such that they are no longer in conformance. Thus, a departure event driven traffic shaping (DEDTS) algorithm is adopted to shape the cell streams in such a way that the outgoing ATM stream of each individual connection conforms to the ATM traffic descriptors, Peak Cell Rate (PCR), Sustainable Cell Rate (SCR), Cell Delay Variation Tolerance (CDVT), and Burst Tolerance (BT) declared in the traffic contract. A proposed architecture can be implemented to support up to thousands of virtual connections using off the shelf components.

## 1 Introduction

Asynchronous Transfer Mode (ATM) has been considered a promising technique for transporting multimedia services (including voice, video, and data) over a wide area network. However, there are still some problems that have to be resolved prior to the ubiquitous deployment of ATM networks. Among them, congestion control is one of the most important and challenging problems, and much research effort has been devoted to solving this problem. Congestion control becomes difficult in ATM networks because of a large product of bandwidth and round-trip

delay, diverse quality of service (QoS) requirements, and various traffic characteristics. One effective congestion control technique is to enforce users' traffic, which is called usage parameter control (UPC), at the network edges such that the network resources are protected from being overused by malicious users or misbehaving terminal equipment. ATM connections are monitored and enforced based on their traffic contracts by network operators at the User-Network Interface (UNI) and Inter-Network Interface (INI) (Figure 1). If a connection does not conform to the traffic contracts, the UPC (i.e., policing) function will take appropriate actions with violating cells, such as discarding them or tagging them to a lower priority.

The Generic Cell Rate Algorithm (GCRA), also known as the leaky-bucket algorithm, is widely accepted as an effective policing scheme [1, 2, 3, 4, 5] in the ATM network and thus has been standardized in ATM Forum [6] and ITU-T [7]. The GCRA is used to determine if the connection conforms to the negotiated traffic descriptors: Peak Cell Rate (PCR) and Sustainable Cell Rate (SCR) associated with Cell Delay Variation Tolerance (CDVT) and Burst Tolerance (BT), respectively.

The conformance check for the PCR and SCR parameters by the UPC function should take into consideration the cell delay variation (CDV) caused by contention of cell streams when being multiplexed to the network. In order to have a low violation rate (e.g., $10^{-10}$) at the UPC/NPC function, one may assign large CDVT and BT values to the connections, which may introduce some flaws. For instance, it creates an opportunity for tricky users who may choose the PCR and the SCR lower than the actual usage while choosing large CDVT and BT values. As a result, traffic with different patterns would be considered as conforming to the negotiated parameters and could pass transparently through the UPC/NPC function. Thus, an accurate estimate of the maximum allowable CDV at a given interface is critical, which creates a burden for network operators.

From network operators' point of view, smaller tolerances to the traffic variation (e.g., CDVT) are generally preferable because they allow better network design. One possible solution to reduce the CDVT value is to use a traffic shaper at a given interface. The shaping function that helps users' traffic comply with the negotiated traffic descriptors can be operated by users at the UNI and/or by different network carriers at the INI. Cell streams that have been individually shaped according to the GCRA may fail to conform to the their traffic descriptors

after being multiplexed to an output stream to the network.

This is because contention among different connections to the upstream may perturb their shaped traffic patterns. Thus, it is important to perform traffic shaping after the multiplexer, called ATM shaping multiplexer.

We adopt a so-called departure event driven traffic shaping (DEDTS) algorithm to strictly enforce each connection is conformance to its traffic descriptors at the multiplexed output stream. We also design an implementation architecture for the ATM shaping multiplexer, which can be implemented to accommodate any number of virtual connections by using off the shelf components.

This paper is organized as follows. Section 2 presents the regularity condition – dual leaky-bucket. Section 3 describes the departure event-driven traffic shaping algorithm (DEDTS). Section 4 shows an implementation architecture for the ATM shaping multiplexer that performs the DEDTS algorithm. Section 5 presents the performance study for two different shaping algorithms (arrival event driven and departure event-driven) through computer simulations. Finally, Section 6 presents conclusions.

# 2 Regularity Condition – Dual Leaky-Bucket

Throughout this paper, we use the so-called dual leaky-bucket, denoted GCRA($T_p$,$\tau_p$; $T_s$,$\tau_s$), as a regularity condition, which defines the conformance to the traffic parameters $T_p$ and $T_s$ with associated CDVT ($\tau_p$) and BT ($\tau_s$) at a given interface. Sustained Emission Interval ($T_s$) and Peak Emission Interval ($T_p$) are obtained by the inverse of the SCR and PCR normalized by the line bandwidth, respectively. The parameter BT ($\tau_s$) includes the IBT (intrinsic burst tolerance, $\tau_s^*$) and the CDVT at a given interface. IBT defines the burstiness of a traffic source in conjunction with the SCR at the physical service access point (PHY-SAP) of an equivalent terminal.

Figure 2 shows a dual leaky-bucket, composed of two leaky-buckets (a sustainable rate bucket and a peak rate bucket) for each connection. We have adopted the "continuous-state" representation [6] rather than the discrete credit token representation in order to eliminate division operations.

As an incoming cell of a connection arrives at the dual leaky-bucket and finds its bucket levels below the limits $\tau_s$ and $\tau_p$, it is a conforming cell. If either one of the buckets exceeds the limit, it is considered a violating cell. Both buckets drain out at a continuous rate of 1 unit of content per cell time, and, after transmitting one cell, increase by the amount of $T_s$ and $T_p$, respectively. The sizes of the sustainable and peak rate buckets are $(T_s + \tau_s)$ and $(T_p + \tau_p)$, respectively.

The CDVT ($\tau_p$) and BT ($\tau_s$) have different values at different points in the network. For example, at the PHY-SAP of an equivalent terminal, $\tau_s$ and $\tau_p$ are defined to accommodate only the source traffic characteristics. On the other hand, at the public UNI, they need to include the accumulated CDV that occurs at the private ATM network. In this paper, we denote $\tau_s^*$ and $\tau_p^*$ as intrinsic tolerances for the traffic descriptors at the PHY-SAP. Determination of the CDV tolerance is beyond the scope of this paper and will not be discussed.

In general, there may be multiple traffic patterns that simultaneously conform to GCRA($T_p$,$\tau_p$; $T_s$,$\tau_s$). Figure 3 shows a typical traffic pattern with the associated traffic parameters. Here, we formulate the maximum back-to-back cells $M$ and the maximum burst size ($MBS$) that can be seen at the output of the dual leaky-bucket. Note that these parameters are connection traffic descriptors rather than source traffic descriptors. Although, in the ITU-T and ATM Forum, the term MBS is defined in conjunction with IBT to represent the source traffic characteristics, we use it as a general term to represent the maximum burst size at a given interface along the connection path. Instead, the intrinsic MBS (IMBS) is used to represent the maximum burst size associated with IBT. Thus, the $MBS$ is defined as the maximum number of cells that are transmitted at a rate *greater than or equal to*[1] the peak cell rate, and is thus limited as follows:

$$M = \left\lfloor \frac{\tau_p}{T_p - 1} \right\rfloor + 1 \tag{1}$$

$$MBS = \left\lfloor \frac{\tau_s + \tau_p}{T_s - T_p} - (M - 1) \right\rfloor + 1. \tag{2}$$

For a case where $\tau_p = 0$ (i.e., at the PHY-SAP), the minimum cell interval will not be smaller than the $T_p$ and thus $M = 1$ and $MBS = 1 + \lfloor \tau_s/(T_s - T_p) \rfloor$ [6] (i.e., IMBS).

[1]On the contrary, the IMBS is defined as the maximum number of cells that are transmitted at the PCR. The reason that some cells may be transmitted at a rate greater than the PCR is due to non-zero CDVT.

