The Computer Journal, 46(6), © British Computer Society 2003; all rights reserved

A Cost-Based Admission Control Algorithm for Digital Library Multimedia Systems Storing Heterogeneous Objects

ING-RAY CHEN AND NARESH VERMA

Department of Computer Science, Virginia Polytechnic Institute and State University, Northern Virginia Center, 7054 Haycock Road, Falls Church, VA 22043, USA
Email: {irchen, naverma}@vt.edu

We present and analyze a cost-based admission control algorithm for handling mixed workloads in modern multimedia systems such as a digital library multimedia system that must provide access services to heterogeneous objects stored in the library. The cost-based scheme considered in the paper is based on the concept of 'rewards' and 'penalties' associated with requests of various media object types. Instead of admitting object requests until resources are exhausted as a condition for admission control, resources are reserved to requests of different media types dynamically based on the cost-based scheme so that the system is capable of maximizing the total reward received by the system in response to workload changes in the environment. We analyze the maximum queue sizes for admitting discrete media requests to meet the imposed response-time constraints and for improving the total reward received by the system by exploiting leftover resources from servicing continuous media requests. A solution for the total reward obtainable is derived and validated via simulation.

Received 21 May 2002; revised 11 January 2003

1. INTRODUCTION

With the proliferation of Web technologies, digital library multimedia objects are constantly being accessed by a huge population online everyday. A digital library typically consists of mixed-workload multimedia object types, including video, audio, images and text (html files). It is desirable to allow accesses to all these mixed object types with certain performance guarantees. Video and audio objects are different from image and text in that they are continuous media, rather than discrete media. Technologies for handling continuous media services have been studied quite intensively in the past 10 years and streaming video/audio broadband services reportedly will become the next major frontier for online multimedia [1].

Less attention has been paid to how to service mixed workload objects effectively for multimedia servers designed to provide online digital library multimedia services [2, 3, 4, 5]. The basic technical challenge is how to provide performance guarantees to image/text types of requests while at the same time satisfying the real-time requirement of video/audio types of requests. Since continuous media are resource demanding with different data rates at different times due to compression, techniques have been proposed to take advantage of the 'leftover' time after servicing video/audio requests to service image/text requests. This effort includes the design and evaluation of several disk scheduling algorithms to *squeeze in* discrete data requests amid servicing continuous data requests such that the response time for discrete data requests is minimized without adversely affecting the quality of service (QoS) requirement of video/audio requests [2, 3]. These algorithms are designed based on the assumption that video/audio and image/text requests share the same set of system resources with video/audio streaming services always taking a higher priority over image/text data. However, in Internet Web applications while continuous video/audio streaming services are appealing, image/text services account for more than 70% of the data bytes accessed on the Web [6]. It is thus not justified to always give resource-demanding video/audio object requests a higher priority over image/text requests.

To address this issue, Shenoy and Vin [4] recently proposed a two-level disk scheduling framework. At the high level (level 1), a *class-independent* scheduler is used to govern the allocation of disk bandwidth to various application classes. At the low level (level 2), class-specific schedulers are used to order the requests into a common schedule queue for disk access. Among other advantages, this approach can adapt to workload changes by performing re-allocations of disk bandwidth dynamically by using the class-independent scheduler at level 1, and can minimize the seek time and rotational latency overhead during disk access and satisfy the QoS requirements of various classes by using class-specific schedulers at level 2. While the design of level 2 class-specific schedulers was discussed in detail, no discussion was given on how the class-independent scheduler at level 1 can adapt to workload changes to reallocated disk bandwidth at run time. In another work, To and Hamidzadeh [5] suggested trading off continuous media (CM) throughput for discrete media (DM) throughput by using the CM-to-DM throughput redirection ratio as a performance metric. They suggested ways of trading CM throughput with DM throughput. One way is to use a buffer discounting technique (i.e. by allocating more buffer space to each CM request) to serve CM requests. This approach reduces the number of CM requests admissible. More importantly, since each CM request is allocated with more buffer space, optimal disk reads (by reading an optimal amount of data per access) can be performed to serve CM streams to minimize or eliminate the rotational latency in a cycle. As a result, the disk server can effectively exploit the bandwidth leftover in a cycle to serve more DM requests, thus achieving a high CM-to-DM throughput redirection ratio. A question that remains to be answered, however, is how much bandwidth should be redirected from CM to serve DM requests.

This paper proposes and analyzes a cost-based admission control algorithm to address the issue of how many resources should be allocated to service video/audio (CM) and image/text (DM) requests. By means of reserving separate resources to service CM and DM requests, the algorithm explicitly determines the numbers of CM and DM requests that can be admitted into the system without causing resource overload. Our approach is to dynamically partition system resources based on workload changes at run time with the goal to maximize a ‘value’ metric (or to minimize a cost metric) while at the same time ensuring that the response time requirements of both types of object requests are satisfied (at the expense of rejecting requests when the system is overloaded). Image/text data requests can have their own resources, without having to use the ‘leftover’ bandwidth after serving video/audio requests.

The basic idea is to assign a value/penalty pair to each request type, indicating the reward that such a request will bring to the system if it is serviced successfully and the loss to the system if the request cannot be served (rejected) due to lack of resources. Thus, resource partitioning is not only based on the workload characteristics of various types of mixed workloads but also based on the value/penalty characteristics of requests. In the extreme case that the value assigned to each video/audio request is very high compared with that assigned to each image/text request, our algorithm degenerates to the one with video/audio always having a higher priority than that of image/text requests. Moreover, the penalty parameter is optional. If it is set to zero, our algorithm degenerates to the case where priority assignments are derived from the ‘value’ parameter alone. The concept of reward/penalty for multimedia servers was first discussed by Chen *et al.* [7] in the context of admission control. Lee and Sabata in their work [8] generalized the concept of rewards to application-specific benefit functions and applied it to admission control and QoS negotiation in multimedia servers. Our work is the first to apply it to mixed-workload multimedia servers.

The rest of the paper is organized as follows. Section 2 gives a background of round-based disk scheduling and admission control algorithms for typical video servers. Then the system model for a mixed-workload digital library multimedia server is described. Section 3 describes our cost-based admission control algorithm for servicing mixed-workload digital library multimedia servers with the objective of maximizing the total reward obtainable by the system without violating the disk bandwidth and response time requirements of requests. Section 4 presents data analysis and simulation results for evaluating the proposed admission control algorithm along with result interpretations. Finally, Section 5 concludes the paper and outlines some future research areas.

2. SYSTEM MODEL

2.1. Mixed workload types

We assume that a typical digital library multimedia system will service three mixed workload types: video, image and text. Video objects are of VBR[1] types with audio information incorporated as in MPEG. Images and texts are separate into two workload types since bandwidth and size requirements are vastly different for these two discrete workload types. Integrated requests that access several object types concurrently are not considered.

2.2. Multimedia server

We assume a multimedia server with a disk array. Video, image and text objects are stored as files on disk. The basic transfer unit between the server and disk is one data block stripped evenly over all disks in the disk array. An object can span several data blocks, especially for video objects. When servicing an image or a text object request, as many disk blocks required to cover the requested object are retrieved by the server. However, when servicing a video object request via streaming, only as many data blocks covering the playback time of a service round duration are retrieved in each service round due to the VBR property associated with video data. For video requests, a dual buffering scheme is used. A disk buffer is used to hold the data retrieved from the disk array, while a network buffer is used to hold the data retrieved from the previous round and pushes the data to the display device locally or over the network to the remote client site. For each admitted video streaming service, the server keeps track of which part of the requested video object the user is currently requesting and always retrieves as many data blocks as necessary in a service round so as not to miss the

[1]Video streams normally exhibit variable bit rate (VBR) properties since video data are normally compressed and stored on the server before delivery.

real-time continuity requirement. Since video objects are VBR in nature, it is possible that a fraction of a data block retrieved in the previous round may remain unconsumed in the network buffer in the current round. We assume the system has enough buffer space to handle video, image and text requests. Thus, the limiting factor is the disk overload in servicing requests of mixed types.

2.3. Disk scheduling

The system services requests in rounds with the service round duration being T_{SR}. It is based on the notion of *cycle-based* disk scheduling wherein all requests are serviced on a cycle by cycle basis. In general, image/text requests can be serviced in one of two ways: they can be serviced *after* video/audio requests are serviced in a service round, or *interleavingly* with video requests so as to minimize the disk seek time and/or rotational latency. Our approach falls under the latter category. That is, in any service round, we use the classic SCAN algorithm to order all video/text/image requests to be processed in that round such that the disk read/write heads only traverse the disk array in one direction to retrieve all needed data for all requests to minimize the seek time. The numbers of video, image, and text requests to be processed in a single round, denoted by n_{V}, n_{I} and n_{T} respectively are to be controlled by our algorithm via resource allocation. Thus, text and image requests under this disk scheduling algorithm are considered being serviced *in batch*, that is, n_{I} image requests and n_{T} text requests (if exist) would be scheduled for service at the beginning of each service round in a batch manner and would depart the system at the end of the service round. The batch sizes for image and text requests (i.e. n_{I} and n_{T}) depend on the amount of disk bandwidth allocated to them by our algorithm.

We assume that the system maintains two FIFO queues, one for image and one for text requests. The system can *reject* requests for performance considerations or resource reasons. The first n_{I} requests at the front of the image queues and the first n_{T} requests at the front of the text queue (if exist) are served in the current round *in batch* (using resources allocated to them in the current round) while the remaining ones plus any new requests arriving at the server can be serviced in the next round and so on.

The arrival rates for video, image and text objects are λ_{V}, λ_{I} and λ_{T} (requests/s) respectively. For the departure rate, we assume that the average departure rate per video request (after it is admitted) is μ_{V} (requests/s). Since image requests are serviced in batch per round, the average 'batch' departure rate for image/text is $1/T_{\mathrm{SR}}$ (batch/sec). Let μ_{I} represent this parameter. It can be translated into a 'request' departure rate of $n_{\mathrm{I}}/T_{\mathrm{SR}}$ requests/s when the image queue contains at least n_{I} image requests during a service round. Similarly, let μ_{T} be the batch departure rate for servicing text requests. By the same token, $\mu_{\mathrm{T}} = 1/T_{\mathrm{SR}}$ (batch/s).

2.4. Resource partitioning

Once objects are stored onto the disk array, the system has knowledge about the size requirements and disk locations of each object. Further, it has some knowledge about the size distribution of all images and text objects for it to determine statistically how many text/image requests the system is able to service in one service round based on disk bandwidth allocated without causing the disk to be overloaded probabilistically. For video objects, we assume that the system has a histogram of the distribution of the size needed to satisfy the playback requirement. This information is obtained via the bit trace of the video object, e.g. *Star Wars*.

Our algorithm partitions the service round duration T_{SR} into three parts, i.e. video, image and text partitions, based on the cost and workload characteristics associated with video, image and text objects dynamically. Thus, with the knowledge of the amount of resources allocated to each object type, we can theoretically estimate the maximum number of requests of a particular object type the server can admit in a service round based on statistical admission control. We use a typical seek time model [9] in which the seek time plus the rotational latency are constant for accessing a data block as the basis for theoretically estimating the number of requests of each object type the disk can serve simultaneously so that the disk overload probability is less than 10^{-4}. We also assume that the disk transfer rate is a constant.

