Providing Deterministic Delay Guarantees in ATM Networks

Seok-Kyu Kweon and Kang G. Shin, *Fellow, IEEE*

Abstract—**Broadband Integrated Services Digital Network (B-ISDN) is expected to carry various types of traffic, including best-effort as well as real-time traffic like video-on-demand, live multimedia conferences, and remote medical services for which stringent quality of service (QoS) requirements must be met. In particular, per-packet (per-cell) delay guarantees are an important QoS requirement for real-time applications. Although several methods have been proposed to provide delay guarantees in packet-switching networks, they are either too complex for ATM networks which allow only very simple operations to achieve high bandwidths (hence, high-speed switching and routing), or too inefficient (in using network resources) to be cost effective. In this paper, we propose a cell-multiplexing scheme for the real-time communication service in ATM networks. The proposed scheme achieves an efficiency close to that of Packet-by-Packet Generalized Processor Sharing, and incurs an implementation cost similar to that of Rate Controlled Static Priority Queueing. It employs the leaky bucket model as the input traffic description model and regulates the input traffic at the User Network Interface to follow the input traffic specification. Inside the network, it consists of two components, *traffic controller* and *scheduler*. The function of the traffic controller is to shape real-time traffic to have the same input pattern at every switch along the path. The end-to-end delay is bounded by the scheduler which employs a nonpreemptive rate-monotonic priority scheduling policy at each switch on the path. The proposed scheme is compared to three other popular schemes with MPEG-coded movie clips. Finally, we present a hardware implementation of the proposed scheme based on a systolic array priority queue.**

Index Terms—**ATM, broadband ISDN, cell multiplexing, channel admissibility, delay guarantee, multimedia conferencing, rate-monotonic priority scheduling, real-time communication, traffic controller.**

I. Introduction

DUE MAINLY TO their potential for efficiency and flexibility, asynchronous transfer mode (ATM) networks have drawn significant attention as a main technology for implementing broadband Integrated Services Digital Network (B-ISDN). In order to realize the potential of B-ISDN, ATM networks must support a wide variety of traffic and meet diverse service and performance requirements. Among the various B-ISDN services, providing performance (delay and/or delay jitter bound) guarantees is essential to such real-time applications as video and audio conferencing and video-on-demand. Unlike traditional data communication applications, real-time applications must meet stringent performance requirements in terms of delay, delay jitter, throughput, and cell-loss rate.

Among these performance requirements, delay guarantees are particularly important to real-time applications, as the cell-error rate can be kept very low with the use of advanced networking and coding technologies available today. In particular, real-time communication requires the delay of *each* cell, not the average delay, as the quality of service (QoS) requirement. If a cell arrives at the destination after its deadline has expired, its value to the application may be greatly reduced or even worthless. In some cases, a cell missing its deadline is considered lost. Thus, the cell-delivery delay must be bounded and predictable for real-time applications.

The best-effort delivery service cannot satisfy the diverse QoS requirements for different real-time applications, because the associated packet multiplexers do not differentiate between real-time and nonreal-time traffic, nor among real-time messages themselves. In order to provide per-connection end-to-end delay guarantees in packet-switching (and ATM) networks, we need a special packet/cell scheduling scheme. When packets of different connections compete for an outgoing link, the packet-scheduling scheme orders these packets for transmission so that all real-time connections are guaranteed to meet their packet deadlines over the link. In order to meet packet-delay requirements, the packet-scheduling scheme must be supplemented by an appropriate connection admission control (CAC), usually achieved by resource reservation and traffic policing schemes. These schemes are governed by a connection's input traffic specification, i.e., packet-arrival behavior at its source node. Specifying/modeling a real-time application's traffic pattern is a challenging problem, especially when source traffic has variable bit rate (VBR) characteristics, e.g., MPEG-coded video. The leaky bucket model [1], [2] and the $(X_{\min}, X_{\text{ave}}, I, S_{\max})$ model [3], [4] are prototypical example input traffic specifications. In the leaky bucket model, the amount of connection i's traffic generated during time interval $[t, \tau]$ is assumed to be upper bounded by $\sigma_i + \rho_i(\tau - t)$. That is, the size of an instantaneous traffic burst is limited by σ_i and the long-term average traffic-arrival rate is upper bounded by ρ_i. In the $(X_{\min}, X_{\text{ave}}, I, S_{\max})$ model, $X_{\min}$ is the minimum packet inter-arrival time, X_{ave}

Manuscript received July 24, 1996; revised January 5, 1998; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor L. Zhang. This work was supported in part by the National Science Foundation under Grant MIP-9203895 and in part by the Office of Naval Research under Grant N00014-94-1-0229. An earlier version of this paper was presented at INFOCOM '96, San Francisco, CA, March 24–28, 1996.

The authors are with the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109–2122 USA (e-mail: skkweon@eecs.umich.edu; kgshin@eecs.umich.edu).

Publisher Item Identifier S 1063-6692(98)09488-6.

is the average packet inter-arrival time over a time interval of length I, and $S_{\max}$ is the maximum packet size. This model tries to capture the burst duration, as well as the long-term average arrival rate and the short-term peak rate. Although how to schedule a packet and how to characterize source traffic are, in general, orthogonal issues, choosing a good input traffic characterization affects greatly the performance of a packet-scheduling scheme in terms of network utilization and achievable delay bound. Thus, in analyzing the performance of a packet-scheduling scheme, one must also consider the underlying input traffic characterization. A packet-scheduling scheme combined with an input traffic specification is called a *service discipline* [5].

In order to provide deterministic real-time communication services in large-scale high-speed ATM networks, a service discipline must usually be implemented in hardware for fast operation speed. Let us consider the following example to get a feel for operation speed. In a 2.5-Gb/s ATM network, a cell can be transmitted once every 0.17 μs. In the worst case, the link scheduler must determine which cell to transmit first within the next 0.17 μs, while accepting new cells from all incoming links within the same 0.17 μs period. If the link scheduler cannot keep up with this link speed, it should not be used for real-time communication in high-speed networks (as it will keep the link idle and, hence, underutilize the link bandwidth). Scalability is another important factor since switches in backbone networks can easily have hundreds of thousands of concurrent real-time connections/channels with different delay requirements. So, the hardware implementation of a service discipline must be scalable with respect to the number and the type of real-time channels.

Recently, there have been several service disciplines proposed for per-connection end-to-end delay guarantees in packet-switching networks [5]. Virtual Clock is a rate-based traffic control algorithm that can be used in packet-switching networks. Although it can provide delay-guaranteed service with an appropriate connection admission control, it tends to penalize a connection that has received better than guaranteed service during a certain time interval [6]. Framing strategies like Hierarchical Round Robin and Stop-and-Go provide bounded end-to-end delays. However, as pointed out in [5], they suffer the problem of coupling between the guaranteeable delay bound and the bandwidth-allocation granularity. Delay-EDD (Earliest-Due-Date) [3] combined with the $(X_{\min}, X_{\text{ave}}, I, S_{\max})$ model can provide bounded end-to-end delays [3] and is optimal in terms of link utilization, since it adopts deadline scheduling [7]. However, calculating packet deadlines and sorting the packets based on their deadlines are very expensive in time and hardware. Rate Controlled Static Priority Queueing (RCSP) [4] and Real-Time Channel (RTC) [8] are variants of Delay-EDD, but do not require the calculation of packet deadlines inside the scheduler. While RTC employs the leaky bucket model as its input traffic specification and allows an arbitrary number of priority levels for arbitrary link-delay guarantees, RCSP uses the $(X_{\min}, X_{\text{ave}}, I, S_{\max})$ model and allows a finite number of static priority levels for the simplicity of implementation. Since they both are rate-controlled service disciplines [4] and, more generally, nonwork-conserving service disciplines, they reduce the buffer requirement at the intermediate nodes. In nonwork-conserving service disciplines, the traffic-arrival pattern of a connection is reconstructed at every intermediate node, so that the traffic-arrival patterns at the intermediate nodes conform to the original input traffic specification at the source. Nonwork-conserving service disciplines achieve this goal by using a traffic regulator that holds early packets until their logical arrival times or eligibility times. Although they yield larger average packet delays than their counterparts, nonwork-conserving service disciplines can provide the same delay bounds as work-conserving service disciplines. Weighted Fair Queueing (WFQ) [9] is a rate-based packet scheduling scheme and guarantees a minimum throughput to each connection over a period longer than its packet inter-arrival time. However, WFQ itself cannot provide bounded end-to-end delays. Parekh and Gallager proved that Packet-by-Packet Generalized Processor Sharing (PGPS), which happens to be the same as WFQ, can provide bounded end-to-end delays using the leaky bucket model for input traffic specification [6], [10]. WFQ (PGPS) shares the same advantages and disadvantages as Delay-EDD because it also adopts a deadline scheduling.[1] In addition, like Delay-EDD, PGPS is a work-conserving service discipline and, thus, it requires larger buffer space than RCSP or RTC.

