Performance of Batching Schemes for Multimedia on Demand Services

Gennaro Boggia, *Member, IEEE*, Pietro Camarda, Luigi Mazzeo, and Marina Mongiello

Abstract

Recent advances in information and communication technologies have made multimedia on demand services technically and economically feasible. Important aspects of such systems are the resource sharing techniques, which allow the simultaneous service of a large number of users with considerable savings in terms of network bandwidth and server resources. In this paper we report the results of a study which analyzes batching and buffering techniques, which involves serving all video requests issued during a short interval of time with a single stream. The mathematical model, based on queueing networks, allows the evaluation of the main system performance (average and probability distribution of the number of streams, percentage reduction of resources, and so on) as a function of load and batching interval duration. Simulation experiments confirm the analytical model in the whole range of considered conditions.

Index Terms

Multimedia services, batching techniques, analytical and simulation model.

I. INTRODUCTION

MULTIMEDIA systems providing on demand services, such as Video on Demand (VoD), distance learning, internet video broadcast, and so on are now fully feasible thanks to recent advances in networking and computer technologies. One of the most challenging aspects of such systems is the architecture design for providing the required service with the appropriate Quality of Service (QoS). QoS is determined by several parameters (such as I/O bandwidth, memory requirements, and so on) which behave as critical components in designing system architecture; in fact, the system must satisfy the real-time constraints for continuous delivery of video streams at a specified bandwidth.

There are many approaches towards improving the efficiency of Multimedia on Demand systems by managing critical resources [1]. These include disk scheduling and data placement algorithms, techniques for memory management, strategies for scheduling of CPU or other resources, and resource sharing techniques [2]-[3]. Several approaches for optimizing system performance, by sharing the available resources, consider the network bandwidth required to satisfy the client requests as a critical resource. The objective is to reduce the bandwidth demand, increasing the number of requests which can be served simultaneously.

Existing schemes for allocating network resources are broadly classified in user-centered and data-centered approaches. In a user-centered approach, for each client request, the server dedicates specific network resources to deliver the requested video stream. Efficient user-centered scheduling algorithms have been studied extensively in [4].

In the data-centered approach, resources are dedicated to video objects rather than to users. It allows users to share a video stream through the multicast feature of modern communication networks. If more clients request the same video at about the same time, all the requests can be satisfied by one video stream, thereby reducing the demand for both network and server I/O bandwidth.

Data centered approaches have been further classified in server initiated, server initiated with prefetching; client initiated, client initiated with prefetching. The first two categories include server-push techniques; such schemes can generally guarantee a maximum service latency independent of the arrival time of the request. The second two techniques are classified as client-pull techniques.

Server-push techniques are typically designed for “hot” (i.e., frequently accessed) video objects. They employ a fixed number of multicast channels to periodically broadcast video objects to a group of subscribers. The difference between various server-push techniques lies in the broadcast schemes used.

Among server-push techniques, a novel video delivery technique called catching is reported in [5]. This technique can utilize sever and network resources while providing near instantaneous service to clients. The technique is achieved through an integration of the server-push and client-pull video delivery programs. Further minor extensions of catching are selective catching and controlled multicast [5].

In order to achieve higher user capacity and lower network transmission cost, distributed server architecture can be used, in which multiple local servers dynamically cache the contents streamed from the repository [6]. In [7] a number of caching

Manuscript received XXXX; revised XXXX.

G. Boggia, P. Camarda, and M. Mongiello are with the Department of “Elettrotecnica ed Elettronica”(DEE) - Politecnico di Bari. Via Orabona, 4 – 70125 BARI, Italy. (e-mail: {g.boggia, camarda, mongiello}@poliba.it).

L. Mazzeo is with RSI Sistemi (Altran Group). Via A. De Togni, 2 – 20123 Milan, Italy. (e-mail: luigi.mazzeo@rsisistemi.it).

schemes are studied as applied in the local servers, depending on whether the repository is able to multicast movie contents to the local servers.

An important question in providing on-demand access to large, widely shared data files is how to effectively use regional (proxy) servers that can store some of the data close to the clients. The proxy caching problem is more complex in the context of media files because of the need to consider bandwidth at the proxy server and because of the bandwidth sharing possibilities provided by recently proposed multicast delivery techniques. In [8] new analytical models are developed for determining optimal proxy cache context in such environments.

Server-push techniques have the advantage that they utilize server and network resources more efficiently. But this efficiency is achieved through increased service latency, as a client can only start receiving a video object at the beginning of the next broadcast period.

Batching of requests [1] [9] is a client-pull technique in which the requests for the same video, emitted during a short interval of time, i.e. the batching interval, are grouped in a batch and the video display is delayed for this interval. Thus, multiple customers requiring the same video within the batching interval can be served by multicasting the same video stream, with considerable savings in server and network capacity.

In client-pull with prefetching techniques, two clients that request the same video can share a multicast stream without delaying the first client. Merging of video streams and buffering in central memory belong to this class. In merging strategies, the video streams are merged into a single one by adjusting the display rates of requests for the same object. The practical constraints and some applications of these techniques are described in [10], [11], [12], [13].

The buffering technique involves using the central memory as a buffer, holding a given number of frames behind a video stream. Subsequent requests for that video, within the batching interval, can be served by using the buffer rather than requesting another stream from the server [14], [15], [16]. In this technique the user request is served immediately without any delay. In fig. 1 an example of buffering serving N users is shown, each one with its specific playback¹ point in the buffer.

Fig. 1. Example of buffering technique.

Other recently proposed client-pull with prefetching techniques are: periodic broadcast [17], [18], patching [19], tapping [20] and their variants [21].

Patching techniques take advantages of the underlying network multicasting capabilities to reduce server and network resource requirements. Such a scheme relies on sophisticated user equipment which must be able to receive two or more streams simultaneously. In the simplest case of such an approach, the user request is served by an existing stream for that video object (if there is any) which is buffered by the user and simultaneously a new stream is requested from the server for the frames already transmitted [5], [19], [22]. Let us call t_a the time needed to receive and display these frames. At the end of such an operation, the video display continues, taking the frames needed from the user buffer, provided that such a buffer can store all frames received in the time interval t_a . In such a simple case the analytical model developed in the sequel is perfectly suitable for analyzing system performance.

The approaches previously mentioned are not mutually exclusive and can be exploited simultaneously. In fact, as can be easily realized, a user request can exploit patching schemes to join a previously initiated batch of requests and/or the frame rate can be suitably modified to allow the merging of the new user request into an existing batch [23].

These functions may be implemented in a multitiered hierarchical distributed video server as depicted in fig.2. Through a Residential Access Network (RAN), clients are connected to a Local Switching Office (LSO) storing the most popular video programs. User requests can be satisfied by the LSO or may be forwarded either to Regional Video Servers through a Metropolitan Area Network (MAN), or to remote servers through a backbone network [1], [9]. Alternatively, the previous functions can be implemented in a standard internet environment [24].

