Reserve-based Disk Admission Control and Bandwidth Scheduling Strategy for Continuous Media Servers

Wonjun Lee
Computer Science Telecommunications
University of Missouri - Kansas City
Kansas City, MO 64110, USA

Jaideep Srivastava
Computer Science & Engineering Department
University of Minnesota - Twin Cities
Minneapolis, MN 55455, USA

Abstract—**In this paper we present a novel admission control algorithm that exploits the degradability property of continuous media applications to improve the performance of the system. The algorithm is based on setting aside a portion of the resources as reserves and managing it intelligently so that the total utility of the system is enhanced. This *reserve-based admission control strategy (RAC)* is a compromise between purely greedy and non-greedy strategy, and it leads to an efficient protocol that improves the performance of the system. While the protocol is simple for admission decision, it also results in better performance for the system by reserving some resources for the important future applications.**

Keywords— **Soft-QoS Framework, Quality of Service, Continuous Media, Disk Admission Control, Resource Scheduling.**

I. INTRODUCTION

The emergence of varied high-speed networked multimedia systems opens up the possibility that a much more varied collection of continuous media applications could be handled in real-time. However, due to the limitation of available resources in such systems, we still need to develop more intelligent mechanisms for efficient admission control, negotiation, resource allocation, and resource scheduling, so that we could optimize the total system utilization. In particular, there has been increased interest in I/O issues for multimedia or continuous media (CM). The goal of conventional disk scheduling policies is to reduce the cost of seek operations and to achieve a high throughput, while providing fair access to every process that seeks its services. The goal of disk scheduling for continuous media is to meet the deadlines of the periodic I/O requests generated by the stream manager to meet rate requirements. An additional goal is to minimize buffer requirements. In order to ensure continuous and stringent real-time constraints of video delivery in CM servers [1], [2], several factors such as disk bandwidth, buffer capacity, network bandwidth, etc., should be considered carefully and should be handled efficiently. The reservations of these resource factors are required for supporting an acceptable level of display quality and for providing on-time delivery constraints. In particular, disk bandwidth constraint may be the most important factor, given that the I/O bandwidth reserved for each stream on disk depends on latency overhead time, transfer time, defined cycle length, and contention of multiple streams. Hence, we should be able to guarantee that the request of each stream can be fairly supported with good disk utilization and server cost-performance.

The problem here is dual (i.e., admission control and disk I/O bandwidth management in CM server systems) to that of network call admission control and dynamic bandwidth management [3]. Admission control in CM servers or video-on-demand systems restricts the number of applications supported on the resources. For example, the applications may be video streams, and the resources used could be connections on the network or on continuous media servers [4], [5]. To handle these issues in CM server systems, we propose an adaptive admission control strategy which achieves better performance than the conventional greedy admission control strategies generally used for CM servers. It recognizes that CM (e.g., video) applications can tolerate certain variations on QoS parameters. It develops an algorithm for sharing processing resources at the server to share available resources effectively among contending streams [6]. The proposed algorithm provisions are for *reclamation* (i.e., scheduler-initiated negotiation) to reallocate resources among streams to improve overall the QoS.

In this paper, we present a dynamic and adaptive admission control strategy for providing a fair scheduling and better performance for video streaming. We will show that our algorithm provides a better admission ratio (that is, a lower input stream rejection ratio, which is computed as the ratio of the number of rejected streams to the number of total input streams) for multiple CM stream inputs. It still presents similar levels of stream qualities (how well the requested rate is attained in CM servers) compared to the basic greedy admission control algorithm.

The rest of this chapter is organized as follows. In Section 2, we briefly describe an assumed CM server architecture and the admission control constraints (I/O bandwidth requirement and buffer constraint). The detailed description of our admission control and scheduling algorithm is presented in Section 3. Section 4 gives the quantitative results of experimental evaluations, and describes experimental designs and parameters. Finally concluding remarks and on-going work appear in Section 6.

