Using Traffic Regulation To Meet End-to-End Deadlines In ATM Networks

Amitava Raha, Sanjay Kamat and Wei Zhao
Department of Computer Science
Texas A&M University
College Station, Texas 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 achieves a polynomial time complexity by systematically pruning the search space. We observe a dramatic improvement in the system performance in terms of the connection admission probability when traffic is regulated using our algorithm.

Contents

1 Introduction

There is a growing interest in the application of ATM networks 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 networks 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;*
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' output links 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, 13, 20, 21, 22]. However, to reduce the cost and the complexity of design and implementation, most of the currently available switches use simple scheduling policies such as First Come First Serve (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. By regulating the input traffic, its burstiness can be controlled. This tends to reduce the adverse impact of burstiness on the end-to-end delays of other connections. 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 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* [15]. *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, 4, 5, 7, 9, 10, 13, 17, 18, 20, 21, 22, 23]. A modified FCFS scheduling scheme called $FIFO^+$ was proposed and studied in [1]. The switch scheduling policy called "Stop and Go" is presented in [7]. A virtual clock scheduling scheme in which cells are prioritized by a virtual time stamp assigned to them, is discussed in [22]. The use of the Earliest Deadline First scheduling in wide area networks has also been investigated [5, 23]. [20] discusses scheduling at the output link by introducing a regulator at each link of an ATM switch. Scheduling policies based on fair queuing and its derivations are discussed in [4, 13].

The outline of the rest of the paper is as follows. Section 2 defines the system model. Section 3 develops a formal definition of the traffic regulation problem for HRT ATM networks. Section 4 presents our algorithm to select the leaky bucket parameter values. Performance results are presented in Section 5. We conclude in Section 6 with a summary of results and a discussion of possible extensions.

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

Figure 1 shows a typical ATM LAN. In ATM networks [3, 8, 19], messages are packetized into fixed-size cells. The time to transmit a single cell is a constant denoted by CT. We assume that time is normalized in terms of CT. That is, in this paper time is considered a discrete quantity with the cell transmission time (CT) being taken as one time unit.

ATM is a connection-oriented transport technology. Before two hosts begin communication, a connection has to be set up between the two hosts. Figure 2(a) shows a sequence of network components that constitute a typical connection (illustrated by a connection from Host 1 to Host 2 in Figure 1). Cells belonging to a connection pass through a traffic regulator at the entrance to the network (the User Network Interface or UNI) and then traverse one or more ATM switches interconnected by physical links before reaching their destination host.

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 β, 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 ρ, 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 $\beta + \rho * I$.

An ATM switch multiplexes a number of connections onto a physical link. The cells of connections being multiplexed are buffered at the output link. 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

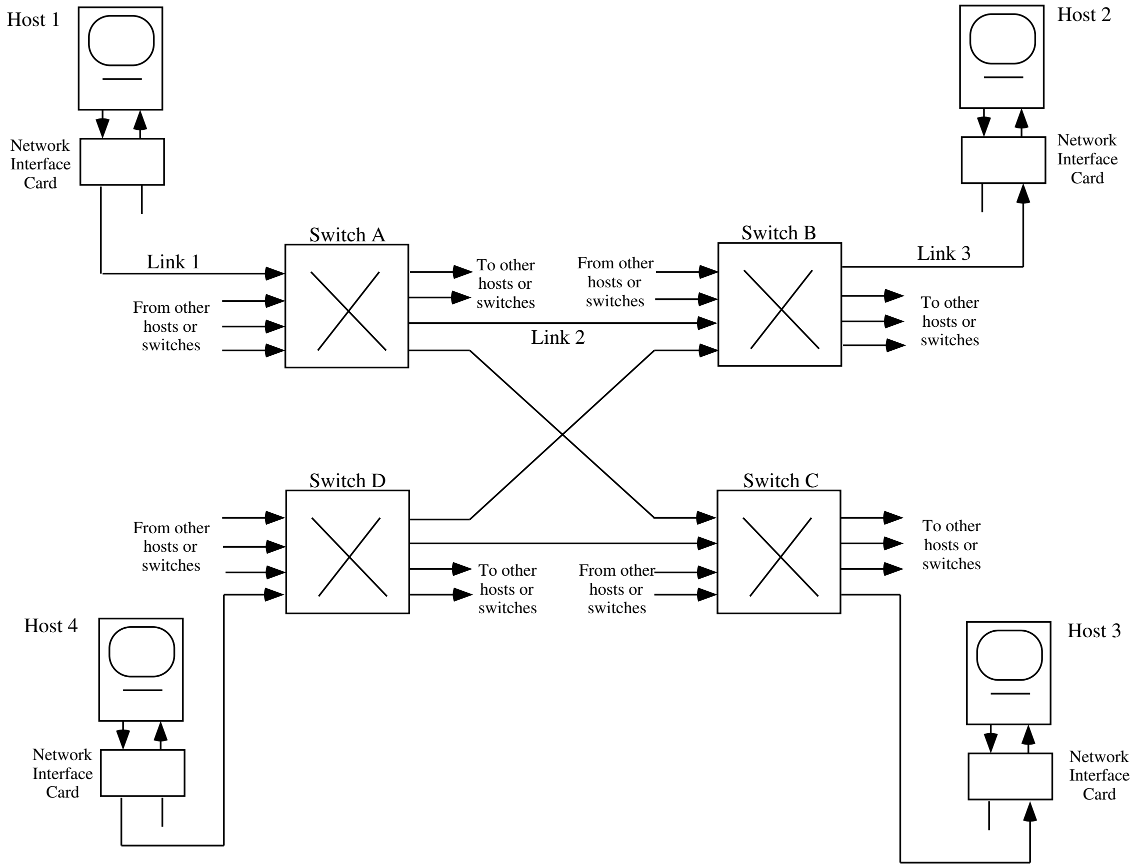


Figure 1: ATM network architecture

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 [15]. A specific set of tasks needs to be executed for the successful operation of the system in a given mode. The set of tasks changes as the system operation mode changes. 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.

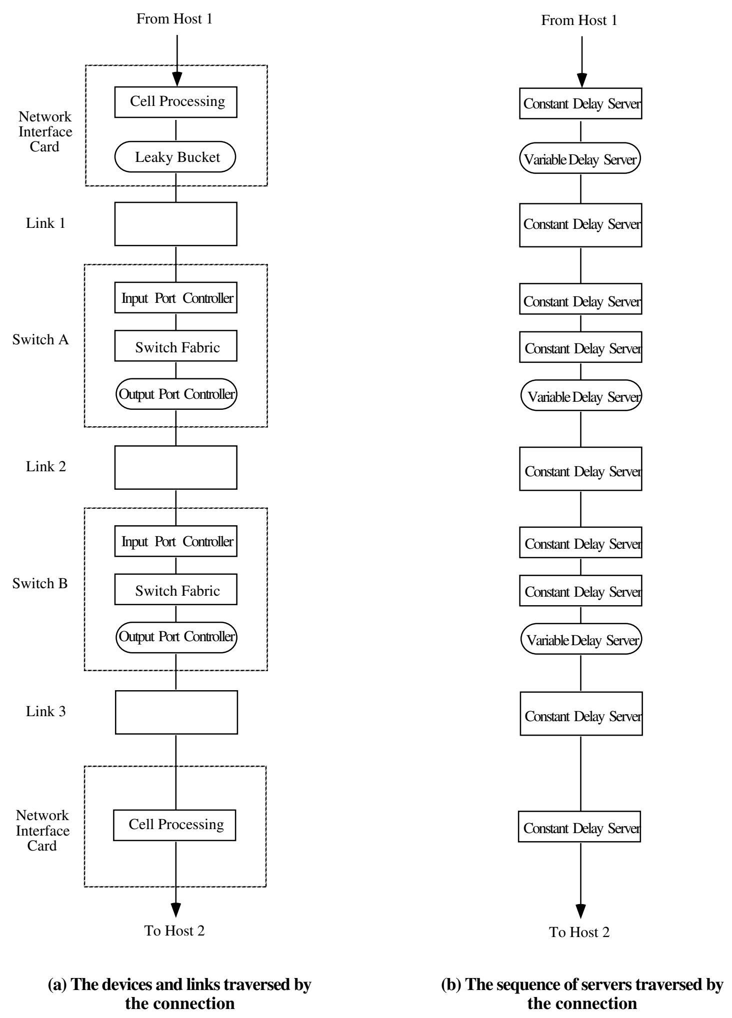


Figure 2: Connection decomposition into servers

- 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.[1] That is,

$$\mathcal{M} = \{M_1, M_2, \ldots, M_i, \ldots, M_N\} \tag{1}$$

where connection M_i is specified by the following information:

  - *Source address*,
  - *Destination address*,

[1] In this paper sets will be denoted using calligraphic font and vectors using boldface font.

- *Connection route*,[2]
- *End-to-end deadline* (D_i),
- *An upper bound on the long term cell arrival rate,*
- *Worst case input traffic characteristics.*

- $\vec{\boldsymbol{D}}$ is a vector that specifies the end-to-end deadlines for connections in $\mathcal{M}$; that is,

$$\vec{\boldsymbol{D}} = < D_1, D_2, \ldots, D_i, \ldots, D_N >, \tag{2}$$

  where D_i is the end-to-end deadline for a cell of connection M_i. That is, if a cell arrives at the source at time t then it should reach at the destination by $t + D_i$.

In a connection (see Figure 2(a)), 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, 15]. Servers can be classified as *constant delay servers* and *variable delay servers.* A constant delay server is one that offers a fixed delay to an arriving cell. For example, physical links are considered as constant delay servers. On the other hand, cells may be buffered in a variable delay server and hence suffer queuing delays. The leaky bucket traffic regulator and the FCFS output link schedulers at ATM switches are examples of variable delay servers. Figure 2(b) shows the logical representation of the connection in Figure 2(a).

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 (σ, ρ), 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]To ensure system stability, we assume that connection routes do not form loops [2, 13, 14].

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 $\vec{\boldsymbol{d}}$ be a vector whose components are the worst case end-to-end delays for connections in $\mathcal{M}$; that is,

$$\vec{\boldsymbol{d}} = < d_1, d_2, \ldots, d_i, \ldots, d_N >, \tag{3}$$

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 $\vec{\boldsymbol{x}} = < x_1, x_2, \ldots, x_N >$ and $\vec{\boldsymbol{y}} = < y_1, y_2, \ldots, y_N >$. $\vec{\boldsymbol{x}} \leq \vec{\boldsymbol{y}}$ if and only if

$$\forall i,\ 1 \leq i \leq N,\ \ x_i \leq y_i. \tag{4}$$

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

$$\vec{\boldsymbol{d}} \leq \vec{\boldsymbol{D}}\ . \tag{5}$$

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. Therefore, we first present an appropriate abstraction of a connection.

Now, we can decompose d_i, the maximum end-to-end delay experienced by a cell of M_i into three parts as follows:

1. d_i^{const} denotes the summation of the delays a cell suffers at all the constant delay servers in its connection route.

2. d_i^{lb} denotes 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.