In this paper an original model for evaluating system performance in the hypotheses of batching and buffering techniques has been proposed and analyzed. In particular, the probability distribution of the number of batches, the percentage reduction of resources, and so on, are evaluated as a function of load and batching interval duration. Discrete event simulation experiments confirm the analytical results in the whole range of conditions considered. Previous studies on this topic have developed approximate methods for analyzing some aspects of batching policies [25], [26]. Specifically, in [25] the user requests are stored and served when resources become free following scheduling schemes such as First Come First Served (FCFS), Round

¹The playback point of a video stream is the current position in the objects display of stream [10].

Robin (RR), Maximum Queue Length (MQL). Further refinements are introduced in [26] where an improvement of MQL, the Maximum Factored Queue Length (MFQL), is analyzed. Some aspects of buffering have been analyzed in [16].

The previously mentioned models only analyze specific schemes of batching and buffering. The approach followed in this paper is more general. In fact, the model developed can be applied either to batching or buffering and minor variants can be exploited for analyzing specific implementations of patching and merging. A few results related to merging technique will be reported in section III-D. In particular, in the model presented in this paper the batching interval (t_B) it is fixed a priori, which corresponds to the maximum waiting time in batching policy, while in the buffering scheme, such a quantity determines the amount of central memory needed to store a part of the stream.

The user requests are not stored waiting for available resources but the approach followed in this paper is aimed at determining the resources needed and thus oriented for designing system infrastructure (server and network bandwidth). As a consequence, the results obtained in this paper are not directly comparable with previous results.

The paper is organized as follows. Section II provides a customer behavior model based on a queueing network, introducing the performance evaluation parameters used to investigate the model performance. In Section III we study the effectiveness of the model and show the numerical results; furthermore, we simulate the system operations and discuss simulation results as a measure to validate the proposed model. Section IV presents the concluding remarks.

Fig. 2. Multimedia on Demand System.

II. SYSTEM MODELING

A. System Description

In this section an analytical model for performance evaluation of a multimedia system which adopts buffering or batching techniques is described. Specifically, N_U users connected to the multimedia system are considered. Each free user generates a new video request following an exponential distribution with parameter λ [requests/min]. Let π_c and S_c be, respectively, the request probability and the duration of video c , with $c = 1, 2, \dots, F$, where F is the total number of available video programs in the system. For sake of simplicity, we suppose that the only interactive user operation is the video request, i.e., there are no VCR-like operations (e.g., pause, stop, fast-forward, rewind, etc.). Let N_A be the maximum number of simultaneous active users, i.e., users with a video display in progress.

The analysis is conducted following two phases. In the first phase, the number of requests for each video program is evaluated developing a multiclass queueing networks model. In the second phase, based on the results of the first, iterative algorithms are developed for evaluating the main system performance parameters (probability distribution of the number of batches, percentage reduction of resources, etc.) as a function of load, batching interval, etc.

As regards the first phase, we describe the system with a closed queueing network (see fig. 3) composed by two service centers and $F + 1$ classes. The first center considers users belonging to class 0, which corresponds to the idle state. It can be modeled by an Infinite Servers (IS) service center, with exponential service time (average value $T_0 = 1/\lambda$). The second center models users in the active state. Each user belongs to a class which corresponds to the requested video. At service completion, the user returns to the first center at class 0. Also this second service center can be modeled as an Infinite Server (IS) service center, with a constraint on the global number of users which cannot exceed the maximum value N_A . The service time in this second center is supposed general with average value T_c . The hypothesis of infinite servers means that there are no upper limits in the number of permitted streams.

The rationale behind these choices can be appreciated by considering that in some implementations, e.g., buffering techniques applied to the residential access network, each active user is served with an individual video stream. Thus each user needs specific access to the memory to fetch the requested frames and the limited memory bandwidth imposes a constraint on the simultaneous number of active users. Obviously, the maximum number of active users, N_A , can be fixed equal to the number of connected users, N_U , when no constraints need to be applied in the considered system. In contrast, no constraint is considered for the number of streams. This choice is determined by the consideration that this model, as reported before, has been developed as an aid to network design (rather than analysis). It should also be pointed out that it is possible to approximate traditional batching systems [4] in which there are no limits to the maximum number of users that can be supported, using a large value for N_U and considering $N_A = N_U$. After the first request, all users that make requests for the same video in the batching interval t_B are grouped together, sharing system resources. In particular, with the buffering technique, the batch

Fig. 3. System modeling with queueing network.

interval duration t_B determines the amount of central memory to be allocated for buffering, temporarily, a portion of frames of the needed video stream. The allocated resources are released when the last user ends video display. Then the service time of an active class c user is equal to the deterministic video duration S_c . With the batching technique, users belonging to the same batch are served together, multicasting the same video stream. The display of the chosen video starts at the end of the batching interval. In this case, the service time is general with an average value given by the sum of the deterministic video duration and the mean waiting time before video start. Since the incoming video c request represents a random point within the batching interval, the mean waiting time is $t_B/2$ [27], then the mean service time is $S_c + t_B/2$.

B. Distribution of idle and active users

As described before, $F + 1$ classes are present in the system and by hypothesis, the routing probabilities are:

$$p_{0,0} = 0; \quad p_{0,c} = \pi_c; \quad p_{k,0} = 1; \quad p_{k,c} = 0; \quad (1)$$

with $c = 1, \dots, F$ and $k = 1, \dots, F$.

Let $n_{(1)}$ and $n_{(2)}$ be, respectively, the number of users at the first center and at the second one; let n_c , with $c = 0, 1, \dots, F$, be the number of class c users. To solve the queueing network, the first step is the resolution of the following homogeneous system of linear equations to find the relative arrival rate [28]:

$$y_c = \sum_{d=0}^F y_d p_{dc} \quad c = 0, 1, \dots, F. \quad (2)$$

Thus, the relative service time at class c is:

$$b_c = y_c \cdot T_c \quad c = 0, 1, \dots, F \quad (3)$$

and the relative service times at service centers are:

$$b_{(1)} = y_0 \cdot T_0 \quad b_{(2)} = \sum_{c=1}^F y_c T_c. \quad (4)$$

In the hypotheses stated, the network admits a product form solution [29] and the state probabilities are given by:

$$P(n_0, n_1, \dots, n_F) = \frac{1}{G} \prod_{i=0}^F \frac{b_i^{n_i}}{n_i!} \quad (5)$$

while the distribution of users at the two service centers assumes the following probability

$$P(n_{(1)}, n_{(2)}) = \frac{1}{G} \frac{b_{(1)}^{n_{(1)}} b_{(2)}^{n_{(2)}}}{n_{(1)}! n_{(2)}!} \quad \text{with } (n_{(1)}, n_{(2)}) \in \mathcal{S} \quad (6)$$

where $\mathcal{S} = \{n_{(1)}, n_{(2)} \mid n_{(1)} + n_{(2)} = N_U, n_{(2)} \leq N_A\}$ is the states space and G is the normalization constant.

