Design, Implementation, and End-to-End Evaluation of a Measurement-Based Admission Control Algorithm for Controlled-Load Service

Cheng Jin Sugih Jamin*
Electrical Engineering and Computer Science Department
University of Michigan
Ann Arbor, MI 48109-2122
{chengjin,jamin}@eecs.umich.edu

May 29, 1998

Abstract

The Internet Engineering Task Force (IETF) is considering the adoption of controlled-load service as a standard for relaxed real-time services on the Internet. The ability of relaxed real-time services to achieve high level of utilization and still provide acceptable quality of service depends crucially on the use of measurement-based admission control (MBAC) algorithms. MBAC algorithms admit new connections based on measured bandwidth usage of existing connections. In this paper we describe the design and implementation of an MBAC algorithm for controlled-load service on a router running FreeBSD 2.2.2 kernel. We then present an end-to-end evaluation which shows that controlled-load service, supported by an MBAC, reduces the packet loss rate and delay variation of real-time traffic while maintaining network efficiency.

1 Introduction

With the growing popularity of the Internet, its use as a medium for real-time audio and video traffic has seen a rapid rise. Currently the Internet offers only a single level of best-effort (BE) service, which, though adequate for traditional traffic such as email and file transfer, does not provide service guarantees to preserve the fidelity of real-time audio and video sessions. While it has been largely accepted that real-time applications on the Internet must adapt to available bandwidth [AMK97], and adjust their quality accordingly, efforts to quantify the effect of network congestion on user-perceived quality has seen limited results. The authors of [MKT95, RKTS94] studied the design of adaptive algorithms given a packet stream with varying inter-arrival times. Aside from [BBS98] and the present paper, we are not aware of any other work that studied the effect of network design on adaptive playback applications.

Two parallel efforts are on-going at the Internet Engineering Task Force (IETF) to introduce service differentiation to the Internet that could potentially preserve the fidelity of real-time sessions in times of network congestion. *Differentiated Services* aims to allow network providers to sell virtual links of guaranteed rate ("premium service") to organizations such as other providers or institutional customers [CW97, NJZ97]; *Integrated Services* aims to allow per-flow negotiation of quality of service on a shared link. The integrated services working group at the IETF is working towards the standardization of two types of real-time service: Guaranteed Service [SPG97] and Controlled-Load (CL) service [Wro97]. In this paper we consider the use of CL service in a differentiated-services environment, for example, when an institution purchases a virtual link under the differentiated services archi-

*This research is supported in part by funding from MCNC/MCI and NSF Career Award ANI-9734145 and by equipment donation from Digital Equipment Corp., Sun Microsystems Inc., and Intel Corp.

tecture to carry its CL traffic.

Controlled-load service provides a relaxed real-time service with low loss and low delay without guaranteeing a delay bound. Given a fixed rate allocated to serve controlled-load traffic, admission control must be employed to control admittance to the service. A simple algorithm that ensures the sum of reserved rates to always be below allocated rate is an effective, though not always efficient, admission control algorithm. We refer to this as the Simple Sum Admission Control algorithm (SSAC) in the remainder of this paper. A Measurement-Based Admission Control algorithm (MBAC) controls admittance based on the declared rate of a flow and measured usage of existing flows. In this paper we show that the application of MBAC to control access to a virtual link purchased under the differentiated services architecture allows CL traffic to take advantage of statistical multiplexing, increasing utilization of the virtual link, without perceptible degradation in the quality of service received.

Our previous work on MBAC support for CL service has concentrated on studying the utilization gain achievable compared against that under the simple sum algorithm. In this paper, we look at the end-to-end service seen by adaptive real-time applications subscribing to CL service when an MBAC algorithm is used. We show that CL service with MBAC has the following characteristics:

- Controlled-load traffic receives "better" service than best-effort traffic, by which we mean that CL traffic sees lower packet loss rate and delay variation.
- When admittance to controlled-load service is controlled by a properly working admission control algorithm, best-effort traffic does not experience excessive loss in the presence of controlled-load traffic.
- MBAC allows higher utilization of the bandwidth allocated to controlled-load traffic than under SSAC with comparable application-perceived service in terms of loss rate and delay variation.

Extant studies of measurement-based admission control algorithm have mostly been based on formal and numerical analysis or simulations. All the results reported in this paper are from experiments with our in-kernel implementation of Measurement-based Admission control for Controlled-load sErvice (MACE). We describe the design and implementation of MACE in Section 3. In Section 4 we describe our experimental setup for results reported in Section 5. In both Sections 3 and 4 we describe some caveats on the implementation of MACE and the experimental setup on a network testbed. MACE has been implemented on Intel's PentiumPro machines running FreeBSD 2.2.2. We will make this implementation available on the Web. We close this paper by summarizing our results and lessons learned in Section 6. We start our discussion with an overview of current real-time adaptive playback applications in the next section.

2 Adaptive Playback of Real-time Applications

A typical real-time session today involves transmitting digitized audio/video samples from senders across the Internet to receivers. The current real-time applications send packets only when new data, e.g. human voice and body movement, is detected. Consequently, real-time packets tend to arrive at a receiver in bursts. We call the time between when a packet is transmitted by the sender and when it is delivered ("played back") to the user, the packet's *playback delay*. Without buffering at either the sender or receiver, a packet's playback delay is then the transmission, propagation, and queueing delays the packet sees on the network. To ensure smooth playback of a burst of packets, receivers can buffer incoming packets and smooth out variations in delay (jitter) that packets experience in the network by delaying the playback of the leading packet of the burst. Adaptive playback applications change the playback point for each burst of incoming packets, i.e. based on observed packet delay variance, the playback algorithm can adjust the playback delay of the first packet of the incoming burst.

It has become widely accepted that good programming practices of real-time Internet applications should incorporate some form of adaptive playback [AMK97]. The adaptive nature of such

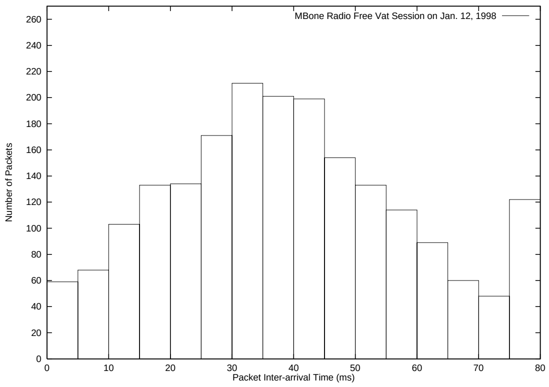

Figure 1: Vat Session with Poor Audio Quality.

