# Using Traffic Regulation To Meet End-to-End Deadlines In ATM LANs

Amitava Raha Sanjay Kamat Wei Zhao

Department of Computer Science
Texas A & M University
College Station, TX 77843-3112 *

## Abstract

This paper considers the support of hard real-time connections in ATM networks. In an ATM network, a set of hard real-time connections can be admitted only if the worst case end-to-end delays of cells belonging to individual connections are less than their deadlines. Although there are several approaches to manage the network resources in order to meet the delay requirements of connections, we focus on the use of traffic regulation to achieve this objective. Leaky buckets provide simple and user-programmable means of traffic regulation. We design and analyze an efficient optimal algorithm for selecting the burst parameters of leaky buckets to meet connections' deadlines. Our algorithm is optimal in the sense that it always selects burst parameters to meet the delay requirements of hard real-time connections whenever some such assignment exists. The exponential size of the search space makes this problem a challenging one. Our algorithm is efficient and we observe a dramatic improvement in the system performance in terms of the connection admission probability when traffic is regulated using our algorithm.

## 1 Introduction

There is a growing interest in the application of ATM LANs for distributed *Hard Real-Time* (HRT) systems. In a distributed HRT system, tasks execute at different nodes and communicate amongst themselves by exchanging messages. The messages exchanged by time-critical tasks have to be delivered by certain deadlines for successful operation of the system. Examples of such systems include supervisory command and control systems used in manufacturing, chemical processing, nuclear plants, tele-medicine, warships, etc. This paper addresses the issue of guaranteeing end-to-end deadlines of time-critical messages in ATM LANs that support distributed HRT systems.

Since ATM is a connection-oriented technology in which messages are packetized into fixed-size cells, guaranteeing message deadlines is tantamount to ensuring that the *worst case end-to-end delay* of a cell is less than its deadline. To provide such guarantees on worst case end-to-end delays, the access to various network resources must be properly managed. Three orthogonal approaches can be taken:

1. *route selection for connections;*
2. *output link scheduling at ATM switches;* and
3. *traffic regulation at the User Network Interface (UNI).*

The first approach seeks to keep the delays within bounds by selecting appropriate routes for connections such that congestion is avoided. Since typical ATM networks for HRT applications are LANs, the scope of this approach is limited. The second approach focuses on scheduling at the ATM switches where traffic from different connections is multiplexed. Drawing on the similarities with CPU scheduling, classical real-time scheduling policies such as the First Come First Serve (FCFS), Earliest Deadline First (EDF), Generalized Processor Sharing (GPS), Fair Queuing (FQ), etc., are employed [1, 10, 15]. However, to reduce the cost and the complexity of design and implementation, most of the currently available switches use simple scheduling policies such as FCFS with hardly any provision for implementing more complex scheduling policies.

In this paper, we focus on the third approach. The third approach tries to control the delays within the network by appropriately regulating the input traffic of each connection. Most of existing ATM networks provide for traffic regulation at the UNI. It is relatively easy to tune the regulation parameters as desired. This justifies our focus on using the traffic regulator as an access control mechanism for HRT ATM networks.

It must be noted, however, that all the three approaches are important and they complement each other. Although we focus on traffic regulation, our results complement the previous work on route selection and output link scheduling. In particular, we assume that proper route selection has already been done and the output link scheduling policy used is FCFS. However, our analysis and methodology can be easily extended to systems using output link scheduling policies other than FCFS.

The idea behind traffic regulation is to regulate a connection's traffic so that it has a lower impact on the delays of cells belonging to other connections. When

*The work reported in this paper was supported in part by the National Science Foundation under Grant NCR-9210583, and Texas Advanced Research program under grant 999903157.



0-8186-7216-1/95 $04.00 © 1995 IEEE

two or more connections are multiplexed on a single link, an increase in burstiness of one connection's traffic adversely impacts the delays faced by cells of others. Regulating a connection's traffic makes the traffic less bursty, thereby reducing the delays encountered by cells of other connections. However, traffic regulation has its overheads. Regulating a connection's traffic may need delaying some of the cells belonging to the connection. Thus, regulating a connection's traffic may reduce the delays of other connections possibly at the cost of increased delays for the regulated connection itself. Consequently, choosing an appropriate level of traffic regulation is an important issue in HRT systems.

In this paper, we study the impact of input traffic regulation on the worst case end-to-end delays experienced by cells of a set of hard real-time connections. In particular, we consider the leaky bucket regulator. The amount of regulation of a connection depends on the leaky bucket parameters assigned to it. The amount of traffic regulation chosen for a connection affects not only the delays of that connection, but also those of others sharing resources with that connection. Thus, the leaky bucket parameters for the entire set of connections must be carefully assigned to ensure that every connection meets its end-to-end deadline. In particular, we desire an *optimal* algorithm for selecting the leaky bucket parameters, i.e., an algorithm that *always* produces an assignment of parameters for which the end-to-end deadlines of a given set of connections are met, whenever such an assignment exists. To find such an assignment may require an exhaustive search of all the possible leaky bucket parameter assignments. The *optimal* algorithm presented in this paper is computationally efficient and can be utilized during connection setup. The results presented in this paper are directly applicable to currently available ATM networks without making any modifications to the hardware and are compatible with the ATM standards.

We evaluate the system's capability to support hard real-time connections in terms of a metric called *admission probability* [12]. *Admission probability* is the probability of meeting the end-to-end deadlines of a set of randomly chosen connections. We observe that the admission probability increases with a proper choice of leaky bucket parameters at the UNI.

