# An Efficient Network Architecture Motivated by Application-Level QoS

**Geoffrey G. Xie**
Department of Computer Science
Naval Postgraduate School
Monterey, CA 93943
*xie@cs.nps.navy.mil*

**Simon S. Lam**
Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-1188
*lam@cs.utexas.edu*

**CS-97-Xg-2** **May 1997**

**Abstract**

We consider application-level quality of service (QoS) requirements in design of networks that provide delay and loss guarantees. Using this approach, we have designed a novel network architecture that in many aspects is more efficient than ones motivated exclusively by packet QoS. An overview of our work is presented in this paper. The centerpiece of our design is a traffic model that enables delivery of application specific information to the network. Based on the traffic model, we have developed (i) efficient network techniques for providing application-level delay guarantees, (ii) an admission control algorithm for a statistical service that bounds application data losses below a specified value, and (iii) network techniques for managing application data losses to achieve fairness and protection of high priority data units marked by applications (e.g., I frames of MPEG applications).

**Keywords:** application-level QoS, deterministic traffic model, burst scheduling, admission control, selective early ADU discard

# 1 Introduction

Inside current networks, all data is encapsulated in packets. As every network has a maximum packet length, often an application data unit (ADU) is too large to be carried in a single packet. In such a case, the ADU must be segmented for network delivery. The advantage of this approach, evidenced by the success of the Internet, is that a single network can be shared by a diverse range of applications.

Encapsulation and segmentation of application data, however, have drawbacks. In particular, they cause the following tension between the network and an application: While the network has been designed to optimize the performance of packets (throughput, end-to-end delays, delay jitters, etc.), an application emphasizes on the performance of its data units rather than that of individual packets. For example, consider an application that sends a sequence of video pictures over an IP network. Each picture may be segmented into a sequence of IP packets. The loss rate of the pictures is much more important to the application than the loss rate of the packets.

Mapping application-level QoS requirements to the network level, given the complexity of network dynamics, is not trivial. There may even be cases where the network cannot provide what an application needs. This problem has become more acute because (i) emerging multimedia applications require stringent QoS from the network, and most of their data units (e.g., pictures) are large and must be segmented, and (ii) small packets, such as cells in ATM networks, are being used as means for good real-time performance.

There have been studies on how to bridge QoS at the network and application levels. However, they focused mainly on techniques that can be used at the end hosts (i.e. sender and receiver) to predict and/or enhance application-level QoS for a particular grade of network-level QoS. In [3], the idea of application-level framing was proposed to avoid application layer throughput degradation at the receiver; such degradation is caused by the transport layer suspending delivery of data when some packets are lost or out of order. Specifically, it was proposed that the sender transmits, along with data packets, application level framing (ADU boundary) information to the receiver. As a result, the transport layer at the receiver can still deliver other ADUs to the application layer when it has to hold a particular ADU because some packets of that ADU are lost or out of order. In [2], the relation between application-level QoS and network-level QoS was investigated in the context of an ATM network. The study concentrated on the impact of source peak transmission rate on message losses given a constant level of network cell losses. It was observed that the message loss rate of a connection may differ greatly with a different peak rate. A set of heuristics were developed, which can be used by an application to determine the network QoS required to ensure an appropriate level of losses of its messages.

We have taken a different approach, focusing directly on the network. In particular, we consider application-level requirements in design of networks that provide delay and loss guarantees [11, 12, 20,

19]. Using this approach, we have designed a novel network architecture that in many aspects is more efficient than ones motivated exclusively by packet QoS.

An overview of our work is presented in this paper. The centerpiece of our design is a traffic model that enables delivery of application specific information to the network. Based on the model, we have developed (i) efficient network techniques for providing application-level delay guarantees, (ii) an admission control algorithm for a statistical service that bounds application data losses below a specified value, and (iii) network techniques for managing application data losses to achieve fairness and protection of high priority data units marked by applications (e.g., I frames of MPEG applications).

The balance of this paper is organized as follows. In Section 2, our traffic model is described. In Section 3, network techniques for providing application-level delay guarantees are presented. In Section 4, an admission control algorithm that guarantees statistical performance of ADUs is described. In Section 5, network techniques for managing application data losses are presented.

## 2 Traffic Model

For clarity of exposition, we assume in the balance of this paper that packets are of fixed size (such as ATM cells). The results and specifications to be presented can be modified in a straightforward manner for networks where the packet size is variable, but bounded.

In designing networks that provide delay guarantees to delay sensitive variable bit rate (VBR) flows, we introduced the concept of a burst [11], which models a sequence of packets that encapsulate an application data unit. For video, for example, a burst models a sequence of packets that carry the encoded bits of a picture. The first and last packets of each burst are marked and the first packet carries information on the bit (or packet) rate of the burst. In summary, our traffic model is *deterministic* because the boundary and the exact bandwidth requirement of each ADU are encoded and accessible by the network.

The burst concept and traffic model are particularly applicable to network design in support of *application-level* services [11, 20]. Specifically, they allow each ADU's packets to be processed as a whole and independently from other ADUs. Consequently much more predictable network performance for ADUs can be rendered.

In what follows, we assume that each flow conforms to the deterministic traffic model defined below, and we will use the terms *ADU* and *burst* interchangeably.

**Definition 1 (deterministic traffic model)** *A flow is modeled as a sequence of bursts, each of which is a sequence of packets that carry the bits of an ADU. The first and last packets of each burst are marked, and the first packet carries some information on the ADU (including its bandwidth requirement).*[1]

There are costs associated with requiring flows to conform to the above traffic model, namely: the extra bits for encoding ADU information in some packets, and the extra processing by a network node to retrieve and act on the information. However, they are bearable with today's network technologies (especially hardware), and the benefits, as we will present in the remainder of this paper, outweigh the costs. In fact, our approach is consistent with the technology trend, in which the network is required to be more active, i.e. to perform more application specific computations [6], for better and more flexible services.

