Group Priority Scheduling*

Simon S. Lam and Geoffrey G. Xie
Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-1188

TR-95-28 July 21, 1995

September 30, 1996 (second revision)

Abstract

We present an end-to-end delay guarantee theorem for a class of real-time guaranteed-deadline (GD) servers. The theorem can be instantiated to obtain end-to-end delay bounds for a variety of source control mechanisms and GD servers. We then propose the idea of group priority and specialize the theorem to a subclass of GD servers that use group priority in packet scheduling. With the use of group priority, the work of packet schedulers can be substantially reduced. We work out a detailed example, for the class of burst scheduling networks, to illustrate how group sizes can be designed such that the worst-case end-to-end delay of application data units in a real-time flow is unaffected by the use of group priority. Group priority can also be used in packet schedulers that provide integrated services (best-effort as well as real-time services) to achieve statistical performance gains, which we illustrate with empirical results from simulation experiments.

1 Introduction

Consider a packet switching network that delivers packets from sources to destinations. The source of a flow may negotiate with the network for service guarantees (e.g., throughput, delay, loss rate) at connection setup time. A flow requiring service guarantees is modeled as a sequence of packets traversing a fixed path through the network. A flow is said to be *real-time* if the network provides an end-to-end

*Research supported in part by National Science Foundation grant no. NCR-9506048 and the Texas Advanced Research Program grant no. 003658-220. An earlier version appeared in *Proceedings IEEE INFOCOM '96*.

delay guarantee to each packet in the flow. Conceptually, a network provides end-to-end delay guarantees to flows by implementing each communication channel as a guaranteed-deadline (GD) server, together with an appropriate admission control mechanism. Each packet arrival from a real-time flow to a GD server is given a *deadline*, or *priority value*, and the server ensures that the packet departs by its deadline.[1] Many GD service disciplines have been proposed [1, 2, 3, 4, 10, 11, 15].

In this paper, we assume that packets in the same real-time flow are served in FIFO order.[2] At a GD server, there is a packet scheduler which repeatedly searches for the smallest value in a set of priority values, one for each flow. (The priority value of a flow is the priority value of its head-of-line packet.) For networks of the future, we envision that a high-speed channel will be shared by hundreds, perhaps, thousands of flows, which would require a highly efficient search algorithm. Towards this goal, several heap search algorithms have been designed and studied [7, 12].

We propose the idea of group priority to substantially reduce the work of a packet scheduler in updating heaps (or any other sorted priority data structure). The basic idea is simple. Consecutive packets in a flow are partitioned into groups. Each group is given a priority value equal to the largest of the priority values of packets in the group.[3] The priority value of a flow is defined to be the priority value of its head-of-line group. Since each flow's priority value changes less frequently, for any group size larger than one, less work is required for heap updates. Empirical results in [12] show that such work reduction can be substantial, especially for a heavily utilized channel.

With the use of group priority, the largest of the packet deadlines in a group becomes the deadline to be ensured by a GD server for all packets in the group. Since the deadlines of most packets in a real-time flow are "relaxed" at each channel in the path of the flow, the worst-case end-to-end delay would increase. Thus, we observe that group priority is a mechanism for trading an increase in the worst-case end-to-end delay of packets for a decrease in the work of packet schedulers. We will, however, demonstrate that for variable bit rate (VBR) flows, group sizes can be chosen such that the worst-case end-to-end delay of *application data units* in each flow is unaffected (see Section 4.2).

Observe that most application data units are too large to be carried in a single packet and must be segmented for network delivery. To an application, the end-to-end delays (and loss rate) of its data units, rather than packets, are the relevant measures of performance. For example, a video picture to be sent over an IP network may be segmented into a sequence of IP datagrams. To an end user, the delay incurred to deliver the entire video picture is more important than the delays of individual IP datagrams. As another example, an email message may be segmented into a sequence of cells for delivery over an ATM network. The delay incurred to deliver the email message is more important than the delays of individual cells.

---

[1] We will use *deadline* and *priority value* interchangeably.

[2] Packets are given deadlines that are nondecreasing in arrival order.

[3] Note that group priority subsumes individual priority as a special case.

The balance of this paper is organized as follows. In Section 2, we introduce the class of GD servers and prove an end-to-end delay guarantee theorem. The theorem can be instantiated to obtain end-to-end delay bounds for a variety of source control mechanisms and GD servers; in particular, different GD servers can be used in the same end-to-end path. With the theorem, the problem of deriving the worst-case end-to-end delay for a real-time flow is reduced to a set of single-node problems.

In Section 3, we develop the idea of group priority in packet scheduling. For a subclass of GD servers, called the *priority subclass*, we prove a relaxed deadline theorem. The delay guarantee theorem in Section 2 is then specialized to a smaller subclass of GD servers that use group priority in packet scheduling.

In Section 4, we work out a detailed example, for the class of burst scheduling networks [9]. We derive end-to-end delay bounds for application data units,[4] each encapsulated in a sequence of packets, called a *burst*. We illustrate how to choose group sizes such that the end-to-end delays of bursts are unaffected by the use of group priority in packet scheduling.

In integrated-services packet switching networks, we believe that it is desirable to use the same packet scheduling algorithm for all service classes. Different services, such as, real-time delivery at a specified loss rate and best effort delivery, are provided to individual flows through the use of different admission control conditions and mechanisms [9, 13]. In Section 5, we present empirical results from simulation experiments, which demonstrate that the use of group priority in packet scheduling has another advantage in such integrated-services networks. Specifically, when some channels in such a network are heavily utilized, the use of group priority actually improves the network's statistical performance (loss rates, delays, and queue sizes), aside from reducing the work of packet schedulers.

2 End-to-End Delay Guarantee

Consider a packet switching network in which each packet is of variable, bounded size (in number of bits). Each communication channel in the network is statistically shared and will be referred to as a *server*.

We will focus upon a flow, say f, which is a sequence of packets. Packets in the flow traverse a path of $K+2$ nodes. Node 0 is the source of the flow, and node $K+1$ the destination. Nodes 1 through K are servers. The network is to provide an end-to-end delay guarantee to the flow. Such a flow will be called a real-time flow. (We do not care whether or not the network also provides delay guarantees to other flows that statistically share the same servers.)

Packets in flow f traverse the path in FIFO order. Specifically, the ordering of packets in flow f is preserved at every node along the path.

[4] or the data units of a protocol at the transport level or higher

Notation for server k

β_k	a nonnegative time constant associated with node k (seconds)
$\tau_{k,k+1}$	channel propagation time from node k to node $k+1$ (seconds) (each channel is assumed to be reliable and FIFO)
α_k	$= \beta_k + \tau_{k,k+1}$ (seconds)
s_k^{max}	maximum packet size at node k (bits)
C_k	transmission rate of channel from node k to node $k+1$ (bits/second)
p	an arbitrary packet served (the packet may belong to any flow)

Notation for flow f

i, j	indices of flow f's sequence of packets
$A_k^f(i)$	arrival time of packet i at node k (time when last bit of packet arrives)
$P_k^f(i)$	packet-dependent component of packet i's deadline at node k
$L_k^f(i)$	departure time of packet i at node k (time when last bit of packet leaves)
$s^f(i)$	size of packet i (bits)

$A_k^f(i)$, $P_k^f(i)$, and $L_k^f(i)$, for $i \geq 1$, are positive real numbers. Indices i and j are positive integers. Additionally, we also use m, n, and l as positive integers whose meanings depend upon context.

2.1 Guaranteed-deadline servers

A GD server provides the following service to each real-time flow f it serves:

- packets in flow f depart in FIFO order
- server ensures that the departure time of packet i is bounded as follows:

$$L_k^f(i) \leq P_k^f(i) + \beta_k \tag{1}$$

where the deadline on the right hand side has two components: (i) a packet-dependent component, $P_k^f(i)$, which depends on packet i (its arrival time, flow, size, priority, etc.), and (ii) a nonnegative constant, β_k. Note that each component may vary from one server to another, and β_k may be zero.

The function $P_k^f(\cdot)$ is not yet specified. Any function may be used so long as for a real-time flow, the server can ensure that, for all i, packet i departs by its deadline.

Many service disciplines in the literature belong to this class. They differ in packet deadlines, scheduling algorithms, and admission control conditions. Some examples and references are given in Section 2.4.

2.2 Delay guarantee theorem

Consider a real-time flow f traversing the path from node 0 to node $K+1$. Nodes 1 to K are GD servers (different service disciplines may be used at different nodes). The following lemma is immediate from the definition of α_k.

Lemma 1. For packet $i = 1, 2, \ldots$ in flow f and node $k = 1, 2, \ldots, K$, the arrival time of packet i at node $k+1$ is bounded as follows:

$$A_{k+1}^f(i) \leq P_k^f(i) + \alpha_k. \tag{2}$$

We next present a general delay guarantee formula for flow f. The formula makes use of *reference clock values* at nodes 1 to K, which are described below.

