Using Traffic Regulation to Meet End-to-End Deadlines in ATM Networks

Amitava Raha, Sanjay Kamat, Xiaohua Jia, *Member*, *IEEE Computer Society*, and Wei Zhao, *Member*, *IEEE*

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. There are several approaches to managing the network resources in order to meet the delay requirements of connections. This paper focuses on the use of traffic regulation to achieve this objective. Leaky buckets provide simple and user-programmable means of traffic regulation. An efficient optimal algorithm for selecting the burst parameters of leaky buckets to meet connections' deadlines is designed and analyzed. The algorithm is optimal in the sense that it always selects a set of burst parameters whose mean value is minimal and by which the delay requirements of hard real-time connections can be met. The exponential size of the search space makes this problem a challenging one. The algorithm is efficient through systematically pruning the search space. There is an observed dramatic improvement in the system performance in terms of the connection admission probability when traffic is regulated using this algorithm.

Index Terms—ATM network, hard real-time communication, network delay analysis, traffic regulation.

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 are executed at different nodes and communicate among themselves by exchanging messages. For successful operation of the system, the messages exchanged by time-critical tasks have to be delivered by certain deadlines. Examples of such systems include supervisory command and control systems used in manufacturing, chemical processing, nuclear plants, telemedicine, and warships. This paper addresses the issue of guaranteeing end-to-end deadlines of time-critical messages in ATM networks that support distributed HRT systems.

ATM is a connection-oriented technology in which messages are packetized into fixed-size cells. Therefore, guaranteeing message deadlines is tantamount to ensuring that the *worst case end-to-end delay* of a cell does not exceed its deadline. To provide such guarantees, 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 selects appropriate routes for connections such that the delays are within bounds. The scope of this approach is limited because typical ATM networks for HRT applications are LANs. The second approach focuses on scheduling at the ATM switches' output links where traffic from different connections is multiplexed. Similar to CPU scheduling, classical real-time scheduling policies, such as the First Come First Serve (FCFS), Earliest Deadline First (EDF), Generalized Processor Sharing (GPS), and Fair Queuing (FQ), are employed [2], [12], [13], [16], [23], [24], [25]. However, most commercially available switches use a high priority queue for HRT connections and this queue is served in a FCFS manner.

This paper focuses on the third approach: controlling the network delays by regulating the input traffic of each connection. Regulating the input traffic can smooth the burstiness of the traffic, which tends to reduce the adverse impact of burstiness on the end-to-end delays of other connections. Most of the existing ATM networks provide for traffic regulation at the UNI. It is relatively easy to tune the regulation parameters as desired. This is justification for focusing on using the traffic regulator as an access control mechanism for HRT ATM networks. It must be noted that all three approaches are important and that they are complementary. The results of using traffic regulation complement the previous work on route selection and output link scheduling.

The idea behind traffic regulation is to regulate a connection's traffic so that it has a lower impact on the delays of cells of 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 of the cells of others. Regulating a connection's traffic makes the traffic less bursty, thereby reducing the delays of other connections' cells. However, regulating a connection's traffic may delay some of its own cells. Therefore, it is important to choose an appropriate degree of traffic

- *A. Raha is with Fijitsu Software Corp., USA. E-mail: amitava@fsc.fujitsu.com.*
- *S. Kamat is with Bell Laboratories, 101 Crawfords Corner Road, Room 4C-510, Holmdel, NJ 07733. E-mail: sanjayk@dnrc.bell-labs.com.*
- *X. Jia is with the Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong. E-mail: Jia@cs.cityu.edu.hk.*
- *W. Zhao is with the Department of Computer Science, Texas A&M University, College Station, Texas. E-mail: Zhao@cs.tamu.edu.*

Manuscript received 2 Sept. 1997; revised 28 May 1998.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 105572.

regulation so that all connections can meet their end-to-end deadlines.

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 degree of regulation of a connection depends on the leaky bucket parameters assigned to it. The degree 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 assigned carefully to ensure that every connection meets its end-to-end deadline. Berger proposed an analysis model of leaky bucket regulators in [1]. In his work, Berger examined a single leaky bucket, analysing the job blocking probabilities (or system throughput) versus parameters such as job arrival patterns, capacity of token bank, and size of job buffer. In contrast to Berger's work in [1], we consider the interactions of a set of leaky bucket regulators, searching for a vector of burst parameters of the leaky bucket regulators to meet the end-to-end deadlines of all HRT connections.

Our algorithm for searching burst parameters of leaky buckets is *optimal* in that it can find the vector of burst parameters whose mean value is minimal and by which the delay requirements of all HRT connections can be met whenever such an assignment exists. Our algorithm is computationally efficient and can be utilized during connection setup. The results presented in this paper are directly applicable to ATM networks that are available currently, without making any modifications to the hardware. The results are also compatible with the proposed ATM standards. We evaluate the system's capability to support hard real-time connections in terms of a metric called *admission probability* [18]. *Admission probability* is the probability of meeting the end-to-end deadlines of a set of randomly chosen connections. We have observed that the admission probability increases with a proper choice of leaky bucket parameters at the UNI.

Although we focus on traffic regulation for meeting end-to-end deadlines, our work also complements many of the previous studies that concentrate essentially on designing and analyzing scheduling policies for ATM switches [2], [5], [6], [8], [10], [11], [12], [13], [16], [20], [21], [23], [24], [25]. A modified FCFS scheduling scheme called $FIFO^+$ was proposed and studied in [2]. The switch scheduling policy called "Stop and Go" is presented in [8]. A virtual clock scheduling scheme in which cells are prioritized by a virtual time stamp assigned to them, is discussed in [25]. The use of the Earliest Deadline First scheduling in wide area networks has also been investigated [6]. Zhang and Ferrari [23] discuss scheduling at the output link by introducing a regulator at each link of an ATM switch. Kweon and Shin [12] use the rate-monotonic scheduling policy in which the input to the scheduler is constrained by regulating the traffic of each connection traversing the scheduler. Scheduling policies based on fair queueing and its derivations are discussed in [5], [16]. In our analysis of network delays, we assume the output link scheduling policy used is FCFS. However, our analysis and methodology can be applied easily to systems using other scheduling policies.

The outline of the rest of the paper is as follows. A glossary of terms is shown in Table 1. 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. Section 6 provides a summary of results and conclusions.

2 System Model

This section presents 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

Fig. 1 shows a typical ATM LAN. In ATM networks [4], [9], [22], 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 as a discrete quantity with the cell transmission time (CT) being taken as one time unit.

ATM is a connection-oriented transport technology. A connection has to be set up between two hosts before they can begin communication. Fig. 2a shows a sequence of network components that constitute a typical connection (illustrated by a connection from Host 1 to Host 2 in Fig. 1). Cells of 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. There are two parameters associated with each leaky bucket regulator: 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, is the token generation rate in the bucket. The number of cells that may be transmitted by a leaky bucket regulator in any interval of length I is bounded by $\beta + \lfloor \rho \cdot I \rfloor$.

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

2.2 Connection Model

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

TABLE 1
Glossary of Terms

Term	Definition	Section of first use
$\mathcal{M} = \{M_1, M_2, \ldots, M_N\}$	The set of N connections.	2.2
$\vec{\boldsymbol{D}} = < D_1, D_2, \ldots, D_i, \ldots, D_N >$	A vector of end-to-end deadlines for connections in $\mathcal{M}$.	2.2
$\vec{\boldsymbol{\beta}} = < \beta_1, \beta_2, \ldots, \beta_i, \ldots, \beta_N >$	A vector of burst parameters of the leaky bucket for connections in $\mathcal{M}$.	3
$\vec{\boldsymbol{\rho}} = < \rho_1, \rho_2, \ldots, \rho_i, \ldots, \rho_N >$	A vector of rate parameters of the leaky bucket for connections in $\mathcal{M}$.	3
$\vec{\boldsymbol{d}} = < d_1, d_2, \ldots, d_i, \ldots, d_N >$	A vector of the worst case end-to-end delays for connections in $\mathcal{M}$.	2.3
$\vec{\boldsymbol{d}}\,(\vec{\boldsymbol{\beta}}) = < d_1(\vec{\beta}), d_2(\vec{\beta}), \ldots, d_N(\vec{\beta}) >$	A vector of the worst case end-to-end delays for a given of $\vec{\beta}$.	3
$\tilde{A}_M$	The set of *all* burst vectors.	4.1
$\mathcal{A}_{\mathcal{M}} = \{\vec{\boldsymbol{\beta}} \mid \vec{\boldsymbol{d}}\,(\vec{\boldsymbol{\beta}}) \leq \vec{\boldsymbol{D}}\ \}$	The set of burst vectors for which the connection set $\mathcal{M}$ is admissible.	3
$\vec{\beta}^*$	The minimum $\vec{\beta}$ in $\mathcal{A}_{\mathcal{M}}$ in terms of its norm.	3
β_i^{max}	The minimum value of β_i for which d_i^{lb} is zero.	4.2
$\vec{\beta}^{max} = < \beta_1^{max}, \beta_2^{max}, \ldots, \beta_N^{max} >$	The burst vector that specifies the values of β^{max} for connections in $\mathcal{M}$.	4.2
d_i^{const}	The summation of the delays a cell of connection M_i suffers at all the constant delay servers in its route.	2.3
$d_{i,j}^{fcfs}$	The maximum delay experienced by a cell of connection M_i at FCFS server j.	2.3
d_i^{lb}	The worst case queueing delay experienced by a cell at the leaky bucket of M_i.	2.3
d_i^{net}	The sum of the worst case queueing delays a cell of M_i suffers at all the variable delay servers.	2.3
$(C_{i,lb}^{in\_server}, P_{i,lb}^{in\_server})$	The input traffic descriptor to the leaky bucket of M_i.	4.2
$\Gamma(I)$	Parameters of the rate-function traffic descriptor.	2.2
$\Phi_{i,lb}(I)$	The number of cells of M_i that can arrive at the input of the leaky bucket in an interval I.	A.1
$Q_{i,lb}$	The maximum queue length at the leaky bucket of M_i.	A.1

concerning a set of HRT connections. Hereafter, we will omit the qualifier "HRT" for connections since we only deal 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:

$$\mathcal{M} = \{M_1, M_2, \ldots, M_i, \ldots, M_N\}, \quad (1)$$

  where connection M_i is specified by the following information:

  - *Source address,*
  - *Destination address,*
  - *Connection route,*[1]
  - *End-to-end deadline* (D_i),
  - *An upper bound on average cell arrival rate,*
  - *Worst case input traffic characteristics.*
- $\vec{D}$ is a vector that specifies the end-to-end deadlines of connections in $\mathcal{M}$:

$$\vec{D} = < D_1, D_2, \ldots, D_i, \ldots, D_N >, \quad (2)$$

  where D_i is the end-to-end deadline of a cell of connection M_i. That is, if a cell arrives at the source