Finally, our algorithm applies a policy that if during a service round, there exists some leftover residual time in the video partition after video objects have been serviced, then some image/text requests waiting in their queues not having been serviced in that round (if any exists) can be scheduled for service using the residual time. Note that this policy will only increase the reward obtainable by the system since it allows more image/text requests to be serviced.

2.5. Performance metric

The goal of our cost-based admission control algorithm is to maximize the reward obtainable by the system without compromising the quality of service requirements, viz. bandwidth and response time requirements, of all requests. Suppose that in per unit time, the system completes the services of $\mathcal{N}_{\mathrm{V}}$ video requests, $\mathcal{N}_{\mathrm{I}}$ image requests and $\mathcal{N}_{\mathrm{T}}$ text requests, while rejecting $\mathcal{M}_{\mathrm{V}}$ video requests, $\mathcal{M}_{\mathrm{I}}$ image requests and $\mathcal{M}_{\mathrm{T}}$ text requests due to admission control. Then, the 'reward rate' at which rewards are obtained by the system is:

$$v_{\mathrm{V}}\mathcal{N}_{\mathrm{V}} + v_{\mathrm{I}}\mathcal{N}_{\mathrm{I}} + v_{\mathrm{T}}\mathcal{N}_{\mathrm{T}} - q_{\mathrm{V}}\mathcal{M}_{\mathrm{V}} - q_{\mathrm{I}}\mathcal{M}_{\mathrm{I}} - q_{\mathrm{T}}\mathcal{M}_{\mathrm{T}}$$

where v_{V}, v_{I} and v_{T} are the reward values earned by the system after servicing a video, an image, and a text request respectively and q_{V}, q_{I} and q_{T} are the corresponding penalty values taken away from the system after rejecting a request. We consider the reward/penalty values being the 'average' values applying to a given request type (video, image or text). In general, it is fair to say that video has the highest values, followed by image and text objects. Although there can be overlapping values among different types of objects, these three different types of objects will likely be centered

TABLE 1. Notation.

Symbol	Meaning
T_{SR}	service round duration
λ_{V}	collective video arrival rate
λ_{I}	collective image arrival rate
λ_{T}	collective text arrival rate
μ_{V}	departure rate of a video request
μ_{I}	batch departure rate of image requests
μ_{T}	batch departure rate of text requests
v_{V}	reward associated with a video request
v_{I}	reward associated with an image request
v_{T}	reward associated with a text request
q_{V}	penalty associated with a video request
q_{I}	penalty associated with an image request
q_{T}	penalty associated with a text request
$\mathcal{R}_{\mathrm{V}}$	reward rate obtainable from servicing video requests
$\mathcal{R}_{\mathrm{I}}$	reward rate obtainable from servicing image requests
$\mathcal{R}_{\mathrm{T}}$	reward rate obtainable from servicing text requests
$\mathcal{R}$	total reward rate obtainable from servicing video/image/text requests
f_{V}	disk bandwidth allocated to video requests in a service round
f_{I}	disk bandwidth allocated to image requests in a service round
f_{T}	disk bandwidth allocated to text requests in a service round
$(f_{\mathrm{V}}^*, f_{\mathrm{I}}^*, f_{\mathrm{T}}^*)$	optimal $(f_{\mathrm{V}}, f_{\mathrm{I}}, f_{\mathrm{T}})$ for maximizing $\mathcal{R}$
n_{V}	number of video requests to be serviced based on f_{V} in a service round
n_{I}	number of image requests to be serviced based on f_{I} in a service round
n_{T}	number of text requests to be serviced based on f_{T} in a service round
$(n_{\mathrm{V}}^*, n_{\mathrm{I}}^*, n_{\mathrm{T}}^*)$	optimal $(n_{\mathrm{V}}, n_{\mathrm{I}}, n_{\mathrm{T}})$ for maximizing $\mathcal{R}$
N_{V}	number of video requests that the system is able to service in a service round if all disk bandwidth is allocated to service video requests only
N_{I}	number of image requests that the system is able to service in a service round if all disk bandwidth is allocated to service image requests only
N_{T}	number of text requests that the system is able to service in a service round if all disk bandwidth is allocated to service text requests only
$K_{\mathrm{I}} \times n_{\mathrm{I}}$	maximum queue size for admitting image requests
$K_{\mathrm{T}} \times n_{\mathrm{T}}$	maximum queue size for admitting text requests

in separate zones with distinct 'average' values. Table 1 lists the notation used in the paper for easy reference.

3. ALGORITHM

We design our cost-based admission control algorithm with the following steps:

(i) First, we derive a solution for the reward rate obtainable by our algorithm as a function of model parameters based on simple queueing arguments. This reward rate derived represents a lower bound as it does not consider utilizing the leftover bandwidth in the video partition to serve additional image/text requests (i.e. over n_{I} image and n_{T} text requests) in any cycle.

(ii) Second, we apply the reward rate function derived to build a lookup table so as to perform bandwidth allocation dynamically at run time in response to workload changes to maximize the reward rate obtainable by the system. The lookup table contains an estimation of the reward rate value the system will obtain under a workload condition, as well as the best bandwidth allocation under which the reward rate value will be maximized.

(iii) Finally, while the bandwidth allocated to service image requests would only allow n_{I} image requests to be serviced in a service round without disk overload, we admit up to $K_{\mathrm{I}} \times n_{\mathrm{I}}$ image requests, $K_{\mathrm{I}} \geq 1$, as long as $K_{\mathrm{I}} \times T_{\mathrm{SR}}$ does not exceed the maximum allowable response time per image request. For example, if $T_{\mathrm{SR}} = 1$ s and the maximum allowable response time per image request is 2 s, then K_{I} would be 2. We admit text requests in a similar way. This admission control policy ensures that the response time requirement of image/text requests is met. Further, by admitting more image/text requests into the system, we can utilize left-over bandwidth from the video partition in a service duration to serve additional image/text requests to further improve the total reward rate obtainable by the system.

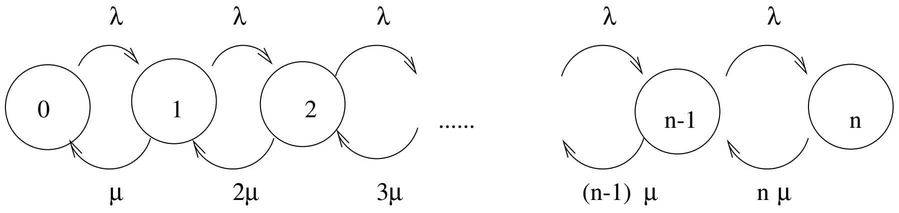

FIGURE 1. $M/M/n_{\mathrm{V}}/n_{\mathrm{V}}$ model for the partition that serves video requests.

3.1. Formulation

Consider that the disk bandwidth is allocated to video, image, text requests by the ratios of f_{V}, f_{I} and f_{T} respectively such that $f_{\mathrm{V}} + f_{\mathrm{I}} + f_{\mathrm{T}} = 1$ (normalized with respect to the total disk bandwidth). That is, in a service round of length T_{SR}, the amounts of time the disk used to service video, image, and text requests are $f_{\mathrm{V}}T_{\mathrm{SR}}$, $f_{\mathrm{I}}T_{\mathrm{SR}}$ and $f_{\mathrm{T}}T_{\mathrm{SR}}$ respectively. We first apply statistical admission control to compute how many requests of class type i the system is able to handle concurrently so that the probability of disk overload is below a threshold probability (say 10^{-4}), given that f_iT_{SR} is allocated to execute class type i in each service cycle [9]. Thus, the allocation of $f_{\mathrm{V}}T_{\mathrm{SR}}$, $f_{\mathrm{I}}T_{\mathrm{SR}}$ and $f_{\mathrm{T}}T_{\mathrm{SR}}$, to service video, image, and text requests respectively translates into the maximum number of requests that the system can handle concurrently, i.e. n_{V}, n_{I} and n_{T} for video, image and text requests respectively. Let $(n_{\mathrm{V}}, n_{\mathrm{I}}, n_{\mathrm{T}})$ denote the set that corresponds to a disk bandwidth allocation set $(f_{\mathrm{V}}, f_{\mathrm{I}}, f_{\mathrm{T}})$.

Now consider a queueing system model in which the system strictly only admits requests of a given class type by using the bandwidth allocated to that class type only. When a region of a particular class type is full, the system will reject requests of that type. From the discussion earlier, we know that the system can admit at most n_{V} video requests, n_{I} image requests and n_{T} text requests without causing disk overload. The system behaves like managing three separate partitions, one for each media object type as follows.

The first partition serving only video requests behaves like a $M/M/n_{\mathrm{V}}/n_{\mathrm{V}}$ queue[2] since each admitted video request acts as if a separate server has been reserved to serve it (via bandwidth reservation) until it departs. The arrival rate of video requests is λ_{V} and the departure rate of each video request is μ_{V}. Figure 1 shows an $M/M/n_{\mathrm{V}}/n_{\mathrm{V}}$ queue for modeling the video partition. The probability that only j video slots out of the allocated n_{V} slots are being occupied, $P_{\mathrm{V}}(j)$, $0 \leq j \leq n_{\mathrm{V}}$, is well known in queueing theory and is given by:

$$P_{\mathrm{V}}(j) = \frac{\dfrac{1}{j!}\left(\dfrac{\lambda_{\mathrm{V}}}{\mu_{\mathrm{V}}}\right)^{j}}{1 + \displaystyle\sum_{k=1}^{n_{\mathrm{V}}} \frac{1}{k!}\left(\frac{\lambda_{\mathrm{V}}}{\mu_{\mathrm{V}}}\right)^{k}}.$$

Here we note that with probability $P_{\mathrm{V}}(j)$, only j slots are occupied by video requests (with the other $n_{\mathrm{V}} - j$ slots not used) and the departure of each of the j video requests will bring a reward of v_{V} to the system. The rate at which one of the j video requests departs the system is $j\mu_{\mathrm{V}}$ since each video request departs independently of one another with a departure rate of μ_{V}. Consequently, with probability $P_{\mathrm{V}}(j)$ the reward rate gained due to request departures is $j\mu_{\mathrm{V}}v_{\mathrm{V}}$. The expected reward rate gained is the sum of $j\mu_{\mathrm{V}}v_{\mathrm{V}}P_{\mathrm{V}}(j)$, $1 \leq j \leq n_{\mathrm{V}}$. Alternatively, when the system rejects a video request arrival because all n_{V} slots are filled, the system is penalized by q_{V}. Since the rejection rate is $\lambda_{\mathrm{V}}P_{\mathrm{V}}(n_{\mathrm{V}})$, the reward rate lost due to request rejections is $\lambda_{\mathrm{V}}q_{\mathrm{V}}P_{\mathrm{V}}(n_{\mathrm{V}})$. Summarizing the above, let $\mathcal{R}_{\mathrm{V}}$ denote the reward rate obtainable from the video partition. Then we have:

$$\mathcal{R}_{\mathrm{V}} = \left(\sum_{j=1}^{n_{\mathrm{V}}} j\mu_{\mathrm{V}} \times v_{\mathrm{V}} \times P_{\mathrm{V}}(j)\right) - \lambda_{\mathrm{V}}q_{\mathrm{V}} \times P_{\mathrm{V}}(n_{\mathrm{V}}). \quad (1)$$