Notation for flow f

$v^f(i)$	a time constant associated with packet i (seconds)
$V_k^f(i)$	reference clock value of packet i at node k

$v^f(i)$ and $V_k^f(i)$ are positive real numbers, for $i \geq 1$. The time constant, $v^f(i)$, can be interpreted as the service time of packet i (excluding waiting time) guaranteed by each node in the path. $V_k^f(i)$ is determined as follows for $i \geq 1$, with $V_k^f(0)$ defined to be 0 :

$$V_k^f(i) = \max\{V_k^f(i-1), A_k^f(i)\} + v^f(i) \tag{3}$$

Thus $V_k^f(i)$ can be thought of as the *expected finishing time* of packet i at node k, and is to be used as a time reference in our delay guarantee formula for flow f. As such, reference clock values are neither computed nor actually implemented by the nodes.

Node k ensures that packet i departs before its deadline, which is $P_k^f(i) + \beta_k$. Therefore, $A_{k+1}^f(i)$ depends upon $P_k^f(i)$ as shown in (2), and $V_{k+1}^f(i)$ depends upon $A_{k+1}^f(i)$ as shown in (3).

A concrete way to interpret $v^f(i)$ is to assume that packet i has been allocated a throughput of $\lambda^f(i)$ bits/second at each node, such that: $v^f(i) = s^f(i)/\lambda^f(i)$. We note that adaptive throughput allocation on a per packet basis is unrealistic in practice. However, adaptive throughput allocation on a per burst basis has been proposed [9], where each burst consists of a number of packets; see Section 4.

Lemma 2. For packet $i = 1, 2, \ldots$ in flow f and node $k = 1, 2, \ldots, K-1$

$$V_{k+1}^f(i) \leq V_k^f(i) + \max_{1 \leq j \leq i} \{v^f(j) + (P_k^f(j) - V_k^f(j))\} + \alpha_k. \tag{4}$$

Theorem 1. For packet $i = 1, 2, \ldots$ in flow f, the arrival time of packet i at the destination is bounded as follows:

$$\begin{aligned} A_{K+1}^f(i) \quad \leq \quad & V_1^f(i) + \sum_{k=1}^{K-1} \max_{1 \leq j \leq i} \{v^f(j) + (P_k^f(j) - V_k^f(j))\} + \\ & (P_K^f(i) - V_K^f(i)) + \sum_{k=1}^{K} \alpha_k. \end{aligned} \tag{5}$$

This is our delay guarantee theorem. (Proofs of Lemma 2 and Theorem 1 are given in the Appendix.) By definition, the end-to-end delay for packet i is $A_{K+1}^f(i) - A_1^f(i)$. The delay guarantee in (5) can be instantiated to obtain end-to-end delay upper bounds for a variety of source control mechanisms and different service disciplines at nodes 1 through K. Specifically, the delay guarantee in (5) provides an upper bound on the end-to-end delay of packet i if

- a source control mechanism is chosen such that $V_1^f(i) - A_1^f(i)$ at node 1 has a finite upper bound, and
- a GD server is chosen for node k, $1 \leq k \leq K$, such that the term, $P_k^f(j) - V_k^f(j)$, has a finite upper bound.

Note that different service disciplines may be chosen for different nodes, and the term, $P_k^f(j) - V_k^f(j)$, may be positive or negative. With Theorem 1, *the problem of deriving an end-to-end delay upper bound for a real-time flow traversing a network path is reduced to a set of single-node problems.*

2.3 Examples of source control

The goal of source control is to upper bound $V_1^f(i) - A_1^f(i)$. A widely used mechanism is leaky bucket control. If the source of flow f is controlled by a leaky bucket with token rate ρ and bucket depth σ, then for all packet i in the flow [5],

$$V_1^f(i) \leq A_1^f(i) + \frac{\sigma}{\rho} \tag{6}$$

To obtain an end-to-end upper bound for flow f, $V_1^f(i)$ is instantiated to $A_1^f(i) + \sigma/\rho$ in the delay guarantee formula of Theorem 1.

Another example of source control is the separation timing constraint between consecutive bursts in a flow [9]; see Section 4 for more details.

2.4 Examples of GD servers

The GD class of servers is general and includes many service disciplines in the literature. There are differences in their $P_k^f(\cdot)$ functions, β_k constants, scheduling algorithms, and admission control conditions. We next discuss four well-known examples.

For a VC server, the P values are virtual clock values computed as follows [15], for all $j \geq 1$,

$$P_k^f(j) = \max\{P_k^f(j-1), A_k^f(j)\} + v^f(j) \tag{7}$$

where $P_k^f(0) = 0$, and $v^f(j)$ is equal to $s^f(j)/\lambda^f(j)$. Under certain admission control conditions, the VC server provides the guaranteed deadline in (1) with $\beta_k = s_k^{max}/C_k$ [11]. From (3) and (7), it is trivial to show that, for all j,

$$P_k^f(j) - V_k^f(j) = 0 \tag{8}$$

For a PGPS server, $P_k^f(j)$ is the *virtual-time finishing time* of packet j. Under certain admission control conditions, the PGPS server provides the guaranteed deadline in (1) with $\beta_k = s_k^{max}/C_k$ [10]. It is shown in [5] that if the server allocates a minimum rate of λ^f to every packet of flow f, such that $v^f(j) = s^f(j)/\lambda^f$, then the following holds for all j,

$$P_k^f(j) - V_k^f(j) \leq 0 \tag{9}$$

For a Delay-EDD server [2], the P values of packets are computed as follows [14], for all $j \geq 1$,

$$P_k^f(j) = \max\{A_k^f(j) + d_k^f, P_k^f(j-1) + v^f\} \tag{10}$$

where $P_k^f(0) = -v^f$, d_k^f is a local delay bound for every packet in flow f, and $v^f(j) = v^f = s^f/\lambda^f$, with s^f and λ^f being the same for all j. If certain schedulablity conditions are met, then a Delay-EDD server provides the guaranteed deadline in (1) with $\beta_k = 0$ [2]. By induction, it is easy to show that for all $j \geq 1$

$$P_k^f(j) - V_k^f(j) = d_k^f - v^f \tag{11}$$

For a leave-in-time server [3], the P values of packets are computed as follows[5] for $j \geq 1$,

$$P_k^f(j) = \max\{A_k^f(j), V_k^f(j-1)\} + d_k^f(j) \tag{12}$$
$$V_k^f(j) = \max\{A_k^f(j), V_k^f(j-1)\} + v^f(j) \tag{13}$$

where $V_k^f(0) = 0$, $d_k^f(j)$ is the local delay bound of packet j, and $v^f(j) = s^f(j)/\lambda^f$, with the reserved rate λ^f the same for every packet in flow f. Under certain admission control conditions, a leave-in-time server provides the guaranteed deadline in (1) with $\beta_k = s_k^{max}/C_k$ [3]. Subtracting (13) from (12), we have for all j:

$$P_k^f(j) - V_k^f(j) = d_k^f(j) - v^f(j) \tag{14}$$

For example, suppose every server in the path of flow f is one of the four GD servers described above. In this case, to obtain an end-to-end delay upper bound for f, we simply replace the term $P_k^f(j) - V_k^f(j)$, for $1 \leq k \leq K$, in the delay guarantee formula of Theorem 1 by the appropriate term on the right hand side of (8), (9), (11), or (14).

In summary, we have shown how to obtain end-to-end delay upper bounds for a variety of source control mechanisms and GD servers. In the next section, we illustrate how to apply the delay guarantee formula in Theorem 1 to the idea of group priority.

3 Group Priority

We first prove a theorem about relaxing deadlines for a subclass of GD servers. We then specialize the delay guarantee theorem to a subclass of GD servers that use group priority. The previous model of a flow, which is a sequence of packets, is generalized with the addition of two kinds of structure in the sequence: (i) groups, which are meaningful only to packet schedulers, and (ii) bursts, which represent data units sent from sources to destinations.

[5] The packet arrival time, $A_k^f(j)$, should be interpreted as the time when packet i becomes *eligible* in [3].

3.1 Relaxed deadline theorem

Consider a subclass of GD servers, called the *priority subclass*, with the following additional properties:

- *work conserving*—The server does not idle when there are bits to send.
- *nonpreemptive*—The transmission of a packet cannot be preempted.
- *priority service*—In selecting the next packet to serve, the packet in queue with the smallest deadline is chosen. Ties between packets of different flows are broken arbitrarily, and ties between packets of the same flow are broken by arrival times (to preserve the FIFO property).

Note that each service discipline in the priority subclass is almost completely specified. Only the P functions—$P_k^f(\cdot)$ for all f—remain to be specified. Also, since β_k is a constant, the server can use the P values of packets, rather than their deadlines, as priorities.