1. To ensure stability, we assume that connection routes do not form loops [3], [16].

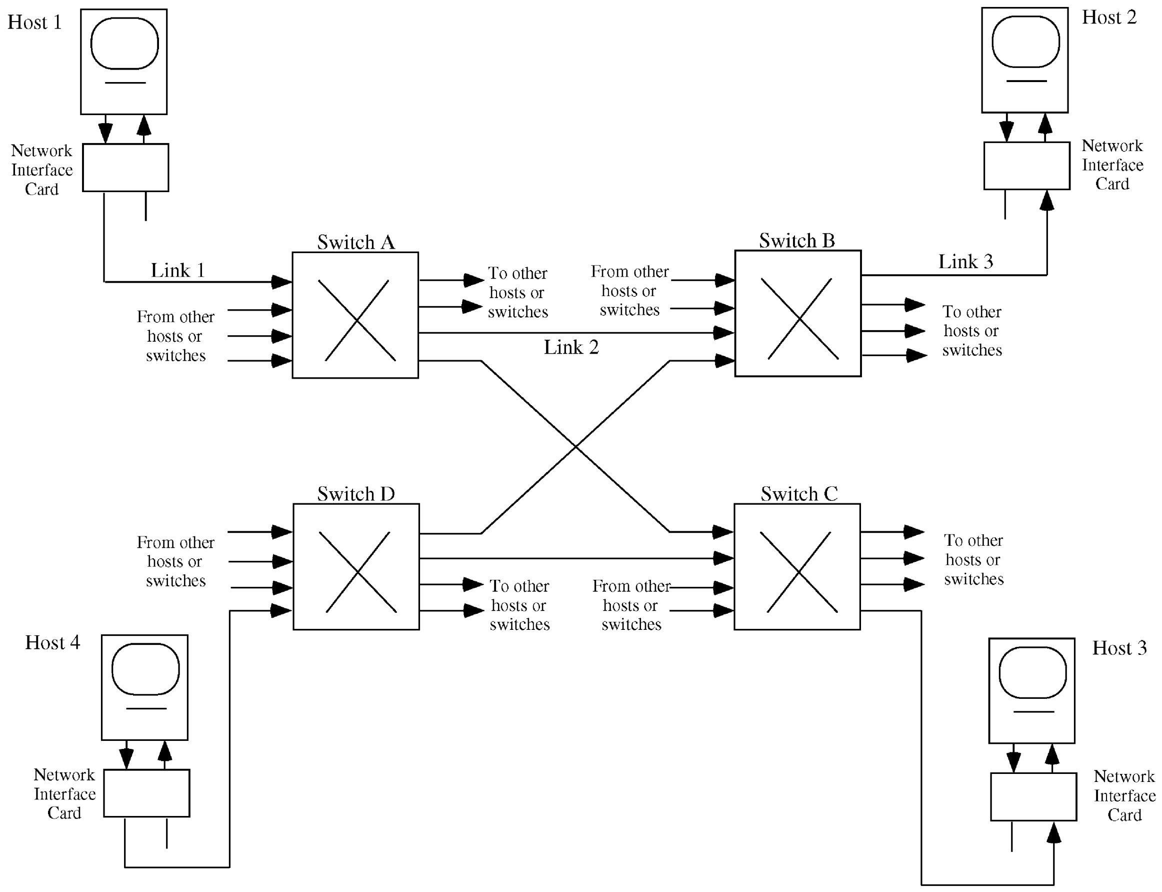

Host 1
Network Interface Card
Host 2
Network Interface Card
Switch A
Link 1
From other hosts or switches
To other hosts or switches
From other hosts or switches
Switch B
Link 3
To other hosts or switches
Link 2
Switch D
Switch C
From other hosts or switches
To other hosts or switches
From other hosts or switches
To other hosts or switches
Host 4
Network Interface Card
Host 3
Network Interface Card

Fig. 1. ATM network architecture.

at time t, then it should reach the destination by $t + D_i$.

In a connection (see Fig. 2a), 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* [3], [18]. Servers can be classified as *constant delay servers* and *variable delay servers*. A constant delay server is one that offers a fixed delay to every arriving cell. For example, physical links are considered 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. Fig. 2b shows the logical representation of the connection in Fig. 2a.

The traffic pattern of a connection at a point in the network is characterized by a *traffic descriptor*. The traffic at the source of a connection is the raw traffic (unregulated) generated by applications. It is described by the *periodic descriptor* (C, P), which means that a *maximum* of C cells may arrive at the connection in any interval of length P. The periodic descriptor is very general to describe real-time traffic at the application level. The classical periodic or *synchronous* traffic (i.e., C contiguous cells arriving at the beginning of every period of length P) is a special case of this kind of traffic. Most hard real-time traffic (at source) is assumed to be synchronous [14], [15] and, hence, is specified adequately by this traffic descriptor.

The raw traffic of a connection is regulated by the leaky bucket regulator before it gets into the network. After being regulated by a leaky bucket with parameters (β, ρ), the traffic pattern becomes $\beta + \lfloor \rho \cdot I \rfloor$ for any interval of length I. After the regulation, the traffic traverses through ATM switches. The traffic pattern will become increasingly irregular as cells are multiplexed and buffered at the switches. For the description of more general traffic patterns, we use *rate function* descriptor [17], [18], $\Gamma(I)$, to describe the traffic after leaky bucket regulation. $\Gamma(I)$ specifies the maximum arrival rate of cells in any interval of length I. That is, a maximum of $I \cdot \Gamma(I)$ cells of the connection may arrive in any interval of length I. $\Gamma(I)$ is general enough to describe any traffic pattern. For example, the traffic pattern described by $\beta + \lfloor \rho \cdot I \rfloor$ can be expressed by the rate function in (61) in Appendix A.1.

The following notations are used in the rest of the paper to describe traffic at different network points:

- $(C_{i,lb}^{in\_server}, P_{i,lb}^{in\_server})$: input traffic to the leaky bucket of connection M_i.
- $(\Gamma_{i,j}^{in\_server}(I))$ and $(\Gamma_{i,j}^{out\_server}(I))$: input and output traffic at FCFS server j of connection M_i.

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 do not exceed 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,

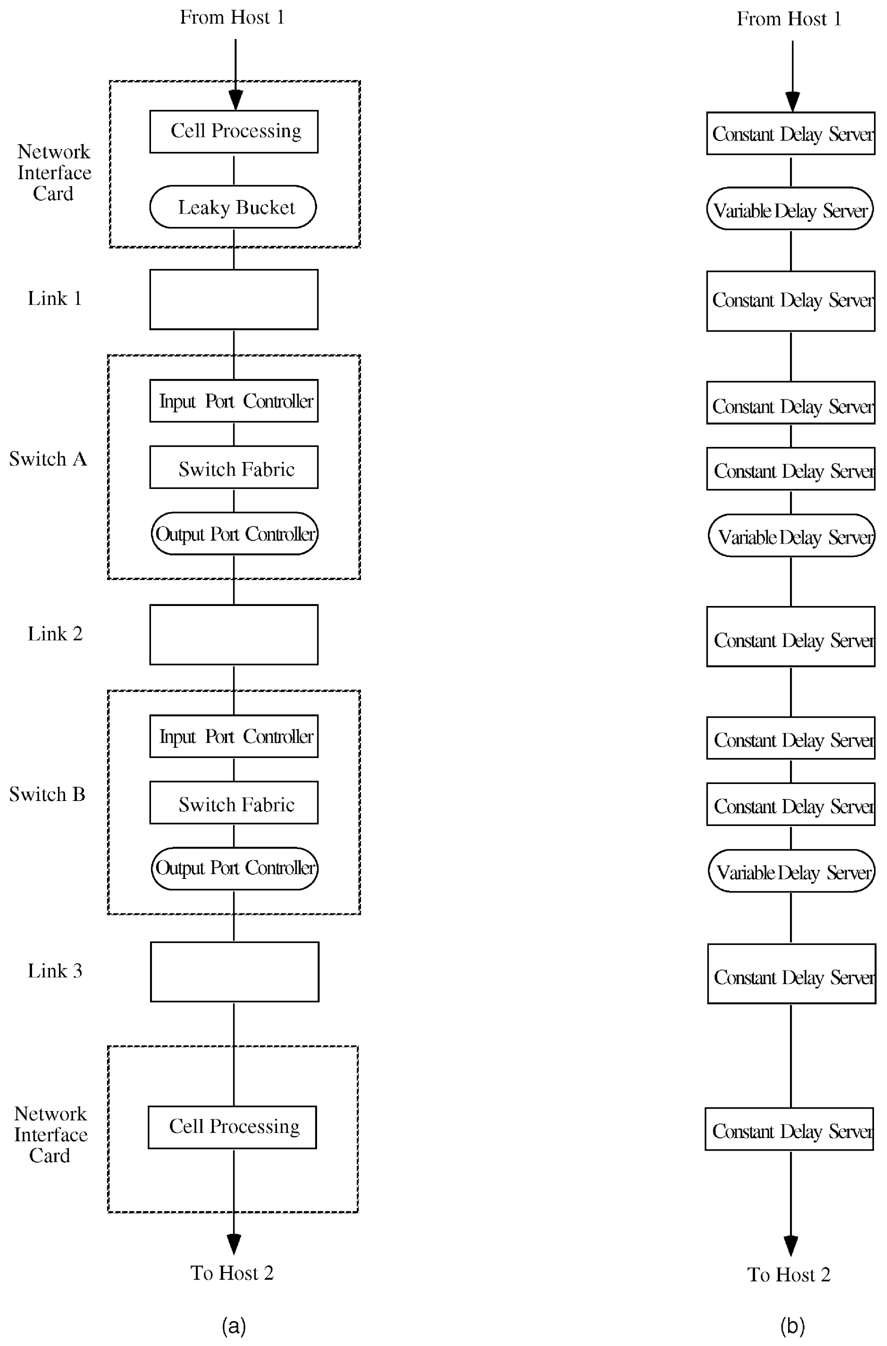


Fig. 2. Connection decomposition into servers. (a) The devices and links traversed by the connection. (b) The sequence of servers traversed by the connection.

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

where d_i is the worst case 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

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

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

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

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

Consider connection M_i, $M_i \in \mathcal{M}$. To compute d_i, we need to investigate the delays in every network component traversed by a cell of M_i. Now, we can decompose d_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 queuing delay experienced by a cell at the leaky bucket which regulates

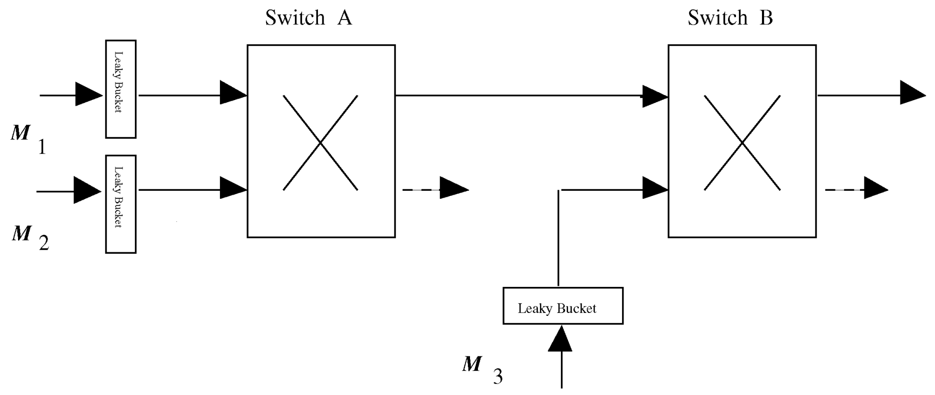


Fig. 3. Experimental setup.