Furthermore, in order to support any real numbers for the PCR, the CDVT ($\tau_p$) needs to be set to a non-zero value in some cases. For example, if $T_p = 1.5$ (allowing every 2 cells transmitted in 3 time slots) there may be 2 cells transmitted back-to-back due to the slotted system in ATM networks. Thus, a minimum of $\tau_p = 0.5$ needs to be assigned to the case of $T_p = 1.5$. If $\tau_p$ is always set to zero, the GCRA($T_p$,0; $T_s$,$\tau_s$) will only allow certain rates of PCR that gives $T_p$ to be integers.

# 3 ATM Shaping Multiplexer Algorithm

When the dual leaky-bucket algorithm is used for shaping users' traffic by delaying violation cells, a cell buffer is required to temporarily store those violating cells. As pointed out earlier, as multiple shaped connections are multiplexed to a single transmission line, each cell stream may no longer comply with the associated GCRA($T_p$,$\tau_p$; $T_s$,$\tau_s$) due to contention of the multiple connections at the output stream. Here, we will present an ATM shaping multiplexer that strictly shapes each individual connections to conform to its GCRA($T_p$,$\tau_p$; $T_s$,$\tau_s$).

## 3.1 Shaping and Scheduling

The conventional method of designing the traffic shaper for multiple connections is to separate the traffic shaping function and the scheduling function [8, 9]. Figure 4 shows an abstract model of conventional traffic shaping and scheduling. When a cell of a virtual connection (VC) arrives at the traffic shaper, it is stored in its associated VC buffer. If both bucket levels are below the limits, the cell is eligible to be transmitted and is moved to a scheduler (e.g., a first-in first-out server). The bucket levels are increased when the cell passes through the dual leaky-bucket. If two or more cells from different connections are eligible to transmit at the same time slot, only one cell will be chosen by the scheduler for transmission and the rest of them will wait in a buffer. We call this approach the *Arrival Event Driven Traffic Shaping* because the bucket levels are increased at the cells' arrival time. Although the output cell stream of each buffered dual leaky-bucket complies with the GCRA($T_p$,$\tau_p$; $T_s$,$\tau_s$), as multiples of them are multiplexed into a single transmission line, each shaped cell stream may violate GCRA($T_p$,$\tau_p$; $T_s$,$\tau_s$) at the output of the multiplexer. This is due to cell contention among eligible cells for transmission.

For example, if a properly shaped cell stream is delayed due to the loss of contention, it may clump with its following cells, introducing a larger CDV. Computer simulations have shown that this CDV is not negligible. Some intelligent scheduling schemes other than the FIFO discipline may overcome this problem by rearranging cells' departure sequence. However, it may be too complex to be practical.

In order to strictly limit the CDVT and BT at the output of the shaping multiplexer, the actual departure time of the cell plays a key role in shaping function. Unlike the AEDTS, the *Departure Event Driven Traffic Shaping* (DEDTS) increases the bucket levels when a cell departs from the shaping multiplexer (as shown in Figure 5).

## 3.2 Departure Event-Driven Traffic Shaping (DEDTS) Algorithm

To facilitate the implementation of the shaping multiplexer, we time-stamp cells based on dual leaky-bucket levels. The DEDTS keeps a real time (RT) clock that ticks and increases by one in every cell time slot. Cells in the shaping multiplexer cannot be transmitted until their departure times (DTs) are due (equal to RT) or overdue (greater than RT). However, using only the time-stamp method may cause a bottleneck problem when it is required to update multiple connections' bucket levels and calculate multiple cells' DTs in the same cell time slot.

In our DEDTS algorithm, bucket level update and DT calculation are executed only when the precedent cell of the same virtual channel connection departs from the shaping multiplexer. Since the departure event occurs at most once in a time slot, the DEDTS needs only to calculate one DT at a time, thus eliminating the bottleneck problem.

The following four steps briefly describe the DEDTS algorithm:

1. When a connection (say $VC_i/VP_i$) is established, source traffic descriptors such as the sustainable cell rate (SCR) and the peak cell rate (PCR) are determined and some intermediate parameters are initialized for the DT calculation. At a given interface, the CDVT ($\tau_p$) and BT ($\tau_s$) are also determined on the traffic contracts.

2. When a valid cell arrives at the shaping multiplexer, it is stored in a logical queue (a linked list of cells that have the same VCI/VPI), and the number of backlogged cells ($NB_i$) is increased by one. For each logical queue, only the head-of-line (HOL) cell is

assigned a DT. The DT calculation algorithm will be discussed in the next section. When a cell arrives and finds the logical queue empty ($NB_i = 0$), the cell will be assigned a DT with either the arrival time (AT) or a pre-calculated DT, whichever is larger.

3. Cells that have the same DT are linked together in a timing queue. As the real time clock ticks and reaches the DT value, all cells in the timing queue become eligible for transmission and will join a departure queue, which links all the cells that are eligible for transmission. The detailed operations will be explained in Section 4.

4. As soon as a cell departs from the shaping multiplexer, a new DT is calculated and assigned to the new head-of-line cell in the same logical queue. If there is no cell in the same logical queue, the newly calculated DT will be stored in a look-up table memory for the future arriving cell.

### 3.3 DT Calculation and Bucket Levels Update

Here, we describe how the DEDTS algorithm calculates the departure time and updates the bucket levels. The required variables for calculating the DT are shown in Figure 6. Some variables such as $DT$, $X_s$ and $X_p$ change dynamically, while others such as $T_s$, $T_p$, $\tau_s$ and $\tau_p$ are constant and remain unchanged during the lifetime of a call (except in Available Bit Rate (ABR) flow control; these parameters may change). Let us denote $C_i^k$ as the $k^{th}$ cell of $VC_i/VP_i$, and $DT^k$ and $t_d(k)$ as the *calculated* and *actual* departure time for the $k^{th}$ cell, respectively. $X_s^k$ and $X_p^k$ are defined as the sustainable and peak rate bucket levels after transmitting the $k^{th}$ cell from the shaping multiplexer at time $t_d(k)$. In Figure 7 we show these variables in a timing diagram to assist the explanation of the algorithm. The horizontal line represents the time axis. The solid arrows approaching to the lower horizontal line are actual cells arriving at the shaping multiplexer. The solid arrows emanating from the upper horizontal line are actual cells departing from the shaping multiplexer. The small triangles in the middle line represent the calculated DT. The dashed arrows on the time axis represent the tentative DT values.

The DEDTS calculates the DT based on both bucket levels ($X_s$ and $X_p$), which are updated at the same time. When calculating the $DT$ for the next cell, we use the stored values in a

look-up table such as $DT$, $X_s$ and $X_p$. Note that $DT$[2], $X_s$ and $X_p$ are real numbers, allowing us to support any sustainable or peak cell rates.

For better explanation of the algorithm (Figure 8), let us assume that the current real time RT is exactly the departure time of cell $C_i^k$ on the time axis, i.e., $RT = t_d(k)$. Note that the actual departure time $t_d(k)$ and its calculated departure time $DT^k$ of cell $C_i^k$ may not be the same because of cell contention of multiple connections, as shown in Figure 7. Note that cell arrival time, $t_a(k)$, is shown for illustration purposes and is not used in the DT calculation.