II. BACKGROUND AND OBJECTIVES

In this section, we outline the assumed CM server system and its architecture [5], and results of a human perception experiment to justify the degradability of CM (e.g., video streams) data. In order to achieve high performance in designing our CM server, we should consider the resource constraints as well as the properties of CM streams. These CM streams have their own features and special *QoS* metrics. Since we decided to design our CM server and clients based on the lossy UDP, we adopted the QoS metrics suitable for the lossy protocol. These QoS met-

This work was supported in part by DARPA through the SPAWARSYSCEN under Contract Number N66001-97-C-8525 and Air Force contract number F30602-96-C-0130.

0-7803-6494-5/00/$10.00 © 2000 IEEE

rics play an important role in our QoS-driven CM server (in particular, QoS Manager). With respect to QoS metrics, Steinmetz [7], [8] has surveyed and specified QoS factors which affect human perception on various aspects of multimedia. However, Steinmetz did not consider the lossiness of the streams, hence making the parameters unsuitable for evaluating lossy CM applications. Wijesekera [9] defined a set of metrics that are suitable for the lossy nature of UDP-based CM communication. Continuity of a CM stream is metricized by three components: *rate, drift* and *content*. For the purposes of describing these metrics, we envision a CM stream as a flow of data units (referred to as *logical data units - LDUs*. The ideal rate of a flow and the maximum permissible deviation from it constitute our *rate* parameters. Given the ideal rate and the beginning time of a CM stream, there is an ideal time for a given LDU to arrive or to be displayed. Given the envisioned fluid-like nature of CM streams, the appearance time of a given LDU may deviate from this ideal. In addition to the timing, the ideal contents of a CM stream are specified by the ideal contents of each LDU. Due to loss, delivery or resource overload problems, the appearance of LDUs may deviate from this ideal, and may consequently lead to discontinuity. The aggregate number of such unit losses is the *aggregate loss* of a CM stream, while the largest consecutive non-zero loss is its *consecutive loss*.

Human response to video and audio is quite interesting. According to [9], up to 23% of aggregate video loss and 21% of aggregate audio loss are tolerable. The acceptable values for consecutive loss of both video and audio are approximately 2 LDU. Up to about 20% of video and 7% of audio rate variations are tolerable. They give an upper bound for ADF and CDF (described in Table I).

	ALF	CLF	Rate Variation
Video	< 23%	< 3	< 20%
Audio	< 21%	< 3	< 7%

TABLE I
ACCEPTABLE (TOLERABLE) RANGE OF LOSS FACTORS & RATE VARIATION IN CM STREAMS

Given certain resources, we want to support as many CM streams whose QoS parameters are all acceptable as possible. When the load of the CM server is low, it is possible to meet this requirement. However, as more and more clients require CM streams, the QoS of CM server will degrade. It is a good choice to make it degrade gracefully. To guarantee that each client is served with some reasonable quality, admission control is also necessary. Thus, one of the bottom lines which we should consider with respect to the resource management among competitive applicaitons in CM servers, is to meet the above measured characteristics of CM as much as possible.

A. Overview of CM Server System

Our CM server system is based on a client-server model and is based on two file systems: i.e., *Presto File System (PFS)* and conventional *Unix file system (UFS)*. The core part of the *CM server* involves the *network manager*, the *QoS manager*, *I/O managers*, and *proxy servers*. These sub-modules in the CM server are threads running in a single process [5]. For brevity, we will not describe the details of other functionalities (but those of Admission Controller and QoS Manager) of our CM server system.