3. d_i^{net} denotes the summation of the worst case delays a cell suffers at all the variable delay servers after the leaky bucket. d_i^{net} can be obtained by decomposing it further as follows. Let the route for connection M_i consist of n 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^{net} can be obtained as the sum of $d_{i,j}^{fcfs}$s; i.e,

$$d_i^{net} = d_{i,1}^{fcfs} + d_{i,2}^{fcfs} + \ldots + d_{i,n}^{fcfs}. \tag{6}$$

d_i, the end-to-end delay for M_i, can now be obtained as

$$\begin{aligned} d_i &= d_i^{const} + d_i^{lb} + d_i^{net} \\ &= d_i^{const} + d_i^{lb} + d_{i,1}^{fcfs} + d_{i,2}^{fcfs} + \ldots + d_{i,n}^{fcfs}. \end{aligned} \tag{7}$$

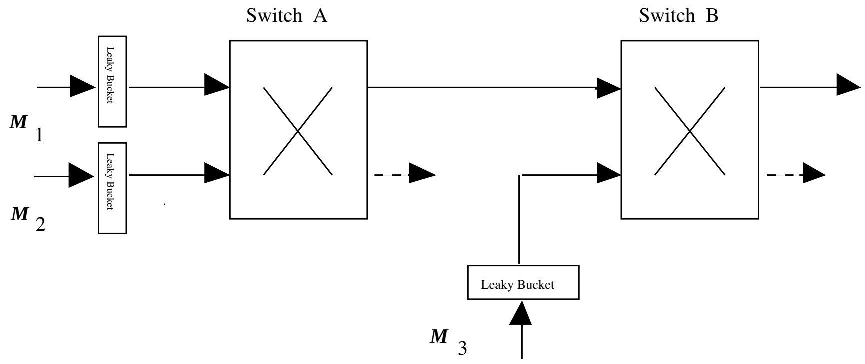


Figure 3: Experimental setup

Since d_i^{const} is a constant, we focus on obtaining upper bounds on d_i^{lb} and d_i^{net}.

In computing d_i^{lb} and $d_{i,j}^{fcfs}$s, one needs information about the worst case traffic pattern of M_i at various points along its connection path. Since traffic descriptors provide only approximate information about the actual traffic, d_i computed using different traffic descriptors vary in their conservativeness. 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 [16]. While the primary objective of this paper is to study the performance gains obtained through traffic regulation, we also evaluate the impact of traffic descriptors on system performance.

3 Problem definition

In this section, we formally define the problem of leaky bucket parameter selection for HRT ATM networks. To motivate the discussion, we first examine some experimental data. We consider a simple network consisting of two ATM switches (see Figure 3). Each ATM switch has 2 input lines and 2 output lines. There are 3 connections in the system, M_1, M_2, and M_3, all of which share the same output link at the second switch. Connections M_1 and M_2 enter the network at switch A and traverse through switches A and B, while connection M_3 traverses the switch B only. All the three connections carry identical streams of video data. The video source under study is a two hour encoding of the movie "Starwars" [6]. The video source provides a total of 171,000 frames, at the rate of 24 frames per second.

In this experiment we vary β_1 (the leaky bucket burst parameter of M_1) while keeping β_2 and β_3 (the leaky bucket burst parameters of M_2 and M_3, respectively) constant. Figure 4 shows that as β_1 increases, the delay (measured in CT units) experienced by M_1 tends to decrease. However, when β_1 is increased, the delay experienced by M_3 tends to increase. Further, we observe that for $\beta_1 > 1400$, the delays experienced by M_1 and M_3 reach constant values. An intuitive explanation of these results is provided below.

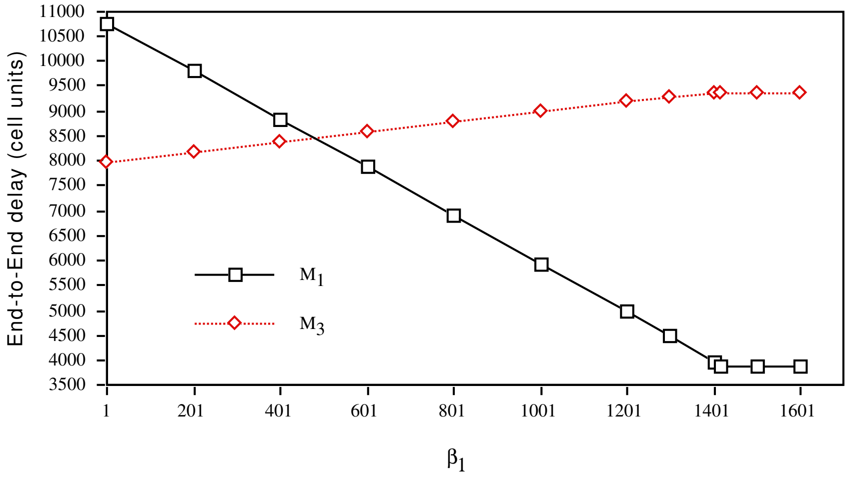


Figure 4: Experimental results

An increase in β_1 tends to increase the burst size of M_1's traffic into the network. When a larger burst size is allowed at the output of M_i's leaky bucket, the need to buffer M_i's cells within the leaky bucket decreases. This tends to lower connection M_1's delay as β_1 is increased. On the other hand, the increased burstiness of M_1 as β_1 is increased adversely impacts connection M_3's traffic, increasing M_3's worst case cell delay. However, for large values of β_1 ($\beta_1 > 1400$), the delays of either connection are unaffected by any increase in β_1. This is because, at such large values of β_1, the burst size allowed for M_1 is so high that no cells of M_1 are queued at the leaky bucket input buffer, i.e., there is virtually no traffic regulation for connection M_1.

Clearly, the experimental results shown in Figure 4 indicate that the choice of β_is for different connections plays a critical role in the worst case cell delays suffered by the connections. Since we are interested in guaranteeing the deadlines of *every* connection in $\mathcal{M}$, we have to carefully choose β_i's.

Before proceeding to formally define the problem, we need some notations:

- $\vec{\rho}$ is the *rate vector*, i.e.,
$$\vec{\rho} = < \rho_1, \rho_2, \ldots, \rho_i, \ldots, \rho_N >, \quad (8)$$
where ρ_i is the rate parameter assigned to the leaky bucket regulating connection M_i at its network interface. We assume that ρ_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.,
$$\vec{\beta} = < \beta_1, \beta_2, \ldots, \beta_i, \ldots, \beta_N >, \quad (9)$$
where β_i is the burst parameter assigned to the leaky bucket regulating connection M_i's traffic.

- $m(\vec{\beta})$ is the *norm* of the *burst vector* $\vec{\beta}$, i.e.,

$$m(\vec{\beta}) = \sum_{i=1}^{N} \beta_i. \tag{10}$$

  Since we analyze a slotted system, β_i's take positive integral values only. Therefore, the minimum value of $m(\vec{\beta})$ is N.

- $\vec{\boldsymbol{d}}\ (\vec{\beta})$ is the worst case end-to-end delay vector, i.e.,

$$\vec{\boldsymbol{d}}\ (\vec{\beta}) = < d_1, d_2, \ldots, d_i, \ldots, d_N >, \tag{11}$$

  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.,

$$\mathcal{A}_{\mathcal{M}} = \{\vec{\beta} \mid \vec{\boldsymbol{d}}\ (\vec{\beta}) \leq \vec{\boldsymbol{D}}\ \}. \tag{12}$$

Our main goal is to meet (5), i.e., 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. Such an algorithm will take connection set $\mathcal{M}$ as its input and return vector $\vec{\beta}$ as its output.

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 β_i, it is desirable to select vector $\vec{\beta}$ having a small norm, $m(\vec{\beta})$. 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 norm among all vectors in $\mathcal{A}_{\mathcal{M}}$ [14]. 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.

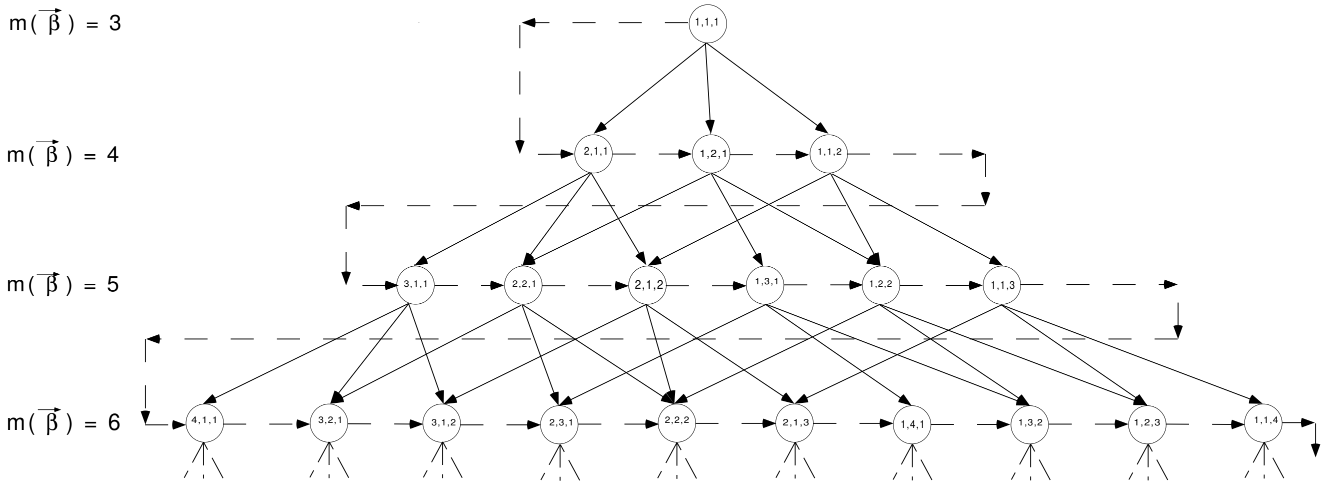


Figure 5: Example search space graph.

4.1 Parameter selection as a 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 $< 0, 0, \ldots, 0 >$ as its output 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 the $\vec{\beta}$ 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 $\longrightarrow$ be a relation on $\tilde{\mathcal{A}}_M$ defined as follows. Given $\vec{\beta}$, $\vec{\beta}' \in \tilde{\mathcal{A}}_M$, $\vec{\beta} \longrightarrow \vec{\beta}'$ if and only if $\exists j,\ 1 \le j \le N$, such that

$$\beta'_i \;=\; \beta_i + 1 \quad \text{if } i = j; \tag{13}$$

and

$$\beta'_i \;=\; \beta_i \quad \text{if } i \neq j. \tag{14}$$

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

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

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

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 having norm p. Figure 5 illustrates such a graph when $N = 3$.

```
Select_Burst_Parameters(M)
    for p = N to ∞ do                    {Main iterative loop}
        foreach β⃗ ∈ L_p do
            d⃗ = Compute_d⃗ (β⃗);
            if (d⃗ ≤ D⃗ ) then
                return(β⃗);
            endif
        endforeach
    endfor        {End of Main Loop}.
```

Figure 6: Pseudocode of the breadth-first search method

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 6 shows the pseudocode for such a method. As shown by the dotted search path in Figure 5, 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[3] 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 6:

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

In Section 3, we described an experiment with three video connections. An interesting observation was that when β_1 was increased beyond 1400, there was no change in the worst case end-to-end delays of any of the connections. The following theorem asserts that such behavior is to be expected

[3]For the procedure to compute the worst case end-to-end delay of connections, please refer [16].