1. As soon as cell $C_i^k$ departs from the shaping multiplexer, the next cell's departure time ($DT^{k+1}$) is calculated and the bucket levels ($X_s^k$ and $X_p^k$) are updated in two steps. Because each bucket's contents are drained out continuously at a rate of 1 unit per cell time since the last time the bucket level was updated at $t_d(k-1)$, both bucket levels should have decreased by $[t_d(k) - t_d(k-1)]$ at time $t_d(k)$, or

$$\begin{aligned} X_s^* &= max\{0, X_s^{k-1} - [t_d(k) - t_d(k-1)]\} \\ X_p^* &= max\{0, X_p^{k-1} - [t_d(k) - t_d(k-1)]\}. \end{aligned} \tag{3}$$

   Note that the bucket levels never go below zero. With the assumption of cell $C_i^k$ departing at the current time, we update $X_s^k$ and $X_p^k$ by adding $T_s$ and $T_p$ to $X_s^*$ and $X_p^*$, respectively.

$$\begin{aligned} X_s^k &= X_s^* + T_s \\ X_p^k &= X_p^* + T_p. \end{aligned} \tag{4}$$

2. Now, in order to calculate the next cell's DT ($DT^{k+1}$), we first choose a tentative DT ($DT'$) one slot away from the current departure time, i.e., (see Figure 7).

$$DT' = t_d(k) + 1. \tag{5}$$

3. We then check whether $X_s$ and $X_p$ are greater than the limit $\tau_s$ and $\tau_p$ at time $DT'$. Before checking, we must reduce $X_s$ by the amount that the bucket would have drained out its content at $DT'$ since the last time the bucket level was updated ($t_d(k)$) in step 1.

[2]In a slotted environment such as ATM, only integer values of the DT are allowed. Although we keep the real numbers of the DT in the look-up table, we will assign integers by rounding up the DT.

Note that the slope of the bucket level decreasing is 1 as shown in Figure 7). Thus, the bucket levels ($X_s$ and $X_p$) at time $DT'$ become

$$\begin{aligned} X'_s &= X^k_s - 1 \\ X'_p &= X^k_p - 1. \end{aligned} \tag{6}$$

4. If both bucket levels $X'_s$ and $X'_p$ are not greater than the limits (i.e., $X'_s \leq \tau_s$ and $X'_p \leq \tau_p$), $DT^{k+1}$ is assigned with $DT'$. On the other hand, if either bucket level is greater than the limits, $DT^{k+1}$ is assigned a time when both bucket levels will reduce to the limits, as shown in Figure 7. We choose the earliest possible time for the departure time as $C^{k+1}_i$ among many choices that will pass through the GCRA($T_p$,$\tau_p$; $T_s$,$\tau_s$) test. By delaying the cell's departing time by max $[(X'_s - \tau_s), (X'_p - \tau_p)]$, we can expect that both bucket levels will reduce to the limits at $DT^{k+1}$. Thus, $DT^{k+1}$ will be assigned as the following:

$$DT^{k+1} = DT' + max[(X'_s - \tau_s), (X'_p - \tau_p)]. \tag{7}$$

# 4 Implementation Architecture

In this section, we present an implementation architecture for the departure-event driven traffic shaping, a Memory-based architecture. In our previous work [8] we proposed a Sequencer-based architecture that has been implemented with the existing VLSI chip called the Sequencer [12]. However, as the number of connections becomes large, the Sequencer-based architecture requires many Sequencer chips and is not cost effective. On the other hand, the Memory-based architecture that can be implemented using off the shelf parts can handle any number of VCs.

Figure 9 shows a Memory-based architecture, which is divided into two units. The storage unit is composed of a Cell Memory, an input FIFO, and an output FIFO. The control unit, composed of a VCI/VPI register, a CAM (Content Addressable Memory), a microprocessor, a look-up table, and a Timing Processor, generates an appropriate addresses to read (write) cells from (to) the Cell Memory based on the departure-event driven traffic shaping algorithm. The Timing Processor includes a Queue-Controller and an Auxiliary Memory, which is a key component in the Memory-based architecture. We will discuss it in detail in the following

section. Here, we will briefly explain the operations of the Memory-based shaping multiplexer. Its detailed design in hardware implementation can be found in Ref. [13].

## 4.1 Timing Processor

The operations of the Timing Processor are illustrated in Figure 10. The Timing Processor has three major queues. First, cells that belong to the same VC are linked together in a logical queue, called the VC-Queue. Second, cells that have the same time-stamp (DT) are linked together in a queue, called the Timing-Queue. And last, cells whose departure time is due or overdue are linked together in a logical queue, called the Departure-Queue. The contents in the VC-Queue are addresses of cells stored in Cell Memory. The contents of both the Timing-Queue and the Departure-Queue are Channel Identifier (CI) values. There is also an Idle-Address Linked List (IALL) which keeps the idle spaces of the Cell Memory.

Newly arriving cells are appended to corresponding VC-Queues based on the CI values. When a cell becomes the head-of-line (HOL) cell of a VC-Queue, it is assigned a DT and appended to the corresponding Timing-Queue. As the real time clock ticks and as the real-time (RT) pointer passes the DT, the Timing-Queue whose DT is identical with RT will be appended to the Departure-Queue as shown in Figure 10. As shown in Figure 10, as the RT pointer moves from 1 to 2, cells at DT=2 are appended to the Departure-Queue. The HOL cell of Departure-Queue is read out one at a time slot. Its content, CI, is then used to access the HOL cell of the corresponding VC-Queue, where the cell address is obtained to transmit the cell in the Cell Memory.

## 4.2 Write-In Procedure

When a cell arrives at the shaping multiplexer, it is first stored in the input FIFO in Figure 9. Its VCI/VPI is extracted from the cell header to access the corresponding Channel Identifier (CI) from the Content Addressable Memory (CAM). The cell address of the Cell Memory is fetched from the IALL. The address is then used as an index to store the cell in the Cell Memory.

Meanwhile, the CI is used to access the corresponding VC's $NB$ (number of backlogged cells) from the look-up table, which has $N$ entries for $N$ VCs. If the $NB$ is equal to zero, its precalculated DT is accessed from the look-up table and compared with RT. The value of

$max(RT, \lceil DT \rceil)$ is used as an index to link the cell to a Timing-Queue. On the other hand, if $NB$ is not equal to zero, it will simply be appended to the VC-Queue. Note that only the head of line (HOL) cell from each VC will be able to join the Timing-Queue, while all other cells are stored in the VC-Queue and wait until they become the HOL cells.

### 4.3 Read-Out Procedure

As the real time clock ticks, the value of RT is used by the Queue Controller as an index to access a Timing-Queue, whose DT is equal to RT. This Timing-Queue will then be appended to the tail of the Departure-Queue.

Cells in the Departure-Queue are the cells that are eligible to be transmitted. In every cell time slot, the HOL cell in the Departure-Queue will be transmitted and then the next cell will become a new HOL cell.

Whenever there is a cell to be sent out, the first step is to fetch its CI from the Departure-Queue. The CI is used as an index to find a VC-Queue, where cells' addresses are linked together. The cell body of the HOL cell in the VC-Queue will be read out from the Cell Memory, placed into the Output FIFO, and transmitted to the network. Meanwhile, the CI is also used to read out the corresponding contents of the look-up table for the microprocessor to calculate the DT of the next cell. If the NB is not zero (i.e., the VC-Queue is not empty), a newly calculated DT will be used as an index to join the new HOL cell to an associated Timing-Queue. On the other hand, if the NB is zero, the newly calculated DT will be stored in the look-up table for the next arriving cell. At last, the available cell location will be linked to the IALL.

## 5 Simulation Study

So far, we have discussed the algorithm and the implementation architecture of the shaping multiplexer. In this section, we compare the performances of the arrival event-driven and the departure event-driven traffic shapings with two experiments. The first experiment compares the average delay and the second experiment measures the violation rate to the GCRA ($T_p$,$\tau_p$; $T_s$,$\tau_s$).

Intuitively, the DEDTS will introduce more delay than the AEDTS. That is because, for the DEDTS, once a cell is delayed due to the loss of contention, the following cells that belong to the same virtual connection may also be further delayed to avoid GCRA violation. This is especially important for CBR (constant bit rate) services. If the DEDTS algorithm is operated with tight CDV tolerances when shaping CBR traffic, the delay that is added by the shaper will be accumulated and increase monotonically. Thus, CBR traffic has to be treated differently as in queue management for meeting its delay requirement. So let us only consider VBR (variable bit rate) traffic and find out how much more delay introduced by the DEDTS when compared with the AEDTS. Our simulation study shows that there is not much difference in the average delay between these two event-driven traffic shaping algorithms. The DEDTS allows each connection to strictly conform to its traffic contracts while introducing a negligible extra delay. Thus, it is preferable to the AEDTS.