While we focus on traffic regulation for meeting end-to-end deadlines, our work also complements much of the previous studies which essentially concentrate on designing and analyzing scheduling policies for ATM switches [1, 3, 4, 6, 7, 10, 14, 15].

# 2 System model

In this section, we present the network model, the connection model and the traffic descriptors used to specify the worst case traffic pattern of HRT connections.

## 2.1 Network model

In ATM networks [5], messages are packetized into fixed-size cells. We assume that time is normalized in terms of the constant transmission time of a single cell and is considered a discrete quantity with the cell transmission time being taken as one time unit.

ATM is a connection-oriented transport technology in which a connection has to be set up between two hosts before they can begin communication. In most ATM networks, the traffic is regulated at the source using leaky buckets. A *leaky bucket* regulator consists of a token bucket and an input buffer. The cells from the source associated with the leaky bucket are buffered at the leaky bucket. A pending cell from the input buffer is transmitted if at least one token is available in the token bucket. Associated with each leaky bucket regulator are two parameters, viz., the *burst parameter* and the *rate parameter*. The burst parameter, denoted by $\beta$, is the size of the token bucket, i.e., the maximum number of tokens that can be stored in the bucket. The rate parameter, denoted by $\rho$, gives the rate at which the token bucket is filled with tokens. The number of cells that may be transmitted by a leaky bucket regulator in any interval of length $I$ is bounded by $\beta + \rho * I$.

In most commercially available switches, cells of connections with stringent delay requirements (i.e., Class A traffic) are buffered in a high-priority queue and served in an FCFS order. Hence, in this paper, we consider FCFS scheduling policy for HRT connections.

## 2.2 Connection model

Typically, an HRT system operates in a *modal* fashion; i.e., the system operates in one of the several possible modes at any given instant [12]. During each mode a specific set of tasks needs to be executed for the successful operation of the system in that mode. This has an important implication for ATM networks catering to HRT systems. In order to support HRT applications, the network must *guarantee* that all cells from a given *set* of connections are transmitted before their deadlines.

We will use the following notations concerning a set of HRT connections. Hereafter, we will omit the qualifier "HRT" for connections since we will be dealing only with HRT connections.

- $N$ denotes the total number of connections in the system.
- $\mathcal{M}$ is the set of $N$ connections competing for resources within the ATM network. That is, $\mathcal{M} = \{M_1, M_2, \ldots, M_N\}$.
- $\vec{D}$ is a vector that specifies the end-to-end deadlines of connections in $\mathcal{M}$; that is, $\vec{D} = < D_1, D_2, \ldots, D_i, \ldots, D_N >$, where $D_i$ is the end-to-end deadline for a cell of connection $M_i$. That is, if a cell of $M_i$ arrives at the source at time $t$ then it should reach at the destination by $t + D_i$.

Each of the network components traversed by a connection's cells can be modeled as a server. *Thus, a connection can be considered to be a stream of cells being served by a sequence of servers* [2, 12]. Servers can be classified as *constant servers* and *variable servers*. A constant server is one that offers a fixed delay to

an arriving cell. For example, physical links are considered as constant servers. On the other hand, cells may be buffered in a variable server and hence suffer queuing delays. The leaky bucket traffic regulator and the FCFS output link schedulers at ATM switches are examples of variable servers.

The traffic pattern of a connection at a point in the network is characterized by a *traffic descriptor*. In this paper, we consider the following traffic descriptors:

- *Periodic* descriptor: This descriptor uses the notation $(C, I)$, which means that a maximum of $C$ cells belonging to the connection may arrive in any interval of length $I$. The classical *synchronous* traffic (i.e., $C$ contiguous cells arriving at the beginning of every period of length $I$) is a special case of this kind of traffic. Most hard real-time traffic (at source) is assumed to be synchronous, and hence is adequately specified by this traffic descriptor.

- *Linear* descriptor: This descriptor uses the notation $(\sigma, \rho)$, which means that a maximum of $\sigma + \rho \cdot I$ cells belonging to the connection may arrive in any interval of length $I$. This descriptor is commonly used to describe the traffic regulated by a leaky bucket server.

- *Rate function* descriptor: This descriptor uses the notation $\Gamma(I)$ to specify the maximum arrival rate of cells in any interval of length $I$. Equivalently, a maximum of $I \cdot \Gamma(I)$ cells belonging to the connection may arrive in any interval of length $I$.

The reader may note that a connection's *actual* traffic pattern may differ from that implied by the traffic descriptor used to describe the connection's traffic.

### 2.3 Delay computations

The set of connections $\mathcal{M}$ is *admissible* in an ATM network if and only if the worst case delays of cells are no more than their deadlines. Let $\vec{d}$ be a vector whose components are the worst case end-to-end delays for connections in $\mathcal{M}$; that is, $\vec{d} = < d_1, d_2, \ldots, d_i, \ldots, d_N >$, where $d_i$ is the maximum end-to-end delay experienced by a cell of connection $M_i$. Define the relation "$\leq$" on vectors as follows. Let $\vec{x} = < x_1, x_2, \ldots, x_N >$ and $\vec{y} = < y_1, y_2, \ldots, y_N >$. $\vec{x} \leq \vec{y}$ if and only if $\forall i,\ 1 \leq i \leq N,\ x_i \leq y_i$.

With this relation, $\mathcal{M}$, the set of HRT connections, is admissible if and only if

$$\vec{d} \leq \vec{D}\ . \qquad (1)$$

Hence, to check whether $\mathcal{M}$ is admissible, we need a systematic method of computing the maximum end-to-end delay experienced by a cell of each connection.