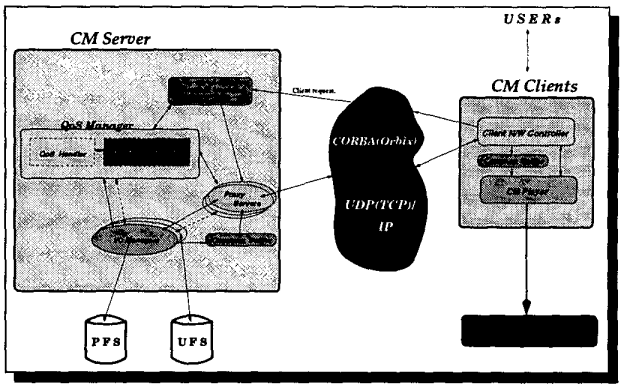


Fig. 1. CM Server-Clients Architecture

B. Admission Control

The admission control for input CM streams is managed by the *QoS Manager*, which is composed of the *Admission Controller* and the *QoS Handler*. The Admission Controller provides two constraint tests: (1) I/O bandwidth test; and (2) available buffer test. Each request (stream) arrives with some *rate* value (e.g., playing-back rate : i.e., frames per second) and the Admission Controller determines whether to admit it or not. The QoS Handler module takes care of data rate handling according to a rate input parameter provided by the client's request. In case the playing time gets delayed due to some kind of system overhead, this module will drop some frames properly so as to keep the data rate. This module also controls the dynamic QoS negotiation and management. To check the availability of disk I/O bandwidth in admission control, we use an I/O bandwidth test: that is, check if sum of latency overhead time and transfer time is less than a given cycle length. The available Buffer test is as follows: the total buffer requirement must be smaller than the available buffer size. Hence, the requirement of admission control is to satisfy both I/O bandwidth and avaialble buffer tests.

III. ADAPTIVE ADMISSION CONTROL AND RESOURCE SCHEDULING

In this section, we describe a widely used greedy admission control mechanism and its limitations. Next we explain a dynamic and adaptive admission control and scheduling algorithm which involves QoS degradation and adaptation to enhance the system resource utilization.

A. Background

In section II-B, we explained the two fundamental admission constraints in terms of I/O bandwidth and buffer size. Here we consider the only first constraint as the condition of admission decision in the commonly used greedy admission control and

schedule mechanisms. Most of the existing admission control approaches are purely greedy strategy in the sense that a new application (video stream) can be accepted only if the server could give the client all the requested resources. The major disadvantage of the greedy methods, which have been accepted as much of the work in soft-real time systems such as continuous media servers, is that they have focused on either conservative approaches based on bandwidth peaks, or statistical methods [10], [11] which model arrival rates and stream bandwidths with probability distributions and determine a satisfactory level of performance by the disk and network subsystems, either separately or as a system. These approaches are too conservative and admit too few streams, thereby under-utilizing the server resources. Although probabilistic methods exist to amortize the cost of this failure, this is undesirable in general [12].

We propose an enhanced admission control algorithm in that it is capable of re-adjusting resources according to the amount of remaining resources. The key idea here is to assign a portion of the resources as the reserves, and when the applications start to dip into the reserves, another strategy is invoked. In heavy loaded traffic (when a fairly large number of client requests want services; for example, the total disk bandwidth utilization gets over 70%), if the remaining available resources become smaller than some value (**threshold:** T_{res}), we assign only some portion of the requested resources to the new requests according to the **done_ratio** and *available resources*. For the degraded streams to adapt based on the availability of resource, **resource-negotiation** is required. We start with a brief description about the typical greedy approach and then discuss our new strategy, which is the focus of this paper. Table II describes the attributes used in the algorithms.