The second partition serving only image requests can be modeled by a $M/M/1^{[n_{\mathrm{I}}]}/K_{\mathrm{I}} \times n_{\mathrm{I}}$ queue since up to n_{I} image requests can be serviced in a single *batch* by the system in a service round.[3] We use the notation $1^{[n_{\mathrm{I}}]}$ to indicate that a group of up to n_{I} image requests can be serviced in batch in one service round. The arrival rate to this queue is λ_{I} and the *batch* departure rate is μ_{I}. Here, $K_{\mathrm{I}} \times n_{\mathrm{I}}$ stands for the size of the image queue (a design parameter) with $K_{\mathrm{I}} \geq 1$. Figure 2 shows a $M/M/1^{[n_{\mathrm{I}}]}/2 \times n_{\mathrm{I}}$ queueing system for modeling the image partition for the case $K_{\mathrm{I}} = 2$. The choice of K_{I} depends on a trade-off between a lower rejection rate (and hence a higher reward rate obtainable) and a higher response time. As K_{I} increases, the rejection rate decreases (since the queue size is larger) at the expense

[2]The notation $M/M/n/n$ means that (i) the arrival process is a Poisson process having the Markovian property with the interarrival time being exponentially distributed; (ii) the departure process is also a Poisson process having the Markovian property with the service time being exponentially distributed; (iii) there are n servers; (iv) there are n slots. Thus, treating the video partition as a $M/M/n_{\mathrm{V}}/n_{\mathrm{V}}$ queue assumes that the interarrival times of video requests are exponentially distributed with rate λ_{V} and the service time per video request is also exponentially distributed with rate μ_{V}. If these assumptions are not true, a more general queueing system may be used. For example, if the service time is generally distributed, we can use a $M/G/n_{\mathrm{V}}/n_{\mathrm{V}}$ queue instead. The same way of calculating the reward rate would still apply.

[3]The queueing system for image second partition is only an approximation since the batch time for processing n_{I} image requests is exactly T_{SR}, not exponentially distributed with the mean time at T_{SR}. Nevertheless, we show later by simulation that the reward value obtained is insensitive to this approximation.

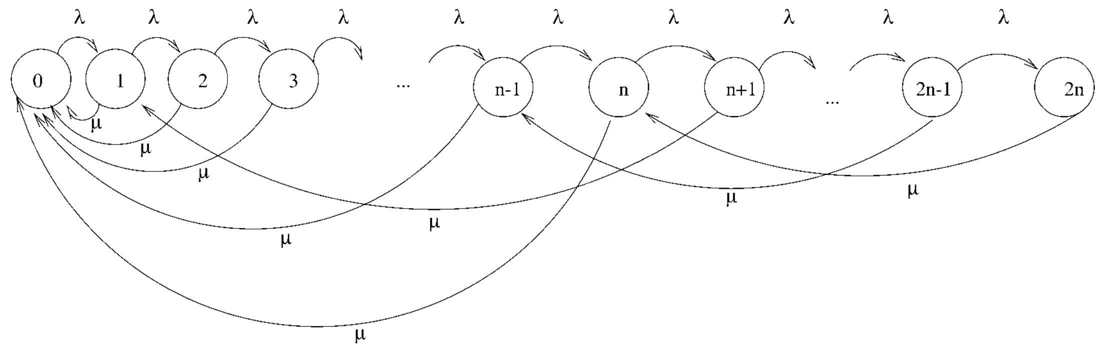

FIGURE 2. $M/M/1^{[n_{\mathrm{I}}]}/2n_{\mathrm{I}}$ model for the partition that serves image requests.

of a higher response time.[4] When $K_{\mathrm{I}} = 1$, the probability that j image requests are in the system, $P_{\mathrm{I}}(j)$, $0 \le j \le n_{\mathrm{I}}$, is given by (see Appendix A for the derivation):

$$P_{\mathrm{I}}(j) = \begin{cases} \dfrac{\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\left(\dfrac{\lambda_{\mathrm{I}}}{\lambda_{\mathrm{I}}+\mu_{\mathrm{I}}}\right)^{j+1} & \text{if } 0 \le j < n_{\mathrm{I}}-1 \\ \left(\dfrac{\lambda_{\mathrm{I}}}{\lambda_{\mathrm{I}}+\mu_{\mathrm{I}}}\right)^{n_{\mathrm{I}}} & \text{if } j = n_{\mathrm{I}}. \end{cases} \tag{2}$$

When $K_{\mathrm{I}} = 2$, $P_{\mathrm{I}}(j)$, $0 \le j \le 2 \times n_{\mathrm{I}}$, is given by (see Appendix B for the derivation):

$$P_{\mathrm{I}}(j) = \begin{cases} \left[\left(\dfrac{\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)\left(\dfrac{\lambda_{\mathrm{I}}+\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)^{2n_{\mathrm{I}}-1-j} - \left(\dfrac{\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)\right. \\ \quad \left.\times\left(\dfrac{\lambda_{\mathrm{I}}+\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)^{n_{\mathrm{I}}-2-j}\left(\dfrac{\lambda_{\mathrm{I}}+(n_{\mathrm{I}}-j)\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)\right] P_{\mathrm{I}}(2n_{\mathrm{I}}) \\ \quad \text{if } 0 \le j \le n_{\mathrm{I}}-1 \\ \left(\dfrac{\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)\left(\dfrac{\lambda_{\mathrm{I}}+\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)^{2n_{\mathrm{I}}-1-j} P_{\mathrm{I}}(2n_{\mathrm{I}}) \\ \quad \text{if } n_{\mathrm{I}} \le j \le 2n_{\mathrm{I}}-1 \\ \dfrac{1}{\left(\dfrac{\lambda_{\mathrm{I}}+\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)^{2n_{\mathrm{I}}} - n_{\mathrm{I}}\left(\dfrac{\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)\left(\dfrac{\lambda_{\mathrm{I}}+\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)^{n_{\mathrm{I}}-1}} \\ \quad \text{if } j = 2n_{\mathrm{I}}. \end{cases} \tag{3}$$

The calculation of the reward rate from the image partition, $\mathcal{R}_{\mathrm{I}}$, is a bit different from that for the video partition because image requests are served in batch and the image queue size has been extended to $K_{\mathrm{I}}n_{\mathrm{I}}$. Here we first note that when the number of image requests occupying the $K_{\mathrm{I}}n_{\mathrm{I}}$ slots in the image queue, say j, is less than n_{I}, all j requests will be serviced in one service round since the system is able to service n_{I} image requests in batch. In this case the request departure rate is $j\mu_{\mathrm{I}}$ and the reward rate gained is $j\mu_{\mathrm{I}}v_{\mathrm{I}}$. When j is greater than or equal to n_{I}, however, only n_{I} requests will be serviced in one service round and the remaining $j - n_{\mathrm{I}}$ would be served in the next service round. In this case, the request departure rate is $n_{\mathrm{I}}\mu_{\mathrm{I}}$ and the reward rate gained is $n_{\mathrm{I}}\mu_{\mathrm{I}}v_{\mathrm{I}}$. Consequently, with probability $P_{\mathrm{I}}(j)$ the reward rate gained due to request departures is $j\mu_{\mathrm{V}}v_{\mathrm{V}}$ if $j < n_{\mathrm{I}}$ and $n_{\mathrm{I}}\mu_{\mathrm{V}}v_{\mathrm{V}}$ otherwise. The expected reward rate gained due to request departures is the sum of these reward rates weighted on their respective probabilities $P_{\mathrm{I}}(j)$, $1 \le j \le n_{\mathrm{I}}$. Again, when the system rejects an image request arrival because all $K_{\mathrm{I}}n_{\mathrm{I}}$ slots are filled, the system is penalized by q_{I}. As the rejection rate is $\lambda_{\mathrm{I}}P_{\mathrm{I}}(K_{\mathrm{I}}n_{\mathrm{I}})$, the reward rate lost due to rejections of image requests is $\lambda_{\mathrm{I}}q_{\mathrm{I}}P_{\mathrm{I}}(K_{\mathrm{I}}n_{\mathrm{I}})$. Summarizing the above, the reward rate obtainable from the image partition, $\mathcal{R}_{\mathrm{I}}$, is calculated as:

$$\mathcal{R}_{\mathrm{I}} = \left(\sum_{j=1}^{n_{\mathrm{I}}-1} j\mu_{\mathrm{I}} \times v_{\mathrm{I}} \times P_{\mathrm{I}}(j)\right) + \left(\sum_{j=n_{\mathrm{I}}}^{K_{\mathrm{I}}n_{\mathrm{I}}} n_{\mathrm{I}}\mu_{\mathrm{I}} \times v_{\mathrm{I}} \times P_{\mathrm{I}}(j)\right) - \lambda_{\mathrm{I}}q_{\mathrm{I}} \times P_{\mathrm{I}}(K_{\mathrm{I}}n_{\mathrm{I}}). \tag{4}$$

The third partition serving only text requests behaves like a $M/M/1^{[n_{\mathrm{T}}]}/K_{\mathrm{T}} \times n_{\mathrm{T}}$ queue where n_{T} is the number of text slots reserved for text requests and $K_{\mathrm{T}} \times n_{\mathrm{T}}$ is again a design parameter standing for the queue size for holding text requests. The arrival rate is λ_{T} and the *batch* departure rate is μ_{T}. Following the discussion above, the reward rate contribution from the text partition, $\mathcal{R}_{\mathrm{T}}$, is calculated as:

$$\mathcal{R}_{\mathrm{T}} = \left(\sum_{j=1}^{n_{\mathrm{T}}-1} j\mu_{\mathrm{T}} \times v_{\mathrm{T}} \times P_{\mathrm{T}}(j)\right) + \left(\sum_{j=n_{\mathrm{T}}}^{K_{\mathrm{T}}n_{\mathrm{T}}} n_{\mathrm{T}}\mu_{\mathrm{T}} \times v_{\mathrm{T}} \times P_{\mathrm{T}}(j)\right) - \lambda_{\mathrm{T}}q_{\mathrm{T}} \times P_{\mathrm{T}}(K_{\mathrm{T}}n_{\mathrm{T}}). \tag{5}$$

While a video request is served continuously on a cycle by cycle basis until it departs, an image or a text request completes its service immediately when it is serviced in a single cycle. Thus, the video departure rate μ_{V} per request can be estimated *a priori* at static time, depending on the application domain (e.g. a video user will spend minutes for viewing a news clip and hours for viewing

[4]If $T_{\mathrm{SR}} = 1$ s, then K_{I} should be either 1 or 2 so that the maximum response time for an image request does not exceed 2 s, which is about what a user can tolerate in typical applications.

a movie). Alternatively, the 'batch' service rate for image (text) requests can be parameterized as:

$$\mu_{\mathrm{I}} = \frac{1}{T_{\mathrm{SR}}} \quad \text{and} \quad \mu_{\mathrm{T}} = \frac{1}{T_{\mathrm{SR}}}$$

due to the fact that a batch of n_{I} image requests (n_{T} text requests) will be serviced in one service cycle.