The following theorem is about two related systems, an original system and a modified system; we use the term *system* to refer to a particular implementation of a server k in the priority subclass. The arrival times and sizes of packets are the same in each system. The arrival time of an arbitrary packet p at server k is $A_k(p)$. In the original system, the deadline and departure time of packet p are $P_k(p) + \beta_k$ and $L_k(p)$, respectively. In the modified system, the deadline and departure time of packet p are $P'_k(p) + \beta_k$ and $L'_k(p)$, respectively. Furthermore, for all p, it is assumed that $P'_k(p) \geq P_k(p)$; the modified system is said to have *relaxed deadlines* compared to the original system.

Theorem 2. If, for all packet p, the deadline $P_k(p) + \beta_k$ is met in the original system, that is,

$$L_k(p) \leq P_k(p) + \beta_k \tag{15}$$

then, for all packet p, the relaxed deadline $P'_k(p) + \beta_k$ is met in the modified system, that is,

$$L'_k(p) \leq P'_k(p) + \beta_k. \tag{16}$$

A proof of Theorem 2 is given in the Appendix.

3.2 Groups

We proceed to develop the concept of groups, which is meaningful only to packet schedulers along the path of a flow, but not to the flow's source and destination. Consider a real-time flow whose sequence of packets is partitioned into groups of different sizes. The group sizes are parameters whose values are to be chosen such that some network performance measures are optimized (see Section 4.2 for a design example and Section 5 for empirical performance results).

Notation for groups. We use h to denote the sequence index which identifies a particular group of packets in flow f, and $h(i)$ to denote the set of packets in the group that includes packet i.

At each GD server, the largest of the deadlines of packets in a group is used as the group's deadline, i.e., to be used for scheduling every packet in the group. More specifically, consider packet arrivals from flow f to node k. For packet i in the flow, its individual priority value is $P_k^f(i) + \beta_k$. With the use of group priority, packet i is scheduled using its group's deadline, i.e.,

$$P_k^{'f}(i) + \beta_k = \max_{j \in h(i)} \{P_k^f(j)\} + \beta_k \tag{17}$$

Note that most packets in the group have relaxed deadlines. Methods for implementing the group priority idea depend upon the server's $P_k^f(.)$ function, that is, the service discipline.

3.3 End-to-end delay guarantee

We proceed to specialize the delay guarantee in Theorem 1 to a subclass of GD servers that use group priority. In the balance of this paper, we will consider servers in the priority subclass that specify $P_k^f(i)$ to be the virtual clock value of packet i in flow f, as computed by (7). More specifically, the priority values of packets in flow f are computed assuming that packet i in the flow is allocated a thoughput of $\lambda^f(i)$ bits/second at each server on the path, and that some admission control mechanism ensures that the capacity of each server is not exceeded [11].

From (3) and (7), we see that the virtual clock value of packet i in flow f at server k is equal to its reference clock value $V_k^f(i)$. With the use of group priority, the packet-dependent deadline, $P_k^{'f}(i)$, of packet i is equal to the virtual clock value of the last arrival in its group.

Let i_1 denote the first packet of some group. In order for the packet scheduler to compute $P_k^{'f}(i_1)$ when packet i_1 gets to the head-of-line position of flow f, without waiting for the arrival of the last packet in the group, it is sufficient that the packet arrival times satisfy a jitter constraint.[6] We then have at node k, for $k = 1, 2, \ldots, K$,

$$P_k^{'f}(i) - V_k^f(i) \leq \sum_{n \in h(i), n \neq i} v^f(n) \tag{18}$$

[6] For example, the interarrival time between packets i and $i+1$ in the same group is less than or equal to $v^f(i)$. For a constraint that is weaker and easier to implement, see (21) in Section 4. A description of an architecture and algorithms to implement this constraint for real-time flows can be found in [9].

With group priority, packet i is scheduled with $P_k^{\prime f}(i)$ rather than $P_k^f(i)$ for all i in flow f. Substituting (18) into the delay guarantee of Theorem 1, we have the following result.

Corollary 1. If server k on the path of flow f, for $k = 1, 2, \ldots, K$, schedules packet j using its group priority $P_k^{\prime f}(j)$, for $1 \leq j \leq i$, then the following end-to-end delay guarantee holds for packet i:

$$A_{K+1}^f(i) \leq V_1^f(i) + (K-1) \max_{1 \leq j \leq i} \{v_g^f(j)\} + (v_g^f(i) - v^f(i)) + \sum_{k=1}^{K} \alpha_k \tag{19}$$

where for $1 \leq j \leq i$,

$$v_g^f(j) = \sum_{n \in h(j)} v^f(n) \tag{20}$$

and $\alpha_k = (s_k^{max}/C_k) + \tau_{k,k+1}$.

3.4 Modeling application data units

When an application data unit (more generally, a data unit of a protocol that uses the network layer) is too large to be carried in a single packet, it is segmented for network delivery. To an application, the end-to-end delays and loss rate of their data units are the relevant measures of performance.

We proposed in [9] to generalize the model of a flow to a sequence of packets partitioned into *bursts*, each of which is a sequence of packets that carry an application data unit. Thus for packet schedulers that use group priority, the sequence of packets in a flow is partitioned in two different ways: into bursts and into groups. Note that burst partitioning is an inherent characteristic of the flow, while group partitioning is optional and performed by packet schedulers to reduce work and improve network performance. A description of how to do group partitioning is presented below.

4 A Detailed Example

For the class of burst scheduling networks, we illustrate how to implement the group priority idea and apply the delay guarantee in Corollary 1. As in [9], we assume that all packets have a fixed size. In this section, we focus upon a particular real-time flow, f, and will omit the superscript f in the following notation for clarity. The flow travels the path of $K+2$ nodes introduced in Section 2.

Notation for bursts

(m, l)	the lth packet in the mth burst of flow f
$A(m, l)$	arrival time of packet (m, l) at a node
$D(m, l)$	end-to-end delay of packet (m, l) (from arrival at node 1 to arrival at node $K + 1$)
b_m	size of burst m (packets)
δ_m	maximum duration of burst m (seconds)
λ_m	$= b_m/\delta_m$, average rate of burst m (packets/second)
g_m	$\leq b_m$, group size chosen for burst m (packets)

For burst m, we assume that the group size g_m is chosen at the network entrance such that $g_m \leq b_m$. (Alternatively, g_m can be computed by each packet scheduler from b_m and QoS parameter values negotiated for the flow at connection setup time. A method for determining group sizes is given in Section 4.2.) The burst's sequence of packets is partitioned into groups such that each group consists of g_m packets except for the last group whose size may be smaller. There are two special cases that are noteworthy: (i) $g_m = 1$ for all m (group priority is not used), and (ii) g_m is chosen to be the same as b_m.

To be guaranteed an end-to-end delay bound, a flow is required to satisfy the following burst-based flow specification when its packets arrive at the network entrance (node 1):

Flow Specification

- The first packet of burst m carries information[7] on λ_m, b_m, and g_m .
- Packets in burst m satisfy a *jitter* timing constraint, namely: for $l = 1, 2, \ldots, b_m$,

$$0 \leq A(m, l) - A(m, 1) \leq \frac{l - 1}{\lambda_m} \tag{21}$$

- Bursts in the flow satisfy a *separation* timing constraint, namely: for $m \geq 1$,

$$A(m + 1, 1) - A(m, 1) \geq \delta_m \tag{22}$$

[7]This informaton is implementation-dependent. Some other set of parameters may be used.

The information carried in the first packet of each burst allows every server in the path to allocate a reserved rate to the flow on a per burst basis. Such adaptive rate allocation is very appropriate for VBR flows, such as compressed video.

We note that the concept of a burst already exists in the ATM literature. It is known as a block in the *ATM block transfer* (ABT) capability being standardized by ITU-T [6]. The objective of ABT is to allow a source to dynamically negotiate its throughput reservation on the basis of a block of cells. A block is bracketed by two RM cells. A leading RM cell requests a reserved bandwidth for the block, and a trailing RM cell releases the reserved bandwidth. In particular, a block of cells is either discarded or accepted entirely by an ATM switch.

Two admission control mechanisms are described in [9] to provide two classes of VBR service: (i) real-time delivery with no loss—at connection set up time, a flow is allocated a reserved rate equal to the largest rate of its bursts, and (ii) real-time delivery at a specified loss rate. Specifically, overbooking is used in admitting flows for class (ii) with the following consequence: When the first packet of a burst arrives at a server, if the server cannot allocate a reserved rate equal to the average rate of the burst, the burst will be discarded.

The jitter timing constraint requires that packets of the same burst arrive at the network entrance within some bounded duration. Without this requirement, it would be impossible for any network to provide an upper bound on the end-to-end delay of the burst. This is a rather weak constraint and can be easily satisfied if the packets of a burst are derived from the same application data unit at the source, i.e., they arrive at the same time. The jitter timing constraint can be exploited to compute virtual clock values very efficiently for each flow at a server; specifically, the main steps of the computation are performed only once per burst [9].