Consider connection $M_i$, $M_i \in \mathcal{M}$. To compute $d_i$, the maximum end-to-end delay experienced by a cell of $M_i$, we need to investigate the delays in every network component traversed by the connection. Let $d_i^{const}$ denote the summation of the delays a cell suffers at all the constant servers in its connection route. Let $d_i^{lb}$ denote the worst case delay experienced by a cell at the leaky bucket regulating $M_i$'s traffic. For the special case where $M_i$'s traffic is not regulated, $d_i^{lb}$ is set to 0. Let the route for connection $M_i$ consist of $K$ switches. Let $d_{i,j}^{fcfs}$ be the maximum delay experienced by a cell of connection $M_i$ at the $j^{th}$ FCFS server along the connection path. $d_i$, the worst case end-to-end delay of $M_i$, can now be obtained as

$$d_i = d_i^{const} + d_i^{lb} + d_{i,1}^{fcfs} + d_{i,2}^{fcfs} + \ldots + d_{i,K}^{fcfs}. \qquad (2)$$

Since $d_i^{const}$ is a constant, we focus on obtaining upper bounds on $d_i^{lb}$ and $d_{i,j}^{fcfs}$.

In computing $d_i^{lb}$ and $d_{i,j}^{fcfs}$, one needs information about the worst case traffic pattern of $M_i$ at various points along its connection path. In this paper, we use a procedure to compute $d_i$ for each of the three traffic descriptors that has been studied and analyzed in [13].

## 3 Problem definition

In this section, we formally define the problem of leaky bucket parameter selection for HRT ATM networks. Before proceeding to formally define the problem, we need some notations:

- $\vec{\rho}$ is the *rate vector*, i.e., $\vec{\rho} = < \rho_1, \rho_2, \ldots, \rho_i, \ldots, \rho_N >$, where $\rho_i$ is the rate parameter assigned to the leaky bucket regulating connection $M_i$ at its network interface. We assume that $\rho_i$ is assigned a value equal to the long term worst case cell arrival rate of $M_i$.

- $\vec{\beta}$ is the *burst vector*, i.e., $\vec{\beta} = < \beta_1, \beta_2, \ldots, \beta_i, \ldots, \beta_N >$, where $\beta_i$ is the burst parameter assigned to the leaky bucket regulating connection $M_i$'s traffic.

- $m(\vec{\beta})$ is defined as $m(\vec{\beta}) = \sum_{i=1}^{N} \beta_i$. Since we analyze a slotted system, $\beta_i$'s take positive integral values only. Therefore, the minimum value of $m(\vec{\beta})$ is $N$.

- $\vec{d}\ (\vec{\beta})$ is the worst case end-to-end delay vector, i.e., $\vec{d}\ (\vec{\beta}) = < d_1, d_2, \ldots, d_i, \ldots, d_N >$, where $d_i$ is the worst case cell delay of connection $M_i$ when $\vec{\beta}$ is chosen as the burst vector.

- $\mathcal{A}_{\mathcal{M}}$ is the set of *burst vectors* for which the connection set $\mathcal{M}$ is admissible, i.e., $\mathcal{A}_{\mathcal{M}} = \{\vec{\beta} \mid \vec{d}\ (\vec{\beta}) \leq \vec{D}\ \}$.

Our main goal is to meet (1), i.e., to ensure that the end-to-end deadlines of all the connections in a given set are met. We have chosen traffic regulation as our means of achieving this objective. Specifically, we desire a practical method of choosing appropriate amount of traffic regulation with leaky buckets as regulators. In terms of the above notations, given a set of HRT connections, $\mathcal{M}$, the method must find vector

$\vec{\beta}$ that belongs to $\mathcal{A}_{\mathcal{M}}$. We are going to design and analyze a $\vec{\beta}$-selection algorithm for this purpose.

Clearly, when $\mathcal{A}_{\mathcal{M}}$ is empty, no assignment of $\vec{\beta}$ can make the connection set admissible. A parameter selection algorithm is considered *optimal* if it always produces vector $\vec{\beta} \in \mathcal{A}_{\mathcal{M}}$ whenever $\mathcal{A}_{\mathcal{M}}$ is nonempty and declares the connection set to be inadmissible (say, by returning $< 0, 0, \ldots, 0 >$) if $\mathcal{A}_{\mathcal{M}}$ is empty.

Note that the definition of an optimal $\vec{\beta}$-selection algorithm allows choosing any $\vec{\beta} \in \mathcal{A}_{\mathcal{M}}$. Since burstiness of the traffic stream from $M_i$ is higher for larger values of $\beta_i$, it is desirable to select vector $\vec{\beta}$ having a small value of $m(\vec{\beta})$. It can be shown that whenever $\mathcal{A}_{\mathcal{M}}$ is nonempty, there exists a *unique* vector $\vec{\beta}^* \in \mathcal{A}_{\mathcal{M}}$ that has the smallest value of $m(\vec{\beta})$ among all vectors in $\mathcal{A}_{\mathcal{M}}$ [11]. Note that $\vec{\beta}^*$ does not exist when $\mathcal{A}_{\mathcal{M}}$ is empty.

A $\vec{\beta}$-selection algorithm is considered *strongly optimal* if it always produces $\vec{\beta}^*$ whenever $\mathcal{A}_{\mathcal{M}}$ is nonempty.

# 4 Algorithm development

In this section, we develop an efficient *strongly optimal* algorithm. We first formulate the problem as a search problem and investigate some useful properties of the search space. These properties help us in the development of the algorithm.

