Real-Time Block Transfer Under a Link-Sharing Hierarchy

Geoffrey G. Xie, *Member, IEEE*, and Simon S. Lam, *Fellow, IEEE*

***Abstract*— Most application data units are too large to be carried in a single packet (or cell) and must be segmented for network delivery. To an application, the end-to-end delays and loss rate of its data units are much more relevant performance measures than ones specified for individual packets (or cells). The concept of a burst (or block) was introduced to represent a sequence of packets (or cells) that carry an application data unit. In this paper, we describe how a real-time variable bit-rate (VBR) service, with quality of service (QoS) parameters for block transfer delay and block loss rate, can be provided by integrating concepts and delay guarantee results from our previous work on burst scheduling, together with ideas from asynchronous transfer mode (ATM) block transfer. Two new contributions are presented herein. First, we design an admission control algorithm to provide the following two classes of service: bounded-delay block transfer with no loss, and bounded-delay block transfer at a specified block loss rate. Secondly, we show how to extend existing end-to-end delay bounds to networks with hierarchical link sharing.**

***Index Terms*— Admission control, ATM block transfer, burst scheduling, delay guarantee, hierarchical link sharing.**

I. Introduction

In all packet switching networks, packets have a maximum size (in number of bits). 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 are more relevant performance measures than ones specified for individual packets. For example, consider an application that sends live video over an asynchronous transfer mode (ATM) network. Each picture is segmented into a sequence of cells at the sender. Clearly the delays incurred to deliver whole pictures are much more important to the performance of the application than the delays of individual cells. From this observation, we introduced the concept of a burst to represent a sequence of packets that carry an application data unit, and designed the class of burst scheduling networks to provide delay guarantees to bursts [16], [17].

The *ATM block transfer* (ABT) capability being standardized by ITU-T is based upon a similar observation [14]. The objective of ABT is to allow a bursty data source to dynamically negotiate its bandwidth reservation on the basis of a block of cells. Note that a higher layer protocol data unit, fragmented into a number of ATM cells, is lost if any one of its cells is lost. Therefore, even a low cell loss rate can cause a significant loss rate for the higher layer protocol. As a result, the higher layer protocol's throughput may be much less than the protocol session's throughput measured in delivered cells. The concept of a block was introduced to represent a sequence of cells, which may contain a single data unit or multiple data units for the higher layer protocol. A block is bracketed by two resource management (RM) cells. A leading RM cell requests a reserved bandwidth for the block, and a trailing RM cell releases the reserved bandwidth. Cells are handled in blocks by a switch. In particular, a block of cells is either discarded or accepted entirely. (This is similar to the idea of early packet discard proposed in recent studies on IP over ATM [22], [25].)

For the ABT service, the concept of cell loss rate can be generalized to *block loss rate*. Such a generalization is backward-compatible with the existing ATM traffic management (TM) 4.0 service architecture [10] since a block is a sequence of cells, with a single cell being a special case. Similarly, the concept of cell transfer delay for real-time variable bit-rate (VBR) services can be generalized to *block transfer delay* which, we believe, is a more relevant performance measure to many applications; for example, if every picture in a video sequence is carried by a block of cells, then the block transfer delays are the same as picture delays.

In this paper, we describe how a real-time VBR service, called *real-time block transfer*, with quality of service (QoS) parameters for block transfer delay and block loss rate can be provided by integrating concepts and delay guarantee results from our previous work on burst scheduling [16] and group priority [17], together with ideas from ATM block transfer. In particular, we present an admission control algorithm for real-time block transfer that supports the following two service classes: bounded-delay block transfer with no loss (i.e. deterministic delay guarantee), and bounded-delay block transfer at a specified block loss rate. The algorithm has been evaluated using a simulator driven by MPEG video traces.

It is envisioned that future integrated services networks will support not only link sharing by multiple service classes (real-time service, best-effort service, etc.) but also by multiple

Manuscript received January 5, 1997; revised September 5, 1997; approved by IEEE/ACM Transactions on Networking Editor J. Crowcroft. Manuscript originally submitted to IEEE Journal on Selected Areas of Communications and forwarded to IEEE/ACM Transactions on Networking by Editor J. Crowcroft. This work was supported in part by the National Science Foundation under Grant NCR-9506048 and in part by the Texas Advanced Research Program under Grant 003658-220. The work of G. G. Xie was supported by an Intel Graduate Fellowship received while he was with the University of Texas, Austin. This paper was presented in part at the IEEE INFOCOM '97, 1997.

G. G. Xie is with the Department of Computer Science, Naval Postgraduate School, Monterey, CA 93943 USA (e-mail: xie@cs.nps.navy.mil).

S. S. Lam is with the Department of Computer Sciences, University of Texas, Austin, TX 78712-1188 USA.

Publisher Item Identifier S 1063-6692(98)01450-2.

administrative classes (different agencies and organizations) [9]. Specifically, packets (or cells) from sessions belonging to different service classes and administrative classes interact with one another when they are statistically multiplexed at an output link of a switch. The link's packet scheduling algorithm plays an important role in controlling such link sharing. Hierarchical link sharing has been proposed as a solution [1]. In this paper, we also describe a general approach for extending the end-to-end delay bounds for bursts[1] from networks in which links are shared by service classes only [16], [17] to networks in which links are hierarchically shared (e.g., by administrative classes first, and then by service classes within each administrative class). Specifically, each logical server in a link-sharing hierarchy is modeled as a fluctuation constrained (FC) server [18]. Two theorems are presented. They can be used to derive block delay guarantees of FC servers based on existing delay guarantee results of constant-rate servers. The theorems are general; they are proved for a large class of well-known servers.

The balance of this paper is organized as follows. In Section II, the basic idea of block-based admission control for ABT and the concept of burst scheduling are described. The end-to-end delay bounds of burst scheduling networks are shown. In Section III, an admission control algorithm for real-time block transfer services is presented. Experimental results from using the algorithm are discussed. In Section IV, hierarchical link sharing is introduced. Also described is a general approach for extending end-to-end delay bounds to networks with hierarchical link sharing.

II. Real-Time Block Transfer

A. Block-Based Admission Control

In ABT,[2] dynamic bandwidth reservation and allocation for a block of cells can be carried out in two ways: 1) ABT with delayed transmission (ABT/DT) and 2) ABT with immediate transmission (ABT/IT). For our discussion, the focus is on the latter. In ABT/IT, the block is sent *immediately* after a preceding RM cell, which contains a request for a cell rate and a cell delay variance. The block proceeds on a switch-by-switch basis, with each switch either forwarding the block with guaranteed QoS for *every cell* in the block or discarding the *entire* block if a required resource such as bandwidth is not available. In other words, the switches perform admission control on a block-by-block basis.

With block-based admission control, cell losses are concentrated over a small number of blocks, and bandwidth is not wasted on delivery of partial blocks. Therefore, ABT is able to avoid the situation in which cell losses spread over a large number of higher layer data units, causing throughput degradation of such data units. With block-based admission control, ABT is also able to offer QoS measured in terms of blocks. In particular, the concept of cell loss rate can be generalized to *block loss rate*.

[1]End-to-end delay bounds for blocks are obtained by specifying a burst [16], [17] to represent a block of ATM cells.

[2]A short overview of ABT can be found in [2].

B. Burst Scheduling

For burst scheduling networks [16], we model a flow as a sequence of bursts, each of which models a sequence of packets that carries an application-specific data unit. A burst corresponds to a block in ABT with some minor differences in detail. In particular, instead of two special packets being used, the first packet of each burst is marked and stored in it is information on the size of the burst (in number of packets) and the average rate of the burst. Moreover, for efficient packet scheduling and delay jitter control, packets of each burst satisfy a jitter timing constraint [16].

The following delay bound results are taken from [16] and [17].