The separation timing constraint is a source control mechanism that ensures that at node 1 the difference between the virtual clock value and arrival time of $(m, 1)$, the first packet of burst m, is upper bounded by $1/\lambda_m$ for all m. The constraint also ensures that each *active flow* [11] contains at most one active burst—this makes it easy for a server to allocate reserved rates to flows on a per burst basis and also ensure that its capacity is not exceeded by the aggregate rate allocated to flows.

To preserve the jitter and separation timing constraints when the packets of a flow arrive at node k, for $k = 2, \ldots, K$, the packets pass through a flow regulator before they arrive at the queue of the server. If a packet arrives to a node ahead of schedule, it is delayed by the flow regulator to the extent that the packet is ahead of its deadline; such delays increase the end-to-end delay lower bound for packets, but do not impact the end-to-end delay guarantee of Corollary 1. Lastly, note that Corollary 1 is applicable because the server at node k (excluding the flow regulator) is work conserving.

4.1 Delay bounds

If the channel capacity, for every channel on the path, is not exceeded by the aggregate reserved rate of active flows [11], a tight upper bound on the end-to-end delay of the first packet of a burst can be derived as a special case of Corollary 1.

Corollary 2.

$$D(m,1) \leq \frac{g_m}{\lambda_m} + (K-1) \max_{1 \leq n \leq m} \{\frac{g_n}{\lambda_n}\} + \sum_{k=1}^{K} \alpha_k. \tag{23}$$

Corollary 2 generalizes a theorem in [9] for the special case of individual priority (that is, $g_m = 1$ for all m). A tight end-to-end delay lower bound is the following:

$$D(m,1) \geq (K-1)\frac{g_m}{\lambda_m} + \sum_{k=1}^{K} \alpha_k \tag{24}$$

Since flow regulators preserve the jitter timing constraint for each burst in a real-time flow [9], the delay of packet (m,l) is bounded as follows:

$$D(m,l) \leq D(m,1) + \frac{l}{\lambda_m} \tag{25}$$

The end-to-end delay of burst m, denoted by D_m, measured from the time when packet $(m,1)$ arrives at node 1 to the time when packet (m,b_m) arrives at node $K+1$, is bounded as follows:

$$D_m \leq D(m,1) + \frac{b_m}{\lambda_m} = D(m,1) + \delta_m \tag{26}$$

4.2 How to determine group sizes

The source of a real-time flow negotiates with the network to agree upon QoS parameter values, which determine flow characteristics and service guarantees. (For a commercial network, the cost of flow delivery would depend upon these negotiated values.) In this example, we consider the following QoS parameters.

λ_{max}	maximum rate to be reserved for a burst ($\lambda_m \leq \lambda_{max}$ for all m), to be guaranteed by source
δ_{max}	maximum burst duration ($\delta_m \leq \delta_{max}$ for all m), to be guaranteed by source
D_{max}	maximum end-to-end delay of any burst in flow, to be guaranteed by network

Note that λ_m is an average rate determined by b_m and δ_m. Thus, to conform to the negotiated value of λ_{max}, it is sufficient that the source controls its burst sizes, such that, for all m,

$$\frac{b_m}{\delta_m} \leq \lambda_{max} \tag{27}$$

If the flow conforms to Flow Specification at its network entrance, the network will ensure that burst delays do not exceed D_{max}. The negotiated values of D_{max} and δ_{max} are used to determine the group sizes for bursts, as described below.

We first derive a *uniform upper bound* on $D(m,1)$ for all m. Let $m*$ denote the index of the *slowest* burst in the flow, that is, for all m,

$$\lambda_{m*} \leq \lambda_m \tag{28}$$

Suppose each burst is allocated a reserved rate equal to its average rate. For the special case of individual priority ($g_m = 1$ for all m) the slowest burst in the flow determines the uniform upper bound of $D(m,1)$, which is:

$$D(m,1) \leq \frac{K}{\lambda_{m*}} + \sum_{k=1}^{K} \alpha_k \tag{29}$$

For the general case of group priority, if g_{m*} is chosen to be 1 and g_m a positive integer such that

$$\frac{g_m}{\lambda_m} \leq \frac{1}{\lambda_{m*}} \tag{30}$$

it is easy to observe, from (23), that the same uniform upper bound in (29) applies. This observation suggests that subject to (30), *group priority can be used without increasing the worst-case delay of any burst in the flow.* To illustrate the potential benefit of using group priority, consider interframe-encoded pictures in a video flow, which have very large size fluctuations, e.g., for MPEG sequences studied in [8], an I picture is up to 30 times the size of a B picture. From (30), g_m can be as large as 30 for an I picture. Thus we see that for such video traffic, the frequency of priority changes for a flow can be significantly decreased, which reduces the work of packet schedulers.

We next consider the QoS parameter, D_{max}. In order for the network to provide the bound D_{max} to every burst, the reserved rates of bursts in the flow must be lower bounded to avoid having a burst that travels too slowly. Specifically, the minimum reserved rate for a burst should be

$$\lambda_{min} = K/(D_{max} - \delta_{max} - \sum_{k=1}^{K} \alpha_k) \tag{31}$$

Note that if λ_{min} is larger than λ_{max}, there is a conflict between the negotiated values of λ_{max} and D_{max}. A renegotiation between source and network would be required.

Suppose that this is not true. To derive a condition for determining group sizes for bursts, we consider two possible scenarios.

First, one or more bursts in the flow may be so slow that the uniform upper bound is very large, in fact, it is larger than the value of D_{max} negotiated between source and network. To ensure that the uniform upper bound is less than D_{max}, each burst in the flow must be allocated a reserved rate not less than λ_{min} at each server, i.e., the reserved rate of burst m is chosen to be $max\{\lambda_m, \lambda_{min}\}$.

Second, the value of D_{max} negotiated between source and network is larger than the uniform delay upper bound. In this case, a group size larger than 1 may be used for scheduling even for the slowest burst in the flow. This group size, denoted by g_{min}, is chosen to be a positive integer such that the following holds:

$$\frac{g_{min}}{\lambda_{m*}} \leq \frac{1}{\lambda_{min}} \tag{32}$$

In this case, the condition for selecting group sizes for bursts can be relaxed from (30) to the following:

$$\frac{g_m}{\lambda_m} \leq \frac{g_{min}}{\lambda_{m*}} \tag{33}$$

From the above discussion, the general condition for selecting the group sizes of bursts is:

$$g_m \leq min\{b_m, \lfloor \frac{\lambda_m}{\lambda_{min}} \rfloor\} \tag{34}$$

which follows from (32) and (33), and the requirement, $g_m \leq b_m$.

5 Empirical Results

We conducted experiments using a discrete-event simulator from [9]. The network simulated is illustrated in Figure 1. There are six switches labeled SW. Each switch has a buffer pool for 1200 packets, which is shared by all video flows.

Each thin arrow in Figure 1 represents a channel, which (except for L2 and L3) is labeled by its capacity in megabits per second (Mbps). The channel propagation delay, in milliseconds (ms), is shown in parentheses. Channels L2 and L3 have the same capacity C. The value of C can be changed from one experiment to another.

Each thick arrow represents a set of channels, one for each video flow. Each such channel has a capacity larger than λ_{max} of the flow it carries; the capacity varies from 10 to 15 Mbps, and the propagation delay also varies.

5.1 Video flows and ABR traffic

The simulated network carries twelve video flows, as well as some ABR traffic. In Figure 1, the source of each video flow is labeled VS, and the destination VD. The video flows travel from their sources through three different switches (SW1, SWa, SWb)

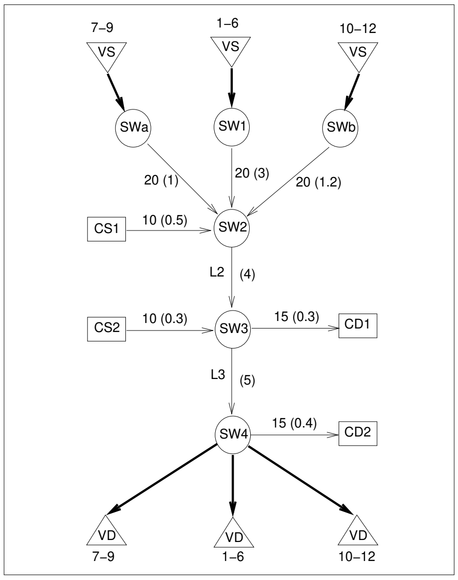

Figure 1: Simulated network.