Figure 11 shows the simulation model. In this simulation study, we assume that $N$ heterogeneous traffic sources generate cells and they are multiplexed into a FIFO buffer with an infinite buffer size. The multiplexed output stream is fed into a shaping multiplexer. At the output of the shaping multiplexer, a fictitious policing function monitors each individual connection with the GCRA($T_p$,$\tau_p$; $T_s$,$\tau_s$).

We use an ON-OFF source model as traffic sources in a discrete-time environment where time is slotted. Let us assume the traffic source model alternates between active and silent modes. During the active mode cells are generated in back to back, and during the silent mode no cells are generated. The length of the active and silent periods are geometrically distributed with average lengths $E[B]$ and $E[I]$, respectively. Let us define $p = \text{Prob}\{\textit{starting a new burst per time slot}\}$ and $q = \text{Prob}\{\textit{the arrived cell that is the last cell in the burst}\}$. Then the probability that the burst has $i$ cells is

$$P[B = i] = (1 - q)^{i-1} q, \quad i \geq 1.$$

The probability that an idle period lasts for $j$ time slots is

$$P[I = j] = (1 - p)^{j} p, \quad j \geq 0.$$

Thus,

$$E[B] = \frac{1}{q} \quad \text{and} \quad E[I] = \frac{1-p}{p}.$$

The offered load ($\rho$) is equal to $E[B]/(E[B]+E[I])$.

The source traffic descriptors, the SCR and the PCR, are set as follows. The SCR is set to the offered load ($\rho$) of each connection. The PCR is chosen arbitrarily in order to see the effect of limiting the peak cell rate. The sources generate a peak cell rate of 1 cell/slot. The IBT ($\tau_s^*$) is set to accommodate the IMBS, which is assigned the mean burst length ($E[B]$) of the traffic source. The intrinsic CDVT ($\tau_p^*$) is assigned one, as discussed in Section 2. The source traffic types and their source traffic descriptors are listed in Table 1.

In our experiments, we investigate an average delay of the shaping multiplexer and the violation rate of GCRA ($T_p$,$\tau_p$; $T_s$,$\tau_s$) by varying the CDV tolerances. The results shown in Figure 12 are obtained by generating $50 \times 10^6$ cells with the same traffic patterns for the AEDTS and DEDTS. The CDVT ($\tau_s$) and BT ($\tau_p$) vary proportionally to the IBT ($\tau_s^*$) and intrinsic CDVT ($\tau_p^*$).

$$\tau_s = \eta \cdot \tau_s^*, \quad \tau_p = \eta \cdot \tau_p^*.$$

where $\eta \geq 1$. If $\eta$ is less than 1, the DEDTS may not be able to clear out the buffer.

As $\eta$ increases, the average delay of AEDTS and DEDTS decrease. When $\eta$ becomes $10^4$, the average delay is almost zero, representing that there is no shaping effect. In other words, if the CDV tolerances are large enough, an arriving cell will be transmitted immediately and experiences no delay. As shown in Figure 12 (a), the average delays of the two algorithms are almost identical. The difference is about 2 cell slots in this experiment.

Figure 12 (b) shows the GCRA ($T_p$,$\tau_p$; $T_s$,$\tau_s$) violation rates at the output of the shaping multiplexer. This result is also obtained by the same simulation setup as the first experiment. The output stream shaped by the DEDTS algorithm always conforms to the GCRA ($T_p$,$\tau_p$; $T_s$,$\tau_s$) while the one shaped by the AEDTS algorithm violates GCRA ($T_p$,$\tau_p$; $T_s$,$\tau_s$) noticeably. Although the AEDTS reduces the CDV tolerances to some degree as compared to the "No shaping" case, the violation rate is still quite high. Again, the simulation experiment shows that as $\eta$ becomes $10^4$, all cells are considered as conforming even without the shaping function.

For our simulation setup, when $\eta = 30$, the cell delay variation impact to regularity condition GCRA is worst.

From the simulation experiments, we conclude that AEDTS and DEDTS algorithms have similar performances in average delay. The AEDTS requires larger CDV tolerances at the UPC/NPC functions to achieve low cell violation rate and does not provide the guarantee of burstiness, while the DEDTS guarantees each connection strictly conforming to any given CDVT values. Thus, the shaping multiplexer eliminates the difficulty of determining the CDVT values. A network operator may just set the CDVT based on the delay requirement.

# 6 Conclusion

In order to meet various quality of service (QoS) requirements, users' traffic must be enforced to conform to a set of traffic descriptors, sustainable and peak cell rates (SCR and PCR), and associated cell delay variation tolerance (CDVT) and burst tolerance (BT). If a connection does not comply with the traffic contracts, some action has to be taken against the violating traffic.

It is very challenging to implement an ATM shaping multiplexer capable of shaping traffic from thousands of virtual connections on an input line using exiting hardware technology. Even though the traffic of each individual virtual connection may have been shaped properly, the statistical multiplexing may perturb each connection's traffic flow pattern, and that may violate the negotiated traffic parameters. Moreover, it is difficult to determine the CDVT values due to the multiplexing of random traffic from many virtual connections.

We have presented the *Departure Event Driven Traffic Shaping* (DEDTS) algorithm along with its implementation architecture for an ATM shaping multiplexer. It guarantees that each virtual connection's traffic flow will strictly conform to the negotiated parameters and that no cell will be enforced by the UPC/NPC functions. The algorithm combines both shaping and scheduling functions using the time stamping cells based on the DEDTS algorithm, which eliminates the bottleneck problem when shaping thousands of virtual connections at the input line. The DEDTS eliminates the difficulty of choosing proper CDVT and BT values. Our simulation study shows that an extra delay that may be introduced in the departure event driven traffic shaping is negligible.

# References


[1] J. S. Turner, "New directions in communications (or which way to the information age?)," *IEEE Communications Magazine*, 24(10):8–15, October 1986.

[2] E. P. Rathgeb, "Modeling and Performance Comparison of Policing Mechanisms for ATM Networks," *IEEE Journal on Selected Areas in Communications*, SAC-9(1):325–334, April 1991.

[3] M. Sidi, W.-Z. Liu, I. Cidon, and I. Gopal, "Congestion control through input rate regulation," *IEEE Transactions on Communications*, 41(3), March 1993.

[4] P. Boyer, F. M. Guillemin, M. J. Servel, and J.-P. Coudreuse, "Spacing cells protects and enhances utilization of ATM network links," *IEEE Network*, 6(5):38–49, September 1992.

[5] F. Guillemin, P. Boyer, A. Dupuis, and L. Romoeuf, "Peak rate enforcement in ATM networks," in *IEEE INFOCOM*, vol. 2, pages 753–758 (6A.1), Florence, Italy, May 1992.

[6] The ATM Forum, Traffic Management Specification Version 4.0, April 1996.

[7] ITU-T, "Recommendation I.371: Traffic Control and Congestion Control in B-ISDN," Perth, November 1995.

[8] H. J. Chao, "Architecture design for regulating and scheduling user's traffic in ATM networks," *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 77–87, Baltimore, Maryland, August 1992.

[9] H. Zhang and D. Ferrari, "Rate-controlled static-priority queueing," Technical Report TR-92-003, Computer Science Division, University of California at Berkeley, Berkeley, California, November 1992.

[10] L. Zhang, "Virtual Clock: A New Traffic Control Algorithm for Packet Switching Networks," *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 19-29, Philadelphia, PA, September 1990.