End-to-End Delay Bounds for Burst Scheduling Networks: Consider a flow that traverses a sequence of nodes, indexed by $0, 1, 2, \cdots, K+1$, where node 0 denotes the source, node $K+1$ the destination, and the other nodes packet switches. If the capacity of every link on the network path is not exceeded, then the end-to-end delay[3] of every burst m of the flow, denoted by $D_m, m = 1, 2, \cdots$, has the following upper and lower bounds:

$$D_m \le \frac{b_m + 1}{\lambda_m} + (K-1) \max_{1 \le n \le m} \left\{ \frac{1}{\lambda_n} \right\} + \sum_{k=1}^{K} \alpha_k \quad (1)$$

$$D_m \ge (K-1) \frac{1}{\lambda_m} + \sum_{k=1}^{K} \alpha_k \quad (2)$$

where

b_m	size of burst m (packets);
λ_m	average rate of burst m (packets/second);
α_k	small constant associated with the link from switch k to $k+1$.

While our design of burst scheduling networks consists of specialized components such as source and flow regulators, the concept of burst scheduling, i.e., modeling a flow as a sequence of bursts and providing bounded delays to bursts (blocks) when link capacity is not exceeded, is quite general. It can be realized by many designs other than ours. In the balance of this paper we refer to burst scheduling as a general concept independent of the design details.

C. Integration of ABT and Burst Scheduling

ABT is able to minimize block losses through the use of block-based admission control and provide low block loss rate as QoS. Burst scheduling networks provide bounded block transfer delays as QoS when link capacity is not exceeded. Integrating the concepts and results from ABT and burst scheduling, we define a real-time VBR service, called real-time block transfer, that provides the following two classes of services: 1) bounded-delay block transfer with zero block loss and 2) bounded-delay block transfer at a specified block loss rate.

Admission control at the flow level is the key for our integration. For class 1, peak rate reservation can be used for

[3]Measured from the time when the first cell enters the network to the time when the last cell leaves the network.

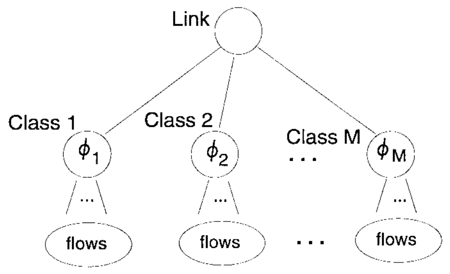

Fig. 1. System model for admission control.

admission control to ensure that the link capacity allocated to this class is not exceeded without discarding blocks. For class 2, overbooking of the class's allocated capacity is allowed at the time of connection setup while block-based admission control is used to ensure that link capacity is not exceeded at any time by discarding blocks if necessary. To limit the block loss rate to a specified value, the extent of overbooking is controlled by flow level admission control. In next section, we will describe in detail a flow level admission control algorithm that we have designed for real-time block transfer services.[4]

III. Admission Control

Consider a flow whose source requests a real-time block transfer service. We assume that at the time of connection setup, the source supplies the network two sets of flow parameters: 1) QoS parameters: block loss probability (BLP) and block transfer delay bound (BTD)[5] and 2) traffic parameters—sustained cell rate (SCR), peak cell rate (PCR), and cell rate variation (CRV). Whether or not to admit the flow is a decision made by each switch in the path of the flow. Specifically, each switch in the path accepts the flow only if doing so will not cause violation of QoS guarantees to accepted flows; the network admits the flow only if all switches in the path accept the flow.

A. System Model

Our system model for admission control by a particular switch is shown in Fig. 1. There are M classes of real-time block transfer service. They share a link[6] with capacity C b/s. Each class $s, s = 1, 2, \cdots, M$ is associated with a weight ϕ_s, which is a relative measure of the class's allocated share of the link bandwidth. For ease of presentation and without loss of generality, we assume that $\Sigma_{s=1}^{M} \phi_s = 1$. Therefore, class s has a share of the link equal to $r_s = \phi_s \cdot C$ (b/s). Each class offers a target block loss rate denoted by p. Without loss of generality, we assume that

$$0 = p_1 < p_2 < \cdots < p_M < 1.$$

In other words, class 1 provides deterministic service, i.e., class 1 service defined in Section II-C, and the other classes provide different levels of statistical service, i.e., class 2 service defined in Section II-C.

[4]The algorithm will be presented in the context of ATM networks. Our design should be applicable to other types of networks.

[5]BTD will not be considered further in our design of admission control, assuming that an appropriate burst scheduling algorithm is used by the network to ensure block delay guarantees.

[6]The link could be a logical one; see Section IV.

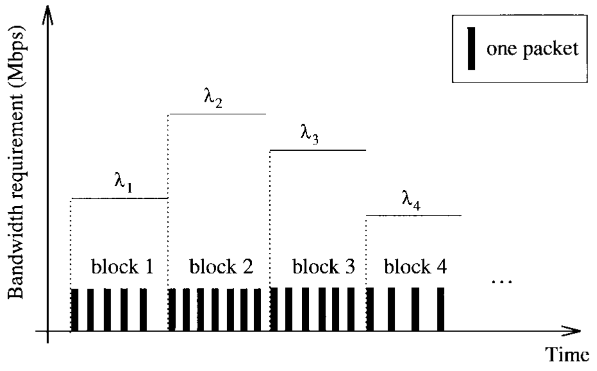

Fig. 2. A flow modeled by blocks.

We also assume that an appropriate scheduling algorithm (such as $\mathrm{WF}^2\mathrm{Q}+$ [1]) is used by the link to provide a firewall between the service classes and guarantee each class its link share (see Section IV). As a result, admission control for each class can be performed independently.

B. Preliminary

In addition to the system model, there are a couple of factors (or implicit assumptions) that are key to understanding our flow level admission control algorithm. They are discussed next.

1) Traffic Model: In what follows, a flow is modeled as a sequence of blocks, each of which models an application data unit. The first and last packet of each block is marked, and the first packet carries the bandwidth requirement (i.e., the average rate) of the block. Fig. 2 illustrates a flow modeled by blocks. (λ_m denotes the bandwidth requirement of block m.) Note that unlike our burst flow specification [16], interpacket spacing within a block is not specified in this traffic model. This is because such spacing, while having a direct impact on block delays and block delay jitter, does not significantly affect admission control.

2) Algorithm Specification of Block-Based Admission Control: Block-based admission control by a service class is formally specified below. Let r be the link capacity allocated to the service class. The variable A is used to store the aggregate rate allocated by the service class, and is initialized to 0.

- Upon arrival of first packet of block m

```
        $Block\_Admission\_Control\ (m)$
1   if $(A + \lambda_m > r)$
2       then   discard block $m$;
3       else   admit block $m$;
4              $A := A + \lambda_m$;
```

- upon departure of last packet of block m

$$A := A - \lambda_m.$$

The above algorithm has extremely low processing cost. Furthermore, it is performed only once per block. Note that the average interblock arrival time is usually much larger than the average interpacket arrival time. (This is especially true in an ATM network.) Therefore, the algorithm is suitable for high speed networks.

C. Derivation of Flow Level Admission Control Conditions

For flows requesting the deterministic service, admission control should be based on peak rate reservation so that the class's link share would never be exceeded and thus no block would be discarded.[7]

Next consider a statistical class s. Flows that request this class of service can tolerate some block losses. Therefore, the admission control for them can be more aggressive to increase utilization of the class's bandwidth by taking advantage of statistical multiplexing.

Specifically with block-based admission control, the objective of the flow level admission control becomes very specific: to allow as much overbooking as possible while bounding the probability that the class's unallocated capacity is not sufficient for a newly arrived block—denoted as the overflow probability of the class—by p_s. The overflow probability and statistical multiplexing gains are closely related. In particular, if two classes have the same overflow probability, the utilization is higher, because of statistical multiplexing gains, for the one that has a larger capacity and is shared by more flows. In what follows, we derive a set of conditions that are sufficient to limit the overflow probability of class s at approximately p_s. The goal is to find, among these conditions, one that is both accurate and easy to check.