Since $n_{(1)} = N_U - n_{(2)}$ and the maximum value for $n_{(2)}$ is N_A , we have [29]:

$$G = \sum_{(n_{(1)}, n_{(2)}) \in \mathcal{S}} \frac{b_{(1)}^{n_{(1)}} b_{(2)}^{n_{(2)}}}{n_{(1)}! n_{(2)}!} = \sum_{i=0}^{N_A} \frac{b_{(1)}^{N_U-i} b_{(2)}^i}{(N_U-i)! i!} \quad (7)$$

Once this constant has been found, it allows the evaluation of the marginal state probability for the number of users at each center:

$$P(n_{(1)} = i) = P(n_{(1)} = i, n_{(2)} = N_U - i)$$

with $i = N_U - N_A, \dots, N_U$, and

$$P(n_{(2)} = i) = P(n_{(1)} = N_U - i, n_{(2)} = i)$$

with $i = 0, \dots, N_A$.

The marginal state probability for the number of users at each class c ($c = 1, \dots, F$) is

$$P(n_c = i) = \sum_{k=i}^{N_A} P(n_c = i \mid n_{(2)} = k) P(n_{(2)} = k)$$

Considering that each user makes an independent choice, we have [28]:

$$P(n_c = i) = \sum_{k=i}^{N_A} \binom{k}{i} \left(\frac{b_c}{b_{(2)}} \right)^i \left(\frac{b_{(2)} - b_c}{b_{(2)}} \right)^{k-i} P(n_{(2)} = k). \quad (8)$$

where $b_c/b_{(2)}$, as can be easily realized exploiting renewal theory arguments, represents the probability that the video c is chosen by a generic user.

C. Batch Modeling

The second phase is based on the results of the first and allows the evaluation of the probability distribution of the number of batches for each video, the distribution of the total number of batches and the related performance indices.

Let n_{Bc} be the number of batches for the video c , with $c = 1, \dots, F$. By the total probability theorem, we have:

$$P(n_{Bc} = i) = \sum_{j=i}^{N_A} P(n_{Bc} = i \mid n_c = j) P(n_c = j) \quad (9)$$

with $c = 1, \dots, F$ and $i = 0, \dots, N_A$. The index j starts from i because the number of users must be greater or equal to the number of batches.

It should be noted that:

- $P(n_{Bc} = 0 \mid n_c = j) = 0$ $j > 0$. In presence of active users, the minimum value for n_{Bc} is 1, i.e., all users require the same video in the batching interval.
- $P(n_{Bc} = 0 \mid n_c = 0) = 1$. Without active users, n_{Bc} is certainly 0.
- $P(n_{Bc} = i \mid n_c = 0) = 0$ $i > 0$. Without active users, there are no batches.

For evaluating the conditional probability $P(n_{Bc} = i \mid n_c = j)$ (with $i = 1, \dots, N_A$ and $j = i, \dots, N_A$) let us consider i batches with j active class c users and let $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_i)$ be the vector of the number of users for each class c batch, where α_k is the number of users of the k^{th} batch. Obviously, $\alpha_k \geq 1$ and $\alpha_1 + \alpha_2 + \dots + \alpha_i = j$. Summing over all possible states α we obtain:

$$P(n_{Bc} = i \mid n_c = j) = \sum_{\alpha_1 + \alpha_2 + \dots + \alpha_i = j} P(\alpha_1, \alpha_2, \dots, \alpha_i). \quad (10)$$

Now, referring to the time diagram for request arrivals in Fig. 4, where t_k is the instant of the k^{th} user request, the probability $P(\alpha_1, \alpha_2, \dots, \alpha_i)$ is given by the following product:

$$\begin{aligned} & P(t_{\alpha_1} - t_1 \leq t_B, t_{\alpha_1+1} - t_1 > t_B) \cdot \\ & \cdot P(t_{\alpha_1+\alpha_2} - t_{\alpha_1+1} \leq t_B, t_{\alpha_1+\alpha_2+1} - t_{\alpha_1+1} > t_B) \cdot \\ & \cdot \dots \cdot \\ & \cdot P(t_{\alpha_1+\dots+\alpha_{i-1}} - t_{\alpha_1+\dots+\alpha_{i-2}+1} \leq t_B, \\ & \quad , t_{\alpha_1+\dots+\alpha_{i-1}+1} - t_{\alpha_1+\dots+\alpha_{i-2}+1} > t_B) \cdot \\ & \cdot P(t_{\alpha_1+\dots+\alpha_i} - t_{\alpha_1+\dots+\alpha_{i-1}+1} \leq t_B). \end{aligned} \quad (11)$$

Fig. 4. Time diagram for request arrivals.

Each of the first $i - 1$ terms is the probability that $t_q - t_p$, with $1 \leq p \leq q \leq j$, is less than or equal to t_B with $t_{q+1} - t_p$ greater than t_B . This probability (see Appendix A) is given by:

$$\begin{aligned} \Phi(q - p + 1) &= P(t_q - t_p \leq t_B, t_{q+1} - t_p > t_B) = \\ &= \frac{e^{-\lambda_c t_B} (\lambda_c t_B)^{q-p}}{(q-p)!} \end{aligned} \quad (12)$$

where λ_c is the accepted request rate for video c at the active state center, which will be evaluated later.

The last term of (11) is equal to 1 if $\alpha_i = 1$ (i.e., there are no other user arrivals after the instant $t_{\alpha_i+\dots+\alpha_{i-1}+1}$), otherwise it is the probability that the time between a first arrival and an x^{th} one, $t_{p+x} - t_{p+1}$, is less than or equal to t_B ; which, simply by definition, is the distribution of an Erlang random variable [27] with parameters $x - 1$ and c . Thus:

$$\begin{aligned}\Psi(x) &= P(t_{p+x} - t_{p+1} \leq t_B) = \\ &= \begin{cases} 1 - \sum_{k=0}^{x-2} \frac{e^{-\lambda_c t_B} (\lambda_c t_B)^k}{k!} & x \geq 2 \\ 1 & x = 1. \end{cases}\end{aligned}\quad (13)$$

Using (12) and (13), the equation (11) becomes:

$$P(\alpha_1, \dots, \alpha_i) = \Phi(\alpha_1) \dots \Phi(\alpha_{i-1}) \Psi(\alpha_i). \quad (14)$$

Thus, we have:

$$\begin{aligned}P(n_{Bc} = i \mid n_c = j) &= \sum_{\alpha_1 + \alpha_2 + \dots + \alpha_i = j} P(\alpha_1, \alpha_2, \dots, \alpha_i) = \\ &= \sum_{\alpha_1 + \alpha_2 + \dots + \alpha_i = j} \Phi(\alpha_1) \dots \Phi(\alpha_{i-1}) \Psi(\alpha_i).\end{aligned}\quad (15)$$

The previous expression can be evaluated, very simply and with low computational costs, using a recursive expression (see Appendix B).