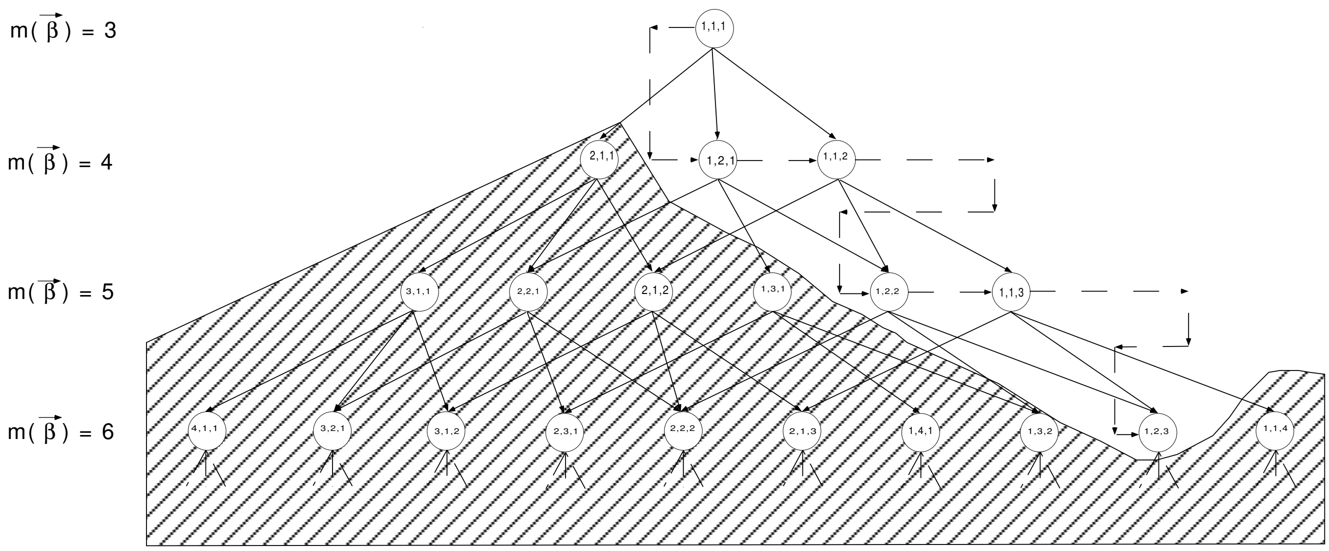


Figure 7: Example of bounded search space

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 [16] for the proof.

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

$$\vec{\beta} \leq \vec{\beta}^{max}, \tag{15}$$

where $\vec{\beta}^{max} = (\beta_1^{max}, \beta_2^{max}, \ldots, \beta_N^{max})$. How β_i^{max} may be computed from the traffic description of M_i is discussed in [16]. Thus, for a given set $\mathcal{M}$, $\vec{\beta}^{max}$ can be precomputed.

Consider, the example in Figure 5. If we assume that $\vec{\beta}^{max} = <1, 2, 3>$, then after applying Theorem 4.1 we get another graph shown in Figure 7. The shaded region in Figure 7 is automatically eliminated from consideration.

Using Theorem 4.1, we modify the breadth-first search procedure shown in Figure 6 to take into account the bounded search space. The resulting pseudocode is shown in Figure 8. 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.

Select_Burst_Parameters($\mathcal{M}$, $\vec{\beta}^{max}$)
for $p = N$ **to** $m(\vec{\beta}^{max})$ **do** {Main iterative loop}
  foreach $\vec{\beta} \in \mathcal{L}_p$ **do**
    $\vec{d}$ = Compute_$\vec{d}$ ($\vec{\beta}$);
    if ($\vec{d} \leq \vec{D}$) **then**
      return($\vec{\beta}$);
    endif
  endforeach
endfor {End of Main Loop}
return ($< 0, 0, \ldots, 0 >$).

Figure 8: Pseudocode of the bounded breadth-first search algorithm

4.3 Search space pruning

The breadth-first search algorithm defined in Figure 8 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 9 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. $< 1, 1, 1 >$, $< 1, 2, 1 >$, $< 1, 2, 2 >$, or
2. $< 1, 1, 1 >$, $< 1, 1, 2 >$, $< 1, 2, 2 >$.

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 9, 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 $\vec{\beta} \xrightarrow{*} \vec{\beta}'$) if $\vec{\beta}'$ can be reached from $\vec{\beta}$ (by a directed path in G). For example, in Figure 9, 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.

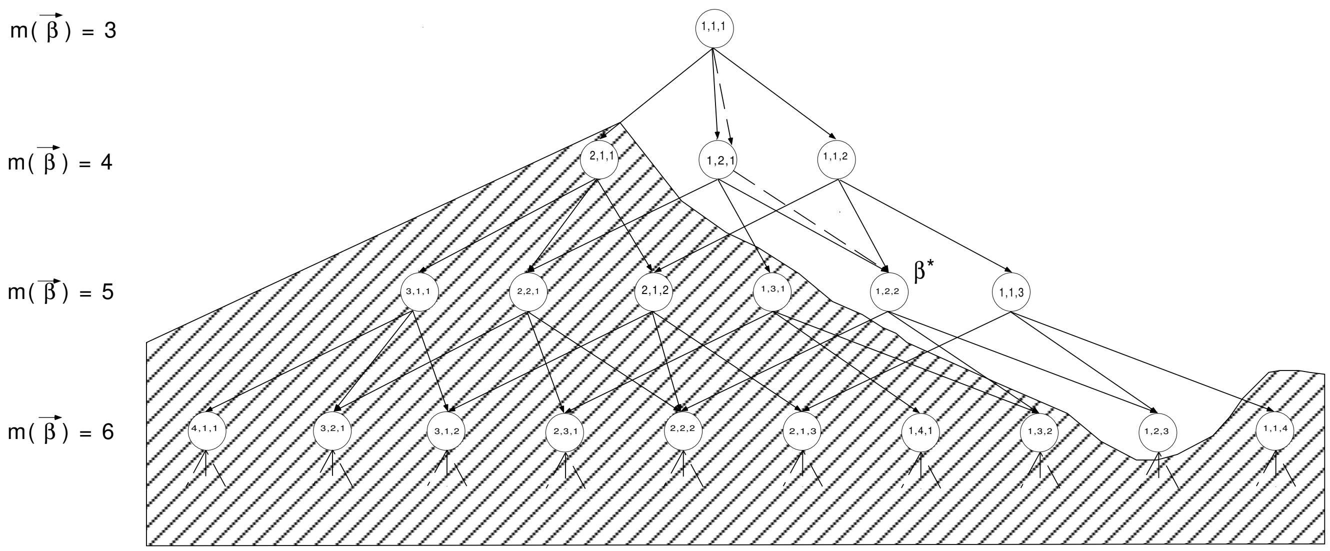


Figure 9: Example graph of search space after pruning.

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 $\vec{\beta}^p \xrightarrow{*} \vec{\beta}^{p+k}$ if $\exists \vec{\beta}^{p+1}, \vec{\beta}^{p+2}, \ldots, \vec{\beta}^{p+k-1} \in \tilde{\mathcal{A}}_M$ such that $\vec{\beta}^p \longrightarrow \vec{\beta}^{p+1} \longrightarrow \vec{\beta}^{p+2} \longrightarrow \ldots \longrightarrow \vec{\beta}^{p+k-1} \longrightarrow \vec{\beta}^{p+k}$.

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

$$s_i = \begin{cases} 1 & \text{if } d_i(\vec{\beta}) \le D_i \\ \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

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* $\vec{\beta}^*$,
   - $\vec{\beta} \longrightarrow \vec{\beta}'$,
   - *and* $s_{\triangle(\vec{\beta}, \vec{\beta}')} = 0$.

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 $\vec{\beta} \longrightarrow \vec{\beta}'$ and $s_{\triangle(\vec{\beta},\vec{\beta}')} = 0$, is also an ancestor of $\vec{\beta}^*$

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 $s_{\triangle(\vec{\beta},\vec{\beta}')} = 0$. 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 efficient algorithm and its properties

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

Figure 10 shows the pseudocode of the algorithm. The algorithm is derived from the one in Figure 8 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, $\beta_j > \beta_j^{max}$.[4]

The following two theorems assert the correctness property and the complexity of the algorithm from Figure 10.

THEOREM 4.3 *For a connection set $\mathcal{M}$, the algorithm in Figure 10 is strongly optimal.*

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

THEOREM 4.4 *The time complexity of the algorithm of Figure 10 is $O(N \sum_{i=1}^{N} \beta_i^{max})$.*

Proof: In the algorithm shown in Figure 10, the maximum number of iterations is $\sum_{i=1}^{N} \beta_i^{max} - N + 1$. During each iteration the algorithm calls three procedures (lines 3, 4, and 8). The worst

[4]Note that line 8 in the algorithm can be modified to select the connection whose deadline is missed and for which $(\beta^{max} - \beta)$ has the minimum value. This modification improves the average case time complexity of the algorithm without changing the worst case one.

```
Select_Burst_Parameters($\mathcal{M}$, $\vec{\beta}^{max}$)
    $\vec{\beta} = <1, 1, \ldots, 1>$;
    for $p = N$ to $m(\vec{\beta}^{max})$ do                {Main iterative loop}
        $\vec{d}$ = Compute_$\vec{d}$ ($\vec{\beta}$);
        $\vec{s}$ = Compute_$\vec{s}$ ($\vec{d}$, $\vec{D}$);
        if ($\vec{d} \leq \vec{D}$) then
            return($\vec{\beta}$);
        else
            $j$ = Find_index_$j$();          {Such that $s_j = 0$}
            if ($\beta_j = \beta_j^{max}$) then
                return($<0, 0, \ldots, 0>$);
            else
            {Increasing component $j$ by 1}
                $\beta_j = \beta_j + 1$;
                {By Theorem 4.2, the new burst vector
                is also an ancestor of $\vec{\beta}^*$}
            endif
        endif
    endfor        {End of Main Loop}.
```

Figure 10: Pseudocode of the efficient algorithm

case time complexity of the procedure Compute_$\vec{d}$ ($\vec{\beta}$) (line 3) is a function of the network size, i.e., the longest path in the network. Hence, for a given network the time complexity of Compute_$\vec{d}$ ($\vec{\beta}$) can be bounded by a constant. The procedure Compute_$\vec{s}$ ($\vec{d}$, $\vec{D}$) (line 4) requires N steps of comparisons, therefore its time complexity is $O(N)$. Finally, the worst case time complexity of the procedure Find_index_j() (line 8) is $O(N)$.

Hence, the time complexity of the algorithm of Figure 10 is $O(N \sum_{i=1}^{N} \beta_i^{max})$. □

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 11. This network consists of two stages, with a total of 11 ATM switches. Each ATM switch has 10 input lines and 10 output lines. 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 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. The program was written in the C programming language and run in a Sun/Solaris environment. In each run of the program 200 connection sets were randomly generated. 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 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.