Assume that the service of the class is currently shared by a set of n flows (indexed by $1, 2, \cdots, n$). Let $\mathrm{BLP_i}$, $\mathrm{SCR_i}$, $\mathrm{PCR_i}$, and $\mathrm{CRV_i}$ be the QoS and traffic parameters[8] that are supplied to the switch by flow i. At any time, with block-based admission control, at most one block from each flow has its reserved rate allocated. Denote $\lambda_i(t)$ the reserved cell rate for the block of flow i that is either allocated a rate or being processed by block-based admission control at time t. ($\lambda_i(t) = 0$ if there is no such block.) In our analysis below, $\lambda_i(t)$'s are considered as independent random variables. Define

$$Y_s(t) = \sum_{i=1}^{n} \lambda_i(t) - r_s. \tag{3}$$

Consider a block that is being processed by block-based admission control at time t. It will be discarded if $(Y_s(t) > 0)$. Therefore, the goal is to find conditions sufficient for

$$\Pr(Y_s(t) > 0) \le p_s. \tag{4}$$

From a generalized version of the central limit theorem [19] (included in the Appendix as Theorem 3), we have

$$\frac{Y_s(t) - E[Y_s(t)]}{\sqrt{\mathrm{var}[Y_s(t)]}} \to N(0,1) \tag{5}$$

as $n \to \infty$, where $N(0,1)$ is the standard normal distribution. Therefore, we can approximate $(\Pr(Y_s(t) > 0) \le p_s)$ by

$$\Pr\left(X > \frac{-E[Y_s(t)]}{\sqrt{\mathrm{var}[Y_s(t)]}}\right) \le p_s \tag{6}$$

where $X \sim N(0,1)$.

The following condition is sufficient for (6):

$$\frac{-E[Y_s(t)]}{\sqrt{\mathrm{var}[Y_s(t)]}} \ge Z_s \tag{7}$$

where Z_s is the constant that satisfies $(\Pr(X > Z_s) = p_s)$. Define

$$Z = \frac{-E[Y(t)]}{\sqrt{\mathrm{var}[Y(t)]}}. \tag{8}$$

We have

$$E[Y_s(t)] = \sum_{i=1}^{n} E[\lambda_i(t)] - r_s \tag{9}$$

$$\mathrm{var}[Y_s(t)] = \sum_{i=1}^{n} \mathrm{var}[\lambda_i(t)]. \tag{10}$$

Therefore

$$Z = \frac{r_s - \sum_{i=1}^{n} E[\lambda_i(t)]}{\sqrt{\sum_{i=1}^{n} \mathrm{var}[\lambda_i(t)]}}. \tag{11}$$

Combining (7) and (9)–(11), we have the following sufficient condition for (4):

$$\frac{Z_s}{Z} \le 1. \tag{12}$$

We refer to the value of (Z_s/Z) as the statistical multiplexing intensity (SMI) of class s. It should never exceed the threshold of one to bound the block rate of the class below p_s. In practice, it is difficult to obtain the exact value of Z. However, Z can be estimated as follows:

$$\hat{Z} = \frac{r_s - \sum_{i=1}^{n} \mathrm{SCR_i}}{\sqrt{\sum_{i=1}^{n} \mathrm{CRV_i}}}. \tag{13}$$

In summary, the following admission control condition can be used for class s:

$$\frac{Z_s}{\hat{Z}} \le 1. \tag{14}$$

Note that the source of flow i may not have a good estimate of $\mathrm{CRV_i}$ at the time of connection setup. In such a case, $\mathrm{CRV_i}$ is upper bounded by $\mathrm{SCR_i} \cdot (\mathrm{PCR_i} - \mathrm{SCR_i})$, which can be used as a pessimistic estimate.[9] (See Theorem 4 in the Appendix.)

[7] If the switch has *a priori* knowledge of the traffic characteristics of flows in the class, peak rate reservation may not be necessary. But it is unrealistic with today's networks.

[8] Note that $\mathrm{BLP_i}$ is the probability of block losses through one switch (i.e. a single hop) required for flow i. We discuss how to distribute an end-to-end block loss requirement to individual switches in a different report [28].

[9] Moreover, online measurement of SCR and CRV can be carried out for admitted flows to improve the admission control accuracy of the algorithm. We are currently investigating such techniques. Note that there have been several recent studies on measurement-based admission control, e.g., [12].

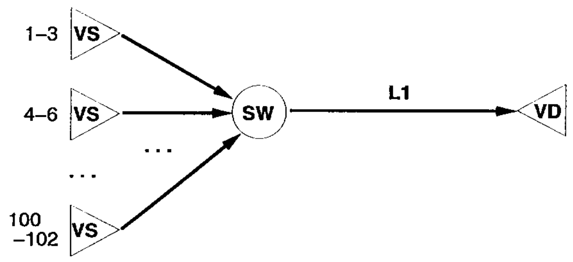

Fig. 3. Simulated network.

D. Algorithm Specification

The following admission control algorithm follows from the analysis in the previous section. The variables B_s and V_s are used to store, respectively, the available bandwidth and the total cell rate variance of class s. Initially, $B_s = r_s$ and $V_s = 0$. We assume that if a source does not have a good estimate of CRV at the time of connection setup, it will let the network know by setting $\mathrm{CRV} = -1$.

```
Flow_Admission_Control (BLP, PCR, SCR, CRV)
1   $s := find\_class(\mathrm{BLP})$;
2   if $(s = 1)$
3       then if $(B_1 - \mathrm{PCR} \geq 0)$
4               then accept the flow;
5                    $B_1 := B_1 - \mathrm{PCR}$;
6               else reject the flow;
7       else if $(\mathrm{CRV} = -1)$
8               then $\mathrm{CRV} := \mathrm{SCR} * (\mathrm{PCR} - \mathrm{SCR})$;
9            $\mathrm{SMI} := Z_s * (\mathrm{sqrt}(V_s + \mathrm{CRV}) / (B_s - \mathrm{SCR}))$;
10           if $(SMI \leq 1)$
11              then accept the flow;
12                   $B_s := B_s - \mathrm{SCR}$;
13                   $V_s := V_s + \mathrm{CRV}$;
14              else reject the flow.
```

The algorithm is straightforward and simple to implement. First, BLP is used to find the service class desired by the source. Assume the class found is s. If s is 1, the source requests the deterministic service. Therefore, the admission decision is based upon PCR of the flow and the bandwidth currently available for class 1. Otherwise, the source requests a statistical service. The admission decision is then based on the projected SMI value of class s.

E. Experimental Results

We have evaluated the admission control algorithm by performing a set of simulation experiments. The simulation configuration is shown in Fig. 3. The nodes labeled by VS denote groups of three video sources, and VD their destination. Each video source generates 53-byte packets from a trace file obtained from an MPEG video sequence and packets for each MPEG frame (or picture) are modeled as a block. A profile of the MPEG sequences used in our experiments is given in

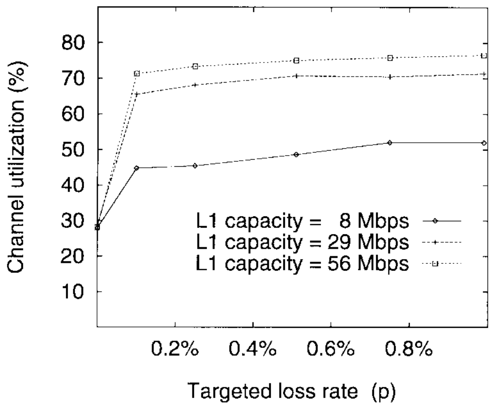

Fig. 4. Channel utilization improvement.

TABLE I
PROFILE OF MPEG SEQUENCES

MPEG sequence	encoding pattern	picture rate (Mb/s) min	ave	max	SCR	CRV
Airwolf	(3,6)	0.14	0.89	3.31	0.89	0.512
Energizer	(3,6)	0.17	0.76	2.34	0.76	0.385
SimpsonsI	(3,6)	0.14	0.92	2.60	0.92	0.482
Hardboiled	(0,1)	0.23	1.07	2.67	1.07	0.136
Terminator	(3,6)	0.14	1.15	3.86	1.15	1.194
Canyon	(3,6)	0.08	0.28	0.70	0.28	0.046
Jamie	(3,6)	0.04	0.71	2.87	0.71	0.515
TheWall3	(1,2)	0.03	1.07	3.39	1.07	0.281
Reds Nightmare	(10,30)	0.89	0.75	3.62	0.75	0.458
UnderSiege	(3,6)	0.17	0.59	2.02	0.59	0.227

Table I. Their durations vary from 10 s to several minutes. Each sequence is used by 10 or 11 video sources.