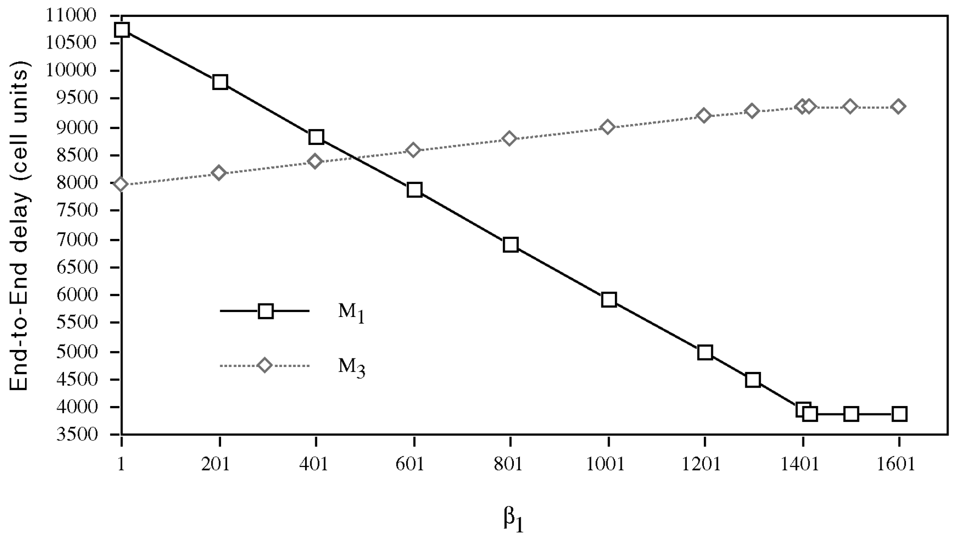


Fig. 4. Experimental results.

M_i's traffic. When M_i's traffic is not regulated, d_i^{lb} is 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: Suppose the route of connection M_i traverses switches (FCFS servers) $1, 2, \ldots, K$. Let $d_{i,j}^{fcfs}$ be the cell delay of connection M_i at FCFS server j. d_i^{net} can thus be expressed as:

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

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

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

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

3 Problem Definition

In this section, we formally define the problem of leaky bucket parameter selection for HRT ATM networks. First, we examine some experimental data. We consider a simple network consisting of two ATM switches (see Fig. 3). Each ATM switch has two input lines and two output lines. There are three connections in the system, M_1, M_2, and M_3. All three connections share the same output link at the second switch (switch B). Connections M_1 and M_2 enter the network at switch A and traverse through switches A and B. Connection M_3 enters the system at switch B and traverses switch B only. The three connections carry identical streams of video data. The video source under study is a two-hour encoding of the movie "Starwars" [7]. 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. Fig. 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. Furthermore, we observe that, for $\beta_1 > 1{,}400$, the delays experienced by M_1 and M_3 reach constant values. An intuitive explanation of these results is provided below.

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 ($\beta_1 > 1,400$), 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 on M_1's traffic. In this experiment, M_2's traffic was not monitored because it is expected to exhibit the same trend as M_3 when β_1 increases.

The experimental results shown in Fig. 4 indicate clearly that the choice of β_is for different connections plays a critical role in the worst case cell delays experienced by all the connections. Before the formal definition of the problem, we need some notations:

- $\vec{\rho}$ is the *rate vector*, i.e.,

$$\vec{\rho} = < \rho_1, \rho_2, \ldots, \rho_i, \ldots, \rho_N >, \tag{8}$$

  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 average cell arrival rate of M_i.
- $\vec{\beta}$ is the *burst vector*, i.e.,

$$\vec{\beta} = < \beta_1, \beta_2, \ldots, \beta_i, \ldots, \beta_N >, \tag{9}$$

  where β_i is the burst parameter assigned to the leaky bucket regulating 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 takes positive integers only. Therefore, the minimal value of $m(\vec{\beta})$ is N.
- $\vec{d}(\vec{\beta})$ is the worst case end-to-end delay vector for a given selection of $\vec{\beta}$, i.e.,

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

  where $d_i(\vec{\beta})$ is the worst case cell delay of connection M_i when $\vec{\beta}$ is the chosen burst vector. Using (7), $d_i(\vec{\beta})$ can be expressed as

$$\begin{aligned} d_i(\vec{\beta}) &= d_i^{const} + d_i^{lb}(\vec{\beta}) + d_i^{net}(\vec{\beta}) \\ &= d_i^{const} + d_i^{lb}(\vec{\beta}) + d_{i,1}^{fcfs}(\vec{\beta}) + d_{i,2}^{fcfs}(\vec{\beta}) + \ldots \\ &\quad + d_{i,K}^{fcfs}(\vec{\beta}), \end{aligned} \tag{12}$$

  where $d_i^{lb}(\vec{\beta})$ and $d_{i,j}^{fcfs}(\vec{\beta})$ are the worst case queuing delays at the leaky bucket and at jth FCFS server, respectively, when the burst vector is $\vec{\beta}$. The computation of $d_i^{lb}(\vec{\beta})$ and $d_{i,j}^{fcfs}(\vec{\beta})$ are given by (60) and (63), respectively, in Appendix A.
- $\mathcal{A}_{\mathcal{M}}$ is the set of *burst vectors* for which the connection set $\mathcal{M}$ is admissible, i.e.,

$$\mathcal{A}_{\mathcal{M}} = \{\vec{\beta} \mid \vec{d}(\vec{\beta}) \leq \vec{D}\}. \tag{13}$$

Our main goal is to ensure that the end-to-end deadlines of all the connections in a given set are met, i.e., to meet (5). We have chosen traffic regulation as our means of achieving this objective. 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 will 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.

When $\mathcal{A}_{\mathcal{M}}$ is empty, it is clear that no assignment of $\vec{\beta}$ can make the connection set admissible. Our parameter selection algorithm will return an all-zero $\vec{\beta}$ when $\mathcal{A}_{\mathcal{M}}$ is empty. Furthermore, since the degree of regulating the traffic stream from M_i is higher for smaller values of β_i (i.e. the regulated traffic is less bursty and thus has less impact to others), it is desirable to select a vector $\vec{\beta}$ having a small norm, $m(\vec{\beta})$. Let $\vec{\beta}^*$ be the minimum in $\mathcal{A}_{\mathcal{M}}$ in terms of its norm. That is, $\vec{\beta}^*$ satisfies:

$$m(\vec{\beta}^*) = \min_{\vec{\beta}\in\mathcal{A}_{\mathcal{M}}} (m(\vec{\beta})). \tag{14}$$

We define a $\vec{\beta}$-selection algorithm to be *optimal* if it always produces the $\vec{\beta}^*$ whenever $\mathcal{A}_{\mathcal{M}}$ is nonempty.

We now prove that $\vec{\beta}^*$ is unique if it exists for a set of connections $\mathcal{M}$. The following lemma is introduced to illustrate the relationships between $\vec{\beta}$ and $d_i^{net}(\vec{\beta})$.

Lemma 3.1 *Consider a connection set $\mathcal{M}$. $d_i^{net}(\vec{\beta})$ does not decrease as $\vec{\beta}$ increases.*

Lemma 3.1 is intuitively valid. As $\vec{\beta}$ increases, more cells of those connections whose β values are increased will get through the leaky buckets and be injected into the network. This will add more cells to the following FCFS servers. Therefore, $d_i^{net}(\vec{\beta})$, the total delays of an M_i's cell at all the FCFS servers, will increase or remain the same if the servers still have the capacity to transmit the increased amount of cells without buffering them. So, $d_i^{net}(\vec{\beta})$ will never decrease as $\vec{\beta}$ increases. The formal proof of Lemma 3.1 is given in [17].

The next theorem proves the uniqueness of $\vec{\beta}^*$.

Theorem 3.1 *For a given system, $\vec{\beta}^*$, which satisfies*

$$m(\vec{\beta}^*) = \min_{\vec{\beta}\in\mathcal{A}_{\mathcal{M}}} (m(\vec{\beta})), \tag{15}$$

is unique if $\mathcal{A}_{\mathcal{M}} \neq \emptyset$.

Proof. We prove the theorem by contradiction. Assume that $\mathcal{A}_{\mathcal{M}} \neq \emptyset$ and that there are two distinct vectors $\vec{\beta}' =< \beta_1', \ldots, \beta_N' >$ and $\vec{\beta}'' =< \beta_1'', \ldots, \beta_N'' >$ which satisfy (15). Thus,

$$m(\vec{\beta}') = \sum_{i=1}^{N} \beta_i' = \sum_{i=1}^{N} \beta_i'' = m(\vec{\beta}'') = \min_{\vec{\beta}\in\mathcal{A}_{\mathcal{M}}} (m(\vec{\beta})). \tag{16}$$

We can construct a new vector $\vec{\beta} =< \beta_1, \ldots, \beta_N >$ from $\vec{\beta}'$ and $\vec{\beta}''$ such that for $j = 1, \ldots, N$,

$$\beta_j = \min(\beta_j', \beta_j''). \tag{17}$$

Using (17) in (10), we have

$$m(\vec{\beta}) = \sum_{i=1}^{N} \beta_i < m(\vec{\beta}') = m(\vec{\beta}''). \tag{18}$$

Since a leaky bucket regulator is at the entrance of each connection, $d_i^{lb}(\vec{\beta})$ is independent from any β_j where $j \neq i$. Thus, we have:

$$d_i^{lb}(\vec{\beta}) = d_i^{lb}(\beta_i). \tag{19}$$

Then, because of (17), we have, for $1 \leq i \leq N$

$$d_i^{lb}(\vec{\beta}) = \begin{cases} d_i^{lb}(\vec{\beta}') & \text{if } \beta_i = \beta_i'. \\ d_i^{lb}(\vec{\beta}'') & \text{if } \beta_i = \beta_i''. \end{cases} \tag{20}$$

Also, because $\vec{\beta} \leq \vec{\beta}'$ and $\vec{\beta} \leq \vec{\beta}''$, from Lemma 3.1, we have,

$$d_i^{net}(\vec{\beta}) \leq \min(d_i^{net}(\vec{\beta}'), d_i^{net}(\vec{\beta}'')). \tag{21}$$

Further, because $\vec{\beta}'$ and $\vec{\beta}''$ are distinct, for a given i, $1 \leq i \leq N$, we have two cases:

Case 1: $\beta_i < \beta_i'$

That is, $\beta_i = \beta_i''$. Hence, from (20), we get

$$d_i^{lb}(\vec{\beta}) = d_i^{lb}(\vec{\beta}'') \tag{22}$$

and, from (21), we get

$$d_i^{net}(\vec{\beta}) \leq d_i^{net}(\vec{\beta}''). \tag{23}$$

Therefore,

$$d_i^{lb}(\vec{\beta}) + d_i^{net}(\vec{\beta}) \leq d_i^{lb}(\vec{\beta}'') + d_i^{net}(\vec{\beta}''). \tag{24}$$

But, $\vec{\beta}''$ satisfies (15). Therefore,

$$d_i^{lb}(\vec{\beta}) + d_i^{net}(\vec{\beta}) \leq D_i. \tag{25}$$

Case 2: $\beta_i = \beta_i'$

Hence, from (20) we get

$$d_i^{lb}(\vec{\beta}) = d_i^{lb}(\vec{\beta}') \tag{26}$$

and, from (21), we get

$$d_i^{net}(\vec{\beta}) \leq d_i^{net}(\vec{\beta}'). \tag{27}$$

Therefore,

$$d_i^{lb}(\vec{\beta}) + d_i^{net}(\vec{\beta}) \leq d_i^{lb}(\vec{\beta}') + d_i^{net}(\vec{\beta}'). \tag{28}$$

But, $\vec{\beta}'$ satisfies (15). Therefore,

$$d_i^{lb}(\vec{\beta}) + d_i^{net}(\vec{\beta}) \leq D_i. \tag{29}$$

Because of (25) and (29), $\vec{\beta}$ is also a feasible assignment for which the connection set is admissible. However, because of (18) and the definition of $\vec{\beta}^*$, $\vec{\beta}'$ and $\vec{\beta}''$ cannot be $\vec{\beta}^*$. This is a contradiction. Hence, the theorem holds. □

4 Algorithm Development

In this section, we develop an *optimal* and efficient 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 Parameter Selection as a Search Problem

By definition, an 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 $\exists j$, $1 \leq j \leq N$, such that

$$\beta_i' = \begin{cases} \beta_i + 1 & \text{if } i = j \\ \beta_i & \text{if } i \neq j. \end{cases} \tag{30}$$

Note that $m(\vec{\beta}') = m(\vec{\beta}) + 1$ and $\vec{\beta}'$ differs from $\vec{\beta}$ only in the jth component. 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 over $\tilde{\mathcal{A}}_M$, with a node set $\mathcal{V}$ and an edge set $\mathcal{E}$ given by

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

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 *levelp* consists of all $\vec{\beta}$ vectors having norm p ($p \geq N$). Fig. 5 illustrates such a graph when $N = 3$.