Attributes	Descriptions
s_i	stream i
q_i	initially requested resource of s_i
v_i	currently serviced resource of s_i : $= q_i * d_i$
$\mathcal{RS}$	set of all the running streams : $\bigcup_i s_i$, where $\{s_i \mid F_{min,i} < d_i \leq 1.0\}$
$\mathcal{DS}$	set of degraded streams : $\mathcal{RS} - \bigcup_i s_i$, where $\{s_i \mid d_i = 1.0\}$
s_{min}	stream j of which d_j is smallest in $\mathcal{DS}$: $s_{min} = \{s_j \mid \forall_{s_j, s_k \in \mathcal{DS}} d_j \leq d_k\}$
m	number of streams in $\mathcal{RS}$
done_ratio	ratio of serviced resource to requested resource for a stream
remaining_ratio	ratio of resource yet serviced to requested resource for a stream : $(= 1.0 - done\_ratio)$
$remaining\_rate_i$	amount of resource which is yet serviced in stream i : $(= q_i - v_i)$
sum_remaining_ratio	total sum of remaining_ratio of degraded streams : $\sum_k e_k$, where $s_k \in \mathcal{DS}$
MIN_FRACT	minimum amount of resource to be assigned for a stream
MIN_RESERVE	minimum amount of reserve which must be at least maintained in $\mathcal{DR}$ Amode
d_i	done_ratio of s_i
e_i	remaining_ratio of s_i : $(= 1.0 - d_i)$
$F_{min,i}$	MIN_FRACT of s_i
T_{total}	total available resources initially given
T_{res}	amount of resources assigned to the reserve
T_{free_res}	available resources in Reserve : $\begin{cases} T_{total} - T_{used} & \text{if } T_{used} > T_{total} - T_{res} \\ 0 & \text{otherwise} \end{cases}$
$T_{free\_non\_res}$	available resources outside Reserve : $\begin{cases} T_{total} - T_{res} - T_{used} & \text{if } T_{used} \leq T_{total} - T_{res} \\ 0 & \text{otherwise} \end{cases}$
T_{alloc}	allocated resources to the requesting stream
T_{avail}	available resources to assign : $(= T_{total} - T_{used})$
T_{used}	total resources currently used : $\sum_i v_i$, where $s_i \in \mathcal{RS}$
$\Psi(T_{free\_res})$	heuristic function for reserve assignment in *Admission Test* : e.g. $\frac{T_{free\_res}}{k}$, where $k = k_1^{(T_{used} - T_{res}) \cdot k_2}$
$\Phi(T_{free\_res})$	heuristic function for reserve assignment on *Close of streams* : e.g. $\frac{T_{free\_res}}{k}$, where $k = k_3 \cdot (T_{used} - T_{res}) + 1$
congestBit	bit flag to indicate congested/uncongested state : $\begin{cases} 1 & \text{if } T_{used} > T_{total} - T_{res} \\ 0 & \text{otherwise} \end{cases}$
$\mathcal{SU}(t)$	total system utilization defined by $\sum_k d_k$, where $\forall_k s_k \in \mathcal{RS}$ at time t
$\mathcal{QT}(t)$	total video quality defined by $\mathcal{SU}(t)/m$ at time t

TABLE II
ATTRIBUTES USED IN ALGORITHM

B. Basic Greedy Algorithm

The naive approach to admission control is to use a **greedy strategy**, where applications (streams) are admitted as long as there are resources [13], [14]. On application arrival, the policy admits the application if the resources available are greater than the resources requested. On application departure, the policy just adds the released resources from the application to the resources available. Table III (Algorithm 1) presents the details of the greedy strategy.

C. Reserve-based Admission Control and Scheduling Algorithm

Due to the shortcomings of basic greedy admission control algorithms, we should be able to think of another strategy to allow more streams to run concurrently in the continuous media servers by degrading the requested quality of newly arriving streams and by adapting the returned resources to these streams. Since we cannot make a good estimate of the future traffic, it is very difficult to implement a perfect algorithm to determine the amount of degraded resources. Yet, we believe that our new *soft-QoS* framework noticeably reduces the rejection ratio and lets more input streams run in the system.

C.1 Reserve-based Admission Control

Given the average seek time (t_s), cycle length (T_{svc}), disk block size (b), rotation time (t_r), and disk transfer rate (R), the consumption rate of each client request (q_i) is a variable in the I/O bandwidth constraint. We initialize the total available rate (T_{avail}) with $T_{svc}(= T_{total})$, and set T_{res} (the **threshold** value for criteria to check *congested_bit*). Initially the *congested_bit* is 0 (not congested). The resource (here I/O bandwidth) is **congested** if the resource usage (T_{used}) is more than (T_{svc} - T_{res}). We can set the T_{res} amount of resources for admitting more requests at a reduced quality. Here, T_{usage} + T_{avail} = T_{svc}. That is, the *congested_bit* is set (= 1) only when T_{avail} becomes smaller than T_{res}; otherwise, the basic I/O bandwidth constraint is just applied (i.e., the new request is admitted without being degraded). Under the *congested_bit* being set, we restrict the amount of assignment of resources to the new request because there is no sufficient resource remaining any longer). A brief diagram for explaining the strategy of the adaptive admission control and resource scheduling is shown in Figure 2(a) and 2(b) [15]. We explain the details immediately below.