MPEG sequence	encoding pattern (M, N)	λ (Mbps)		
		min	max	ave
Terminator	(3, 6)	0.14	3.86	1.15
ParentsSon	(3, 6)	0.37	5.97	1.51
RedsNightmare	(10, 30)	0.89	3.62	0.75
Student	(1, 4)	0.48	2.47	1.27
Driving	(3, 9)	0.17	8.48	1.88
Airwolf 2	(3, 6)	0.14	3.31	0.89
Simpsons I	(3, 6)	0.14	2.60	0.92
Canyon	(3, 6)	0.76	0.70	0.28
FlowerGarden	(3, 6)	1.39	13.25	5.04
UnderSiege	(3, 6)	0.17	2.02	0.59
StarTrek II	(0, 1)	0.28	1.15	0.62
Energizer	(3, 6)	0.17	2.34	0.76

Table 1: MPEG sequences used in experiments

to SW2. From there, they all travel through SW3 and SW4 to their destinations. The video flows were generated using traces obtained from MPEG video sequences. A profile of the video sequences is shown in Table 1. Two of the sequences, Student and Driving, were encoded by us. The other ten sequences were obtained from http://w3.eeb.ele.tue.nl/mpeg. In Table 1, the parameters N and M determine the repeating pattern of I, B, and P pictures in the sequence [8].

MPEG sequence	$g_{min} = 1$ max	$g_{min} = 1$ ave	$g_{min} = 2$ max	$g_{min} = 2$ ave	$g_{min} = 4$ max	$g_{min} = 4$ ave
Terminator	27	7.1	55	14.4	110	29.3
ParentsSon	16	3.4	32	7.3	64	14.6
RedsNightmare	40	7.3	81	14.5	162	29.4
Student	5	2.0	10	4.8	20	9.4
Driving	51	9.3	102	20.0	205	36.7
Airwolf 2	23	5.4	47	11.2	94	23.0
Simpsons I	18	5.5	37	11.5	74	23.5
Canyon	9	2.6	18	5.9	36	10.8
FlowerGarden	9	2.8	19	6.2	38	13.3
UnderSiege	12	2.8	24	6.1	48	11.6
StarTrek II	4	1.6	8	3.7	16	7.8
Energizer	14	3.6	28	7.9	56	15.0

Table 2: Group sizes for three g_{min} values

Pictures are represented as bursts defined in Section 4. The average rate of a picture is computed as follows. Each packet is 53 bytes long with a 48-byte payload. Let b_m be the number of packets needed to carry the bits of picture m. The average rate of picture m is $53 \times 8 \times b_m \times 30$ bits per second, where we have used 1/30 second as δ_m for all m. For most of our experiments (all of the performance results illustrated in this section), the packets in a burst were generated with a fixed interpacket gap.[8]

We did not try to identify values for the QoS parameters, λ_{max} and D_{max}, appropriate for a particular multimedia application. We simply used the average rates of the largest and smallest pictures in a sequence as values for λ_{max} and λ_{min}, respectively. In Table 1, λ_{ave} is the average of the average rates over all pictures.

The group size for each picture in a video sequence was calculated to be the largest integer that satisfies the inequality in (33); we experimented with several values of g_{min}. The maximum and average group sizes for each of the twelve video sequences are shown in Table 2.

In addition to the video flows, the network carried two ABR traffic flows: a flow from CS1 to CD1 via L2, and the other from CS2 to CD2 via L3. Each was a Poisson source whose rate was set to be between 0.20 and 0.21 of the capacity C of channel L2 (also L3) for each experiment.

For L2 and L3, 0.2 of the channel capacity C was allocated to ABR traffic by assigning virtual clock values to ABR packets as priority values [9]. Whenever there was nothing to send from the video flow queues, the entire channel capacity was available to ABR traffic.

[8] We conducted several experiments in which the packets of a burst were generated in batches of 40 each, with a fixed interbatch gap. Compared to the results presented herein, we observed that such batch arrivals had no impact on the worst-case end-to-end delay of bursts. The queue sizes were larger. The average end-to-end burst delays were actually smaller because, with batch arrivals, the burst durations were smaller on the average.

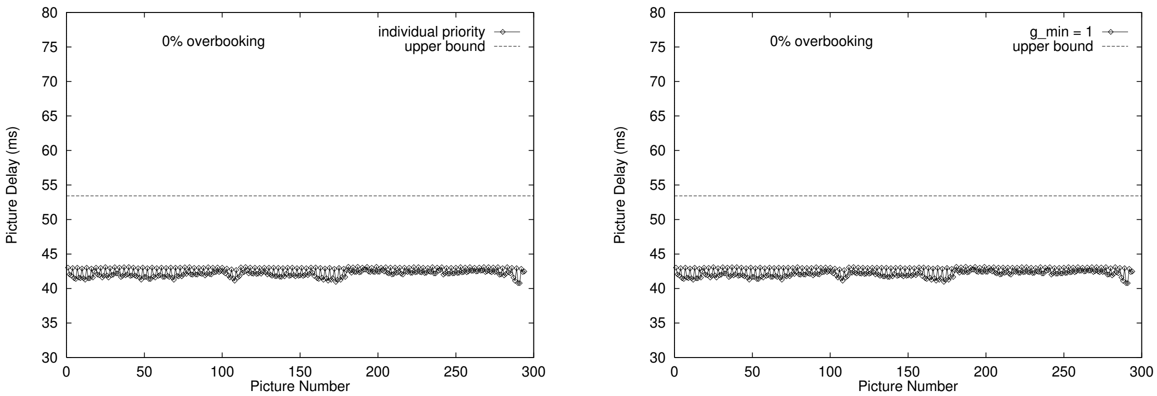

Figure 2: End-to-end picture delays of Energizer sequence

We ran each experiment for 10 seconds of simulated time. About 300 pictures were delivered for each video flow. Three of the MPEG sequences were not long enough, and their traces were wrapped around.

5.2 End-to-end picture delays

Since the reserved rate of a flow changes from burst to burst, it is possible that the largest pictures of all video flows are served by L2 (or L3) at the same time. From Table 1, the sum of λ_{max} over all twelve video flows is 49.77 Mbps. Since only 0.8 of the channel capacity C is allocated to video flows, to ensure that the channel capacity of L2 (L3) is not exceeded, we must have $C = 62.21$ Mbps. We refer to this case as 0% overbooking, which can be implemented by admitting a flow at connection setup time only if the flow is allocated, by each server on its path, a rate equal to its peak rate.

Clearly, with 0% overbooking, the capacity of every channel in the path of a video flow will not be exceeded by the aggregate rate allocated to active flows. The end-to-end delays of bursts in each video flow must be less than the uniform upper bound in (29) plus 1/30 second. This bound holds for individual priority and for group priority with $g_{min} = 1$.

The end-to-end delays of pictures in the Energizer sequence are shown in Figure 2, as well as the upper bound. The utilization of channel L2 (also L3) is about 42%, which includes utilization due to ABR traffic.

5.3 Overbooking to increase utilization

In integrated-services networks, we believe that it is desirable to use the same packet scheduling algorithm for all service classes, with and without overbooking. Flows admitted into a service class with overbooking would receive a real-time delivery

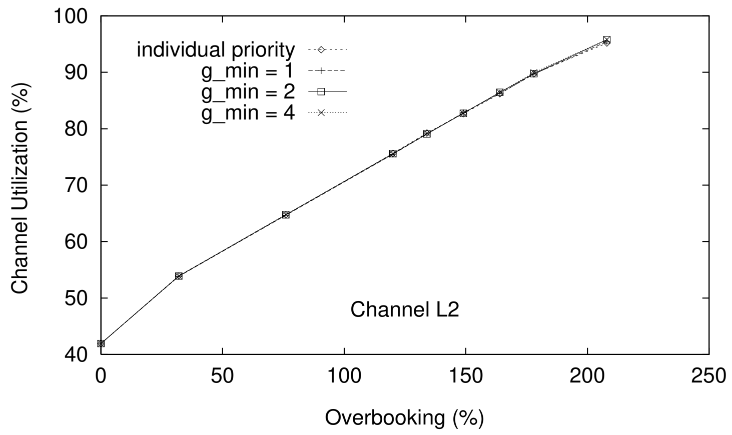

Figure 3: Channel utilization versus overbooking

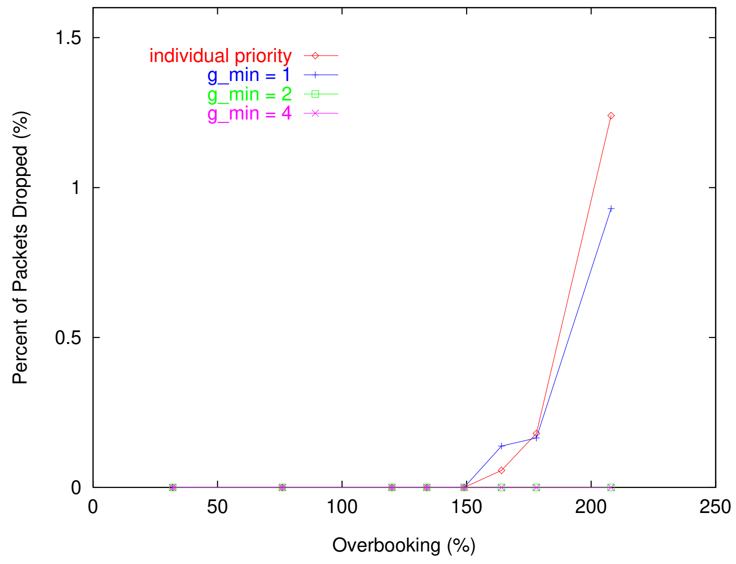

Figure 4: Packet loss rate versus overbooking

service at a specified loss rate (i.e., some bursts may be discarded) or only a best-effort service.