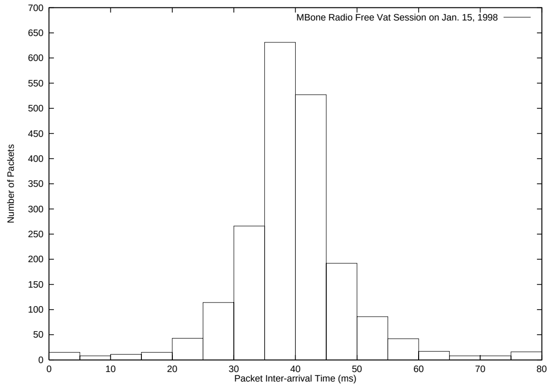

Figure 2: Vat Session with Better Audio Quality.

applications makes the specification of a metric to quantify the effect of network congestion on their performance that much harder to define. In [BBS98], the authors compute the *utility function* of adaptive applications as a function of delay variation. Users of an adaptive application gain utility from the use of the application if delay variation stays below a certain user-specified subjective threshold. Users get lower utility from signal reconstructed from packets with delay variation beyond this threshold. Delay variation aside, packet loss also effects signal reconstruction. Adaptive applications can tolerate some loss in their data stream; however there is no consensus to how much loss these adaptive applications can tolerate, though a low loss rate is generally desirable.

Figure 1 shows a trace of 2000 packets from a real-time multicast audio session on the MBone[1] captured with the `tcpdump` packet capture tool. The session captured was a continuous 64kbps stream of 320-byte UDP packets (excluding protocol headers) generated at a fixed 40ms inter-arrival time. The observed per-packet delay at the receiver is highly variable, as shown by the wide spread of the inter-arrival times in the figure. While the audio tool used, *vat*, was able to adapt and play the received audio stream, we subjectively found the quality of the resulting audio unsatisfactory. Figure 2 shows another trace of packets from the same audio session captured at a later date. Note that the delay variation has a much tighter spread compared to that in Figure 1. The tighter inter-arrival times enabled a much smoother playback of the received audio stream.

In the remainder of this paper, we study the end-to-end delay variation and loss rate seen by various flows under different network resource allocations. Lacking a quantitative metric to judge the services received, we can only provide comparative statements on how the delay distribution received under one service is tighter than that under another service. Given the subjective nature of perceived application performance, we do not make any claim that the difference in delay variation seen will translate into perceptible application performance.

3 Design and Implementation

This section describes in detail the design and implementation of our Measurement-based Admission control for Controlled-load sErvice module (MACE) in FreeBSD 2.2.2 kernel. Before we dive into the detail of MACE implementation, we first list our implementation guidelines:

- MACE implementation should provide only minimal service necessary to give priority to CL class traffic. We do not optimize our design for any particular transport protocol or specific application.
- MACE implementation should be modular such that each of its components can be replaced by a different implementation.
- The presence of MACE should be transparent to the rest of the system.

[1]The MBone is a virtual network on the Internet that supports multicast delivery.

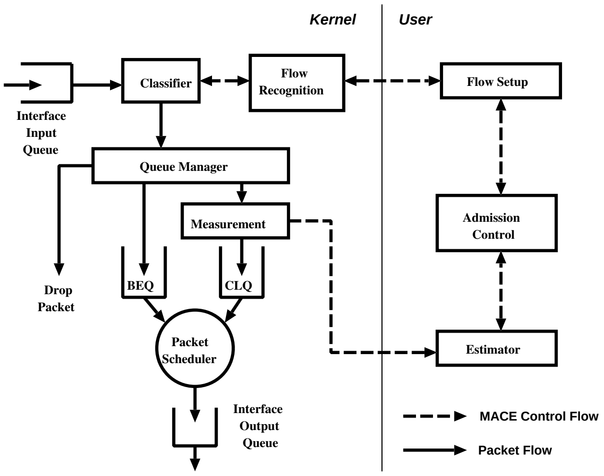

Figure 3: Overview of the Implementation

Figure 3 shows the schematic of our MACE implementation, which includes eight components. Four components: the packet classifier, the queue manager, the measurement unit, and the packet scheduler must be incorporated into the kernel because they are in the critical path of packet forwarding. The flow recognition unit promotes flows from BE class to CL class. It examines each packet that is not yet classified as a CL packet and promotes a flow if certain criteria are met. The flow recognition unit is placed in kernel to minimize the per packet processing time. The other three components, the flow setup unit, the estimator, and the admission control unit may be implemented as user level processes.

We focus our design on the output end of the packet forwarding path of a router. A two level priority queue is implemented to support CL service in addition to BE service available in the traditional one queue implementation. MACE implementation classifies and stores CL packets into a separate storage from BE packets. MBAC is applied to limit the aggregate CL traffic bandwidth usage to the allocated value. Our implementation does not use an explicit reservation protocol, e.g. RSVP for flow setup signalling; rather, network administrators use the flow setup unit to specify the criteria for flow setup. A per flow state is maintained for each recognized flow as in RSVP.

In the following paragraphs, we give a detailed description of each component of MACE.

Packet Classifier. The packet classifier determines the type of service each incoming packet receives. The packet classifier matches each incoming packet against a set of flow descriptors in the classifier table. If a match is found, the packet is considered a CL packet; otherwise, it is considered a BE packet. For the purposes of this paper, we use a single hash table for flow descriptor lookups. Assuming the network carries only IPv4 packets, we use the $<$source address, source port, destination address, destination port$>$ fields of the IPv4 packet header as our flow descriptor. Our implementation allows for the use of the other fields in the IPv4 header, but they are not considered in this paper.

Queue Manager. The queue manager maintains a two-level strict priority queue, with CL traffic at the higher level. CL packets are therefore isolated and protected from BE packets.

The current FreeBSD 2.2.2 kernel implements a single interface queue of 50 packets for each Ethernet interface; we replace this single interface queue by our two-level priority queue. The total size of the priority queue is 50 packets. In the absence of CL traffic, a router with MACE implementation behaves exactly like a traditional router providing only BE service. When CL traffic is present and the MACE queue is filled, incoming BE packets are dropped unconditionally. CL packets are not dropped until the number of CL packets in the MACE queue reaches 50. We originally implemented a system with separate CL and BE queues of 50 entries each. This design was abandoned because the comparison is unfair in the sense that the queue size was twice that of the original FreeBSD kernel.

Measurement. The measurement unit keeps a counter for the number of bits transmitted on each network interface. Periodically, an average bandwidth usage is computed from this counter, and the counter is cleared. The counter is updated whenever a new packet is enqueued. Counting when packets are enqueued gives us the arrival rate, which could be higher than the service rate of the outgoing link. This makes MACE unit more conservative.

Packet Scheduler. The scheduler ensures that CL packets always get priority over BE packets. The packet scheduler is called by the device independent network output function and the device interrupt handler routine. As long as there are CL packets queued, the scheduler will dispatch the packet at the head of the CL queue; otherwise the packet at the front of the BE queue is dispatched only when the output interface queue is empty.