To evaluate the rate λ_c , let us suppose that each idle user generates a new request, and thus transits to the active state, following an exponential distribution with parameter λ . The average accepted request rate at the active state center is:

$$\lambda_{AS} = \lambda \cdot E[n_R] \quad (16)$$

where $E[n_R]$ is the mean number of accepted video requests. Since no other requests are accepted when the active center is full, the random variable n_R assumes the values:

$$n_R = \begin{cases} (N_U - n_{(2)}) & n_{(2)} < N_A \\ 0 & n_{(2)} = N_A. \end{cases} \quad (17)$$

Its average value is:

$$\begin{aligned}E[n_R] &= \sum_{i=0}^{N_A-1} (N_U - i) P(n_{(2)} = i) = \\ &= \sum_{i=0}^{N_A} (N_U - i) P(n_{(2)} = i) - (N_U - N_A) P(n_{(2)} = N_A)\end{aligned}\quad (18)$$

then:

$$E[n_R] = N_U - E[n_{(2)}] - (N_U - N_A) P(n_{(2)} = N_A) \quad (19)$$

where $E[n_{(2)}]$ is the mean number of active users.

The accepted request rate at the active center, for class c is:

$$\lambda_c = \lambda_{AS} \cdot \pi_c \quad (20)$$

where π_c is the video c request probability.

The distribution of total number of batches, n_B , is

$$P(n_B = i) = \sum_{n_{B_1} + n_{B_2} + \dots + n_{B_F} = i} P(n_{B_1}, n_{B_2}, \dots, n_{B_F}) \quad (21)$$

with $i = 0, 1, \dots, N_A$.

This expression can be evaluated with an efficient algorithm as reported in Appendix C.

TABLE I
REQUEST PROBABILITY AND DURATION FOR EACH VIDEO

Video	1	2	3	4	5	6	7	8	9	10
Request Probability (π_i)	0.2	0.2	0.15	0.15	0.075	0.075	0.075	0.025	0.025	0.025
Duration S_i [min]	120	90	85	100	60	120	90	75	90	120

D. Performance Indices

At this point some parameters for evaluating the performance of the analyzed multimedia system are defined. The first performance parameter is the mean number of batches in the system:

$$E[n_B] = \sum_{c=1}^F E[n_{B_c}] = \sum_{c=1}^F \sum_{i=0}^{N_A} i \cdot P(n_{B_c} = i). \quad (22)$$

Another useful performance parameter is the probability of unsuccessful video request, P_U , i.e., the probability that a user, making a video request, does not find available system resources because there are already N_A active users. Such a parameter can be evaluated by considering the probability that the remaining $N_U - 1$ users occupy all available resources [30], i.e.,

$$P_U = P(n_{(2)} = N_A) \quad (23)$$

with $N_U - 1$ users in the system.

The percentage reduction of resource requirement, $R\%$, can be estimated by:

$$R\% = \left(1 - \frac{E[n_B]}{E[n_{(2)}]}\right) \cdot 100 \quad (24)$$

where $E[n_{(2)}]$ is the mean number of active users given by:

$$E[n_{(2)}] = \sum_{i=0}^{N_A} i \cdot P(n_{(2)} = i). \quad (25)$$

The model allows the evaluation of other significant performance parameters. As an example, by equation (21) the variance of the total number of batches can be easily evaluated. Such a quantity can be exploited in designing a suitable network infrastructure for the VoD service.

III. NUMERICAL RESULTS

In this section we provide some numerical results of the analytical model and compare them with the results obtained from simulation in a simple VoD system. We consider $N_U = 100$ users, $F = 10$ available video programs, a given request probability and a video duration for each video (see Table I). The results illustrated refer to the buffering techniques. The application to batching may be easily obtained as described in section II by considering as the mean service time the value $S_c + t_B/2$.

The behavior of the mean number of batches, $E[n_B]$, is reported as a function of the video request rate per user, λ [requests/min], the batching interval, t_B [min] and the maximum number of active users, N_A . Other important results are concerned with $R\%$, that is the percentage reduction of resource requirement when resources are shared among the users of a batch, with respect to the direct allocation of resources to users; $R\%$ behavior is shown as a function of λ with parameter t_B and N_A . Moreover, we investigate the unsuccessful video request probability P_U as a function of the arrival rate λ and for different values of N_A and as a function of N_A with λ as a parameter. In all the following figures, the plain lines represent analytical results, while the simulation results are reported using the symbol '+'.

Generally speaking, the simulation results follow the analytical results quite well. Specifically, the mean number of active users is approximated almost exactly, while there is a slight discrepancy for the mean number of batches. This behavior can be explained by considering the fact that the analytical expression for evaluating the mean number of active users does not contain any approximation and as a consequence, it appears natural that the simulation, being based on the same hypothesis, gives results that are almost identical to the analysis. The small difference in the other results is due to the approximations introduced in the analytical model for their evaluation. Specifically, exploiting an average value of the accepted request rate (λ_{AS}) for all states, represents a slight approximation that, as highlighted before, affects the obtained results only marginally.

Fig. 5. Mean number of active users and batches vs. λ ($N_A = 60$ users).

A. Mean number of active users and batches

A comparison of the mean number of active users, $E[n_{(2)}]$, and mean number of batches, $E[n_B]$, is illustrated in fig. 5, where the maximum number of active users is $N_A = 60$. We note that $E[n_B]$ approaches to $E[n_{(2)}]$ for λ close to zero. This means that, for small values of λ , the probability of more than one request for the same video in the time interval t_B is not relevant, then each batch has approximately one user. Note that the differences between the two curves become larger when λ grows. Here $E[n_B]$ assumes increasing values but always keeps below $E[n_{(2)}]$. This is due to the fact that there is a higher number of active users that are grouped in batches.

Figures 6 and 7 represent, respectively, the dependence of $E[n_{(2)}]$ and of $E[n_B]$ on the maximum number of active users. In these figures, we consider a batching interval of $t_B = 10$ minutes. The difference between the two curves becomes meaningful for large values of where the system has high blocking probability. For decreasing values of N_A each curve reaches the saturation for smaller values of λ .

Fig. 6. Mean number of active users vs. λ ($t_B = 10$ min).

Fig. 7. Mean number of batches vs. λ ($t_B = 10$ min).

Fig. 8 shows $E[n_B]$ as a function of N_A ; we fix t_B at 10 minutes and observe $E[n_B]$ for different values of λ . For a fixed value of λ , an increase in N_A causes an increase in $E[n_B]$ even though t_B does not change. This is due to the fact that more

requests arrive to the system in the batching interval of fixed duration t_B ; such requests, eventually for different video objects, are then grouped in different batches, thus increasing the number of batches. As far as λ is concerned, we observe that the curves with higher values of λ are always disposed above those corresponding to lower values of λ . This is due to the fact that if λ grows, for fixed N_A and t_B , there is a greater number of batches.