In graph G, let $\mathcal{L}_p$ be the set of all $\vec{\beta}$ vectors at level p. Note that a node at level p can have edges only to nodes at level $p + 1$. Based on this representation of the search space, a simple breadth-first search method can be constructed to find $\vec{\beta}^*$. Fig. 6 shows the pseudocode for such a method. This search method first examines all $\vec{\beta}$ vectors in $\mathcal{L}_p$ before proceeding to those in $\mathcal{L}_{p+1}$, shown by the dotted search path in Fig. 5. For each $\vec{\beta}$ vector considered, the method uses the procedure[2] 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 following questions about the method shown in Fig. 6 may be raised:

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 algorithm of searching $\vec{\beta}^*$ will not terminate.

2. Details of procedure Computer_$\vec{d}(\vec{\beta})$ are in Appendix A.3. The final version of Computer_$\vec{d}$ has one more parameter, which will be discussed later.

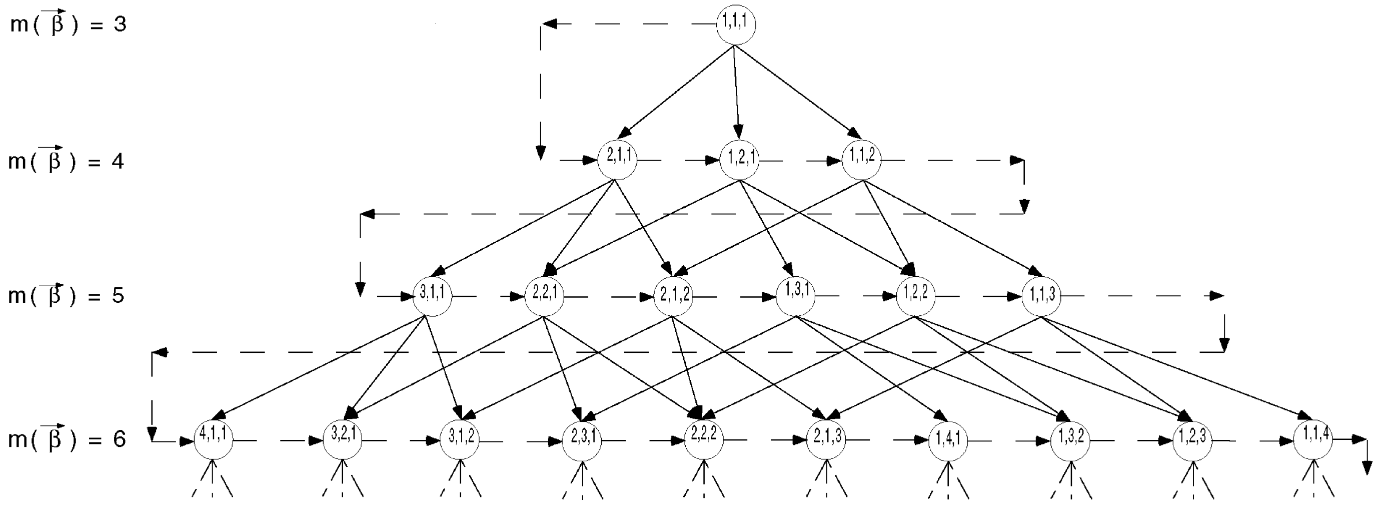


Fig. 5. Example search space graph.

2. Even if $\mathcal{A}_{\mathcal{M}}$ is nonempty, the exhaustive nature of the breadth-first search results in exponential time complexity.

In the next subsection, we overcome the first difficulty by bounding the search space. In the subsequent subsection, we reduce further 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 the experiment described in Section 3, an interesting observation was that when β_1 was increased beyond $1,400$, 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 in any ATM network and leaky bucket regulators. Let β_i^{max} be the minimum value of β_i for which d_i^{lb} is zero.

Theorem 4.1. *For a connection M_i whose traffic is described by $(C_{i,lb}^{in\_server}, P_{i,lb}^{in\_server})$ and regulated by a leaky bucket with parameters (β_i, ρ_i), β_i^{max} is bounded by $C_{i,lb}^{in\_server}$.*

Proof. Let $Q_{i,lb}$ be the maximal queue length at the leaky bucket of M_i. From Lemma A.2 in Appendix A.1, we have

$$Q_{i,lb} = C_{i,lb}^{in\_server} - \beta_i. \tag{31}$$

Since β_i^{max} is defined to be the smallest integer value for which the maximum waiting time of a cell at the leaky bucket queue becomes zero, i.e., $Q_{i,lb}$ becomes zero, solve for the value of β_i which makes (31) zero. We have

$$\beta_i^{max} = C_{i,lb}^{in\_server}. \tag{32}$$

□

Theorem 4.2. *For any connection M_i, increasing the value of β_i beyond β_i^{max} has no impact on the worst case end-to-end cell delay of M_i or any other connection.*

Proof. For a connection M_i, since β_i^{max} is the minimal value of β_i for which d_i^{lb} is zero, no cell of M_i is queued at its leaky bucket when β_i reaches β_i^{max}. That is, any cell that arrives at the leaky bucket will get through the leaky bucket without being buffered. There is virtually no traffic regulation on M_i's traffic in this case. Therefore, increasing the value of β_i beyond β_i^{max} has no impact on the worst case end-to-end cell delay of M_i or any other connection. □

An important consequence of Theorem 4.2 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

$$\vec{\beta} \leq \vec{\beta}^{max}. \tag{33}$$

Note that $\vec{\beta}^{max}$ can be precomputed for a given set $\mathcal{M}$.

Consider the example in Fig. 5. If we assume that $\vec{\beta}^{max} = <1,2,3>$, then, after applying Theorem 4.2, we get another graph shown in Fig. 8. The shaded region in Fig. 8 is automatically eliminated from consideration.

Using Theorem 4.2, we modify the breadth-first search procedure shown in Fig. 6 to take into account the bounded search space. The resulting pseudocode is shown in Fig. 7. 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 algorithm.

4.3 Search Space Pruning

The breadth-first search algorithm defined in Fig. 7 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 allow us to examine *only one* vector $\vec{\beta}$ at each level.

Consider the example in Fig. 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

```
Select_Burst_Parameters($\mathcal{M}$)
01.  for $p = N$ to $\infty$ do                          {Main iterative loop}
02.      foreach $\vec{\beta} \in \mathcal{L}_p$ do
03.          $\vec{d}$ = Compute_$\vec{d}$ ($\vec{\beta}$);
04.          if ($\vec{d} \le \vec{D}$ ) then
05.              return($\vec{\beta}$);
06.          endif
07.      endforeach
08.  endfor                                           {End of main loop}.
```

Fig. 6. Pseudocode of the breadth-first search method.

$\vec{\beta}^* = < 1, 2, 2 >$. We would like our search to be guided along 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 guide the search along the direct path from the root to $\vec{\beta}^*$ in G, we must choose an appropriate *child* node at each level. For example, in Fig. 9, at node $< 1, 1, 1 >$, we may choose either $< 1, 2, 1 >$ or $< 1, 1, 2 >$ as our next candidate node. However, if we are at node $< 1, 1, 2 >$, we must guide our search to choose $< 1, 2, 2 >$ and not $< 1, 1, 3 >$ as the next candidate node.

In order to select 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 Fig. 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:

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 $\tilde{\mathcal{A}}_M$ 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 $\vec{s}(\vec{\beta}) = < s_1, s_2, \ldots, s_N >$ be the *status vector* associated with a node $\vec{\beta}$, where

$$s_i(\vec{\beta}) = \begin{cases} 1 & \text{if } d_i(\vec{\beta}) \le D_i \\ 0 & \text{otherwise.} \end{cases} \tag{34}$$

Vector $\vec{s}(\vec{\beta})$ 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.

The following lemmas are introduced to help the proof of the theorem which defines the direct search of $\vec{\beta}^*$.

Lemma 4.1. *Consider a connection set $\mathcal{M}$. $d_j(\vec{\beta})$ does not decrease as the increase of β_is for any i where $i \ne j$.*

Lemma 4.1 is valid. As we have said before, $d_j(\vec{\beta}) = d_j^{const} + d_j^{lb}(\vec{\beta}) + d_j^{net}(\vec{\beta})$. d_j^{const} is a constant and $d_j^{lb}(\vec{\beta})$ is independant from β_i if $j \ne i$. Only $d_j^{net}(\vec{\beta})$ changes as the increase of β_is for any i where $i \ne j$. From Lemma 3.1, we have that $d_j^{net}(\vec{\beta})$ does not decrease as the increase of $\vec{\beta}$. Therefore, Lemma 4.1 holds. The formal proof of this lemma can be found in [17].

Lemma 4.2. *Consider a connection set $\mathcal{M}$ and assume that $\vec{\beta}^*$ exists for $\mathcal{M}$. If $\vec{\beta}$ is an ancestor of $\vec{\beta}^*$, for any y $(1 \le y \le N)$ which makes $s_y(\vec{\beta}) = 0$, then $\beta_y < \beta_y^*$.*

Proof. Since $\vec{\beta}$ is an ancestor of $\vec{\beta}^*$, we have $\vec{\beta} < \vec{\beta}^*$. That is,

$$\forall y,\ 1 \le y \le N,\ \ \beta_y \le \beta_y^*. \tag{35}$$

```
Select_Burst_Parameters($\mathcal{M}$)
01.  Compute($\vec{\beta}^{max}$);
02.  for $p = N$ to $m(\vec{\beta}^{max})$ do                  {Main iterative loop}
03.      foreach $\vec{\beta} \in \mathcal{L}_p$ and $\vec{\beta} \le \vec{\beta}^{max}$ do
04.          $\vec{d}$ = Compute_$\vec{d}$ ($\vec{\beta}$);
05.          if ($\vec{d} \le \vec{D}$ ) then
06.              return($\vec{\beta}$);
07.          endif
08.      endforeach
09.  endfor                                           {End of main loop}
10.  return ($< 0, 0, \ldots, 0 >$).
```

Fig. 7. Pseudocode of the bounded breadth-first search algorithm.