Flow Recognition. The flow recognition unit detects potential CL connections and attempts to add each one to the flow descriptor table of the classifier. Aside from the table used by the classifier, the flow recognition unit also maintains a separate flow recognition table to keep track of potential CL connections. This flow table is similar to the classifier flow table, with an additional packet counter field used to record the number of packets sent by a CL candidate. When the packet counter exceeds a predefined threshold, the flow recognition unit attempts to move the new flow to the CL service class. At this point, admission control must be performed to ensure that the addition of this flow will not violate the allocated bandwidth for CL class. If admitted, the flow descriptor for this flow will be added to the classifier's flow descriptor table. Since a flow's lifetime is generally not known *a priori*, MACE relies on a timeout mechanism to remove stale flows. If an admitted CL flow does not inject any packet into the network within a timeout period, we declare the flow "stale" and remove it from the classifier flow descriptor table.

Flow Setup. The flow setup unit assists the flow recognition unit in building the flow table. It also provides the admission control unit with a bandwidth usage estimate of each new flow.

To aid the flow recognition unit, the flow setup unit uses a list of descriptors of applications that may receive CL service. Each descriptor contains a port number and user specified token bucket parameters. This list is stored as a plain text file which network administrators can easily modify. The flow setup units export the list of port numbers that the flow recognition unit should monitor. The flow recognition unit keeps the list in kernel to speed up the recognition process. Whenever a CL flow is recognized (by having a specific port number and number of packets above the threshold), the flow recognition unit calls the flow setup unit. The flow setup unit finds the bandwidth usage estimate of the new CL flow from the list of descriptors and invokes the admission control unit.

Admission Control. The admission control algorithm used in this paper is the one studied in [JSD97]:

$$v\mu > \widehat{r} + \widehat{\nu} \tag{1}$$

where μ is the bandwidth allocated to CL traffic, v the target utilization, $\widehat{r}$ a rate estimate of the new connection requesting CL service, and $\widehat{\nu}$ the measured aggregate bandwidth of existing CL connections. A detailed explanation of the algorithm and recommendations on parameter choices can be found in [JSD97].

Estimator. The estimator computes the average bandwidth usage. It periodically samples the counter maintained by the measurement unit. We use the same moving time window mechanism presented in [JSD97] to obtain a bandwidth usage estimate from bandwidth averages. This moving time window mechanism uses a fixed window size expressed in number of samples. The maximum bandwidth during each measurement window is recorded by the estimator. The bandwidth usage estimate is updated on three occasions. At the end of a measurement window, we update the estimate to be the recorded maximum sample. Whenever an individual sample exceeds the estimate, we immediately update the estimate with the new sampled value. We also update the estimate when a new flow is admitted by adding the flow's reserved rate to the current estimate.

3.1 Caveat Implementor

Next, we present two caveats while implementing MACE. The first one is deciding where in kernel to implement CL service support, and the second is in deciding how to implement the packet scheduler.

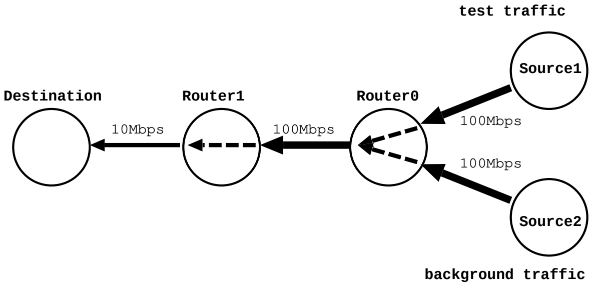

Figure 4: Experiment Setup

We want to implement CL service support in the critical path of packet forwarding to achieve efficiency and speed. There are two possible places to implement MACE, the packet input queue and the packet output queue. We first experimented with adding support for CL service at both the input queue and the output queue. We found out that the normal processing of input packets is done by Ethernet device interrupt handler, which must be fast to keep up with packet arrivals. Adding CL service support at the input queue slows down the interrupt handler because each packet must now be classified and put into the proper queue. The slow-down of interrupt handler causes more packets to be dropped at a busy router since the possibility of overrunning the Ethernet interface buffer is now higher. This is a well known problem of input-queueing [Kes97]. Furthermore, we observed little input queueing at the IP level, hence we do not add CL service support at the input queue. On the other hand, we observed more packet queueing at the output queue, and output queueing is done at an interrupt level lower than the device interrupt handler's so classifying and queueing each packet does not decrease the kernel's efficiency in receiving incoming packets. In our final implementation, MACE is done only at the packet output queue.

The second caveat comes from the implementation of the packet scheduler. In our implementation, outgoing Ethernet packets are always queued first in the MACE queue. Packets are then dispatched to the output interface by the scheduler according to their classification. In the first version, we dispatched BE packets to the interface as long as the CL queue was empty. This seems reasonable at first, but we then discovered that it was possible for BE packets to get queued on the device interface queue inside the kernel. On a busy router, the interface queue almost always contains more than one packet; often, the interface queue contains more BE packets than it can transmit between successive CL packet arrivals thus causing CL packets to be queued behind BE packets. As a result, CL packets on a busy router receive no better service. We revised this implementation so a BE packet is dispatched only when the interface queue is empty. A CL packet is dispatched to the interface if one is awaiting service; if no CL packet is awaiting service, then BE packets are dispatched one by one only when the device is ready. This ensures that in the worst case, a CL packet is delayed by at most one BE packet, in addition to CL packets ahead of itself. With this implementation, CL packets are no longer queued behind BE packets and indeed receive better service than BE packets.

4 Experiment Setup

Extant studies on the performance of measurement-based admission control and controlled-load service have mostly been done through theoretical analysis and simulation. In this paper, we look at the performance of MACE on a network testbed.

Our network testbed consists of four 200 Mhz PentiumPro PCs and one 167 Mhz UltraSparc Sun workstation. Each PentiumPro PC is equipped with three 100 Mbps Ethernet interface cards; the Sun workstation is equipped with five 10 Mbps interfaces. The testbed setup is depicted in Figure 4. All nodes in our testbed are connected with point-to-point links. The node labeled `Destination` in the figure is the Sun workstation. A 10 Mbps bottleneck link connects `Destination` and `Router1`. In all our experiments, the node labeled `Router0` never sees congestion; the node labeled `Router1` experiences congestion due to the slower speed of the bottleneck link and is the focus of our study. We send UDP traffic from the two source nodes to `Destination`.