Fig. 9 shows $E[n_B]$ versus N_A for different values of the parameter t_B . For a given value of N_A and λ , the mean number of batches decreases with the duration of t_B since a higher number of requests may arrive to the system for a wider t_B .

Fig. 8. Mean number of batches vs. maximum number of active users ($t_B = 10$ min).

Fig. 9. Mean number of batches vs. maximum number of active users ($\lambda = 1$ requests/min).

Figures 10 and 11 show $E[n_B]$ as a function of the batching interval. As expected, with the request rate and the maximum number of active users fixed, if t_B increases, the mean number of batches decreases. In fact, with higher values for t_B , we have more requests in the same batching interval and then a lower number of batches.

Fig. 10. Mean number of batches vs. batching interval ($\lambda = 0.1$ requests/min).

B. Percentage reduction of resource utilization

The percentage reduction of resource utilization, $R\%$, becomes meaningful when evaluating the bandwidth reduction in the case of shared bandwidth. Moreover, $R\%$ does not depend on the particular video object required by each batch and has a global meaning.

Fig. 11. Mean number of batches vs. batching interval ($N_A = 60$ users).

Fig. 12 shows the behavior of $R\%$ versus λ for several values of N_A and $t_B = 10$ min. $R\%$ grows for higher values of N_A , and in any case this value strictly depends on the physical resources used to implement the system. In Fig. 13, $R\%$ is

Fig. 12. Percentage reduction of resources vs. λ ($t_B = 10$ min).

reported as a function of λ with parameter t_B for $N_A = 60$. $R\%$ increases for higher values of t_B , which means more main memory in the case of buffering or larger response time for delivery of video objects in the case of batching.

Fig. 13. Percentage reduction of resources vs. λ ($N_A = 60$ users).

C. Unsuccessful video request probability

Fig. 14 shows the probability P_U that a customer request is refused, as a function of the arrival rate λ for various values of N_A . For small values of λ we note that P_U keeps close to zero; gradually, for increasing values of λ , P_U grows and reaches the maximum value for high values of λ . Such behavior is due to the fact that for small λ there is a small number of active users, then a new video request is likely to be accepted. When λ grows, the number of users in the active state rises and the blocking probability is higher. The smaller N_A , the smaller the value of λ where P_U becomes meaningful. Such a result is

Fig. 14. Unsuccessful video request probability vs. λ .

intuitively justified by a reduced capability of the system to satisfy customer requests. In fig. 15, P_U is reported as a function of N_A ; as expected, the block has a high probability for small values of N_A and decreases for increasing values of N_A .

Fig. 15. Unsuccessful video request probability vs. N_A .

D. Further applications

In this subsection we give an example on how to apply the model developed in this paper to other batching techniques. We would like to point out that, as said before in the introduction, our model can be applied to batching techniques with a fixed a priori t_B , which corresponds to the user maximum waiting time. Nevertheless, this model is useful for network design (see next section) given that it helps in dimensioning VoD system infrastructure to guarantee the performance needed.

In particular, here we consider a simple comparison with some results obtained by Lau et al. in [10] where video requests in a batching interval (t_B) are merged together adjusting the display rates.

To make a fair comparison, we refer to the same values adopted in the performance section of [10]. In particular, in the sequel we only consider one video class (i.e., $F = 1$ and $\pi_1 = 1$ in our model) of $S_1 = 120$ minutes of duration. To take into account that in the system studied by Lau et al. there are an infinite number of users without limitation on the number of active users, here in our model we consider a large number of users ($N_U = 200$) and $N_A = N_U$.

As stated in sec. II-A, to characterize our model in this context, it is necessary to estimate the value of the service time T_1 for the generic user, taking into account that the normal video duration is modified. In fact, videos are merged together regulating the display rate by factor Δ . However, considering the same value of $\Delta = 0.05$ used in [10], for a small value of t_B , we can neglect the variation of video duration and consider $T_1 \simeq S_1$, i.e., the normal video duration.

It is important to note that a complete comparison requires a more detailed study and a more accurate evaluation of parameters like service time T_1 and percentage reduction of resource requirements $R\%$. In fact, in this case, the parameter given by Eq. 24 only gives a good estimation of resource saving after flow merging.

Figure 16 shows results obtained with our model compared with the ones reported in literature, in terms of percentage reduction of resource requirements as a function of the mean inter-arrival time between video requests. In the evaluation, the mean inter-arrival time is obtained as $1/\lambda_{AS}$, where λ_{AS} is given by eq. 16, and $t_B = 2$ min. The figure shows a marked similarity between the results, confirming the validity of our general approach to different batched VoD systems.

Fig. 16. Percentage reduction of resources for various models with merging technique ($t_B = 2$ min).

E. Some design guidelines

The results obtained in the previous sections can be exploited for choosing the system parameters that guarantee the performance needed. As an example, with a fixed value of the number of users and of the video request rate, the minimum number of active users N_A , that the system must be able to support with a given unsuccessful probability, can be derived from the previous figures 14-15. From the knowledge of N_A , it is possible to evaluate, for different values of t_B , the mean number of batches from figures 6-11, and the mean percentage reduction of system resources through figures 12-13. Thus, we can choose the best value of t_B in order to obtain a given level of system performance. This last value (t_B) is directly related to the dimension of infrastructure elements such as memory buffers. Alternatively, fixing the system resources in terms of mean number of batches, the maximum waiting time in batching schemes or the maximum size of memory in buffering schemes can be obtained.

In addition, a thorough analysis, not reported in this paper for brevity, can be developed exploiting the distribution of the number of batches given by equations (21) and (37). In fact, the knowledge of the probability distribution of the number of batches allows a statistical evaluation of the aggregate bandwidth, needed in order to guarantee a given loss probability [31], [32]. This last probability can be considered as a suitable Quality of Service (QoS) index. Moreover, the estimation of the aggregate bandwidth required by the video system can be fruitfully exploited to dimension the network link bandwidth.

F. Simulation environment

The simulator is an event oriented simulation program written in C language and using the libraries of the SMPL simulation language [33]. Two main events characterize system behavior, both describing the transitions of the user requests between the two service centers. Event 1 occurs when a user leaves service center 1 to reach service center 2 in the queuing network. This corresponds to a transition from the idle state to the active state. After the service, the user joins service center 1 and returns to the idle state, after which event 2 occurs. In the case of lack of resources of center 2, the request is not accepted and the user makes a new attempt after a new round of service in center 1. The system clock advances when one of these two events occurs.