[11] H. J. Chao, "Design of Leaky Bucket Access Control Schemes in ATM Networks," in *Conference Record of the International Conference on Communications (ICC)*, Denver, CO, June 1991.

[12] H. J. Chao and N. Uzun, "A VLSI Sequencer chip of ATM traffic shaper and queue manager," *IEEE Journal of Solid-State Circuits*, 27(11), November 1992.

[13] P.-C. Chen, "The Design of A Timing Processor for ATM Traffic Shaper," Master's thesis, Polytechnic University, July 1995.

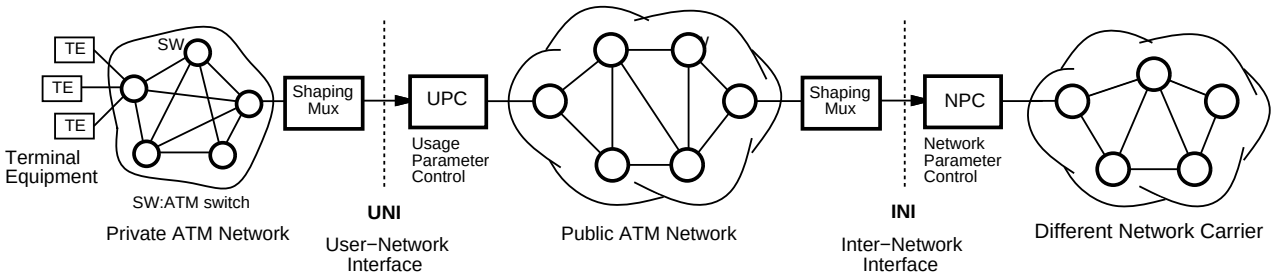
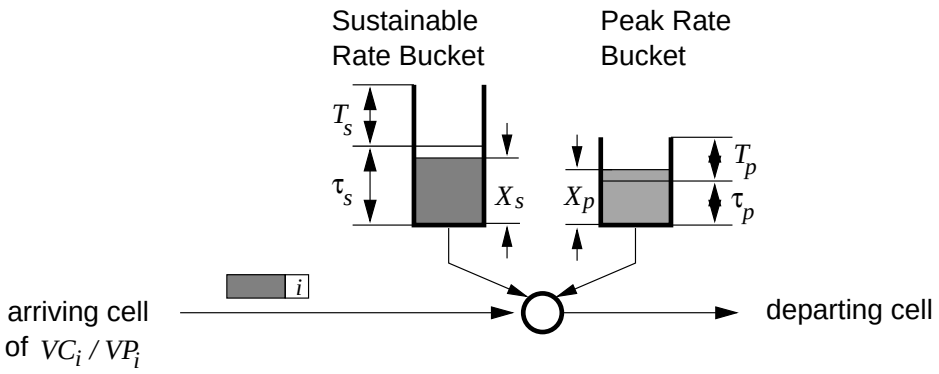
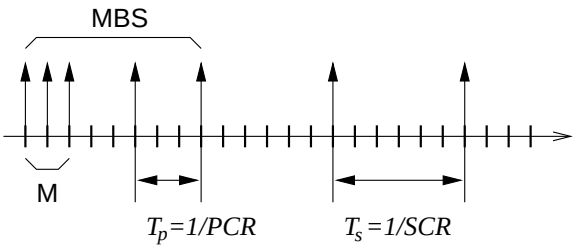



Figure 1: ATM shaping multiplexers and Usage/Network Parameter Control (policing) function at the User-Network Interface (UNI) and Inter-Network Interface (INI).




$T_s$ : Sustained Emission Interval
$T_p$ : Peak Emission Interval
$\tau_s$ : Burst Tolerance
$\tau_p$ : Cell Delay Variation Tolerance
$X_s$ : Sustainable Rate Bucket Level
$X_p$ : Peak Rate Bucket Level

Figure 2: A continuous-state representation of the dual leaky-bucket.




M : Number of maximum back to back cells
MBS : Maximum Burst Size
PCR : Peak Cell Rate
SCR : Sustainable Cell Rate

Figure 3: A typical traffic pattern conforming to the dual leaky-bucket algorithm.

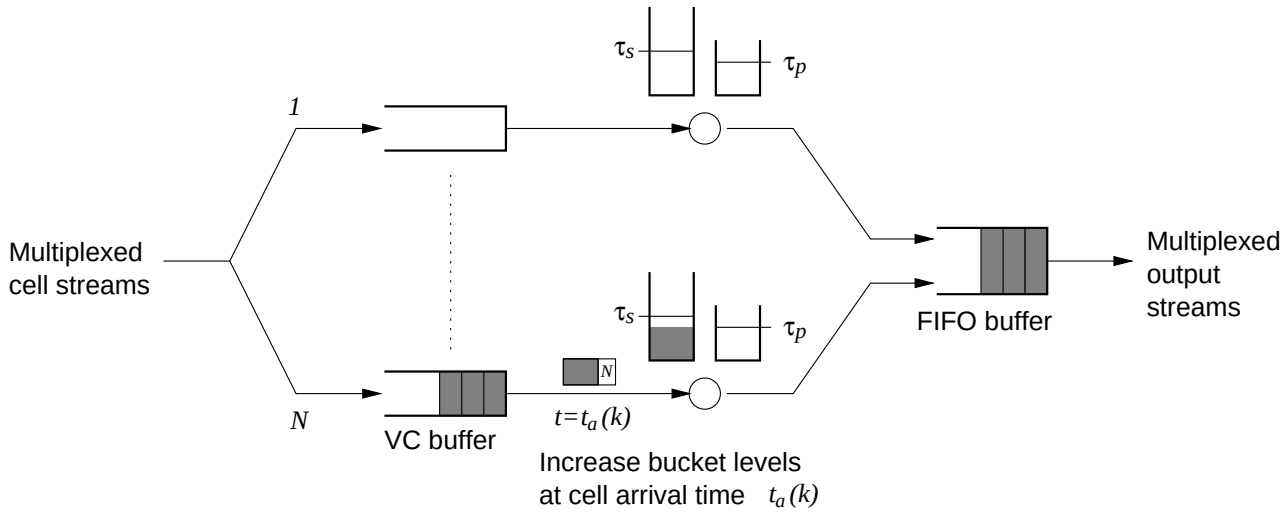



Figure 4: An abstract model of the Arrival Event Driven Traffic Shaping.