Although Delay-EDD, RCSP, RTC, and PGPS can provide bounded end-to-end delays in ATM networks, they differ in input traffic specification, connection admissibility, and implementation complexity. Among them, PGPS is the most efficient in connection admissibility because of its optimal deadline scheduling and the efficiency of its input traffic specification. However, it may not be a good candidate for realizing real-time communication in ATM networks because of its implementation complexity due to: 1) the high overhead associated with virtual finish time calculation and deadline-based sorting and 2) its large buffer space requirement. Since the number of real-time connections supported by a service discipline can be very large, the scalability of a service discipline is very important, especially when it is implemented in hardware. The large buffer space requirement of PGPS significantly degrades its scalability. In Section IV, the implementation issue of PGPS will be examined.

In this paper, we propose a new service discipline called the *Traffic-Controlled Rate-Monotonic Priority Scheduling* (TCRM) that provides user-requested delay guarantees in ATM networks. As a rate-controlled service discipline, the TCRM has a simple structure similar to that of RCSP, but it tries to simulate the behavior of PGPS, thus achieving connection admissibility close to that of PGPS. This scheme requires traffic regulation at the User Network Interface (UNI) and scheduling at each link along the path. We divide the multiplexer of each link into two components, traffic controller and scheduler. Using this mechanism, TCRM: 1) has efficiency close to PGPS in terms of connection admissibility;

[1] Although WFQ (PGPS) is not deadline scheduling but rate-based in essence, it requires the sorting of packets based on their virtual finish times which can be viewed as their deadlines. Hence, we consider WFQ as deadline scheduling.

2) is simple enough to operate in a high-speed switching environment like ATM networks; and 3) requires only a very small buffer space for each real-time connection.

Section II describes the mechanism of the proposed scheme and presents an admission-test algorithm for this scheme. Section III compares the proposed scheme with other schemes using MPEG-coded movie clips. In Section IV, we propose a simple implementation of the proposed scheme and discuss its implementation complexity. The paper concludes with Section V.

II. The Proposed Scheme

A *real-time channel* is defined as a virtual circuit through which a real-time communication service is provided. Before requesting the setup of a new real-time channel, the user must determine the parameters of its input traffic specification and present them to the network service provider along with its QoS requirements. Based on the user's input traffic specification and QoS requirements, the network service provider selects an appropriate path—that is, QoS routing—which traverses multiple nodes and links, and reserves the network resources needed to satisfy the user-specified QoS requirement. At run-time, the service provider guarantees the QoS using the reserved resources.

Before discussing the proposed cell-multiplexing scheme, we must first consider the method to characterize source traffic. One of the most important requirements of a good input traffic specification is that it should be "enforceable" with a simple traffic-policing scheme. If a complex input traffic specification is employed, a traffic-policing scheme that works for each connection at the network entrance will be expensive and, hence, it cannot be used for large-scale high-speed networks. Although many complex input traffic specifications like multiple leaky bucket regulator model or even empirical traffic envelopes were employed for providing end-to-end delay guarantees [11]–[13], the usefulness of these models in a real world is difficult to prove. In our approach, we employ a leaky bucket model as the input traffic specification as in PGPS. One can imagine the leaky bucket model, simply called the (σ_i, ρ_i) model here, as placing a smoothing buffer at the network entrance, the size and average drain rate of which are σ_i and ρ_i, respectively, so that the burstiness of input traffic inside the network is limited, thus lowering the network resource-reservation requirement. In this case, the smoothing buffer works as a traffic policer at the network entry.

The parameter pair of the leaky bucket model for a variable bit rate (VBR) source (σ_i, ρ_i) is obtained from the empirical traffic envelope of the source [13]. The empirical traffic envelope $E^*(t)$ is defined as the maximum amount of traffic that has arrived during any time interval of length t, i.e.,

$$E^*(t) = \max_{\tau \geq 0} A[\tau, \tau + t] \quad \forall\, t \geq 0$$

where $A[\tau, \tau+t]$ is the amount of traffic that has arrived during a time interval $[\tau, \tau+t]$. Then, (σ_i, ρ_i) is given as a parameter pair satisfying the following relation:

$$\sigma_i + \rho_i t \geq E^*(t) \quad \forall\, t \geq 0.$$

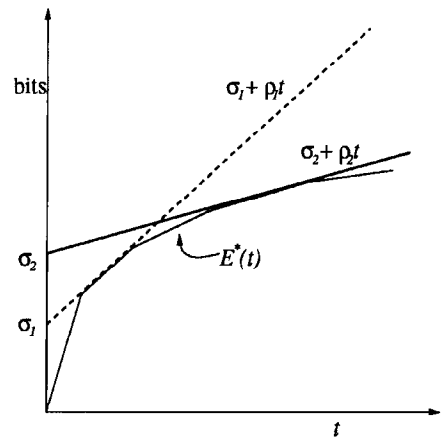


Fig. 1. Two choices for (σ_i, ρ_i) parameter pair.

There usually exist multiple pairs satisfying the above relation. For example, in Fig. 1, two such parameter pairs are shown for a VBR traffic, (σ_1, ρ_1) and (σ_2, ρ_2). They both describe the source traffic, while having different burstiness characteristics and throughput requirements. By choosing different parameter pairs, one can provide different delay bounds to a real-time channel under the same cell-service discipline. For instance, for PGPS, (σ_1, ρ_1) model will provide a smaller delay bound than (σ_2, ρ_2) model at the expense of higher resource requirements. So, we assume that input traffic specification parameters can be chosen depending on the performance requirement of an application.

Our scheme requires cooperation between the UNI and each ATM switch along the path in order to provide real-time communication service. The UNI regulates each channel i's traffic so as to bound the cell-arrival rate at the network entrance by ρ_i. The network service provider ensures that the requested channel i gets its (minimum) service rate ρ_i at every switch along the path through an appropriate admission control and run-time processing.

A. Traffic Shaping at UNI

Given the traffic model parameter pair (σ_i, ρ_i), the user requests a cell-transmission rate ρ_i from the network. After establishing the channel based on an appropriate admission test, the user begins to transmit its traffic according to the (σ_i, ρ_i) model. At the network entrance, the UNI regulates the incoming traffic in such a way that the maximum cell-transmission rate into the network is smaller than ρ_i, or the minimum cell inter-transmission time is larger than L/ρ_i, where L denotes the length of one cell (53 bytes). That is, when the kth cell of channel i arrived at the UNI at time A_k, its transmission time X_k is calculated as

$$X_k = \begin{cases} A_1, & k = 1 \\ \max(X_{k-1} + \dfrac{L}{\rho_i}, A_k), & k \geq 2. \end{cases} \tag{1}$$

Until X_k, the UNI holds the cell in its buffer. Since one cell is permitted to be transmitted every time interval of length L/ρ_i, the minimum and maximum guaranteed service rates of the queue are ρ_i over an interval of length L/ρ_i. Since the

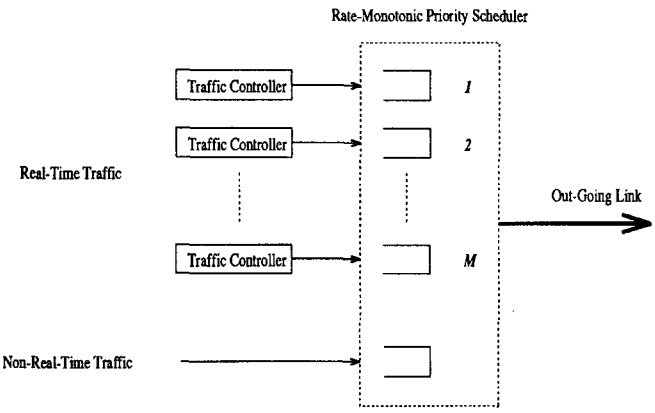


Fig. 2. Structure of TCRM.

traffic can arrive in a burst, the maximum size of which is σ_i, the UNI must have a buffer space of σ_i bits to avoid cell loss.

Unlike other approaches like PGPS, RCSP, and RTC, our scheme does not allow the burst of cells belonging to channel i to be instantaneously transmitted into the network. Such a strict nonwork-conserving service discipline at the network entrance, as we shall see later, does not change the end-to-end delay bound, while significantly reducing the buffer requirement inside the network.

B. Traffic Regulation and Scheduling Inside the Network

We model an ATM switch as an output-buffered multiple-input–multiple-output switch [14]. In this model, no cells are lost due to contention within the switch fabric, and contention exists only among those cells sharing the same outgoing link. Assuming that the switching delay is negligible as compared to the queueing delay at an output buffer, we concentrate on controlling the queueing delay at the output buffer in order to achieve a bounded end-to-end delay.

As a rate-controlled service discipline, TCRM consists of *traffic controllers* and a *scheduler* (see Fig. 2). A traffic controller is assigned to each individual real-time channel and the scheduler is shared by all the real-time channels.

1) Traffic Controller: The traffic controller executes the same traffic regulation function of the UNI. That is, it keeps the cell-arrival rate at the scheduler below ρ_i. It holds the incoming cells until their supposed arrival times, and then transfers them into the scheduler. This supposed arrival time is called the *logical arrival time* in [8] and [4]. The logical arrival time of an incoming cell is calculated based on that of the previous cell of the same channel. Thus, the logical arrival time of the kth cell at the nth node, $X_{k,n}$, is calculated as

$$X_{k,n} = \begin{cases} A_{1,n} & k = 1 \\ \max\left(X_{k-1,n} + \dfrac{L}{\rho_i}, A_{k,n}\right) & k \geq 2 \end{cases} \quad (2)$$

where $A_{k,n}$ is the actual arrival time of the kth cell at node n. Note that the inter-logical arrival time of incoming cells is at least L/ρ_i. Assuming that the cell-arrival rate at the traffic controller is under ρ_i, we can ensure that at most one cell for channel i can exist in the traffic controller of channel i, since the traffic controller is permitted to transfer a cell every L/ρ_i s. Hence, the traffic controller requires buffer space for storing only one cell.