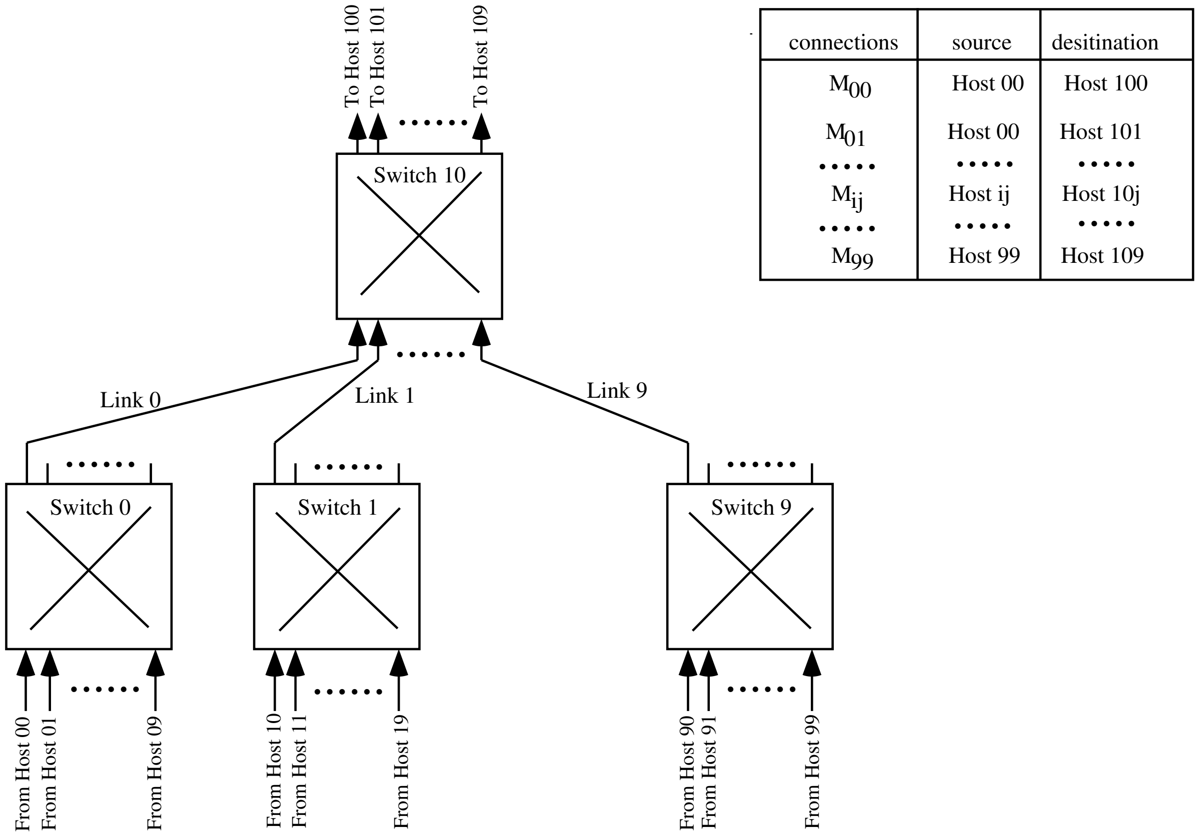


Figure 11: Example network used in simulation

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

- System A. In this system connection traffic is unregulated, i.e., the burst vector selected for the connection set is $\vec{\beta}^{max}$.

- 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 the efficient strongly optimal algorithm is used.

Figures 12 and 13 show the performance figures corresponding to the cases where D_i is set to be $2P_i$ and $1.5P_i$ respectively. It is common practice in a hard real-time system that deadlines are associated with periods [11, 12]. In Figures 12 and 13, 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 12 (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 utilization, 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 12 (b) and (c) and Figures 13 (b) and (c)), systems B1 and B2 perform better than System A. However, in Figure 12 (a) and Figure 13 (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 [16]. 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. Traditionally, a leaky bucket has been used as a policing mechanism to intervene when the source traffic at the input of the network does not conform to its negotiated characteristics.

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 [16], 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. Some commercial switches provide leaky buckets at the switch, making traffic regulation feasible within the network. It would be interesting to study the performance gain that can be attained by tuning the leaky bucket parameters both at the UNI and at the ATM switches. Another interesting area is soft real-time systems. For such systems, traffic regulation can be used not only to improve the worst case delays of a cell but also to reduce its loss probability.

References

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

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

[3] M. de Prycker. *Asynchronous Transfer Mode: Solution for Broadband ISDN.* Ellis Horwood, 1991.

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

[5] D. Ferrari and D. C. Verma. A scheme for real-time channel establishment in wide-area networks. *IEEE Journal on Selected Areas in Communications*, SAC-8(3):368–379, Apr. 1990.

[6] M. Garrett. *Contributions toward real-time services on packet switched networks.* PhD thesis, Columbia University, 1993.

[7] S. J. Golestani. A framing strategy for congestion management. *IEEE Journal on Selected Areas in Communications*, 9(7):1064–1077, Sept. 1991.

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

[9] C. R. Kalmanek, H. Kanakia, and S. Keshav. Rate controlled servers for very high-speed networks. In *Proceedings of IEEE Global Telecommunications Conference*, pages 300.3.1–300.3.9, Dec. 1990.

[10] D. D. Kandlur, K. G. Shin, and D. Ferrari. Real-time communication in multi-hop networks. In *Proceedings of the 11th International Conference on Distributed Computing Systems*, pages 300–307, May 1991.

[11] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the Association for Computing Machinery*, 20(1):46–61, Jan. 1973.

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

[13] A. K. J. Parekh. *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1992.

[14] A. Raha. *Real Time Communication in ATM Networks.* PhD thesis, Department of Computer Science, Texas A M University, 1995. In preparation.

[15] A. Raha, S. Kamat, and W. Zhao. Guaranteeing end-to-end deadlines in ATM networks. In *Proceedings of the 15th IEEE International Conference on Distributed Computing Systems*, June 1995.

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

[17] S. S. Sathaye, W. S. Kish, and J. K. Strosnider. Responsive aperiodic services in high-speed networks. In *Proceedings of the IEEE International Conference on Distributed Computing Systems*, pages 337–346, May 1993.

[18] L. Trajkovic and S. J. Golestani. Congestion control for multimedia services. *IEEE Network*, 6(5):20–26, Sept. 1992.

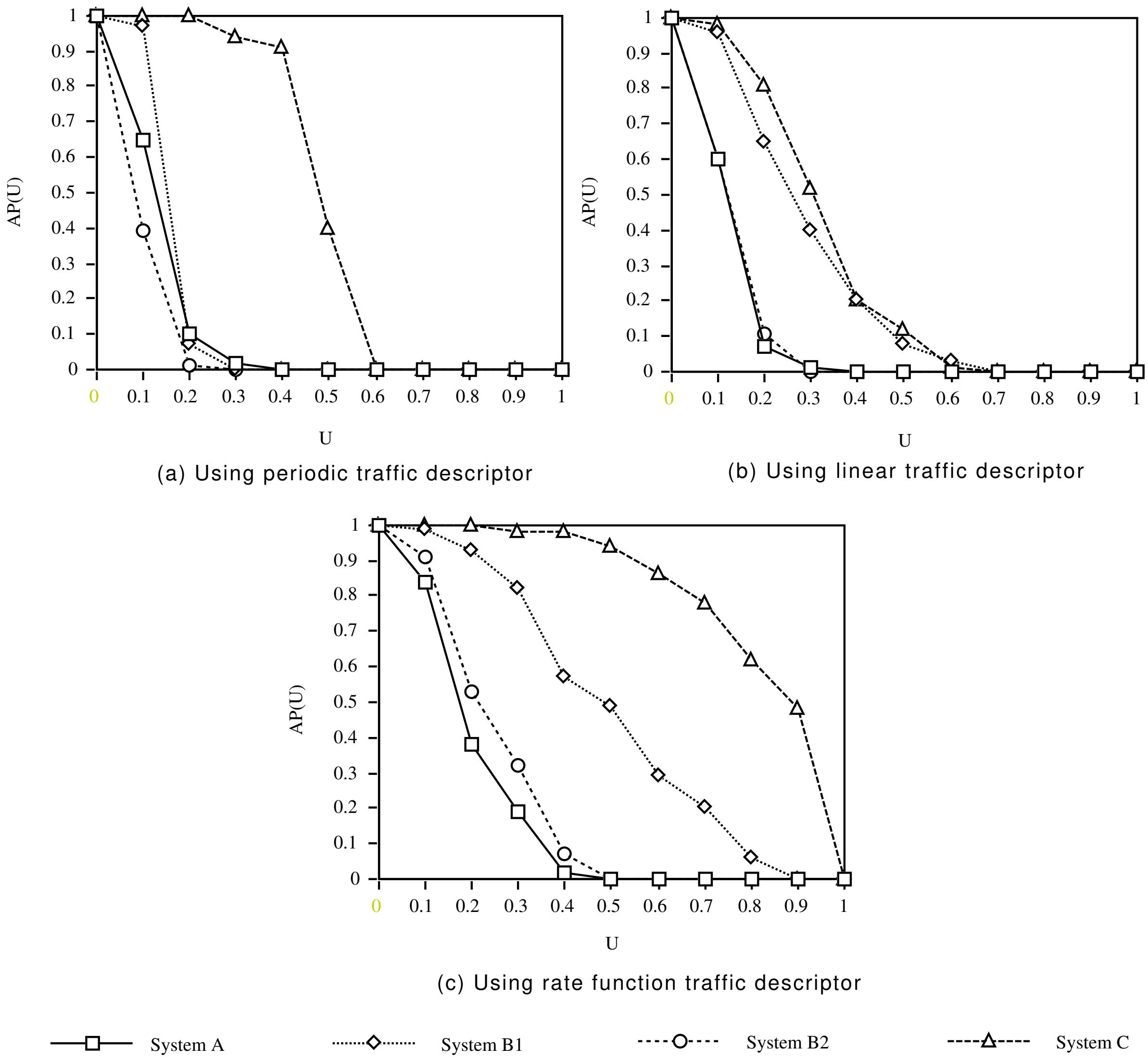
[19] R. J. Vetter. ATM concepts, architectures, and protocols. *Communications of the ACM*, 38(2), Feb. 1995.

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

[21] H. Zhang and S. Keshav. Comparison of rate-based service disciplines. In *Proceedings of ACM SIGCOMM'91*, pages 113–121, Sept. 1991.

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

[23] Q. Zheng and K. G. Shin. On the ability of establishing real-time channels in point-to-point packet-switched networks. To appear in IEEE Transactions on Communications.

