

IMPROVING UVOD SYSTEM EFFICIENCY WITH BATCHING

Vincent C. H. Lee, Jack Y. B. Lee
Department of Information Engineering
The Chinese University of Hong Kong
Shatin, NT, Hong Kong
E-mail: {chlee9, yblee}@ie.cuhk.edu.hk

Abstract: *A unified video-on-demand (UVoD) architecture generalizing the traditional true-video-on-demand (TVoD) and near-video-on-demand (NVoD) architectures is recently proposed. In a traditional video server, the available resources are divided into a number of video channels. Each user is allocated a dedicated channel for the entire viewing duration in a TVoD system and the user can perform interactive VCR controls. By contrast, multiple users share a multicast video channel in a NVoD system. This reduces the resource requirement at the expense of long startup latency and limited interactive controllability. The recently proposed UVoD architecture divides the available channels into unicast and multicast channels. By using intelligent client buffering, UVoD can achieve latency similar to TVoD while at the same time dramatically reduces the resource requirement. This paper investigates the integration of batching into the unicast channels to further reduce startup latency, especially at high loads. Simulation results show that the startup latency can be reduced up to 85% in a 100-channel, 10-movie system under heavy loads.*

KEYWORDS: *Video-on-demand, Batching, UVoD, TVoD, NVoD, Performance Evaluation*

INTRODUCTION

In true-video-on-demand (TVoD) systems, each user has its own dedicated channel for streaming video. This ensures a short startup latency and enables the user to perform interactive VCR control at any time. The downside of TVoD is resource requirement. Specifically, the video server has a limited number of channels for serving users. Once all these channels are occupied, arriving users will be denied service. Therefore the resource requirement is proportional to the desired system capacity.

Another type of VoD system, commonly called near-video-on-demand (NVoD) [1] system, repeatedly transmits video stream over multicast or broadcast channels. Users viewing the same video can then share the same multicast video channel instead of occupying separate, dedicated channels. Therefore the resource requirement in a NVoD system is independent of the desired system capacity. In theory, a NVoD system can serve any number of users, albeit at the expense of longer start-up latency and limited VCR controllability.

Lee [2] recently proposed a UVoD architecture where both unicast and multicast channels are employed for video delivery. By using intelligent client buffering and channel switching (see Section 1), a UVoD system can achieve latency similar to TVoD while at the same time dramatically reduces the resource requirement. This paper extends Lee’s study by applying batching technique [3] to the unicast channels in UVoD to further reduce system latency. We use simulation to evaluate and compare performances of the proposed algorithm with the original UVoD architecture. Results show that the latency can be substantially reduced (e.g. 85% in a 100-channel, 10-movie system) under heavy loads and the latency remains at a low level (within 66 seconds) even under extremely high load. This batched UVoD architecture opens a new way to provision near-TVoD service at a fraction of the cost.

The rest of the paper is organized as follows: Section 1 presents the original UVoD architecture; Section 2 presents principles of batching; Section 3 presents the proposed batched UVoD architecture; Section 4 presents performance results; and Section 5 concludes the paper.