2) Nonpreemptive Rate-Monotonic Priority Scheduling: After the cell stream of real-time channel i passes through the traffic controller, the cell-arrival rate at the scheduler of every switch is bounded by ρ_i. If we can provide the minimum cell drain rate ρ_i at the scheduler, the unbounded accumulation of cells at the scheduler will never happen. In that case, the minimum cell inter-arrival time and the cell delay bound at the scheduler will be given as L/ρ_i. In order to provide the minimum cell-drain rate ρ_i, we employ the well-known rate-monotonic priority scheduling as the scheduling policy of TCRM. Liu and Layland [7] proved that the rate-monotonic priority scheduling is optimal among all fixed-priority scheduling policies when the deadline of each task is the same as the task period. If we treat a cell as a "task," then the rate-monotonic priority cell scheduling is optimal among fixed-priority scheduling policies[2] that achieve the guaranteed throughput, because the cell inter-arrival period is the same as the cell-delivery deadline ($= L/\rho_i$). This scheduling policy assigns higher priority to channels with higher request rates, i.e., higher ρ_i.

Liu and Layland's analysis [7] is based on a preemptive scheduling policy. However, preemptive scheduling is not desirable for cell transmissions, since, if in-progress transmission of a cell is interrupted, the cell will be lost and has to be retransmitted, thus wasting network resources. Thus, we have to use the nonpreemptive rate-monotonic priority scheduling policy as the cell scheduling policy.[3] The nonpreemptive rate monotonic priority scheduler assigns a priority level to each real-time channel according to its required throughput and in-progress cell transmission will not be preempted. As a result, the scheduler provides the minimum throughput ρ_i to each channel i.

Since the cell inter-arrival time is larger than, or equal to L/ρ_i and one cell is permitted to be transmitted every L/ρ_i, at most one cell of channel i can stay in the scheduler at any time. Hence, the scheduler needs a buffer of one cell for each real-time channel.

Let us look at how the rate-monotonic priority scheduler works. In Fig. 2, the scheduler transmits cells according to ρ_i values. If there are no cells belonging to real-time channels, cells from nonreal-time traffic queue are transmitted. In any case, the cells held at the traffic controllers are not allowed to be transmitted.

C. Admission Control

In order to provide throughput guarantees at the nonpreemptive rate-monotonic priority scheduler, we need an appropriate admission control for real-time channels. The admission-control test involves every node along the path of

[2] Although fixed-priority scheduling policies are less efficient than deadline-scheduling policies in terms of network utilization [7], we prefer the implementation simplicity of the rate-monotonic priority scheduling.

[3] Employing a nonpreemptive policy does not affect the optimality of the rate-monotonic priority scheduling, since the nonpreemptive rate-monotonic priority scheduling policy is optimal among nonpreemptive fixed-priority policies, which can be proved using the same arguments in the proof of [7, Th. 2].

the real-time channel. If any node along the path fails this test, the channel request must be denied.

At the scheduler, the throughput guarantee is made not for bit by bit but for cell by cell. That is, when each cell arrives at the scheduler with the minimum cell inter-arrival time L/ρ_i which is guaranteed by the traffic controller, the scheduler must complete the transmission of the cell before the next cell's earliest arrival time, which is the current cell's arrival time plus L/ρ_i. We need a schedulability test to verify whether or not the worst case delivery time is smaller than, or equal to, the local delay bound L/ρ_i. Using a method similar to Kandlur's [8], we derive the schedulability test for the proposed scheme. Consider a set of real-time channels $\{i, i = 1, \cdots, M\}$ which share a common link ℓ, where M is the number of existing real-time channels on link ℓ. Denote the throughput of channel i by ρ_i, and assume that channels are indexed in the descending order of priority so that $\rho_i \geq \rho_j$ if $i < j$. Then, the schedulability test is given as

$$\sum_{j=1}^{i-1} C\left\lceil \frac{L/\rho_i}{L/\rho_j} \right\rceil + 2C \leq \frac{L}{\rho_i}, \qquad \text{for } i = 1, \cdots, M \quad (3)$$

where C is one cell transmission time, L/ρ_i is the link delay bound of channel i's cell, and all channels $j, 1 \leq j < i$, have higher priority than channel i. Note that the first term of (3) denotes the sum of all the transmission times of cells belonging to the channels of higher priority than channel i in the worst case.[4] C of the second term denotes the time to complete in-progress cell transmission. The other C denotes the transmission time of a cell belonging to channel i. Conceptually, the schedulability condition implies that the transmission of a cell of channel i must be finished within its link delay bound, even in the worst case. If the schedulability condition fails, the cells of channel i cannot be transmitted in the worst case. Therefore, the schedulability condition is also the necessary condition.

Using this argument, we can show that TCRM emulates circuit switching in the cell level. Let us define $T_i(t, s)$ as channel i's traffic transmitted over a link during a time interval $[t, s)$ for any t and s such that $t \leq s$. Then,

$$L\left\lfloor \frac{s-t}{L/\rho_i} \right\rfloor \leq T_i(t, s) \leq L\left\lceil \frac{s-t}{L/\rho_i} \right\rceil. \quad (4)$$

The lower bound is derived from the fact that a local delay bound is guaranteed by the scheduler and the upper bound comes from traffic regulation by the traffic controller. Therefore, the average traffic service rate of channel $i, R_i(t, s)$, during the interval $[t, s)$ is given as

$$R_i(t, s) = \rho_i \quad (5)$$

where $s - t = k(L/\rho_i)$ and $k = 1, 2, \cdots$. In other words, the throughput of channel i is guaranteed to be ρ_i during any time interval of length L/ρ_i, implying that TCRM emulates circuit switching, or time division multiple access (TDMA), in the cell level. However, since our scheme allows best-effort traffic to be transmitted when time slots reserved by bursty real-time connections are not used, it does not lose the statistical multiplexing gain of ATM networks, unlike TDMA.

[4] Here, the worst case means that all the channels of higher priority than channel i generate their cells concurrently with channel i.

Next, we derive a simple admission-control test from (3). The schedulability test can be rewritten as

$$\sum_{j=1}^{i-1} \left\lceil \frac{\rho_j}{\rho_i} \right\rceil + 2 \leq \frac{L}{\rho_i C}, \qquad \text{for } i = 1, \cdots, M.$$

When a new channel of priority k and cell service rate ρ'_k are requested, we need to conduct the following schedulability test for all $i \geq k$:

$$\left\lceil \frac{\rho'_k}{\rho_i} \right\rceil + \sum_{j=1}^{i-1} \left\lceil \frac{\rho_j}{\rho_i} \right\rceil + 2 \leq \frac{L}{\rho_i C}.$$

By moving the second and third terms from the left-hand side to the right-hand side, we get

$$\left\lceil \frac{\rho'_k}{\rho_i} \right\rceil \leq \frac{L}{\rho_i C} - \sum_{j=1}^{i-1} \left\lceil \frac{\rho_j}{\rho_i} \right\rceil - 2. \quad (6)$$

Now, let us define the residual link capacity sequence R_i

$$R_i = \frac{L}{\rho_i C} - \sum_{j=1}^{i-1} \left\lceil \frac{\rho_j}{\rho_i} \right\rceil - 2, \qquad i = 1, \cdots, M. \quad (7)$$

Notice that R_i indicates how many more channels with the throughput requirement ρ_i can be accepted. Using R_i, we can construct a new schedulability test algorithm as follows.

Schedulability Test:

1) When a new channel of cell service rate ρ'_k and priority k is requested:
   a) Calculate R_k using ρ'_k. If $R_k \geq 0$, go to Step 2). Otherwise, reject the channel request.
   b) Check if $\lceil (\rho'_k/\rho_i) \rceil \leq R_i$ for $i \geq k$. If that is true, go to Step 3). Otherwise, reject the channel request.
   c) Update R_i's and ρ_i's for $i \geq k$ as

$$R_{i+1} \leftarrow R_i - \left\lceil \frac{\rho'_k}{\rho_i} \right\rceil$$
$$\rho_{i+1} \leftarrow \rho_i.$$

2) When an existing channel k is disconnected, update R_i's and ρ_i's for $i > k$ as

$$R_{i-1} \leftarrow R_i + \left\lceil \frac{\rho'_k}{\rho_i} \right\rceil$$
$$\rho_{i-1} \leftarrow \rho_i.$$

By storing the residual link capacity sequence R_i, we can reduce the amount of calculation for the channel schedulability test. The computational complexity of the above algorithm is $O(M)$.