There are two types of traffic in our experiments: CL and BE traffic (referred to as *testing traffic* from here on), whose behaviors we study in the paper, and background traffic, which creates the re-

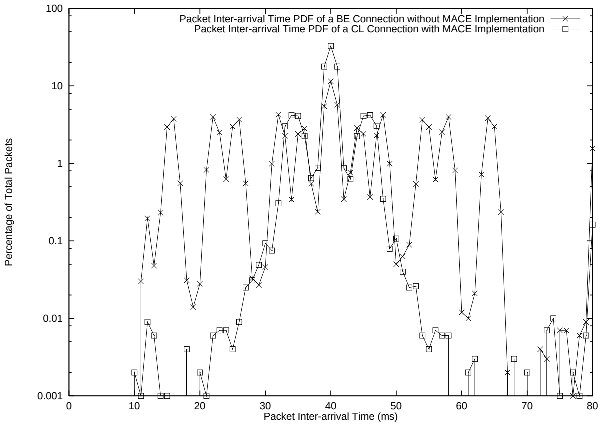

Figure 5: Unexpected Histogram of Packet Inter-arrival Time with Fixed Size Congestion Packet

quired network congestion for our experiments. We found that if CL and BE traffic sources were run side by side on the same workstation as the background traffic sources, background traffic sources could not receive enough CPU time to create the desired level of congestion. Thus, we send the two types of traffic sources from separate nodes in our testbed.

In the next two paragraphs, we describe the creation of different levels of network congestion. In section 4.1, we describe the source models used in generating flows that we want to study, and in section 4.2, the experiment scenarios used for data collection.

To create different levels of network congestion, we use a simple source model that continuously sends UDP packets. The source model has two tunable parameters: the average packet size and the packet generation interval. We change the average packet size to vary the average packet transmission time at a router; and we change the packet generation interval to vary the packet arrival rate at a route. We adjust both parameters to create different levels of network congestion. UDP connections are used instead of TCP connections for background traffic because a TCP connection reduces its sent rate when network is congested, i.e. when packet loss is detected.

While designing and testing our background traffic, we discovered two problems that could have invalidate our results if they were not addressed. We originally used fixed size packets, which gave rise to inter-arrival times of testing traffic as recorded in Figure 5. One would expect the histogram function to be a relatively smooth bell shaped curve; however, we see sharp "peaks" and "valleys" in the graph. Randomization of the congestion traffic packet size eliminates this problem as shown in Figure 9(Section 5.2). We also discovered another potential problem with the FreeBSD kernel. The FreeBSD 2.2.2 kernel implements ICMP source quench. Under congestion, routers with this implementation send messages to senders advising them to reduce their connection rates. As a result of receiving and processing many such packets, the senders are not able to generate packets very fast. This is inconsistent with our objective of creating network congestion, hence we disabled this feature of FreeBSD 2.2.2 in our experiments.

4.1 Source Models for Testing Traffic

In our experiments, we model a testing traffic source as either a constant bit rate (CBR) source or an ON/OFF process with constant peak rate. The CBR source sends fixed size packets at constant inter-arrivals. This is a model we use for pre-recorded music broadcasted over the MBone.

We use two types of ON/OFF sources: one with exponentially distributed ON/OFF times, the other with Pareto distributed ON/OFF times. During each ON period, packets are generated at a fixed rate. The EXP1 model shown in Table 1 is a model for packetized voice encoded using ADPCM at 32 kbps. Pareto distribution is a heavy-tailed distribution that can be described by two parameters: its location and shape (β). A Pareto shape parameter less than 1 gives data with infinite mean; shape parameter less than 2 results in data with infinite variance. The Pareto location parameter can be computed from the formula: $mean \times (shape - 1) \ / \ shape$. Each ON period of the Pareto-ON/OFF model is a Pareto distributed random number. During each ON period, packets are generated at a constant bit rate. The OFF times are also Pareto distributed. We use Pareto ON/OFF models with finite mean and infinite variance. Aggregation of Pareto ON/OFF sources models traffic exhibiting long range dependency (LRD) [WTSW95].

In our experiments, we only use one kind of

Source Model	$\overline{T}_{on}$ (ms)	$\overline{T}_{off}$ (ms)	Peak Rate	β
POO1	400	400	64 kbps	1.5
POO2	40	360	256 kbps	1.1
POO3	313	2925	64 kbps	1.1
EXP1	313	325	64 kbps	
EXP2	20	180	256 kbps	

Table 1: ON/OFF Source Models Used in Our Experiments

CBR source, a source sending at 64 kbps. We use several variations of the two ON/OFF source models as shown in Table 1. Each row in Table 1 corresponds to an ON/OFF source model. The columns in the table show various parameters for each model, with $\overline{T}_{on}$ being the average ON time, $\overline{T}_{off}$ the average OFF time, `Peak Rate` the connection data rate during ON times, and β the shape parameter of Pareto ON/OFF models.

We implement each source model with a program that sends UDP packets. Similar to the generation of background traffic, we can tune the testing traffic for packet generation intervals but fix the packet data size to be a constant 320 bytes. We achieve different connection rates by varying packet inter-arrival times, e.g. for connections with data rate of 64 kbps, we generate a packet every 40 ms, and for connections with data rate of 256 kbps, we generate a packet every 10 ms. In Table 1, the `Peak Rate` refers to the actual data rate, i.e. the amount of data sent in bits per second. The overall bandwidth consumption is higher than the data rate because each packet also carries protocol headers besides data. Thus, during ON times, a connection with 64 kbps peak rate sends 320 bytes of data every 40 ms, with 12 byte RTP header, 8 byte UDP header, 20 byte IP header, 26 byte Ethernet header and trailer. The total size of each packet is therefore $320 + 12 + 8 + 20 + 26 = 386$ bytes. The overall bandwidth consumption of a 64 kbps connection is therefore $386 bytes/40 ms = 77.2 kbps$. Hence, a 64 kbps source sending 320 byte data appears to be sending 386 byte packets to the measurement unit and is considered to be generating 77.2 kbps by the admission control unit.

4.2 Experiment Scenarios

In this section, we describe the four scenarios used in our experiments. In the first and second scenarios, we start the congestion traffic and wait until it fills the pipe, then we start the testing traffic. In the third and fourth scenarios, which we use to investigate MBAC, we report only data collected after a warm-up period of 100 seconds. About 30% of the link bandwidth is allocated to CL traffic.

Before presenting the four scenarios, we list the values used for all MACE system parameters. For flow recognition unit, the packet counter has a threshold of 100 packets for 64kbps traffic and 10 packets for 256kbps traffic. The value of packet threshold changes the number of available flows that can be recognized. We want each CL flow to have statistically significant number of packets before we recognize it as a flow; however, we do no claim that our choices are optimal. We make this a tunable parameter so network administrators can tailor the flow recognition unit to their needs. The measurement unit samples bandwidth usage every 5 seconds, and the estimator computes the bandwidth averages every 100 seconds. The sampling period determines how sensitive the measurement unit is to the burstiness of network traffic. Small sampling period gives more accurate instantaneous bandwidth usage; large sampling period gives smoother averages. The estimator window size affects flow admittance dynamics. An admission control algorithm with small estimator window size could potentially admits a small number of flows continuously, and large window size causes flow arrivals to be bursty. We made the window size to be 20 sampling periods so the estimator has enough samples. Again these parameters are tunable.