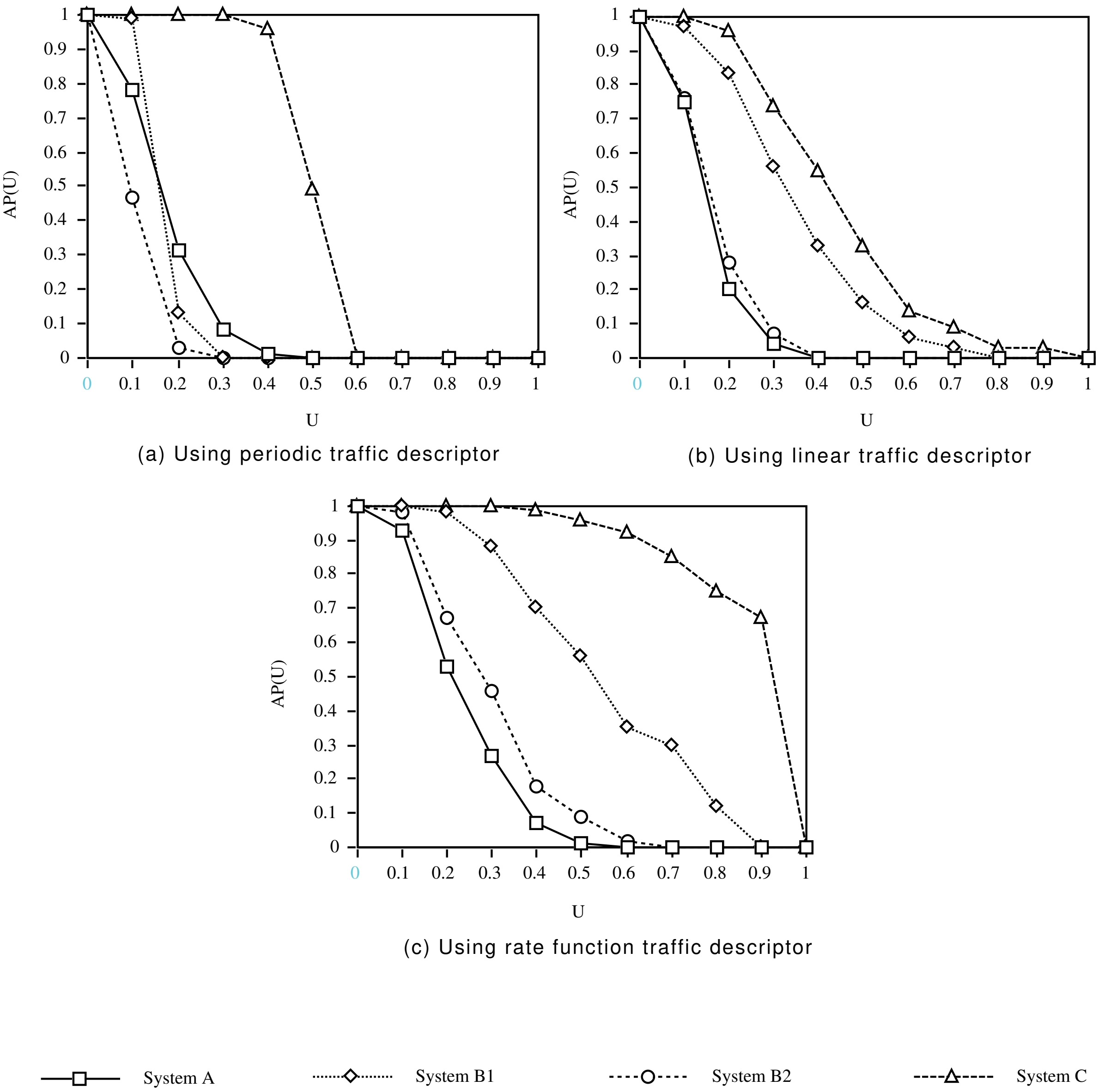


(a) Using periodic traffic descriptor

(b) Using linear traffic descriptor

(c) Using rate function traffic descriptor

Figure 12: Admission probability for periodic traffic



(a) Using periodic traffic descriptor

(b) Using linear traffic descriptor

(c) Using rate function traffic descriptor

Figure 13: Admission probability for periodic traffic

A Analysis of leaky bucket

In this section, we analyze the connection M_i which is regulated by a leaky bucket with parameters (β_i, ρ_i). We derive expressions for the maximum buffer requirement (B_i), the maximum queueing delay (d^{lb}_{max}) and the output parameters of the traffic after the leaky bucket.

Before we derive the expressions for B_i, d^{lb} and the output traffic parameters, we need to discuss some additional notations assumptions and definitions which will be useful in the remainder of this appendix.

A.1 Notations

$A_{Traffic\ descriptor}(t, t+P)$ denotes the number of of cells that have arrived[5] at the input of the leaky bucket, in the interval $[t, t+P]$.

If the traffic at the input of the leaky bucket is characterized by

$(C^{in}_{i,0}, I_i)$, we have

$$A_1(t, t+P) \leq \lfloor \frac{P}{I_i} \rfloor C^{in}_{i,0} + \min(C^{in}_{i,0}, P - \lfloor \frac{P}{I_i} \rfloor I_i). \tag{17}$$

$(\Gamma^{in}_{i,0}(I))$, we have

$$A_2(t, t+P) \leq P\Gamma^{in}_{i,0}(P). \tag{18}$$

$(C^{out}_{i,0}, I_i)$ denotes the traffic description at the output of the leaky bucket if (C, I) is used as the internal traffic descriptor.

$(\Gamma^{in}_{i,0}, I_i)$ denotes the traffic description at the output of the leaky bucket if the maximum rate function is used as the internal traffic descriptor.

A.2 Assumptions

Throughout our analysis we make the following assumptions.

- To ensure the stability of the system,

$$\rho_i \geq \begin{cases} \frac{C^{in}_{i,0}}{I_i} + \frac{1}{I_i} & \text{if } M_i \text{ modeled by } (C^{in}_{i,0}, I_i) \\ \\ \lim_{I\to\infty} \Gamma^{in}_{i,0}(I) + \frac{1}{I_i} & \text{if } M_i \text{ modeled by } (\Gamma^{in}_{i,0}(I)). \end{cases} \tag{19}$$

- The bucket size is at least one, i.e.,

$$\beta_i \geq 1. \tag{20}$$

[5] A cell is said to have arrived at time t, if the last bit of the cell arrives at the leaky bucket input buffer at time t.

A.3 Useful definitions and properties

Some useful concepts to understand the functioning of a leaky bucket regulator, are that of a *regeneration point* and *overflow point*. In this section we shall define *regeneration point* and *overflow point* and study some useful properties.

DEFINITION A.1 *For a connection regulated by a leaky bucket with parameters (β, ρ), a regeneration point, is any time t, when the number of cells queued at the leaky bucket server is zero and the number of tokens in the leaky bucket at time t^+ is β.*

The time $t = 0$ is a trivial example of a regeneration point for any connection M_i.

DEFINITION A.2 *For a connection regulated by a leaky bucket with parameters (β, ρ), an overflow point, is any time t, when a new token arrives at the leaky bucket and there are β tokens in the leaky bucket at time t^-.*

The next two lemmas give two useful properties for connections described by the (C, I) traffic descriptor.

LEMMA A.1 *For the connection M_i, modeled by $(C_{i,0}^{in}, I_i), C_{i,0}^{in} < I_i$ and regulated by a leaky bucket (β_i, ρ_i), if t is a regeneration point then there is an overflow point in the interval $(t, t + I_i]$.*

Proof: We prove this by contradiction.

Let us assume that for the connection M_i, modeled by $(C_{i,0}^{in}, I_i), C_{i,0}^{in} < I_i$ and regulated by a leaky bucket (β_i, ρ_i), if t is a regeneration point then there is no overflow point in the interval $(t, t + I_i]$.

Since, t is a regeneration point, by definition A.1, at time t the number of cells in the queue is zero and at time t^+ the number of tokens in the leaky bucket is β_i.

Therefore, if there is no overflow,the number of tokens available at time t', $t < t' \leq t + I_i$, is at least

$$\beta_i + \lfloor \rho_i \cdot (t' - t) \rfloor. \tag{21}$$

Since, the leaky bucket size is β_i and there is no overflow point in $(t, t + I_i]$, for $t < t' \leq t + I_i$, we have

$$\beta_i + \lfloor \rho_i \cdot (t' - t) \rfloor - A_1(t, t') < \beta_i + 1. \tag{22}$$

By algebraic manipulation of (22), we get for $t < t' \leq t + I_i$,

$$\lfloor \rho_i \cdot (t' - t) \rfloor - 1 < A_1(t, t'). \tag{23}$$

Hence, for $t' = t + I_i$, we have

$$\lfloor \rho_i \cdot I_i \rfloor - 1 < A_1(t, t + I_i). \tag{24}$$

Using (19), substitute the minimum value of ρ_i, i.e., $\frac{C_{i,0}^{in}}{I_i} + \frac{1}{I_i}$ in (24). Then we have

$$C_{i,0}^{in} < A_1(t, t + I_i). \tag{25}$$

But (25) contradicts (17), the definition of the $(C_{i,0}^{in}, I_i)$ descriptor.

Therefore, our assumption that there is no overflow point in the interval $(t, t + I_i]$ leads to a contradiction.

Hence, for the connection M_i, modeled by $(C_{i,0}^{in}, I_i), C_{i,0}^{in} < I_i$ and regulated by a leaky bucket (β_i, ρ_i), if t is a regeneration point then there is exists an overflow point in $(t, t + I_i]$. □

LEMMA A.2 *For the connection M_i, modeled by $(C_{i,0}^{in}, I_i), C_{i,0}^{in} < I_i$ and regulated by a leaky bucket (β_i, ρ_i), the interval between any two consecutive regeneration points is at most I_i.*

Proof: Let t be a regeneration point for the connection M_i, regulated by a leaky bucket (β_i, ρ_i).

By Lemma A.1 there exists an overflow point in the interval $[t, t + I_i]$. Let the first overflow point in the interval $[t, t + I_i]$ be at time t_1, $t \leq t_1 \leq t + I_i$. That is, at t_1 a new token arrives at the leaky bucket and it is discarded since the leaky bucket is already filled with β_i tokens at time t_1^-.

Hence, at time t_1^-, we have

$$\beta_i + \lfloor \rho_i \cdot (t_1^- - t) \rfloor - A_1(t, t_1^-) = \beta_i. \tag{26}$$

That is,

$$\lfloor \rho_i \cdot (t_1^- - t) \rfloor = A_1(t, t_1^-). \tag{27}$$

But (27) implies that the number of cells in the leaky bucket buffer at time t_1^- is zero. Therefore, by definition A.1, t_1^- is a regeneration point. Since t_1^- lies in the interval $[t, t + I_i]$, the lemma is proved. □

A.4 Expression for B_i

The following theorem, gives the maximum buffer requirement at the leaky bucket regulating connection M_i.

THEOREM A.1 *For a connection M_i regulated by a leaky bucket with parameters (β_i, ρ_i), the maximum queue length B_i is given by,*

$$B_i = \begin{cases} \max_{0 \le t \le I_i}(C_{i,0}^{in} - \beta_i - \lfloor \rho_i \cdot t \rfloor) & \text{if } M_i \text{ modeled by } (C_{i,0}^{in}, I_i) \\ \\ \max_{\forall I}(I\Gamma_{i,0}^{in}(I) - \beta_i - \lfloor \rho_i \cdot I \rfloor) & \text{if } M_i \text{ modeled by } \Gamma_{i,0}^{in}(I) \end{cases} \tag{28}$$

Proof: We will only prove (28), for M_i modeled by $(C_{i,0}^{in}, I_i)$. The proof for M_i modeled by $\Gamma_{i,0}^{in}(I)$ is similar.
For M_i modeled by $(C_{i,0}^{in}, I_i)$.

Let t^r be a regeneration point. The maximum queue length at the leaky bucket in the interval $[t^r, t^r + I_i]$ is

$$\max_{t^r \le t \le t^r + I_i}(A_1(t^r, t) - \beta_i - \lfloor \rho_i \cdot (t - t^r) \rfloor). \tag{29}$$

From definition A.1, it follows that the maximum queue length at the leaky bucket will occur between two regeneration points. Further, by Lemma A.2, the maximum interval between two consecutive regeneration points is I_i.