D. Bounding End-to-End Delays in Multihop Connections

Using the fact that TCRM guarantees the minimum throughput ρ_i for channel i, we can derive the end-to-end delay bound of channel i. Given the input traffic specification (σ_i, ρ_i) of channel i, during any time interval of length t, the amount of traffic generated by the user may not exceed $\sigma_i + t \cdot \rho_i$. For convenience, we assume that σ_i is an integer multiple of the length of one cell, L. Due to the UNI's traffic regulation, the cell-arrival rate at the network entrance is limited by ρ_i and the burst is held at the buffer of the UNI. We assign a buffer space of σ_i for channel i.

We now show the boundedness of end-to-end delivery delays. First, we show that the queue size at the input buffer of the UNI cannot be larger than σ_i. During a time interval $[s,t)$ for any s,t such that $s \leq t$, the maximum amount of traffic that has arrived at the UNI, $x_i(s,t)$, is given by

$$x_i(s,t) = \sigma_i + \rho_i(t-s)$$

under the leaky bucket model. However, since the traffic is transmitted cell by cell, the maximum number of cells that have arrived at the UNI is given by

$$x_i(s,t) = \sigma_i + L\left\lfloor \frac{s-t}{L/\rho_i} \right\rfloor.$$

During the same interval, the minimum number of cells transmitted at the UNI, $y_i(s,t)$, is given by

$$y_i(s,t) = L\left\lfloor \frac{s-t}{L/\rho_i} \right\rfloor$$

which comes from the fact that channel i is guaranteed to have the minimum throughput ρ_i in the cell level. Therefore, the maximum backlog at the input buffer during the interval $[s,t)$ is given by $B_{\max} = x_i(s,t) - y_i(s,t) = \sigma_i$, and the maximum number of cells that can exist at the buffer is σ_i/L.

Using this fact, we can derive the following theorem on the end-to-end delivery delay bound in ATM networks.

Theorem 2.1: If a real-time channel i is specified by (σ_i, ρ_i) and its guaranteed throughput is ρ_i, then the end-to-end delivery delay of any cell belonging to channel i is bounded by

$$D_i = \frac{\sigma_i}{\rho_i} + N\frac{L}{\rho_i} + \sum_{k=1}^{N} e_k \tag{8}$$

where N is the number of hops that channel i must take and e_k is the propagation delay at the kth link.

Proof: The proof of this theorem is straightforward, since the end-to-end delivery delay D_i is given by

$$\begin{aligned} D_i &= \text{the maximum queueing delay at the UNI} \\ &\quad + \sum_{k=1}^{N} \text{the maximum queueing delay at the } k\text{th node} \\ &\quad + \sum_{k=1}^{N} \text{the propagation delay at the } k\text{th link} \\ &= \frac{\sigma_i}{L} \cdot \frac{L}{\rho_i} + N\frac{L}{\rho_i} + \sum_{k=1}^{N} e_k. \end{aligned} \tag{9}$$

The first term in (9) comes from the fact that the maximum number of cells in the buffer of the UNI is σ_i/L, and each cell's queueing delay is L/ρ_i because of the UNI's traffic regulation. The reason why the delay at the traffic controller of each link is omitted in the second term is that the traffic controller does not hold any cell if it has arrived at the latest arrival time from the previous node. In such a case, the logical arrival time is the same as the actual arrival time. Even if the cell has arrived earlier than its latest arrival time, it is held at the traffic controller only until its latest arrival time. This is self-evident from the calculation of the logical arrival time in (1) and (2). The third term is needed because of the propagation delay at each switch. By arranging the equation, we obtain

$$D_i = \frac{\sigma_i}{\rho_i} + N\frac{L}{\rho_i} + \sum_{k=1}^{N} e_k. \qquad \square$$

In general, because the burst size will be much larger than the length of one cell, the end-to-end delay bound D_i will be dominated by σ_i/ρ_i.

III. Comparative Evaluation of Channel Admissibility

To demonstrate the efficiency of TCRM, it is necessary to investigate the maximum link utilization by real-time traffic. Liu and Layland proved that the least upper bound of link utilization of the preemptive version of the rate-monotonic priority scheduling is approximately 70% when the number of real-time channels is large, while it is 100% for the deadline-driven scheduling [7]. The least upper bound of link utilization does not mean that link utilization greater than the bound cannot be achieved. A higher utilization can be achieved if task periods are suitably related to each other, but still its link utilization is lower than that of the deadline-driven scheduling. This is similar to the case of nonpreemptive rate-monotonic priority scheduling, although its link utilization is lower than that of the preemptive version. For example, the maximum link utilization of nonpreemptive rate-monotonic priority scheduling for one channel is 50%, which can be calculated easily from (3), while the preemptive version can achieve 100%. From this, we may conclude that the link utilization of our scheme can be very low compared to PGPS in some cases. As in a preemptive version, however, higher utilization can also be achieved for a nonpreemptive version. Rather than deriving a theoretical link utilization bound, in this section, we investigate empirical link utilizations of PGPS, RTC, RCSP, and TCRM using traces of MPEG-compressed videos.

As stated earlier, both RTC and PGPS adopt the (σ_i, ρ_i) model, while RCSP employs the $(X_{\min}, X_{\text{ave}}, I, S_{\max})$ model. Here, $S_{\max}$ is the length of an ATM cell, i.e., 53 bytes. Although RTC adopts the leaky bucket model as its input traffic specification, it interprets ρ_i not as the average cell-arrival rate but as the maximum traffic-arrival rate [15]. For this reason, RTC can accept real-time channels only when the sum of the peak traffic arrival rates is smaller than the link capacity.

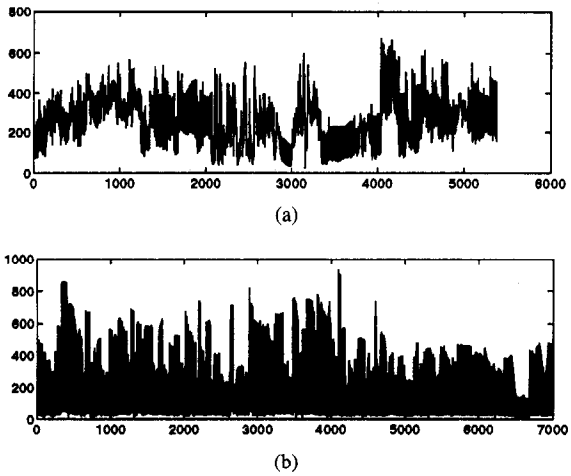


Fig. 3. Frame sizes of compressed video sequences. (a) *Star Wars*. (b) *Honey, I Blew Up the Kids*.

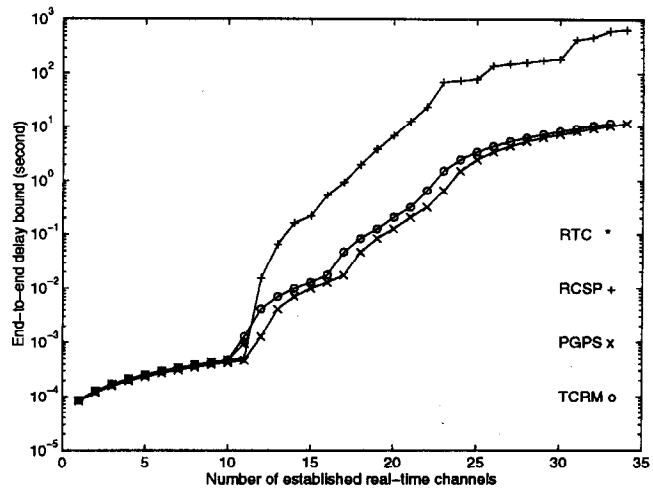


Fig. 4. Achievable end-to-end delay bounds: *Star Wars*.

Our network model is a homogeneous ATM network which consists of 11 serially connected nodes. We have chosen this multihop network in order to consider inter-dependency of delays at different links under PGPS and TCRM [see (8)]. Also, we consider such a simple network configuration rather than a more general configuration in which real-time channels are routed and switched at intermediate nodes because the end-to-end delay bound of a real-time channel is not affected by switching and routing because of its *firewall* property [16]. As long as each intermediate link provides a local delay bound, the end-to-end delay bound of a real-time channel does not change, regardless whether it joins and leaves other channels at the intermediate links or not. In the network, the first node is the sender and the 11th node is the receiver. Thus, the path from the sender to the receiver has ten intermediate links. Each link has the transmission bandwidth of 100 Mbits/s. The traffic data used in the calculations are obtained from two MPEG-coded movie clips: *Star Wars* (Sequence 1) and *Honey, I Blew Up the Kids* (Sequence 2). An MPEG-coded video yields an exemplary type of VBR traffic. These two sequences consist of frames generated once every 1/30 s. The frame sizes of two sequences are plotted in Fig. 3, where the x axis is the frame number and the y axis is the frame size measured in cells. In this example, we consider two cases; one is multiplexing homogeneous traffic, and the other is multiplexing heterogeneous traffic. In multiplexing homogeneous traffic, we consider either sequence alone and calculate the end-to-end delay bound against the number of real-time channels established. In multiplexing heterogeneous traffic, we attempt to establish real-time channels for both sequences together, and calculate the number of real-time channels of Sequence 1 that can be established while varying the number of real-time channels of Sequence 2.