In our first scenario, we use 40 64 kbps CBR sources to generate testing traffic. Of the 40 CBR sources, 20 of them request CL service, the rest BE service. We run 20 background traffic sources with 20 ms packet inter-arrival times and randomized data packet size. Each background traffic source has a lifetime of 10000 packets, and each testing connection has a lifetime of 5000 packets; therefore, both background flows and testing flows have the same lifetime of 200 seconds. We record the loss

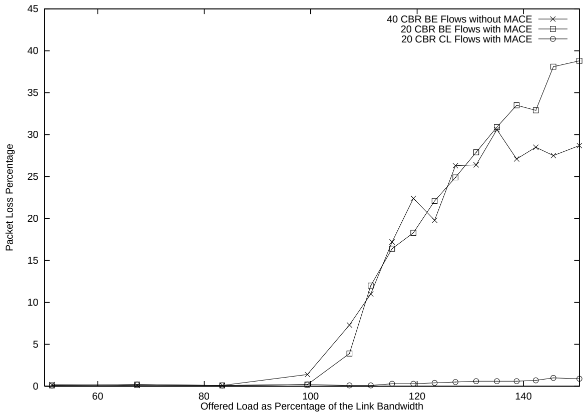

Figure 6: Packet Loss Comparison of CL and BE Connections with CBR Source Model

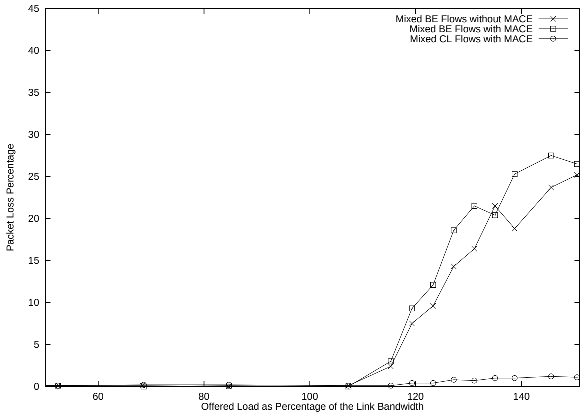

Figure 7: Packet Loss Comparison of CL and BE Connections with Mixed Source Models

rate and packet delay of testing traffic under different levels of network congestion with and without MACE implementation.

We pick the POO1 model in the second scenario to simulate more realistic audio sessions. We use 40 POO1 sources as the testing traffic, with 20 of them requesting CL service, and subject them to the same background traffic as under the first scenario. Again, under MACE, we admit all 20 CL connections. Since each source has a randomized lifetime, repeating an experiment with and without MACE yields two set of connections with different lifetimes. To ease comparison, we decide to use a record-and-replay scheme for this scenario. We first run an experiment for 200 seconds and record the lifetime and starting time of each of the 20 CL connection. We then playback this session with the same starting times and lifetimes with and without MACE.

To test the effectiveness of MBAC, we first use each of the source models in a separate experiment so only homogeneous sources under CL service are examined. In the third scenario, for each CL connection modeled by a source with data rate of 64 kbps, we reserve 77.2 kbps; for a CL connection modeled by a 256 kbps source, we only reserve 154 kbps, which, incidentally, corresponds to the bandwidth requirement of a connection with peak data rate of 128 kbps. We only reserve half the bandwidth requirement of 256 kbps sources to study the effect of statistical multiplexing gain on link utilization under MBAC. A properly working MBAC should bound the amount of CL bandwidth usage under the amount allotted to the CL service class. An MBAC that fails to bound CL traffic consumption to the alloted amount could potentially starve BE traffic. While we provide better service to CL traffic, we must still provide acceptable service to BE traffic.

In each experiment, we run a daemon process that forks out a fixed number of sources requesting CL service and maintains that number of CL connections by creating a new one whenever an old one departs. The number of connections that request CL service is so large that there are always connections waiting to be admitted into the CL class. All of these sources have exponentially distributed lifetimes with an average of 1000 seconds. Each experiment lasts three times the average flow lifetime. We record the bandwidth utilization in each experiment under both MBAC and Simple Sum Admission Control (SSAC).

In the fourth scenario, we mixed the source models of testing traffic. The source models in this scenario are: CBR sources with 64 kbps data rate, POO2, POO3, EXP1, and EXP2 sources. As in the third scenario, we reserve 77.2 kbps for each CL connection modeled by the CBR, POO2, and EXP1 source models, and 154 kbps for each CL connection modeled by the POO3 and EXP2 source models. We use background traffic with offered load of 130% of the 10 Mbps bottleneck link. Connections under this scenario all have average lifetimes of 1000 seconds, and each experiment is run for

3000 seconds.

5 Results

We present the results of our experiments on packet loss rate, packet delay, and MBAC in this section. For packet loss rate and packet delay experiments, we allocate enough bandwidth for CL traffic so MBAC is not used. Later experiments concentrate on MBAC studies. We show that with MACE, traffic under CL service experiences lower packet loss and more uniform packet delay than traffic under the traditional BE service; furthermore, the presence of CL traffic does not cause background traffic to lose excessive number of packets. We also show that under MBAC, the utilization of allocated CL bandwidth is higher than that under SSAC, with comparable delay and loss performance.

5.1 Packet Loss

Most interactive real-time applications designed for the Internet can adapt to varying network load and occasional packet losses. Under heavy network load, when packet loss rate can be higher than 10%, even if the application can reconstruct the signal, the resulting quality is likely to be unacceptable. By putting these interactive traffic into a CL class, we hope to keep the loss rate reasonably low. In this section we present results from experiments conducted under the first and fourth scenarios described in the previous section. We show that loss rates of connections under CL service in both scenarios are much lower than that of traditional BE connections.

We first compare the aggregate loss rate of 64kbps CBR connections under BE service against identical connections under CL service. Data was obtained from an experiment under the first scenario described in the last section. Shown in Figure 6 is the plot of aggregate loss rates of 20 BE connections with MACE, 20 CL connections with MACE, and 40 BE connections without MACE implementation. The x-axis is the average offered load of background traffic expressed as a percentage of the 10 Mbps bottleneck link speed. The y-axis is the packet loss percentage of the respective traffic types.

The loss rate of CL traffic with MACE is significantly lower than that of BE traffic with or without MACE. While BE traffic without MACE experiences loss rate as high as 30%, the worst loss rate of CL traffic is just over 1%. Packet loss of CL traffic occurs primarily during the flow recognition phase when the initial 100 packets from each CL connection are classified as BE packets. It is clear that the presence of a priority scheduler reduces CL packet loss. This improvement can be beneficial especially under highly congested network conditions. We also observe that the loss rate of BE traffic under MACE is not significantly worse than the loss rate of BE traffic without MACE.