Therefore, B_i the maximum queue length is given by

$$B_i = \max_{\forall t^r}(\max_{t^r \le t \le t^r + I_i}(A_1(t^r, t) - \beta_i - \lfloor \rho_i \cdot (t - t^r) \rfloor)). \tag{30}$$

Also, from (17), we have

$$A_1(t^r, t) \le \lfloor \frac{(t - t^r)}{I_i} \rfloor C_{i,0}^{in} + \min(C_{i,0}^{in}, (t - t^r) - \lfloor \frac{(t - t^r)}{I_i} \rfloor I_i). \tag{31}$$

Substituting, (31) in (30) we get

$$B_i = \max_{\forall t^r}(\max_{t^r \le t \le t^r + I_i}(\lfloor \frac{(t - t^r)}{I_i} \rfloor C_{i,0}^{in} + \min(C_{i,0}^{in}, (t - t^r) - \lfloor \frac{(t - t^r)}{I_i} \rfloor I_i) - \beta_i - \lfloor \rho_i \cdot (t - t^r) \rfloor)). \tag{32}$$

Without, loss of generality we assume $t^r = 0$. Hence, (32) can be rewritten as

$$B_i = \max_{0 \le t \le I_i}(C_{i,0}^{in} - \beta_i - \lfloor \rho_i \cdot t \rfloor). \tag{33}$$

Similarly, we can prove (28) for the descriptor $\Gamma^{in}(I)$. □

A.5 Expression for d^{lb}_{max}

The next lemma gives the bounds on the maximum queueing delay, d^{lb}_{max}, experienced by a cell of connection M_i, at a leaky bucket regulator, (β_i, ρ_i).

LEMMA A.3 *For a connection M_i, if B_i at the leaky bucket regulator occurs at time t_{max} then*

$$d^{lb}_{max} = \frac{(B_i - 1)}{\rho_i} + \frac{\lfloor \rho_i \cdot t_{max} \rfloor}{\rho_i} + \frac{1}{\rho_i} - t_{max}. \tag{34}$$

Proof:

When there are more than one cell queued at the leaky bucket server, the server serves the queued cells of the connection at a constant rate ρ_i. Hence, when the queue size is maximum the last cell in the buffer will experience the maximum delay. Therefore, in order to find the maximum queueing delay, we need to find the waiting time of the last cell in the buffer when the queue length is maximum.

Let t_{max} be the time when the queue length at the leaky bucket reaches the maximum value, i.e., B_i.

Let t_p, $t_p \leq t_{max}$ be the latest time there was a token in the leaky bucket. Then t_p is given by

$$t_p = \frac{\lfloor \frac{t_{max}}{\frac{1}{\rho_i}} \rfloor}{\rho_i} \tag{35}$$

i.e.,

$$t_p = \frac{\lfloor \rho_i \cdot t_{max} \rfloor}{\rho_i}. \tag{36}$$

Further, let t_n be the earliest time, after t_{max}, that there is a token available in the leaky bucket. Hence, the transmission of the first cell of the B_i cells in the leaky bucket queue will begin at t_n. t_n is given by

$$t_n = t_p + \frac{1}{\rho_i}. \tag{37}$$

Substituting, (36) in (37) we get

$$t_n = \frac{\lfloor \rho_i \cdot t_{max} \rfloor}{\rho_i} + \frac{1}{\rho_i}. \tag{38}$$

The queue length at the leaky bucket increases when there are no tokens available in the leaky bucket. Therefore, we have

$$t_{max} < t_n. \tag{39}$$

Let $T_1 = t_n - t_{max}$. Using (38), we get

$$T_1 = \frac{\lfloor \rho_i \cdot t_{max} \rfloor}{\rho_i} + \frac{1}{\rho_i} - t_{max}. \tag{40}$$

After t_n, a cell in the leaky bucket queue will be served periodically, with period being $\frac{1}{\rho_i}$. Hence, the maximum waiting time is given by

$$\frac{(B_i - 1)}{\rho_i} + T_1. \tag{41}$$

Using (40), (41) becomes

$$\frac{(B_i - 1)}{\rho_i} + \frac{\lfloor \rho_i \cdot t_{max} \rfloor}{\rho_i} + \frac{1}{\rho_i} - t_{max}. \tag{42}$$

That is,

$$d^{lb}_{max} = \frac{(B_i - 1)}{\rho_i} + \frac{\lfloor \rho_i \cdot t_{max} \rfloor}{\rho_i} + \frac{1}{\rho_i} - t_{max}. \tag{43}$$

□

The following theorem gives the bounds on the worst case delay experienced by the cell at the leaky bucket.

THEOREM A.2 *For a connection M_i, regulated by a leaky bucket parameterized by (β_i, ρ_i)*

$$\frac{(B_i - 1)}{\rho_i} < d^{lb}_{max} \leq \frac{B_i}{\rho_i}. \tag{44}$$

The proof follows by algebraic manipulations of (34).

A.6 Traffic description at the output of leaky bucket

Next, we derive traffic parameters of the connection M_i, at the output of the leaky bucket.

THEOREM A.3 *For a connection M_i, regulated by a leaky bucket, (β_i, ρ_i), if the input traffic descriptor is*

$(C^{in}_{i,0}, I_i)$ *then the output traffic parameters are* $(C^{out}_{i,0}, I_i)$*, where*

$$C^{out}_{i,0} = \min(C^{in}_{i,0}, \beta_i + \lfloor \rho_i \cdot I_i \rfloor). \tag{45}$$

$(\Gamma^{in}_{i,0}(I))$ *then the output traffic parameter* $(\Gamma^{out}_{i,0}(I))$ *is given by*

$$\Gamma^{out}_{i,0}(I) = \frac{\min(I\Gamma^{in}_{i,0}(I), \beta_i + \lfloor \rho_i \cdot I \rfloor)}{I}. \tag{46}$$

The proof of this theorem is obvious. □

B Computation of β_i^{max}

In this section for the connection M_i we derive an expression for β_i^{max}, the minimum value of β_i for which the maximum queueing delay of a cell at the leaky bucket queue is zero.

THEOREM B.1 *For a connection M_i regulated by a leaky bucket with parameters (β_i, ρ_i), β_i^{max} is given by*

$$\beta_i^{max} = \begin{cases} \max_{0 \le t \le I_i}(C_{i,0}^{in} - \lfloor \rho_i \cdot t \rfloor) \ \le \ C_{i,0}^{in} & \textit{if } M_i \textit{ modeled by } (C_{i,0}^{in}, I_i) \\ \\ \max_{\forall I}(I\Gamma_{i,0}^{in}(I) - \lfloor \rho_i \cdot I \rfloor) & \textit{if } M_i \textit{ modeled by } \Gamma_{i,0}^{in}(I) \end{cases} \tag{47}$$

Proof: We will only prove (47), for M_i modeled by $(C_{i,0}^{in}, I_i)$. The proof for M_i modeled by $\Gamma_{i,0}^{in}(I)$ is similar.

By (42) of Lemma A.3, if B_i is attained at t_{max} then the maximum queueing delay for a cell in the queue of a leaky bucket is

$$\frac{(B_i - 1)}{\rho_i} + \frac{\lfloor \rho_i \cdot t_{max} \rfloor}{\rho_i} + \frac{1}{\rho_i} - t_{max}. \tag{48}$$

Also, from Theorem A.1 we have

$$B_i = \max_{0 \le t \le I_i} (C_{i,0}^{in} - \beta_i - \lfloor \rho_i \cdot t \rfloor). \tag{49}$$

Now, if B_i is attained at t_{max}, then (49) becomes

$$B_i = C_{i,0}^{in} - \beta_i - \lfloor \rho_i \cdot t_{max} \rfloor. \tag{50}$$

Now by substituting (50) in (48), the maximum waiting time for a cell in the queue of a leaky bucket is

$$\frac{(C_{i,0}^{in} - \beta_i - \lfloor \rho_i \cdot t_{max} \rfloor - 1)}{\rho_i} + \frac{\lfloor \rho_i \cdot t_{max} \rfloor}{\rho_i} + \frac{1}{\rho_i} - t_{max}. \tag{51}$$

Solving, for the value of β_i which makes (51) zero, we have

$$\beta_i = C_{i,0}^{in} - \rho_i \cdot t_{max}. \tag{52}$$

Since, β_i^{max} is the smallest integer value for which the maximum waiting time of a cell at the leaky bucket queue is zero. By algebraic manipulation of (52) we have

$$\beta_i^{max} = C_{i,0}^{in} - \lfloor \rho_i \cdot t_{max} \rfloor. \tag{53}$$

(53) can be rewritten as

$$\beta_i^{max} = \max_{0 \le t \le I_i} (C_{i,0}^{in} - \lfloor \rho_i \cdot t \rfloor). \quad (54)$$

Similarly, we can prove (47) for the traffic described by $\Gamma_{i,0}^{in}(I)$. $\square$

C Procedure to compute end-to-end delays

In this section we present an algorithm to compute the upper bound of the end-to-end cell delays for every hard real-time connection.

In an ATM network supporting hard real-time application, a unidirectional connection is set up between every source destination pair. We model the network as a set of servers that serve individual connections [2, 15]. Every such connection traverses a sequence of servers. Those servers that offer only constant delays to a connection's cells and do not change the cell traffic behavior of a connection are considered separately from those which offer variable delays to cells and hence affect a connection's traffic. As a consequence of the above step, each connection is represented as a path in a Connection-server graph whose nodes are servers which potentially affect the connection's traffic. The cells from different connections contend with each other for the network resources (servers). In order to find the upper bound of the end-to-end cell delays we must first find the upper bound on the cell delays at every server in the network which services the hard real-time connections. Once the delays at every server is known then the end-to-end delay is easily obtained by summing the delays along the path of the connection. We present a procedure that computes the cell delay at every server in the network.

Let V be the set of servers in the Connection-Server graph. Let $V^\star$, $V^\star \subseteq V$, be a set of servers, such that for each server, $v^\star$, $v^\star \in V^\star$, the input traffic of all the connections is completely known. In this paper we assume that the paths of the hard real-time connections do not form any loops. Due to the absence of loops, there always exist a non-empty set $V^\star$.

```
procedure ComputeEndToEndDelays;
begin
    V* ⇐ Find_V*(V) ;
    while ( V ≠ φ) do begin
        V ⇐ V − V* ;
        foreach v* in V* do begin
            compute_delay(v*);
            compute_output(v*);
        endforeach;
        V* ⇐ Find_V*(V) ;
    endwhile;
end;
```

Figure 14: Computing the end-to-end delays

Figure 14 shows the procedure to compute the delays at every server in V. The procedure shown in Figure 14 is a simple iterative procedure. It first starts with the procedure call **Find_**$V^\star(V)$ which evaluates the set $V^\star$, $V^\star \subseteq V$. For each server $v^\star$ in $V^\star \subseteq V$, the algorithm then invokes the procedure compute_delay($v^\star$) to compute the upper bound of the cell delays at server $v^\star$. After the delays are computed the algorithm calls procedure compute_output($v^\star$). The procedure compute_output($v^\star$) computes the output traffic parameters of all the connections using $v^\star$. Since, the Connection-server graph has no loops, once the traffic parameters at the output of all servers in $V^\star$ are computed, a new set of vertices become active, i.e., have their input traffic fully described. The procedure is iteratively repeated until all the servers become eventually become active. Thus, the set V is eventually become empty and the algorithm terminates.

