UVoD: An Unified Architecture for Video-on-Demand Services

Jack Y. B. Lee, *Member, IEEE*

***Abstract*— Existing video-on-demand (VoD) systems can be classified into two categories: true-VoD (TVoD) and near-VoD (NVoD). TVoD systems allocate a dedicated channel for every user to achieve short latency. NVoD systems make use of multicast technologies to enable multiple users to share a single channel to reduce system cost. This paper proposes a VoD architecture called UVoD that unifies the existing TVoD and NVoD architectures by integrating unicast with multicast transmissions. A performance model of the system is derived and numerical results show that one can achieve significant performance gain over TVoD (over 500%) under the same latency constraints.**

***Index Terms*—NVoD, TVoD, unified architecture, UVoD, video-on-demand.**

I. Introduction

THE KEY problem in deploying large-scale video-on-demand (VoD) applications today is economy rather than technology. To provide a true-VoD (TVoD) service where the user can watch any movie at any time, the system must reserve dedicated video channel at the video server and the distribution network for each user. As high-quality video consumes huge amount of storage space and transmission bandwidth even after compression, the cost in providing such TVoD service is often prohibitive.

On the other hand, near-VoD (NVoD) does find many successful applications such as pay-movies in hotel and cable TV. NVoD makes use of broadcast or multicast technologies to enable multiple users to share a single video channel to reduce system cost substantially. The tradeoffs are limited video selections, fixed playback schedule, and limited or no interactive control.

TVoD systems can be considered as one extreme where service quality is maximized, while NVoD systems as the other extreme where system cost is minimized. This paper proposes a VoD architecture called UVoD that unifies the existing TVoD and NVoD architectures by integrating unicast with multicast transmissions. Moreover, the proposed architecture significantly outperforms TVoD in terms of capacity even under the same latency constraints.

Manuscript received April 13, 1999. The asscociate editor coordinating the review of this letter and approving it for publication was Prof. V. S. Frost. The work described in this paper was supported in part by a grant from the Research Council of the Hong Kong Special Administrative Region, China.

The author is with the Department of Information Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong (email: jacklee@computer.org).

Publisher Item Identifier S 1089-7798(99)07358-5.

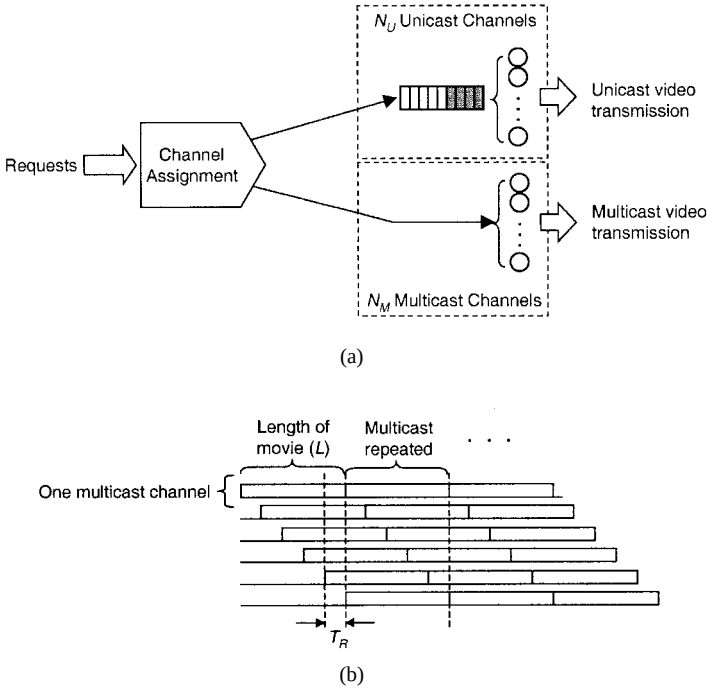

Fig. 1. Architecture of the UVoD System.

II. Architecture

Fig. 1(a) depicts the architecture of the proposed UVoD system. There are totally N channels, of which N_U are unicast and N_M are multicast channels. Let there be M movies of length L seconds each. We assume that $N_M \geq M$ so that each movie can be multicasted using at least one channel. For each channel, the assigned movie is repeated over and over the same way as in a NVoD system. We divide the N_M multicast channels equally among those M movies so that each movie is multicasted over $\lfloor N_M/M \rfloor$ channels. For channels multicasting the same movie, each channel is offset by

$$T_R = \frac{L}{\lfloor N_M/M \rfloor} \tag{1}$$

seconds as shown in Fig. 1(b). The N_U unicast channels share the same request queue and serve incoming requests in the first-come-first-serve manner.