The admission control algorithm is implemented for channel L1. Each of the video sources makes a reservation with the network, and starts sending packets only after the reservation is successful. In our experiments, all sources requested the same class of statistical service with a target block loss rate of p. At the channel, packets are scheduled based on their virtual clock values [31]. The capacity of L1 as well as the value of p were varied in different experiments. Each experiment was run for 10 s of simulated time.

1) Channel Utilization: In Fig. 4, the channel utilization as a function of the target block (picture) loss rate is plotted. The result shows that the channel is used much more efficiently with a statistical service than a deterministic service (with zero loss rate). The price to pay is a small nonzero picture loss probability. The utilization increase is more significant with a higher channel capacity, from below 30% to above 70% in the case where the capacity of L1 is 56 Mb/s. This is because the improvement is due to statistical multiplexing gains, which are larger with more flows sharing the channel. (Up to 50 flows were accepted when the capacity was set to 56 Mb/s while the number was six for the 8-Mb/s case.) Note that for the same channel capacity, the magnitude of utilization gain levels off as p increases. (A similar observation was first discussed in [20].) Therefore, our admission control algorithm can be used to provide low block loss rate while achieving high channel utilization.

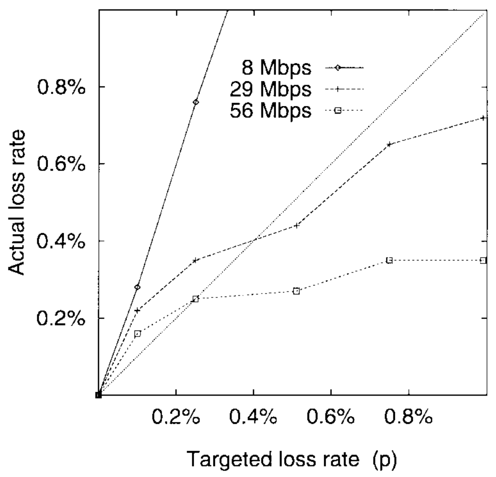

Fig. 5. Actual versus expected loss rate.

TABLE II
ACTUAL LOSS RATE

L1 = 8 Mb/s		L1 = 29 Mb/s		L1 = 56 Mb/s	
Expected	Actual	Expected	Actual	Expected	Actual
0%	0%	0%	0%	0%	0%
0.10%	0.28%	0.10%	0.22%	0.10%	0.16%
0.25%	0.76%	0.25%	0.35%	0.25%	0.25%
0.51%	1.52%	0.51%	0.44%	0.51%	0.27%
0.75%	1.49%	0.75%	0.65%	0.75%	0.35%
0.99%	1.77%	0.99%	0.72%	0.99%	0.35%

2) Actual Loss Rate: For the channel utilization gain to be meaningful, the actual block (picture) loss rates in the experiments must be close to their respective target values. In Fig. 5, we compare the actual picture loss rate in each experiment, averaged over five simulation runs using different random seeds, with the target value. From the figure, it can be concluded that our admission control algorithm predicts the actual loss rate well when a large number of flows share the channel. (Around 30 flows were admitted when the channel capacity of L1 was set to 56 Mb/s.) This agrees with our analysis; the larger the number of flows sharing the channel, the better the approximation based on the central limit theorem. Note that the solid 45° line represents perfect prediction by the central limit theorem. (The exact loss rates are listed in Table II for reference.)

F. Related Work:

In this section, we review some related work in the area of admission control for statistical service.

Clark *et al.* [4] proposed predicted service as the statistical service component of their integrated services model for the Internet. Later, Jamin *et al.* [15] designed a measurement-based admission algorithm for predictive service and reported very good performance in network utilization. However, predictive service by design requires that applications adapt to large changes in end-to-end packet delays. It is not appropriate for applications such as remote teaching and video on demand. Guerin *et al.* [13] and others (e.g., [6]) proposed to use effective bandwidth for admission control. While the admission condition is simple using effective bandwidth, sophisticated traffic modeling, policing, and monitoring are required. In comparison, our block traffic model is simpler.[10] Chong *et al.* [3] and others proposed to use neural nets and fuzzy logic combined with online traffic measurements for "intelligent" admission control and reported very positive experimental results. However, the performance of such an algorithm depends heavily upon how the neural net was trained and the data used for the training.

Note that none of the above proposals considered the performance of application data units, which we believe is more important to applications than packet performance.

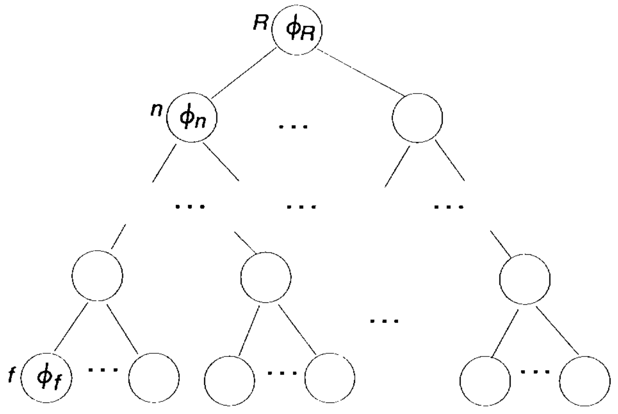

Fig. 6. Link-sharing tree model.

IV. HIERARCHICAL LINK SHARING

Existing delay bounds for blocks (bursts), e.g., the ones presented in Section II-B, were derived for networks with a flat one-level link-sharing structure. Specifically, each link in the networks can be modeled as a single server. In this section, we discuss how to generalize the delay bounds to networks in which links are hierarchically shared.

A. Directed Tree Model

First, we describe a directed tree model, largely borrowed from [1], for representation of a link-sharing hierarchy (see Fig. 6). The root node, denoted by R, corresponds to the physical link, each leaf node corresponds to a flow with a queue of packets, and each nonleaf node (except for the root) corresponds to a link-sharing entity, e.g., an administrative agency, a traffic type, or a service class. A nonleaf node n is called backlogged if at least one flow in its *leaf descendent node set*, denoted by $\mathrm{leaf(n)}$, is backlogged. Conceptually, node n is a logical server for its descendents. The amount of work done by n in the time interval $[t_1, t_2]$ is defined to be $W_n(t_1, t_2) = \sum_{f \in \mathrm{leaf(n)}} W_f(t_1, t_2)$, where $W_f(t_1, t_2)$ is

[10]We know that it is not trivial to police preventively the variance of a flow. However such policing is not required in our algorithm. Specifically when processing a new flow request, our algorithm does not need an accurate value of the flow's rate variance because of the existence of an upper bound on variance (see Theorem 4 in the Appendix). Once admitted, the flow's variance and average rate will be monitored and policed if necessary based on measurements. Traffic monitoring and policing are integral parts of our framework. Because of the complexity of the topic and space limit, we have decided to treat it separately in a future report.

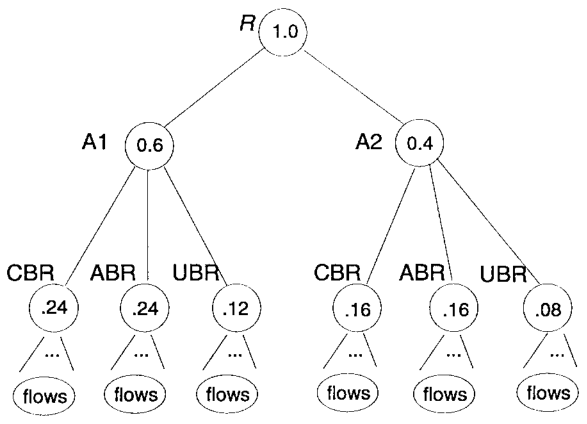

Fig. 7. An example link-sharing tree.

the amount of flow f traffic served in the interval $[t_1, t_2]$. To achieve link sharing, each node m is assigned a weight $\phi_m > 0$, which is a relative measure of the link share desired by entity m. For ease of presentation and without loss of generality, we assume that

$$\phi_R = 1 \tag{15}$$

$$\sum_{m \in \text{children}(n)} \phi_m \le \phi_n \quad \forall \text{ non-leaf node } n. \tag{16}$$

Notation: Let $H(m)$ be the number of ancestors that node m has. Let $p^h(m)$ be the ancestor node of m that is h levels higher than m in the tree. Clearly $p^1(m) = \text{parent(m)}$ and $p^{H(m)}(m) = R$. For ease of presentation, we also set $p^0(m) = m$.