D Uniqueness of $\vec{\beta}^*$

In this section we prove the uniqueness of β^*. To prove the uniqueness of β^* we must first study the impact of varying the burst parameter of a single connection on the worst case cell delays in the network. Hence, we first present some results associated with varying the burst parameters.

We prove the results when the traffic is described using the $\Gamma(I)$ traffic descriptor. The results for the other traffic descriptors are same and can be proved using similar proof techniques.

D.1 Notations

Before we proceed with the proof, we need some notations and lemmas. The relevant notations are introduced below.

$\Gamma_{i,j}^{in}(I)$ is the maximum arrival rate of cells belonging to connection M_i at the input of the j^{th} server of the connection measured over an interval of length I. By convention, we treat the leaky bucket at the UNI (if present) to be the 0^{th} server. If there are n switches constituting the route for M_i, then $j = 1, 2, \ldots, n$ denote the FCFS servers at the output links of switches $1, 2, \ldots, n$ respectively.

$\Gamma_{i,j}^{out}(I)$ is the maximum departure rate of connection M_i at the output of M_i's j^{th} server.

Note, that if in the path of connection M_i j is a preceding server of k then $\forall I$ we have $\Gamma_{i,j}^{out}(I) = \Gamma_{i,k}^{in}(I)$.

D.2 Impact of varying β on the cell delays and traffic at the leaky bucket

The following lemma discusses the impact of varying the burst parameter vector of leaky buckets at the UNI on the cell delays across the leaky buckets and also on the traffic rates at the output of a leaky bucket. The lemma asserts that the worst case cell delay of a connection through the leaky bucket will decrease if the burst parameter assigned for the leaky bucket regulating that connection is increased. The output traffic rate for the connection however must increase.

LEMMA D.1 *Consider $\mathcal{M}$, a set of connections whose traffic at the UNI is regulated by leaky buckets with rate parameters given by $\vec{\rho}$ and burst parameters given by $\vec{\beta}$. Let $d_{i,0}^{lb}$ be the delay and $\Gamma_{i,0}^{out}(I)$ be the output traffic rate of connection $M_i, M_i \in \mathcal{M}$, at the leaky bucket server. Assume that $\vec{\beta}$ is changed to $\vec{\beta}'$ such that*

$$\beta_i' > \beta_i \tag{55}$$

and

$$\beta_k' = \beta_k \qquad \textit{for } k \neq i \ . \tag{56}$$

Let $d_{i,0}^{lb'}$ be the new delay and $\Gamma_{i,0}^{out'}(I)$ be the new traffic output rate of connection M_i when $\vec{\beta}$ is changed to $\vec{\beta}'$. Then the following relations hold.

1.
$$d_{i,0}^{lb'} \leq d_{i,0}^{lb}. \tag{57}$$

2.
$$\Gamma_{i,0}^{out'}(I) \geq \Gamma_{i,0}^{out}(I). \tag{58}$$

Proof:

Proof of (57) .

To prove (57), we need to show that

$$d_{i,0}^{lb'} - d_{i,0}^{lb} \leq 0. \tag{59}$$

First, since $\beta_i' > \beta_i$, without loss of generality β_i' can be written as

$$\beta_i' = \beta_i + l, \tag{60}$$

where l is a positive integer.

In Appendix A, it was shown that the maximum queueing delay for connection M_i at a leaky bucket having β_i as the burst parameter and ρ_i as the rate parameter is bounded as follows:

$$\frac{1}{\rho_i} \cdot (\max_{\forall t}(t\Gamma_{i,0}^{in}(t) - \beta_i - \lfloor \rho_i \cdot t \rfloor) - 1) \leq d_{i,0}^{lb} \leq \frac{1}{\rho_i} \cdot (\max_{\forall t}(t\Gamma_{i,0}^{in}(t) - \beta_i - \lfloor \rho_i \cdot t \rfloor)). \tag{61}$$

Consequently, the difference $d_{i,0}^{lb'} - d_{i,0}^{lb}$ is bounded as

$$\begin{aligned} d_{i,0}^{lb'} - d_{i,0}^{lb} &\leq \frac{1}{\rho_i} \cdot (\max_{\forall t}(t\Gamma_{i,0}^{in}(t) - \beta_i' - \lfloor \rho_i \cdot t \rfloor)) - \\ &\quad \frac{1}{\rho_i} \cdot (\max_{\forall t}(t\Gamma_{i,0}^{in}(t) - \beta_i - \lfloor \rho_i \cdot t \rfloor) - 1) \\ &= \frac{1}{\rho_i} \cdot (\beta_i - \beta_i') + \frac{1}{\rho_i} \\ &= \frac{-l}{\rho_i} + \frac{1}{\rho_i} \\ &= \frac{-l}{\rho_i}(l-1). \end{aligned} \tag{62}$$

Since $l \geq 1$, we have

$$d_{i,0}^{lb'} - d_{i,0}^{lb} \leq 0. \tag{63}$$

Proof of (58) .

In Appendix A, it is shown that $\Gamma_{i,0}^{out}(I)$ for leaky bucket with parameters $\vec{\beta}$ and $\vec{\rho}$ is given by

$$\Gamma_{i,0}^{out}(I) = \frac{\min(I\Gamma^{in}i, 0, \beta_i + \lfloor \rho_i \cdot I \rfloor)}{I}. \tag{64}$$

(58) is obvious by observing that $\Gamma_{i,0}^{out}(I)$ for a leaky bucket is a nondecreasing function of β_i.

□

The previous lemma showed that the worst case cell transfer delay across a leaky bucket is a nondecreasing function of β_i. Intuitively, increasing the value of β_i means decreasing the degree of traffic regulation for stream M_i. This is obvious from the fact that the output traffic at the leaky bucket is *bounded* by $\beta_i + \rho_i t$ and with ρ_i set to long term input traffic rate, a large value of β_i implies that the output traffic is virtually same as the input traffic.

D.3 Impact of varying β on the cell delays and traffic at the FCFS servers in the network

Before we discuss the impact of the burst parameter vector $\vec{\beta}$ on the delay and output traffic rates of an FCFS server we present two important properties of an FCFS server.

LEMMA D.2 *For a FCFS server A serving a set of connections $S(A)$ with input traffic described by $\Gamma_{i,A}^{in}(I)$, $M_i \in S(A)$ if the input traffic is changed to $\Gamma_{i,A}^{in'}(I)$ such that $\forall I$, $M_i \in S(A)$,*

$$\Gamma_{i,A}^{in'}(I) \geq \Gamma_{i,A}^{in}(I) \tag{65}$$

then the following relations hold. For $M_i \in S(A)$,

1.
$$d_{i,A}^{fcfs'} \geq d_{i,A}^{fcfs}. \tag{66}$$
where $d_{i,A}^{fcfs'}$, and $d_{i,A}^{fcfs}$ are the worst case delays at server A when the input traffic described by $\Gamma_{i,A}^{in'}(I)$ and $\Gamma_{i,A}^{in}(I)$ respectively.

2.
$$\Gamma_{i,A}^{out'}(I) \geq \Gamma_{i,A}^{out}(I), \tag{67}$$
where $\Gamma_{i,A}^{out'}(I)$, and $\Gamma_{i,A}^{out}(I)$ are the traffic descriptions at the output of the server A when the input traffic described by $\Gamma_{i,A}^{in'}(I)$ and $\Gamma_{i,A}^{in}(I)$ respectively.

The proof of this lemma is given in [14].

LEMMA D.3 *For a FCFS server A serving a set of connection $S(A)$ let $d_{i,A}^{fcfs}$ be the delay and $\Gamma_{i,A}^{out}(I)$ be the output traffic rate of connection M_i at A. Assume that $\forall M_i,\ M_i \in S(A)$, $d_{i,A}^{fcfs}$ is changed to $d_{i,A}^{fcfs'}$ such that*

$$d_{i,A}^{fcfs'} > d_{i,A}^{fcfs}. \tag{68}$$

Then the following relation holds. $\forall M_i,\ M_i \in S(A)$,

$$\Gamma_{i,j}^{out'}(I) \geq \Gamma_{i,j}^{out}(I). \tag{69}$$

The proof of this lemma is given in [14].

The following lemma discusses the impact of the burst parameter vector $\vec{\beta}$ on the delay and output traffic rates of an FCFS server.

LEMMA D.4 *Let A be the j^{th} FCFS server along the route constituting connection M_i. Let $d_{i,j}^{fcfs}$ be the delay and $\Gamma_{i,j}^{out}(I)$ be the output traffic rate of connection M_i at A when the traffic of connections in $\mathcal{M}$ is regulated at the UNI with leaky bucket regulators defined by $\vec{\beta}$ and $\vec{\rho}$. Assume that $\vec{\beta}$ is changed to $\vec{\beta}'$ such that*

$$\beta_i' > \beta_i \tag{70}$$

and

$$\beta_k' = \beta_k \qquad for\ k \neq i\ . \tag{71}$$

Let $d_{i,j}^{fcfs'}$ be the new delay and $\Gamma_{i,j}^{out'}(I)$ be the new output traffic rate of connection M_i when $\vec{\beta}$ is changed to $\vec{\beta}'$. Then the following relations hold.

1.
$$d_{i,j}^{fcfs'} \geq d_{i,j}^{fcfs}. \tag{72}$$

2.
$$\Gamma_{i,j}^{out'}(I) \geq \Gamma_{i,j}^{out}(I). \tag{73}$$

Proof:

Proof of (72) .

Let $\mathcal{S}(A)$ be the subset of $\mathcal{M}$ that contains the connections that are served by A. That is,

$$\mathcal{S}(A) = \{M_i \mid M_i \text{ is served by } A\}. \tag{74}$$

Let $H_A(h)$ be the hop number of connection M_h at server A if the route for M_h includes A as one of its servers, i.e., when $M_h \in \mathcal{S}(A)$. $H_A(h)$ is undefined if $M_h \in \mathcal{S}(A)$. Further, $H_A(i) = j$

since, A is the j^{th} server for connection M_h. In [2, 14], it is shown that the maximum delay of a cell from M_i at A, the j^{th} server in M_i's connection path, is given by

$$d_{i,j}^{fcfs} = \max_{\forall t}(\sum_{m:M_m \in \mathcal{S}(A)} t \cdot \Gamma_{m,H_A(m)}^{in}(t) - t). \tag{75}$$

Noting that $\Gamma_{m,H_A(m)}^{in}(I) = \Gamma_{m,H_A(m)-1}^{out}(I)$ and rewriting (75), we get

$$\begin{aligned} d_{i,j}^{fcfs} &= \max_{\forall t}(\sum_{m:M_m \in \mathcal{S}(A)} t \cdot \Gamma_{m,H_A(m)-1}^{out}(t) - t) \\ &= \max_{\forall t}(\sum_{m:M_m \in \mathcal{S}(A), m \neq i} t \cdot \Gamma_{m,H_A(m)-1}^{out}(t) + t \cdot \Gamma_{i,j-1}^{out}(t) - t) \\ &= \max_{\forall t}(t \cdot \sum_{m:M_m \in \mathcal{S}(A), m \neq i} \Gamma_{m,H_A(m)-1}^{out}(t) + \Gamma_{i,j-1}^{out}(t) - 1) \end{aligned} \tag{76}$$

Similarly, if the burst parameter vector is changed to $\vec{\beta}'$, we obtain $d_{i,j}^{fcfs'}$ as

$$d_{i,j}^{fcfs'} = \max_{\forall t}(t \cdot \sum_{m:M_m \in \mathcal{S}(A), m \neq i} \Gamma_{m,H_A(m)-1}^{out'}(t) + \Gamma_{i,j-1}^{out'}(t) - 1). \tag{77}$$

We shall prove (72) by induction.