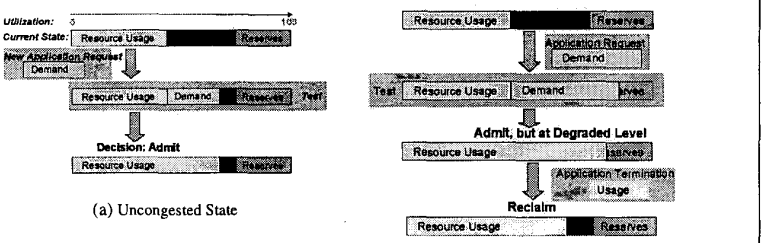
On adding the request q_i, if the resource remains uncongested, then admit it with no degradation.

$$T_{alloc} = q_i.$$

That is, under the situation of the *congested_bit* being reset (= 0), the new requests are simply admitted. Let the current usage be T_{used}; then, adding the application q_i will increase the usage to ($T_{new} = T_{used} + q_i$). If ($T_{new} \geq T_{svc} - T_{res}$); then, adding q_i will make the resource congested. When the resource gets congested, we have to dip into the resources.

$$T_{free\_non\_res} = \max(0, T_{svc} - T_{res} - T_{used}).$$

This is the unused resource outside the reserves.
Let T_{avail} be the amount of resources in the reserves. Then,



(a) Uncongested State

(b) Congested State

Fig. 2. Negotiated Admission Control

Algorithm 1 *Greedy Admission Control (event E_i, q_i)*

```
▷  E_i = {START, CLOSE}
1  switch EVENT of
2    case "START" :
▷      with parameters of q_i /* get requested resource */
3      if ( T_avail ≥ q_i ) then
▷        Admit the application
4          T_alloc ← q_i;
5          T_used ← T_used + q_i;
6          T_avail = T_total − T_used;
7      else /* not enough resources*/
8        Reject this application instance;
9    case "CLOSE" :
▷      with parameters of appl_inst_ID
10     T_used ← T_used − T_alloc;
11     T_free_res = T_total − T_used;
12     Delete the application instance;
13   end switch
```

TABLE III

ALGORITHM FOR *Greedy Admission Control*

T_{free_res} = min(T_{res},($T_{svc} - T_{used}$)).

The new application is allocated, and then

$T_{used} = T_{free\_res} + \min((q_i - T_{free\_res}), \Psi(T_{free\_res}))$.

That is, we allocate the resources that are outside the reserves and, in addition, we give a maximum $\Psi(T_{free\_res})$ of the resources in the reserves. $\Psi(.)$ is a function of T_{free_res} and returns an appropriate amount of resources according to T_{free_res}. Several different policies such as a fixed, variable, and exponential policy could be applied for this function. If $\Psi(T_{free\_res})$ is too small (in this case, we had better not support the new request because display quality may be too poor), we simply reject the request. If the running stream (s_k) is over, then the resource v_i is reclaimed. We return the resource occupied by the stream and adjust (if required) the resource allocation of streams not being fully serviced. Here the unused resource is allocated to applications such that the least serviced streams could get the returned resource first.

C.2 Negotiation Algorithm

we have implemented two heuristic-based methods and tested their performance: (1) *Reclamation within Returned Reserve ($\mathcal{RWR}$)* : This algorithm redistributes the only resources returned from the leaving streams (v_i). In $\mathcal{DRA}$, we used T_{avail} (instead of v_i) for reclamation. The T_{alloc} per stream is calculated according to the ratio of *remaining_ratio* to *sum_remaining_ratio*.

The key idea here is not to touch the reserves, but to utilize the only returned resource due to the departure of streams. We will validate the better performance of this policy compared to that of $\mathcal{DRA}$. (2) *Dynamic Reserve Adaptation ($\mathcal{DRA}$)* : On departure of streams, $\mathcal{DRA}$ returns the assigned resources to the leaving streams (q_i) back to the available resource pool and re-calculates the new T_{avail} using the total available resource (i.e. old $T_{avail} + q_i$). Among the requests that are not fully serviced yet, we select a request (s_{min}) from the queue ($\mathcal{DS}$), whose *done_ratio* is the smallest first, and assign the proper resource to the request. The **maximum $\Phi(T_{free\_res})$-rule** applies here. Assign T_{alloc} = min ($\Phi(T_{free\_res})$, remaining_rate$_{min}$)to the selected request (s_{min}). The loop continues until there is no more available resource to assign or T_{alloc} is too small to assign. When the resources in the reserves are not used for a long time, some of it is reclaimed. For every k period, if there is no request to the reserve, then we release some amount of the reserve to be reclaimed.