Interarrival times between two video requests for the same user are generated randomly by suitable exponential distributions. Users require videos following the request probability shown in Table I. Through simulation we measure the mean number of active users in the system, the mean number of batches and the unsuccessful video request probability as follows:

$$\begin{aligned} E[n_{(2)}] &= \frac{1}{T_{max}} \int_0^{T_{max}} n_{users}(t) dt; \\ E[n_B] &= \frac{1}{T_{max}} \int_0^{T_{max}} n_{batches}(t) dt; \quad P_U = \frac{N_{V_{ref}}}{N_{V_{req}}}. \end{aligned} \quad (26)$$

where $n_{users}(t)$ and $n_{batches}(t)$ are respectively the number of users in the active state and the number of batches at the time t , and $N_{V_{ref}}$ is the number of refused requests during the simulation, and $N_{V_{req}}$ denotes the number of attempted requests over the simulation duration, that is T_{max} .

IV. CONCLUSIONS

In this paper we have developed an analytical model to study the performance of a multimedia on-demand system with resource sharing, exploiting batching and buffering techniques. The rationale behind the development of this analytical model can be understood by considering that the computational complexity of the model is surprisingly low, allowing, as a consequence, the analysis of quite realistic systems. The analytical results have been validated by a discrete event simulation model, which, as expected, produces results in a time that is orders of magnitude greater than the time employed by the analytical model.

As we can see from the results obtained, the developed model allows a good system design providing a tradeoff among contrasting needs. In fact, high values of the batching interval give high performance in terms of reduction of resource requirement, e.g., transmission bandwidth, I/O bandwidth for video servers, and so on. But this reduction causes an increase in the waiting time for each user with the batching technique or an increase of memory required to store the video frames with the buffering technique. Moreover, we have the maximum resource saving for high values of the video request rate, but, in this case, the system operates with highly unsuccessful probability. Finally, the developed model can be exploited for analyzing other data centered approaches applied hierarchically on the considered system. As an example, the gain obtained by stream merging approaches, applied after batching, can be quantitatively evaluated. In fact, the distribution of the number of batches for each video (equation (9)), and the inter-batching time distribution (which can be evaluated quite easily, see figure 4), again allow us to apply the batch modeling of section II-C to evaluate, in the considered VoD systems, the merging features applied on streams derived from batching of user requests.

APPENDIX A

In this appendix the probability $P(t_q - t_p \leq t_B, t_{q+1} - t_p > t_B)$ with $t_1 \leq t_p \leq t_q \leq t_{j-1}$ is evaluated, where $t_1, t_2, \dots, t_j$ are the request arrival instants. We have:

$$\begin{aligned} P(t_q - t_p \leq t_B, t_{q+1} - t_p > t_B) &= \\ &= P(t_{q+1} - t_q + t_q - t_p > t_B, t_q - t_p \leq t_B) = \\ &= P(t_{q+1} - t_q + \tau > t_B, \tau \leq t_B). \end{aligned} \quad (27)$$

Considering the total probability theorem and the memoryless property of exponential distribution, observing that τ is an Erlang random variable with parameters λ_c and $q - p$, eq. (27) becomes:

$$\begin{aligned} &\int_0^{t_B} P(t_{q+1} - t_q + \tau > t_B \mid \tau = x) f_\tau(x) dx \\ &= \int_0^{t_B} P(t_{q+1} - t_q > t_B - x) \frac{e^{-\lambda_c x} \lambda_c^{q-p} x^{q-p-1}}{(q-p-1)!} dx = \\ &= \int_0^{t_B} e^{-\lambda_c(t_B-x)} \frac{e^{-\lambda_c x} \lambda_c^{q-p} x^{q-p-1}}{(q-p-1)!} dx. \end{aligned} \quad (28)$$

In fact, the first term in the integral is the probability that in the time interval $t_B - x$ there are no arrivals, i.e., $e^{-\lambda_c(t_B-x)}$ by exponential distribution. Thus:

$$P(t_q - t_p \leq t_B, t_{q+1} - t_p > t_B) = \frac{e^{-\lambda_c t_B} (\lambda_c t_B)^{q-p}}{(q-p)!}. \quad (29)$$

APPENDIX B

Here we show how to obtain the expression (15). Since the minimum dimension for each batch is one (i.e., each $\alpha_k \geq 1$ where α_k is still the number of users in the k^{th} batch) we can observe that it is possible to rewrite (15) as:

$$\begin{aligned} P(n_{Bc} = i \mid n_c = j) &= P(i \mid j) = \\ &= \sum_{m=i-1}^{j-1} \left(\sum_{\alpha_1 + \dots + \alpha_{i-1} = m} \Phi(\alpha_1) \dots \Phi(\alpha_{i-1}) \right) \Psi(\alpha_i = j - m) = \\ &= \sum_{m=i-1}^{j-1} \tilde{P}(i-1 \mid m) \Psi(j-m) \end{aligned} \quad (30)$$

where $i > 0$.

Using the expression (12) for each $\Phi(\alpha_i)$, we have:

$$\begin{aligned} \tilde{P}(i \mid m) &= \sum_{\substack{\alpha_1 + \dots + \alpha_i = m \\ \alpha_k \geq 1}} \prod_{z=1}^i \frac{e^{-\lambda_c t_B} (\lambda_c t_B)^{\alpha_z - 1}}{(\alpha_z - 1)!} = \\ &= e^{-\lambda_c t_B} \sum_{\substack{\alpha_1 + \dots + \alpha_i = m \\ \alpha_k \geq 1}} \prod_{z=1}^i \frac{(\lambda_c t_B)^{\alpha_z - 1}}{(\alpha_z - 1)!}. \end{aligned} \quad (31)$$

Using the multinomial formula:

$$\begin{aligned}
\left(\sum_{z=1}^M x_z \right)^N &= \sum_{\substack{n_1 \dots + n_M = N \\ n_k \geq 0}} \frac{N!}{n_1! \dots n_M!} x_1^{n_1} \dots x_M^{n_M} = \\
&= N! \cdot \sum_{\substack{n_1 \dots + n_M = N \\ n_k \geq 0}} \prod_{z=1}^M \frac{x_z^{n_z}}{n_z!},
\end{aligned} \tag{32}$$

and since each $\alpha_k \geq 1$, with the substitution $\beta_k = \alpha_k - 1$, for each $k = 1, \dots, i$, eq. (31) becomes:

$$\begin{aligned}
\tilde{P}(i | m) &= e^{-i\lambda_c t_B} \sum_{\substack{\beta_1 \dots + \beta_i = m-i \\ \beta_k \geq 0}} \prod_{z=1}^i \frac{(\lambda_c t_B)^{\beta_z}}{\beta_z!} = \\
&= e^{-i\lambda_c t_B} \frac{\left(\sum_{k=1}^i \lambda_c t_B \right)^{m-i}}{(m-i)!} = e^{-i\lambda_c t_B} \frac{(i\lambda_c t_B)^{m-i}}{(m-i)!}.
\end{aligned} \tag{33}$$

It is possible to rewrite (13) with a recursive expression:

$$\Psi(x) = \Psi(x-1) - \frac{e^{-\lambda_c t_B} (\lambda_c t_B)^{x-2}}{(x-2)!} = \Psi(x-1) - \chi(x)$$

with $x \geq 2$ and $\Psi(1) = 1$.