**Notation.**

| | |
|---|---|
| $i$ | integer; flow index |
| $m$ | positive integer; index of $m$th burst in a flow |
| $(m, l)$ | index of $l$th packet in burst $m$ |
| $b_m$ | size of burst $m$ (in packets) |
| $\lambda_m$ | bandwidth requirement for burst $m$ (in packets/second) |

## 3 Burst Scheduling Techniques

The first version of our deterministic traffic model was constructed as a component of a network architecture, called Burst Scheduling, that provides end-to-end delay and end-to-end jitter guarantees to packets of an VBR flow [11]. Since then, the model has motivated the development of several techniques [11, 12, 18], refereed to as burst scheduling techniques, that are useful in design of efficient networks that provide *application-level delay guarantees*.

Burst scheduling techniques are particularly useful when network channels employ rate-based packet service disciplines (e.g., [5, 14, 17, 21, 22]), which can be described generally as follows. Consider a particular channel. A reserved rate is allocated to each flow that shares the channel. Each packet in a flow is tagged, upon arrival, a priority (or deadline) computed using the rate allocated to the flow. The channel schedules packets for service based upon their priority tags. Rate-based service disciplines have the desirable firewall property; that is, the delay guarantee to a flow is unaffected by other (even misbehaving) flows that share the same channel.

[1] This part of our model can have different implementations. For example in ATM networks, it may be desirable to use RM cells to mark the boundary of a burst and carry rate information.

In the remainder of this section, some of the burst scheduling techniques are briefly described.

### 3.1 Burst-based Rate Allocation

With our traffic model, burst-based rate allocation can be implemented at a channel. Specifically, a reserved rate is not allocated to a flow until the first packet of a burst arrives, and the rate is subsequently deallocated when the last packet of the burst departs [11]. (At any time at most one burst in each flow is allocated its reserved rate.) As a result, the rate allocated to a VBR flow is variable, i.e. it changes from one burst to the next.

Burst-based rate allocation has the advantages of allocating a rate that is *exactly* what a flow needs at all time. Consequently, the channel has knowledge of the exact input load at all time. (In contrast, estimation based on queue length measurements might not always be accurate.) The performance of bursts (such as their delays) at the channel can also be predicted more accurately. These properties are desirable for admission control (see Section 4), and overload control (see Section 5).

Rate-based packet service disciplines can be modified, in a straightforward manner, to support burst-based rate allocation. Their packet delay guarantees can also be generalized to burst delay guarantees while retaining the firewall property. We have done both for the Virtual Clock [22] service discipline [11].

### 3.2 Application-Level Delay Guarantee: an Example

As mentioned earlier, our traffic model enables burst scheduling techniques that are especially useful in providing application-level delay guarantees. Next we will illustrate this point with an example network architecture, which is described in detail in [11]. The network architecture consists of two major components: a regulator at each source to enforce a burst-based flow specification, and a packet scheduler at each channel to provide delay guarantees to bursts.

#### 3.2.1 Flow Specification

In addition to conforming to the deterministic traffic model, each flow is enforced by a source regulator to satisfy a burst-based flow specification defined below.

**Definition 2 (Flow Specification) [11]** *A flow conforms to the following conditions when entering the network*

- *Packets in burst $m$ satisfy a* jitter *timing constraint, namely: for $l = 1, 2, \ldots, b_m$,*

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

*where $A(m,l)$ is the arrival time of packet $(m,l)$.*

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

$$A(m+1,1) - A(m,1) \geq \frac{b_m}{\lambda_m}. \tag{2}$$

Timing constraint (1) specifies a jitter bound over the packets of each burst; e.g., this constraint is satisfied if all packets of a burst arrive at the same time. Such a jitter bound is necessary if a network is to provide a tight burst delay bound. Timing constraint (2) specifies a minimum separation between two consecutive burst arrivals in a flow. This is a form of source control [11].

### 3.2.2 Packet Scheduler

The packet scheduler at each channel uses the Virtual Clock service discipline [22], modified slightly to allow variable rate allocation. Specifically, it creates a new queue for each new flow, as well as a flow regulator for the queue [11]. There are three state variables associated with each queue: $P$, $Q$, and $E$, defined below. There are four variables associated with burst $m$: $a_m$, $m$, $\lambda_m$ and $u_m$. The flow regulator can read and write all of these variables.

| | |
|---|---|
| $P$ | virtual clock value of head-of-line packet in queue; initially 0 |
| $Q$ | time when burst is eligible for selection by scheduler; initially 0 |
| $E$ | boolean flag, indicating that the flow has an eligible burst; initially **false** |
| $m$ | burst number; initially 1 |
| $a_m$ | arrival time of first packet of burst $m$ |
| $u_m$ | time ahead (in seconds) of burst $m$; initialized to zero at source and carried by first packet |

The flow regulator performs two main actions: (i) reconstruction of a burst by delaying packets if necessary[2] to satisfy the jitter and separation timing constraints, and (ii) computation of virtual clock values for packets. They are specified below. The specification uses a function, $now()$, and a procedure $update$($P$,$E$). When called, $now()$ returns the current time from a local clock. The $update$ procedure is specified as follows:

```
                  procedure update(P, E)          // execute once per burst
1                 Q := max{a_m + u_m, P} ;
```

[2]Therefore, the packet scheduler can be non-work-conserving.