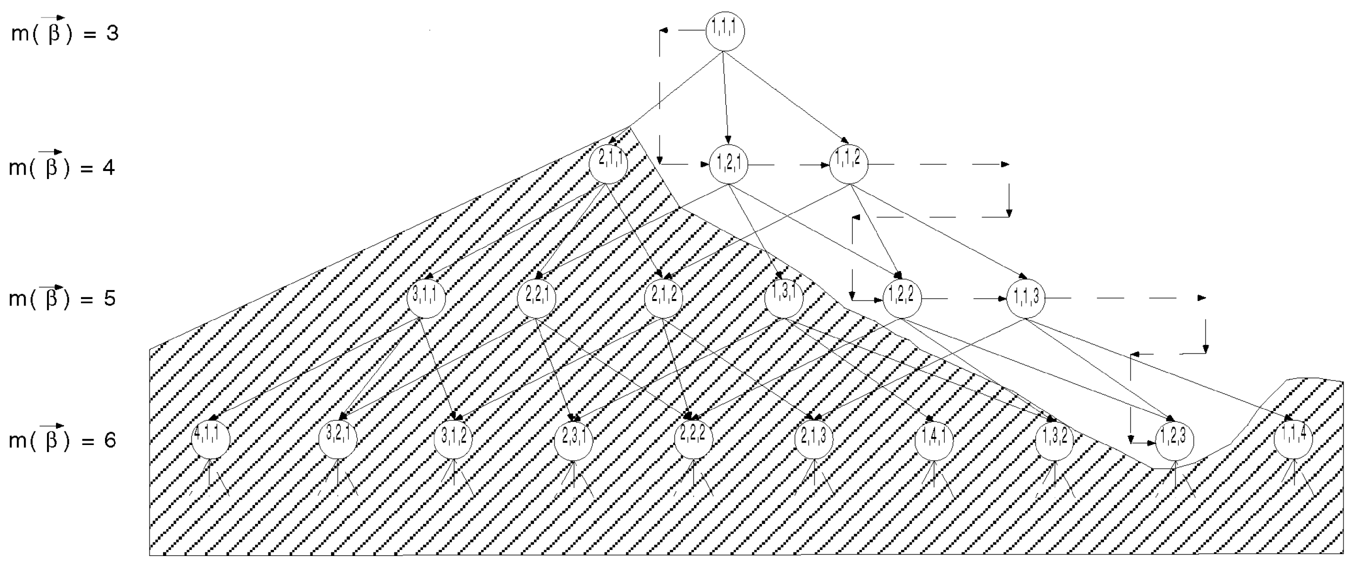


Fig. 8. Example of bounded search space.

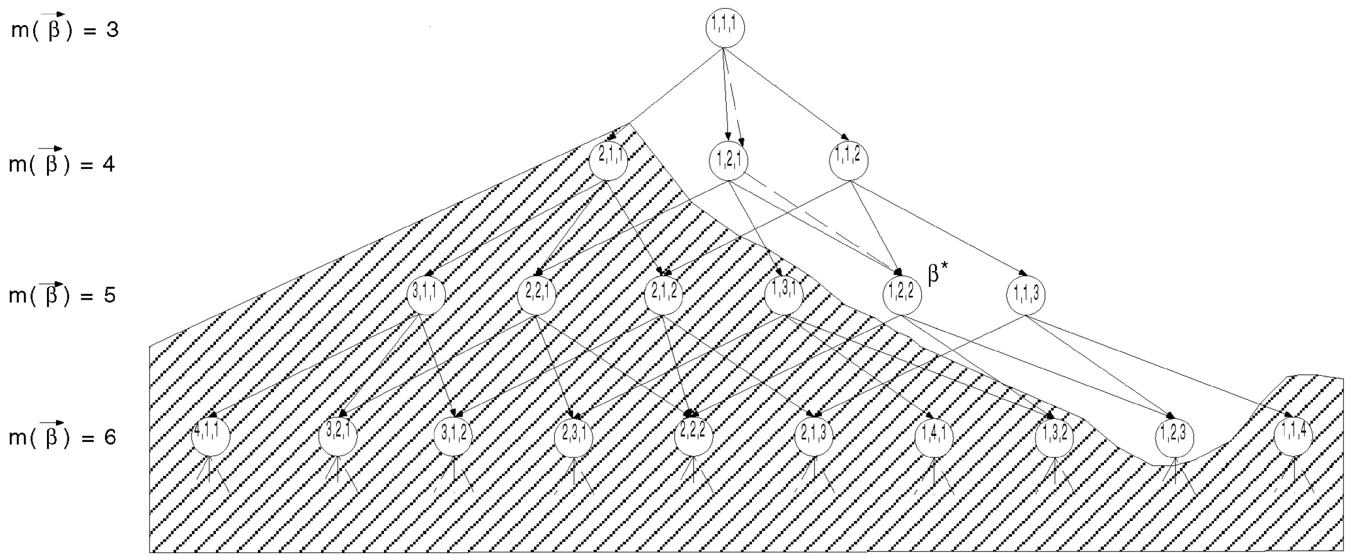


Fig. 9. Example graph of search space after pruning.

Now, we prove: For any y $(1 \leq y \leq N)$ if $s_y(\vec{\beta}) = 0$, then $\beta_y < \beta^*_y$. Because of (35), we only need to prove $\beta_y \neq \beta^*_y$. We prove it by contradiction.

Assume $\beta_y = \beta^*_y$. Since $s_y(\vec{\beta}) = 0$, we have $d_y(\vec{\beta}) > D_y$. From Lemma 4.1, we have that $d_y(\vec{\beta})$ does not decrease no matter how much we increase β_is if $i \neq y$. We now increment the corresponding components of $\vec{\beta}$ to make $\vec{\beta}$ equal to $\vec{\beta}^*$. $d_y(\vec{\beta}^*) > D_y$ still holds. This contradicts the definition of $\vec{\beta}^*$ and the existance assumption of $\vec{\beta}^*$. Therefore, $\beta_y \neq \beta^*_y$. So, $\beta_y < \beta^*_y$. □

Theorem 4.3. *If $\vec{\beta}^*$ exists, then the following are true:*

1. $< 1, 1, \ldots, 1 >$ *is an ancestor of $\vec{\beta}^*$, and*
2. *If $\vec{\beta}$ is an ancestor of $\vec{\beta}^*$, then there exists a $\vec{\beta}'$ such that*
   a) $\vec{\beta} \longrightarrow \vec{\beta}'$,
   b) $s_{\triangle(\vec{\beta},\vec{\beta}')}(\vec{\beta}) = 0$, *and*
   c) *Either $\vec{\beta}'$ is $\vec{\beta}^*$ or an ancestor of $\vec{\beta}^*$.*

Proof. Statement 1 is true since $< 1, 1, \ldots, 1 >$ is an ancestor of all $\vec{\beta}$s.

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

$$\beta_x \leq \beta^*_x. \tag{36}$$

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

Then, from Lemma 4.2, we have

$$\beta_y < \beta^*_y. \tag{37}$$

Now, construct $\vec{\beta}'$ such that, for all z, $z \neq y$ $\beta'_z = \beta_z$ and $\beta'_y = \beta_y + 1$.

From the definition of "$\longrightarrow$," we have, obviously,

$$\vec{\beta} \longrightarrow \vec{\beta}'. \tag{38}$$

Thus, a) holds.

From the definition of "$\triangle$," we have $\triangle(\vec{\beta}, \vec{\beta}') = y$. Also, because $s_y(\vec{\beta}) = 0$, that is

$$s_{\triangle(\vec{\beta},\vec{\beta}')}(\vec{\beta}) = 0. \tag{39}$$

Thus, b) holds.

By (36) and (37), for all x, $1 \leq x \leq N$, we have

$$\beta'_x \leq \beta^*_x. \tag{40}$$

Thus,

$$\vec{\beta}' \leq \vec{\beta}^*. \tag{41}$$

Hence, either $\vec{\beta}' = \vec{\beta}^*$ or there exists a directed path in G such that

$$\vec{\beta}' \xrightarrow{*} \vec{\beta}^*. \tag{42}$$

So, $\vec{\beta}'$ is an ancestor of $\vec{\beta}^*$. Thus, c) holds. □

Now, consider the claims made by the above theorem. The first claim in Theorem 4.3 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 4.3 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 from the root node. Once we are at level p examining a node $\vec{\beta} \in \mathcal{L}_p$, the second claim helps us to choose the child node of $\vec{\beta}$ if $\vec{d}(\vec{\beta}) > \vec{D}$. The theorem states that we can 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, $\vec{\beta}^*$ can be found by our search starting from $< 1, 1, \ldots, 1 >$ and using the status vector $\vec{s}$ to guide the search along a directed path leading to $\vec{\beta}^*$.

4.4 The Efficient Algorithm and Its Properties

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

Fig. 10 shows the pseudocode of the algorithm. The algorithm is derived from the one in Fig. 7 by pruning the search space. The algorithm is an iterative procedure, starting from the root, i.e., $< 1, 1, \ldots, 1 >$. During each iteration, the algorithm selects a node from the next level. The node is selected (line 9) with the help of status vector $\vec{s}$ (computed by function $\mathrm{Compute\_}\vec{s}(\vec{d}, \vec{D})$ in line 8). This iterative process continues until either $\vec{\beta}^*$ is found or, for some j, $\beta_j > \beta_j^{max}$.[3]

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

Theorem 4.4. *For a connection set $\mathcal{M}$, the algorithm in Fig. 10 is optimal.*

Proof of this theorem follows from Theorem 4.3 and the pseudocode of the algorithm.

3. Note that line 9 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.

Theorem 4.5. *The time complexity of the algorithm in Fig. 10 is $O(N \sum_{i=1}^{N} \beta_i^{max})$.*

Proof. In the algorithm shown in Fig. 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 4, 8, and 9). The worst case time complexity of the procedure $\mathrm{Compute\_}\vec{d}(\vec{\beta}, j)$ (line 4) is a function of the network size, i.e., the longest path in the network. Hence, for a given network, the time complexity of $\mathrm{Compute\_}\vec{d}(\vec{\beta}, j)$ can be bounded by a constant. The procedure $\mathrm{Compute\_}\vec{s}(\vec{d}, \vec{D})$ (line 8) 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 9) is $O(N)$.

Hence, the time complexity of the algorithm of Fig. 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 Fig. 11. It 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 symmetrical 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.

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 HRT connections can be admitted, given the traffic load in terms of the average utilization of the links U.