Thus, for $j > i$, eq. (30) is rewritten as:

$$\begin{aligned}
P(i|j) &= \sum_{m=i-1}^{j-2} \tilde{P}(i-1|m) \Psi(j-m) + \tilde{P}(i-1|j-1) \Psi(1) = \\
&= \sum_{m=i-1}^{j-2} \tilde{P}(i-1|m) [\Psi(j-m-1) - \chi(j-m)] + \tilde{P}(i-1|j-1) = \\
&= P(i|j-1) - \sum_{m=i-1}^{j-2} \tilde{P}(i-1|m) \chi(j-m) + \tilde{P}(i-1|j-1).
\end{aligned} \tag{34}$$

Using (33) and with the substitution $z = m - i + 1$, after some algebra, the sum in (34) is:

$$\begin{aligned}
&e^{-i\lambda_c t_B} \sum_{m=i-1}^{j-2} \frac{((i-1)\lambda_c t_B)^{m-i+1}}{(m-i+1)!} \frac{e^{-\lambda_c t_B} (\lambda_c t_B)^{j-m-2}}{(j-m-2)!} = \\
&= e^{-i\lambda_c t_B} \sum_{z=0}^{j-i-1} \frac{((i-1)\lambda_c t_B)^z}{z!} \frac{(\lambda_c t_B)^{j-i-1-z}}{(j-i-1-z)!} = \\
&= e^{-i\lambda_c t_B} \frac{(i\lambda_c t_B)^{j-i-1}}{(j-i-1)!} = \tilde{P}(i|j-1).
\end{aligned}$$

Thus, using (33), the (34) becomes the recursive expression

$$\begin{cases} P(i | j) = P(i | j-1) - A_{i,j-1} + B_{i,j-1} \\ A_{i,j} = \frac{i\lambda_c t_B}{(j-i)} A_{i,j-1} \\ B_{i,j} = \frac{(i-1)\lambda_c t_B}{(j-i+1)} B_{i,j-1} \end{cases} \tag{35}$$

with $1 \leq i \leq N_A$, $i < j \leq N_A$ and the initial conditions

$$\begin{aligned}
P(n_{Bc} = i | n_c = i) &= e^{-(i-1)\lambda_c t_B}; \\
A_{i,i} &= \tilde{P}(i|i) = e^{-i\lambda_c t_B}; \\
B_{i,i} &= \tilde{P}(i-1|i) = [(i-1)\lambda_c t_B] e^{-(i-1)\lambda_c t_B}.
\end{aligned} \tag{36}$$

To obtain all $P(n_{Bc} = i | n_c = j)$ values, it is straightforward to show (considering the range of variation of i and j) that the computational cost of the method is $O(N_A^2)$.

APPENDIX C

Here we show how to evaluate the distribution of total number of batches, n_B , given by (21).

Exploiting the total probability theorem and considering the vectors $\mathbf{n}_B = (n_{B_1}, n_{B_2}, \dots, n_{B_F})$ and $\mathbf{n} = (n_0, n_1, \dots, n_F)$, we have

$$P(\mathbf{n}_B) = \sum_{n_0 + n_1 + \dots + n_F = N_U} P(\mathbf{n}_B | \mathbf{n}) P(\mathbf{n}) \quad (37)$$

Considering that the number of class i batches depends only on the number of class i users, it is easy to realize that

$$P(\mathbf{n}_B | \mathbf{n}) = \prod_{i=1}^F P(n_{B_i} | n_i). \quad (38)$$

Exploiting (5), we have

$$P(\mathbf{n}_B) = \sum_{n_0 + n_1 + \dots + n_F = N_U} f_0(n_0) \prod_{i=1}^F f_i(n_{B_i}, n_i) \quad (39)$$

with

$$\begin{aligned} f_0(n_0) &= \frac{1}{G} X_0(n_0) \\ f_i(n_{B_i}, n_i) &= P(n_{B_i} | n_i) X_i(n_i) \\ X_i(n_i) &= \frac{b_i^{n_i}}{n_i!}, \quad i = 0, \dots, F; \\ f_i(0, 0) &= 1 \quad i = 0, \dots, F. \end{aligned}$$

Thus, considering (21), globally we obtain:

$$P(n_B = i) = \sum_{n_{B_1} + \dots + n_{B_F} = i} \sum_{n_0 + \dots + n_F = N_U} f_0(n_0) \prod_{z=1}^F f_z(n_{B_z}, n_z).$$

A brute force approach to evaluate the previous expression for realistic systems is practically impossible due to its high computational complexity. To cope with such a problem, in the sequel, a simple iterative algorithm is derived. Let

$$h_m(j, k) = \sum_{n_{B_1} + \dots + n_{B_m} = j} \sum_{n_0 + \dots + n_m = k} f_0(n_0) \prod_{z=1}^m f_z(n_{B_z}, n_z). \quad (40)$$

Obviously, we have $h_F(i, N_U) = P(n_B = i)$. We can re-write $h_m(j, k)$ as follows:

$$\sum_{s=0}^j \sum_{n_{B_1} + \dots + n_{B_{m-1}} + s = j} \sum_{t=s}^k \sum_{n_0 + \dots + n_{m-1} = k-t} f_0(n_0) \dots f_m(s, t).$$

The index t starts from s because, as can be easily verified, for $t < s$, $f_m(s, t) = 0$. We have

$$\begin{aligned} h_m(j, k) &= \sum_{s=0}^j \sum_{t=s}^k f_m(s, t) \sum_{n_{B_1} + \dots + n_{B_{m-1}} + s = j} \\ &\quad \sum_{n_0 + \dots + n_{m-1} = k-t} f_0(n_0) f_1(n_{B_1}, n_1) \dots f_{m-1}(n_{B_{m-1}}, n_{m-1}) \end{aligned}$$

which becomes

$$h_m(j, k) = \sum_{s=0}^j \sum_{t=s}^k f_m(s, t) h_{m-1}(j-s, k-t); \quad m \geq 2, \quad (41)$$

with the following initial conditions which can be easily derived from (40)

$$h_1(j, k) = \sum_{n_0=0}^k f_0(n_0) f_1(j, k-n_0). \quad (42)$$

The computational complexity of this algorithm is $O(iFN_U)$.