## 4.1 The search problem

By definition, a strongly optimal algorithm takes $\mathcal{M}$ as its input and returns $\vec{\beta}^*$ whenever $\mathcal{A}_{\mathcal{M}}$ is nonempty and returns $< 0, 0, \ldots, 0 >$ when $\mathcal{A}_{\mathcal{M}}$ is empty. We can view the problem of selecting $\vec{\beta}^*$ as a *search* problem, where the search space consists of all $\vec{\beta}$ vectors.

Let $\tilde{\mathcal{A}}_M$ be the set of *all* $\vec{\beta}$ vectors. $\mathcal{A}_{\mathcal{M}}$, the set consisting of $\vec{\beta}$ vectors with which the deadlines of $\mathcal{M}$ are met, is a subset of $\tilde{\mathcal{A}}_M$. Let $\rightarrow$ be a relation on $\tilde{\mathcal{A}}_M$ defined as follows. Given $\vec{\beta}$, $\vec{\beta}' \in \tilde{\mathcal{A}}_M$, $\vec{\beta} \rightarrow \vec{\beta}'$ if and only if $\exists j$, $1 \leq i, j \leq N$, such that

$$\beta_i' = \beta_i + 1 \quad \text{if } i = j; \tag{3}$$

and

$$\beta_i' = \beta_i \quad \text{if } i \neq j. \tag{4}$$

Note that $m(\vec{\beta}') = m(\vec{\beta}) + 1$ and $\vec{\beta}'$ differs from $\vec{\beta}$ only in the $j$th component. We let $\Delta(\vec{\beta}, \vec{\beta}')$ denote the index $j$. For example, if $\vec{\beta} = < 1, 4, 2, 7, 5 >$ and $\vec{\beta}' = < 1, 4, 2, 8, 5 >$ then $\vec{\beta} \rightarrow \vec{\beta}'$ and $\Delta(\vec{\beta}, \vec{\beta}') = 4$.

The relation $\rightarrow$ allows us to define an acyclic directed graph $G$ with a node set $\mathcal{V}$ and an edge set $\mathcal{E}$ given by

- $\mathcal{V} = \tilde{\mathcal{A}}_M$,
- $\mathcal{E} = \{ (\vec{\beta}, \vec{\beta}') \mid \vec{\beta} \rightarrow \vec{\beta}' \}$.

```
Select_Burst_Parameters(M)
01.   for p = N to ∞ do   {Main iterative loop}
02.       foreach β ∈ L_p do
03.           d = Compute_d (β);
04.           if (d ≤ D ) then
05.               return(β);
06.           endif
07.       endforeach
08.   endfor        {End of Main Loop}.
```

Figure 2: The breadth-first search method

Thus, $G$ is a graph representation of $\tilde{\mathcal{A}}_M$, the search space. Graph $G$ can also be considered as a *rooted leveled* graph; vector $< 1, 1, \ldots, 1 >$ is the root and *level* $p$ consists of all $\vec{\beta}$ vectors such that $m(\vec{\beta}') = p$. Figure 1 illustrates such a graph when $N = 3$.

In graph $G$, let $\mathcal{L}_p$ be the set of all $\vec{\beta}$ vectors from level $p$. Note that a node from level $p$ may have edges to other nodes belonging to level $p + 1$ only.

Based on this representation of the search space, a simple breadth-first search method can be constructed to find $\vec{\beta}^*$. Figure 2 shows the pseudocode for such a method. As shown by the dotted search path in Figure 1, this search method first examines all the $\vec{\beta}$ vectors in $\mathcal{L}_p$ before proceeding to all those in $\mathcal{L}_{p+1}$. For each $\vec{\beta}$ vector considered, the method uses the procedure[1] Compute_$\vec{d}$ $(\vec{\beta})$ to evaluate $\vec{d}$ $(\vec{\beta})$. The first $\vec{\beta}$ encountered in the search path that satisfies the deadline constraint $\vec{d}$ $(\vec{\beta}) \leq \vec{D}$ is clearly the $\vec{\beta}^*$ vector. At this point, the reader may raise the following questions about the method shown in Figure 2:

1. For a given connection set $\mathcal{M}$, set $\mathcal{A}_{\mathcal{M}}$ may be empty. In such a case, $\vec{\beta}^*$ is not defined and the method to locate $\vec{\beta}^*$ will not terminate.
2. Even if $\mathcal{A}_{\mathcal{M}}$ is nonempty, the exhaustive nature of the breadth-first search method results in exponential time complexity.

In the next subsection, we overcome the first difficulty by bounding the search space. In the subsequent subsection, we further reduce the search complexity by pruning the search space and adopting a search method that is more efficient than the breadth-first search.

## 4.2 Bounding the search space

The following theorem asserts the behavior that is to be expected in any ATM network. The proof of this theorem (and each of the subsequent theorems in this subsection) is not provided due to lack of space. The reader is referred to [13] for the proof.

**THEOREM 4.1** *Given any connection $M_i$, there exists a constant $\beta_i^{max}$, such that increasing $\beta_i$ beyond*

[1] For the procedure to compute the worst case end-to-end delay of connections, please refer [13].