We expect real-time audio and video traffic on the Internet to be better modeled by traffic with ON/OFF times instead of CBR traffic [Flo96, GW94, BSTW95]. We conducted an experiment under the fourth scenario with mixed BE and CL traffics modeled by CBR 64 bkps, POO1, POO3, and EXP1 source models. Figure 7 graphs the aggregate packet loss rate of BE and CL connections. The figure confirms that traffic under CL service indeed experiences lower loss rate than under BE service. Figure 8 shows that the packet loss rate of background traffic in this experiment, given the ON/OFF CL sources, is lower than that under the first scenario, where the CL sources generate CBR traffic, by around 5% to 10%.

5.2 Packet Delay

Low packet loss rates alone may not be sufficient to guarantee good audio and video quality on the Internet. Another important factor that influences the quality of real-time traffic is the delay experienced by individual packets. Real-time audio and video applications typically compute playback points based on individual packet delays [MKT95]. Large packet delay variations cause playback points to be postponed, resulting in loss of fidelity. Hence, one of the objectives in introducing CL service is to reduce the delay variations experienced by real-time audio and video traffic.

A straight forward way to compute packet delay is for each packet to carry a timestamp. The sender stamps each packet before transmission. The difference between the value stored in the times-

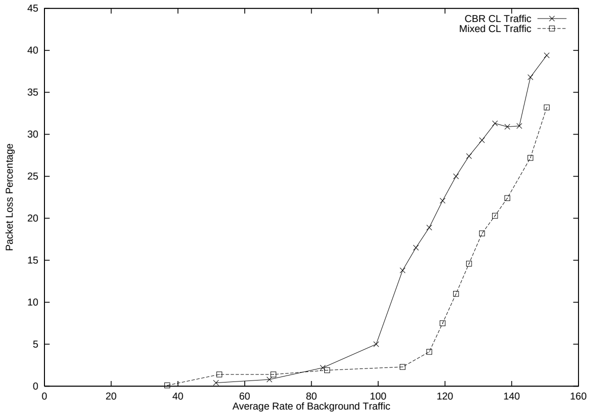

Figure 8: Packet Loss Comparison of Background Traffics with Two Types of CL Traffics

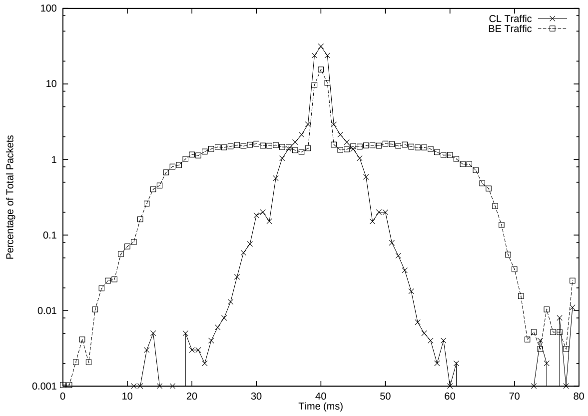

Figure 9: Histograms of Aggregate Packet Inter-arrival Time of CBR BE and CL Traffics

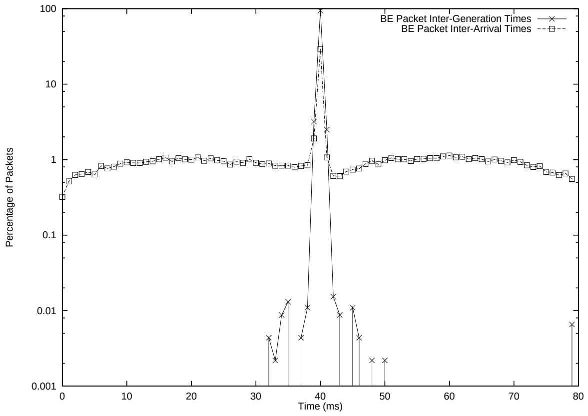

Figure 10: Histograms of Aggregate Packet Generation Times and Inter-arrival Times of BE Connections.

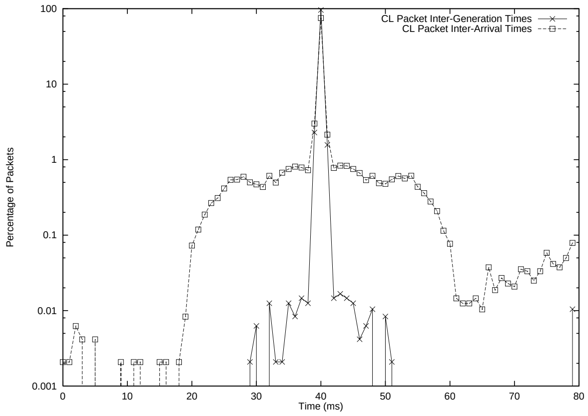

Figure 11: Histograms of Aggregate Packet Generation Times and Inter-arrival Times of CL Connections.

tamp and the time when the packet is received is the end-to-end delay. This calculation requires synchronized clocks at both ends of the transmission. Since we are only interested in delay variations, we only need to look at packet inter-arrival times. Since packets are generated with fixed inter-arrival at the sender, large variations of packet inter-arrival times seen by the receiver imply high delay variations.

In this section, we present results of experiments from the first, second, and fourth scenarios. In the first scenario, we study 64 kbps CBR traffic under CL service; in the second scenario, we study LRD traffic (POO1), and in the fourth scenario, we look at the highly bursty traffic, POO2 and EXP2, under CL service. We show that under CL service, 64 kbps connections all have significantly lower delay variations than under BE service, with 256 kbps connections seeing moderately improved delay variations.

Figure 9 shows the histogram of aggregate packet inter-arrival times for an experiment under the first scenario. The traffic we study is 64 kbps CBR connections under both CL and BE services. On the x-axis, we show the range of inter-arrival times, and on the y-axis, we show the percentage of packets with a particular inter-arrival time (note that the y-axis is in log scale). We removed all packets whose inter-arrival times are greater than 80 ms. Since packets are generated every 40 ms, and queueing delay of 40 ms is not possible in our experiment, the only way a packet can have inter-arrival times greater than or equal to 80 ms is when one or more packets that immediately preceded this one was dropped, in which case, the 80 ms is not a true inter-arrival, and we discount them in our figure. This not only removes the effect of lost packets, but also removes the first packet from each ON time, which has an inter-arrival time that is actually the preceding OFF time. Under CL service, 99% of all packets have inter-arrival times that differ by less than 25% from the ideal inter-arrival time, while only 58% of BE packets are within this range.