IV. SIMULATION RESULTS

For results presented in this section, we simulated an environment with a configuration which corresponds to a Sun Ultra Sparcstation with a Seagate Barracuda 4GB disk. We simulate the situation where the applications with soft-QoS requirements arrive at the system in an arbitrary order. The applications also come with a play-out rate (which maps into an amount of required resource), depending on the user profile and the data content, which is a value of the data rate parameters. For example, we may have a set of video distribution applications that provide streaming video with different types of content to the remote users. Ideally, the application load should be characterized using experimental data. However, for the simulations, we assume the following conditions. The applications arrive based on a Poisson process with an average inter-arrival rate (λ). The duration of the applications is described using a Gaussian process with the mean (μ_d) and the standard deviation (σ_d) of the distribution. The experiments are run by fixing the generating process parameters and then generating many application traffic traces. The admission control and scheduling algorithm is tried on each one of the application traffic traces. Table IV shows the

actual attributes used in our experiment.

Parameter	*Description*
λ	average inter-arrival rate of a Poisson process with which applications arrive are based on
μ_d	mean of a Gaussian process to describe the duration behavior of applications
σ_d	standard deviation factor of a Gaussian distributed scale process for the duration of applications

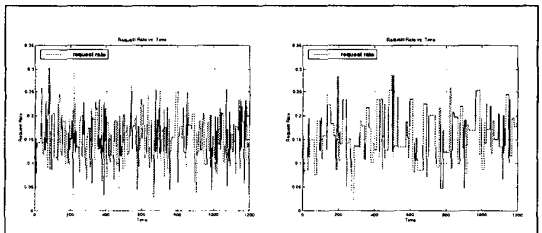
TABLE IV
SIMULATION PARAMETERS

A. Measured Metrics

We measure the following metrics:

1. *Number of running streams over time.*
2. *Disk I/O bandwidth utilization*: The disk I/O bandwidth utilization is described as T_{used}/T_{total}.
3. *Total quality*: The total quality is computed as *Total system utilization* divided by the number of current running streams, where total system utilization is computed as the sum of all the *done_ratio*s of the streams supported by the resource.
4. *Effect of resource reserve size*: We study the effect of the amount of resources kept aside for the reserves ($T_{reserve}$) on the number of admitted streams in time.

Figures 3(a) and 3(b) show the data rates of input streams which change in time. *Heavy traffic* is generated by setting λ = 0.333, μ_d = 90, and σ_d = 3.5. *Medium traffic* is generated by having λ = 0.125, μ_d = 60, and σ_d = 3.5. The values on the x-axis are normalized. Under the heavy traffic, the resource demands arrive more frequently than under the medium traffic.



(a) Heavy Traffic (b) Medium Traffic

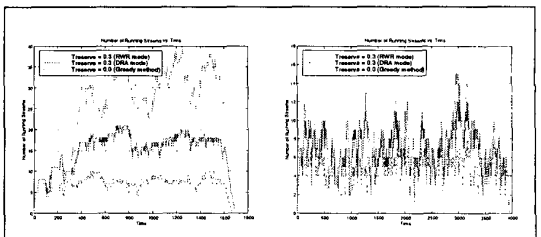
Fig. 3. Request Data Rate over Time

B. Simulation Results

We will present the detailed experimental result of each experiment in the following subsections.

B.1 Experiment 1: Number of Running Streams

As depicted in Figure 4, the $\mathcal{RWR}$ algorithm achieves a 200 $\sim$ 300% increase in the number of streams that can be serviced simultaneously by the server. In case of the $\mathcal{DRA}$ algorithm, it also achieves about a 100% increase. In other words, our $\mathcal{RAC}$ algorithm (both $\mathcal{RWR}$ and $\mathcal{DRA}$) noticeably reduces the rejection ratio compared to the basic greedy method.