2 $delay(Q - now())$ ;
3 $P := Q + 1/\lambda_m$ ;
4 $E :=$ **true**;

where the procedure $delay(x)$ introduces a delay equal to $x$ if $x > 0$; else, it is a null operation with no delay.

A flow regulator is specified by the following two actions:

- Enabling condition: arrival of a burst $m$ packet to queue

```
1   if (packet is first in burst m)
2       then  record arrival time in a_m and values of u_m and λ_m ;
3             if (queue was empty before arrival)
4                 then update(P, E) ;
```

- Enabling condition: departure of a burst $m$ packet from queue (selected for service by scheduler)

```
1   if (departed packet is not last in burst m)
2       then  P := P + 1/λ_m
3       else  E := false ;
4             m := m + 1 ;
5             if (queue is not empty)
6                 then update(P, E) ;
```

The above algorithm is highly efficient. Specifically, the flow regulator executes the procedure $update(P,E)$ only once per burst, i.e. for the first packet in each burst. For any other packet in burst $m$, it simply increments $P$ by the value of $1/\lambda_m$, instead of executing the original Virtual Clock algorithm [22]. With the exception of the first packet of each burst, there is no need to store packet arrival times. Furthermore, at any time, only one virtual clock value is stored per flow, rather than one per packet. These improvements are made possible by considering application-level QoS, specifically by using the burst-based flow specification [11].

### 3.2.3 Delay Bounds

Next we present a theorem that summarizes the delay guarantees provided by a network that uses the example network architecture. Consider a flow that traverses a sequence of nodes, indexed by

$0, 1, 2, \ldots, K+1$, where node 0 denotes the source, node $K+1$ the destination, and the other nodes packet switches. We assume that no channel has been overbooked, i.e. each flow is admitted at connection setup on the basis of its peak rate (i.e. the maximum bandwidth requirement of a burst). The following delay guarantee theorem holds for the flow.

**Theorem 1 [11]** *Let $e_k$ be the constant overhead, i.e. transmission delay plus propagation delay, associated with the channel from node $k$ to $k+1$. The end-to-end delay of the first packet of burst $m$, denoted by $D(m,1)$, for $m = 1, 2, \ldots$, has the following upper and lower bounds:*

$$D(m,1) \leq \frac{1}{\lambda_m} + (K-1) \max_{1 \leq n \leq m} \{\frac{1}{\lambda_n}\} + \sum_{k=1}^{K} e_k, \tag{3}$$

$$D(m,1) \geq (K-1)\frac{1}{\lambda_m} + \sum_{k=1}^{K} e_k. \tag{4}$$

*Consequently, 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 $K+1$, has the following upper and lower bounds:*

$$D_m \leq \frac{b_m + 1}{\lambda_m} + (K-1) \max_{1 \leq n \leq m} \{\frac{1}{\lambda_n}\} + \sum_{k=1}^{K} e_k, \tag{5}$$

$$D_m \geq (K-1)\frac{1}{\lambda_m} + \sum_{k=1}^{K} e_k. \tag{6}$$

We have evaluated the network architecture by performing a set of simulation experiments using MPEG video data [11]. In Figure 1, we show the end-to-end delay performance of a particular video sequence named Energizer. The result agrees with the theoretical bounds in Theorem 1.

### 3.3 Group Priority

For a channel that employs a rate-based packet service discipline, the scheduler must repeatedly search for the smallest element in a set of deadlines. For high speed networks of the future, it is likely that a channel will be shared by hundreds, and perhaps, thousands of flows. Thus the search algorithm should be highly efficient. Furthermore, each search must be carried out within a time bound, i.e. the search must be finished by the end of the current packet transmission. Otherwise, the channel would be idled, ready packets would incur additional delays, and delay guarantees would not hold.

We developed a search algorithm based upon a novel data structure, called *adaptive heap*, which behaves like a heap most of the time, but adaptively changes its strategy when necessary to satisfy the