A. Multiplexing Homogeneous Traffic

First, let us consider Sequence 1. The maximum number of cells per frame is 670, and the average number of cells per frame is 227.69. Since the size of a cell is 424 bits (53 bytes $\times$ 8 bits/byte $=$ 424 bits), the peak traffic-generation rate is 8.52 Mbits/s, and the average traffic generation rate is 2.89 Mbits/s. We set the user's end-to-end delay requirement to 10 $\times$ 1/30 s. Then, the local delay bound at each link is 1/30 s. We now want to compute the number of real-time channels that can be established. By reserving a bandwidth equal to the peak traffic generation rate for each channel, the number of channels that can be established in a circuit-switched network is

$$\left\lfloor \frac{100 \text{ Mbits/s}}{8.52 \text{ Mbits/s}} \right\rfloor = 11 \text{ (channels)}.$$

To calculate the end-to-end delay bounds under TCRM, we first determine the burst size σ_i and the required throughput ρ_i from the frame size sequence $\{f_i\}_{i=1,\cdots,N}$, where N is the number of frames in the sequence. We assume that the link capacity is evenly distributed to the real-time channels on the link. Then, the throughput ρ_i is obtained by dividing the link capacity by the number of channels plus one, which is needed to pass the schedulability test. Note that this throughput must be larger than the average traffic-generation rate of a real-time channel (here, 2.89 Mbits/s) to bound end-to-end delays. The bucket size σ_i is given as the maximum number of cells accumulated at the UNI when the average cell drain rate is ρ_i. Then,

$$\sigma_i = \max_m \max_n \left\lceil \sum_{l=n}^{n+m} \left(f_l - \frac{\rho_i}{30L} \right) \right\rceil, \qquad m, n \in \{1, \cdots, N\}. \tag{10}$$

By plugging σ_i and ρ_i into (8), we calculate the end-to-end delay bounds for Sequence 1. In this example, we assume that the propagation delay is negligible. The results are plotted in Fig. 4; one can establish a maximum of 21 channels, given that the user-requested end-to-end delay bound is 1/3 s.

For PGPS, we derive (σ_i, ρ_i) in the same way as we did for TCRM, except that we obtain ρ_i by dividing the link capacity by the number of channels, because PGPS uses the link utilization test for its admission control. Using the formula

in [10], we calculate the end-to-end delay bounds. Given that the user-requested bound is 1/3 s, a maximum of 22 channels can be established.

With RTC, we can establish 11 channels based on the schedulability test in [8]. This number is the same as that of a circuit-switching network. This is due to the fact that RTC interprets ρ_i as the maximum traffic-arrival rate, as stated earlier. The end-to-end delay bounds are derived based on the worst case response time in [8].

For RCSP, we use here only one priority level with a local delay bound of 1/30 s because the characteristics of all the channels are assumed to be homogeneous. In [17], the local delay bound is calculated differently, depending on whether the peak utilization exceeds 1 or not. We calculated the parameters $X_{\min}, X_{\text{ave}}$, and I from the frame size sequence as follows. First, the minimum cell inter-arrival time $X_{\min}$ is simply given by a frame interval divided by the maximum frame size. For the average cell inter-arrival time X_{ave}, we must consider the average cell-arrival rate. As with TCRM and PGPS, the average traffic arrival rate is given by the link bandwidth divided by the number of real-time channels established. Then, the average cell-arrival rate is the average traffic arrival rate divided by the cell size, and X_{ave} is given by the reciprocal of the average cell-arrival rate. I is determined so that the average cell inter-arrival time during any time interval over I is larger than X_{ave}. According to our calculation, I becomes very large as the peak link utilization exceeds 1. This is why the delay bound jumps suddenly as the peak link utilization exceeds 1. Given that the user-requested end-to-end delay bound is 1/3 s, a maximum of 15 channels can be established.

In Fig. 4, until the 11th channel is established, all four schemes show reasonable end-to-end delay bounds. However, when the peak utilization exceeds 1 (i.e., the number of channels is greater than 11), RCSP is shown to exhibit rapidly increasing delay bounds. RTC does not guarantee bounded delays when the peak utilization exceeds 1. TCRM and PGPS show reasonable end-to-end delays, even when the number of real-time channels is fairly large. The reason why TCRM is slightly less efficient than PGPS is that TCRM employs a fixed-priority scheduling, while PGPS adopts an optimal deadline-based scheduling.

Next, let us consider Sequence 2. The maximum number of cells per frame is 930, and the average number of cells per frame is 118.29. Thus, the peak traffic-generation rate is 11.83 Mbits/s, and the average traffic generation rate is 1.50 Mbits/s. Then, the number of channels that can be established in a circuit-switched network is 9, and the maximum number of channels that can provide bounded delays based on the average traffic generation is 66. The end-to-end delay bounds are plotted in Fig. 5. This figure shows a result similar to that of Sequence 1.

B. Multiplexing Heterogeneous Traffic

In order to compare the channel admissibility in a heterogeneous environment, we calculate the number of establishable real-time channels of Sequence 2 with a fixed number of real-time channels of Sequence 1 already established. To this end, we need to fix the traffic description parameters. We use the same network model and user requirements considered in the homogeneous case, and we omit the analysis of RTC since its channel admissibility is quite low, as shown in the case of multiplexing homogeneous traffic.

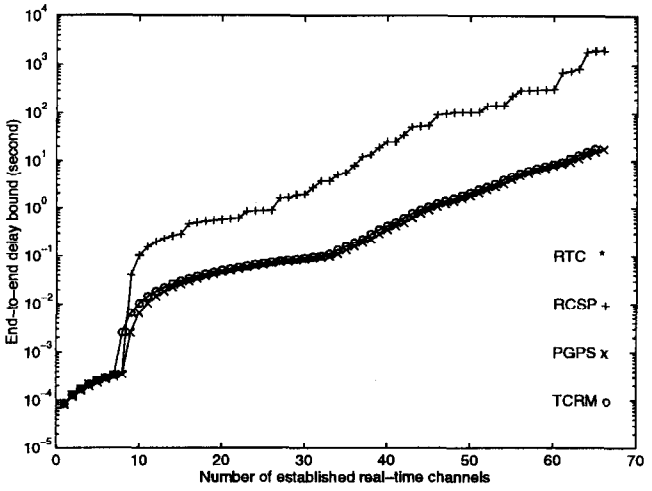


Fig. 5. Achievable end-to-end delay bounds: *Honey, I Blew Up the Kids.*

TABLE I
TRAFFIC DESCRIPTION PARAMETERS FOR TCRM, PGPS, AND RCSP

		Sequence 1	Sequence 2
TCRM	σ_i	9.761×10^5 (bits)	7.334×10^5 (bits)
	ρ_i	4.762×10^6 (bits/sec)	2.564×10^6 (bits/sec)
PGPS	σ_i	9.761×10^5 (bits)	7.334×10^5 (bits)
	ρ_i	4.762×10^6 (bits/sec)	2.564×10^6 (bits/sec)
RCSP	X_{min}	4.975×10^{-5} (sec)	3.584×10^{-5} (sec)
	X_{ave}	6.36×10^{-5} (sec)	6.784×10^{-5} (sec)
	I	0.1 (sec)	0.1 (sec)

Based on the user requirements and traffic characteristics, we derived the traffic description parameters for TCRM, PGPS, and RCSP as shown in Table I. We obtain these values from the case when the maximum number of real-time channels are established given that the user end-to-end delay bound is 1/3 s. For example, the parameters of RCSP for Sequence 1 are derived when the number of real-time channels is 15 (see Fig. 4). Interestingly, the parameters of TCRM and PGPS have the same values. This is because the end-to-end delay bounds of both methods are very close to each other (see (8) and the end-to-end delay bound equation of PGPS [10]). Note that Sequence 1 requires a higher bandwidth than Sequence 2, although the peak traffic generation of Sequence 2 is higher than that of Sequence 1. This indicates that Sequence 2 is more bursty than Sequence 1.

Using the derived traffic parameters, we first establish a fixed number of real-time channels consisting of Sequence 1, then determine the maximum number of establishable real-time channels consisting of Sequence 2. As the number of channels of Sequence 1 increases, that of Sequence 2 decreases. Fig. 6 shows the results for RCSP, PGPS, and TCRM. Note that the channel admissibility of TCRM is very close to that of PGPS, while that of RCSP is not very good.

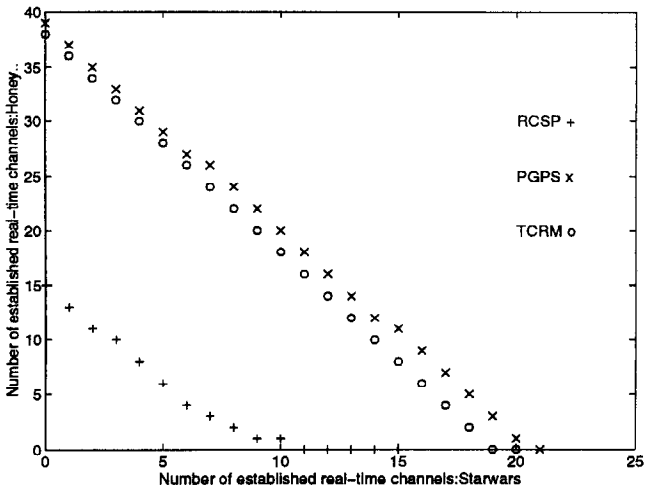


Fig. 6. Channel accommodability for the heterogeneous case.