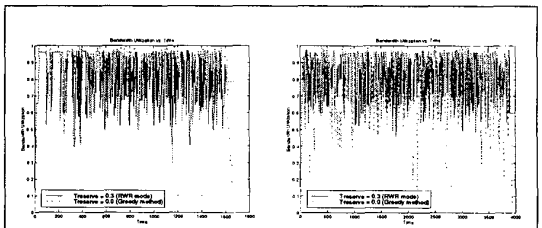


(a) Heavy Traffic (b) Medium Traffic

Fig. 4. Number of Current Running Streams over Time

B.2 Experiment 2: Disk Bandwidth Utilization

Figure 5 illustrates the disk bandwidth utilization over time for $\mathcal{RAC}$ and greedy strategies. We can observe that the bandwidth utilization of $\mathcal{RAC}$ looks slightly better than that of the basic greedy algorithm most of the time. The hevier the input traffic is, the more the difference between the two strategies increases.

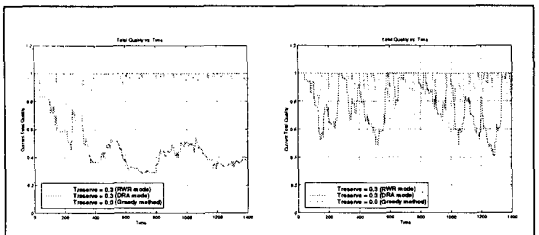


(a) Heavy Traffic (b) Medium Traffic

Fig. 5. Bandwidth Utilization over Time

B.3 Experiment 3: Total Quality

To investigate the display quality of the $\mathcal{RAC}$ algorithm, we plot another curve. Figure 6 illustrates the total quality which is quantified by dividing the sum of each application's done_ratio by the number of current running streams. It is observed that the total quality achieved by the $\mathcal{RWR}$ algorithm is almost the same as that achieved by the greedy algorithm. Only in the initial part of simulation, is it occasionally below 1.0, but still over 0.85, which is fairly good. However, the quality gets to be almost 1.0 after that. In contrast, the performance of $\mathcal{DRA}$ algorithm in total quality is quite low most of the time. This arises due to the fact that the $\mathcal{DRA}$ algorithm holds off the reserves for future streams, which arrive soon before some of the current running streams depart. Under medium traffic, though the performance of $\mathcal{DRA}$ gets better, it is still too low. Here we could conclude that we should be more careful to dip into the reserves for resource reclamation. In the $\mathcal{RWR}$ mode, we hold off the reserves by reclaiming the only returned resources.

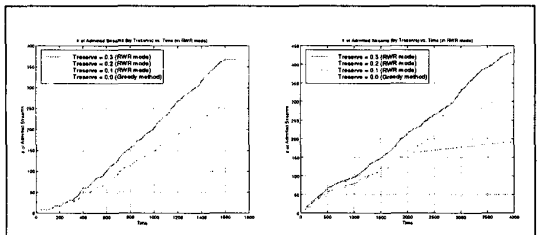


(a) Heavy Traffic (b) Medium Traffic

Fig. 6. Total Quality over Time

B.4 Experiment 4: Number of Streams vs. Threshold

We were interested in studying the effect of the amount of resources kept aside for the reserves ($T_{reserve}$) on the performance of the system. Using simulation experiments, we were able to verify that for the fixed policy for resource assignment (where we assigned $\frac{1}{K}$ of the resources to the applications after the resource utilization entered the reserves), the performance depended on the $T_{reserve}$. This observation is explained by understanding that, from a system perspective, keeping more resources in the reserves implies that the system believes probability of more applications can be supported without a loss of quality. If the statistics of the application traffic is kept constant, there is an optimal value of $T_{reserve}$ that maximizes the total quality. We ran simulations with the fixed policy and varied the amount of resources assigned to the reserves. As we had expected, we observed that for fixed application traffic characteristics the total quality varied with the value of $T_{reserve}$. Figure 7 shows the relationship between the number of admitted streams and time in four different $T_{reserve}$ values. From the experimental observations, we see that the system benefit gets maximized for a reserve value between 0.2 and 0.3.