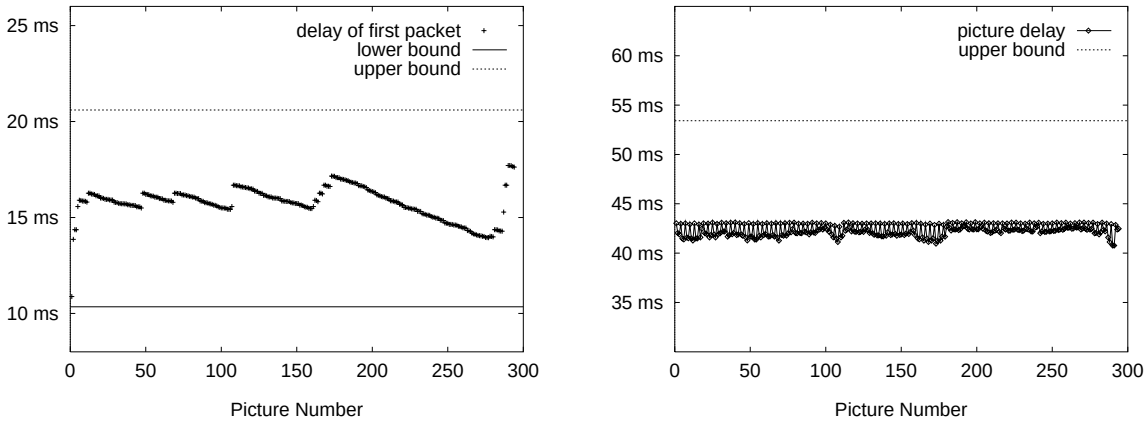


Figure 1: End-to-end packet and picture delays of Energizer sequence

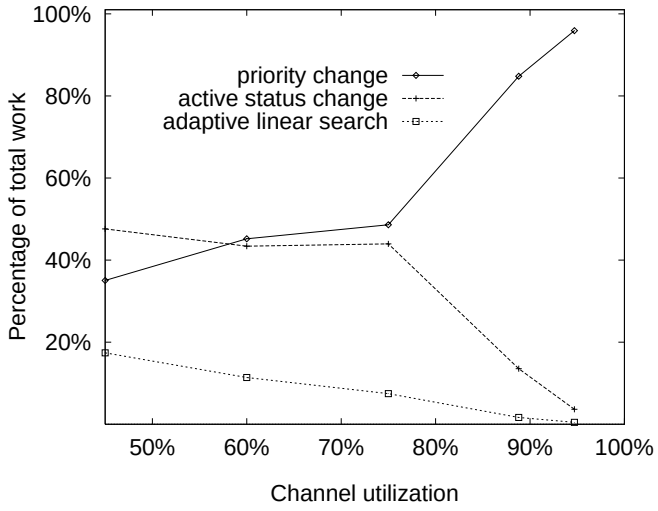


Figure 2: Typical breakdown of heap search overhead

time bound [18]. The algorithm has optimal worst-case time performance and good average performance.

We later discovered that, as illustrated in Figure 2, the cost for the adaptive heap search is dominated by the work due to priority changes when the channel utilization is high [18]. This discovery has motivated the idea of *group priority* [12, 18]. Specifically, consecutive packet arrivals in a flow are partitioned into groups. The largest deadline among packets in a group is assigned to every packet in the group. (Thus all packets except one in the group have relaxed deadlines.) As a result, a flow changes its priority value less frequently, i.e. from group to group instead of from packet to packet.

Group priority, while causing larger delays for some packets, is quite feasible when considering application-level QoS. In particular, we observed that large group sizes can be chosen for most bursts

such that the worst-case end-to-end burst delay of a VBR flow is unaffected by the use of group priority [12]. Specifically, the delay bounds in (5) and (6) can be generalized as follows

$$D_m \leq \frac{b_m + g_m}{\lambda_m} + (K-1) \max_{1 \leq n \leq m} \{\frac{g_n}{\lambda_n}\} + \sum_{k=1}^{K} e_k \quad (7)$$

$$D_m \geq (K-1)\frac{g_m}{\lambda_m} + \sum_{k=1}^{K} e_k, \quad (8)$$

where $g_m$ is the group size used for burst $m$. (Methods for choosing appropriate group sizes are described in [12].)

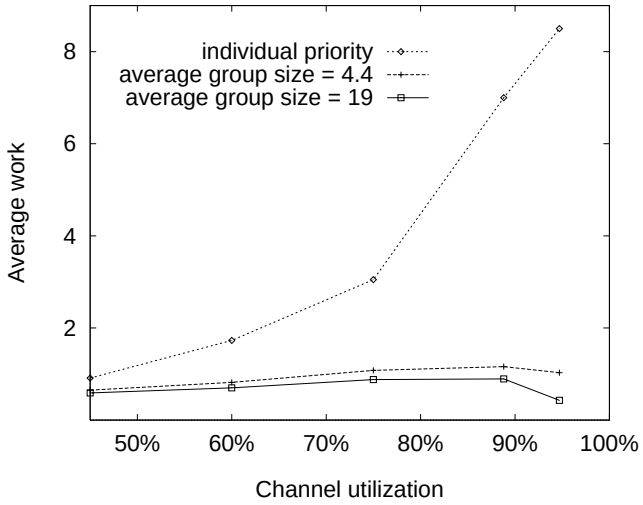


Figure 3: Reduction of search overhead with group priority

Group priority has two advantages. First, with group priority, the scheduler updates less often its data structure for storing priority values (e.g., a heap). As a result, the amount of work for scheduler search is reduced. The reduction can be very significant when the channel utilization is high, as we have observed in experiments [18]. In Figure 3, the result from one of the experiments is shown. Note that the average group size is one in the "individual priority" case.

Secondly, we have discovered empirically that the use of group priority results in much better statistical performance (i.e. delay, queue size, and loss probability) for networks where some channels are heavily utilized [12]. An example is given in Figure 4, which shows the picture delay performance of the Energizer video sequence in a network where some channels are severely overbooked. (By 164% overbooking, the sum of peak flow rates exceeds the channel's capacity by 164%. $g$ denotes the average group size for all flows that share the channel.) The good performance is due to the fact that with group priority, the delay bound of a burst is decoupled from its reserved rate. As a result, all bursts in a VBR