In this example,[5] we have verified that the performance of PGPS is superior to that of RCSP and RTC. This is, as pointed out in [18], because the end-to-end delay bounds are simply given as the sums of the worst case delays at intermediate links along the path of a channel in RCSP and RTC. They ignore the inter-dependency of link delays at different intermediate links unlike PGPS. In PGPS [10], if we ignore minor terms, the end-to-end delay bound is given by

$$D_{\text{end-to-end}} = \frac{\sigma_i + KL}{\rho_i} \tag{11}$$

where K is the number of hops of the path that channel i takes. Thus, the worst case cell delay under PGPS consists of the time needed to clear the maximum burst of size σ_i at a drain rate ρ_i and the time needed to transmit an ATM cell at ρ_i at all the intermediate links. Although the end-to-end delay bound of TCRM in (8) consists of the same terms used in (11) except propagation delay, its physical meaning is different from that of PGPS. While backlogs can be built up at any intermediate link in PGPS because of its work-conserving policy, bursts are always held at the UNI in TCRM. Also, at the intermediate links, both the maximum and the minimum throughput guarantees are given as ρ_i in TCRM, whereas only the minimum throughput is guaranteed in PGPS.

Holding the burst at the network entry was also considered in [18], which was published at the same conference where an earlier version of this paper was published. Although their work has been done independently of our work, their approach and ours have many similarities. Besides the function of the UNI, service disciplines employed inside the network share the same structure. Their service discipline, Rate-Controlled Service with Earliest-Deadline-First (RCS-EDF), consists of rate controllers and a link scheduler like TCRM. The only difference is that their link scheduler is a deadline-based scheduler, while our link scheduler is a rate-monotonic priority scheduler. Because of its deadline-based scheduling policy, RCS-EDF can achieve the same channel admissibility as PGPS shown in this section. While they gave a theoretical result on how to simulate the performance of PGPS using a rate-controlled service discipline, our goal is to provide a service discipline which is simple to implement, while still achieving reasonably good performance.

[5] Delay-EDD [3] is omitted in this comparison, as it shares many similarities with RTC and RCSP. In particular, it shares the same input traffic specification with RCSP, but its admission control test for deterministic performance guarantees is based solely on the peak traffic generation rate and, thus, its channel admissibility is very poor.

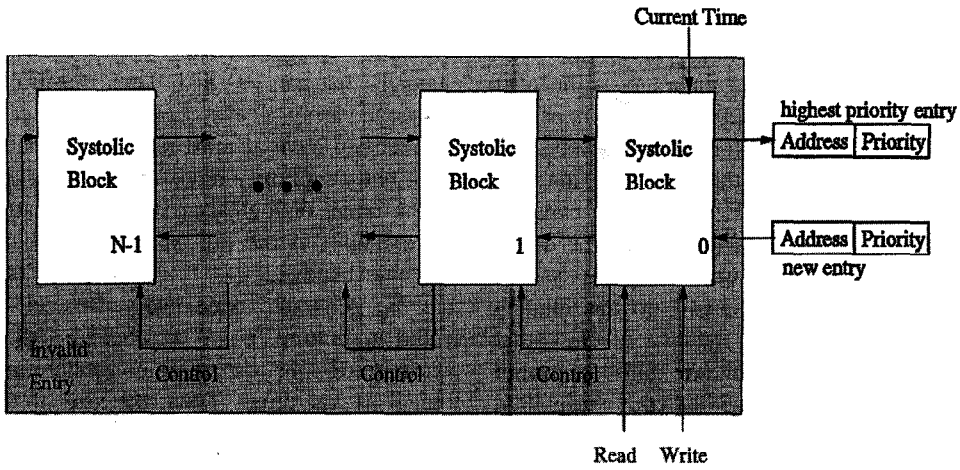
IV. Implementation

We now present a simple hardware implementation of TCRM and discuss its implementation cost against other schemes.

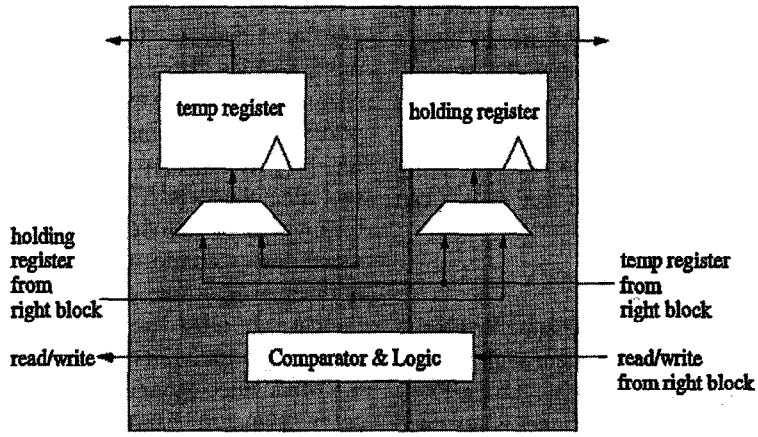
A. Traffic Controller

As stated in Section II, a traffic controller is assigned to each real-time channel. In an actual hardware implementation, however, one traffic controller is made responsible for all real-time channels for cost and flexibility reasons. Had one traffic controller been assigned to each real-time channel, a traffic controller must be assigned (deassigned) every time a real-time channel is established (terminated). Although the traffic controller can be implemented with a modified calendar queue, as discussed in [19], its scalability is limited for two reasons. First, the number of the time bins of the calendar queue depends on the range of logical arrival times. For instance, if the link capacity is 1 Gb/s and if the minimum throughput that can be allocated to a real-time channel is 1 kb/s, a cell of a particular channel may arrive once every 10^6 cell-transmission times. Then, the number of the time bins needed is at least 10^6 if a single time bin corresponds to one cell-transmission time. As the link capacity grows while the minimum allocable throughput is fixed, the number of the time bins also increases. Although a hierarchical structure can mitigate this problem, the result is that the hardware implementation becomes more complex. Second, the storage requirement of each time bin is the maximum number of real-time channels that can be established, assuming that cells are directly stored in the time bins. Consider the worst case scenario that all the channels have cells with the same logical arrival times. In such a case, all cells are stored in the same time bin, leaving the other time bins empty. In order to avoid cell loss, each time bin must be able to store a cell per channel. In this example, the number of real-time channels that can be established simultaneously is 10^6 and, thus, the total storage requirement of the calendar queue is 10^{12} cells. Since the number of cells that can co-exist in the queue is at most 10^6, the memory utilization will be extremely poor ($= 10^{-6}$). In addition, such an excessive storage requirement limits the scalability of a traffic controller implemented with the calendar queue.

Employing a dynamic priority queue is one way to avoid this inefficient memory usage. In the dynamic priority queue approach, even if all cells have the same logical arrival time, they are not stored in the same time bin, but stored in separate entries while keeping the same priority. Thus, the number of entries needed is the maximum number of cells that can simultaneously reside in the traffic controller. In this approach, each cell is indexed with its logical arrival time, and the priority queue sorts cells using their indexes. Thus, the cell



(a)



(b)

Fig. 7. (a) The systolic array priority queue and (b) systolic array block.

with the earliest logical arrival time is assigned the highest priority. Only the cell in the highest priority level will be checked for its eligibility. If the cell is *on time*, i.e., its logical arrival time equals the current time, it will be transferred to the scheduler. Otherwise, the cell is held in the priority queue until its logical arrival time is reached.

There are several hardware implementations of a dynamic priority queue [20]–[25]. Among them, the systolic array priority queue [24], [25], despite its high implementation cost, is the only candidate known to date that satisfies both the speed and scalability requirements. Hence, we use a systolic array priority queue to design the traffic controller. The example traffic controller based on the systolic array priority queue in Fig. 7 consists of an array of identical blocks, with each block holding a single entry. When a new cell of channel i arrives, it is stored in a shared memory, and its logical arrival time is calculated using (2). The calculation can be done using a simple adder which needs two state variables, channel i's minimum cell inter-arrival time, (L/ρ_i), and the time when the last cell of channel i was transferred to the scheduler. The former is constant for channel i and the latter is recorded every time a cell is transferred to the scheduler. Then, the address and the logical arrival time of the cell are recorded in the address field and the priority field of the new entry in the systolic array priority queue, respectively. That is, the logical arrival time of the cell is assigned to the priority index of the entry, the address field of which contains the location of the cell in the shared memory. When a newly arrived cell is inserted into the new entry of the queue, only the zeroth block compares the priority of its entry with that of the new entry. During the next cycle, the cell with the larger logical arrival time is inserted into the left neighbor's block. The left neighbor's block repeats the same process of comparing and sending the lower priority entry to the next block each cycle. Thus, the systolic array does not become fully sorted until several cycles after inserting the newly arrived cell. However, both insertion and removal of an entry are constant-time operations from the viewpoint of an outgoing link. Because each block passes lower priority entries to the next block, the zeroth block always holds the highest priority entry (i.e., the one with the smallest logical arrival time) in the queue. The logical arrival time of the entry in the zeroth block is compared with the current time. If they match, the entry is removed from the priority queue, and the corresponding cell is *logically transferred* to the scheduler, meaning that the cell's address and the throughput reserved for channel i are transferred into the scheduler. Once an entry is removed from the block, the zeroth block gets the entry from its left neighbor block, performing a right shift on the entire queue.