REFERENCES

- [1] V. O. K. Li, W. Liao, X. Qiu, and E. W. M. Wong, "Performance Model of Interactive Video-on-Demand Systems," *IEEE J. Select. Areas Commun.*, vol. 14, no. 6, pp. 1099–1109, Aug. 1996.
- [2] D. J. Gemmell, H. M. Vin, D. D. Kandhlur, and P. V. Rangan, "Multimedia storage servers: a tutorial and survey," *IEEE Computer*, vol. 28, no. 5, pp. 40–49, May 1995.
- [3] D. Meliksetian, F. F. Yu, and C. Y. R. Chen, "Methodologies for designing video servers," *IEEE Trans. Multimedia*, vol. 2, no. 1, pp. 62–69, Mar. 2000.
- [4] A. Dan and D. Sitaram, "A generalized interval caching policy for mixed interactive and long video workloads," in *Proc. of SPIE's Conference on Multimedia Computing and Networking*, San Jose, California, Mar. 1996, pp. 344–351.
- [5] L. Gao, Z. Li Zhang, and D. Towsley, "Catching and selective catching: Efficient latency reduction techniques for delivering continuous multimedia streams," in *Proc. ACM Multimedia'99*, Orlando, FL, Nov. 1999, pp. 203–206.
- [6] J. M. Almeida, D. L. Eager, and M. Vernon, "A hybrid caching strategy for streaming media files," in *Proc. IS & SPIE Conference on Multimedia Computing and Networking 2001, (MMCN 2001)*, San Jose, CA, Jan. 2001, pp. 200–212.
- [7] S. H. G. Chan and F. Tobagi, "Distributed servers architecture for networked video services," *IEEE/ACM Trans. Networking*, vol. 9, no. 2, pp. 125–136, Apr. 2001.
- [8] D. L. Eager, M. Ferris, and M. Vernon, "Optimized caching in systems with heterogeneous client populations," *Performance Evaluation*, vol. 42, pp. 163–185, Sept. 2000.
- [9] J. P. Nussbaumer, B. V. Patel, F. Schaffa, and J. P. G. Sterbenz, "Networking requirements for interactive video on demand," *IEEE J. Select. Areas Commun.*, vol. 13, no. 5, pp. 779–787, June 1995.
- [10] S. W. Lau, J. C. S. Lui, and L. Golubchik, "Merging video streams in a multimedia storage server: Complexity and heuristics," *Multimedia Systems*, vol. 6, pp. 29–42, 1998.
- [11] L. Golubchik, J. C. S. Lui, and R. Muntz, "Reducing I/O demand in video on demand storage servers," in *Proc. ACM SIGMETRICS/PERFORMANCE Joint international conference on Measurement and Modeling of Computer Systems'95 Conference*, Ottawa, Canada, May 1995.
- [12] D. Eager and M. Vernon, "Dynamic skyscraper broadcasts for video-on-demand," in *Proc. Workshop on Multimedia Information Systems (MIS'98)*, Istanbul, Sept. 1998.
- [13] D. Eager, M. Vernon, and J. Zahorjan, "Optimal and efficient merging schedules for video-on-demand servers," in *Proc. ACM Multimedia'99*, Orlando, FL, Nov. 1999, pp. 199–201.
- [14] K. C. Almeroth and M. H. Ammar, "The use of multicast delivery to provide a scalable and inter-active video-on-demand service," *IEEE J. Select. Areas Commun.*, vol. 14, no. 6, pp. 1110–1122, Aug. 1996.
- [15] M. Kamath, D. Towsley, and K. Ramamritham, "Continuous media sharing in multimedia data-base systems," in *Proceedings of the Fourth International Conference on Database Systems for Advanced Applications (DASFAA'95)*, Singapore, pp. 79–86.
- [16] P. S. Yu, J. L. Wolf, and H. Shachnai, "Design and analysis of a look-ahead scheduling scheme to support pause-resume for video-on-demand applications," *Multimedia Systems*, vol. 3, no. 4, pp. 137–149, 1995.
- [17] M. K. Bradshaw, B. Wand, S. Sen, L. Gao, J. Kurose, P. Shenoy, and D. Towsley, "Periodic broadcast and patching services - implementation, measurement, and analysis in an internet streaming video testbed," in *Proc. ACM Multimedia'01*, Ottawa, Canada, Oct. 2001, pp. 280–290.
- [18] S. Sen, L. Gao, and D. Towsley, "Frame-based periodic broadcast and fundamental resource tradeoffs," in *Proc. IEEE IPCCC 2001*, Apr. 2001.
- [19] K. Hua, Y. Cai, and S. Sheu, "Patching: a multicast technique for true video-on-demand services," in *Proc. ACM Multimedia'01*, Bristol, UK, Sept. 1998, pp. 191–200.
- [20] S. W. Carter and D. D. E. Long, "Improving video-on-demand server efficiency through stream tapping," in *Proc. ICCCN '97*, Las Vegas, NV, Sept. 1997.
- [21] L. Gao and D. Towsley, "Threshold-based multicast for continuous media delivery," *IEEE Trans. Multimedia*, vol. 3, no. 4, pp. 405–414, Dec. 2001.
- [22] S. Sen, L. Gao, J. Rexford, and D. Towsley, "Optimal patching schemes for efficient multimedia streaming," in *Proc. of International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV-99)*, Basking Ridge, NY, June 1999.
- [23] N. L. S. Fonseca and R. A. Facanha, "The look-ahead-maximize-batch batching policy," *IEEE Trans. Multimedia*, vol. 4, no. 1, pp. 114–120, Mar. 2002.
- [24] S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams," in *Proc. INFOCOM'99*, New York, Mar. 1999.
- [25] H. Shachnai and P. S. Yu, "On analytic modeling of multimedia batching schemes," *Performance Evaluation*, vol. 33, pp. 201–213, Aug. 1998.
- [26] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "The maximum factor queue length batching scheme for video-on-demand systems," *IEEE Trans. Comput.*, vol. 50, no. 2, pp. 97–110, Feb. 2001.
- [27] L. Kleinrock, *Queueing systems, Volume I: Theory*. Wiley & Sons, 1975.
- [28] S. Lavenberg, editor, *Computer performance modeling handbook*. Academic Press, 1983.
- [29] N. M. V. Dijk, *Queueing Networks and Product Forms. A System Approach*. Wiley & Sons, 1993.
- [30] A. L. Garcia, *Probability and Random Processes for Electrical Engineering*. Addison Wesley, 1989.
- [31] Z.-L. Zhang, J. Kurose, J. D. Salehi, and D. Towsley, "Smoothing, statistical multiplexing, and call admission control for stored video," *IEEE J. Select. Areas Commun.*, vol. 15, no. 6, pp. 1148–1166, Aug. 1997.
- [32] G. Boggia, P. Camarda, and D. Striccoli, "Aggregate bandwidth estimation in stored video distribution systems," *Int. J. Commun. Systems*, vol. 15, no. 6, pp. 531–551, July-Aug. 2002.
- [33] M. H. Dougall, *Simulating Computer Systems: Techniques and Tools*. MIT Press, 1987.