Base Case Consider the case when $j = 1$; that is, the input to the FCFS server is the output of the leaky bucket regulator. From Lemma D.1 we have for $0 \leq t < \infty$,

$$\Gamma_{i,0}^{out'}(t) \geq \Gamma_{i,0}^{out}(t) \tag{78}$$

and for $k \neq i$,

$$\Gamma_{k,0}^{out'}(t) = \Gamma_{k,0}^{out}(t). \tag{79}$$

Comparing (76) and (77) we have

$$d_{i,1}^{fcfs'} \geq d_{i,1}^{fcfs}. \tag{80}$$

Induction Hypothesis Let (72) hold for the case $j = k$, $k > 1$,. Therefore,

$$d_{i,k}^{fcfs'} \geq d_{i,k}^{fcfs}. \tag{81}$$

Induction Step Consider the case $j = k + 1$. Using (81) and the result of Lemma D.3 we get $\forall I$,

$$\Gamma_{i,k}^{out'}(I) \geq \Gamma_{i,k}^{out}(I). \tag{82}$$

But, $\forall I$,

$$\Gamma_{i,k}^{out'}(I) = \Gamma_{i,k+1}^{in'}(I) \tag{83}$$

and

$$\Gamma_{i,k}^{out}(I) = \Gamma_{i,k+1}^{in}(I). \tag{84}$$

Therefore, using (82), (83) and (84) we have

$$\Gamma_{i,k+1}^{in'}(I) \geq \Gamma_{i,k+1}^{in}(I). \tag{85}$$

Hence, applying the result from Lemma D.2 we get

$$d_{i,k+1}^{fcfs'} \geq d_{i,k+1}^{fcfs}. \tag{86}$$

From (80), (81) and (86) and by applying the principle of mathematical induction (72) follows..

Proof of (73) .

The proof for this result is similar to the proof of (72).

□

D.4 Impact of varying β on the network delay d^{net}

The following Theorem discusses the impact of varying the burst parameter vector of leaky buckets at the UNI on the end-to-end cell delays across all the connections.

THEOREM D.1 *Consider a connection set $\mathcal{M}$. Let β_i, the burst parameter associated with the leaky bucket regulating M_i's traffic at the UNI be considered variable while all the β_j's, $(j \neq i)$ are held constant. Consider all the worst case delays as functions of β_i. Then the following are true:*

1. *$d_i^{lb}(\beta_i)$ does not increase if $1 \leq \beta_i \leq \beta_i^{max}$.*
   $d_i^{lb}(\beta_i) = 0$ if $\beta_i \geq \beta_i^{max}$.
2. *$d_i^{net}(\beta_i)$ does not decrease if $1 \leq \beta_i \leq \beta_i^{max}$.*
   $d_i^{net}(\beta_i)$ is a constant if $\beta_i \geq \beta_i^{max}$.
3. *For $j \neq i$, $d_j^{lb}(\beta_i)$ is a constant.*
4. *For $j \neq i$, $d_j^{net}(\beta_i)$ does not decrease if $1 \leq \beta_i \leq \beta_i^{max}$.*
   $d_j^{net}(\beta_i)$ is a constant if $\beta_i \geq \beta_i^{max}$.

Proof:

Proof of assertion 1.

Follows directly from Lemma D.1 and the definition of β_i^{max}.

Proof of assertion 2.

Follows from Lemma D.4.

Proof of assertion 3.
Obvious, since changing leaky bucket parameter for M_i at the UNI does not affect the traffic from other connections.

Proof of assertion 4.
Since the connection paths do not form a loop, the varying β_i will only impact the connections sharing at least FCFS server with connection M_i.

Because of assertion 2 and Lemma D.4 we have the following. For any server A in the path of connection M_i

$$d_{i,A}^{fcfs} \text{ does not decrease if } 1 \leq \beta_i \leq \beta_i^{max} \tag{87}$$

However, for the FCFS scheduling policy all the connections sharing the same server will have the same worst case cell delay. Therefore, for server A and connection M_j traversing A we have

$$d_{j,A}^{fcfs} \text{ does not decrease if } 1 \leq \beta_i \leq \beta_i^{max} \tag{88}$$

However,

$$d_j^{net} = \sum_{\forall k \text{in } M_j\text{'s path}} d_{j,k}^{fcfs}. \tag{89}$$

Therefore, for $j \neq i$, $d_j^{net}(\beta_i)$ does not decrease if $1 \leq \beta_i \leq \beta_i^{max}$.

Similarly using assertion 1 we can show that for $j \neq i$, $d_j^{net}(\beta_i)$ is a constant if $\beta_i \geq \beta_i^{max}$. □

D.5 Proof of uniqueness of $\vec{\beta}^*$

In this section we shall prove that for a set of connections if $\vec{\beta}^*$ exists then it is unique.

THEOREM D.2 *For a set of connections if the vector $\vec{\beta}^*$ if it exists then is unique.*

Proof: Suppose there is more than one vector that can be $\vec{\beta}^*$. Let $\vec{\beta} =< \beta_1, \ldots, \beta_N >$ and $\vec{\beta}' =< \beta'_1, \ldots, \beta'_N >$ be two distinct vectors that are $\vec{\beta}^*$. Thus,

$$m(\vec{\beta}) = \sum_{i=1}^{N} \beta_i = \sum_{i=1}^{N} \beta'_i = m(\vec{\beta}'). \tag{90}$$

We can construct a new vector $\vec{\beta}'' =< \beta''_1, \ldots, \beta''_N >$ from $\vec{\beta}$ and $\vec{\beta}'$ such that $\forall j$

$$\beta''_j = \min(\beta_j, \beta'_j). \tag{91}$$

Therefore, we have

$$m(\vec{\beta}'') = \sum_{i=1}^{N} \beta''_i \leq m(\vec{\beta}') = m(\vec{\beta}). \tag{92}$$

Let $d_{i,0}^{lb}$, $d_{i,0}^{lb'}$ and $d_{i,0}^{lb''}$ be the upper bound on the delay and $\Gamma_{i,0}^{out}(I)$, $\Gamma_{i,0}^{out'}(I)$, and $\Gamma_{i,0}^{out''}(I)$ be the output traffic rate of connection $M_i, M_i \in \mathcal{M}$, at the leaky bucket server when the vectors $\vec{\beta}$, $\vec{\beta}'$ and $\vec{\beta}''$ are used respectively.

Because of (91) and Lemma D.1 we have for $1 \leq i \leq N$

$$d_i^{lb}(\beta_i^{''}) = \max(d_i^{lb}(\beta_i), d_i^{lb}(\beta_i^{'})), \tag{93}$$

and $\forall I$,

$$\Gamma_{i,0}^{out''}(I) = \min(\Gamma_{i,0}^{out}(I), \Gamma_{i,0}^{out'}(I). \tag{94}$$

Let d_i^{net}, $d_i^{net'}$ and $d_i^{net''}$ be the upper bound on the cell delay experienced by M_i's cell in the network when the vectors $\vec{\beta}$, $\vec{\beta}'$ and $\vec{\beta}''$ are used respectively.

Because of (94) and by applying Lemma D.4 and assertion 2 of Theorem D.1 we get for $1 \leq i \leq N$

$$d_i^{net''} \leq \min(d_i^{net}, d_i^{net'}). \tag{95}$$

Further, because allocations of $\vec{\beta}$ and $\vec{\beta}'$ are distinct without loss of generalization we can assume that there exists x, y, $1 \leq x, y \leq N$ such that $\beta_x \leq \beta_x'$ and $\beta_y > \beta_y'$.

Therefore, using (91) we get

$$\beta_x^{''} = \beta_x \tag{96}$$

and

$$\beta_y^{''} = \beta_y^{'}. \tag{97}$$

From (93), (96) and (97) we get

$$d_x^{lb}(\beta_x^{''}) = d_x^{lb}(\beta_x) \tag{98}$$

and

$$d_y^{lb}(\beta_y^{''}) = d_y^{lb}(\beta_y^{'}). \tag{99}$$

From (95) (96) and (97) we get

$$d_x^{net''} \leq \min(d_x^{net}, d_x^{net'}). \tag{100}$$

and

$$d_y^{net''} \leq \min(d_y^{net}, d_y^{net'}). \tag{101}$$

From (98), (99), (100) and (101) we get

$$d_x^{lb}(\beta_x^{''}) + d_x^{net''} \leq d_y^{lb}(\beta_x) + d_x^{net} \tag{102}$$

and

$$d_y^{lb}(\beta_y^{''}) + d_y^{net''} \leq d_y^{lb'}(\beta_y) + d_y^{net'}. \tag{103}$$

But since $\vec{\beta}$ and $\vec{\beta}'$ are feasible assignment vectors from (102) and (103) we get

$$d_x^{lb}(\beta_x^{''}) + d_x^{net''} \leq d_x^{lb}(\beta_x) + d_x^{net} \leq D_x \tag{104}$$

and

$$d_y^{lb}(\beta_y^{''}) + d_y^{net''} \leq d_y^{lb'}(\beta_y) + d_y^{net'} \leq D_y. \tag{105}$$

Hence, $\vec{\beta}^{''}$ is also a feasible assignment for which the connection set is admissible.

But because of (92) and the definition of $\vec{\beta}^*$, $\vec{\beta}$ and $\vec{\beta}'$ cannot be $\vec{\beta}^*$. Hence are assumption that there are two vectors which are $\vec{\beta}^*$ is wrong. Hence, the theorem. □

E Proof of Theorem 4.2

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* $\vec{\beta}^*$,
   - $\vec{\beta} \longrightarrow \vec{\beta}'$,
   - *and* $s_{\triangle(\vec{\beta},\vec{\beta}')} = 0$.

Proof:

1 *1.* is true since $< 1, 1, \ldots, 1 >$ is a trivial ancestor of all $\vec{\beta}$'s.

2 Let $\vec{\beta}$, $\vec{\beta} \neq \vec{\beta}^*$ be an ancestor of $\vec{\beta}^*$. Therefore, for all x, $1 \leq x \leq N$ we have

$$\beta_x \leq \beta_x^*. \tag{106}$$

Since, $\vec{\beta}^*$ is unique there exists y, $1 \leq y \leq N$ such that $s_y = 0$ and

$$\beta_y < \beta_y^*. \tag{107}$$

Now construct $\vec{\beta}'$ such that for all z, $z \neq y$ $\beta'_z = \beta_z$ and $\beta'_y = \beta_y$. Then using definition of "$\longrightarrow$" we have

$$\vec{\beta} \longrightarrow \vec{\beta}'. \tag{108}$$

Also, by (106) and (107) for all x, $1 \leq x \leq N$ we have

$$\beta'_x \leq \beta_x^*. \tag{109}$$

Hence, there exists a directed path in G such that

$$\vec{\beta}' \stackrel{*}{\longrightarrow} \vec{\beta}^*. \tag{110}$$

Therefore, $\vec{\beta}'$ is an ancestor of $\vec{\beta}^*$.

□