Each systolic array block consists of a holding register, which stores the entries in a sorted order, as well as a temporary register, which holds entries en route to the next block to the left. The lower priority entry in a block is passed during an enqueue operation. Multiplexors, a comparator, and decision logic also make up the rest of the block. The diagram of a systolic array block is given in Fig. 7. Queue capacity is increased by adding more blocks to the end of the queue (without using a central controller), thus making the systolic array priority queue scalable. Also, there is no bus loading problem which usually limits the scalability of the other priority queue structures [26]. As a result, the traffic controller built with a systolic array priority queue has good scalability.

The number of priority levels (i.e., the number of possible logical arrival times) affects the cost and delay of the comparator. It increases linearly with each extra bit in the priority field. The delay of the comparator can be controlled by employing a parallel comparator [27] at the expense of higher implementation cost. Another drawback is that the systolic array priority queue requires twice as many registers as the other structures [20]–[25] because of the temporary register in each block. Moon *et al.* proposed a modified systolic array priority queue in order to solve this problem [26] which can be used to implement a traffic controller.

B. Rate-Monotonic Priority Scheduler

In the rate-monotonic priority scheduler, a channel's priority is fixed according to its throughput requirement and, thus, it can be implemented with a fixed priority queue, e.g., a FIFO priority queue [21]. In this approach, each channel is assigned its own private FIFO queue. Thus, when a cell of channel i arrives, it is inserted into the channel's FIFO queue. During a cell-transmission stage, a priority encoder scans the heads of these FIFO queues (in decreasing order) and removes a cell from the first nonempty FIFO queue. Although this architecture does not require any complex function and, thus, is simple to implement, it has poor scalability because increasing priority levels requires adding more FIFO queues, which results in: 1) added hardware cost and increased complexity of the priority encoder and 2) increased delay in the priority encoder.

The scalability of a systolic array priority queue makes it also suitable for implementing a rate-monotonic priority scheduler. The scheduler built with a systolic array priority queue has basically the same structure as the traffic controller built with it, except for the following differences. First, while the priority index indicates a cell's logical arrival time in the traffic controller, in the scheduler, it indicates the reserved throughput of a real-time channel. Since the reserved throughput of a channel is constant, the priority index need not be calculated every time a cell arrives at the scheduler. Second, the cell in the zeroth block, i.e., the one with the highest priority, is immediately transmitted via an outgoing link when the link is idle, whereas in the traffic controller, the cell with the highest priority is held until its logical arrival time is reached.

C. Implementation Cost

As mentioned earlier, in spite of its high implementation cost, the use of a systolic array priority queue in implementing the traffic controller can be justified because of its good scalability. Let N denote the maximum number of real-time channels that can co-exist. Then, the traffic controller needs N systolic array blocks to guarantee no cell losses, since each channel can have at most one cell in the scheduler. The size of an entry's priority index in a systolic array block depends on the range of logical arrival times in the traffic controller. For example, if the link speed is 1 Gb/s and the minimum allocable bandwidth is 1 kb/s, cells of a particular channel may be generated once every 10^6 cell-transmission times. If the minimum "grain" of logical arrival time is one cell transmission time, the traffic controller can have 10^6 different logical arrival times. Then, the size of the priority index is given by $\log_2 10^6 \approx 20$.[6] A 20-bit comparator is expensive, but this cost is unavoidable in any scheme that uses timing information in scheduling cells, e.g., PGPS and RCSP.

Let us consider the implementation cost of a rate-monotonic priority scheduler. The number of systolic array blocks in the scheduler also equals the maximum number of real-time channels that can be established. The difference is in the size of priority indexes of the entries in each block. In the scheduler, the priority indexes represent the bandwidths allocated to real-time channels. Theoretically, TCRM may assign arbitrary bandwidths to individual real-time channels and, thus, in the worst case scenario, the number of priority levels equals the number of real-time channels. In reality, however, this is not the case. In most applications, certain typical bandwidths are assigned to individual channels, e.g., 64 kb/s voice channels, 1.5 Mb/s video channels, etc. Thus, it is reasonable to assume a set of "standard" bandwidths that can be assigned to individual real-time channels. In this case, if two channels have the same bandwidth requirement, the cells of both channels are assigned the same priority index. If there are more than two cells with the same priority index in the scheduler, ties are broken arbitrarily. If a real-time channel with a nonstandard bandwidth requirement is requested, one may assign the nearest higher standard bandwidth. Then, if the number of allowed bandwidth levels is 1000, a 10-bit comparator is needed for each systolic array block.

The other implementation costs of TCRM come from a table for storing throughputs of real-time channels and a table for storing the logical arrival times of the cells seen last in the scheduler for each channel. In order to avoid the calculation of the minimum cell inter-arrival time, L/ρ_i, one may store the minimum cell inter-arrival time instead of the throughput requirement. In this case, the minimum cell inter-arrival time is used to find the priority level of a cell in the scheduler. Storing per-connection state information is unavoidable in any cell scheduling scheme when per-connection QoS guarantees need to be provided.

In terms of implementation cost and scalability, the implementation of TCRM with a systolic array priority queue is much better than the other schemes that are known to provide optimal performance: PGPS [6], [10] and RCS-EDF [18]. Let us first consider PGPS. PGPS can also be implemented using the same systolic array priority queue as TCRM. Considering

[6] In order to avoid cell losses when multiple cells have an identical logical arrival time, the time grain must be smaller than one cell transmission time.

scalability, this implementation is the only choice for PGPS. Since PGPS is a work-conserving service discipline, it does not require two priority queues, but a single priority queue. It may seem advantageous to have a single priority queue, but PGPS requires excessive storage space because of its work-conserving service policy; that is, the buffer requirement of PGPS increases as we move along the downstream nodes [10]. Even without considering the increase of buffer requirement at the downstream nodes, the minimum buffer requirement for channel i at every link is larger than σ_i. The burst size σ_i actually depends on applications, but it is generally large. For instance, the burst size of Sequence 1 in the previous section is 2302 cells ($9.761 \times 10^5/424$). Thus, PGPS requires a buffer of 2302 cells for a channel of Sequence 1, whereas the buffer requirement of TCRM is only 2 cells. Moreover, the number of systolic array blocks in the priority queue must be equal to the sum of burst sizes of all the real-time channels. In contrast, TCRM requires only 2 blocks for each channel.

Besides its buffer requirement, PGPS has a large range of deadlines because of its work-conserving service policy. The range of deadlines in PGPS is close to the worst case end-to-end delay bound, i.e., approximately σ_i/ρ_i for channel i. The range of deadlines determines the size of the priority index of comparators and a longer range makes the implementation of PGPS costlier. By contrast, TCRM has a smaller range of logical arrival times and, thus, requires cheaper comparators.

Finally, PGPS requires the calculation of the virtual finish time of each cell, which is quite complex and computationally expensive. Although some approximation approaches like Self-Clocked Fair Queueing (SCFQ) [28], [29] may simplify the calculation, they cannot avoid the large buffer requirement and the large range of deadlines because they still employ work-conserving service policies.

Now, let us consider RCS-EDF, which achieves the same performance as PGPS. Although RCS-EDF is a nonwork-conserving service discipline, its implementation cost, however, is much lower than that of PGPS. In RCS-EDF, the rate controller and the EDF scheduler basically have the same structure, since they both sort cells using timestamps which are logical arrival times in the rate controller and deadlines in the EDF scheduler. Because of the nonwork-conserving service property, the ranges of logical arrival times and deadlines are identical. The only difference is that the EDF scheduler does not need a comparator which monitors the logical arrival time of the highest priority cell. Note that the highest priority cell is not held, but transmitted immediately in the EDF scheduler. Overall, RCS-EDF has two systolic array priority queues which are almost identical in terms of implementation cost. Compared to TCRM, RCS-EDF has a rate controller which is the same as a traffic controller of TCRM, but has a different scheduler. Thus, we need only to compare schedulers of TCRM and RCS-EDF in terms of implementation cost. First, both schemes require the same number of systolic array blocks, since the maximum number of cells that can co-exist is the same because of their nonwork-conserving nature. However, considering the component blocks, the EDF scheduler of RCS-EDF is more expensive than the rate-monotonic priority scheduler if we assume a standard set of bandwidths employed in TCRM. In the above example, if the number of allowed bandwidth levels is 1000 and the minimum allocable bandwidth is 1 kb/s while the link speed is 1 Gb/s, TCRM requires a 10-bit comparator for each systolic array block. On the other hand, RCS-EDF requires a 20-bit comparator, since its priority indexes are not bandwidth levels, but cell deadlines, and since cell deadlines are independent of bandwidth levels.