Let $\mathcal{R}$ denote the system reward rate, defined as the reward amount received by the system per time unit. Then,

$$\mathcal{R} = \mathcal{R}_{\mathrm{V}} + \mathcal{R}_{\mathrm{I}} + \mathcal{R}_{\mathrm{T}}. \tag{6}$$

When $K_{\mathrm{I}} = 2$ and $K_{\mathrm{T}} = 2$, $\mathcal{R}$ is given by:

$$\begin{aligned}
\mathcal{R} = {} & \sum_{i=1}^{n_{\mathrm{V}}} i\mu_{\mathrm{V}} \times v_{\mathrm{V}} \times \frac{\frac{1}{i!}\left(\frac{\lambda_{\mathrm{V}}}{\mu_{\mathrm{V}}}\right)^{i}}{1+\sum_{j=1}^{n_{\mathrm{V}}}\frac{1}{j!}\left(\frac{\lambda_{\mathrm{V}}}{\mu_{\mathrm{V}}}\right)^{j}} \\
& + \sum_{i=1}^{n_{\mathrm{I}}} i\mu_{\mathrm{I}} \times v_{\mathrm{I}} \times \left[\left(\frac{\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)\left(\frac{\lambda_{\mathrm{I}}+\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)^{2n_{\mathrm{I}}-1-j}\right. \\
& \left. - \left(\frac{\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)\left(\frac{\lambda_{\mathrm{I}}+\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)^{n_{\mathrm{I}}-2-j}\left(\frac{\lambda_{\mathrm{I}}+(n_{\mathrm{I}}-j)\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)\right] \\
& \times \left[\left(\frac{\lambda_{\mathrm{I}}+\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)^{2n_{\mathrm{I}}} - n_{\mathrm{I}}\left(\frac{\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)\left(\frac{\lambda_{\mathrm{I}}+\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)^{n_{\mathrm{I}}-1}\right]^{-1} \\
& + \sum_{i=n_{\mathrm{I}}}^{2n_{\mathrm{I}}-1} n_{\mathrm{I}}\mu_{\mathrm{I}} \times v_{\mathrm{I}} \times \left[\left(\frac{\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)\left(\frac{\lambda_{\mathrm{I}}+\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)^{2n_{\mathrm{I}}-1-j}\right] \\
& \times \left[\left(\frac{\lambda_{\mathrm{I}}+\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)^{2n_{\mathrm{I}}} - n_{\mathrm{I}}\left(\frac{\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)\left(\frac{\lambda_{\mathrm{I}}+\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)^{n_{\mathrm{I}}-1}\right]^{-1} + n_{\mathrm{I}}\mu_{\mathrm{I}} \\
& \times v_{\mathrm{I}} \times \left[\left(\frac{\lambda_{\mathrm{I}}+\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)^{2n_{\mathrm{I}}} - n_{\mathrm{I}}\left(\frac{\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)\left(\frac{\lambda_{\mathrm{I}}+\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)^{n_{\mathrm{I}}-1}\right]^{-1} \\
& + \sum_{i=1}^{n_{\mathrm{T}}} i\mu_{\mathrm{T}} \times v_{\mathrm{T}} \times \left[\left(\frac{\mu_{\mathrm{T}}}{\lambda_{\mathrm{T}}}\right)\left(\frac{\lambda_{\mathrm{T}}+\mu_{\mathrm{T}}}{\lambda_{\mathrm{T}}}\right)^{2n_{\mathrm{T}}-1-j}\right. \\
& \left. - \left(\frac{\mu_{\mathrm{T}}}{\lambda_{\mathrm{T}}}\right)\left(\frac{\lambda_{\mathrm{T}}+\mu_{\mathrm{T}}}{\lambda_{\mathrm{T}}}\right)^{n_{\mathrm{T}}-2-j}\left(\frac{\lambda_{\mathrm{T}}+(n_{\mathrm{T}}-j)\mu_{\mathrm{T}}}{\lambda_{\mathrm{T}}}\right)\right] \\
& \times \left[\left(\frac{\lambda_{\mathrm{T}}+\mu_{\mathrm{T}}}{\lambda_{\mathrm{T}}}\right)^{2n_{\mathrm{T}}} - n_{\mathrm{T}}\left(\frac{\mu_{\mathrm{T}}}{\lambda_{\mathrm{T}}}\right)\left(\frac{\lambda_{\mathrm{T}}+\mu_{\mathrm{T}}}{\lambda_{\mathrm{T}}}\right)^{n_{\mathrm{T}}-1}\right]^{-1} \\
& + \sum_{i=n_{\mathrm{T}}}^{2n_{\mathrm{T}}-1} n_{\mathrm{T}}\mu_{\mathrm{T}} \times v_{\mathrm{T}} \times \left[\left(\frac{\mu_{\mathrm{T}}}{\lambda_{\mathrm{T}}}\right)\left(\frac{\lambda_{\mathrm{T}}+\mu_{\mathrm{T}}}{\lambda_{\mathrm{T}}}\right)^{2n_{\mathrm{T}}-1-j}\right] \\
& \times \left[\left(\frac{\lambda_{\mathrm{T}}+\mu_{\mathrm{T}}}{\lambda_{\mathrm{T}}}\right)^{2n_{\mathrm{T}}} - n_{\mathrm{T}}\left(\frac{\mu_{\mathrm{T}}}{\lambda_{\mathrm{T}}}\right)\left(\frac{\lambda_{\mathrm{T}}+\mu_{\mathrm{T}}}{\lambda_{\mathrm{T}}}\right)^{n_{\mathrm{T}}-1}\right]^{-1} \\
& + n_{\mathrm{T}}\mu_{\mathrm{T}} \times v_{\mathrm{T}} \times \left[\left(\frac{\lambda_{\mathrm{T}}+\mu_{\mathrm{T}}}{\lambda_{\mathrm{T}}}\right)^{2n_{\mathrm{T}}} - n_{\mathrm{T}}\left(\frac{\mu_{\mathrm{T}}}{\lambda_{\mathrm{T}}}\right)\right. \\
& \left. \times \left(\frac{\lambda_{\mathrm{T}}+\mu_{\mathrm{T}}}{\lambda_{\mathrm{T}}}\right)^{n_{\mathrm{T}}-1}\right]^{-1} - \lambda_{\mathrm{V}}q_{\mathrm{V}} \times \left[\frac{1}{n_{\mathrm{V}}!}\left(\frac{\lambda_{\mathrm{V}}}{\mu_{\mathrm{V}}}\right)^{n_{\mathrm{V}}}\right] \\
& \times \left[1+\sum_{j=1}^{n_{\mathrm{V}}}\frac{1}{j!}\left(\frac{\lambda_{\mathrm{V}}}{\mu_{\mathrm{V}}}\right)^{j}\right]^{-1} - \lambda_{\mathrm{I}}q_{\mathrm{I}} \times \left[\left(\frac{\lambda_{\mathrm{I}}+\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)^{2n_{\mathrm{I}}}\right. \\
& \left. - n_{\mathrm{I}}\left(\frac{\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)\left(\frac{\lambda_{\mathrm{I}}+\mu_{\mathrm{I}}}{\lambda_{\mathrm{I}}}\right)^{n_{\mathrm{I}}-1}\right]^{-1} - \lambda_{\mathrm{T}}q_{\mathrm{T}} \\
& \times \left[\left(\frac{\lambda_{\mathrm{T}}+\mu_{\mathrm{T}}}{\lambda_{\mathrm{T}}}\right)^{2n_{\mathrm{T}}} - n_{\mathrm{T}}\left(\frac{\mu_{\mathrm{T}}}{\lambda_{\mathrm{T}}}\right)\left(\frac{\lambda_{\mathrm{T}}+\mu_{\mathrm{T}}}{\lambda_{\mathrm{T}}}\right)^{n_{\mathrm{T}}-1}\right]^{-1}.
\end{aligned} \tag{7}$$

For the case when K_{I} or K_{T} is greater than 2, no closed-form solution exists. However, one can use existing performance analysis tools such as SPNP [10] to define and evaluate a queueing model similar to Figure 2 (e.g. with the last state being $4n$ when $K_{\mathrm{I}} = 4$) to numerically compute $\mathcal{R}_{\mathrm{I}}$ and $\mathcal{R}_{\mathrm{T}}$ based on Equations (4) and (5).

When given a set of model parameter values of λ_{V}, μ_{V}, λ_{I}, μ_{I}, λ_{T}, μ_{T}, v_{V}, q_{V}, v_{I}, q_{I}, v_{T} and q_{T}, one can determine the best partition $(n_{\mathrm{V}}, n_{\mathrm{I}}, n_{\mathrm{T}})$, say $(n_{\mathrm{V}}^*, n_{\mathrm{I}}^*, n_{\mathrm{T}}^*)$, that will maximize the reward rate. To search for the optimal set $(n_{\mathrm{V}}^*, n_{\mathrm{I}}^*, n_{\mathrm{T}}^*)$, we formulate the search problem as an optimization problem as follows. Let N_k be the maximum number of requests of type k that the system is able to admit statistically when all the time duration T_{SR} is allocated to service requests of class k only. Thus, there will be three parameters, i.e. N_{V}, N_{I} and N_{T} for video, image and text request types respectively which we can compute at static time. Then the optimal set $(n_{\mathrm{V}}^*, n_{\mathrm{I}}^*, n_{\mathrm{T}}^*)$ is the one that maximizes $\mathcal{R}$ (using Equation (7)) subject to the condition that

$$\left\lfloor \frac{N_{\mathrm{T}}}{N_{\mathrm{V}}} \times n_{\mathrm{V}} \right\rfloor + \left\lfloor \frac{N_{\mathrm{T}}}{N_{\mathrm{I}}} \times n_{\mathrm{I}} \right\rfloor + n_{\mathrm{T}} = N_{\mathrm{T}}. \tag{8}$$

Condition (8) above is required to ensure that individual bandwidth resources allocated to the video, image and text partitions sum to the total disk bandwidth available. Here we normalize the total disk bandwidth with respect to the bandwidth required to service one text request, thus treating the system as containing a total of N_{T} text slots in a service duration. Since a video slot and an image slot require on average $N_{\mathrm{T}}/N_{\mathrm{V}}$ and $N_{\mathrm{T}}/N_{\mathrm{I}}$ text slots respectively, a valid $(n_{\mathrm{V}}, n_{\mathrm{I}}, n_{\mathrm{T}})$ combination must satisfy Condition (8) above so the total number of text slots is equal to N_{T}.

The number of possible cases of $(n_{\mathrm{V}}, n_{\mathrm{I}}, n_{\mathrm{T}})$ from which the optimal set $(n_{\mathrm{V}}^*, n_{\mathrm{I}}^*, n_{\mathrm{T}}^*)$ can be found is upper bounded by the number of ways of dividing N_{T} into three sets subject to Condition (8) above. Thus the time complexity involved in enumerating and applying Equation (7) and Condition (8) is $O(N_{\mathrm{T}}^2)$. Once we find the best $(n_{\mathrm{V}}^*, n_{\mathrm{I}}^*, n_{\mathrm{T}}^*)$ set for each arrival rate set $(\lambda_{\mathrm{V}}, \lambda_{\mathrm{I}}, \lambda_{\mathrm{T}})$, we can then build a lookup table recording their relationship, along with the reward rate obtainable.

In cases when N_{T}^2 is a large number, the exhaustive algorithm of time complexity $O(N_{\mathrm{T}}^2)$ to find the best $(n_{\mathrm{V}}^*, n_{\mathrm{I}}^*, n_{\mathrm{T}}^*)$ set may still be computationally expensive. We consider a *nearest neighbor* search algorithm as an alternative to further reduce the time complexity to $O(N_{\mathrm{T}})$. This approach yields a near-optimal solution with results fairly close to those obtained by exhaustive search. The idea is to first fix one value among n_{V}, n_{I} and n_{T}, after which we fix one of the remaining two. We adopt the following simple heuristic: the object type among all with the largest product of arrival rate and reward value is selected first and the one with the second largest product is selected next. The rationale is that an object type selected this way will likely generate a larger reward than others, so it is more important to fix a proper value for this object type. Consider the case that n_T is selected first. Then instead of trying every possible combination of (n_V, n_I, n_T), only n_T varies first to take on all possible values in the range $(0, N_T)$ one at a time. During an iteration while n_T is tested at a particular value, the remaining $N_T - n_T$ text slots are allocated to n_V and n_I in proportion to the arrival rate and reward value products, i.e.

$$n_V = \left\lfloor \frac{N_T - n_T}{N_T/N_V} \right\rfloor \times \frac{\lambda_V N_V}{\lambda_V N_V + \lambda_I N_I}$$

and

$$n_I = \left\lfloor \frac{N_T - n_T}{N_T/N_I} \right\rfloor \times \frac{\lambda_I N_I}{\lambda_V N_V + \lambda_I N_I}.$$

The best reward value yielded in these $N_T + 1$ iterations with n_T value varying from 0 to N_T will fix n_T in this case. Suppose n_I is selected next. Then, given that n_T is already fixed, n_I will vary in the range of $(0, \lfloor (N_T - n_T)/(N_T/N_I) \rfloor)$ while n_V will take the residue to see if the reward can be further improved. Overall, the nearest neighbor algorithm requires at most $N_T + 1$ iterations to fix n_T and at most $N_I + 1$ iterations to fix n_I (and consequently n_V too). If the text object is not selected first, the number of iterations would be even less. In general, the nearest neighbor algorithm for searching for a near optimal solution will be of complexity $O(N_T)$.

3.2. Dynamic admission control

Our algorithm makes use of the lookup table at run time. First, it uses the table as a basis to dynamically adapt to workload changes by performing bandwidth reallocations to serve video, image and text requests. It does so by changing to another (n_V^*, n_I^*, n_T^*) value set periodically based on the monitored input arrival rates detected from the last monitoring period. The monitoring period needs to be fine tuned. It can be determined by the service provider based on anticipated busy/slow switch hours at which a change of arrival rates is likely; it can also be preset by analyzing user profiles collected over a long period of time. Note that a change of the (n_V^*, n_I^*, n_T^*) set corresponds to a change of bandwidth allocation (f_V^*, f_I^*, f_T^*).

Second, the (n_V^*, n_I^*, n_T^*) value set used in the current monitoring period is used as the basis for performing admission control. For CM objects (video), the system will admit at most n_V^* video requests so that the bandwidth and delay requirements of video requests are satisfied. For DM objects (image and text), we use the response time as the guiding criterion to allow possibly more than n_I^* image and n_T^* text requests to be admitted. We first note that n_I^* image and n_T^* text requests will be processed in each service duration T_{SR} based on our algorithm. Therefore, if we set the image (text) queue size to be $K_I n_I^*$ ($K_T n_T^*$ respectively) then the worst case response time for an image (a text) request will be $K_I T_{SR}$ ($K_T T_{SR}$ respectively). This provides a bound on the worst case response time for each image/text request, as well as a condition for a sanity check to determine if a switch to another (n_V^*, n_I^*, n_T^*) value set should be performed dynamically. After K_I and K_T are determined this way, the system can admit up to $K_I n_I^*$ image and $K_T n_T^*$ text requests, with the bandwidth and response time requirements of these admitted DM requests satisfied. Allowing more image and text requests to be admitted into their respective queues over sizes n_I and n_T respectively will only improve the reward rate obtainable since it decreases the probability of these discrete media requests being rejected immediately on arrival (and thus decreases the probability of penalties being assessed due to rejections).

Lastly, we further exploit the leftover bandwidth from the video partition to serve additional image/text requests. In any service round, we use the classic SCAN algorithm to order all requests such that the disk read/write heads only traverse the disk in one direction to retrieve all needed data for all video/image/text requests to minimize the seek time. We also maintain a common schedule queue dynamically formed on a cycle by cycle basis filled with n_V^* video requests, n_I^* image requests and n_T^* text requests. If we discover that in any cycle the total time to service all the requests is smaller than T_{SR}, then additional image and text requests at the front of their queues (if available) can also be moved into the common schedule queue for execution in the current cycle. The ratio by which image to text requests will be put into the common schedule queue is based on f_I^*/f_T^*. That is, with probability $f_I^*/(f_I^* + f_T^*)$ an image request is selected to be put into the common queue and with probability $f_T^*/(f_I^* + f_T^*)$ a text request is selected. The transfer of image or text requests from their waiting queues to the common schedule queue stops when adding another request will make the total service time exceed T_{SR}.

4. ANALYSIS AND SIMULATION VALIDATION

We have conducted detailed numerical analysis and simulation validation of a digital library multimedia server to demonstrate the effectiveness of our approach. The disk subsystem selected is a disk array with four disks with an average seek time of 11 ms, a rotational latency of 5.5 ms, and a collective read/write rate $\mu = 33.3$ MBps. We set T_{SR} as 1 s[5] and set the block size D equal to four sectors, i.e. 2K bytes stripped evenly across the four disks in the disk array with the sector size 512 bytes. This disk organization well justifies our assumption that an entire image/text object can be processed in a single cycle.

The size of an image object is assumed to be uniformly distributed in [10 kB, 500 kB] while that of a text object is uniformly distributed in [1 kB, 50 kB]. The sizes of image and text objects requested are randomly generated from their respective ranges. For the video clips, we take a trace file of the movie *Star Wars* which consists of 7200 groups

[5]Other T_{SR} values were also used with the results exhibiting a similar trend and thus they are not reported. For a classic tradeoff analysis between the magnitude of T_{SR} versus the memory requirement, and the number of requests admissible, see [9].

of pictures (GOPs) each covering 0.5 s of playback time. Each GOP consists of 12 frames (I, P and B frames) arranged in a fixed sequence with their sizes varying from one GOP to another GOP. Different video clips are simulated by going to different starting GOPs in the trace file. When starting a simulation run, the disk array is filled with images, texts and video clips until it is 90% full. The location of each image, text and video clip is recorded in a file allocation table. During a service round, all requests being served are ordered based on the SCAN algorithm [11] in the common schedule queue. The seek time, rotational latency and read time of each request are computed depending on the last read/write head position, the size of object retrieved (this depends on how many blocks are to be retrieved in the current cycle), and the disk location (sector and track numbers) of the starting data block that contains the object. This allows the service time to serve requests in the common schedule queue to be computed and compared with the duration of the service round, as described in Section 3.2.

With the simulation environment parameters determined, we first compute the values of N_V off-line by using a classic statistical admission control using the object size distribution information described above so that the probability of disk overload does not exceed 10^{-4}. By using the average seek time of 11 ms, rotational latency of 5.5 ms and the read/write rate $\mu = 33$ MBps, it turned out that $N_V = 53$, that is, 53 video requests can be serviced per cycle so that the probability of disk overload is less than 10^{-4}; we then performed a similar procedure for image requests only, which yields $N_I = 37$, meaning 37 image requests can be processed per second by the disk without causing disk overload statistically. Lastly, we obtained $N_T = 57$.

Since video requests are served across cycles until they depart, while image and text requests are served only in a single cycle and depart, this means that the system is able to sustain an image request rate λ_I (λ_T) of approximately N_I (N_T) arrivals per T_{SR}, i.e. 37 image (correspondingly 57 text) arrivals per second. To create a high-traffic situation, we consider λ_V (the arrival rate of video requests) in the range of [10, 100] arrivals/min, λ_I in the range of [100, 2000] arrivals/min, and λ_T in the range of [100, 2000] arrivals/min. Below we analyze the effects of these arrival rates, along with other variables including μ_V (departure rate of video requests), v_V, v_I, v_T, q_V, q_I and q_T (reward/penalty values) on the performance of cost-based reservation algorithms.

4.1. Comparison basis: video-first and greedy algorithms

Our algorithm is compared with two algorithms:

(i) *Video-First*: the video-first algorithm always gives the highest priority to video requests (an example of which can be found in [5]). Under this algorithm, all resources are allocated to video requests with the leftover bandwidth being used to serve image and text requests. Specifically, $(n_V, n_I, n_T) = (N_V, 0, 0)$. To provide a fair comparison with the reservation algorithm, we allow image and text requests to be admitted into the system with queue size limits of $K_I n_I^*$ and $K_T n_T^*$, respectively, where n_I^* and n_T^* are the optimal values determined by the reservation algorithm based on Equation (7) and Condition (8). If in any cycle there is leftover bandwidth available, image and text requested waiting in the queues will be served based on the ratio of f_I^*/f_T^* as discussed in Section 3.2. A new image (text) request arriving at the system when its queue is full will be rejected.

(ii) *Greedy*: the greedy algorithm allocates disk bandwidth to each object type in proportion to the product of its reward and arrival rate so that an object type with a higher 'reward arrival rate' will get more disk bandwidth. Specifically,

$$n_V = \frac{v_V \lambda_V N_V}{v_V \lambda_V + v_I \lambda_I + v_T \lambda_T} \tag{9}$$

$$n_I = \frac{v_I \lambda_I N_I}{v_V \lambda_V + v_I \lambda_I + v_T \lambda_T} \tag{10}$$

and

$$n_T = \frac{v_T \lambda_T N_T}{v_V \lambda_V + v_I \lambda_I + v_T \lambda_T}. \tag{11}$$

4.2. Analysis

In this section, we apply Equation (7) and Condition (8) to compare the lower-bound reward rates[6] obtainable and the corresponding (n_V, n_I, n_T) sets by our proposed reservation algorithm versus those by the video-first and greedy algorithms. We also analyze the sensitivity of model parameters upon the reward rate obtainable. Recall that our algorithm determines the (n_V, n_I, n_T) set either exactly by exhaustive search (with time complexity $O(T_N^2)$) or approximately with nearest neighbor search (with time complexity $O(T_N)$). The greedy algorithm determines the (n_V, n_I, n_T) set based on Equations (9)–(11). For the video-first algorithm, $(n_V, n_I, n_T) = (N_V, 0, 0)$.

Table 2 shows lower-bound reward rates obtained by our optimal algorithm in column 4 and the corresponding optimal (n_V, n_I, n_T) sets in column 5 under several combinations of object arrival rates for a case in which $\mu_V = 1$, $K_I = 2$ and $K_T = 2$, and video requests are considered more important than other object types such that the reward and penalty values of video object types are higher than others, i.e. $v_V = 20$, $v_I = 5$, $v_T = 2$, $q_V = 10$, $q_I = 2$ and $q_T = 1$. For comparison, Table 2 also lists the lower-bound reward rates obtained by the nearest neighbor algorithm in column 6 and the corresponding (n_V, n_I, n_T) sets in column 7. By comparing the reward rates obtained in column 4 (exhaustive search) with those in column 6 (nearest neighbor search), we see that the approximate solutions obtained based on the nearest neighbor (nnb) search technique with time complexity $O(N_T)$ are fairly accurate against exact solutions obtained via exhaustive search with time complexity $O(N_T^2)$.

[6]Recall that Equation (7) gives the lower bound of the reward rate obtainable since it does not model the leftover disk bandwidth from the video partition to serve pending image and text requests.

TABLE 2. Reward rates obtainable and (n_V, n_I, n_T) calculated.

λ_V	λ_I	λ_T	Optimal reward	Optimal (n_V, n_I, n_T)	nnb reward	nnb (n_V, n_I, n_T)	Greedy reward	Greedy (n_V, n_I, n_T)	vf reward
10	100	100	14.9	(25, 12, 11)	14.9	(27, 9, 14)	14.4	(12, 21, 11)	−3
10	400	400	48.7	(14, 16, 17)	48.7	(14, 16, 17)	46.0	(4, 25, 14)	−23
10	800	800	83.2	(0, 22, 23)	82.7	(1, 21, 23)	78.6	(2, 26, 14)	−50
10	1200	1200	99.7	(0, 24, 20)	99.6	(0, 22, 23)	93.5	(1, 26, 15)	−76
10	1600	1600	97.8	(0, 24, 20)	97.3	(0, 22, 23)	91.0	(1, 26, 15)	−103
10	2000	2000	85.2	(0, 24, 20)	85.2	(0, 24, 20)	78.4	(1, 26, 15)	−130
50	100	100	22.7	(44, 4, 3)	22.7	(44, 4, 3)	18.1	(31, 11, 6)	8
50	400	400	46.4	(24, 13, 11)	46.4	(24, 13, 11)	43.5	(14, 19, 12)	−11
50	800	800	76.6	(1, 22, 22)	75.2	(5, 19, 22)	72.3	(8, 22, 14)	−38
50	1200	1200	93.1	(0, 24, 20)	92.5	(0, 23, 21)	80.2	(6, 24, 13)	−65
50	1600	1600	91.2	(0, 24, 20)	90.0	(0, 23, 21)	79.1	(4, 24, 15)	−91
50	2000	2000	78.6	(0, 24, 20)	75.4	(0, 27, 15)	65.5	(4, 25, 14)	−118
100	100	100	15.7	(44, 4, 3)	15.7	(44, 4, 3)	13.9	(39, 7, 4)	2
100	400	400	38.4	(27, 11, 11)	38.2	(25, 12, 11)	37.5	(22, 15, 10)	−17
100	800	800	68.3	(1, 22, 22)	65.9	(3, 18, 26)	60.0	(14, 19, 12)	−44
100	1200	1200	84.7	(0, 24, 20)	84.6	(0, 22, 23)	66.6	(10, 21, 13)	−70
100	1600	1600	82.8	(0, 24, 20)	81.6	(0, 23, 21)	62.2	(8, 22, 14)	−97
100	2000	2000	70.2	(0, 24, 20)	67.1	(0, 27, 15)	49.6	(7, 23, 14)	−124

$v_V = 20$, $v_I = 5$, $v_T = 2$, $q_V = 10$, $q_I = 2$ and $q_T = 1$.

Also shown in Table 2 are lower-bound reward rates obtainable and the corresponding (n_V, n_I, n_T) sets from the greedy algorithm in columns 8 and 9, and lower-bound reward rates obtainable from the video-first (vf) algorithm in column 10. Here we observe that by comparing the optimal reward rates obtainable by our reservation algorithm in columns 4 (exact) and 6 (approximate) with those obtainable by the greedy and video-first algorithms in columns 8 and 10, our reservation algorithm significantly outperforms both the greedy and video-first algorithms over a wide range of request arrival rates. In particular, for the video-first algorithm, we see that the lower-bound reward rate obtainable can even be a negative value (representing loss) due to improper allocations of resources to video requests exclusively.

Below we analyze the effects of arrival rates (λ_V, λ_I and λ_T), per-video request departure rate (μ_V) and reward/penalty values (v_V, q_V, v_I, q_I, v_T and q_T) on the reward rate obtainable. Figure 3 shows the sensitivity of arrival rates on the reward rate obtainable by our proposed reservation algorithm with the reward/penalty values and the per-video departure rate set at $v_V = 20$, $v_I = 5$, $v_T = 2$, $q_V = 10$, $q_I = 2$, $q_T = 1$ and $\mu_V = 1$. Here the X-coordinate is the video request arrival rate λ_V varying in the range of 10–100 while the Y-coordinate is the reward rate. Each data point in the diagram is a lower-bound reward rate obtainable by the reservation algorithm. Five curves are shown with varying discrete object arrival rates. For ease of presentation, the image request arrival rate is set to be the same as the text arrival rate. We see that when the system is relatively lightly loaded by image and text requests (as in the bottom two curves), the system's reward increases as the video request rate increases. However, when the video arrival rate exceeds a threshold such that the system becomes more heavily loaded, the reward rate then decreases as the video arrival rate increases. In contrast, when the system is heavily loaded by image and text requests (as in the top two curves), the system's obtainable reward decreases as the video request rate increases due to more and more requests being rejected.

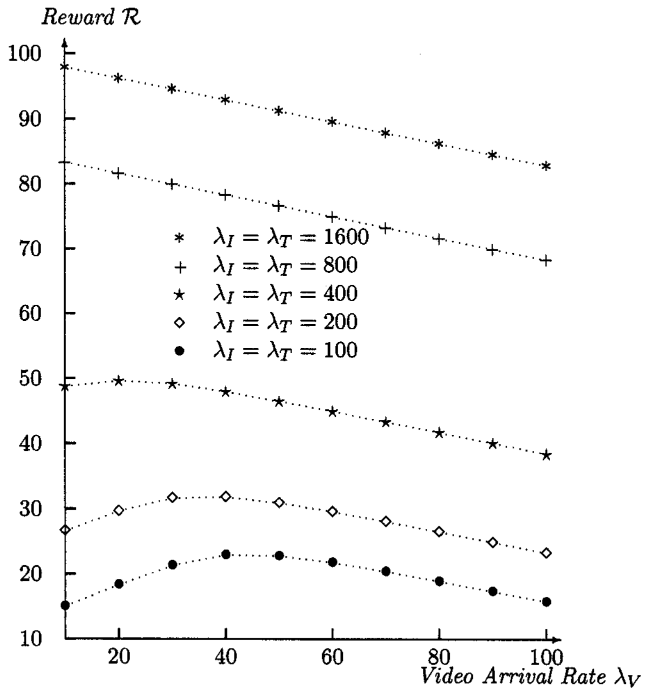

FIGURE 3. Sensitivity of request arrival rates upon reward rate obtainable.

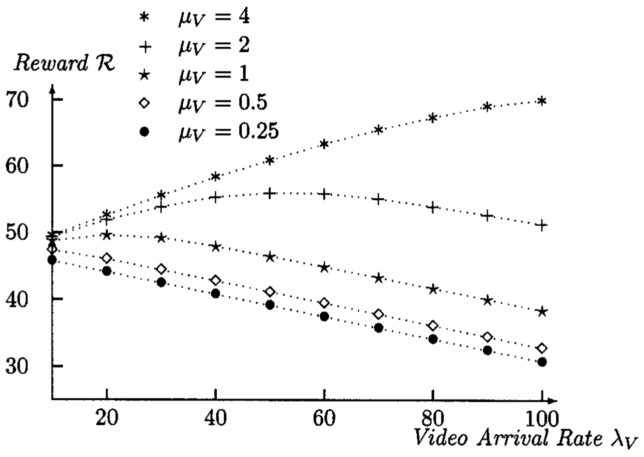

FIGURE 4. Sensitivity of per-video departure rate upon reward rate.

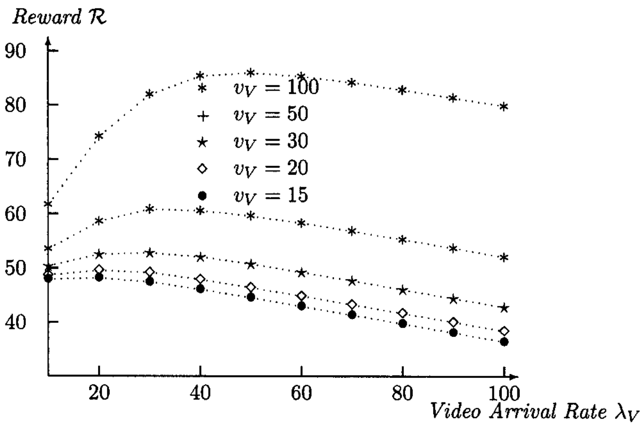

FIGURE 5. Sensitivity of per-video reward value v_V upon reward rate.

Figure 4 shows the sensitivity of the per-video departure rate (μ_V) upon the lower-bound reward rate obtainable under the same set of reward/penalty values, with $\mu_V = 1$ being the base case for comparison (the middle curve). Here we present the case in which $\lambda_I = \lambda_T = 400$ as other arrival-rate cases exhibit the same trend. We observe that as the per-video departure rate increases (toward the upper curves), the reward rate obtainable increases until the video arrival rate exceeds a threshold beyond which the system is heavily loaded and must reject requests to avoid system overload. Alternatively, when the per-video departure rate decreases (toward the bottom curves), each video request will stay in the system for a relatively long period of time. In this case, the system tends to admit fewer video requests (i.e. smaller n_V) since the reward rate obtainable from the video partition is low. As a result, as the video arrival rate increases (toward the right of the X-coordinate), the system tends to reject video requests, thus lowering the total reward rate obtainable due to penalties applied.

Figure 5 shows the sensitivity of the per-video reward value (v_V) upon the reward rate obtainable. There are five curves with varying v_V values shown in the figure for the case in which $\lambda_I = \lambda_T = 400$, $v_I = 5$, $v_T = 2$, $q_V = 10$, $q_I = 2$, $q_T = 1$ and $\mu_V = 1$. The base case for comparison is the second bottom curve at which $v_V = 20$. Here we observe that as v_V increases (toward the upper curves), the total reward rate obtainable also increases because each video request departure will bring a higher value to the system. Moreover, as the per-video reward value increases, the system tends to admit more video requests in the video partition. Consequently, the system is able to accommodate a higher video request rate. This is reflected by the fact that as the per-video reward value increases, the *threshold* video request rate beyond which the reward rate decreases (because of system overload) is shifted toward a higher value. A similar trend is also observed as we analyze the sensitivity of the reward value of discrete object requests (image or text) on the reward rate obtainable.

4.3. Simulation results

In Section 4.2, we compared the reservation algorithm with both the video-first and greedy algorithms without considering the effect of leftover bandwidth from the video partition to serve pending image and text requests. Thus the comparison was on the lower-bound reward rates obtainable by these algorithms. In this section, we compare these algorithms by means of a detailed simulation study that models the use of leftover bandwidth from the video partition to serve additional image/text requests.

For the purpose of validating analytical results, we choose the six cases listed in the middle rows of Table 2, with the first three cases representing light-load situations and the last three cases representing heavy-load situations. Figure 6 shows the reward rate $\mathcal{R}$ obtained by our reservation algorithm (theory and via simulation) versus that obtained by the video-first and greedy algorithms (via simulation). While the system is under a (n_V, n_I, n_T) set, it admits and rejects users, as well as scheduling requests, in accordance with the dynamic admission control algorithm described in Section 3.2, which considers the use of leftover bandwidth to serve pending image and text requests. Each reward rate obtained from simulation is within 95% confidence interval with 10% confidence accuracy by utilizing the batch mean analysis method. For comparison purposes, we also show the theoretical reward rate obtained from Equation (7).

Here we first observe that when the system operates under our cost-based admission control algorithm, the reward rates reported via simulation are close to the predicted lower-bound reward rates obtainable when the system is lightly loaded (the first two cases) since all requests regardless of object types can be served satisfactorily. However, when the system is more heavily loaded (the next three cases), we see that the reward rate obtained by our dynamic reservation algorithm is higher than that calculated from Equation (7). This is so because the theoretical reward rate is based on the assumption of no cross-over among video/image/text partitions, while in the simulation we allow the leftover bandwidth from the video partition to be used by pending image/text requests, the effect of which is more pronounced

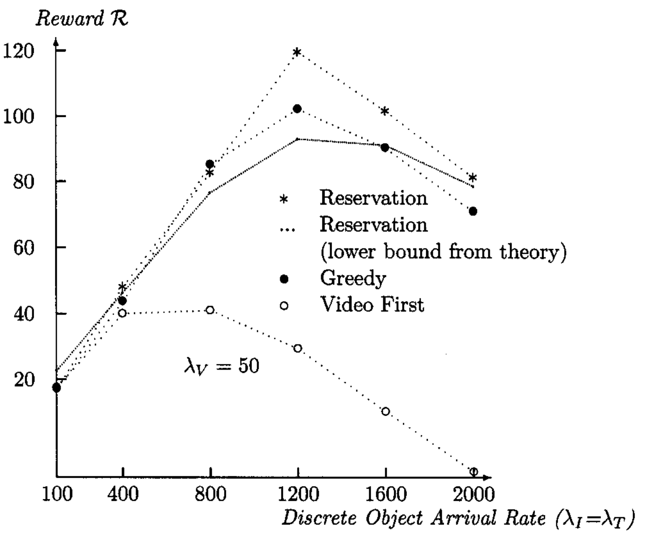

FIGURE 6. Comparing reward rate $\mathcal{R}$.

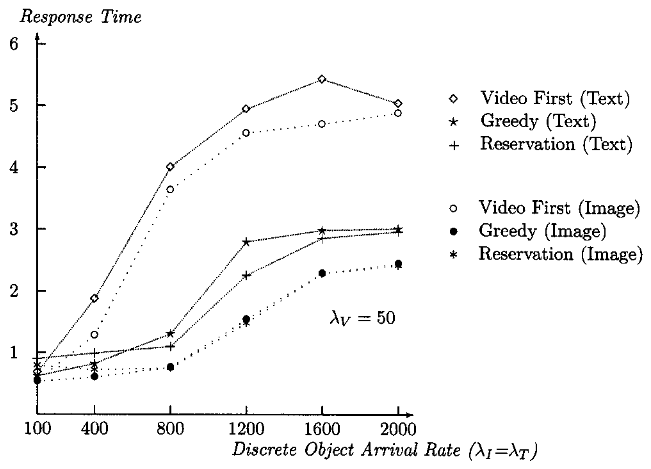

FIGURE 7. Comparing response time per DM request.

under heavy-load conditions. That is, under heavy-load conditions, it is more likely that there will always be pending image/text requests waiting to be served in the image/text queues. Finally when the system is very heavily loaded (case 6), the predicted lower-bound reward rate obtainable is close to that obtained from the simulation again because in this case the high arrival rates of DM requests make the image and text queues virtually full all the time consuming all the resources in the system, i.e. the system reaches its limit in this case and the theoretical lower-bound reward rate is close to the highest achievable reward rate by the system.

Next we observe that in the first two cases, the reward rates under our reservation algorithm and video-first algorithm are virtually the same. This is because the arrival rates in the first two cases represent lightly loaded situations, so the system is able to accommodate most users anyway. However, as the system becomes more heavily loaded, the disk array begins to experience overload and requests are rejected more often. In this case, our reservation algorithm outperforms the video-first algorithm in terms of the reward rate obtained by the system since it can more judiciously allocate resources to requests for different object types with the goal of optimizing the overall system reward, instead of always allocating resources to video users first. The same trend is also observed as we compare the reservation algorithm with the greedy algorithm, i.e. when the system is heavily loaded, the system reward rate obtained by the reservation algorithm is consistently greater than that obtained by the greedy algorithm by a margin of 10–15%. Here we should mention that the reward rate obtained is 'value per unit time' so a difference of 10–15% is considered significant.

Figure 7 compares the average response time per DM request (image or text) obtained by the reservation algorithm versus those by the video-first and greedy algorithms under the same set of testing conditions as in Figure 6. Only the response times of admitted DM requests are measured in the simulation. Similar to Figure 6, we see that when the system is relatively lightly loaded, the response time difference between our reservation algorithm and the video-first algorithm is small. However, as the system becomes more heavily loaded, since our reservation algorithm specifically reserves resources to serve image and text requests, the difference in response time per DM request becomes more significant (between 1 and 5 s).

We also observe that the greedy algorithm compares favorably with the reservation algorithm in the first two cases under which the system is lightly loaded and 'excessive' resources are allocated to serve DM requests by the greedy algorithm. As a result, all DM requests admitted essentially would be served within a single service round. This is in contrast to the reservation algorithm which exercises a tighter control over resources allocated to serve DM requests in these cases with the objective of maximizing the reward rate of the system. Consequently, some DM objects admitted may have to wait one more service round before being serviced. The waste of resources by the greedy algorithm in cases 1 and 2 benefits DM requests in terms of improved response time per DM request. For cases 4–6 in which resources allocated to DM requests are meager compared with the reservation algorithm, we see that the reservation algorithm compares favorably with the greedy algorithm without compromising the response time per DM request performance metric.

Figure 8 breaks down the usage of the system in servicing video, image and text requests, i.e. proportions of the time it services video, image and text requests. Also shown are the same three utilization values obtained by the video-first algorithm. The diagram does not show the greedy algorithm to avoid cluttering since it exhibits the same trend as the reservation algorithm. Here we see that the utilization to serve video requests is high all the time in the video-first algorithm when compared with the reservation algorithm. Correspondingly, in the video-first algorithm the utilizations to serve image and text requests are much lower since in

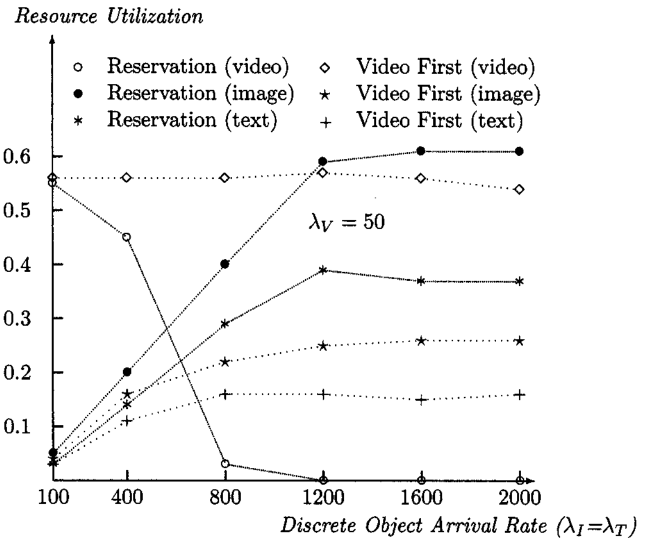

FIGURE 8. Comparing utilization of various object request types.

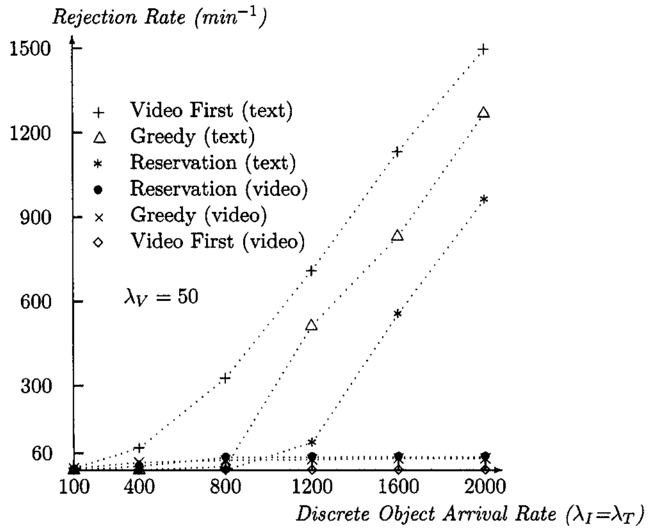

FIGURE 9. Comparing rejection rates of various object request types.

the video-first algorithm, video requests will consume the resources first with the leftover (if any) being consumed by image and text requests. In cases where image/text arrival rates are high, the low utilization to serve image/text requests therefore contributes to the very low reward rate obtained by the video-first algorithm.

Finally, Figure 9 shows the rejection rates (number of requests rejected per second) of video and text requests versus those by the video-first and greedy algorithms. The diagram only shows text requests (representing the DM requests) versus video requests (representing the CM requests) to avoid cluttering as image requests exhibit the same trend as text requests. When the system is heavily loaded, we see that our reservation algorithm significantly rejects fewer text requests compared with both the video-first and greedy algorithms. This is achieved by rejecting more video requests, i.e. the video rejection rate is 50 (per min) for the reservation algorithm compared with the video rejection rate of nearly 0 in the video-first algorithm and 41 in the greedy algorithm in the last case. This is to be compared with, say, the text rejection rate of 960 (per min) in the reservation algorithm versus 1500 in the video-only algorithm and 1260 in the greedy algorithm. The lower rejection rates of image and text requests when the system is heavily loaded contribute to the higher reward rate obtained by the reservation algorithm since fewer penalties are applied due to fewer image/text clients being rejected.

5. CONCLUSION

In this paper, we have proposed, analyzed and validated a priority-based, resource-reservation admission control algorithm to address the issue of how much resource should be allocated to service video/audio (CM) and image/text (DM) requests for handling mixed workloads in modern multimedia systems such as a digital library multimedia server that must provide access services to heterogeneous objects stored in the library. The priority considered in the paper is defined in terms of 'rewards' and 'penalties' associated with requests of various object types. Instead of arbitrarily admitting object requests until resources are exhausted as a condition for admission control, we allocate the resources to requests *a priori* based on this priority-based scheme so as to maximize the total reward received by the system. The algorithm derives from queueing theory and can be applied at run time to dynamically adjust the resource reservations to objects of various types to adapt to environmental changes such as the changing arrival rates of object requests. We compared our reservation algorithm with the video-first and greedy admission control algorithms analytically and via simulation and demonstrated that our algorithm can significantly improve the reward value obtained by the system without sacrificing other performance metrics such as the response time per DM request and system utilization.

There are a few future research areas including (i) investigating other applicable priority schemes not based on 'rewards' and 'penalties'; (ii) investigating if such cost-based admission control algorithms discussed in the paper can yield even higher reward rates in non-conventional disk scheduling systems such as those not based on scheduling cycles (i.e. not servicing requests in rounds).

ACKNOWLEDGEMENT

Naresh Verma was supported by a National Science Foundation IGERT grant #9987586.

REFERENCES

[1] Lawton, G. (2000) Video streams into the mainstream. *IEEE Comp.*, July, 12–17.

[2] Golubchik, L., Lui, J. C. S., Silva, E. and Gail, H. R. (1999) Evaluation of tradeoffs in resource management techniques for multimedia storage servers. In *IEEE Int. Conf. on Multimedia Computing and Systems*, Florence, Italy, June, pp. 292–296.

[3] Romboyannakis, Y., Nerjes, G., Muth, P., Paterakis, M., Triantafillou, P. and Weikum, G. (1998) Disk scheduling for mixed-media workloads in a multimedia server. In *6th ACM Int. Conf. on Multimedia*, Bristol, UK, September, pp. 297–302.

[4] Shenoy, P. J. and Vin, H. M. (1998) Cello: a disk scheduling framework for next generation operating systems. In *7th ACM Int. Conf. on Measurement and Modeling of Computer Systems (SIGMETRICS '98)*, Madison, June, pp. 44–55.

[5] To, T. P. J. and Hamidzadeh, B. (2000) Run-time optimization of heterogeneous media access in a multimedia server. *IEEE Trans. Multimedia*, **2**, 49–61.

[6] Chandra, S., Ellis, C. S. and Vahdat, A. (200) Differentiated multimedia web services using quality aware transcoding. In *IEEE INFOCOM 2000*, pp. 961–969.

[7] Chen, I. R. and Chen, C. M. (1996) Threshold-based admission control policies for multimedia servers. *Comp. J.*, **39**, 757–766.

[8] Lee, W. and Sabata, B. (1999) Admission control and QoS negotiation for soft-real time applications. In *IEEE Int. Conf. on Multimedia Computing and Systems*, Florence, Italy, June, pp. 147–152.

[9] Chang, E. and Zakhor, A. (1996) Cost analyses for VBR servers. *IEEE Multimedia*, **3**, 56–71.

[10] Trivedi, K. S. (1999) *SPNP User's Manual Version 6*. Duke University, USA.

[11] Worthington, B. L., Ganger, G. R. and Patt, Y. N. (1994) Scheduling algorithms for modern disk drives. In *3rd ACM Int. Conf. on Measurement and Modeling of Computer Systems (SIGMETRICS '94)*, May, pp. 241–251.

APPENDIX A. DERIVATION OF $P(J)$ FOR $M/M/1^{[N]}/N$

For notational convenience, we will use P_j to represent the probability that the system is in state j. Also, we drop the subscript from the number of slots n, the arrival rate λ and the group departure rate μ.

First we apply global balance to the $n+1$ states in the $M/M/1^{[n]}/n$ system to obtain the following linear equations:

$$\begin{array}{rcl}
0 &=& \mu(P_1+P_2+\ldots+P_n)-\lambda P_0\\
0 &=& \lambda P_0-(\mu+\lambda)P_1\\
0 &=& \lambda P_1-(\mu+\lambda)P_2\\
\ldots &=& \ldots\\
0 &=& \lambda P_{n-2}-(\mu+\lambda)P_{n-1}\\
0 &=& \lambda P_{n-1}-\mu P_n.
\end{array}$$

This gives the relation that for $0 \le i < n$

$$P_i=\frac{\mu}{\lambda}\left(\frac{\lambda+\mu}{\lambda}\right)^{n-1-i}P_n.$$

Since $\sum_{i=0}^{n} P_i = 1$, the above relation gives us:

$$\frac{\mu}{\lambda}\left[\left(\frac{\lambda+\mu}{\lambda}\right)^{n-1}+\left(\frac{\lambda+\mu}{\lambda}\right)^{n-2}+\ldots\right.$$
$$\left.\ldots+\left(\frac{\lambda+\mu}{\lambda}\right)^{2}+\left(\frac{\lambda+\mu}{\lambda}\right)^{1}+1\right]P_n+P_n=1.$$

Applying the formula that

$$1+x+x^2+\ldots+x^{n-1}=\frac{x^n-1}{x-1}$$

the above relation becomes

$$\left[\left(\frac{\lambda+\mu}{\lambda}\right)^{n}-1\right]P_n+P_n=1.$$

Thus,

$$P_i=\begin{cases}\dfrac{\mu}{\lambda}\left(\dfrac{\lambda}{\lambda+\mu}\right)^{i+1} & \text{if } 0\le i<n-1\\[2ex] \left(\dfrac{\lambda}{\lambda+\mu}\right)^{n} & \text{if } i=n.\end{cases}$$

APPENDIX B. DERIVATION OF $P(J)$ FOR $M/M/1^{[N]}/2N$

For the $M/M/1^{[n]}/2n$ system, there are $2n+1$ states in the system (see Figure 2). Applying global balance to these states, we have:

$$\begin{array}{rcl}
0 &=& \mu(P_1+P_2+\ldots+P_n)-\lambda P_0\\
0 &=& \lambda P_0+\mu P_{n+1}-(\mu+\lambda)P_1\\
0 &=& \lambda P_1+\mu P_{n+2}-(\mu+\lambda)P_2\\
\ldots &=& \ldots\\
0 &=& \lambda P_{n-2}+\mu P_{2n-1}-(\mu+\lambda)P_{n-1}\\
0 &=& \lambda P_{n-1}+\mu P_{2n}-(\mu+\lambda)P_n\\
0 &=& \lambda P_n-(\mu+\lambda)P_{n+1}\\
0 &=& \lambda P_{n+1}-(\mu+\lambda)P_{n+2}\\
\ldots &=& \ldots\\
0 &=& \lambda P_{2n-1}-\mu P_{2n}.
\end{array}$$

By rearranging terms, we obtain the following relations between P_i and P_{2n}, $0 \le i \le 2n$:

$$P_i=\begin{cases}\left[\left(\dfrac{\mu}{\lambda}\right)\left(\dfrac{\lambda+\mu}{\lambda}\right)^{2n-1-i}-\left(\dfrac{\mu}{\lambda}\right)\left(\dfrac{\lambda+\mu}{\lambda}\right)^{n-2-i}\right.\\ \qquad\left.\times\left(\dfrac{\lambda+(n-i)\mu}{\lambda}\right)\right]P_{2n} & \text{if } 0\le i\le n-1\\[2ex] \left(\dfrac{\mu}{\lambda}\right)\left(\dfrac{\lambda+\mu}{\lambda}\right)^{2n-1-i}P_{2n} & \text{if } n\le i\le 2n-1\\[2ex] P_{2n} & \text{if } i=2n.\end{cases}$$

Since $\sum_{i=0}^{2n} P_i = 1$, substituting the expressions for P_i from above, we have:

$$\frac{1}{P_{2n}}=\frac{\mu}{\lambda}\left[1+\ldots+\left(\frac{\lambda+\mu}{\lambda}\right)^{2n-1}\right]-\frac{\mu}{\lambda}\left[1+\frac{\lambda+2\mu}{\lambda}\right.$$
$$+\left(\frac{\lambda+\mu}{\lambda}\right)\left(\frac{\lambda+3\mu}{\lambda}\right)+\ldots+\left(\frac{\lambda+\mu}{\lambda}\right)^{n-2}$$
$$\left.\times\left(\frac{\lambda+n\mu}{\lambda}\right)\right]+1. \qquad \text{(B1)}$$

Let

$$A = 1 + \frac{\lambda + 2\mu}{\lambda} + \left(\frac{\lambda+\mu}{\lambda}\right)\left(\frac{\lambda+3\mu}{\lambda}\right) + \dots$$
$$\dots + \left(\frac{\lambda+\mu}{\lambda}\right)^{n-2}\left(\frac{\lambda+n\mu}{\lambda}\right). \quad \text{(B2)}$$

Then,

$$\frac{\lambda+\mu}{\lambda}A = \frac{\lambda+\mu}{\lambda} + \left(\frac{\lambda+\mu}{\lambda}\right)\frac{\lambda+2\mu}{\lambda} + \left(\frac{\lambda+\mu}{\lambda}\right)^{2}$$
$$\times\left(\frac{\lambda+3\mu}{\lambda}\right) + \dots + \left(\frac{\lambda+\mu}{\lambda}\right)^{n-1}$$
$$\times\left(\frac{\lambda+n\mu}{\lambda}\right). \quad \text{(B3)}$$

Subtracting Equation (B3) from Equation (B2), we have:

$$-\frac{\mu}{\lambda}A = 1 + \frac{\mu}{\lambda}\left[1 + \left(\frac{\lambda+\mu}{\lambda}\right) + \dots + \left(\frac{\lambda+\mu}{\lambda}\right)^{n-2}\right]$$
$$-\left(\frac{\lambda+\mu}{\lambda}\right)^{n-1}\left(\frac{\lambda+n\mu}{\lambda}\right). \quad \text{(B4)}$$

The expression for A hence is given by:

$$A = n\left(\frac{\lambda+\mu}{\lambda}\right)^{n-1}. \quad \text{(B5)}$$

Substituting the expression of A above into Equation (B1), we obtain

$$P_{2n} = \left[\left(\frac{\lambda+\mu}{\lambda}\right)^{2n} - n\left(\frac{\mu}{\lambda}\right)\left(\frac{\lambda+\mu}{\lambda}\right)^{n-1}\right]^{-1}.$$

Hence, the solution for P_i follows:

$$P_i = \begin{cases} \left[\left(\frac{\mu}{\lambda}\right)\left(\frac{\lambda+\mu}{\lambda}\right)^{2n-1-i} - \left(\frac{\mu}{\lambda}\right)\left(\frac{\lambda+\mu}{\lambda}\right)^{n-2-i} \times \left(\frac{\lambda+(n-i)\mu}{\lambda}\right)\right]P_{2n} & \text{if } 0 \le i \le n-1 \\ \left(\frac{\mu}{\lambda}\right)\left(\frac{\lambda+\mu}{\lambda}\right)^{2n-1-i} P_{2n} & \text{if } n \le i \le 2n-1 \\ \left[\left(\frac{\lambda+\mu}{\lambda}\right)^{2n} - n\left(\frac{\mu}{\lambda}\right)\left(\frac{\lambda+\mu}{\lambda}\right)^{n-1}\right]^{-1} & \text{if } i = 2n. \end{cases}$$