Next, we present a back-of-the-envelop analysis of the delay BE traffic sees under congestion. In this experiment, there are 20 background traffic sources, each sending with a peak rate twice that of a testing traffic source. Without MACE, both CL and BE connections are treated as BE traffic. Each background traffic packet has 700 byte data (excluding all protocol headers) plus an exponentially distributed offset with a mean of 100 bytes for an average of 800 bytes per packet. Each testing traffic packet has 332 byte data (without protocol headers). On average, a router sees 2 incoming congestion packets for every 2 BE packets; hence, the average packet data size seen by a router is $800 \times .5 + 332 \times .5 = 566\ bytes$. Together with protocol headers, the average raw packet size is around 620 bytes. A BE packet arriving at a full queue (of 50 packets in this experiment) at a router with 10 Mbps outgoing link experiences a queueing delay of $50 \times 620 \times 8/10,000 = 24.8\ ms$ – a queueing delay that is certainly not negligible considering the packet inter-arrival time of 40 ms. Since the background traffic is sent twice as fast as the testing traffic, the interface queue is filled mostly with packets from these faster connections. An empty entry at the end of the queue has a higher probability of being occupied by a packet from faster connections. Thus packets from slower connections are likely to be queued after many packets from faster connections and have long queueing delays.

Researchers have observed LRD characteristic in Internet audio and video traffic [Flo96, GW94, BSTW95]. To see the effect of LRD, we use the POO1 model from the second scenario. We run an experiment from the second scenario and record flow arrival times and lifetimes of 20 CL connections. We then play the session back with the same flow arrival times and lifetime with and without MACE implementation. In Figures 10 and 11, we show the histogram of aggregate packet generation times at the sender and packet inter-arrival times at the receiver for both CL and BE traffic. Again we remove all packets with inter-arrival time of 80 ms or greater. Independent of objective perceptible difference in quality, we say that based on the Figures 10 and 11, an adaptive playback algorithm will be able to compute a smoother playback function with LRD connections under CL service.

In the fourth scenario, we use more complex traffic sources that include, aside from the simple CBR 64 kbps and EXP1 sources, LRD sources (POO3), highly bursty sources (EXP2), and highly bursty LRD sources (POO2). For this set of data, we show separate plots for connections with 256 kbps peak rate (POO2, EXP2) in Figure 12 and connections with 64 kbps peak rate (CBR, POO3, EXP1) in Figure 13. For 256 kbps connections, CL service provides marginal improvement over BE service in terms of packet delay. The marginal improvement here is the result of connections having fast transmission rate; connections with faster transmission rates fill up the queue with consecutive packets, packets from connections with slower transmission rates are likely to see a queue of packets from connections with faster transmission rates. While the improvement in packet delays of CL traffic over BE traffic is marginal, we note that the loss rates of both types of CL traffic is just under 1% while the loss rate for 256 kbps BE traffic is 20% and 24% for 64 kbps BE traffic.

5.3 Measurement-based Admission Control

In the last two subsections, we studied packet loss rate and packet delay without enabling MBAC on CL traffic; In this section we present our findings on the bandwidth utilization of CL traffic under both MBAC and Simple Sum Admission Control (SSAC). SSAC uses an admission control that limits the sum of reserved peak rates to be below the allocated bandwidth.

We present results of experiments from the third and fourth scenarios. We show that MBAC enables much higher levels of link utilization while maintaining packet loss rate and delay variations

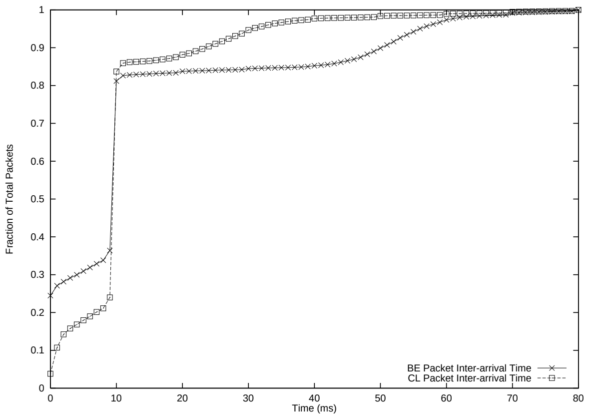

Figure 12: CDFs of Aggregate Packet Inter-arrival Times of Eight 256 kbps connections under CL and BE services

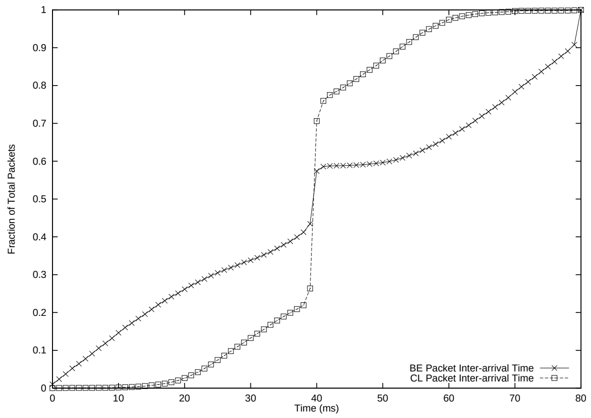

Figure 13: CDFs of Aggregate Packet Inter-arrival Times of Twelve 64 kbps connections under CL and BE services

Source	CL with SSAC		CL with MBAC	
Model	%Utilization	Flows	%Utilization	Flows
POO1	44.5	50	72.5	83
POO2	31	15	61	27
POO3	16	29	46	91
EXP1	49	50	89.6	93
EXP2	29	14	93	47
MIXED	36	19	74	34

Table 2: Comparison of SSAC with MBAC for Single Source Model

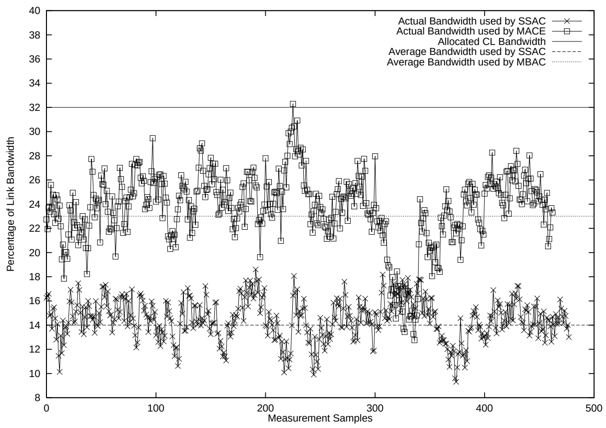

Figure 14: Aggregate Bandwidth Usage in 2500 seconds of POO2 Connections under both MBAC and SSAC.

comparable to that achieved under SSAC.

Table 2 shows that with the same amount of bandwidth allocated to CL traffic, MBAC outperforms SSAC in link utilization for all the source models. The `%Utilization` columns show the average bandwidth utilization under each admission control algorithm; the `Flows` columns show the number of concurrently admitted CL connections under each algorithm. For POO1 and EXP1, we reserved 39% of total link speed, and for the rest of the source models, we reserve 23% of the total link speed. Figure 14 shows a sample path bandwidth utilizations under MBAC and SSAC for POO1 CL connections. As expected, MBAC is very effective when source models are smooth (POO1 and EXP1). When sources are very bursty, such as in the POO3 case, where the peak to average ratio is 10:1, MBAC's effectiveness in increasing link utilization is limited. We also note that physical limitation of the machines prevented us from generating more than 100 offered POO3 flows.