We designed a set of experiments to evaluate network performance when the channel capacity at L2 and L3 is intentionally overbooked. In the experiments described below, overbooking was achieved not by increasing the number of video flows, but using a value of C smaller than 62.21 Mbps. If C is chosen for an experiment such

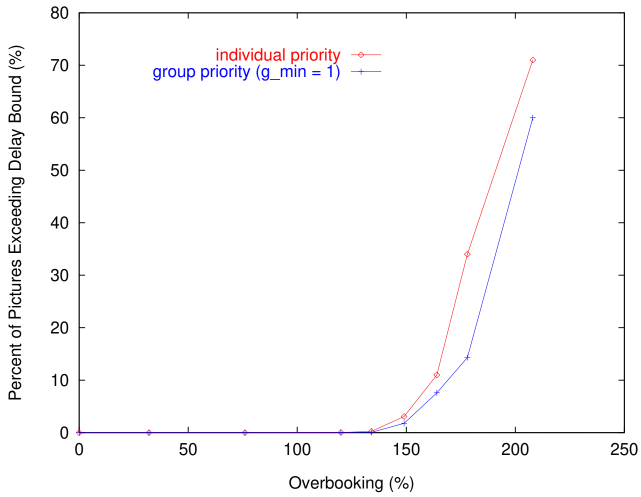

Figure 5: Impact of overbooking on delay bound

that 49.77 Mbps *exceeds* $0.8C$ by $n\%$, the channels in the experiment are said to be $n\%$ overbooked. The experiment is referred to as *$n\%$ overbooking*.

5.4 Channel utilization and loss rate

The objective of overbooking is to increase channel utilization. We performed a series of experiments from 32% to 208% overbooking. Figure 3 shows that the utilization of channel L2 increases almost linearly with overbooking. Three cases of group priority were investigated, for g_{min} equal to 1, 2, and 4. At 208% overbooking, the channel utilization was 0.958 for group priority with $g_{min} = 2$ and 0.952 for individual priority. The utilization for individual priority was smaller because some packets were dropped (due to buffer overflow).

In Figure 4, we show the percentage of packets dropped due to buffer overflow at L2 which has space for 1200 packets shared by all twelve video flows. Note that the loss rate was zero for group priority with g_{min} equal to 2 or 4. It was fairly low for the other two cases, considering that the channel utilization exceeded 0.95.

5.5 Impact on delay bound

We measured the sum of reserved rates of *active* flows as a function of time, and compared it with the channel capacity of L2 (L3). For the experiments at 32% overbooking, at no time was the channel capacity exceeded by the aggregate reserved rate of active flows. But for experiments at 76% overbooking, and higher, the channel capacity was exceeded frequently.

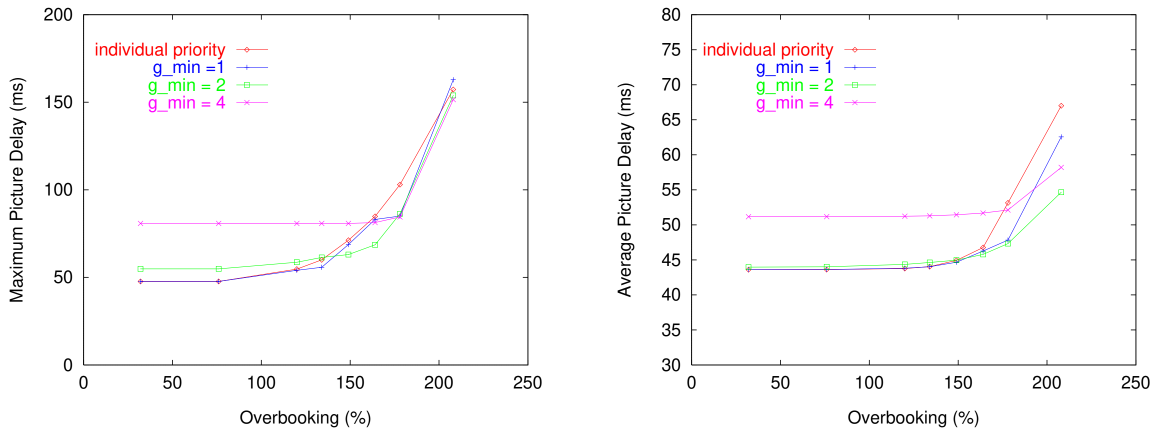

Figure 6: End-to-end picture delays versus overbooking

The delays of individual pictures (bursts) were measured and compared to the upper bound given by (26) for each video sequence. The results were plotted in Figure 5 for two cases: (i) individual priority and (ii) group priority with $g_{min} = 1$. These two cases have the same delay upper bound for each video sequence, determined by the slowest picture in the sequence.

As illustrated in Figure 5, the delay bounds held for all pictures in all video sequences up to 120% overbooking. At 134% overbooking, the delays of a small number of pictures (less than 1%) exceeded their bounds. In all experiments, the fraction of pictures violating their delay bounds was smaller for group priority (with $g_{min} = 1$) than for individual priority.

5.6 Statistical delay performance

In Figure 6, we show end-to-end picture delay versus overbooking. Specifically, we show the maximum and average delays over all pictures in all video sequences. Note that the delay performance of group priority with $g_{min} = 1$ was slightly better than individual priority in all experiments (except for the maximum delay at 208% overbooking). Group priority with g_{min} equal to 2 and 4 performed better than individual priority only when the network was heavily loaded.

In Figure 7, we show the end-to-end picture delays of the Energizer video sequence for individual priority and the three cases of group priority. At 164% overbooking, all three cases of group priority had better performance than individual priority.

5.7 Queue sizes

In Figure 8, we show the maximum and average video queue length at L2 versus overbooking, where *video queue length* denotes the aggregate size of all twelve video flow queues. In Figure 9, we show the video queue length at L2 as a function of