Hierarchical link sharing can have a big impact on the performance of flows that share the link. Specifically, packets of a flow under the link-sharing hierarchy are scheduled jointly by *all* ancestor nodes of the flow before being served by the link [1]. We illustrate this point with the following example.

Example 1: Consider the simple link-sharing hierarchy shown in Fig. 7. Assume that all logical server nodes are currently backlogged. For the link to serve one of the available bit rate (ABR) flows from agency A1 next, both of the following conditions have to be met: 1) it is determined by the scheduler of the root logical server that it is agency A1's turn to receive service and 2) it is determined by the scheduler of the A1 logical server that it is traffic class ABR's turn to receive service. Therefore, if the root logical server used a scheduling algorithm that could not guarantee agency A1 a link share of 0.6, then the performance of A1's VBR flows would be affected negatively.

For hierarchical link sharing to be useful, appropriate scheduling algorithms must be used at the logical servers to minimize the negative impact on flow performance. We next define ideal link sharing to set a performance target for designing such scheduling algorithms. Assume that the link bandwidth is C (b/s).

Definition 1: We say that ideal link sharing is achieved for logical server n if, for any time interval $[t_1, t_2]$ in which n is continuously backlogged, the following holds

$$W_n(t_1, t_2) \ge r_n \cdot (t_2 - t_1) \tag{17}$$

where $r_n = \phi_n \cdot C$, that is, server n is guaranteed its link share whenever it has work to do.

It is straightforward to show that if every logical server is a fluid model fair queueing (FFQ) server,[11] then ideal link sharing will be achieved for every logical server. Unfortunately, FFQ is not feasible. Therefore, scheduling algorithms that are good approximations of FFQ should be used at the logical servers.

B. Prior Work

In [1], Bennett and Zhang proposed a new approximation of FFQ, named worst-case fair weighted fair queueing plus ($\mathrm{WF^2Q+}$), for providing tight delay bounds to flows under a link-sharing hierarchy. They defined a metric for a logical server called normalized bit worst-case fair index (NB-WFI).

Definition 2: A logical server node n is said to guarantee an NB-WFI of $\alpha_{m,n}$ for its child m if, during any time interval $[t_1, t_2]$ in which m is continuously backlogged, the following holds:

$$W_m(t_1, t_2) \ge \frac{\phi_m}{\phi_n} W_n(t_1, t_2) - \alpha_{m,n}. \tag{18}$$

They then showed that delays of flow f packets are bounded by

$$\frac{\sigma_f}{r_f} + \sum_{h=0}^{H(f)-1} \frac{\alpha_{p^h(f)}}{r_{p^h(f)}} \tag{19}$$

where $\alpha_{p^h(f)}$ is the NB-WFI guaranteed to node $p^h(f)$ by node $p^{h+1}(f)$ if the flow is constrained by a leaky bucket (ρ_f, σ_f) with $\rho_f \le r_f$. Based on this result, Bennett and Zhang claimed that small NB-WFI values for the logical servers are necessary for providing tight delay bounds to the flows. They also showed that, among all proposed packet fair queueing servers, a $\mathrm{WF^2Q+}$ server offers the smallest NB-WFI, namely, the length of one packet when a fixed packet length is used by all flows.

The problem of scheduling packets for flows under a link-sharing hierarchy was also studied by Goyal *et al.* in [11], where an algorithm called start-time fair queueing (SFQ) was proposed. SFQ is very similar to the well-known self-clocked fair queueing (SCFQ) [5], except that in SFQ the virtual start tag is used instead of the virtual finish tag of a packet as the priority value of the packet. Goyal *et al.* observed that the intermediate logical servers no longer have constant service rates for their children. Therefore, they analyzed the performance of SFQ in the context of FC servers first defined by Lee [18].

Definition 3: A server is said to be FC with parameters (C, δ), or simply a $FC(C, \delta)$ server,[12] if, for all intervals $[t_1, t_2]$ in a busy period of the server, the work done by the

[11] Also called general processor sharing (GPS) server in the literature [21].

[12] A constant rate server is also FC with $\delta = 0$.

server, denoted by $W(t_1, t_2)$, satisfies

$$W(t_1, t_2) \geq C \cdot (t_2 - t_1) - \delta. \tag{20}$$

In addition, they showed that the service received by a flow from an FC SFQ server is also FC. Thus, if the logical servers (called virtual servers in [11]) are all SFQ servers, the packet delay bound for a flow under a link-sharing hierarchy can be recursively computed. The exact bound is not given in [11].

C. A General Approach

While the authors cited above have made important contributions to the design of WF^{2}Q+ and SFQ, their approach is not the best. Specifically, too much emphasis was put on one good fair queueing algorithm for *both* link sharing and packet scheduling. For future networks, heterogeneous packet scheduling algorithms may be required at different parts of the link-sharing hierarchy to achieve multiple design goals. For example, if implementation complexity is of primary concern, scheduling algorithms like deficit round robin [24] may be more desirable than others. Also, there are a large number of performance results in the literature for one-level servers. It is not obvious from [1] and [11] how these results, e.g., those for weighted fair queueing (WFQ) servers [7] or even first-in first-out (FIFO) servers, can be extended to networks with hierarchical link sharing.

Next we describe an approach in which link sharing and packet scheduling concerns are *separated*. In particular, a link-sharing hierarchy is considered an extension of a one-level constant rate server. Consider a particular flow f under the link-sharing hierarchy. Even with hierarchical link sharing, the flow is in essence scheduled by a one-level server, its parent node, but with a variable service rate. The impact of all nonparent ancestors of f is indirectly accounted for by the service rate fluctuation of the parent node. Therefore, the analysis of flow f performance can be carried out in two steps:

1) characterization of the service rate fluctuation of the parent server;
2) extension of the performance results for one-level servers to account for service rate fluctuations (characterized in the previous step).

For step 1, we characterize the parent server $n = \mathrm{parent(f)}$ as an FC server. That is, there exists a constant $\delta_n \geq 0$ such that, for any time interval $[t_1, t_2]$ in which n is backlogged (busy) throughout, the work (service) done by n satisfies

$$W_n(t_1, t_2) \geq r_n \cdot (t_2 - t_1) - \delta_n. \tag{21}$$

Note that the smaller δ_n is, the less the service rate fluctuation is for n. If all ancestors of n are FFQ servers, $\delta_n = 0$. Therefore, good approximations of FFQ should be used for ancestors of n to ensure a small value for δ_n. Let us look at two examples.

Example 2: Assume that all ancestors of n are WF^{2}Q+ servers and all packets have a fixed length of L b. From Definition 1 and the fact that WF^{2}Q+ guarantees an NB-WFI of L [1], it can be shown that δ_n is at most

$$\left(\sum_{h=0}^{H(n)-1} \frac{r_n}{r_{p^h(n)}} \right) L. \tag{22}$$

Let us make the conservative assumption that $r_{p^h(n)}/r_{p^{h+1}(n)} = 0.5, h = 0, 1, \cdots, H(n) - 1$. (The ratio is usually much smaller in reality.) Then it is easy to show that $\delta_n \leq 2L$.

Example 3: Assume that all ancestors of n are SFQ servers and all packets have a fixed length of L b. Applying Theorem 2 of [11], it can be shown that δ_n is at most

$$\left(1 + \left(\sum_{h=0}^{H(n)-1} \frac{r_n}{r_{p^{h+1}(n)}} (Q(h) + 1) \right) \right) L \tag{23}$$

where $Q(h)$ is the number of branches that $p^{h+1}(n)$ has.