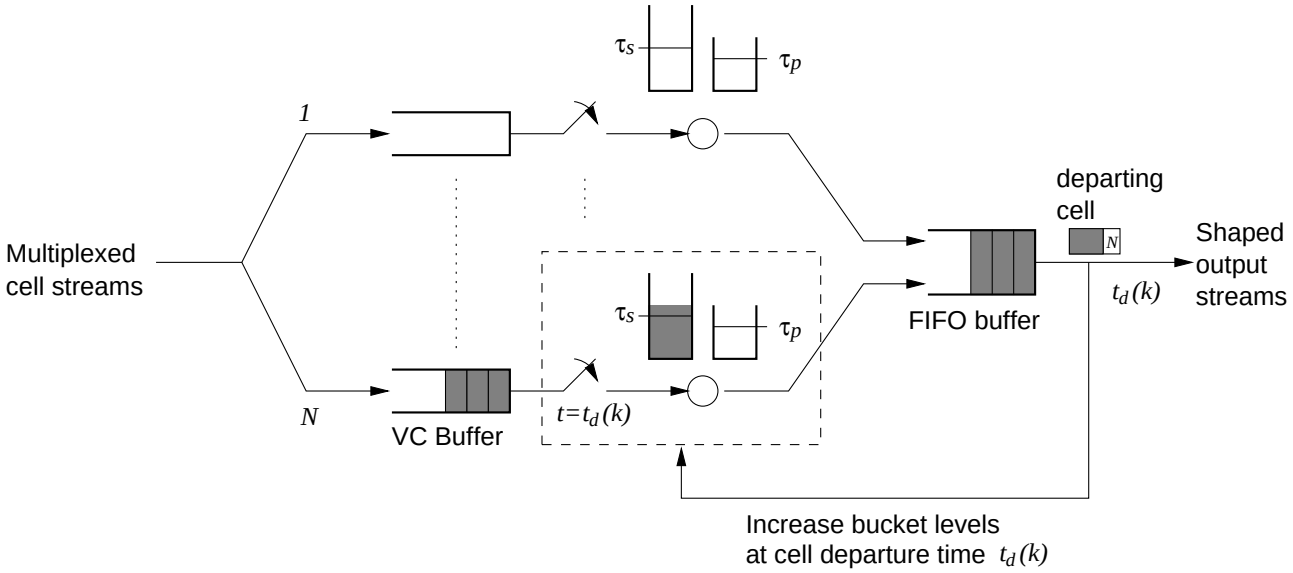
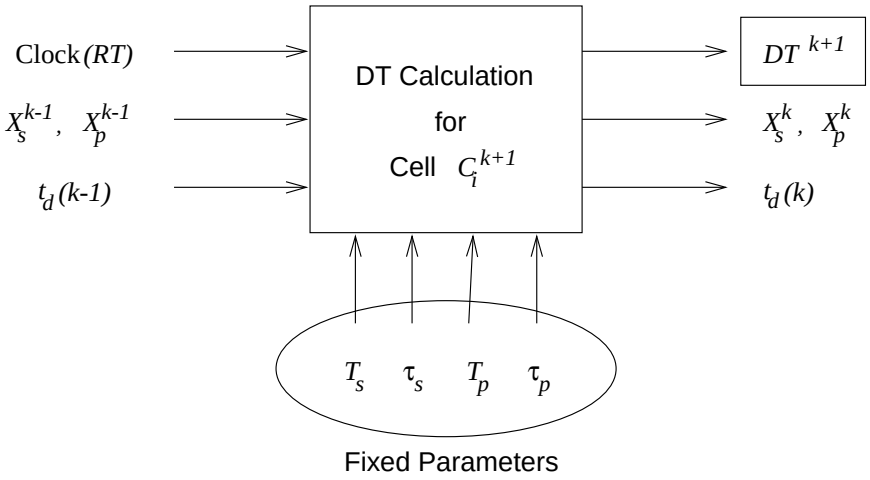



Figure 5: An abstract model of the Departure Event Driven Traffic Shaping.




| | | |
|---|---|---|
| $C_i^{k+1}$ | : | the $(k+1)^{th}$ cell of the $i^{th}$ virtual connection |
| $t_d(k)$ | : | the actual departure time of $C_i^k$ |
| $X_s^k$ | : | the sustainable rate bucket level after transmitting $C_i^k$ at time $t_d(k)$ |
| $X_p^k$ | : | the peak rate bucket level after transmitting $C_i^k$ at time $t_d(k)$ |
| $DT^{k+1}$ | : | the calculated departure time for $C_i^{k+1}$ |

Figure 6: Variables for calculating cells' departure time (DT).

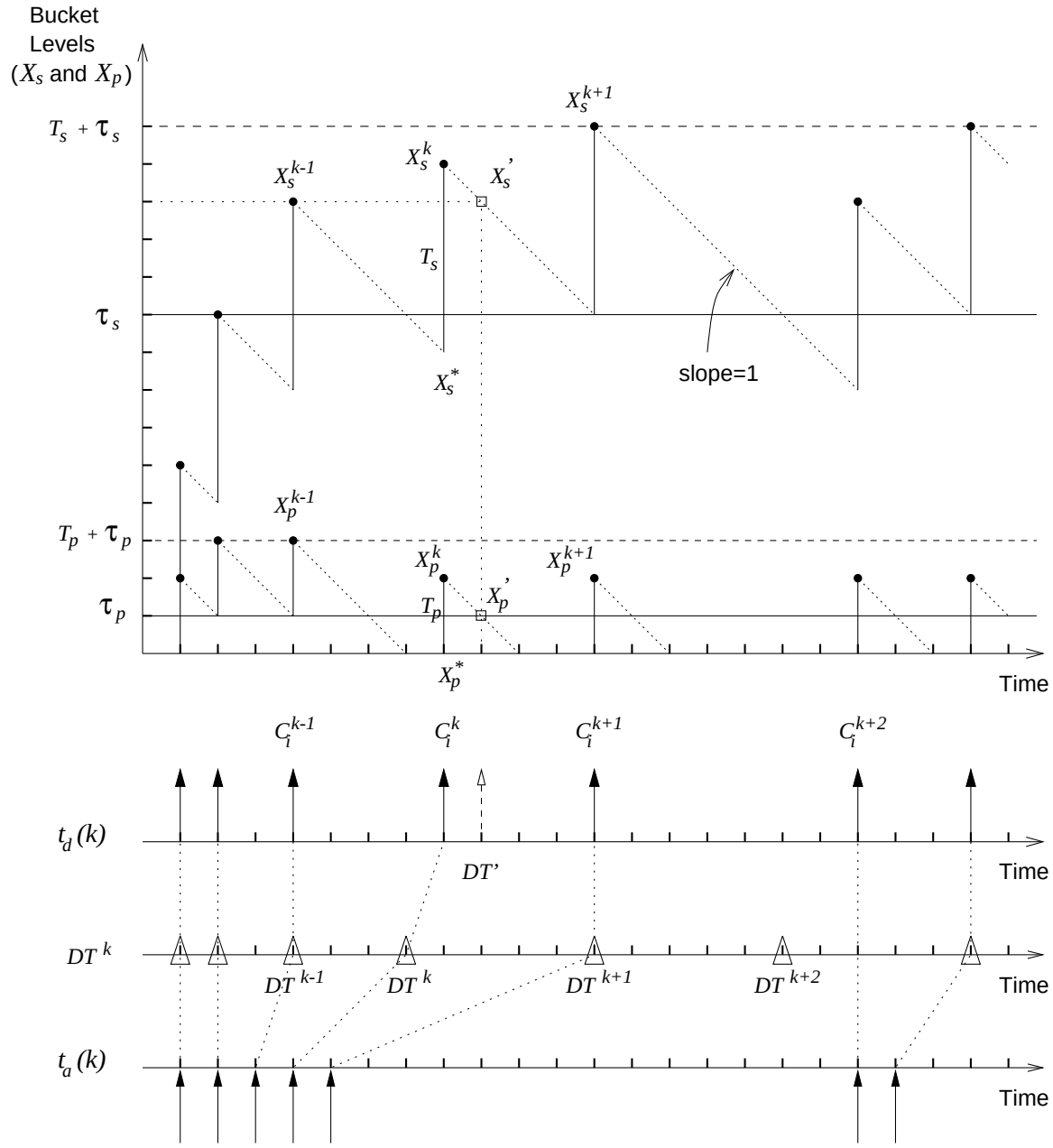



Figure 7: Timing diagram for departure time (DT) calculation.