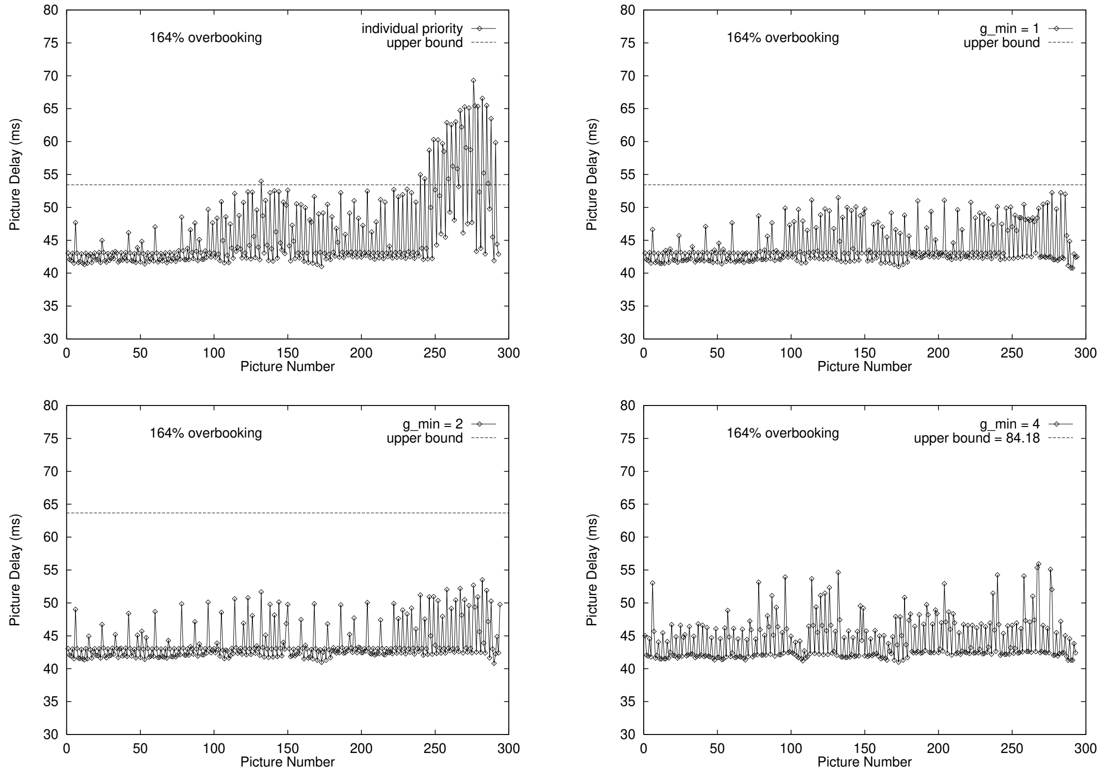

Figure 7: End-to-end picture delays of Energizer sequence

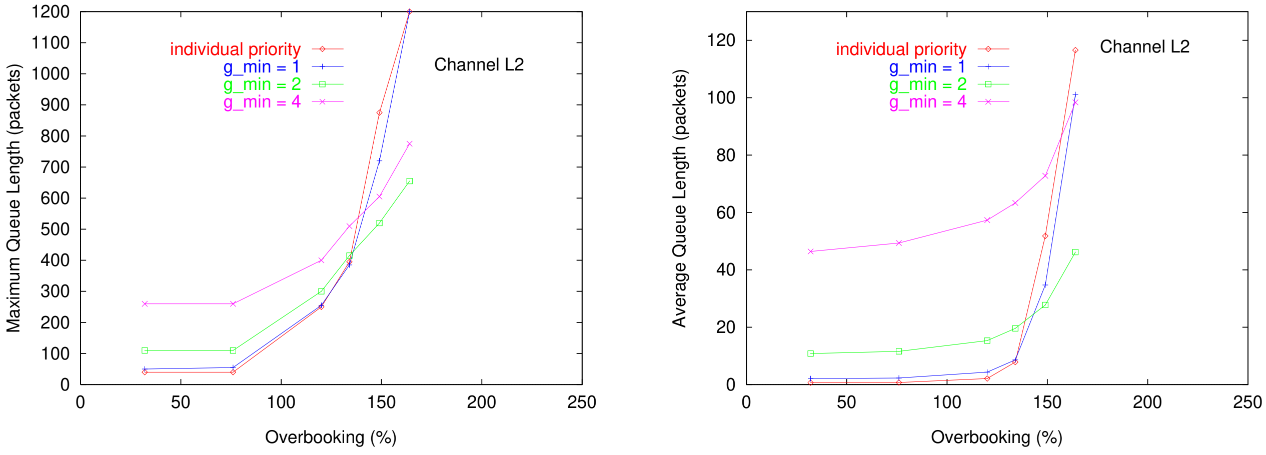

Figure 8: Video queue length versus overbooking

time. At 164% overbooking, the cases of group priority with g_{min} =2 and 4 clearly performed much better than individual priority and group priority with $g_{min} = 1$.

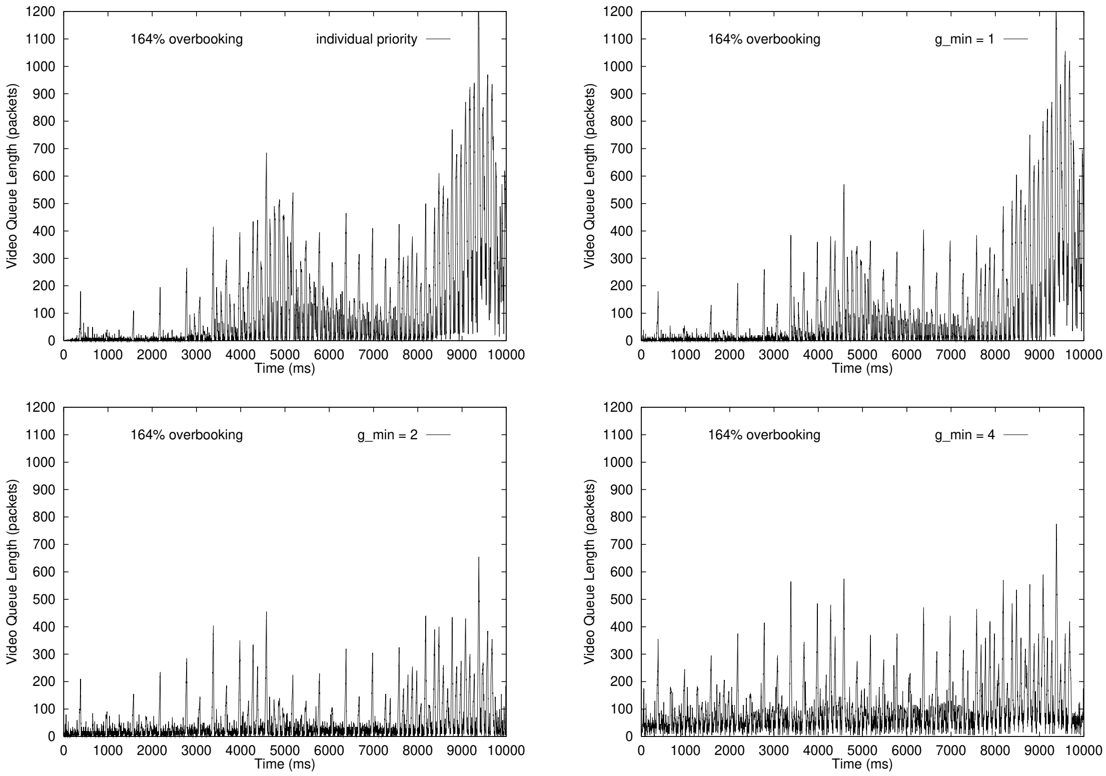

Figure 9: Video queue length at L2 over time

6 Conclusions

We introduced the class of GD servers and proved an end-to-end delay guarantee theorem. The theorem can be instantiated to obtain end-to-end delay bounds for a variety of source control mechanisms and GD servers; in particular, different GD servers can be used in the same end-to-end path. With the theorem, the problem of deriving an end-to-end delay upper bound for a real-time flow is reduced to a set of single-node problems.

We then introduced and developed the idea of group priority. We proved a relaxed deadline theorem for the priority subclass of GD servers. The delay guarantee theorem is then specialized to a smaller subclass of GD servers that use group priority.

We worked out a detailed example for the class of burst scheduling networks [9]. We derived end-to-end delay bounds for bursts (messages), and illustrated how to choose group sizes such that the end-to-end delays of bursts are unaffected by the use of group priority.

Group priority scheduling has two advantages. First the priority of a flow changes less frequently, i.e., from one group to the next rather than from one packet to the next. Hence, the packet scheduler's work in updating its priority data structure (e.g., a heap) would be much reduced for large groups. (An empirical investigation quantifying this reduction can be found in [12].) Second, servers that use group priority have more flexible deadlines; consequently, their packet schedulers can better cope with temporary overloads.

References

[1] Alan Demers, Srinivasan Keshav, and Scott Shenker. Analysis and simulation of a fair queuing algorithm. In *Proceedings of ACM SIGCOMM '89*, pages 3–12, August 1989.

[2] Domenico Ferrari and Dinesh Verma. A scheme for real-time channel establishment in wide-area networks. *IEEE Journal on Selected Areas in Communications*, pages 368–379, April 1990.

[3] Norival R. Figueira and Joseph Pasquale. Leave-in-time: A new service discipline for real-time communications in a packet-switching network. In *Proceedings of ACM SIGCOMM '95*, pages 207–218, August 1995.

[4] S. Jamaloddin Golestani. A self-clocked fair queueing scheme for high speed applications. In *Proceedings of IEEE INFOCOM '94*, pages 636–646, March 1994.

[5] Pawan Goyal, Simon S. Lam, and Harrick Vin. Determining end-to-end delay bounds in heterogeneous networks. In *Proceedings of NOSSDAV*, Durham, NH, April 1995.

[6] ITU-T Rec. I.371. *Traffic Control and Congestion Control in B-ISDN.* Perth, U.K., Nov. 6-14, 1995.

[7] Srinivasan Keshav. On the efficient implementation of fair queueing. *Journal of Internetworking Research and Experience*, 1991.

[8] Simon S. Lam, Simon Chow, and David K.Y. Yau. An algorithm for lossless smoothing of MPEG video. In *Proceedings of ACM SIGCOMM '94*, pages 281–293, London, England, August 1994.

[9] Simon S. Lam and Geoffrey G. Xie. Burst scheduling networks. Technical Report TR-94-20, University of Texas at Austin, Austin, Texas, July 1994. Revised, August 31, 1996. Available from http://www.cs.utexas.edu/users/lam/NRL/ . An abbreviated version in *Proceedings of IEEE INFOCOM '95*, April 1995.

[10] Abhay K. Parekh and Robert G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single node case. *IEEE/ACM Trans. on Networking*, 1(3):344–357, June 1993.