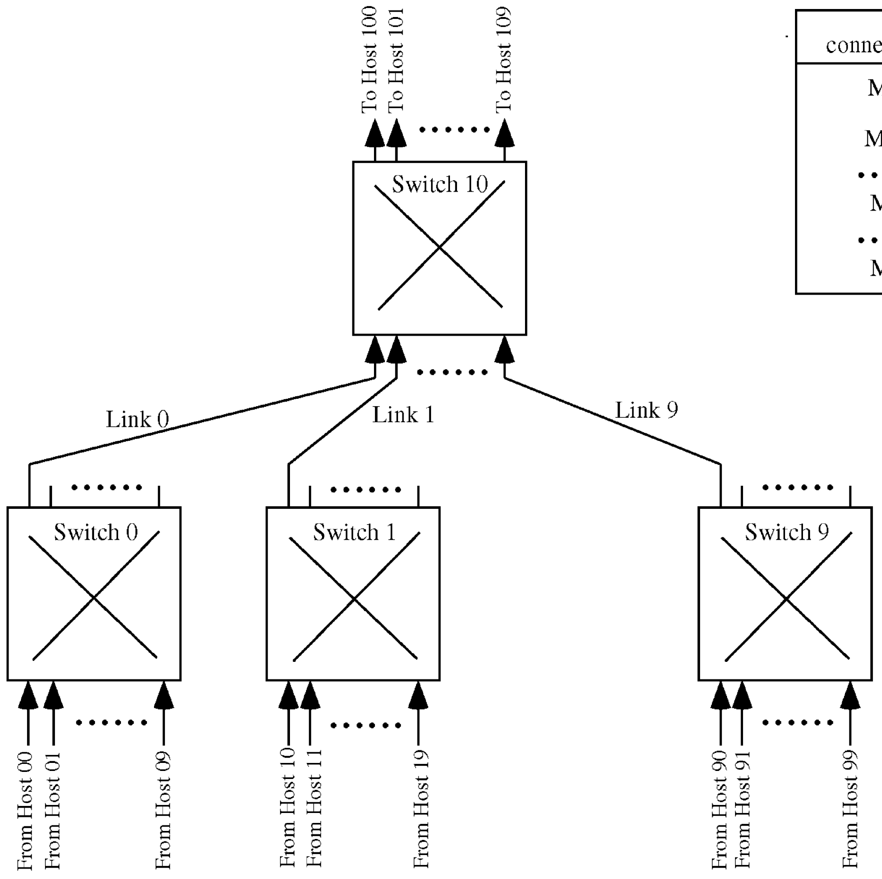
To obtain the performance data, we developed a program to simulate the above ATM network and the connections. The program is written in the C programming language and runs in a Sun/Solaris environment. In each run of the program, 200 connection sets are randomly generated. For each connection, the total number of cells per period is chosen from a geometric distribution with mean 10. The worst case cell arrival rates (C, P) of the connections sharing a particular link at the first stage are chosen as random variables. They are distributed uniformly 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.

For each connection set generated, the following 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 $< 9, 9, \ldots, 9 >$.

```
    Select_Burst_Parameters($\mathcal{M}$)
01. Compute($^{max}$);
02. $= \ < 1, 1, \ldots, 1 >$; $j = 0$
03. for $p = N$ to $m(^{max})$ do                              {Main iterative loop}
04.     $=$ Compute_($, j$);
05.     if $(\leq)$ then
06.         return();
07.     else
08.         $() =$ Compute_($,$);
09.         $j =$ Find_index_$j$();                            {Such that $s_j = 0$}
10.         if $(_j =^{max}_j)$ then
11.             return($< 0, 0, \ldots, 0 >$);
12.         else
13.             $_j = \ _j + 1$;                         {Increasing component $j$ by 1}
                   {By Theorem ??, the new burst vector is also an ancestor of $^*$}
14.         endif
15.     endif
16. endfor                                                     {End of main loop}.
```

Fig. 10. Pseudocode of the efficient algorithm.



connections	source	desitination
M_{00}	Host 00	Host 100
M_{01}	Host 01	Host 101
•••••	•••••	•••••
M_{ij}	Host ij	Host 10j
•••••	•••••	•••••
M_{99}	Host 99	Host 109

Fig. 11. Example network used in simulation.

- System C. In this system, the burst vector produced by our optimal algorithm is used.

Figs. 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. In HRT systems, it is common for deadlines to be associated with periods [14], [15].

From these figures, the following observations can be made:

- System C, where our optimal algorithm is used to set the burst vectors, performs the best. The performance gain is particularly significant when the link utilization becomes high, in comparison with systems A, B1, and B2. For example, in Fig. 12, at $U = 0.5$, $AP(U)$ is close to 1 for System C, but 0 for systems A and B2. This justifies our early claim that the burst vector must be properly set in order to

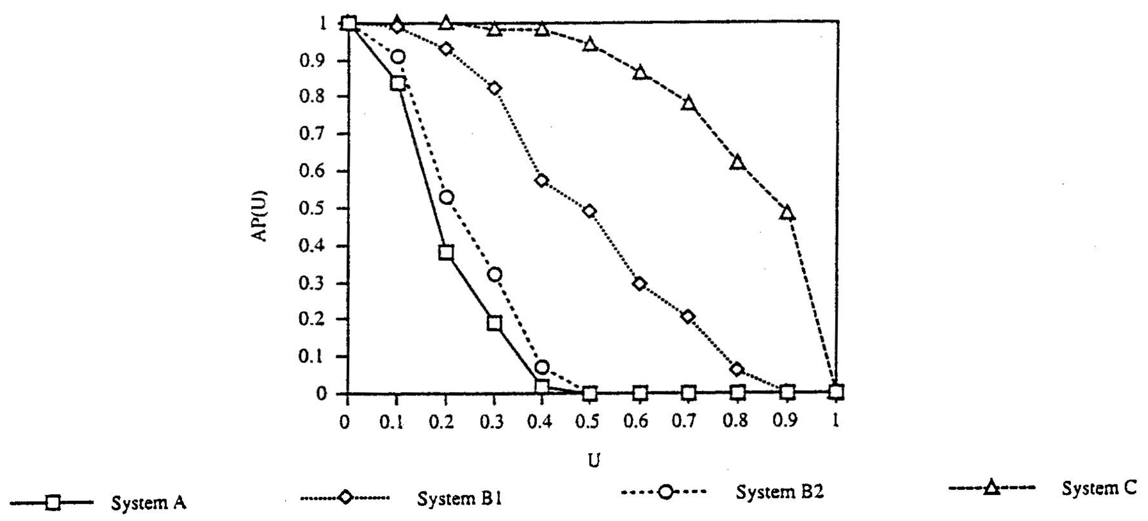


Fig. 12. Admission probability for periodic traffic ($D_i = 2P_i$).

achieve the best system performance with HRT applications.

- In general, the admission probability is sensitive to the average link utilization. As the utilization increases, the admission probability decreases. This is to be expected because the higher the network utilization, the more difficult it is for the system to admit a set of connections. We can also find the decreasing speed of the admission probability in systems A, B1, and B2 is much faster than in System C as the utilization increases. It suggests that, in the situations where the link utilization is high, the proper selection of burst vector becomes more important to the system performance.
- Comparing the performance of systems A, B1, B2, and System C, we can see clearly the difference between no-regulation, regulation with a large $\vec{\beta}$, regulation with a small $\vec{\beta}$, and regulation with an optimal $\vec{\beta}$. System A, which does not have any traffic regulation, performs the worst among all the systems. System B1 with a small $\vec{\beta}$ performs better than system B2, which has a larger $\vec{\beta}$. System C, with the optimal $\vec{\beta}$, performs the best. It is another evidence supporting the direction of our work: choosing the minimal $\vec{\beta}$ which can still meet the end-to-end deadlines. The simulation results further strengthen the need for a good $\vec{\beta}$ selection algorithm.
- The performance of all systems is very stable as D_i changes. The curves in Fig. 12 all have similar shapes as those in Fig. 13, demonstrating that the simulation results are stable. They have not been affected by any system dynamic factors, such as system loading, other system traffics, and so on.

6 Conclusions

In this paper, we have addressed the issue of guaranteeing end-to-end deadlines of HRT connections in an ATM

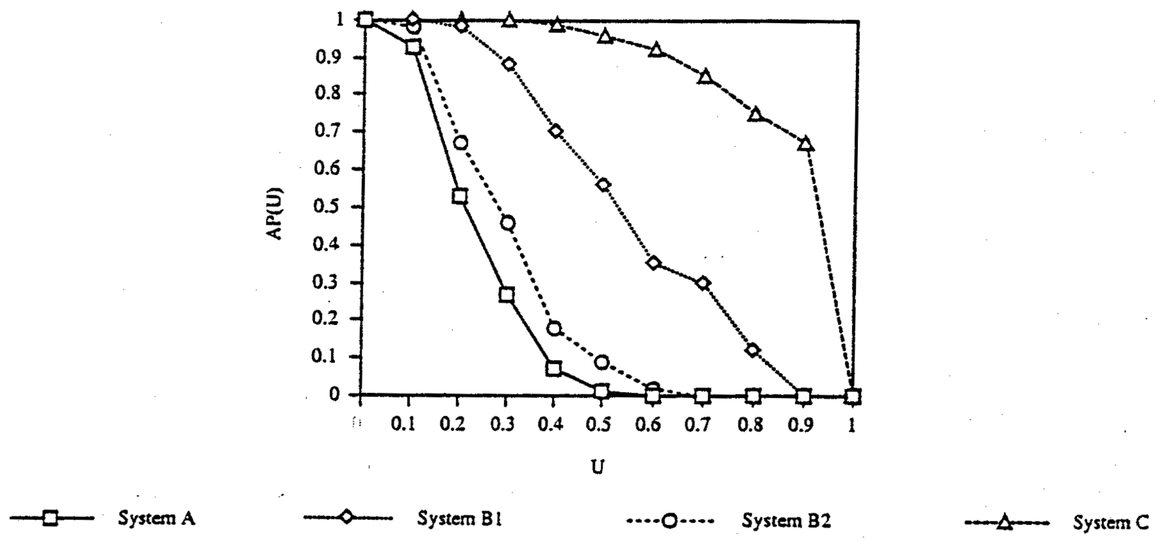


Fig. 13. Admission probability for periodic traffic ($D_i = 1.5P_i$).

network. Much of the previous work in this area has concentrated on scheduling policies used in ATM switches. Our approach to this problem is to regulate the input traffic at the network interface. In particular, we consider leaky bucket traffic regulators. This is the first study that uses traffic regulation (in particular, with leaky buckets) as a method of guaranteeing the end-to-end deadlines of HRT connections. Traditionally, a leaky bucket has been used as a policing mechanism when the source traffic at the input of the network does not conform to its negotiated characteristics.

We have designed and analyzed an efficient and 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 the deadlines of a set of HRT connections can be met, then the algorithm will always find such an assignment. Our algorithm is also efficient. We simulated and compared the performance of ATM networks with different regulation policies. We observed that 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 generic. It is independent of the switch architecture and the scheduling policy used at the ATM switches. It can be used for admission control in any ATM network that uses leaky bucket traffic regulators.

APPENDIX A
COMPUTATION OF $d_i(\vec{\beta})$

Recall that a connection is served by a leaky bucket regulator and a sequence of network servers (ATM switches). Hence, the end-to-end cell delay of a connection is the summation of the worst case delays at the leaky bucket and at all the network servers. Suppose the burst vector is $\vec{\beta}$. The end-to-end cell delay of connection M_i can be expressed as the following (given in Section 3):

$$d_i(\vec{\beta}) = d_i^{const} + d_i^{lb}(\vec{\beta}) + d_{i,1}^{fcfs}(\vec{\beta}) + d_{i,2}^{fcfs}(\vec{\beta}) + \ldots + d_{i,K}^{fcfs}(\vec{\beta}), \tag{43}$$

where $d_i^{lb}(\vec{\beta})$ and $d_{i,j}^{fcfs}(\vec{\beta})$ are the worst case queuing delays at the leaky bucket and jth FCFS server, and K is the total number of servers on connection M_i. In this section, we discuss the computation of $d_i^{lb}(\vec{\beta})$ and $d_{i,j}^{fcfs}(\vec{\beta})$. The delay analysis at a server (or a leaky bucket) involves two steps: 1) the computation of the worst case cell delay at the server; 2) the determination of the traffic descriptions at the output of the server. The output traffic of a server is the input traffic of the next server.