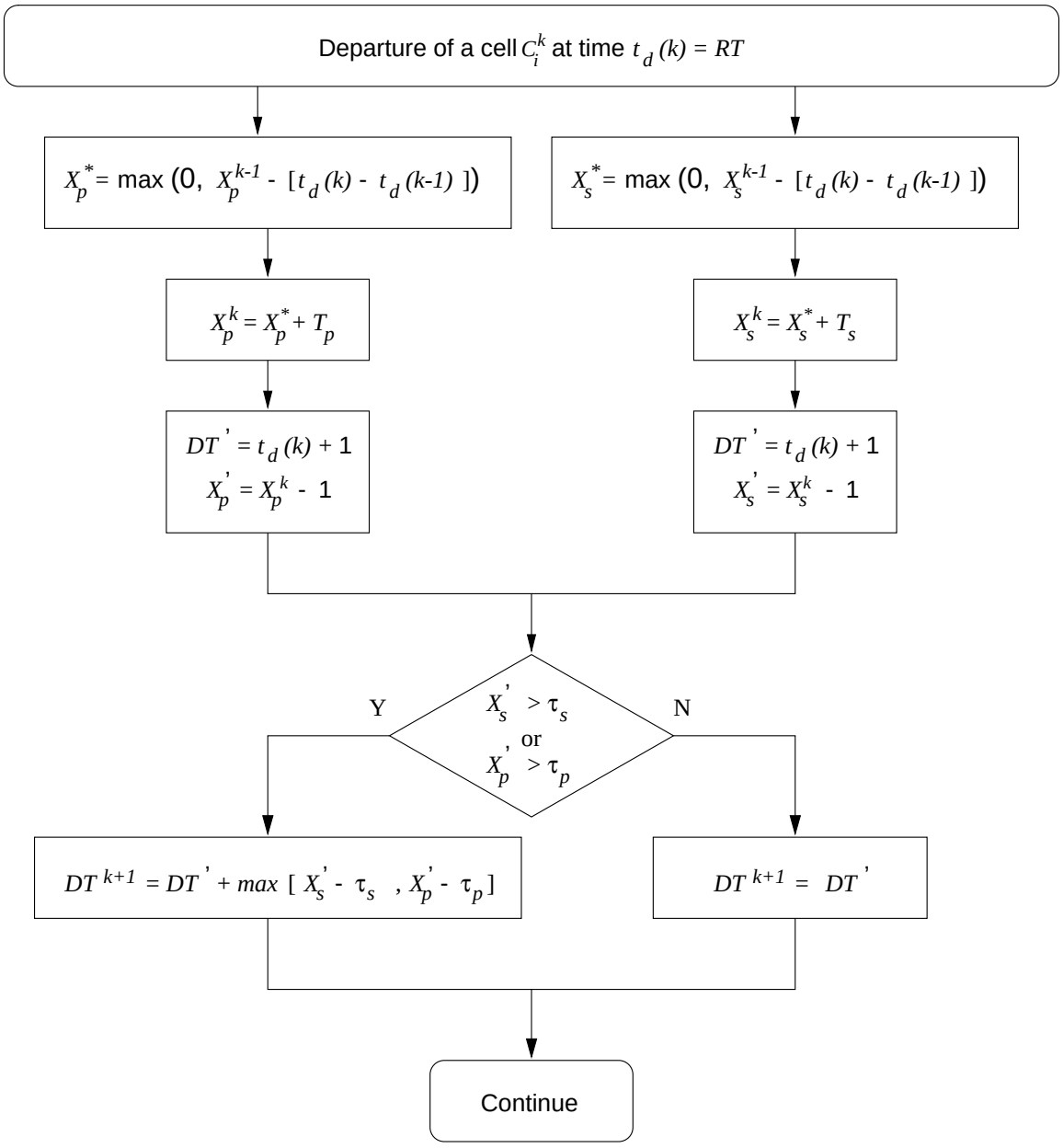



Figure 8: The flow chart of the DEDTS algorithm for DT calculation and bucket level update.

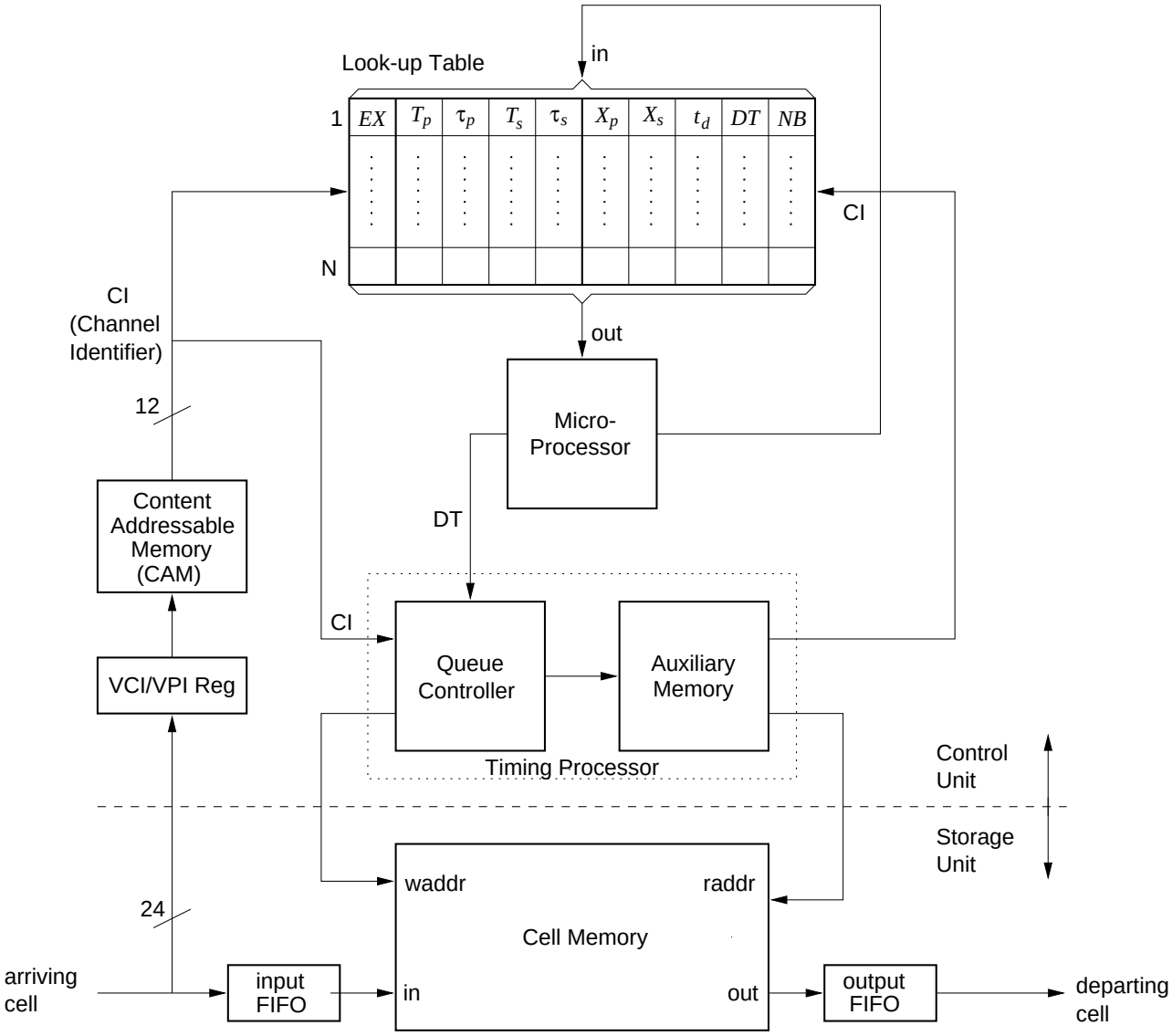



Figure 9: A Memory-based architecture for the ATM shaping multiplexer.