Let us assume that $r_{p^h(n)}/r_{p^{h+1}(n)} = 0.5$ and $Q(h) = Q, h = 0, 1, \cdots, H(n) - 1$. Then δ_n could be as large as $(Q+2)L$. Therefore, in general, SFQ causes larger service rate fluctuations than WF^{2}Q+. On the other hand, SFQ appears to have a smaller implementation cost.

For step 2, we next show how to extend delay guarantee results from constant rate servers to FC servers.

Definition 4: For an $\mathrm{FC(C, \delta)}$ server, its *corresponding constant rate server* is defined to be identical to the FC server except that it has a constant service rate of C b/s.

Consider a class of *work conserving* servers, called the *priority class*, which can be described in general as follows. A priority value is computed and assigned to every packet upon its arrival, and queued packets are scheduled for service in the order of increasing priority values. Ties between packets are broken arbitrarily. Also, within a particular flow, the priority of each packet is nondecreasing in packet arrival time.

Notation: Consider an arbitrary sequence of packet arrivals to a priority server. (A packet arrival is represented by a tuple consisting of the packet arrival time and the packet size.) We use the following notation:

$A(p)$	arrival time of packet p in the arrival sequence;
$s(p)$	size of packet p (in bits);
$P(p)$	priority value assigned to packet p;
$L(p)$	departure time of packet p.

In what follows, we focus on a subclass called service independent priority (SIP) servers, defined below.

Definition 5: An $FC(C, \delta)$ priority server belongs to the SIP subclass if for every packet p of an arrival sequence to the server, $P(p)$ depends exclusively upon the value of C and packet arrivals up to and including p.

It is easy to see that FC versions of virtual clock [31] and delay-earliest due date (delay-EDD) [8] servers belong to the SIP class. FC versions of servers that approximate a hypothetical constant rate FFQ server (such as WFQ [7], packet generalized processor sharing (PGPS) [21], and WF^{2}Q+ [1]) can be defined so that their packet priority values depend upon virtual times in the constant rate FFQ server and are thus SIP servers as well. FC FIFO servers also belong to the SIP class since they in effect use the arrival time of a packet as the packet priority. (Lee obtained some delay bound results for FC FIFO servers with leaky bucket constrained sources [18].)

In particular, we consider two types of SIP servers: *preemptive-resume with no overhead* and *nonpreemptive*. For an SIP server that is preemptive-resume with no overhead, it will immediately stop the service of a packet and serve a newly arrived packet if the new arrival has a smaller priority value. But no work will be lost because of the preemption, i.e., when resuming service for the preempted packet, the server will start from where it stopped. For an SIP server that is *nonpreemptive*, the service of a packet cannot be preempted once it is started.

Next we present two theorems on extension of delay guarantees for SIP servers.[13] (Their proofs are in the Appendix.) They deal with preemptive and nonpreemptive SIP servers, respectively. We say that a delay guarantee has the *firewall* property if the guarantees to packets of a flow are independent of how other flows behave.

Theorem 1: Consider an $FC(C, \delta)$ SIP server that is *preemptive-resume with no overhead*. If its corresponding constant rate server guarantees a departure deadline of $P(p) + \beta$ to every packet p of an arrival sequence, where β is a constant, and the guarantee has the firewall property and is independent of the priority tie-breaking method, then it guarantees to every packet p a departure deadline of $P(p) + \beta + (\delta/C)$.

Corollary 1: An $FC(C, \delta)$ FIFO server guarantees a delay bound of $\beta + (\delta/C)$ to every packet p of an arrival sequence if its corresponding constant rate server guarantees a delay bound of β to every packet p.

For nonpreemptive servers, the service may be out of- order sometimes. It happens when a newly arrived packet has a priority value smaller than that of the packet being served, but preemption is not allowed.

Definition 6: Assume that a nonpreemptive server guarantees to packet p a deadline of $D(p)$. The guarantee is said to account for out-of-order service if, with preemption, the server can guarantee p a deadline of $D(p) - (s_{\max}/C)$ where $s_{\max}$ is the maximum packet size.

Theorem 2: Consider a *nonpreemptive* $FC(C, \delta)$ SIP server. If its corresponding constant rate server guarantees a departure deadline of $P(p) + \beta$ to every packet p of an arrival sequence, and the guarantee has the firewall property, accounts for out-of-order service, and is independent of the priority tie-breaking method, then it guarantees to every packet p a departure deadline of $P(p) + \beta + (\delta/C)$.

Note that both theorems and Corollary 1 are quite general. They do not depend on specific admission control conditions or source control mechanisms. In contrast, most of Lee's analyses on FIFO FC servers [18] were done for leaky bucket constrained sources.

Example 4: This example illustrates that the extended deadline is as tight as the original deadline. Consider a virtual clock server [31] that is FC(1, 4). It is currently shared by two flows; flow A has a reserved rate of 1/6 and flow B has a reserved rate of 5/6. Assume that both flows use a fixed packet length of 1. The first packet of flow A arrived at time 0 and the first packet from flow B, denoted as p, arrived at time 1/5. A constant rate virtual clock server with a unit service rate guarantees to p a departure deadline of $P(p) + 1 = 1.4 + 1 = 2.4$ [27]. p actually will depart from the constant rate server at 2, which is close to 2.4. From Theorem 2, the FC server guarantees a deadline of $2.4 + 4 = 6.4$ to p. The FC server could be idle from time 1 to 5 and, in that case, p would not depart from the FC server until time 6, which is close to 6.4.

[13]Note that the servers need to perform some form of admission control to provide meaningful delay guarantees.

The burst (block) delay bounds presented in Section II-B were derived from a delay guarantee of virtual clock servers that has the firewall property, accounts for out-of-order service, and is independent of priority tie-breaking method [27]. Using Theorem 2, it is straightforward to extend them to networks with hierarchical link sharing.

Corollary 2: Consider the end-to-end burst delay bounds presented in Section II-B. If at each switch k the packets of the flow are served by a $FC(C_k, \delta_k)$ logical server instead of a constant rate link with capacity C_k, then the end-to-end delay of burst m of the flow is bounded as follows:

$$D_m \leq \frac{b_m + 1}{\lambda_m} + (K-1) \max_{1 \leq n \leq m} \left\{ \frac{1}{\lambda_n} \right\} + \sum_{k=1}^{K} \left(\alpha_k + \frac{\delta_k}{C_k} \right) \tag{24}$$

$$D_m \geq (K-1)\frac{1}{\lambda_m} + \sum_{k=1}^{K} \alpha_k. \tag{25}$$

V. Concluding Remarks

Although our admission control algorithm makes decisions entirely based on available bandwidth, our work is very different from those assuming a bufferless model. Specifically for real-time block services, packets are buffered at a channel while waiting for their turn to be transmitted. With block-based admission control, the amount of buffer space required for a flow is bounded, and can be precisely computed based on the maximum block size, the interpacket spacing within a block, and the flow's delay bound for the channel. In this paper, we have assumed that sufficient buffer is allocated to each flow so that no packet loss occurs due to buffer overflow.

In addition to admission control and link sharing, several important components of real-time block transfer services are still under developement. Recently we have investigated how to incorporated active loss management techniques into our framework so that losses can be distributed more evenly among flows subscribing to the same class of statistical service. The results are reported in [28], which also contains more discussions of related work. We will also look into using other statistical tools (such as the large deviation theory [23]) to improve the admission control accuracy.

Appendix

Proof of Theorem 1: Define $N(p)$ as the set of packets in the arrival sequence whose priority values are less than or equal to that of packet p and $W_p(t_1, t_2)$ as the total work (in bits) done by the FC server *for packets in* $N(p)$ in the time interval $[t_1, t_2]$.

We will carry out a proof by contradiction. Specifically, we assume that there exists a packet p in the arrival sequence such that

$$L(p) > P(p) + \beta + \frac{\delta}{C}. \tag{26}$$

Then we will show

$$W_p\left(0, P(p) + \beta + \frac{\delta}{C}\right) \geq \sum_{q \in N(p)} s(q) \tag{27}$$

which contradicts (26).