A.1 Delay Analysis of a Leaky Bucket

The following additional notations, assumptions and definitions are first introduced:

$\Phi_{i,lb}(I)$ denotes the number of cells of M_i that can arrive at the input of the leaky bucket in an interval of length I:

$$\Phi_{i,lb}(I) \le \left\lfloor \frac{I}{P_{i,lb}^{in\_server}} \right\rfloor C_{i,lb}^{in\_server} + \min\left(C_{i,lb}^{in\_server}, I - \left\lfloor \frac{I}{P_{i,lb}^{in\_server}} \right\rfloor P_{i,lb}^{in\_server} \right). \tag{44}$$

$Q_{i,lb}$ denotes the maximal queue length at the leaky bucket of M_i.

For each of the leaky buckets in the system we make the following assumptions:

- To ensure the stability of the system,
$$\rho_i \ge \frac{C_{i,lb}^{in\_server}}{P_{i,lb}^{in\_server}}. \tag{45}$$
- The burst parameter of the leaky bucket is at least one, i.e.,
$$\beta_i \ge 1. \tag{46}$$

Next, to facilitate the understanding of a leaky bucket regulator, we introduce the concept of *regeneration point*.

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

The following lemmas help us to prove the main theorem in this subsection.

Lemma A.1. *For connection M_i whose traffic is described by $(C_{i,lb}^{in\_server}, P_{i,lb}^{in\_server})$, $C_{i,lb}^{in\_server} < P_{i,lb}^{in\_server}$, and regulated by a leaky bucket (β_i, ρ_i), if t is a regeneration point, then there is another regeneration point in interval $(t, t + P_{i,lb}^{in\_server}]$.*

Proof. We prove the lemma by contradiction. Let us assume that, for connection M_i, regulated by a leaky bucket (β_i, ρ_i), if t is a regeneration point, then there is no other regeneration point in the interval $(t, t + P_{i,lb}^{in\_server}]$.

Since t is a regeneration point, by Definition A.1, at time t, the number of cells in the leaky bucket queue is zero and the number of tokens in the leaky bucket is β_i. Therefore, the total number of tokens available at the leaky bucket by time t', $t < t' \le t + P_{i,lb}^{in\_server}$, is

$$\beta_i + \lfloor \rho_i \cdot (t' - t) \rfloor. \tag{47}$$

Since there is no other regeneration point in interval $(t, t + P_{i,lb}^{in\_server}]$, using the definition of the leaky bucket we get

$$\beta_i + \lfloor \rho_i \cdot (t' - t) \rfloor - \Phi_{i,lb}(t' - t) < \beta_i. \tag{48}$$

By algebraic manipulation of (48), we get

$$\lfloor \rho_i \cdot (t' - t) \rfloor < \Phi_{i,lb}(t' - t). \tag{49}$$

Hence, for $t' = t + P_{i,lb}^{in\_server}$, we have

$$\lfloor \rho_i \cdot P_{i,lb}^{in\_server} \rfloor < \Phi_{i,lb}(P_{i,lb}^{in\_server}). \tag{50}$$

Using (45), substitute the minimum value of ρ_i in (50). Then, we have

$$C_{i,lb}^{in\_server} < \Phi_{i,lb}(P_{i,lb}^{in\_server}). \tag{51}$$

But, (51) contradicts (44), the definition of the periodic descriptor. Therefore, our assumption that there is no other regeneration point in the interval $(t, t + P_{i,lb}^{in\_server}]$ leads to a contradiction. Hence, the lemma holds. □

Before deriving the worst case delay at a leaky bucket, we introduce the following lemma which gives the maximal queue length at the leaky bucket.

Lemma A.2. *For a connection M_i regulated by a leaky bucket with parameters (β_i, ρ_i), the maximal queue length $Q_{i,lb}$ is given by,*

$$Q_{i,lb} = C_{i,lb}^{in\_server} - \beta_i. \tag{52}$$

Proof. We know the maximal queue length at the leaky bucket will occur between two regeneration points. Further, by Lemma A.1, the maximal interval between any two consecutive regeneration points is at most $P_{i,lb}^{in\_server}$. Let t^r be a regeneration point. $Q_{i,lb}$, the maximal queue length at the leaky bucket, occurs in the interval $[t^r, t^r + P_{i,lb}^{in\_server}]$:

$$Q_{i,lb} = \max_{t^r \le t \le t^r + P_{i,lb}^{in\_server}} (\Phi_{i,lb}(t - t^r) - \beta_i - \lfloor \rho_i \cdot (t - t^r) \rfloor). \tag{53}$$

Further, from (44), we have

$$\begin{aligned}\Phi_{i,lb}(t - t^r) \le & \left\lfloor \frac{(t - t^r)}{P_{i,lb}^{in\_server}} \right\rfloor C_{i,lb}^{in\_server} \\ & + \min\Bigg(C_{i,lb}^{in\_server}, (t - t^r) \\ & - \left\lfloor \frac{(t - t^r)}{P_{i,lb}^{in\_server}} \right\rfloor P_{i,lb}^{in\_server}\Bigg).\end{aligned} \tag{54}$$

Now, we consider (54) in two cases.

Case 1: $t - t^r < P_{i,lb}^{in\_server}$. That is, $\lfloor \frac{(t-t^r)}{P_{i,lb}^{in\_server}} \rfloor = 0$. Hence, (54) becomes

$$\Phi_{i,lb}(t - t^r) \le \min(C_{i,lb}^{in\_server}, (t - t^r)) \le C_{i,lb}^{in\_server}. \tag{55}$$

Case 2: $t - t^r = P_{i,lb}^{in\_server}$. That is $\lfloor \frac{(t-t^r)}{P_{i,lb}^{in\_server}} \rfloor = 1$. Hence, (54) becomes

$$\Phi_{i,lb}(t - t^r) \le C_{i,lb}^{in\_server} + \min(C_{i,lb}^{in\_server}, 0) = C_{i,lb}^{in\_server}. \tag{56}$$

Therefore, in either case, we have

$$\Phi_{i,lb}(t - t^r) \le C_{i,lb}^{in\_server}. \tag{57}$$

Substituting (57) in (53), we get

$$Q_{i,lb} = \max_{t^r \le t \le t^r + P_{i,lb}^{in\_server}} (C_{i,lb}^{in\_server} - \beta_i - \lfloor \rho_i \cdot (t - t^r) \rfloor). \tag{58}$$

When $t - t^r = 0$, "$C_{i,lb}^{in\_server} - \beta_i - \lfloor \rho_i \cdot (t - t^r) \rfloor$" reaches the maximum. Hence, we get

$$Q_{i,lb} = C_{i,lb}^{in\_server} - \beta_i. \tag{59}$$

□

Next, we use the results of Lemma A.2 to derive a bound on the maximal cell delay, $d_i^{lb}(\vec{\beta})$, experienced by a cell of connection M_i, at the leaky bucket regulator, (β_i, ρ_i).

Theorem A.1. *For a connection M_i, regulated by a leaky bucket parameterized by (β_i, ρ_i)*

$$d_i^{lb}(\vec{\beta}) \le \frac{Q_{i,lb}}{\rho_i}, \tag{60}$$

where $Q_{i,lb}$ is given by (52).

Proof. The proof is obvious from the definition of a leaky bucket. The maximal number of cells buffered at the leaky bucket is $Q_{i,lb}$ (i.e., the maximal queue length) and cells pass through the leaky bucket at least at the rate of ρ_i. Thus, in the worst case, it takes $\frac{Q_{i,lb}}{\rho_i}$ to let all cells pass through the leaky bucket, which is the worst case time during which a cell is buffered before it is transmitted. □

Next, we derive the description of traffic at the output of the leaky bucket.

Theorem A.2. *For a connection M_i, regulated by a leaky bucket (β_i, ρ_i), the upper bound of the output traffic, denoted by $\Gamma_{i,lb}^{out\_server}(I)$, is*

$$\Gamma_{i,lb}^{out\_server}(I) = \frac{\beta_i + \rho_i I}{I}. \tag{61}$$

Proof. The proof of this theorem follows directly from the definition of the leaky bucket. □

A.2 Delay Analysis of an FCFS Server

We now analyze an FCFS server j, $1 \le j \le K$. For any connection M_i, let $\Psi_{i,j}^{in\_server}(I)$ be the maximal number of cells from connection M_i that can arrive at the input of server j in any interval of length I. Suppose the traffic of M_i at the input of server j is described by $\Gamma_{i,j}^{in\_server}(I)$ (the input traffic to the first FCFS server is the output of the leaky bucket regulator given in (61)). Therefore, at server j we have

$$\Psi_{i,j}^{in\_server}(I) = I\Gamma_{i,j}^{in\_server}(I). \tag{62}$$

Given the input traffic description at FCFS server j, the following theorem is used to find an upper bound on the delay experienced by a cell of M_i at server j.

Theorem A.3. *For an FCFS server j, $d_{i,j}^{fcfs}(\vec{\beta})$, the worst case delay experienced by a cell of connection M_i at the server is given by*

$$d_{i,j}^{fcfs}(\vec{\beta}) = \left\lceil \max_{I \le L_j} \left(\sum_{i=1}^{N} \Psi_{i,j}^{in\_server}(I) - I \right) \right\rceil, \tag{63}$$

where L_j, the length of the longest busy interval at server j, is given by

$$L_j = \min\left(I \mid \sum_{i=1}^{N} \Psi_{i,j}^{in\_server}(I) \le I \right) \tag{64}$$

and $\Psi_{i,j}^{in\_server}(I)$ is given by (62).

```
procedure Compute_$\vec{d}$ ($\vec{\beta}, j$);
begin
    if ( $\vec{\beta} = (1, 1, ..., 1)$) then
        initializtion;
        compute delays and traffic output for $N$ connections;
        return;
    $Impact\_svr := H_j$;
    while ( $Impact\_svr \neq \phi$) do begin
        $s := get\_elem(Impact\_svr)$ ;
        $remove\_elem(Impact\_svr, s)$ ;
        $old\_FCFS\_d := FCFS\_d[s]$;
        compute_delay($s$);
        compute_output($s$);
        if $old\_FCFS\_d \neq FCFS\_d[s]$ then
            for ($i = 1$ to $N$, $i \neq j$) do
                if $s \in H_i$ and $Next_i(s) \notin Impact\_svr$ then
                    $append(Impact\_svr, Next_i(s))$;
            endforeach;
    endwhile;
end;
```

Fig. 14. Pseudocode of procedure to computer worst case delay at every variable server.

In (63), $\sum_{i=1}^{N} \Psi_{i,j}^{in\_server}(I)$ is the maximal number of cells from all the connections that can arrive at server j during an interval of length I, and I is the number of cells that server j can transmit during the interval I (recall that, in this paper, time unit is normalized by the time for the transmission of one cell). Therefore, $(\sum_{i=1}^{N} \Psi_{i,j}^{in\_server}(I) - I)$ is the maximal queue length at the input of server j. Maximizing $(\sum_{i=1}^{N} \Psi_{i,j}^{in\_server}(I) - I)$ over all possible values of I, we have the worst case delay of a cell. The formal proof of Theorem A.3 can be obtained by applying Theorem 4.1 given in [3].

The next theorem gives the output traffic description of the connection M_i at FCFS server j.