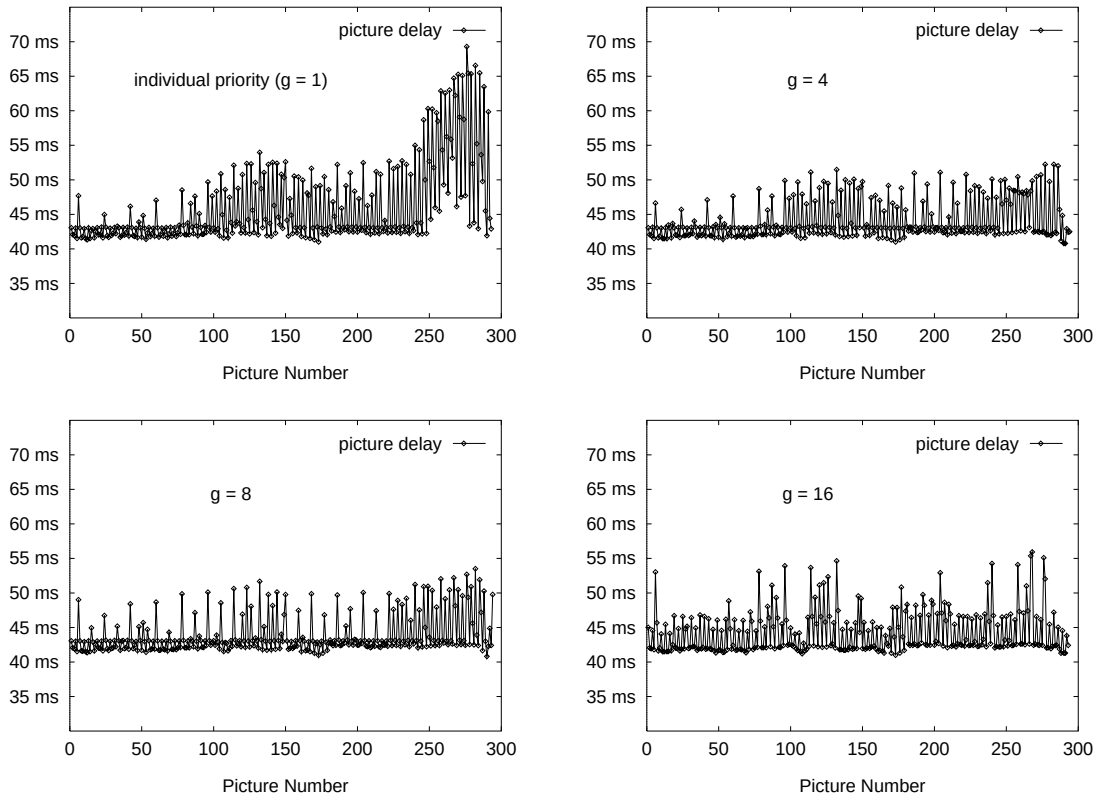


Figure 4: Delay performance of Energizer sequence with 164% overbooking for some channels

flow move through the network at a similar pace. This smoothes out the traffic inside the network, reducing the congestion at the channels that are severely overbooked [12].

# 4 Two Level Admission Control

The delay guarantee presented in the previous section is provided to every burst. Such a deterministic guarantee has a cost of low channel utilization because it requires peak rate reservation. For many applications, however, *statistical* guarantees are acceptable, such as: 99% of the pictures in a video sequence are delivered with delays less than the upper bound.

We have developed an efficient admission control algorithm, which bounds the burst loss rate at a network channel below a specified value while achieving high channel utilization [19, 20].[3] The algorithm is based on a novel approach, made possible by our traffic model, that combines admission control at the flow level and admission control at the burst level.

[3]We have also investigated how to perform nodal allocation of an application's end-to-end ADU loss requirement. See [19].

We assume that the path of each flow is fixed. For a particular flow, its path is a sequence of nodes, each of which consists of an outgoing channel, and a set of buffers for the channel where packets of different flows are queued. Consider a particular node. Let $C$ (bits/second) be the channel capacity dedicated to a statistical service with a target burst loss rate of $p$. (Note that it is straightforward to extend our design to a channel that is hierarchically shared by different agencies, by multiple classes of statistical services, etc. [1, 7, 20]).

### 4.1 Burst Level Admission Control

Our traffic model, as mentioned earlier, makes it possible for the packets of a burst to be processed as a whole and independently from other bursts. Thus, separate admission control can be performed for each burst. Specifically, a burst is accepted only if the burst's reserved rate[4] does not exceed the channel's unallocated capacity; otherwise, the entire burst will be discarded. (Note that a similar mechanism, called the ATM Block Transfer (ABT) capability, is being standardized by ITU-T [9].)

With burst level admission control, packet losses are concentrated over a small number of bursts, and channel capacity is not wasted on delivery of partial bursts.

Burst level admission control, as specified below, is an integral part of our admission control algorithm.

#### Algorithm specification of burst level admission control

The variable $A$ is used to store the aggregate rate allocated, and is initialized to 0. Recall that $\lambda_m$ denotes the bandwidth requirement (or reserved rate) of burst $m$.

- Enabling condition: arrival of first packet of burst $m$

*Burst_Admission_Control ( $m$ )*

```
1    if (A + λ_m > C )
2        then   discard burst m;
3        else   admit burst m;
4               A := A + λ_m;
```

- Enabling condition: departure of last packet of burst $m$

$A := A - \lambda_m$;

[4] The exact value is carried in the first packet of the burst.

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

### 4.2 Flow Level Admission Control

Burst level admission control insures that channel capacity is not exceeded at any time by discarding bursts if necessary. This ensures that delay guarantees can be provided to bursts that are accepted by the node [11]. Since all burst losses are due to burst level admission control, the goal of our flow level admission control becomes very specific: to allow as much overbooking as possible while bounding the probability that the channel's unallocated capacity is not sufficient for a newly arrived burst — denoted as the overflow probability — by $p$. The overflow probability and statistical multiplexing gains are closely related. In particular, if two channels 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. Therefore, our flow level admission control exploits statistical multiplexing gains in an explicit manner.

#### Flow level admission control condition

Assume that the statistical service of the node is currently shared by a set of $M$ flows (indexed by $1, 2, ..., M$). Each flow, say $i$, supplies the following set of traffic (TSpec) parameters when making a reservation with the node: sustained bit (or packet) rate $SR_i$, peak rate $PR_i$, and rate variance $RV_i$. At any time, with burst-based rate allocation, at most one burst in each flow has its reserved rate allocated. Denote $\lambda_i(t)$ the reserved rate for the burst of flow $i$ that is either allocated a rate or being processed by burst level admission control at time $t$. ($\lambda_i(t) = 0$ if there is no such burst for flow $i$ at $t$.)