We use the superscript c to label a term defined for the constant rate server. By definition of SIP servers, p has the same priority value $P(p)$ in both server systems. Thus, we have

$$N^c(p) = N(p). \tag{28}$$

From (26), there exists a time period of which $[A(p), P(p) + \beta + (\delta/C)]$ is a subinterval and in which the FC server is continuously busy with $N(p)$ packets. Let p^* be the packet that started this time period. (Note that p^* may be p itself.) In other words, there is no other $N(p)$ packet in the FC system when p^* arrived. Consider the set of packets in $N(p)$ that arrived prior to $A(p^*)$ in the FC system (same as the packets in $N^c(p)$ that arrived prior to $A(p^*)$ in the constant rate system). Since all of them have been served in the FC system by $A(p^*)$ (but not necessarily in the constant rate system), we have

$$W_p(0, A(p^*)) \geq W_p^c(0, A(p^*)). \tag{29}$$

Since the FC server is busy exclusively with packets in $N(p)$ throughout the interval $[A(p^*), P(p) + \beta + (\delta/C)]$, we have

$$\begin{aligned} W_p&\left(A(p^*), P(p) + \beta + \frac{\delta}{C}\right) \\ &\geq C \cdot \left(P(p) + \beta + \frac{\delta}{C} - A(p^*)\right) - \delta \qquad (30) \\ &= C \cdot (P(p) + \beta - A(p^*)) \qquad (31) \\ &= W^c(A(p^*), P(p) + \beta). \qquad (32) \end{aligned}$$

Combining (29) with (32), we have

$$\begin{aligned} W_p&\left(0, P(p) + \beta + \frac{\delta}{C}\right) \\ &\geq W_p^c(0, A(p^*)) + W^c(A(p^*), P(p) + \beta). \qquad (33) \end{aligned}$$

Consider the constant rate server and a modification to the arrival sequence as follows. Those packets in $N^c(p)$ that arrive at or after $A(p) + (s(p)/C)$ in the original sequence will have their arrival times moved forward in the modified sequence such that they all arrive in the time interval $(A(p), A(p) + (s(p)/C))$ and the original order of arrivals for each flow is preserved. Note that the priority value of p as well as the guaranteed deadline of p are unaffected by the modification. Moreover, since the priority value of each packet within a flow is nondecreasing in packet arrival time (by definition of a priority class server), $N^c(p)$ is unaffected by the modification as well.

The deadline guarantee of $P(p) + \beta$ by the constant rate server still holds for the modified packet arrival sequence because of the firewall property. Moreover, the guarantee is independent of the tie-breaking method. Therefore, it would hold even if p finished service last among all $N^c(p)$ packets. Also with the modified arrival sequence, the most amount of work that the constant rate server can do for $N^c(p)$ packets in the time interval $[0, P(p) + \beta)]$ is $W_p^c(0, A(p^*)) + W^c(A(p^*), P(p) + \beta)$. Therefore, the following must hold:

$$W_p^c(0, A(p^*)) + W^c(A(p^*), P(p) + \beta) \geq \sum_{q \in N^c(p)} s(q). \tag{34}$$

Combining (33) and (34), we have

$$W_p\left(0, P(p) + \beta + \frac{\delta}{C}\right) \geq \sum_{q \in N^c(p)} s(q). \tag{35}$$

Because of (28), $N^c(p)$ can be substituted by $N(p)$ in (35). Therefore, (27) holds.

Proof of Corollary 1: For FIFO servers, $P(p) = A(p)$ for all p. Therefore, a delay bound guarantee of β is equivalent to a deadline guarantee of $P(p) + \beta$. The proof for the deadline guarantee generalization is identical to the one for Theorem 1 except for a simpler reasoning for (34): While a deadline guarantee by a FIFO server does not have the firewall property, there is no need for the condition because all packets in $N^c(p)$ arrive no later than $A(p)$, therefore, we can reason for (34) with the original arrival sequence.

Proof of Theorem 2: Again we carry out a proof by contradiction. Specifically, we assume that for the nonpreemptive FC server there exists a packet p in the arrival sequence such that

$$L(p) > P(p) + \beta + \frac{\delta}{C}. \tag{36}$$

Then we will show

$$W_p\left(0, P(p) + \beta + \frac{\delta}{C}\right) \geq \sum_{q \in N(p)} s(q) \tag{37}$$

which contradicts (36).

In this proof, we also consider a third SIP server, which is identical to the constant rate server except that it is preemptive-resume with no overhead. We use the superscripts c and pre to label terms for, respectively, the nonpreemptive and preemptive constant rate server systems. Note that by definition of SIP servers, p has the same priority value $P(p)$ for all three server systems, i.e.,

$$N^{\mathrm{pre}}(p) = N^c(p) = N(p). \tag{38}$$

It is given that

$$L^c(p) \leq P(p) + \beta \tag{39}$$

and the guarantee in (39) accounts for out-of-order service. Therefore, we have

$$L^{\mathrm{pre}}(p) \leq P(p) + \beta - \frac{s_{\max}}{C}. \tag{40}$$

Note that the above guarantee also has the firewall property.

Similar to the proof for Theorem 1, we can show that there exists a packet $p^* \in N(p)$ such that

$$W_p(0, A(p^*)) \geq W_p^{\rm pre}(0, A(p^*)) \tag{41}$$

$$W_p\left(A(p^*), P(p)+\beta+\frac{\delta}{C}\right) \geq C \cdot (P(p)+\beta-A(p^*))-s_{\max}. \tag{42}$$

The extra term $s_{\max}$ in (42) is due to the fact that a part of the work done by the nonpreemptive FC server in $[A(p^*), P(p) + \beta + \delta/C]$ may be for out-of-order service of a packet not in $N(p)$.

Similar to the proof for Theorem 1, we have

$$W_p^{\rm pre}(0, A(p^*)) + W^{\rm pre}\left(A(p^*), P(p) + \beta - \frac{s_{\max}}{C}\right) \geq \sum_{q \in N^{\rm pre}(p)} s(q). \tag{43}$$

Moreover

$$\begin{aligned} W^{\rm pre}&\left(A(p^*), P(p) + \beta - \frac{s_{\max}}{C}\right) \\ &= C \cdot \left(P(p) + \beta - \frac{s_{\max}}{C} - A(p^*)\right) &(44)\\ &= C \cdot (P(p) + \beta - A(p^*)) - s_{\max}. &(45) \end{aligned}$$

Combining (43) and (45), we have

$$W_p^{\rm pre}(0, A(p^*)) + C \cdot (P(p) + \beta - A(p^*)) - s_{\max} \geq \sum_{q \in N^{\rm pre}(p)} s(q). \tag{46}$$

From (41), (42), and (46), we have

$$\begin{aligned} W_p&\left(0, P(p) + \beta + \frac{\delta}{C}\right) \\ &= W_p(0, A(p^*)) + W_p\left(A(p^*), P(p) + \beta + \frac{\delta}{C}\right) &(47)\\ &\geq W_p^{\rm pre}(0, A(p^*)) + C \cdot (P(p) + \beta - A(p^*)) - s_{\max} &(48)\\ &\geq \sum_{q \in N^{\rm pre}(p)} s(q). &(49) \end{aligned}$$

Because of (38), $N^{\rm pre}(p)$ can be substituted by $N(p)$ in (49). Therefore, (37) holds. □

Theorem 3 ([19]): Assume that $X_1, X_2, \cdots$ are independent random variables centered at expectations with distribution functions (d.f.'s) $F_1, F_2, \cdots$ and variance $\sigma_1^2, \sigma_2^2, \cdots$.

Let $S_n = \Sigma_{k=1}^n X_k$ be their consecutive sums with variance $s_n^2 = \Sigma_{k=1}^n \sigma_k^2, n = 1, 2, \cdots$. Then

$$\frac{S_n}{s_n} \rightarrow N(0, 1) \tag{50}$$

if and only if for all $\varepsilon > 0$

$$\xi_n(\varepsilon) = \frac{1}{s_n^2} \sum_{k=1}^n \int_{|x| \geq \varepsilon \cdot s_n} x^2 dF_k \rightarrow 0 \tag{51}$$

as $n \rightarrow \infty$. □

Theorem 4: For any random variable X that can take value between 0 and $X_{\max}$, the following holds:

$$\mathrm{var}[X] \leq (h - 1)E[X]^2 \tag{52}$$

where h is the peak-to-average ratio ($X_{\max}/E[X]$).