Theorem A.4. *For connection M_i, if the input traffic at FCFS server j is described by $\Gamma_{i,j}^{in\_server}(I)$, then the output traffic parameter $\Gamma_{i,j}^{out\_server}(I)$ is given by*

$$\Gamma_{i,j}^{out\_server}(I) = \min\left(1, \left(1 + \frac{d_{i,j}^{fcfs}\left(\vec{\beta}\right)}{I} \Gamma_{i,j}^{in\_server}\left(I + d_{i,j}^{fcfs}\left(\vec{\beta}\right)\right)\right)\right). \tag{65}$$

Theorem A.4 follows from Theorem 4.2 in [19], which is also implied by Theorem 2.1 in [3].

A.3 Procedure for Computing $d_i(\vec{\beta})$

In this subsection, we outline the procedure to compute the worst case end-to-end cell delays for every hard real-time connection.

Recall that we model an HRT connection as a sequence of variable servers. In such a model, the worst case end-to-end delay experienced by a cell of a connection is computed by first determining the worst case cell delay at each server and then summing up the delays at all servers. In the previous subsections, we presented the expressions for the worst case cell delay at a leaky bucket and an FCFS server.

The computation of delays at a server requires the input traffic of all the connections at the server to be known. It is quite possible that a traffic dependency loop exists in the connection-server graph, i.e., the paths of some connections form a loop. When such a loop exists, none of the servers on the loop has its input traffic completely known. In [19], an on-line admission control algorithm was proposed, which admits a new connection into the system under the condition that the delay requirements of all connections are still met. This on-line algorithm specially considered the situations where such dependency loops exist. We will use this on-line method to compute $\vec{d}(\vec{\beta})$ in the presence of traffic dependency loops.

In our efficient algorithm of searching for the $\vec{\beta}^*$ presented in Section 4.4, each time only one element of $\vec{\beta}$ is changed (incremented by 1). For each change of $\vec{\beta}$, the delays at all the servers which are affected by this change need to be recomputed. In order to compute the delays at all affected servers, the following variables are introduced:

H_i is the sequence of servers on the connection path of M_i.

$Impact\_svr$ is an ordered list of servers which are affected by the change of a $\vec{\beta}$. The initial value of $Impact\_svr$ for the change of β_i is H_i.

$Next_i(s)$ is a function which returns the server next to server s on the connection path of M_i.

$FCFS\_d$ is a vector of delays, one component for each FCFS server in the system. $FCFS\_d[s]$ is the delay at FCFS server s.

$In\_traffic$ is an array used to record the input traffic of connections at all FCFS servers. $In\_traffic[i, j]$ is the input traffic of connection M_i to server j.

Fig. 14 is the procedure for computing the worst case delays at all the servers. The procedure first computes the delays and traffic outputs of connections at all servers for the initial value of $\vec{\beta} = (1, 1, \ldots, 1)$. The on-line admission control algorithm, presented in [19], is used to add the N connections into the system one after another. Every time a new connection is added into the system, the input traffic of the new connection and the delays and the input traffic of existing connections at all FCFS servers are known. The delays and input traffic at all servers after adding the new connection can thus be computed. The order of adding the N connections into the system does not affect the final results because the initial values of β for all connections are the same.

After the initial values of delays and input traffic at all servers have been determined, each time when a β value is changed, its impact to the existing system can be computed by starting from the server where the β value is changed. Suppose β_j is changed. The delay analysis starts from the first server of M_j. Since the delay and the input traffic at each server before the change of β_j has been already computed and recorded in the system, the delay analysis at the servers of M_j can be done one after another along M_j's connection path. Procedure compute_delay(s) (Line 12) computes the worst case cell delays at server s by using the results in (60) and (63). Procedure compute_output(s) (Line 13) computes the output traffic to the succeeding server by using the results in (61) and (65).

Since the change of β_j will also change the delays and input traffic of other connections, the algorithm appends the servers which are affected to list $Impact\_svr$ (Line 14-17). Thus, the algorithm systematically traces all the servers affected by the change of β_j. List $Impact\_svr$ will eventually become empty and, therefore, the algorithm terminates, because the impact caused by the change of β_j converges. The formal proof can be found in [17].

Acknowledgments

This work was supported in part by Strategic Research Grant no. 7000677 from City University of Hong Kong.

References

[1] A.W. Berger, "Performance Analysis of a Rate-Control Throttle where Tokens and Jobs Queue," *IEEE J. Selected Areas in Comm.*, vol. 9, no. 2, pp. 165-170, Feb. 1991.

[2] D.D. Clark, S. Shenker, and L. Zhang, "Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism," *Proc. ACM SIGCOMM '92*, pp. 14-26, Aug. 1992.

[3] R.L. Cruz, "A Calculus for Network Delay," *IEEE Trans. Information Theory*, vol. 37, no. 1, pp. 114-141, Jan. 1991.

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

[5] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," *Proc. ACM SIGCOMM '89*, pp. 1-12, Sept. 1989.

[6] D. Ferrari and D.C. Verma, "A Scheme for Real-Time Channel Establishment in Wide-Area Networks," *IEEE J. Selected Areas in Comm.*, vol. 8, no. 3, pp. 368-379, Apr. 1990.

[7] M. Garrett, "Contributions Toward Real-Time Services on Packet Switched Networks," PhD thesis, Columbia Univ., 1993.

[8] S.J. Golestani, "A Framing Strategy for Congestion Management," *IEEE J. Selected Areas in Comm.*, vol. 9, no. 7, pp. 1,064-1,077, Sept. 1991.

[9] Int'l Telecomm. Union, *ITU-T Recommendation I.311—B-ISDN General Network Aspects,* 1993.

[10] C.R. Kalmenek, H. Kanakia, and S. Keshav, "Rate Controlled Servers for Very High-Speed Networks," *Proc. IEEE Global Telecomm. Conf.*, pp. 300.3.1-300.3.9, Dec. 1990.

[11] D.D. Kandlur, K.G. Shin, and D. Ferrari, "Real-Time Communication in Multi-Hop Networks," *Proc. 11th Int'l Conf. Distributed Computing Systems,* pp. 300-307, May 1991.

[12] S. Kweon and K.G. Shin, "Traffic-Controlled Rate-Monotonic Priority Scheduling of ATM Cells," *Proc. IEEE Infocom '96,* pp. 655-662, Mar. 1996.

[13] T. Ling and N. Shroff, "Scheduling Real-Time Traffic in ATM Networks," *Proc. IEEE Infocom '96,* pp. 198-205, Mar. 1996.

[14] C.L. Liu and J.W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," *J. ACM,* vol. 20, no. 1, pp. 46-61, Jan. 1973.

[15] N. Malcolm and W. Zhao, "Hard Real-Time Communication in Multiple-Access Networks," *J. Real-Time Systems,* Jan. 1995.

[16] A.K.J. Parekh, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks," PhD thesis, Dept. of Electrical Eng. and Computer Science, Massachusetts Inst. of Technology, 1992.

[17] A. Raha, "Real Time Communication in ATM Networks," PhD thesis, Dept. of Computer Science, Texas A&M Univ., 1996.

[18] A. Raha, S. Kamat, and W. Zhao, "Guaranteeing End-to-End Deadlines in ATM Networks," *Proc. 15th IEEE Int'l Conf. Distributed Computing Systems,* June 1995.

[19] A. Raha, S. Kamat, and W. Zhao, "Admission Control for Hard Real-Time Connections in ATM LAN's," *Proc. IEEE Infocom '96,* Mar. 1996.

[20] S.S. Sathaye, W.S. Kish, and J.K. Strosnider, "Responsive Aperiodic Services in High-Speed Networks," *Proc. IEEE Int'l Conf. Distributed Computing Systems,* pp. 337-346, May 1993.

[21] L. Trajkovic and S.J. Golestani, "Congestion Control for Multimedia Services," *IEEE Network,* vol. 6, no. 5, pp. 20-26, Sept. 1992.

[22] R.J. Vetter, "ATM Concepts, Architectures, and Protocols," *Comm. ACM,* vol. 38, no. 2, Feb. 1995.

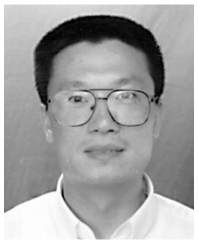
[23] H. Zhang and D. Ferrari, "Rate-Controlled Static Priority Queueing," *Proc. IEEE Infocom '93,* pp. 227-236, Mar. 1993.

[24] H. Zhang and S. Keshav, "Comparison of Rate-Based Service Disciplines," *Proc. ACM SIGCOMM '91,* pp. 113-121, Sept. 1991.

[25] L. Zhang, "Virtual Clock: A New Traffic Control Algorithm for Packet Switching Networks," *Proc. ACM SIGCOMM '90,* pp. 19-29, Sept. 1990.

Amitava Raha received his PhD in computer science from Texas A&M University, College Station, Texas, in 1996. He is currently the senior designer at Fujitsu, where he is involved in the research and development of a management platform to support end-to-end (applications, systems, and network) quality of service management. In addition, he is also involved in the design and development of QoS-based transport for supporting multicasting applications in distance learning. His interests include QoS provisioning in areas of networks and e-commerce, network protocols, middleware, and design patterns.

Sanjay Kamat received the PhD degree in computer science from Texas A&M University, College Station, Texas, in 1995. He received the BTech degree in mechanical engineering in 1985 and the MTech degree in computer science in 1987 from the Indian Institute of Technology, Bombay. Dr. Kamat is currently a member of the technical staff of the High Speed Networks Research Group at Bell Laboratories, Lucent Technologies. He was previously a research staff member at the IBM T.J. Watson Research Center. His research interests include high speed networks, constraint-based routing, policy-based networking, real-time communication, fault tolerance, and distributed systems.



Xiaohua Jia received his ScD in information science from the University of Tokyo in 1991. He is currently an associate professor in the Department of Computer Science at the City University of Hong Kong. Dr. Jia's research interests include operating systems, distributed systems, network systems, computer communications, and real-time communications. He is a member of the IEEE Computer Society.

Wei Zhao received his BSc in physics from Shaanxi Normal University, Xian, China, and his MSc and phD in computer and information science from the University of Massachusetts at Amherst in 1983 and 1986, respectively. In 1990, he joined the Department of Computer Science at Texas A&M University, College Station, Texas, where he is currently a full professor and department head. His current research interests include secured real-time computing and communication, distributed operating systems, database systems, and fault-tolerant systems. Dr. Zhao was a member of the editorial board of the *IEEE Transactions on Computers*. He served as guest editor of a special issue on real-time operating systems for the *ACM Operating Systems Review* and for a speical issue on real-time middleware for the *International Journal of Real-Time Systems*. He was the general and program chair of the IEEE Real-Time Technology and Applications Symposium in 1995 and 1996, respectively. he is the general and program chair of the IEEE Real-Time Systems Symposium in 1999 and 2000, respectively. He will also serve as the general chair of the IEEE International Conference on Distributed Computing Systems in 2001. He is an editor for the *International Journal of Real-Time Systems* and a member of the executive committee of the IEEE Technical Committee on Real-Time Systems. Dr. Zhao and his students received the Outstanding Paper Award from the IEEE International Conference on Distributed Computing Systems and the Best Student Paper Award from the IEEE National Aerospace and Electronics Conference. He has published more than 100 papers in journal, conferences, and book chapters. He is a member of the IEEE.