In [19], we derived, using a generalized venison of central limit theorem, the following sufficient condition to bound the burst loss rate at the node below $p$:

$$\frac{Z_p}{Z} \leq 1, \tag{9}$$

where $Z_p$ is the $(1 - p)$ percentile of the standard normal distribution, and

$$Z = \frac{C - \sum_{i=1}^{M} E[\lambda_i(t)]}{\sqrt{\sum_{i=1}^{M} Var[\lambda_i(t)]}}. \tag{10}$$

We refer to the value of $(Z_p/Z)$ as the Statistical Multiplexing Intensity (SMI) of the channel. It should never exceed the threshold of 1 to bound the burst loss rate at the channel below $p$. In practice, it is

difficult to obtain the exact value of $Z$. However, $Z$ can be estimated as follows:

$$\hat{Z} = \frac{C - \sum_{i=1}^{M} SR_i}{\sqrt{\sum_{i=1}^{M} RV_i}}. \tag{11}$$

In summary, our admission control algorithm accepts a new flow only if the following condition is not violated:

$$\frac{Z_p}{\hat{Z}} \leq 1. \tag{12}$$

Note that the source of flow $i$ may not have a good estimate of $RV_i$ at the time of connection setup. In such a case, $RV_i$ is upper bounded by $SR_i \cdot (PR_i - SR_i)$, which can be used as a pessimistic estimate.[5]

### Algorithm specification of flow level admission control

The variables $B$ and $V$ are used to store respectively the unallocated channel capacity and the aggregate rate variance of admitted flows. Initially $B$ is set to $C$, and $V$ is set to 0. We assume that if a source does not have a good estimate of $RV$ at the time of connection setup, it will let the network know by setting $RV$ to $-1$.

- Enable condition: receiving connection request from flow $i$

*Flow_Admission_Control ( $i$ )*

1 **if** $(B - SR_i \leq 0)$
4 **then** reject the flow;

5 **if** $(RV_i = -1)$
6 **then** $RV_i := SR_i * (PR_i - SR_i)$;
7 $SMI := Z_p * (sqrt(V + RV_i)/(B - SR_i))$;
8 **if** $(SMI \leq 1)$
9 **then** accept the flow;
10 $B := B - SR_i$;
11 $V := V + RV_i$;
12 **else** reject the flow;

[5] On-line measurement of $SR$ and $RV$ can be performed for admitted flows to improve the performance of flow level admission control. We are currently investigating such techniques.

### 4.3 Empirical Evaluation of Admission Control Algorithm

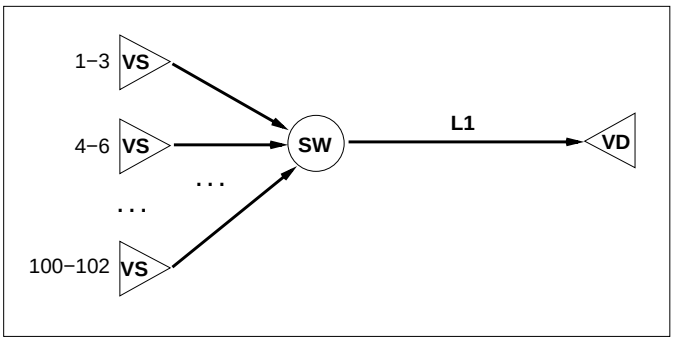


Figure 5: Simulated network for evaluation of admission control

We have evaluated our two level admission control algorithm by performing a set of simulation experiments [19, 20]. The simulation configuration is shown in Figure 5. The nodes labeled by VS denote video sources, and VD their destination. Each video source generates 53-byte packets from a trace file obtained from a MPEG video sequence [19], and each MPEG frame (or picture) is considered an ADU.

The two level admission control algorithm is implemented for channel L1 with a target burst loss rate of $p$. Each video source makes a reservation with L1, and starts sending packets to the network only after the reservation is successful. Packets are scheduled based on their virtual clock values [22]. The channel capacity of L1 as well as the value of $p$ were varied in different experiments. We ran each experiment for 10 seconds of simulated time.

#### Channel utilization

In Figure 6, we plot the channel utilization as a function of the target picture loss rate $p$. 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 non-zero 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 Mbps. This is because the improvement is due to statistical multiplexing gains, which are larger with more flows sharing the channel.

#### Actual picture loss rate

For the channel utilization gain to be meaningful, the actual burst (picture) loss rates in the experiments cannot be much larger than their respective target values. In Figure 7, we compare the actual picture