(a) Heavy Traffic (b) Medium Traffic

Fig. 7. Number of Admitted Streams vs. Threshold

V. Conclusions and Future Research Directions

We introduced an integrated adaptive admission control, and resource scheduling algorithm for CM server systems, and simulation evaluation results of our algorithm and a greedy algorithm with respect to several metrics designed to measure the admission ratio, total quality, and bandwidth utilization. It was observed that under heavy traffic, our algorithm achieves much better performance than the greedy algorithm. Using our scheme, we observed that our admission control and scheduling strategy provides three principle advantages over conventional mechanisms. First, it guarantees better total system utilization. Second, it provides better disk utilization and larger admission ratio for input continuous media streams, which is a major advantage of our scheme. Third, it still presents acceptable play-out qualities compared to the conventional greedy admission control algorithm. Conclusionally, using our scheme, more streams can be running with an acceptable range of data quality in a given system resource. Our ongoing and future work include extending the QoS metrics considered in our algorithm to multi-dimensional ones (e.g., frame size, buffer requirement, compression quality factor, scalability, network bandwidth, etc.). Toward this end, we are studying the constructs of user utility functions and resource request functions via which we could formulate the system and user demands more specifically.

References

[1] D. Kenchammana-hosekote and J. Srivastava, "Scheduling Continuous media in a Video-On-Demand Server," in *Proceedings IEEE Conference on Multimedia Computing and Systems*, May 1994.

[2] C. Martin, P. S. Narayanan, B. Ozden, R. Rastogi, and A. Silberschatz, "The Fellini Multimedia Storage Server," in *Multimedia Information Sotrage and Management* (S. M. Chung, ed.), Kluwer Academic Publishers, 1996.

[3] W. Lee and B. Sabata, "Admission Control and QoS Negotiations for Soft-Real Time Applications," in *Proceedings of the 6th IEEE International Conference on Multimedia Computing and Systems (ICMCS'99)*, (Florence, Italy), June 1999.

[4] W. Lee and J. Srivastava, "CORBA Evaluation of Video Streaming wrt QoS Provisioning," in *Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems (SRDS '98)*, (West Lafayette, Indiana), October 1998.

[5] W. Lee, D. Su, and J. Srivastava, "QoS-based Evaluation of File Systems and Distributed System Services for Continuous Media Provisioning," *To appear in Information and Software Technology, Elsevier Science*, 2000.

[6] W. Lee and J. Srivastava, "An Algebraic QoS-based Resource Management Model for Competitive Multimedia Applications," *To appear in Journal of Multimedia Tools and Applications, Kluwer Academic Publishers*, 2000.

[7] R. Steinmetz and G. Blakowski, "A Media Synchronization Survey: Reference Model, Specification and Case Studies," *IEEE Journal on Selected Areas in Communication*, vol. 14, no. 1, pp. 5–35, 1996.

[8] R. Steinmetz, "Human perception of jitter and media synchronization," *IEEE Journal on Selected Areas in Communication*, vol. 14, no. 1, pp. 61–72, 1996.

[9] D. Wijesekera and J. Srivastava, "Experimental Evaluation of Loss Perception in Continuous Media," *to appear in ACM Springer Multimedia Systems Journal*, 2000.

[10] F. T. Ernst W. Biersack, "Statistical Admission Control in Video Servers with Constant Data Length Retrieval of VBR Streams," in *Third International Conference on Multimedia Modeling*, (Toulouse, France), Nov 1996.

[11] H. M. Vin, P. Goyal, A. Goyal, and A. Goyal, "A Statistical Admission Control Algorithm for Multimedia Servers," in *Proceedings of ACM Multimedia '94)*, (San Francisco), October 1994.

[12] D. Makaroff, G. Neufeld, and N. Hutchinson, "An Evaluation of VBR Disk Admission Algorithms for Continuous media File Servers," in *Proceedings of the ACM Multimedia Conference*, (Seattle, Wa), Dec 1997.

[13] S. W. Lau and J. C. S. Lui, "A Novel Video-On-Demand Storage Architecture for Supporting Constant Frame Rate with Variable Bit Rate Retrieval," in *5th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, (Durham, N.H.), April 1995.

[14] H. M. Vin and P. V. Rangan, "Admission Control Algorithms for Multimedia On-Demand Servers," in *3rd International Workshop on Network and Operating Systems Support for Digital Audio and Video*, 1992.

[15] SRI International, "ErDos Middleware for End-to-End Resource Management," in *DARPA Quorum/HCC PI Meeting Presentation*, (San Diego, CA), July 1998.