Next, we show that the higher link utilization under MBAC does not drastically degrade the service CL traffic receives. We used mixed traffic with 5 source models: CBR with 64 kbps peak rate, POO2, POO3, EXP1, and EXP2. In the mixed traffic, the number of connections under each source model is roughly the same. We ran the experiment under both SSAC and MBAC with background traffic that is 130% of the 10 Mbps bottleneck link speed. For 256 kbps CL connections, only 8% more of total packets are received under SSAC with inter-arrival times of 10 ms or less as shown in Figure 15. When we compare the rest of CL traffic, the inter-arrival time distributions are almost identical under both SSAC and MBAC. Figure 16 shows 64 kbps CL connections which ran side by side with the 256 kbps connections mentioned earlier. The performance is slightly worsened while the average link utilization under MBAC is 50% higher than under SSAC. Overall, MBAC delivers packet delays comparable to that under SSAC.

6 Conclusion

In this paper, we have presented our design and implementation of Measurement-based Admission Control for Controlled-load service. Results obtained from running experiments on actual network setup show that this implementation successfully accomplishes our design goals. While our experiments are of limited scale, the results give evidences that CL service can provide benefits to real-time audio and video connections on the Internet. We learned some important lessons from this implementation about good experiment setup (Section 3), programming techniques (Section 4), and data analysis.

In summary, we find that traffic under CL service has a much lower loss rate than that under BE service. While providing CL service does increase the packet loss rate of background traffic, this increase is marginal. Under CL service, traffic sees tighter packet inter-arrival time, and this holds true for homogeneous as well as mixed traffic sources. In MACE, CL service is complemented by MBAC so the membership in CL class is controlled, resulting in lower packet loss rate and tighter packet inter-arrival time for admitted traffic. It yields significantly higher bandwidth utilization at the cost of slightly worse packet inter-arrival times.

Acknowledgment

We thank the anonymous reviewers for their comments.

References

[AMK97] E. Amir, S. McCanne, and R. Katz. "Receiver-driven Bandwidth Adaptation for Light-weight Session". *ACM Multimedia*, Nov. 1997.

[BBS98] S. Bajaj, L. Breslau, and S.J. Shenker. "Is Service Priority Useful in Networks?". *Proc. of ACM SIGMETRICS*, 1998.

[BSTW95] J. Beran, R. Sherman, M.S. Taqqu, and W. Willinger. "Long-range Dependence in Variable-Bit-Rate Video Traffic". *IEEE Transactions on Communications*, 43:1566–1579, 1995.

[CW97] D. Clark and J.T. Wroclawski. *An Approach to Service Allocation in the Internet.* Internet Draft, Jul. 1997.

[Flo96] S. Floyd. "Comments on Measurement-based Admissions Control for Controlled-Load Service". Submitted to *Computer Communication Review*, 1996. **URL** ftp://ftp.ee.lbl.gov/papers/admit.ps.Z.

[GW94] M. Garrett and W. Willinger. "Analysis, Modeling and Generation of Self-Similar VBR Video Traffic". *Proc. of ACM SIGCOMM '94*, pages 269–279, Sept. 1994.

[JSD97] S. Jamin, S.J. Shenker, and P.B. Danzig. "Comparison of Measurement-based Admission Control Algorithms for Controlled-Load Service". *Proc. of IEEE INFOCOM '97*, Apr. 1997. **URL** http://netweb.usc.edu/jamin/admctl/info97.ps.gz.

[Kes97] S. Keshav. *"An Engineering Approach to Computer Networking"*. Addison Wesley, 1997.

[MKT95] S.B. Moon, J. Kurose, and D. Towsley. Packet audio playout delay adjustment algorithms: Performance bounds and algorithms. Technical report,

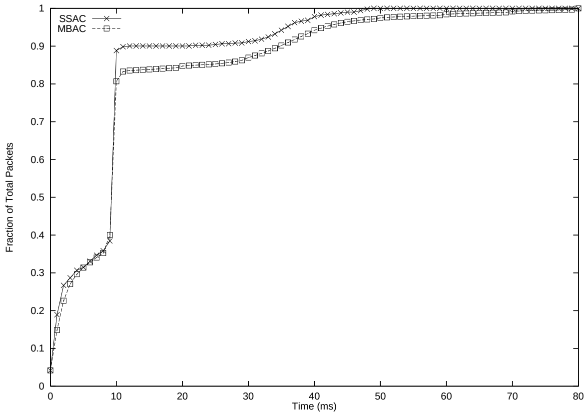

Figure 15: CDFs of Packet Inter-arrival Time under SSAC and MBAC for 256 kbps Connections.

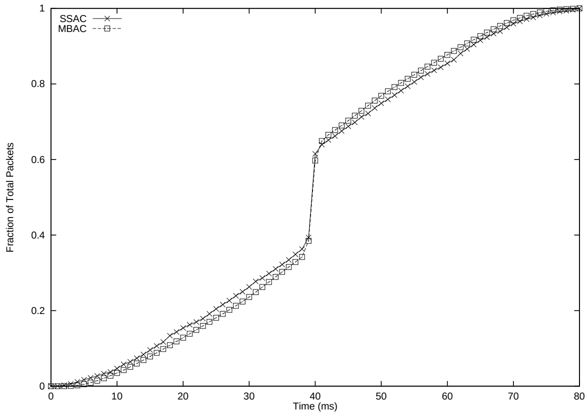

Figure 16: CDFs of Packet Inter-arrival Time under SSAC and MBAC for 64 kbps Connections.

Dept. of Computer Science, Univ. of Massachusetts, Amherst, Aug. 1995.

[NJZ97] K. Nichols, V. Jacobson, and L Zhang. *A Two-bit Differentiated Services Architecture for the Internet*. Internet Draft, Nov. 1997.

[RKTS94] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne. ”Adaptive playout mechanisms for packetized audio applications in wide-area networks.”. *IEEE INFOCOMM*, pages 680–688, 1994.

[SPG97] S. Shenker, C. Partridge, and R. Guerin. Specification of guaranteed quality of service. RFC 2212, Internet Engineering Task Force, September 1997.

[Wro97] J. Wroclawski. Specification of the controlled-load network element service. RFC 2211, Internet Engineering Task Force, September 1997.

[WTSW95] W. Willinger, M.S. Taqqu, R. Sherman, and D.V. Wilson. “Self-Similarity Through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level”. *Proc. of ACM SIGCOMM ’95*, pages 100–113, Aug. 1995.