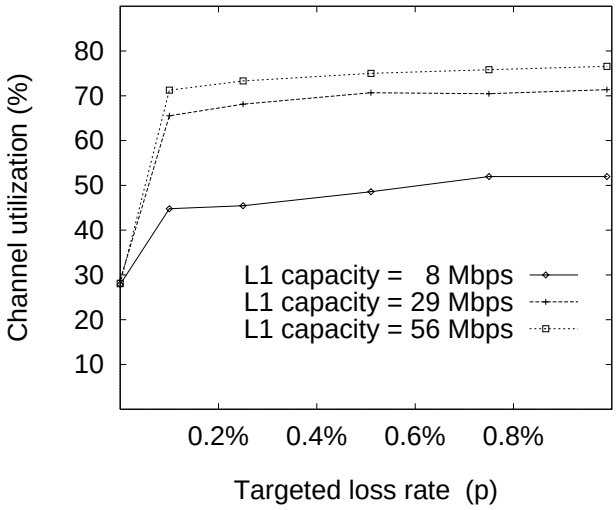


Figure 6: Channel utilization improvement

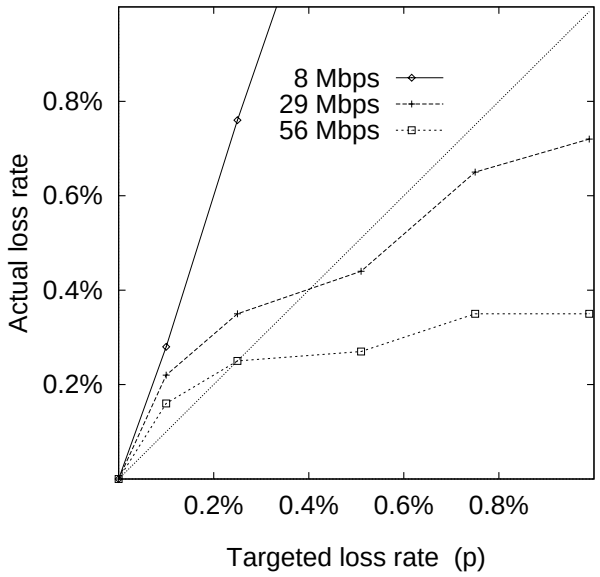


Figure 7: Actual vs. target picture loss rate

loss rate in each experiment, averaged over five simulation runs using different random seeds, with the target value. From the figure, we conclude 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 Mbps.) This agrees with our analysis; the larger the number of flows sharing the channel, the better approximation based on the central limit theorem. Note that the solid 45 degree line represents perfect prediction by the central limit theorem.

# 5 Loss Management Techniques

As discussed in the previous section, admission control can be used at a channel to bound the overall ADU loss rate below a specified value. However, in order for such a statistical service to be viable, considering the fact the service will be shared by many flows, the issue of loss distribution must be addressed. In particular, the ADU losses should be distributed evenly among all flows subscribing to the service and uniformly over the duration of each flow.

We have developed loss management techniques, specifically simple modifications to the admission control algorithm, to enable the channel to anticipate and distribute ADU losses [19].[6] Beside fair distribution of losses, such *active* loss management was also motivated by the fact that to many applications, some of their ADUs are more important than others. For example, to an MPEG decoder, I frames are more important than either P or B frames. With our traffic model, applications can easily mark important ADUs and request that the network give priority to them. Therefore, it is desirable that loss management within a node can facilitate protection of high priority ADUs marked by applications.

In the remainder of this section, we will present our loss management techniques, which are based on selective early ADU discard (SEAD) [19]. Similar to early packet discard proposals [15, 16], SEAD achieves the goals of fair loss distribution and protection of high priority ADUs by taking early control actions. Specifically, a trigger point is set at $\alpha C$, where $0 < \alpha < 1$, and control actions are triggered whenever the aggregate rate allocated by the channel exceeds $\alpha C$.

## 5.1 SEAD-1

For the first technique (named SEAD-1), the flow level admission control algorithm remains the same as the one described in Section 4.2. The burst level admission control algorithm is modified to implement SEAD. In particular, there are two control actions. The first involves discarding unmarked ADUs to protect high priority ADUs, and the second makes use of binary counters, one for each flow ($c_i$ for flow $i$), to prevent a flow from losing consecutive ADUs. Below is a specification of the modified burst level admission control algorithm. Initially the binary counter of each flow is set to zero. $f(m)$ denotes the index of the flow to which burst $m$ belongs.

*Burst_Admission_Control ( $\alpha$, $m$ )*

1 **if** ($A + \lambda_m > C$ )
2 **then** discard burst $m$;

[6] We do not consider losses due to link or transmission errors because the probability of their occurrence is extremely small with today's hardware, and they can be dealt with by link layer protocols.

```
3       else    if (A + λ_m > αC and burst m not marked with high priority)
4               then    if (c_f(m) = 0)
5                       then    discard burst m;
6                               c_f(m) := 1;
7                       else    admit burst m;
8                               A := A + λ_m;
9                               c_f(m) := 0;
10              else    admit burst m;
11                      A := A + λ_m;
```

Note that in SEAD-1, flow level admission control is not modified. The advantage is that high utilization can be maintained at the same time that high priority ADUs have a loss rate much smaller than $p$. However this advantage does come with a price. Namely, those unmarked ADUs can experience more losses. In [19], we showed that the overall ADU loss rate is still bounded with SEAD-1, albeit by a value a little larger than $p$.

We have evaluated the performance of SEAD-1 by simulation experiments. The simulation configuration shown in Figure 5 was used. The channel capacity of L1 is 48 Mbps. For the experiments, I frames are marked high priority, and all other frames are unmarked. A pair of experiments, labeled BASE and SEAD-1, were performed for every simulation run, and in each run a different random seed was used. The original admission control algorithm (i.e. without using SEAD) is used by L1 in BASE experiments. Table 1 contains the result from a typical simulation run. The result shows that by using SEAD-1, the amount of high priority ADU losses is reduced by more than 50%, and the losses are distributed more evenly among flows (as indicated by the fact that the highest loss rate for one flow is much smaller).

| | Channel | ADU losses | | Highest Loss Rate |
|---|---|---|---|---|
| | Util | Total | High Pri | for One Flow |
| BASE | 69 % | 16 | 8 | 2.1% |
| SEAD-1 | 69 % | 54 | 3 | 1.36% |

Table 1: L1 capacity = 48 Mbps, $p = 0.2\%$, $\alpha = 0.95$

SEAD does add some processing cost to the admission control algorithm. However, since the extra processing is performed on a per ADU basis, the associated cost is not significant because the average ADU size is quite large compared to the average packet size. Note that because of burst level admission control, the units for statistical multiplexing become ADUs instead of packets. That explains why

simple binary counters are effective in spreading out ADU losses among flows as we have observed in the experiments.