V. Conclusion

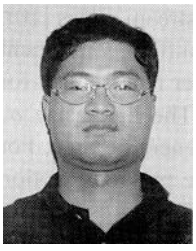
We have proposed a cell multiplexer called the TCRM to realize real-time communication in ATM networks. TCRM: 1) provides bounded end-to-end delays which are essential for real-time communication and 2) is simple enough to operate in large-scale high-speed ATM networks. We have achieved this goal by employing traffic shaping at the UNI and separating the traffic controller from the rate-monotonic priority scheduler. TCRM requires only a small buffer space inside the network, i.e., buffer space for only two cells for each real-time channel, one for the traffic controller and the other for the scheduler. TCRM is shown to not only emulate circuit switching in the cell level, but also provide VBR services without losing statistical multiplexing gain of ATM networks. We have developed a simple admission-test algorithm and also presented a hardware implementation of TCRM using a systolic array priority queue. This implementation scales well (important for large-scale high-speed networks) and requires far less memory than the implementation of PGPS. TCRM has been compared to RTC, RCSP, and PGPS in terms of simplicity and efficiency (channel admissibility). TCRM is similar to RCSP and RTC in terms of implementation complexity, but can achieve high channel admissibility similar to that of PGPS, which is much more complex to implement than TCRM.

By disallowing interaction among real-time channels, TCRM provides deterministic real-time communication services. Using deterministic real-time communication services does not necessarily result in inefficient usage of network resources, since the bandwidth which is reserved but left unused by real-time channels can be used for transmitting nonreal-time traffic. However, it limits the number of real-time channels that can be accommodated. In order to accommodate more real-time channels, different real-time channels must be multiplexed statistically. Currently, we are investigating how often QoS guarantees are violated in the presence of statistical multiplexing.

References

[1] R. L. Cruz, "A calculus for network delay, part I: Network elements in isolation," *IEEE Trans. Inform. Theory*, vol. 37, pp. 114–131, Jan. 1991.
[2] ——, "A calculus for network delay, part II: Network analysis," *IEEE Trans. Inform. Theory*, vol. 37, pp. 132–142, Jan. 1991.
[3] D. Ferrari and D. C. Verma, "A scheme for real-time channel establishment in wide-area networks," *IEEE J. Select. Areas Commun.*, vol. 8, pp. 368–379, Apr. 1990.
[4] H. Zhang and D. Ferrari, "Rate-controlled static-priority queueing," in *Proc. IEEE INFOCOM'93*, 1993, pp. 227–236.
[5] H. Zhang and S. Keshav, "Comparison of rate-based service disciplines," in *Proc. ACM SIGCOMM*, 1991, pp. 113–121.

[6] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single node case," *IEEE/ACM Trans. Networking*, vol. 1, pp. 344–357, June 1993.
[7] C. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment," *J. Assoc. Comput. Mach.*, vol. 20, no. 1, pp. 44–61, Jan. 1973.
[8] D. D. Kandlur, K. G. Shin, and D. Ferrari, "Real-time communication in multi-hop networks," in *Proc. 11th Int. Conf. Distributed Computing Systems*, May 1991, pp. 300–307.
[9] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *Proc. ACM SIGCOMM*, 1989, pp. 1–12.
[10] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The multiple node case," in *Proc. IEEE INFOCOM'93*, 1993, pp. 521–530.
[11] E. Knightly, D. Wrege, J. Liebeherr, and H. Zhang, "Fundamental limits and tradeoffs of providing deterministic guarantees to VBR video traffic," in *Proc. ACM SIGMETRICS*, 1995, pp. 98–107.
[12] D. Saha, S. Mukherjee, and S. K. Tripathi, "Multirate scheduling for guaranteed and predictive services in ATM networks," in *Proc. IEEE RTSS*, Dec. 1996, pp. 155–164.
[13] D. E. Wrege, E. W. Knightly, H. Zhang, and J. Liebeherr, "Deterministic delay bounds for VBR video in packet-switching networks: Fundamental limits and practical trade-offs," *IEEE/ACM Trans. Networking*, vol. 4, pp. 352–362, June 1996.
[14] M. de Prycker, *Asynchronous Transfer Mode: Solution for Broadband ISDN*. London, U.K.: Ellis Horwood, 1991.
[15] D. D. Kandlur, K. G. Shin, and D. Ferrari, "Real-time communication in multi-hop networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 5, pp. 1044–1056, Oct. 1994.
[16] Q. Zheng, "Real-time fault-tolerant communication in computer network," Ph.D. dissertation, Dep. Elect. Eng. Comput. Sci., The University of Michigan, Ann Arbor, 1993.
[17] H. Zhang, "Service disciplines for packet-switching integrated-services networks," Ph.D. dissertation, Dep. Comput. Sci., The University of California, Berkeley, 1993.
[18] L. Georgiadis, R. Guérin, V. Peris, and K. N. Sivarajan, "Efficient network QoS provisioning based on per Node traffic shaping," in *Proc. IEEE INFOCOM'96*, 1996, pp. 102–110.
[19] S. Kweon and K. G. Shin, "Traffic-controlled rate-monotonic priority scheduling," in *Proc. IEEE INFOCOM'96*, 1996, pp. 655–662.
[20] D. Pickers and R. Fellman, "A VLSI priority packet queue with inheritance and overwrite," *IEEE Trans. VLSI Syst.*, vol. 3, pp. 245–252, June 1995.
[21] J. Chao, "A novel architecture for queue management in the ATM network," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 1110–1118, Sept. 1991.
[22] H. J. Chao and N. Uzun, "A VLSI sequencer chip for ATM traffic shaper and queue management," *IEEE J. Solid-State Circuits*, vol. 27, pp. 1634–1643, Nov. 1992.
[23] K. Toda, K. Nishida, E. Takahashi, N. Michell, and Y. Yamaguchi, "Design and implementation of a priority forwarding router chip for real-time interconnection networks," *Int. J. Mini Microcomput.*, vol. 17, no. 1, pp. 42–51, 1995.
[24] P. Lavoie and Y. Savaria, "A systolic architecture for fast stack sequential decoders," *IEEE Trans. Commun.*, vol. 42, pp. 324–334, Feb./Mar./Apr. 1994.
[25] C. Leiserson, "Systolic priority queues," in *Proc. Caltech Conf. on VLSI*, Jan. 1979, pp. 200–214.
[26] S.-W. Moon, J. Rexford, and K. G. Shin, "Scalable hardware priority queue architectures for high-speed packet switches," in *Proc. IEEE Real-Time Technology and Applications Symp.*, 1997, pp. 203–212.
[27] H. J. Chao, "Architecture design for regulating and scheduling user's traffic in ATM networks," in *Proc. ACM SIGCOMM*, 1992, pp. 77–87.
[28] S. J. Golestani, "A self-clocked fair queueing scheme for broadband applications," in *Proc. IEEE INFOCOM'94*, 1994, pp. 636–646.
[29] J. Rexford, A. G. Greenberg, and F. Bonomi, "Hardware-efficient fair queueing architectures for high-speed networks," in *Proc. IEEE INFOCOM'96*, Mar. 1996, pp. 638–646.



Seok-Kyu Kweon received the B.S. and M.S. degrees in electronics from Seoul National University, Seoul, Korea. He is currently working towards the Ph.D degree in electrical engineering and computer science at the University of Michigan, Ann Arbor.

His research interests are traffic management, scheduling algorithms for high-speed networks, statistical QoS provisioning, and QoS routing.

Kang G. Shin (S'75–M'78–SM'83–F'92) received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea, in 1970 and the M.S. and Ph.D. degrees in electrical engineering from Cornell University, Ithaca, NY, in 1976 and 1978, respectively.

He is currently a Professor and Director of the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor. He has authored or coauthored about 600 technical papers and numerous book chapters in the areas of distributed real-time computing and control, computer networking, fault-tolerant computing, and intelligent manufacturing. He co-authored *Real-Time Systems* (New York: McGraw-Hill, 1997) with C. M. Krishna. In 1985, he founded the Real-Time Computing Laboratory, where he and his colleagues are investigating various issues related to real-time and fault-tolerant computing. His current research focuses on Quality of Service (QoS) sensitive computing and networking with emphases on timeliness and dependability. He has also been applying the basic research results to telecommunication and multimedia systems, intelligent transportation systems, embedded systems, and manufacturing applications. From 1978 to 1982, he was on the faculty of Rensselaer Polytechnic Institute, Troy, NY. He has held visiting positions at the U.S. Airforce Flight Dynamics Laboratory, AT&T Bell Laboratories, the Computer Science Division, Department of Electrical Engineering and Computer Science, University of California at Berkeley, International Computer Science Institute, Berkeley, CA, IBM T. J. Watson Research Center, and Software Engineering Institute, Carnegie-Mellon University. He also chaired the Computer Science and Engineering Division, Electrical Engineering and Computer Science Department, University of Michigan for three years beginning in January 1991. He was also an Area Editor of the *International Journal of Time-Critical Computing Systems*, the Program Co-Chair for the 1992 International Conference on Parallel Processing, and has served on numerous technical program committees.

Prof. Shin received the IEEE TRANSACTIONS ON AUTOMATIC CONTROL Outstanding Paper Award in 1987 and, in 1989, the Research Excellence Award from the University of Michigan. He was the Program Chairman of the 1986 IEEE Real-Time Systems Symposium (RTSS), the General Chairman of the 1987 RTSS, and the Guest Editor of the August 1987 Special Issue on Real-Time Systems of the IEEE TRANSACTIONS ON COMPUTERS. He also chaired the IEEE Technical Committee on Real-Time Systems during 1991-1993, was a Distinguished Visitor of the IEEE Computer Society, and an Editor of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS.