Performance Model of Interactive Video-on-Demand Systems

Victor O. K. Li, *Fellow, IEEE*, Wanjiun Liao, Xiaoxin Qiu, and Eric W. M. Wong, *Member, IEEE*

***Abstract*— An interactive video-on-demand (VoD) system allows users to access video services, such as movies, electronic encyclopedia, interactive games, and educational videos from video servers on a broadband network. This paper develops a performance evaluation tool for the system design. In particular, a user activity model is developed to describe the usage of system resources, i.e., network bandwidth and video server usage, by a user as it interacts with the service. In addition, we allow batching of user requests, and the effect of such batching is captured in a batching model. Our proposed queueing model integrates both the user activity and the batching model. This model can be used to determine the requirements of network bandwidth and video server and, hence, the trade-off in communication and storage costs for different system resource configurations.**

I. Introduction

IT is generally believed that interactive video-on-demand (VoD) services which support VCR-like functions will be one of the most demanding residential services to be provided in emerging high-speed networks [2], [5], [10], [17]. A VoD system integrates the entertainment, telecommunication, and computer industries and provides electronic video rental services to geographically dispersed users from remote video servers on a broadband network. Users are no longer restricted to being passive watchers. They are allowed to choose the program contents, to decide the viewing schedule, and to interact with the programs with such operations as pause, jump forward, speed-up, etc. Thus, the video servers must be capable of accommodating numerous concurrent user requests to watch and to interact with different parts of the same video, or different videos.

The generic architecture of a VoD system is shown in Fig. 1. There are four important players, namely, the network provider, the program provider, the service provider, and the user. The user generates requests for service to the service provider, who will obtain the necessary material from the program providers and deliver it to the user on the facilities of the network provider. Thus, the service provider acts as an agent of the user and will be able to access various types of program providers. It is possible that the network, program, and service providers are the same organization, but, in general, they will be distinct. In fact, anyone with marketable materials can offer his services to the user through the service provider.

There are many design issues in VoD systems, such as system architecture [2], [4], [7], [13], [16], [18], media server design [1], [3], [8], [12], [14], [15], and video distribution [5], [11]. The placement of video servers is also an important consideration. Alternatives include centralized video servers [4], [13], [16], centralized video servers with distributed video buffers [2], two- or multitiered hierarchical distributed video servers [5], [11], and fully replicated distributed video servers. A centralized video server system is relatively simple to manage. All requests will be sent to and be served at one site. The distributed server system distributes the requests to many sites, located closer to the users, thus alleviating the congestion in the network and the bottleneck due to the central server. However, managing distributed servers is more complex. One has to decide which video, and how many copies, to maintain at each distributed server. In addition, due to varying rates of requests, the video offerings at each distributed server needs to be changed periodically. Which alternative is preferable highly depends on the trade-off between storage and communication costs, the application needs, the underlying infrastructure, and other factors. In this paper, we propose a performance model which may be used to evaluate the requirements of network bandwidth and video server storage, and hence the trade-off in communication and storage costs, for various placement alternatives. The users are allowed to interact with the programs, and a user activity model is developed to capture the effect of such interactions on the system resource usage. In addition, we allow batching of user requests, and the effect of such batching is captured in a batching model. Our proposed queueing model integrates both the user activity and the batching model. Although we illustrate our model with the two-tiered hierarchical architecture, our model can be used to analyze any architecture. In the next section, we describe the VoD system model. We develop the user activity model in Section III. Section IV contains the queueing model. Section V includes numerical results. An example to illustrate how we can use our performance model to study the trade-off in communication and storage costs is included. We conclude in Section VI.

II. VoD System Model

A user interacts with the VoD system through a VoD manager of the service provider. There will, in general, be many VoD managers, perhaps one in each local switch of

Manuscript received May 31, 1995; revised October 3, 1995. This work was supported in part by the United States Department of Defense Focused Research Initiative Program under Grant F49620-95-1-0531 and the Pacific Bell External Technology Program.

V. O. K. Li, W. Liao, and X. Qiu are with the Communication Sciences Institute, Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089-2565 USA (email: vli@usc.edu).

E. W. M. Wong is with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong.

Publisher Item Identifier S 0733-8716(96)04886-X.

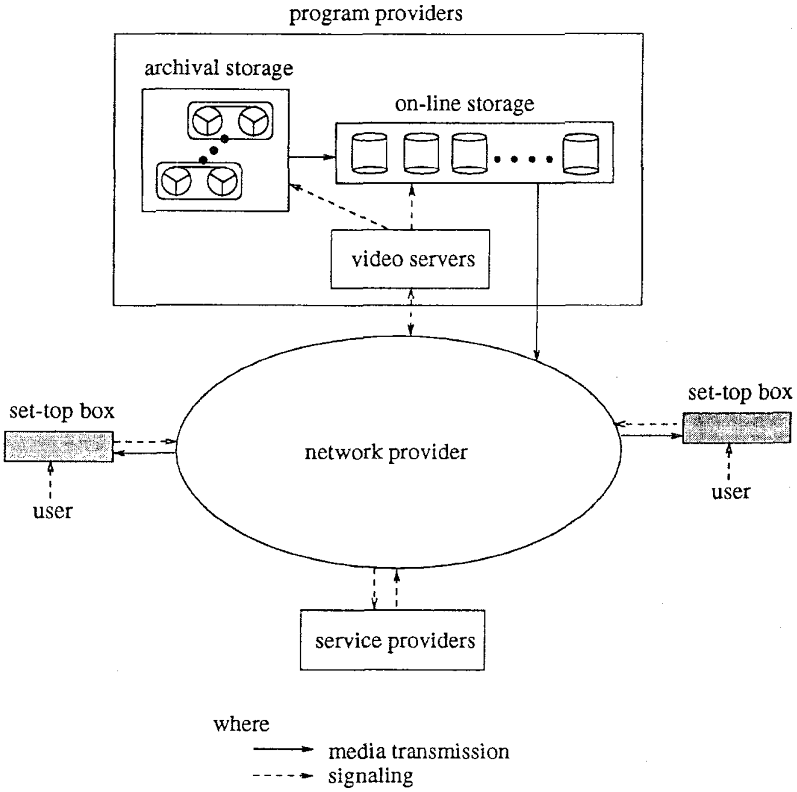


Fig. 1. The generic VoD architecture.

the network. When a request for service is received, the VoD manager will decide whether this request will be accepted or not. For example, if the network is already very congested, or if the video servers cannot take additional requests without degrading the service of existing users, then the user may be blocked, or enqueued in the system. The VoD manager will estimate the backlog, and will advise the user of the anticipated delay. If the request is accepted, then the VoD manager will allocate resources to handle this request. This includes allocating a video buffer for the request,[1] and bandwidth in the network to connect the targeted video server and the video buffer, and the video buffer and the user. In addition, one of the read–write heads of a disk on the targeted server is assigned to this request. Note that if the normal data delivery rate of the video is C bps, the read–write head may be able to operate at a speed higher than C, and thus each read–write head may be able to serve multiple users.

The user has a display such as a TV screen, a set-top box (STB) to control the display, and a device such as a remote control or a keyboard to interact with the system. User interactive operations may include stop, speed-up, slow-down, jump forward, etc. [9]. The user will again interact with the VoD manager to perform such interactions. An interaction may require the read–write head to deliver data to the video buffer at a different rate. These operations have important impacts on the system resources. In the next section, we will quantify these impacts in a user activity model.

To increase the system capacity, it is possible for multiple requests to share the same buffer and read–write head. When an initial request for a program is received, a timer is started. All other requests received for the same program within t_B seconds are processed with this initial request as a batch. Requests which arrive beyond this t_B will not be included in this batch. This t_B depends on how long we are willing to let the customer wait, and should not be more than a few minutes. A read–write head and a video buffer will be assigned for the batch, and all batched users will be fed from the same read–write head and video buffer. When one of the batched users issue an interactive operation, the VoD manager will allocate another read–write head and a separate video buffer for the user, such that its operation will not affect the other users in the batch. If the system is highly loaded, no resources may be available to handle this interactive operation, e.g., no available read head. The user will be so informed, and given a choice of no interaction allowed, or to wait.

III. User Activity Model

Once connected, we assume the user is in one of two states, the normal and the interaction states (see Fig. 2). He starts in the normal state, i.e., the video is being played at the normal speed. He stays in this state for a period of time which is exponential with parameter α. Then he issues an interactive operation, such as stop, speed-up, etc. He stays

[1] In this paper we assume that the video buffer is located at the local switch of the network. This allows multiple users connected to the same local switch to share a video buffer.

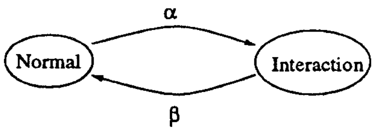


Fig. 2. The user activity model.

in this interaction state for another period of time which is also exponential with parameter β. Then he goes back to the normal state, from where he may again go to the interaction state. This may be repeated multiple times until he disconnects. Different types of interactive operations will affect the system in different ways. Reference [9] lists the following types of possible interactive operations:

1) *Play/Resume*: The start of the presentation from the beginning or the middle.
2) *Stop*: Stopping of the presentation, without picture and sound.
3) *Pause*: Temporarily stopping the presentation, with picture.
4) *Jump Forward*: Jumping to a target time of the presentation in the forward direction, without picture and sound.
5) *Jump Backward*: Jumping to a target time of the presentation in the backward direction, without picture and sound.
6) *Speed Up*: Quickly moving presentation forward, with picture and sound (fast-forward).
7) *Slow Down*: Slowly moving presentation forward, with picture and sound.
8) *Reverse*: Playing a presentation in the reversed direction, with picture and sound.
9) *Fast Reverse*: Quickly moving presentation backward, with picture and sound.
10) *Slow Reverse*: Slowly moving presentation backward, with picture and sound.

We can measure the relative proportion (in terms of duration) of these operations from empirical data. For example, we can observe the system for a long time and measure the duration of time each user is performing a particular operation. Suppose the proportions are as follows: stop/pause, q_1; jump forward, q_2; jump backward, q_3; speed up, q_4; slow down, q_5; reverse, q_6; fast-reverse, q_7; and slow-reverse, q_8, where $\sum_{i=1}^{8} q_i = 1$. Obviously, "stop/pause" will not require data delivery from the video server. In fact, depending on the size of the video buffer, and the size of the jump, even "jump forward" and "jump backward" may not require data delivery either. Basically, if the jump is to a portion of the video already in the video buffer, no data delivery is necessary. If the jump takes you to a portion outside the buffer, then data delivery is necessary. We denote the probabilities that we are within the video buffer in the "jump forward" and "jump backward" operations by p_F and p_B, respectively. Both the "speed up" and the "slow down" operations will require data delivery, but the rates may be different from that of normal playback. Let K_1 be the speed up factor, and K_2 the slow down factor. Speed up may be implemented by retrieving selected frames (say every other frame for a two times speed up) from the video server and delivering only these to the user. In this case, the required data delivery rate $K_1'C$ may actually be the same or even smaller than that for normal playback. Slow down may be implemented by sending data at a reduced rate, resulting in a data delivery rate K_2C. The reverse operations are similar to the play, speed up, and slow down operations. Suppose the fast reverse and slow reverse operations also have data delivery rates of $K_1'C$ and K_2C, respectively.

We can now calculate the average data delivery rate to a user. This is the rate at which data is delivered from the video server to the video buffer, and from the video buffer to the user. The probability a user is in the normal state is $\beta/(\alpha+\beta)$, while the probability of being in the interaction state is $\alpha/(\alpha+\beta)$. While in the interaction state, the user will be performing various operations with the probabilities listed above. Thus, the average data delivery rate R is given by

$$\begin{aligned} R = {} & \left[\frac{\alpha}{\alpha+\beta}(q_1 + q_2 p_F + q_3 p_B)\right] \times 0 \\ & + \left[\frac{\beta}{\alpha+\beta} + \frac{\alpha}{\alpha+\beta}\right. \\ & \qquad \left. \times\, (q_2(1-p_F) + q_3(1-p_B) + q_6)\right] \times C \\ & + \left[\frac{\alpha}{\alpha+\beta}(q_4 + q_7)\right] \times K_1'C \\ & + \left[\frac{\alpha}{\alpha+\beta}(q_5 + q_8)\right] \times K_2C. \end{aligned} \tag{1}$$

We can also calculate the average connection time T of a user, defined as the time from when he is connected to a video server to when he disconnects. Suppose the normal playback time, without user interaction, of a video program is T' (for example, 90 min for a movie), then T may be longer or shorter than T' depending on the user interactions. For example, if we stop the video for t_1 time units, T will be increased by t_1. Jumping forward by t_2 and jumping backward by t_3 will decrease and increase T by t_2 and t_3, respectively. Speeding up for t_4 time units by a factor of K_1 will decrease T by $t_4(K_1-1)$, while slowing down for t_5 time units by a factor of K_2 will increase T by $t_5(1-K_2)$. Reversing for t_6 time units will increase T by $2t_6$ since in this case, the reversing itself takes t_6 time units, and then one ends up at a point of the video which is t_6 time units before the reverse operation. Fast reversing for t_7 time units will increase T by $t_7(K_1+1)$ while slow reversing for t_8 time units will increase T by $t_8(K_2+1)$. Therefore

$$\begin{aligned} T = {} & T' + q_1E(t_1) - q_2E(t_2) + q_3E(t_3) \\ & - q_4E(t_4)(K_1-1) + q_5E(t_5)(1-K_2) + 2q_6E(t_6) \\ & + q_7E(t_7)(K_1+1) + q_8E(t_8)(K_2+1) \end{aligned} \tag{2}$$

where $E(t_i)$, $i = 1, \cdots, 8$, may be found empirically.

IV. Queueing Model

In this section, we will develop a queueing model for the interactive VoD system. While there are many important

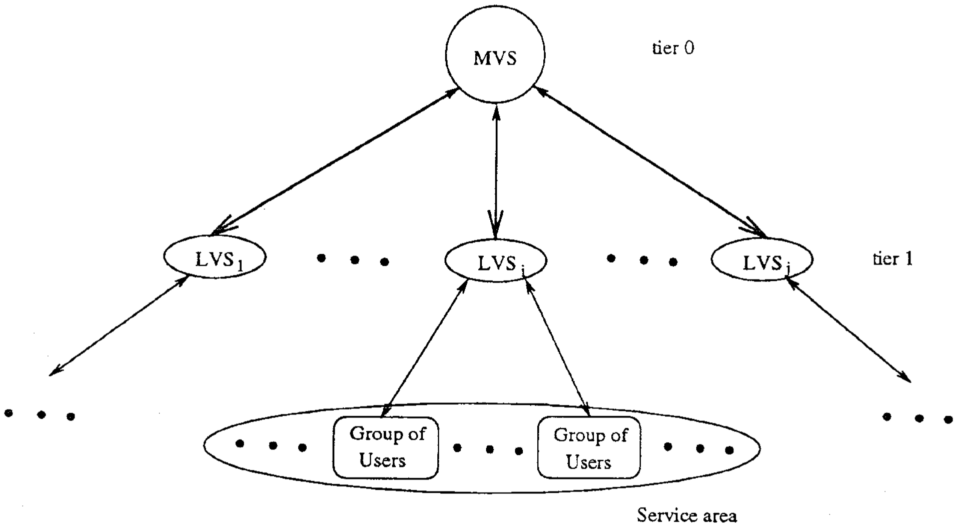


Fig. 3. Logical architecture of server placement.

performance measures, such as the end-to-end blocking probability, the end-to-end service response delay, the amount of video buffers required, the capacity of video servers required, the network capacity required, etc., in this paper we will focus on the first two. One limitation of the queueing model is that it only gives average results. Thus, the blocking probability calculated will be typical of what will be experienced by the majority of the users. It is possible that some users experience better or worse performance. Worse performance will occur when many users are issuing interactive requests simultaneously. However, we believe such cases are rare. In fact, the averaging effect due to the large number of users will guarantee that the results will be an accurate reflection of reality most of the time.

From the system point of view, video servers and their connections in an interactive VoD system forms a queueing network. Each video server can be considered as a serving node with a queue. The queue capacity can either be finite or infinite, depending on the system design.

Let us illustrate our proposed queueing model with the two-tiered architecture. In this architecture, there are a couple of metropolitan video servers (MVS at tier 0), each connected to a number of local video servers (LVS at tier 1). Each LVS serves a particular service area and an MVS serves a group of these areas. In the LVS, most likely located in the local switches of the network provider, the popular video programs are replicated and stored in on-line mass storage systems, such as redundant arrays of inexpensive disks (RAID) [6]. At the MVS, perhaps located at the local access and transport area (LATA) switch of the network provider, the more popular programs are again stored in arrays of disk drives, but the less popular material will be archived in high-capacity optical disks or magnetic tapes. When such less popular videos are requested, they will be loaded into the on-line storage in the metropolitan servers, and then transmitted at high rates to video buffers at the local switches. It is assumed that all the LVS's (MVS's) are identical. Hence, we can focus on only one of them. The same methodology adopted in the following can also be used to study the heterogeneous case in which the LVS's (MVS's) may have different capacities and access delays, etc., but with more computational complexity. Fig. 3 shows the logical connection of one MVS and its group of LVS's. Considering the information flows among different video servers, the queueing model of a two-tiered interactive VoD system is illustrated in Fig. 4.

It is assumed that a certain number of popular videos are placed in each LVS. Denote the service request arrival rate to each LVS from its service area as λ_1. Since the number of users in each service area is large and these users generate the service requests independently, the arrival process to each LVS can be modeled as a Poisson process with rate λ_1. These requests are generally served by the LVS. Only when the LVS is too congested will the blocked requests go up one tier to the MVS. MVS has all the videos which will be potentially requested by users from l_0 service areas, including both popular and unpopular ones. It serves the requests for unpopular videos and the blocked requests for popular videos at LVS's. The service requests for unpopular videos are forwarded directly to the MVS by the VoD manager. The request arrival process to MVS obviously can be modeled as another Poisson process since it is a merging of the arrivals from a large number of service areas. Let λ_2 denote the service request arrival rate for the unpopular videos from each service area. Then, the service request arrival rate to MVS is a function of λ_1 and λ_2, as shown in Fig. 4. Note that the request rate for popular videos is generally much higher than that for unpopular ones. The ratio of the request rate for unpopular to popular videos is denoted as $p = \lambda_2/\lambda_1$, which can be estimated from empirical data. This ratio is important in the system design. For example, the videos must be properly distributed between LVS's and MVS, in order to balance the communication cost and the storage cost. If fewer movies are stored in an LVS, a large number of service requests has to be served by the MVS directly, i.e., p is high. This will cause severe congestion in the MVS and

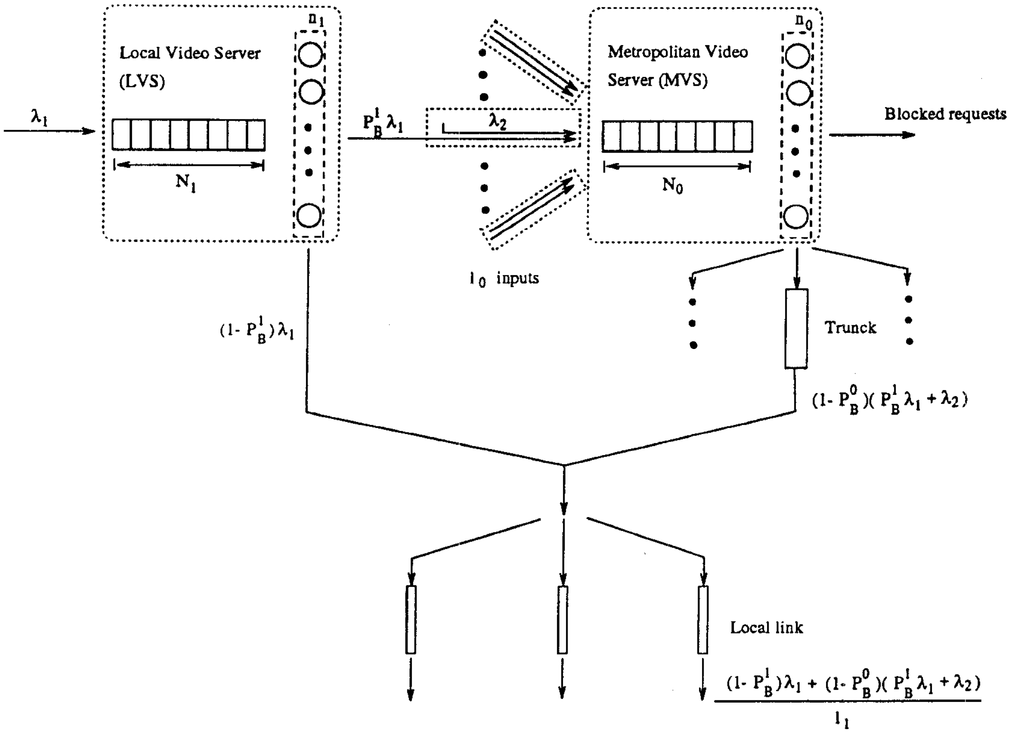


Fig. 4. Queueing model of an interactive VoD system. λ_1 denotes the request arrival rate for popular videos, λ_2 denotes request arrival rate for nonpopular videos, l_0 denotes number of LVS's per MVS, l_1 denotes number of user groups per LVS, P_B^0 denotes blocking probability at MVS, P_B^1 denotes blocking probability at LVS, N_0 denotes queue capacity of MVS, and N_1 denotes queue capacity of LVS.

heavy traffic load in the network. On the contrary, the more videos placed in the LVS, the smaller the ratio p, the less congested the MVS, the less traffic in the network, but the more expensive the LVS.

In the interactive VoD system, each video server consists of an array of disks; each disk has a certain number of read–write heads; and each head can serve more than one user since its I/O transfer rate is generally much higher than the required delivery rate of one user. Therefore, each video server at tier i can support a maximum of n_i users simultaneously, where n_i is called the capacity of the server. This means that a real video server can be considered as n_i virtual ones, each of which can be held by a user during his connection. We use this virtual server mechanism to account for the multiplexing of the read–write head among multiple users. The value of n_i can be calculated as follows. Denote the number of disks in a server at tier i as d_i, the number of heads activated simultaneously per disk as h_i, the I/O rate of each head as I_i, and the required average delivery rate of each user as R. Ideally, n_i is equal to $d_i h_i I_i / R$. However, due the head movement latency, imperfect scheduling, unbalanced loading, and other impairments, this capacity must be reduced by a factor $c < 1$. The server capacity at tier i is thus

$$n_i = c d_i h_i \frac{I_i}{R}. \tag{3}$$

If we assume that the service time of each virtual server is exponentially distributed with mean T (considering the effect of user interaction and the varying lengths of different video programs), each video server at tier i can be modeled as an $M/M/n_i/(n_i + N_i)$ queue, where N_i is the queue capacity. Define the service request blocking probability at tier i as P_B^i. With the $M/M/n_i/(n_i + N_i)$ queueing model developed in Appendix B, these blocking probabilities can be evaluated if the arrival rates are known. The state of the queueing model is the number of video requests. In the case of batched service, it is also the number of batches in the system, since all requests served in the same batch is considered one request.

Since the request arrival rate to an LVS at tier 1 is λ_1, and the average service time of a virtual server is T, the request blocking probability of popular videos at this tier is

$$P_B^1 = p_{n_1+N_1} \tag{4}$$

where $p_{n_1+N_1}$ is the system stationary probability at state $n_1 + N_1$, given by (14). The service response delay at this LVS, D_1, can be evaluated by (16).

The service requests arriving at MVS consist of two parts. One corresponds to requests for unpopular videos, and the other to the blocked requests at LVS's for popular ones. Following the information flows in Fig. 4, the total arrival rate to MVS, λ_0, is

$$\lambda_0 = l_0\left(\lambda_2 + P_B^1 \lambda_1\right) \tag{5}$$

where l_0 is the number of LVS's connected to one MVS. With the average service time of each virtual server T,[2] the request blocking probability at MVS (tier 0), P_B^0, is

$$P_B^0 = p_{n_0+N_0} \tag{6}$$

[2] Here we ignore the additional loading time from archival to on-line storage for unpopular movies.

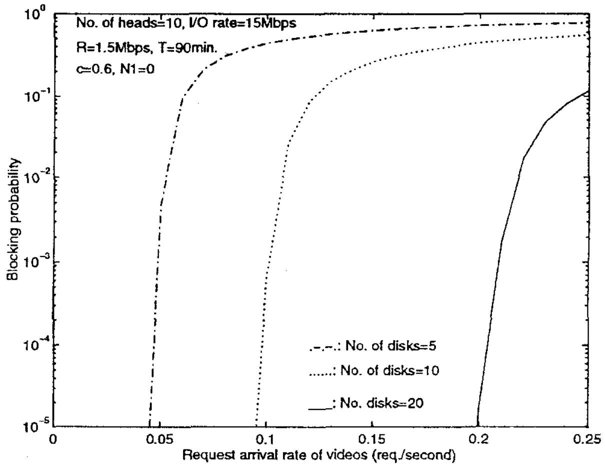


Fig. 5. Impact of the number of disks at a server on the blocking probability.

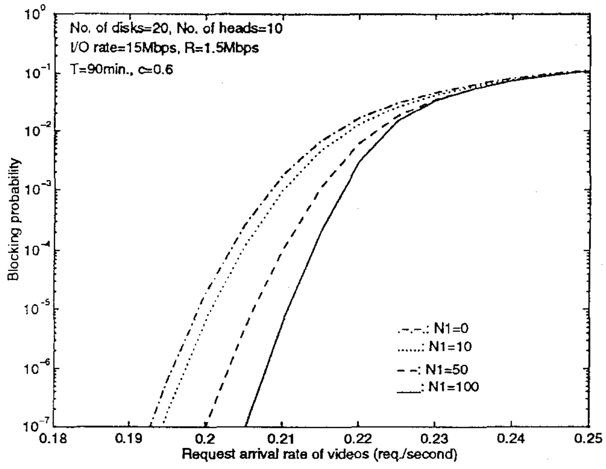


Fig. 6. Impact of queue capacity on the blocking probability at a server.

where $p_{n_0+N_0}$ is the system stationary probability at state n_0+N_0, given by (14). As before, the service response delay, D_0, can also be evaluated by (16).

Therefore, the end-to-end request blocking probabilities for popular and unpopular videos, P_p and P_{np}, are

$$P_p = P_B^1 P_B^0 \quad \text{and} \quad P_{np} = P_B^0 \tag{7}$$

respectively. The end-to-end service response delay of these two kinds of requests, D_p (for popular ones) and D_{np} (for unpopular ones), are

$$D_p = \frac{1-P_B^1}{N_{om}} D_1 + \frac{P_B^1(1-P_B^0)}{N_{om}} D_0 \quad \text{and} \quad D_{np} = D_0 \tag{8}$$

where N_{om} is a normalization constant, $N_{om} = 1 - P_B^1 P_B^0$.

To further increase the system capacity, batched user service, as described in Section II, can be employed. Given a batch interval of t_B seconds, the average effective batch size at a tier i server, B_{eff}^i, is derived in Appendix A, considering the user behavior. Since each batch is served as one video request, the impact of this batched service on the system is to reduce the effective request arrival rate to the video server at tier i by a factor of B_{eff}^i. This reduction in the arrival rate can be translated directly into smaller blocking probabilities or higher system capacities. The price paid is the longer initial service delay (t_B). In our opinion, a certain amount of initial delay is tolerable to the user because of the relatively long video length. Employing the batched service, we can trade the system capacity with the initial service delay.

The above model allows one to calculate the blocking probability for different arrival rates and different video server placements. Once we have the blocking probability, we can calculate the network bandwidth required between the MVS and the video buffers at the local switch, and between the local switch and the users. Appendix C contains the distribution of the required bandwidth, and expressions for the average and the x-percentile bandwidths. It also contains the distribution of the size of the required video buffer, and expressions for the average and the x-percentile buffer size.

V. Numerical Results

To get the numerical results, we assume that $h_0 = h_1 = 10$, $I_1 = I_0$ =15 Mbps, $l_0 = 50$, and $c = 0.6$. With user activity, the average connection time $T = 90$ min. Unless specified otherwise, the user data delivery rate is $R = 1.5$ Mbps.

First, we investigate the behavior of one video server. The impact of the number of disks in a server on the blocking probability is illustrated in Fig. 5. No queueing is assumed here. Given a blocking probability, the required number of disks can be estimated from this figure if the request arrival rate is given. In addition, it is observed that the blocking probability increases sharply as the request arrival rate approaches the capacity.

The impact of the queue capacity on the blocking probability and the service response delay is shown in Figs. 6 and 7, respectively. When the arrival rate is small, the queue is useful to reduce the blocking probability without introducing significant queueing delay. However, when the service request rate increases, the larger queue size is almost useless to combat the blocking even though it causes longer service response delay (Fig. 7). This is because the queue is only useful to relieve temporary congestion due to traffic fluctuations. When the system is operating at or above its capacity, we have permanent congestion, and the requests which correspond to the workload above the system capacity will always be blocked, irrespective of the size of the queue. Fig. 7 also shows how the average service response delay (queueing delay) changes as the request arrival rate increases. For example, for a blocking probability of less than 1%, Fig. 6 says that the system can accommodate a request arrival rate of 0.223 with a queue capacity $N_1 = 100$. However, Fig. 7 tells us that the corresponding delay is 250 s. Some of the users may not tolerate such a delay and renege from the system.

In interactive VoD, the user's behavior has a great influence on the system performance through varying required data delivery rate R. R can be evaluated with the user activity model developed in Section III. Its impact on the blocking probability is illustrated in Fig. 8. Here we assume that there are 20 disks and no queue at the server. It can be seen that

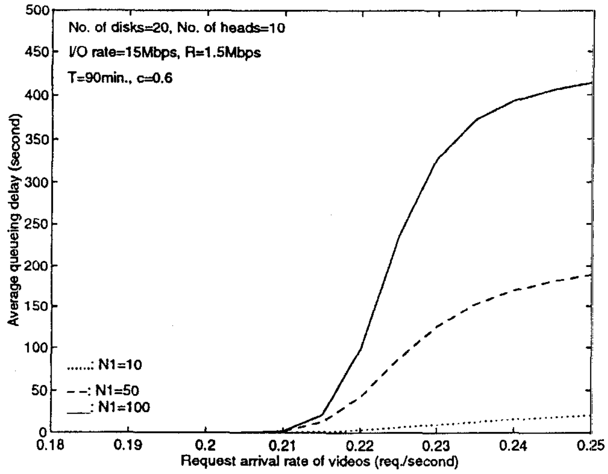


Fig. 7. Impact of queue capacity on the service delay at a server.

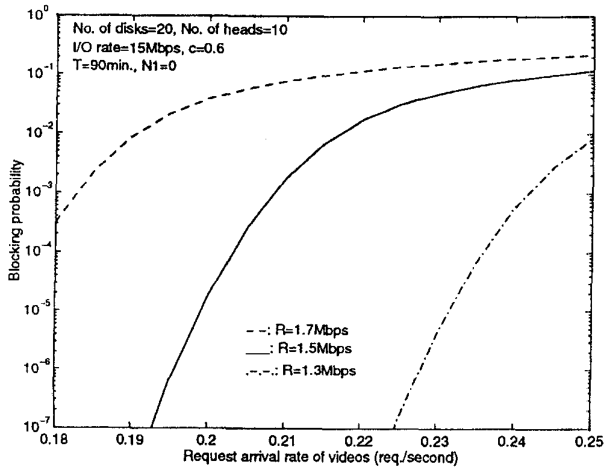


Fig. 8. Impact of the required user data delivery rate on the blocking probability at a server.

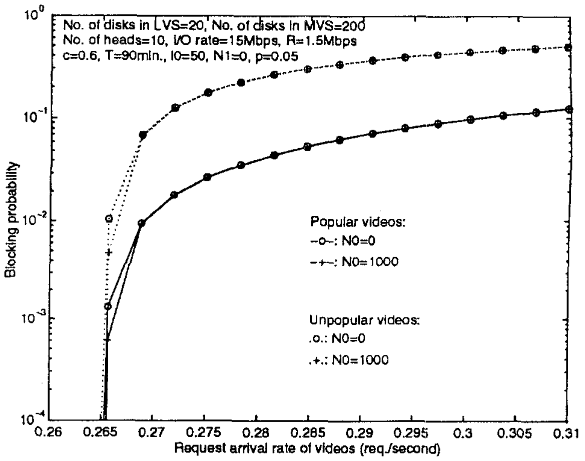


Fig. 9. Impact of queue capacity of MVS on the request blocking probability for videos.

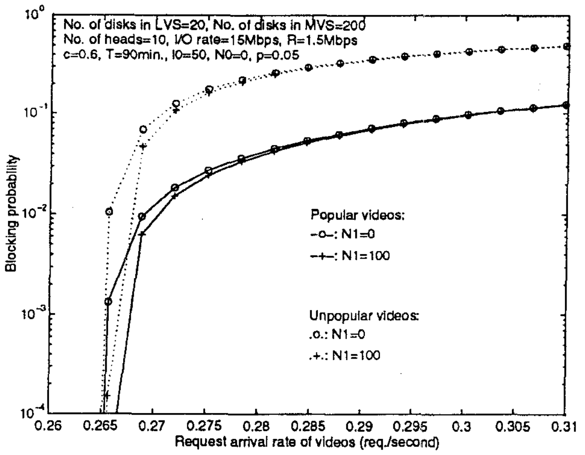


Fig. 10. Impact of queue capacity of LVS on the request blocking probability for videos.

if the user requires smaller average delivery rate (e.g., if he issues frequent stop or slow-down operations), the blocking probability is greatly reduced. On the other hand, the blocking probability is much higher when the user requires higher average delivery rate (e.g., if he frequently speeds up).

Next, we will investigate the performance of the two-tiered system. We assume that there are twenty disks at each LVS and two hundred at an MVS. In the following figures, the blocking probability will be shown as a function of the total request arrival rate from each service area, $\lambda_1 + \lambda_2$, for both popular and unpopular videos. The ratio between λ_2 and λ_1 is $p = 0.05$.

The impact of queue capacity of MVS (N_0) on the request blocking probabilities for popular and unpopular videos is shown in Fig. 9. The queue capacity of MVS varies from zero to 1000 and it is assumed that there is no queue at LVS. It is found that even a queue capacity of 1000 has almost no influence on the blocking probability.

The impact of queue capacity of LVS (N_1) on the request blocking probabilities for popular and unpopular videos is shown in Fig. 10. In this figure, we assume that there is no queue at MVS. The queue capacity of LVS varies from zero to 100. It is found that the system still cannot benefit much from the queue. The reason is the same as before. When the system operates at or above system capacity, a queue will not help.

In our design, we allow the blocked requests at LVS to be forwarded to MVS in order to reduce the request blocking probability for popular videos. We call this strategy "Option 2." However, these overflow requests may introduce more traffic at MVS and cause higher request blocking probability for unpopular videos.[3] The alternative is not allowing overflow traffic from LVS's to MVS. If the request for popular videos is blocked at an LVS, it will be discarded, and the MVS only serves the requests for unpopular videos. We call this "Option 1." The overall request blocking probability can be defined as a weighted sum of the blocking probabilities for popular videos and for unpopular videos. The weights for these two kinds of videos are $\lambda_1/(\lambda_1 + \lambda_2)$ and $\lambda_2/(\lambda_1 + \lambda_2)$, respectively.

[3] Here, we assume that the requests for unpopular and popular videos have the same priority.

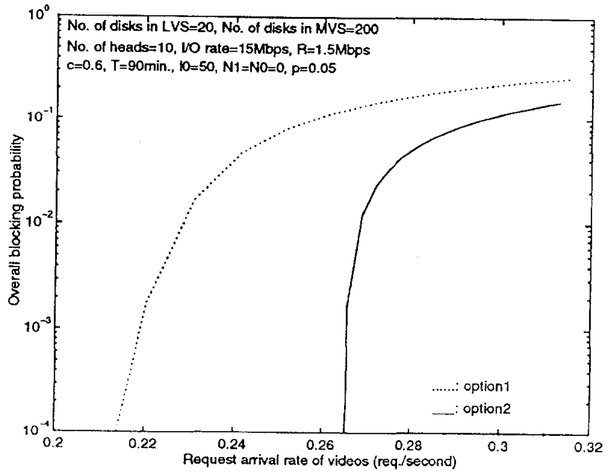


Fig. 11. Overall blocking probabilities of two design alternatives.

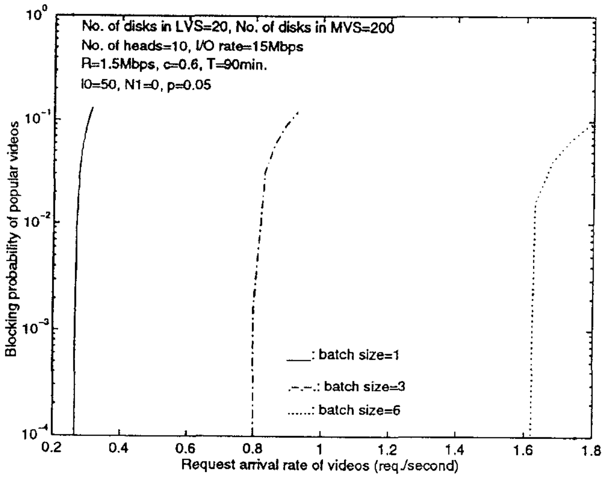


Fig. 12. Impact of batch size on the request blocking probabilities for popular videos.

We compare the overall request blocking probabilities for the above two design options in Fig. 11. It can be observed that the overall system performance of "Option 2" is always better than that of "Option 1."

To investigate the impact of batched service on the system performance, we plot the request blocking probability as a function of the batch size in Figs. 12 and 13 for the popular and unpopular videos, respectively. Note that, in these figures, we assume the same batch size for both MVS and LVS's. In general, these batch sizes are different. Appendix A contains the calculations of the average effective batch size for popular and unpopular videos. It can be seen that if we batch the service, the request arrival rate which can be supported under certain blocking probability requirement is significantly improved. With batched service, the price paid is the initial delay of service response (t_B, the batch interval). The user can still enjoy the freedom of interacting with the program.[4]

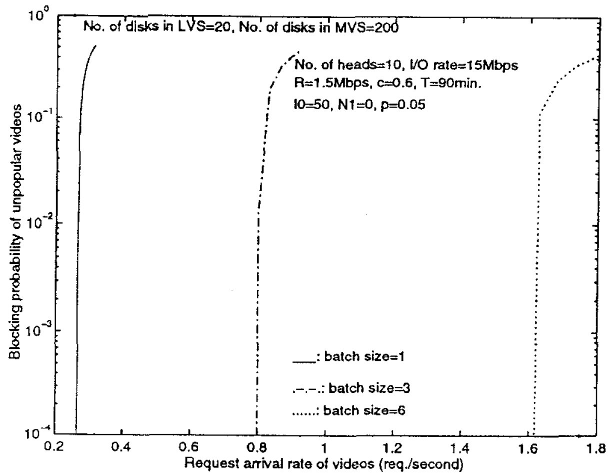


Fig. 13. Impact of batch size on the request blocking probabilities for unpopular videos.

We next illustrate how one can use our performance model to study the trade-off between storage and communication costs. Suppose we assume that the total request arrival rate per service area is $\lambda_1 + \lambda_2 = 0.1$ requests/second and the ratio p of the rate for unpopular to that for popular videos is 0.05. The service requirement is that the blocking probability for both popular and unpopular videos should be less than 1%. Three different server placement methods in the two-tiered system are considered, with the number of disks in each LVS being 0, 5, and 10, respectively. The corresponding required capacity of MVS is evaluated using the queueing model in Section IV. Given the server placement and the service requirement, the required trunk[5] bandwidth and the required local link[6] bandwidth are obtained using the method developed in Appendix C. These results are listed in Table I. Note that the number of disks at an LVS being zero corresponds to a centralized architecture. We assume that the cost of one unit of bandwidth is unity. One unit of bandwidth is the data delivery rate for one connection, which is R. Let r be the ratio of the cost of one unit of storage to the cost of one unit of bandwidth. One unit of storage is the disk space required to store one video, about one GBytes for a compressed movie. Assuming each disk contains sixty units of storage, the total cost of each system scenario is shown as a function of r in Fig. 14, according to Table I. The impact of different server placements on the total system cost can be observed clearly from this figure. When r is small, the network cost (bandwidth cost) is dominant. Then, the distributed architecture has great advantage since most of the requests are served locally. For different values of r, a centralized architecture may be better.

VI. Conclusion

In this paper, we have developed a performance model for an interactive VoD system. A user activity model is developed to study the impact of user behavior on the system design. The

[4] The user will leave the batched group and be assigned a new read–write head and a new video buffer when an interaction occurs.

[5] Trunk refers to the link between the MVS and the local switch.

[6] Local link refers to the link between the local switch and one group of users. To get the numerical results, we assume that the users in one service area is divided into 100 user groups.

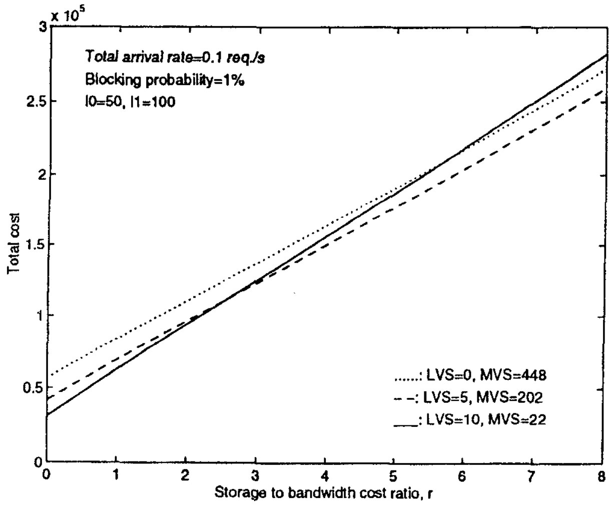


Fig. 14. Cost comparison of three alternative server placements.

TABLE I
An Example of Server Placement and Required Network Bandwidth

No. of disks at an LVS	No. of disks at an MVS	Average bandwidth of a trunk (unit: R Mbps)	Average bandwidth of a local link (unit: R Mbps)
0	448	536	6
5	202	241	6
10	22	26	6

importance of this model is verified by the numerical results, which show that the user behavior has great influence on the system performance. With different required user data delivery rates, the blocking probability varies significantly for the same service request arrival rate. A batched service which allows full user interactivity is also described. We show that with batched service, the system capacity can be greatly improved at the expense of some initial service response delay. Due to the long video viewing time, this initial delay may be tolerable to the user. A batched service model is developed.

Our performance model integrates our user activity model and our batched service model. Although we use a two-tiered architecture to illustrate our model, the methodology employed here can be used to evaluate the performance of any architecture. This model can be used to determine the requirements of network bandwidth and video server, and hence the tradeoff in communication and storage costs for different system resource configurations.

Appendix A
The Average Batch Size

In order to increase the capacity, we hold an incoming request for a video for up to t_B seconds, waiting for the arrival of other requests for the same video, and serving them as a batch. Suppose at most n_B requests may be served simultaneously. We focus on one particular video server at tier i. Let there be n_p^i popular and n_u^i unpopular videos at this server. Without loss of generality, label the popular videos $1, 2, \cdots, n_p^i$, and the unpopular ones, $n_p^i+1, n_p^i+2, \cdots, n_p^i+n_u^i$. Assume requests for each video at this tier is Poisson with rate $\gamma_k^i, k = 1, \cdots, n_p^i + n_u^i$. First we find $E(B_k^i)$, where B_k^i is the batch size for video k at tier i. We wait until the first request arrive for video k. Let L be the number of requests for video k which arrive within the next t_B. If $L > n_B - 1$, the batch size is n_B; otherwise, the batch size is $L+1$. Therefore

$$E(B_k^i) = n_B \sum_{l=n_B-1}^{\infty} p_L(l) + \sum_{l=0}^{n_B-2} (l+1)p_L(l)$$

where $p_L(l) = \frac{(\gamma_k^i t_B)^l e^{-\gamma_k^i t_B}}{l!}, \ l = 0, 1, 2, \cdots$.

This only considers that each user in the batch will not initiate any interactive operation. In our design, we allow, however, full user interactions and each user which interacts with the video will be allocated its own read–write head and video buffer. Recall that according to the user activity model, each user starts in the normal state, and will stay there for a period t_N which is exponential with rate α. Then he goes into the interactive state, and will go back to the normal state again after an exponential period. Once he has entered the interactive state, he will be assigned his own read–write head and buffer, irrespective of whether he switches back to the normal state or not. We need to find the length of time a user will stay in the batch during his connection. He will leave the batch when his connection terminates, or when he switches to the interactive mode. Since we assume the connection time t_C is exponential with mean T, and a user will stay in the normal state for an exponential period of time with mean $1/\alpha$, the period of time he stays in the batch is exponential with rate $\alpha + 1/T$. (The minimum of two exponentials is still exponential with rate equal to the sum of the individual rates.)

Thus, although the total connection time is t_C, for a fraction of the time, the user will not be batched with the other users, and will be using its own buffer. Hence, the expected additional buffer due to one user B_{add} is $[E(t_C) - 1/(\alpha + 1/T)]/E(t_C) = [T - 1/(\alpha + 1/T)]/T = \alpha T/(\alpha T + 1)$. Hence, the average effective batch size for video k at tier i, $B_{\text{eff},k}^i$, is equal to the expected number of users served as a batch, divided by the expected number of buffers used, and is given by

$$B_{\text{eff},k}^i = \frac{E(B_k^i)}{1 + B_{\text{add}} E(B_k^i)}.$$

Assume that the total arrival rates for popular and unpopular videos at tier i are λ_p^i, and λ_u^i, respectively. These arrival rates can be obtained according to the specific system architecture and service discipline. For instance, in our two-tiered system, for the server at tier 1 (LVS), $\lambda_p^1 = \lambda_1$, $\lambda_u^1 = 0$; for the server at tier 0 (MVS), $\lambda_p^0 = l_0 P_B^1 \lambda_1$, $\lambda_u^0 = l_0 \lambda_2$. Finally, we can find the average batch size at tier i for popular and unpopular videos as follows

$$B_{\text{eff}}^i(\text{popular}) = \frac{\sum_{k=1}^{n_p^i} \gamma_k^i B_{\text{eff},k}^i}{\lambda_p^i}$$

$$B_{\text{eff}}^i(\text{unpopular}) = \frac{\sum_{k=n_p^i+1}^{n_p^i+n_u^i} \gamma_k^i B_{\text{eff},k}^i}{\lambda_u^i}$$

where $\lambda_p^i = \sum_{k=1}^{n_p^i} \gamma_k^i$, and $\lambda_u^i = \sum_{k=n_p^i+1}^{n_p^i+n_u^i} \gamma_k^i$.

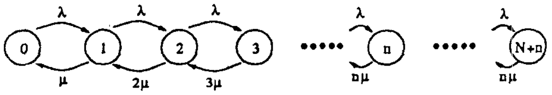


Fig. 15. The Markov chain of an $M/M/n/N+n$ system.

Appendix B
The $M/M/n/N+n$ Queueing Model

We assume there is an infinite population of customers, with total arrival rate λ. Moreover, the system has n servers, each with service rate μ, and N buffers for queueing (excluding those in service). Therefore, we can obtain the following set of birth–death coefficients

$$\lambda_k = \begin{cases} \lambda, & 0 \le k \le n+N-1 \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

$$\mu_k = \begin{cases} k\mu, & 1 \le k \le n \\ n\mu, & n+1 \le k \le n+N. \end{cases} \tag{10}$$

Fig. 15 shows the state-transition-rate diagram for an $M/M/n/n+N$ queue. We can obtain the state probability p_k in state k

$$p_k = \frac{\lambda}{\mu_k} p_{k-1} \quad 1 \le k \le n+N. \tag{11}$$

By recursively using the above set of equations, we have

$$p_k = \begin{cases} \frac{\rho^k}{k!} p_0, & 1 \le k \le n \\ \frac{\rho^k}{n! n^{k-n}} p_0, & n+1 \le k \le n+N \end{cases} \tag{12}$$

where $\rho = \frac{\lambda}{\mu}$. Together with the normalization equation, we can solve for p_0 as follows:

$$p_0 = \left[\sum_{k=0}^{n} \frac{\rho^k}{k!} + \sum_{k=n+1}^{n+N} \frac{\rho^k}{n!\, n^{k-n}} \right]^{-1}. \tag{13}$$

With this stationary distribution of system states, we can find the blocking probability p_{n+N}, as follows

$$p_{n+N} = \frac{\rho^{n+N}}{n! n^N} p_0. \tag{14}$$

The average queue length is

$$\bar{N} = \sum_{k=n+1}^{n+N} (k-n) p_k. \tag{15}$$

By Little's formula, the queueing delay, D, can be evaluated as

$$D = \frac{\bar{N}}{\lambda(1-p_{n+N})}. \tag{16}$$

Appendix C
Distribution of Required Bandwidth and Buffering

The distribution of the bandwidth required, in terms of units of bandwidth R (the average data delivery rate for one connection), is given by the distribution of the number of busy servers, $\{p_k\}$, given in Appendix B.

The average bandwidth required is thus $S = \sum_{k=0}^{n} kp_k + \sum_{k=n+1}^{n+N} np_k$, in units of R, where n is the number of virtual servers, and N is the queue capacity.

Sometimes, we are not only interested in the average bandwidth required, but also the x-percentile bandwidth, defined as the bandwidth required to accept $x\%$ of the requests. Note that with an x-percentile bandwidth, the blocking probability is $1 - x/100$. Thus the blocking probability of a 99-percentile bandwidth is 0.01. Let the x-percentile bandwidth be S_x units of bandwidth, where one unit is R. S_x can be found by solving the following inequality

$$\frac{\sum_{k=0}^{S_x} kp_k + \sum_{k=S_x+1}^{n+N} S_x p_k}{S} \ge x \ge \frac{\sum_{k=0}^{S_x-1} kp_k + \sum_{k=S_x}^{n+N} (S_x - 1) p_k}{S}. \tag{17}$$

The first term of the numerator on the left-hand side corresponds to those cases where requests arrive to find enough bandwidth (virtual servers), and are therefore accepted. The expected number of accepted requests is thus $k \times p_k$. The second term corresponds to those cases where requests arrive to find that there are insufficient bandwidth (virtual servers). When the system is in state k, and there are only S_x virtual servers, where $S_x < k$, the system can only serve S_x of them. The expected number of accepted requests is thus $S_x \times p_k$. The denominator is the normalization factor.

Similarly, the video buffering required, in terms of units of buffer V (V is the buffer size assigned to each user, or in the case of batched service, assigned to each batch) can be found as follows below.[7]

The average buffer size required is $U = \sum_{k=0}^{n} kp_k + \sum_{k=n+1}^{n+N} np_k$, in units of V.

Let the x-percentile buffer size be U_x units of buffer, where one unit is V. U_x can be found by solving the following inequality

$$\frac{\sum_{k=0}^{U_x} kp_k + \sum_{k=U_x+1}^{n+N} U_x p_k}{U} \ge x \ge \frac{\sum_{k=0}^{U_x-1} kp_k + \sum_{k=U_x}^{n+N} (U_x - 1) p_k}{U}. \tag{18}$$

References

[1] D. Anderson, Y. Osawa, and R. Govindan, "A file system for continuous media," *ACM Trans. Computer Syst.*, vol. 8, no. 4, Nov. 1992.

[2] D. Deloddere, W. Verbiest, and H. Verhille, "Interactive video on demand," *IEEE Commun. Mag.*, pp. 155–162, May 1994.

[3] Y. N. Doganata and A. N. Tantawi, "Making a cost-effective video server," *IEEE Multimedia*, pp. 22–30, Winter 1994.

[4] A. D. Gelman *et al.*, "A store-and-forward architecture for video-on demand sevice," in *ICC'91*, 1991, pp. 842–846.

[5] C. Papadimitrious *et al.*, "Multimedia information caching for personalized video-on-demand," *Computer Commun.*, vol. 18, no. 3, pp. 204–216, Mar. 1995.

[6] D. A. Paterson *et al.*, "Introduction to redundant arrays of inexpensive disks (RAID)," in *IEEE COMPCON'89*, 1989.

[7] Y. H. Chang *et al.*, "An open-systems approach to video on demand," *IEEE Commun. Mag.*, pp. 68–80, May 1994.

[8] D. R. Kenchammana-Hosekote and J. Srivastava, "Scheduling continuous media in a video-on-demand server," in *Proc. Int. Conf. Multimedia Computing Syst.*, 1994, pp. 19–28.

[9] W. J. Liao and V. O. K. Li, "Synchronization of distributed multimedia systems with user interactions," Communication Sciences Institute, University of Southern California, Report CSI-95-05-01, May 1995, submitted for publication.

[7] For example, if each batch is assigned a buffer equivalent to 10 minutes' worth of video, V will be approximately 100 Mbytes.

[10] T. D. C. Little and D. Venkatesh, "Prospects for interactive video-on-demand," *IEEE Multimedia*, pp. 14–24, Fall 1994.

[11] ——, "Popularity-based assignment of movies to storage devices in a video-on-demand system," *ACM/Spring-Verlag Multimedia Syst.*, pp. 280–287, 1995.

[12] P. Lougher and D. Shepherd, "Design and implementation of a continuous media storage server," in *Proc. 3rd Int. Workshop Network Oper. Syst.*, 1992.

[13] G. H. Petit, D. Deloddere, and W. Verbiest, "Bandwidth resource optimization in video-on-demand network architectures," in *Proc. 1st Int. Workshop Community Networking Integr. Multimedia Services to the Home*, 1994, pp. 91–97.

[14] P. V. Rangan, H. M. Vin, and S. Ramanathan, "Designing an on-demand multimedia service," *IEEE Commun. Mag.*, pp. 56–64, July 1992.

[15] V. P. Rangan and H. M. Vin, "Designing a multiuser HDTV storage server," *IEEE J. Select. Areas Commun.*, vol. 1, no. 11, 1993.

[16] W. D. Sincoskie, "System architecture for a large scale video on demand service," *Computer Networks ISDN Syst.*, vol. 22, pp. 155–162, 1991.

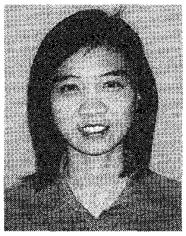
[17] J. Sutherland and L. Litteral, "Residential video service," *IEEE Commun. Mag.*, pp. 36–41, July 1992.

[18] T. S. P. Yum, "Hierarchical distribution of video with dynamic port allocation," *IEEE Trans. Commun.*, vol. 39, no. 8, pp. 1268–1274, Aug. 1991.

Victor O. K. Li (S'80–M'81–SM'86–F'92) was born in Hong Kong in 1954. He received the S.B., S.M., and Sc.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, in 1977, 1979, and 1981, respectively.

Since February 1981, he has been with the University of Southern California (USC), Los Angeles, CA, where he is Professor of Electrical Engineering and Director of the USC Communication Sciences Institute. His research interests include high speed communication networks, personal communication networks, multimedia systems, distributed databases, queueing theory, graph theory, and applied probability. He has lectured and consulted extensively around the world, and has served as an Editor of *IEEE Networks* and of *Telecommunication Systems*, Guest Editor of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS and of *Computer Networks and ISDN Systems*, and is now serving as an Editor of the *ACM/Baltzer Wireless Networks*.

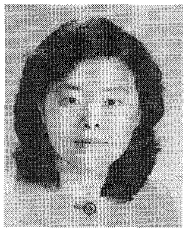
Dr. Li chaired the Computer Communications Technical Committee of the IEEE Communications Society, from 1987 to 1989, and the Los Angeles Chapter of the IEEE Information Theory Group, from 1983 to 1985. He is the Steering Committee Chair of the International Conference on Computer Communications and Networks (IC^{3}N), General Chair of the 1st Annual IC^{3}N, June 1992, Technical Program Chair of the Institution of Electrical Engineers (IEE) Personal Communication Services Symposium, June 1995, and Chair of the 4th IEEE Workshop on Computer Communications, October 1989. He serves on the International Advisory Board of IEEE TENCON'90, IEEE TENCON'94, IEEE SICON'91, IEEE SICON'93, IEEE SICON/ICIE'95, the International Conference on Microwaves and Communications '92, and the International Symposium on Communications '91.



Wanjiun Liao was born in Taiwan in 1968. She received the B.S. and M.S. degrees from National Chiao Tung University, Taiwan, ROC, in 1990 and 1992, respectively. She is currently a Ph.D. student in the Department of Electrical Engineering at the University of Southern California, Los Angeles, CA.

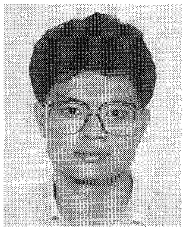
Her research interests include multimedia computing and communications, high-speed networks, and distributed database systems.

Ms. Liao is a recipient of the Acer Long-Term Thesis Award, the Chinese IEE Graduate student Thesis Award, and is a member of the Phi Tau Phi Scholastic Honor Society.



Xiaoxin Qiu received the B.E. and M.E. degrees from Tsinghua University, PROC, in 1990 and 1991, respectively. She is currently a Ph.D. student in the Department of Electrical Engineering of the University of Southern California, Los Angeles, CA.

Her research interests are in the areas of wireless communications networks, personal communication systems, and multimedia communications.



Eric W. M. Wong (S'88–M'88) received the B.Sc. and M.Phil. degrees in electronic engineering from The Chinese University of Hong Kong, in 1988 and 1990, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Massachusetts, Amherst, in 1994.

In August 1994, he joined the City University of Hong Kong as a University Lecturer in Electronic Engineering. His research interests are in high-speed networks, video-on-demand, wireless communications, and dynamic routing.

Dr. Wong received First Prize in the 1988 IEEE Hong Kong Section Student Paper Contest.