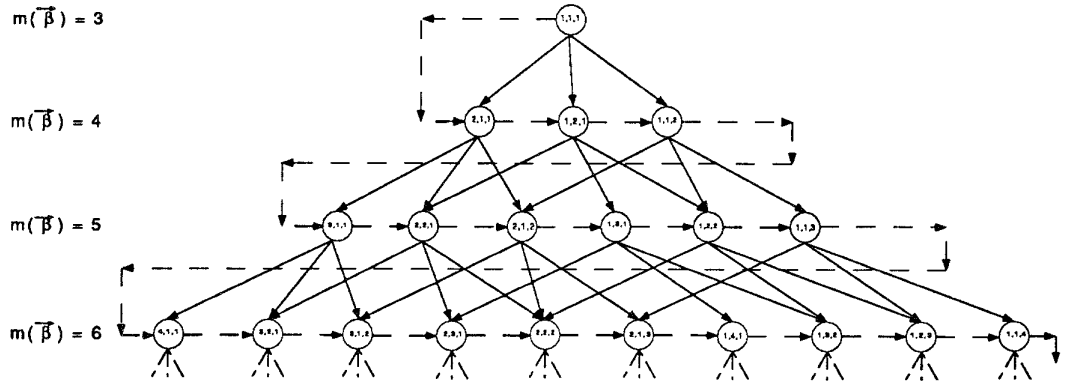


Figure 1: Example search space graph.

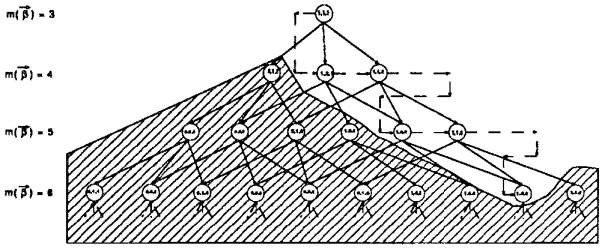

Figure 3: Example of bounded search space

Select_Burst_Parameters($\mathcal{M}$, $\vec{\beta}^{max}$)

```
01.  for p = N to m(β^max) do{Main iterative loop}
02.     foreach β ∈ L_p do
03.        d = Compute_d (β);
04.        if (d ≤ D ) then
05.           return(β);
06.        endif
07.     endforeach
08.  endfor        {End of Main Loop}
09.  return (< 0, 0, ..., 0 >).
```

Figure 4: The bounded breadth-first search algorithm

*$\beta_i^{max}$ has no impact on the worst case delays of $M_i$ or any other connection.*

An important consequence of Theorem 4.1 is that $\tilde{\mathcal{A}}_M$, the search space of candidate $\vec{\beta}$ vectors, can be bounded; we only need to consider $\vec{\beta}$ vectors that satisfy $\vec{\beta} \leq \vec{\beta}^{max}$, where $\vec{\beta}^{max} = (\beta_1^{max}, \beta_2^{max}, \ldots, \beta_N^{max})$. How $\beta_i^{max}$ may be computed from the traffic description of $M_i$ is discussed in [13]. Thus, for a given set $\mathcal{M}$, $\vec{\beta}^{max}$ can be precomputed.

Consider, the example in Figure 1. If we assume that $\vec{\beta}^{max} = < 1, 2, 3 >$, then after applying Theorem 4.1 we get another graph shown in Figure 3. The shaded region in Figure 3 is automatically eliminated from consideration.

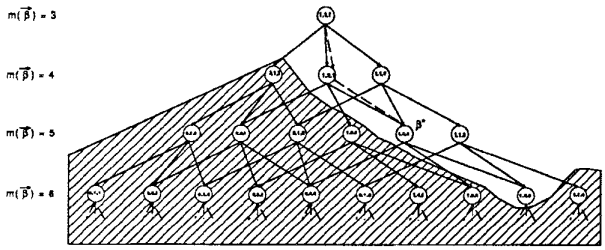

Figure 5: Example of search space after pruning.

Using Theorem 4.1, we modify the breadth-first search procedure shown in Figure 2 to take into account the bounded search space. The resulting pseudocode is shown in Figure 4. However, since the size of $\mathcal{L}_p$ increases exponentially, the complexity of the algorithm is still exponential.

In the next subsection we will prune the search space to design an efficient strongly optimal algorithm.

### 4.3 Search space pruning

The breadth-first search algorithm defined in Figure 4 has an exponential time complexity. Now we consider an alternative to the exhaustive breadth-first search method.

As an alternative to the breadth-first search path, we desire a search path that begins at the root node $< 1, 1, \ldots, 1 >$, and follows the directed edges in graph $G$ to locate $\vec{\beta}^*$ (if $\vec{\beta}^*$ exists). Such a search path would descend in a level-by-level manner examining *only one* vector $\vec{\beta}$ at each level.

Consider the example in Figure 5 which depicts a case with $N = 3$. The shaded region has already been eliminated from consideration based on $\vec{\beta}^{max}$ being $< 1, 2, 3 >$. Assume that $\vec{\beta}^* = < 1, 2, 2 >$. We would like our search method to adopt one of the two paths: $< 1, 1, 1 >$, $< 1, 2, 1 >$, $< 1, 2, 2 >$, or $< 1, 1, 1 >$, $< 1, 1, 2 >$, $< 1, 2, 2 >$.

To ensure that we reach $\vec{\beta}^*$ by adopting a search path that follows the directed edges of $G$, we must choose an appropriate *child* node at each level. For

example, in Figure 5, while at node $< 1,1,1 >$ we may either choose $< 1,2,1 >$ or $< 1,1,2 >$ as our next candidate node. However, if we are at node $< 1,1,2 >$, we must choose $< 1,2,2 >$ and cannot choose $< 1,1,3 >$ as the next candidate node.