We assume that the video clients and the client-side network are capable of receiving two video channels simultaneously and the client devices have additional storage to cache video data for later playback. As a multicast channel can be shared by multiple clients, our goal is to eventually serve users using multicast channels as in NVoD systems. We make use of the

unicast channels and client storage to migrate the client from a unicast channel to a multicast channel to achieve short latency.

A. Admission Control

When a request arrives at time t, the system first checks the multicast channels for the nearest upcoming multicast of the requested movie. Let t_m be the time for the nearest upcoming multicast, then the request will be assigned to wait for the upcoming multicast at time t_m if the waiting time is smaller than a predetermined admission threshold $delta$ as follows:

$$(t_m - t) \leq \delta \tag{2}$$

Otherwise, the request will be assigned to wait for a free unicast channel to start playback. The admission threshold is introduced to reduce the load of the unicast channels. In particular, increasing the threshold reduces the proportion of incoming requests that are routed to the unicast channels and vice versa.

For a new video session starting playback with a unicast channel, the client not only needs to receive video data from the unicast channel, but also concurrently receive video data from the nearest past multicast channel of the requested video. For example, let t_{m-1} and t_m be the starting times of channel $m-1$ and channel m, which are the two multicast channels closest to the request arrival time t where $t_{m-1} < t < (t_m - \delta)$. Then at time t, the client starts receiving video data from multicast channel $m-1$ and caches the data into the client's local storage. At the same time, the client starts video playback as soon as a free unicast channel becomes available. Under this scenario, the latency—defined as the average time a client has to wait before video playback starts—just equals to the waiting time at the unicast channels' queue.

As the client caches video data for the movie starting from time $(t - t_{m-1})$ from multicast channel $m-1$, the unicast channel can be released after a time $(t - t_{m-1})$ and the client will continue video playback from the local cache thereafter. In the worst case, the client needs a cache that can store up to T_R seconds of video. As $0 < t(t_{m-1}) < (T_R - \delta) \ll L$, we can see that the unicast channels are occupied for much shorter duration [average $(T_R - \delta)/2$ s] than in TVoD systems (L s).

B. Channel Allocation

Intuitively, one could vary the number of multicast channels N_M from zero (which reduces to TVoD) to N (which reduces to NVoD) to achieve various degrees of latencies. Given δ, it can be shown that the optimal number of multicast channels is given by

$$N_M^{\text{opt}} = \left\langle \frac{LN}{2LM - \delta N} \right\rangle \frac{M}{N} \tag{3}$$

where the operator $\langle\,\rangle$ rounds the input to the nearest integer.

However, the latency experienced by a user may not be the same and will depend on whether the user is assigned a multicast or unicast channel to start playback. To see why, we note that the latency for starting with unicast is simply the queueing delay while the latency for starting with multicast is equal to

$$D_M = \delta/2 \tag{4}$$

or half of the admission threshold assuming request arrivals are equally probable at any time.

These two latencies are in general different. To tackle this problem, we can model the unicast channels as a G/G/m queue and use the Allen–Cunneen approximation [1] to obtain the average wait (i.e., latency).[1] Then we can configure δ such that the latency is the same for starting with both types of channels.

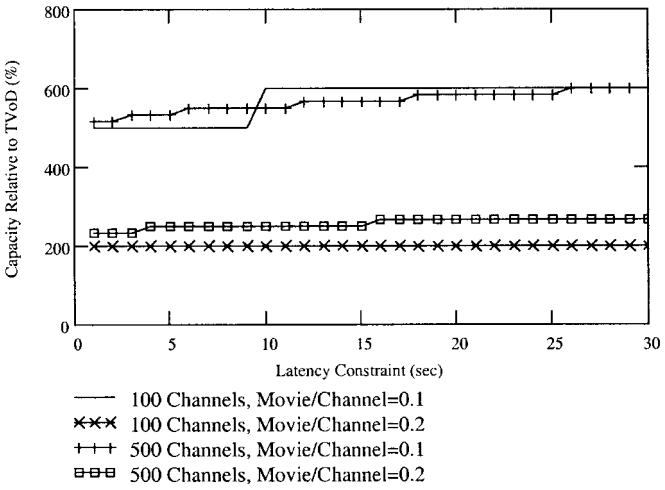

Fig. 2. Performance gain over TVoD versus latency constraint.