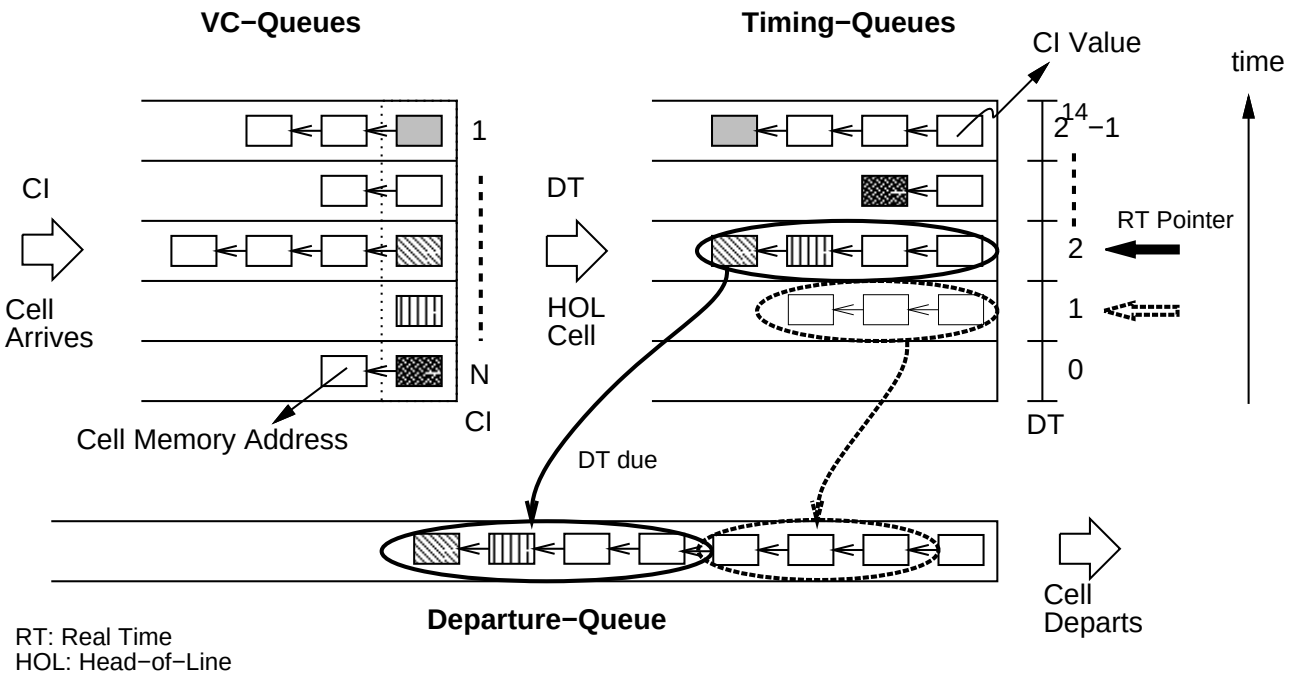



Figure 10: Three major queues in the Timing Processors.

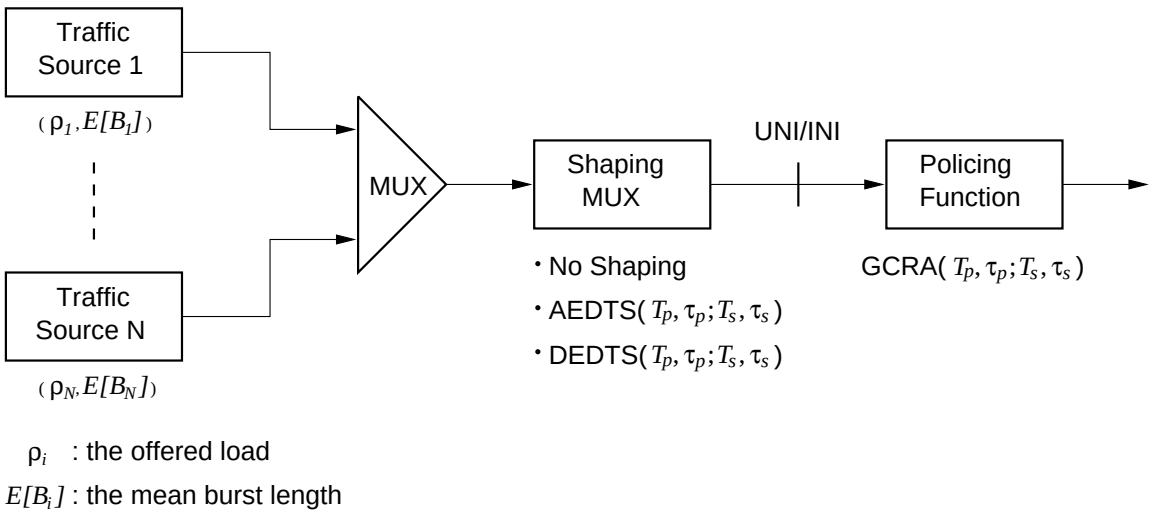



Figure 11: The simulation model.

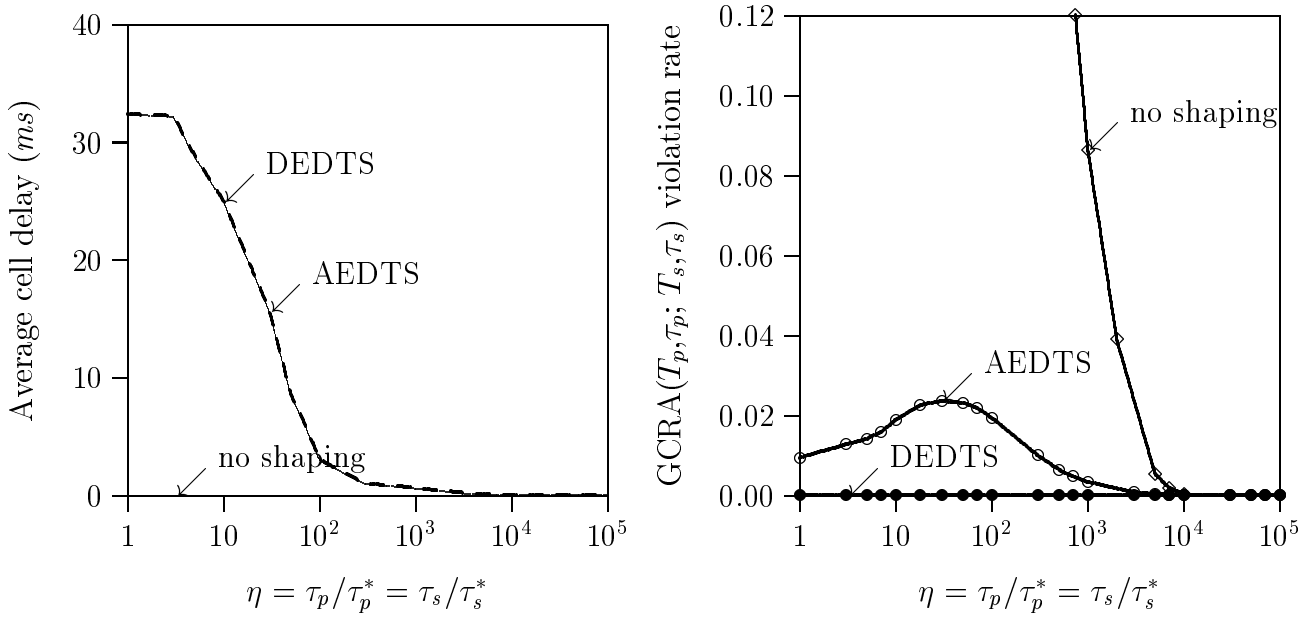



Figure 12: The average delay and GCRA($T_p$,$\tau_p$; $T_s$,$\tau_s$) violation rate.

| Type | No. | SCR($=1/T_s$) | PCR($=1/T_p$) | $\tau_s^*$ | $\tau_p^*$ |
|---|---|---|---|---|---|
| S1 | 28 | 0.005 | 0.010 | 900 | 1 |
| S2 | 12 | 0.010 | 0.040 | 525 | 1 |
| S3 | 10 | 0.010 | 0.020 | 550 | 1 |
| S4 | 8 | 0.025 | 0.050 | 80 | 1 |
| S5 | 5 | 0.050 | 0.500 | 252 | 1 |
| S6 | 1 | 0.100 | 0.500 | 152 | 1 |

Table 1: Traffic source types and paramters.