In order to guide our selection of candidate nodes at each level, we need to know whether a particular node is an *ancestor* of $\vec{\beta}^*$ (if $\vec{\beta}^*$ indeed exists). A node $\vec{\beta}$ is said to be an ancestor of node $\vec{\beta}'$ (denoted $\vec{\beta} \dashrightarrow \vec{\beta}'$) if $\vec{\beta}'$ can be reached from $\vec{\beta}$ (by a directed path in $G$). For example, in Figure 5, the ancestor nodes of $< 1,2,2 >$ are: $< 1,2,2 >$, $< 1,2,1 >$, $< 1,1,2 >$, and $< 1,1,1 >$. To formally define the ancestor relationship, we proceed as follows.

First, each vector in $\tilde{\mathcal{A}}_M$ is considered an ancestor of itself. Let $\vec{\beta}^p$ and $\vec{\beta}^{p+k}$ be two vectors in $\mathcal{A}_M$ such that $m(\vec{\beta}^p) = p$ and $m(\vec{\beta}^{p+k}) = p+k$, $(k > 0)$. Now $\vec{\beta}^p \dashrightarrow \vec{\beta}^{p+k}$ if $\exists \vec{\beta}^{p+1}, \vec{\beta}^{p+2}, \ldots, \vec{\beta}^{p+k-1} \in \tilde{\mathcal{A}}_M$ such that $\vec{\beta}^p \rightarrow \vec{\beta}^{p+1} \rightarrow \vec{\beta}^{p+2} \rightarrow \ldots \rightarrow \vec{\beta}^{p+k-1} \rightarrow \vec{\beta}^{p+k}$.

The next theorem states an important result that will help us construct a directed path from the root node (i.e., $< 1,1,\ldots,1 >$) to $\vec{\beta}^*$ for any given $\mathcal{M}$. We need some notations first.

Let $\vec{s} =< s_1, s_2, \ldots, s_N >$ be the *status vector* associated with a node $\vec{\beta}$, where

$$s_i = \begin{cases} 1 & \text{if } d_i(\vec{\beta}) \leq D_i \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

Vector $\vec{s}$ indicates the status of each stream (i.e., whether the deadlines of individual streams are met or not) when $\vec{\beta}$ is selected as the burst parameter vector.

**THEOREM 4.2** *If $\vec{\beta}^*$ exists, then the following are true:*

1. $< 1,1,\ldots,1 >$ *is an ancestor of* $\vec{\beta}^*$.
2. $\vec{\beta}'$ *is an ancestor of* $\vec{\beta}^*$ *if* $\exists \vec{\beta}$ *such that*
   - $\vec{\beta}$ *is an ancestor of* $\vec{\beta}^*$,
   - $\vec{\beta} \rightarrow \vec{\beta}'$,
   - *and* $s_{\Delta(\vec{\beta},\vec{\beta}')} = 0$.

Now consider the claims made by the above theorem. The first claim in Theorem 4.2 is trivial. It states that if $\vec{\beta}^*$ exists, then there must be at least one path from the root node $< 1,1,\ldots,1 >$ to $\vec{\beta}^*$. The second claim in Theorem 4.2 implies that if $\vec{\beta}$ is an ancestor of $\vec{\beta}^*$ (i.e., $\vec{\beta}^*$ exists) and the assignment of $\vec{\beta}$ does not make $\mathcal{M}$ admissible (i.e., $\vec{d}\,(\vec{\beta}) > \vec{D}$ ), then another assignment $\vec{\beta}'$ derived from $\vec{\beta}$ such that $\vec{\beta} \rightarrow \vec{\beta}'$ and $s_{\Delta(\vec{\beta},\vec{\beta}')} = 0$, is also an ancestor of $\vec{\beta}^*$

```
Select_Burst_Parameters(M, β^max)
01.  β = < 1,1,...,1 >;
02.  for p = N to m(β^max) do {Main iterative loop}
03.      d = Compute_d (β);
04.      s = Compute_s (d , D );
05.      if (d ≤ D ) then
06.          return(β);
07.      else
08.          j = Find_index_j(); {Such that s_j = 0}
09.          if (β_j = β_j^max) then
10.                  return(< 0,0,...,0 >);
11.          else
12.                  β_j = β_j + 1;
             {By Theorem 4.2, the new burst vector
                  is also an ancestor of β* }
13.          endif
14.      endif
15.  endfor      {End of Main Loop}.
```

Figure 6: The algorithm

The first claim helps us to begin the search with the root node. Once we are at level $p$ examining a node $\vec{\beta} \in \mathcal{L}_p$, the second claim helps us to chose the child node of $\vec{\beta}$ if $\vec{d}\,(\vec{\beta}) > \vec{D}$ . The theorem states that we may choose a child node $\vec{\beta}'$ of $\vec{\beta}$ such that $s_{\Delta(\vec{\beta},\vec{\beta}')} = 0$. The theorem ensures that such a child node must also have a directed path to $\vec{\beta}^*$ if $\vec{\beta}^*$ exists. Hence, if $\vec{\beta}^*$ exists then it can be found by beginning our search at $< 1,1,\ldots,1 >$ and using the status vector $\vec{s}$ to select the next node along a directed path that leads us to $\vec{\beta}^*$

Using the pruning technique discussed in this subsection, we next present the strongly optimal algorithm.

## 4.4 The algorithm and its properties

In this subsection, we first present an efficient strongly optimal algorithm and then prove its properties.