### 5.2 SEAD-2

The second technique (named SEAD-2) is almost the same as SEAD-1 except that it requires a small but significant modification to the flow level admission control algorithm. Specifically, $\alpha C$ is used in place of $C$ in flow level admission control, i.e. $B$ is initialized to $\alpha C$ instead of $C$ in the algorithm specification given in Section 4.2.

Compared to SEAD-1. the maximum channel utilization achievable with SEAD-2 is lower. However, the difference can be made insignificant by choosing a $\alpha$ value close to 1, say 0.95. Our experimental results indicate that such a $\alpha$ value is sufficient for reaping all the benefits of SEAD-2, namely:

- loss rate of high priority ADUs significantly reduced,
- loss rate of all ADUs below target value, and
- losses distributed more evenly among flows.

We have evaluated SEAD-2 using simulation experiments. Table 2 contains the result of a typical simulation run. The result shows that while the achieved channel utilization is a little lower, both the amount of high priority ADU losses and the highest loss rate for a flow are significantly lower when SEAD-2 is used.

| | Channel Util | ADU losses | | Highest Loss Rate for One Flow |
|---|---|---|---|---|
| | | Total | High Pri | |
| BASE | 71% | 34 | 17 | 3.1% |
| SEAD-1 | 71% | 86 | 10 | 2.7% |
| SEAD-2 | 67.6% | 28 | 4 | 1.03% |

Table 2: L1 capacity = 48 Mbps, $p = 0.5\%$, $\alpha = 0.95$

## 6 Conclusions

Encapsulation and segmentation of application data cause the following tension between the network and an application: While the network has been designed to optimize the performance of packets, an application emphasizes on the performance of its data units rather than that of individual packets. To

address this problem, we propose to consider application-level requirements in design of networks that provide delay and loss guarantees. By doing so, we are able to develop a novel network architecture that in many aspects is more efficient than ones motivated exclusively The centerpiece of our design is a traffic model that enables delivery of application specific information to the network. The model has made possible efficient network techniques for providing delay guarantees, guaranteeing the loss rate of ADUs, and managing application data losses at a channel to achieve fairness.

## References


[1] Jon C.R. Bennett and Hui Zhang. Hierarchical packet fair queueing algorithms. In *Proceedings of ACM SIGCOMM '96*, pages 143–156, Stanford University, CA, August 1996.

[2] I. Cidon and R. Guerin. An investigation of application level performance in ATM networks. In *Proceedings of IEEE INFOCOM '95*, pages 129–137, Boston, MA, March 1995.

[3] David Clark and David L. Tennenhouse. Architectural considerations for a new generation of protocols. In *Proceedings of ACM SIGCOMM '90*, pages 200–208, August 1990.

[4] T. Connolly, P. Amer, and P. Conrad. An extension to TCP: Partial order service. Technical report, November 1994. Internet Draft, *RFC 1693*.

[5] 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.

[6] D.L. Tennenhouse *et al*. A survey of active network research. *IEEE Communications*, January 1997.

[7] Sally Floyd and Van Jacobson. Link-sharing and resource management models for packet networks. *IEEE/ACM Transactions on Networking*, 3(4):365–386, August 1995.

[8] R. Guerin, H. Ahmadi, and M. Naghshineh. Equivalent capacity and its application to bandwidth allocation in high-speed networks. *IEEE Journal on Selected Areas in Communications*, 9(7):968–981, 1991.

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

[10] Sugih Jamin, Peter Danzig, Scott Shenker, and Lixia Zhang. A measurement-based admission control algorithm for integrated services packet networks. In *Proceedings of ACM SIGCOMM '95*, pages 1–13, August 1995.

[11] Simon S. Lam and Geoffrey G. Xie. Burst scheduling networks. *Performance Evaluation*, 1997. An abbreviated version in *Proceedings of INFOCOM '95*, April 1995. Available from *http://www.cs.nps.navy.mil/people/faculty/xie/pub*.

[12] Simon S. Lam and Geoffrey G. Xie. Group priority scheduling. *IEEE/ACM Transactions on Networking*, April 1997. An abbreviated version in *Proceedings of INFOCOM '96*, march 1996. Available from *http://www.cs.nps.navy.mil/people/faculty/xie/pub*.

[13] Richard G. Ogier, Nina T. Plotkin, and Irfan Khan. Neural network methods with traffic descriptor compression for call admission control. In *Proceedings of IEEE INFOCOM '96*, pages 768–776, San Francisco, CA, April 1996.

[14] 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.

[15] A. Romanow and S. Floyd. The dynamics of TCP traffic over ATM networks. In *Proceedings of ACM SIGCOMM '94*, London, England, August 1994.

[16] J.S. Turner. Maintaining high throughput during overload in ATM networks. In *Proceedings of IEEE INFOCOM '96*, pages 287–295, San Francisco, CA, April 1996.

[17] Geoffrey G. Xie and Simon S. Lam. Delay guarantee of Virtual Clock server. *IEEE/ACM Trans. on Networking*, 3(6):683–689, December 1995.

[18] Geoffrey G. Xie and Simon S. Lam. An efficient adaptive search algorithm for scheduling real-time traffic. In *Proceedings of 1996 IEEE International Conference on Network Protocols (ICNP '96)*, pages 14–22, Columbus, Ohio, October 1996.

[19] Geoffrey G. Xie and Simon S. Lam. Admission control and loss management for an application-level statistical service. Technical Report CS-97-Xg-1, Naval Postgraduate School, January 1997. Available from *http://www.cs.nps.navy.mil/people/faculty/xie/pub*.

[20] Geoffrey G. Xie and Simon S. Lam. Real-time block transfer under a link sharing hierarchy. In *Proceedings of IEEE INFOCOM '97*, Kobe, Japan, April 1997.

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

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