[11] Geoffrey G. Xie and Simon S. Lam. Delay guarantee of Virtual Clock server. *IEEE/ACM Trans. on Networking*, 3(6):683–689, December 1995. Presented at *9th IEEE Workshop on Computer Communications*, October 1994.

[12] Geoffrey G. Xie and Simon S. Lam. An efficient adaptive search algorithm for scheduling real-time traffic. Technical Report TR-95-29, University of Texas at Austin, Austin, Texas, July 1995. To appear in *Proceedings of IEEE ICNP '96, Oct. 1996.*

[13] Geoffrey G. Xie and Simon S. Lam. Real-time block transfer under a link sharing hierarchy. Technical Report TR-96-19, University of Texas at Austin, Austin, Texas, June 1996. To appear in *Proceedings IEEE INFOCOM '97*.

[14] Hui Zhang and Srinivasan Keshav. Comparison of rate-based service disciplines. In *Proceedings of ACM SIGCOMM '91*, pages 113–121, August 1991.

[15] Lixia Zhang. VirtualClock: A new traffic control algorithm for packet switching networks. In *Proceedings of ACM SIGCOMM '90*, pages 19–29, August 1990.

Appendix

Proof of Lemma 2. We use induction on i.
Base case. From (3), for $i = 1$,

$$\begin{aligned}
V_{k+1}^f(1) &= A_{k+1}^f(1) + v^f(1) && (35)\\
&\quad \{by\ Lemma\ 1\} \\
&\leq P_k^f(1) + \alpha_k + v^f(1) && (36)\\
&= V_k^f(1) + v^f(1) + (P_k^f(1) - V_k^f(1)) + \alpha_k && (37)
\end{aligned}$$

Thus the inequality in (4) holds for $i = 1$.

Inductive step. Assume that the inequality in (4) holds for $1 \leq i \leq n$. A proof that the inequality holds for $i = n + 1$ follows.

From (3), for $i = n + 1$,

$$V_{k+1}^f(n+1) = \max\{V_{k+1}^f(n), A_{k+1}^f(n+1)\} + v^f(n+1) \tag{38}$$

- *Case 1.* $V_{k+1}^f(n) \geq A_{k+1}^f(n+1)$:

$$\begin{aligned}
V_{k+1}^f(n+1) &= V_{k+1}^f(n) + v^f(n+1) && (39)\\
&\quad \{by\ induction\ hypothesis\} \\
&\leq V_k^f(n) + \max_{1\leq j\leq n}\{v^f(j) + (P_k^f(j) - V_k^f(j))\} + \alpha_k + v^f(n+1) && (40)\\
&\quad \{by\ (3)\ and\ FIFO\ property\} \\
&\leq V_k^f(n+1) + \max_{1\leq j\leq n}\{v^f(j) + (P_k^f(j) - V_k^f(j))\} + \alpha_k && (41)\\
&\leq V_k^f(n+1) + \max_{1\leq j\leq n+1}\{v^f(j) + (P_k^f(j) - V_k^f(j))\} + \alpha_k && (42)
\end{aligned}$$

- *Case 2.* $V^f_{k+1}(n) < A^f_{k+1}(n+1)$:

$$\begin{aligned} V^f_{k+1}(n+1) &= A^f_{k+1}(n+1) + v^f(n+1) && (43)\\ & \quad \{\textit{by Lemma } 1\} \\ &\le P^f_k(n+1) + \alpha_k + v^f(n+1) && (44)\\ &= V^f_k(n+1) + \alpha_k + v^f(n+1) + (P^f_k(n+1) - V^f_k(n+1)) && (45)\\ &\le V^f_k(n+1) + \max_{1\le j\le n+1}\{v^f(j) + (P^f_k(j) - V^f_k(j))\} + \alpha_k && (46) \end{aligned}$$

□

Proof of Theorem 1. From Lemma 1,

$$\begin{aligned} A^f_{K+1}(i) &\le P^f_K(i) + \alpha_K && (47)\\ &= V^f_K(i) + (P^f_K(i) - V^f_K(i)) + \alpha_K && (48)\\ & \quad \{\textit{applying Lemma } 2\} \\ &\le V^f_{K-1}(i) + \max_{1\le j\le i}\{v^f(j) + (P^f_{K-1}(j) - V^f_{K-1}(j))\} + \alpha_{K-1} + \\ & \quad (P^f_K(i) - V^f_K(i)) + \alpha_K && (49)\\ & \quad \{\textit{applying Lemma } 2 \textit{ repeatedly}\} \\ &\le V^f_1(i) + \sum_{k=1}^{K-1} \max_{1\le j\le i}\{v^f(j) + (P^f_k(j) - V^f_k(j))\} + \\ & \quad (P^f_K(i) - V^f_K(i)) + \sum_{k=1}^{K}\alpha_k && (50) \end{aligned}$$

□

Proof of Theorem 2. The arrival times and sizes of packets are the same in both systems. Since the server is work conserving, each busy period begins and ends at the same time in both systems. Without any loss of generality, it suffices to consider an arbitrary busy period. Each packet within the busy period is identified by its index m in the sequence of departures *in the original system*. The proof is by induction on the number of packets with relaxed deadlines in the modified system for the busy period.

Base case. The modified system has exactly one packet in the busy period, say m_1, with a relaxed deadline, i.e., $P_k(m_1) < P'_k(m_1)$, and $\forall m \neq m_1$, $P'_k(m) = P_k(m)$.

If packets in the modified system depart in the same order as they depart in the original system, (16) in Theorem 2 holds because for all m, from work-conserving and nonpreemptive assumptions,

$$L'_k(m) = L_k(m) \le P_k(m) + \beta_k \le P'_k(m) + \beta_k \qquad (51)$$

Otherwise, there is a reordering of the departure sequence in the modified system because packet m_1 has a relaxed deadline. For the busy period, let $< 1, 2, \ldots, m_1 - 1, m_1, \ldots, m_2, \ldots, b >$ denote a prefix of the departure sequence in the original system, such that $1 \leq m_1 < m_1+1 \leq m_2 \leq b$. That is, m_2 is a packet served behind m_1 in the original system; however, due to the relaxed deadline of m_1, m_1 is served immediately after m_2 in the modified system.

For all $m \neq m_1$, we have (from work-conserving and nonpreemptive assumptions)

$$L'_k(m) \leq L_k(m) \leq P_k(m) + \beta_k = P'_k(m) + \beta_k \tag{52}$$

For packet m_1, we have (from work-conserving and nonpreemptive assumptions)

$$L'_k(m_1) = L'_k(m_2) + \frac{s(m_1)}{C_k} = L_k(m_2) \leq P_k(m_2) + \beta_k = P'_k(m_2) + \beta_k \leq P'_k(m_1) + \beta_k \tag{53}$$

The last inequality follows from $P'_k(m_2) \leq P'_k(m_1)$, which is proved as follows. There are two possible cases. First, m_1 is the first packet served in the original system for this busy period. In the modified system, m_2 can be served ahead of m_1 only if the two packets arrive at the same time and $P'_k(m_2) \leq P'_k(m_1)$.

The second case is for $1 < m_1 < b$, which is proved by contradiction as follows. Suppose $P'_k(m_2) > P'_k(m_1)$. Since m_2 is served ahead of m_1 in the modified system, m_1 must have arrived too late to be selected ahead of m_2 on the basis of a smaller deadline, i.e.,

$$A_k(m_1) > L'_k(m_2^{pre}) \tag{54}$$

where m_2^{pre} denotes the packet served immediately ahead of m_2 in the modified system. There are two possibilities for the identity of packet m_2^{pre}:

(i) m_2^{pre} is $m_1 - 1$, the packet served ahead of m_1 in the original system, in which case, $L'_k(m_2^{pre}) = L_k(m_1 - 1)$, and

(ii) $m_2^{pre} \geq m_1 + 1$, in which case,

$$L'_k(m_2^{pre}) \geq L'_k(m_1 + 1) = L_k(m_1 + 1) - (s(m_1)/C_k) \geq L_k(m_1 - 1) \tag{55}$$

In either case, we have

$$A_k(m_1) > L'_k(m_2^{pre}) \geq L_k(m_1 - 1) \tag{56}$$

Since the inequality, $A_k(m_1) > L_k(m_1 - 1)$, contradicts the assumption that m_1 is served immediately after $m_1 - 1$ in the original system (for a work conserving server), we have proved $P'_k(m_2) \leq P'_k(m_1)$. Thus $L'_k(m_1) \leq P'_k(m_1) + \beta_k$ follows from (53).

Inductive step. Assume that (16) in Theorem 2 holds for the modified system with n packets in the busy period having relaxed deadlines. To show that (16) in Theorem 2 holds for the modified system with $n+1$ packets in the busy period having relaxed deadlines, simply redefine the modified system with n packets in the busy period having relaxed deadlines to be another original system, and then apply the proof for the base step. □