Figure 6 shows the pseudocode of the algorithm. The algorithm is derived from the one in Figure 4 by pruning the search space. The algorithm is an iterative procedure. The algorithm starts from the root, i.e., with an assignment $\vec{\beta} = < 1,1,\ldots,1 >$. During each iteration the algorithm selects a node from the next level. The node is selected (line 8) with the help of status vector $\vec{s}$ (computed in line 4). This iterative process continues until either $\vec{\beta}^*$ is found or for some $j$, $\beta_j > \beta_j^{max}$.

The following theorem assert the correctness property of the algorithm from Figure 6.

**THEOREM 4.3** *For a connection set $\mathcal{M}$, the algorithm in Figure 6 is strongly optimal.*

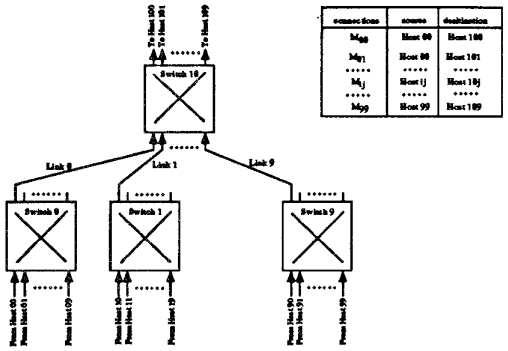

Figure 7: Example network used in simulation

Proof of this theorem follows from Theorem 4.2 and the pseudocode of the algorithm.

## 5 Performance evaluation

In this section, we present performance results to evaluate the impact of leaky bucket regulation on HRT systems.

We consider the sample network architecture shown in Figure 7. This network consists of two stages, with a total of 11, $10 \times 10$ ATM switches. The connections in the network form a symmetric pattern. There are 100 connections in the system and each connection goes through two switches. The connections are arranged in such a way that 10 connections share one output link at each stage. At the first stage, connections $M_{00}, M_{01}, \ldots M_{09}$ are multiplexed in Switch 0 and are transmitted over link 0. At the second stage, connections $M_{00}, M_{10}, \ldots M_{90}$ are multiplexed in Switch 10 and are transmitted over a link to Host 100.

Since our objective is to guarantee the deadlines of hard real-time connections, we consider the traditional HRT source traffic model, i.e., the source traffic is assumed to be *periodic* and described by the parameters $(C, P)$, where $P$ is the period of the message and $C$ is the number of cells in every message. Although the source traffic of a connection is periodic, due to multiplexing in an ATM network the periodicity of the connection traffic may no longer be maintained within the network. Therefore, we need a traffic descriptor to characterize the traffic of connections inside the network. Recall from Section 2.3, that we consider three traffic descriptors to characterize the traffic within the network. The impact of these traffic descriptors on the performance of the system will also be studied.

We evaluate the performance of the system in terms of the admission probability $AP(U)$, which is defined as the probability that a set of randomly chosen hard real-time connections can be admitted given that the average utilization of the shared links is $U$.

To obtain the performance data, we developed a program to simulate the above ATM network and the connections. For each connection, the total number of cells per period were chosen from a geometric distribution with mean 10. The rates of the connections sharing a particular link at the first stage were chosen as random variables uniformly distributed between 0

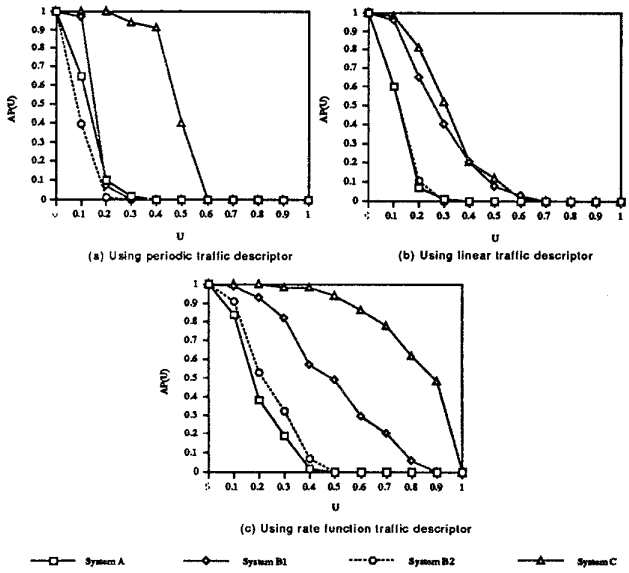


Figure 8: Admission probability ($D_i = 2P_i$).

and $U$ subject to their summation being $U$, the average utilization of the link. Similar results have been obtained with different settings of parameters. We do not present them here due to space limitations.

For each connection set generated, the following network systems are simulated:

- System A. In this system connection traffic is unregulated.
- System B. In this system constant burst vectors are used for all the connections. In particular, System B1 sets the burst vector to be $< 3, 3, \ldots, 3 >$ and System B2 sets the burst vector to be $< 9, 9, \ldots, 9 >$.
- System C. In this system the burst vector produced by our algorithm is used.

Figures 8 show the performance figure corresponding to the case where $D_i$ is set to be $2P_i$. It is common practice in a hard real-time system that deadlines are associated with periods [8, 9]. In Figure 8, sub-figures (a), (b) and (c) show the network performance with the *periodic*, *linear* and *rate function* traffic descriptor, respectively.

From these figures, we can make the following observations:

For all the traffic descriptors, System C, where our optimal algorithm is used to set the burst vectors, performs the best. Sometimes, the margin of performance improvement is significant in comparison with systems A, B1, and B2. For example, in Figure 8 (a), at $U = 0.3$, $AP(U)$ is close to 1 for System C and nearly 0 for systems A, B1, and B2. This justifies our early claim that the burst vector must be properly set in order to achieve the best system performance with hard real-time applications.