Proof: By definition, the variance of X, denoted by $\mathrm{var}[X]$, satisfies

$$\mathrm{var}[X] = \int x^2 \, dF(x) - E[X]^2. \tag{53}$$

Since $x^2 \leq X_{\max} \cdot x$, we have

$$\begin{aligned} \mathrm{var}[X] &\leq \int (X_{\max} \cdot x) \, dF(x) - E[X]^2 &(54)\\ &= x_{\max} \int x \, dF(x) - E[X]^2 &(55)\\ &= X_{\max} E[X] - E[X]^2 &(56)\\ &= (h - 1)E[X]^2. &(57) \end{aligned}$$

□

Acknowledgment

The authors would like to thank the anonymous reviewers for their constructive comments.

References

[1] J. C. R. Bennett and H. Zhang, "Hierarchical packet fair queueing algorithms," in *Proc. ACM SIGCOMM '96*, Stanford, CA, Aug. 1996, pp. 143–156.

[2] T. M. Chen, S. S. Liu, and V. K. Samalam, "The available bit rate service for data in ATM networks," *IEEE Commun. Mag.*, pp. 56–71, May 1996.

[3] S. Chong, S.-Q. Li, and J. Ghosh, "Predictive dynamic bandwidth allocation for efficient support of real-time VBR video over ATM," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 12–23, Jan. 1995.

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

[5] J. Davin and A. Heybey, "A simulation of fair queueing and policy enforcement," *Comput. Commun. Rev.*, vol. 20, pp. 23–29, Oct. 1990.

[6] G. de Veciana, G. Kesidis, and J. Walrand, "Resource management in wide-area ATM networks using effective bandwidth," *IEEE J. Select. Areas Commun*, vol. 13, pp. 1081–1090, Aug. 1995.

[7] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queuing algorithm," in *Proc. ACM SIGCOMM '89*, Aug. 1989, pp. 3–12.

[8] D. Ferrari and D. Verma, "A scheme for real-time channel establishment in wide-area networks," *IEEE J. Select. Areas Commun.*, vol. 8, Apr. 1990, pp. 368–379.

[9] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Trans. Networking*, vol. 3, pp. 365–386, Aug. 1995.

[10] *ATM Traffic Management Specification*, The ATM Forum, Version 4.0, 1995.

[11] P. Goyal, H. Vin, and H. Cheng, "Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks," in *Proc. ACM SIGCOMM '96*, Stanford, CA, Aug. 1996, pp. 157–168.

[12] M. Grossglauser, S. Keshav, and D. Tse, "RCBR: A simple and efficient service for multiple time-scale traffic (extended version)," in *ACM SIGCOMM '95*, (early version).

[13] R. Guerin, H. Ahmadi, and M. Naghshineh, "Equivalent capacity and its application to bandwidth allocation in high-speed networks," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 968–981, 1991.

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

[15] S. Jamin, P. Danzig, S. Shenker, and L. Zhang, "A measurement-based admission control algorithm for integrated services packet networks," in *Proc. ACM SIGCOMM '95*, Aug. 1995, pp. 1–13.

[16] S. S. Lam and G. G. Xie, "Burst scheduling networks," *Perform. Eval.*, to be published. (abbreviated version in *Proc. INFOCOM '95*, Apr. 1995).

[17] ———, "Group priority scheduling," *IEEE/ACM Trans. Networking*, vol. 5, pp. 205–218, Apr. 1997,

[18] K. Lee, "Performance bounds in communication networks with variable-rate links," in *Proc. ACM SIGCOMM '95*, Aug. 1995, pp. 126–137.

[19] M. Loève, *Probability Theory I*, 4th ed. New York: Springer-Verlag, 1977.

[20] R. Nagarajan, "Quality-of-service issues in high-speed networks," Ph.D. dissertation, Univ. Massachusetts, Amherst, Aug. 1993.

[21] 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.

[22] A. Romanow and S. Floyd, "The dynamics of TCP traffic over ATM networks," in *Proc. ACM SIGCOMM '94*, London, U.K., Aug. 1994.

[23] A. Schwarz and A. Weiss, *Large Deviations for Performance Analysis*. London, U.K.: Chapman and Hall, 1995.

[24] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round robin," in *Proc. ACM SIGCOMM '95*, Cambridge, MA, Aug. 1995.

[25] J. S. Turner, "Maintaining high throughput during overload in ATM networks," in *Proc. IEEE INFOCOM '96*, San Francisco, CA, Apr. 1996, pp. 287–295.

[26] Z. Wang and J. Crowcroft, "Analysis of burstiness and jitter in real-time communications," in *Proc. ACM SIGCOMM '93*, Sept. 1993, pp. 13–19.

[27] G. G. Xie and S. S. Lam, "Delay guarantee of virtual Clark server," *IEEE/ACM Trans. Networking*, vol. 3, pp. 683–689, Dec. 1995.

[28] ———, "Admission control and loss management for an application-level statistical service," in *Proc. 5th IEEE Int. Conf. Network Protocols (IEEE ICNP'97)*, Atlanta, GA, Oct. 1997. Available HTTP: http://www.cs.nps.navy.mil/people/faculty/xie/pub.

[29] ———, "Real-time block transfer under a link sharing hierarchy," in *Proc. IEEE INFOCOM'97*, Kobe, Japan, Apr. 1997.

[30] H. Zhang and E. W. Knightly, "RED-VBR: A renegotiation-based approach to support delay-sensitive VBR video," *ACM Multimedia Syst. J.*, to be published; also, in *Proc. NOSSDAV'95*, Durham, NH, Apr. 1995.

[31] L. Zhang, "Virtual Clock: A new traffic control algorithm for packet switching networks," in *Proc. ACM SIGCOMM '90*, Aug. 1990, pp. 19–29.

Geoffrey G. Xie (M'96) received the B.S. degree in computer science from Fudan University, Shanghai, China, in 1986, the M.S. degree in computer science and the M.A. degree in mathematics from Bowling Green State University, Bowling Green, OH, in 1988, and the Ph.D. degree in computer sciences from the University of Texas, Austin, in 1996.

From 1991 to 1993 he worked full-time as a Project Engineer with Schlumberger Austin Systems Center, Austin, TX. He is currently an Assistant Professor with the Department of Computer Science, Naval Postgraduate School, Monterey, CA. His research interests include integrated services computer networks and multimedia systems.

Simon S. Lam (S'69–M'69–SM'80–F'85) received the B.S.E.E. degree (with Distinction) from Washington State University, Pullman, in 1969, and the M.S. and Ph.D. degrees in engineering from the University of California at Los Angeles (UCLA), in 1970 and 1974, respectively.

From 1971 to 1974 he was a Postgraduate Research Engineer at the ARPA Network Measurement Center, UCLA, where he worked on satellite and radio packet switching networks. From 1974 to 1977 he was a Research Staff Member at the IBM T.J. Watson Research Center, Yorktown Heights, NY. Since 1977 he has been on the faculty of the University of Texas, Austin, where he is a Professor of computer sciences. He holds two anonymously endowed Professorships, and served as Department Chair from 1992 to 1994. His research interests in networking include switch and protocol design, performance analysis, distributed multimedia, quality of service guarantees, and security.

Dr. Lam has served on the editorial boards of IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, IEEE TRANSACTIONS ON COMMUNICATIONS, PROCEEDINGS OF THE IEEE, and *Performance Evaluation*. He presently serves as Editor-in-Chief of IEEE/ACM TRANSACTIONS ON NETWORKING. He organized and was Program Chair of the first ACM SIGCOMM Symposium held at the University of Texas, Austin, in 1983. He received the 1975 Leonard G. Abraham Prize Paper Award from the IEEE Communications Society for his paper on packet switching in a multiacess broadcast channel derived from his doctoral dissertation. He is a Fellow of the Association for Computing Machinery (ACM).