III. Performance Evaluation

We computed numerical results from the derived performance model to evaluate the proposed UVoD architecture. We assume that the request arrival process is Poisson, and movie length is 120 min. To compare UVoD with TVoD, we plot the performance gain of UVoD over TVoD versus latency constraint in Fig. 2. The performance gain is computed from the ratio of arrival rates that can be supported under UVoD and TVoD within the given latency constraint. Clearly, the results show that with the same number of channels and latency constraint, UVoD significantly outperforms TVoD. Moreover, we can achieve very good performance gains (e.g., 500% for 500-channels, 50-movies system) with very small latencies (e.g., 2 s). The performance gain continue to increase for larger latencies (not plotted) and the system ultimately reduces to a NVoD system when the latency reaches $0.5T_R$ (i.e., $\delta = T_R$), which can then support an unlimited number of users. We also observe that the performance gain depends primary on the movie-to-channel ratio rather than the scale of the system. This suggests that UVoD can be applied to systems of any scale as long as the movie-to-channel ratio is small (e.g., 0.1).

[1] The detail derivations are omitted due to space limitation.

IV. Related Works

Other researchers [2]–[8] have also investigated the use of multicast delivery to improve VoD system efficiency. UVoD differs from the existing studies in three major ways. First, instead of using the unicast channels only to support interactive control, UVoD employs them to reduce the startup latency experienced by new video session. Second, existing batching approaches incur a substantial amount of delay during session startup (in minutes) to achieve good performance gain. By contrast, UVoD can achieve significant performance gain with latencies as small as a few seconds. This enables UVoD to provide service qualities comparable to TVoD systems. Last but not least, an analytical performance model is obtained for the proposed UVoD architecture, whereas existing studies are primary based on simulation. Our simulation results have shown that the presented performance model is reasonably accurate and hence can facilitate system dimensioning.

V. Conclusions

This paper proposes a VoD architecture that unifies the existing TVoD and NVoD architectures as well as achieves significant performance gain over TVoD. It should be pointed out that the existing architecture requires a relatively small movie-to-channel ratio (0.1) to achieve good performance gain. The author is currently investigating partitioning schemes to cater for applications with a large number of video selections. Secondly, we have ignored interactive VCR-like controls in this paper but it turns out that pause-resume, which is the most common control in movie applications, can be supported without any overhead in UVoD by channel hopping. Interested readers are also referred to [8] for another method to provide interactive controls by client buffering. Apart from these limitations, the proposed UVoD architecture does achieve significant performance gain over TVoD under the same constraints.

Acknowledgment

The authors would like to thank the reviewers for their constructive comments in improving this paper to its final form.

References

[1] A. O. Allen, *Probability, Statistics, and Queueing Theory with Computer Science Applications*, 2nd ed. New York: Academic, 1990.
[2] A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling policies for an on-demand video server with batching," in *Proc. 2nd ACM Int. Conf. on Multimedia*, 1994, pp. 15–23.
[3] H. Shachnai and P. S. Yu, "Exploring waiting tolerance in effective batching for video-on-demand scheduling," in *Proc. 8th Israeli Conference on Computer Systems and Software Engineering*, June 1997, pp. 67–76.
[4] S. W. Carter, D. D. E. Long, K. Makki, L. M. Ni, M. Singhal, and N. Pissinou, "Improving video-on-demand server efficiency through stream tapping," in *Proc. 6th Int. Conf. on Computer Communications and Networks*, Sept. 1997, pp. 200–207.
[5] W. Liao and V. O. K. Li, "The split and merge protocol for interactive video-on-demand," *IEEE Multimedia*, vol. 4, no. 4, pp. 51–62, 1997.
[6] H. K. Park and H. B. Ryou, "Multicast delivery for interactive video-on-demand service," in *Proc. 12th Int. Conf. on Information Networking*, Jan. 1998, pp. 46–50.
[7] E. L. Abram-Profeta and K. G. Shin, "Providing unrestricted VCR functions in multicast video-on-demand servers," in *Proc. IEEE Int. Conf. on Multimedia Computing and Systems*, July 1998, pp. 66–75.
[8] K. C. Almeroth and M. H. Ammar, "The use of multicast delivery to provide a scalable and interactive video-on-demand service," *IEEE J. Select. Areas Commun.*, vol. 14, pp. 1110–1122, Aug. 1996.