In general, we found that the admission probability is sensitive to the average link utilization. As the utilization increases the admission probability decreases. This is expected because higher the network utiliza-

tion, the more difficult it is for the system to admit a set of connections.

In comparison of the impact of traffic descriptors, the rate function traffic descriptor outperforms the periodic traffic descriptor and the linear descriptor in all the cases. This is because beyond the first stage, the rate function traffic descriptor captures the burstiness better than the other two. In fact, the data shows that the gain in performance achieved by a good selection of the burst parameter vector (System C) is offset or diminished if a poor traffic descriptor is used. However, the compound impact of traffic regulation and traffic description is much more complicated. In the case when linear or rate function traffic descriptions are used ( Figures 8 (b) and (c)), systems B1 and B2 perform better than System A. However, in Figure 8 (a), i.e., when the periodic traffic descriptor is used, System A performs better than systems B1 and B2 for higher values of $U$. For a detailed analysis of this phenomenon, see [13]. Generally speaking, these results further strengthen the need for a good $\vec{\beta}$ selection algorithm. If the burst vector is not appropriately selected, the system may perform worse that the one where no traffic regulation is conducted.

## 6 Conclusions

In this paper we addressed the issue of guaranteeing end-to-end deadlines of hard real-time connections in an ATM network. Much of the previous work has concentrated on scheduling policies used in ATM switches. Our approach to this problem involved regulating the input traffic at the network interface. In particular, we consider leaky bucket traffic regulators. This study is the first one that uses traffic regulation (in particular with leaky buckets) as a method of guaranteeing the end-to-end deadlines of hard real-time connections.

We design and analyze an efficient optimal algorithm for selecting the burst parameters of leaky buckets in order to meet connections' deadlines. Our algorithm is optimal in the sense that if there exists an assignment of burst parameters for which cells of hard real-time connections can meet their deadlines then the algorithm will always find such an assignment. Our algorithm is efficient and has a polynomial time complexity. We analyzed and compared the performance of ATM networks with FCFS scheduling policy under different loading conditions and using different traffic description methods. The performance of the network was measured in terms of admission probability. We observed that independent of the traffic descriptor used, there is a dramatic improvement in the system performance when the burst parameters were selected by our algorithm.

Our solution for guaranteeing end-to-end deadlines in HRT ATM networks is effective and efficient. It can be used for admission control in any ATM network that uses leaky bucket traffic regulators. Further, since our approach is independent of the switch architecture and the scheduling policy used at the ATM switches, it complements the performance gain achieved by any scheduling policy.

This is a preliminary study on real-time communications over ATM networks. Many extensions are possible. In [13], we consider further pruning of the search space that can be achieved by accounting for the buffer space limitations at leaky buckets. We are at present studying the performance gain that can be attained when our leaky bucket parameter selection algorithm is used with scheduling policies other than FCFS. Further, our work can be extended to encompass other traffic regulation mechanisms such as dual leaky buckets and sliding windows.

## References


[1] D. D. Clark, S. Shenker, and L. Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. In *Proc. of ACM SIGCOMM'92*, pg. 14–26, Aug. 1992.

[2] R. L. Cruz. A calculus for network delay. *IEEE Trans. on Inf. Th.*, 37(1):114–141, Jan. 1991.

[3] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *Proc. of ACM SIGCOMM'89*, pg. 1–12, Sept. 1989.

[4] S. J. Golestani. A framing strategy for congestion management. *JSAC*, 9(7):1064–1077, Sept. 1991.

[5] International Telecommunication Union. *ITU-T Recommendation I.311 — B-ISDN General Network Aspects*, 1993.

[6] C. R. Kalmanek, H. Kanakia, and S. Keshav. Rate controlled servers for very high-speed networks. In *Proc. of IEEE Global Tele. Conf.*, pg. 300.3.1–300.3.9, Dec. 1990.

[7] D. D. Kandlur, K. G. Shin, and D. Ferrari. Real-time communication in multi-hop networks. In *Proc. of the 11th Int. Conf. on Dist. Comp. Syst.*, pg. 300–307, May 1991.

[8] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *JACM*, 20(1):46–61, Jan. 1973.

[9] N. Malcolm and W. Zhao. Hard real-time communication in multiple-access networks. *Journal of Real-Time Systems*, Jan. 1995.

[10] A. K. J. Parekh. *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks.* PhD thesis, Dpt. of EECS, MIT, 1992.

[11] A. Raha. *Real Time Communication in ATM Networks.* PhD thesis, Dpt. of CS, Texas A&M University, 1995. In preparation.

[12] A. Raha, S. Kamat, and W. Zhao. Guaranteeing end-to-end deadlines in ATM networks. 1995. In *Proceedings of the 15th IEEE Int. Conf. on Dist. Comp. Syst.*, May 1995.

[13] A. Raha, S. Kamat, and W. Zhao. Using traffic regulation to meet end-to-end deadlines in atm networks. Technical Report TAMU 95-016, Dpt. of CS, Texas A&M University, Mar. 1995.

[14] H. Zhang and D. Ferrari. Rate-controlled static priority queueing. In *Proc. of IEEE Infocom'93*, pg. 227–236, Mar. 1993.

[15] L. Zhang. Virtual clock: A new traffic control algorithm for packet switching networks. In *Proc. of ACM SIGCOMM'90*, pg 19–29, Sept. 1990.