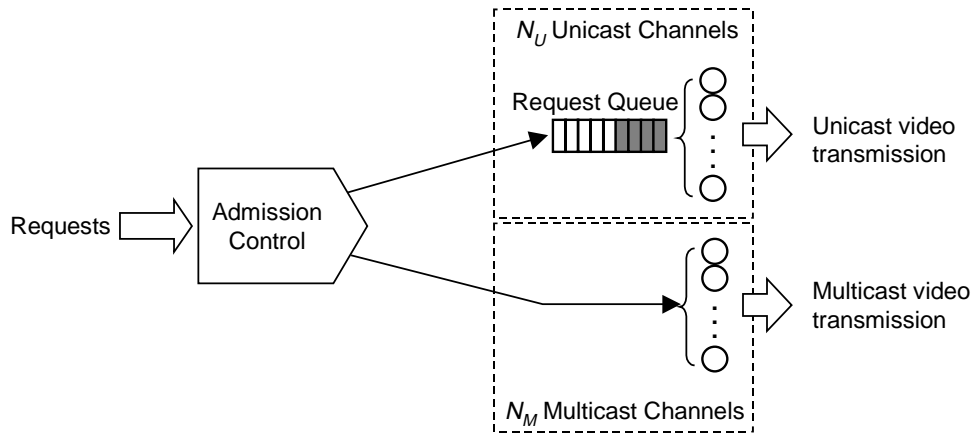


Figure 1 - Architecture of a UVoD system

1. UVOD ARCHITECTURE

Fig. 1 depicts the UVoD architecture. Let N be the total number of available channels, where N_u of them are unicast channels, $N_m = N - N_u$ are multicast channels. Let M be the number of movies of length L seconds each, and movies are assumed to have the same length for simplicity. The multicast channels are evenly assigned to all movies and the architecture requires $N_m > M$ such that each movie can be allocated with at least one multicast channel. Therefore, there are $\lfloor N_m/M \rfloor$ multicast channels allocated for each movie. The movie is then transmitted over all allocated multicast channels repeatedly as shown in Fig. 2.

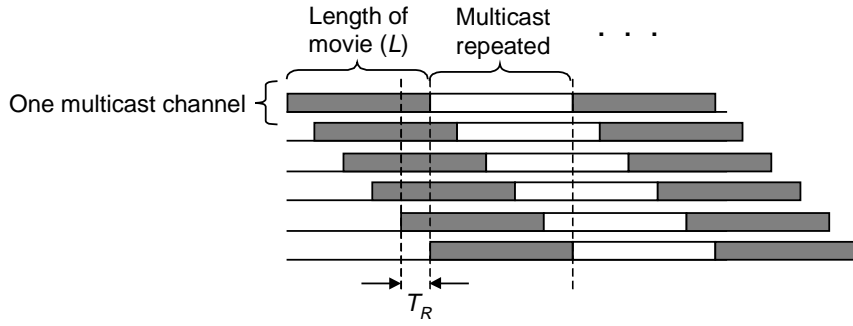


Figure 2 - Scheduling of multicast transmissions for a movie in UVoD

Note that transmission cycles are offset by

$$T_R = \frac{L}{\lfloor N_m/M \rfloor} \quad (1)$$

seconds between adjacent multicast channels allocated to the same movie.

When a user arrives at the system, the admission controller (Fig. 1) will assign the user to wait for either a unicast channel (admit-via-unicast) or a multicast channel (admit-via-multicast) to begin playback. The purpose of the admission controller is to equalize the average waiting time for users served initially by unicast channel and multicast channel [2]. Specifically, a parameter called admission threshold, denoted by δ , is introduced for admission control. Let t be the time a user arrives at the system requesting movie i , and let t_m be the start time of the next multicast cycle for movie i . The system will assign the user to wait for the upcoming multicast cycle if the resultant waiting time will be smaller than the admission threshold, i.e.

$$(t_m - t) \leq \delta \quad (2)$$

The admitted user continues to receive video data from this multicast channel for the entire video session as in a NVoD system. For simplicity, we do not consider interactive VCR controls in this study and the interested readers are referred to Lee [2] for ways to support interactive control in UVoD.

On the other hand, if the resultant waiting time is longer than the admission threshold, the admission controller will assign the user to wait for a free unicast channel to begin video playback. All unicast channels share a single input queue as shown in Fig. 1 and waiting users are served according to the first-come-first-serve (FCFS) queueing discipline. By adjusting the admission threshold, the system can maintain similar latency for both admit-via-multicast and admit-via-unicast users.

For admit-via-unicast users, the client device first starts caching video data from the previous multicast of the requested movie. Then it waits for a free unicast channel to start playback. For example, assuming that the user arrives at time t , and let t_{m-1} and t_m be the nearest epoch times of multicast channel $m-1$ and channel m , for which $t_{m-1} < t < (t_m - \delta)$. Then at time t , the client starts caching video data from channel $m-1$ into the client's local storage. At the same time, the client enters the request queue and starts video playback using unicast once a free unicast channel becomes available. Note that UVoD requires the client devices to receive up to two video channels simultaneously, and have local storage to cache up to T_r seconds of video data.

The admission process is not yet completed as the client still occupies one unicast channel. As the client concurrently caches multicasted video data for the movie starting from movie time $(t - t_{m-1})$, the unicast channel can be released after a time $(t - t_{m-1})$ and the client can continue video playback using the local cache. Hence similar to Liao *et al.* [4], the local cache is used to add time delay to the multicast video stream so that it can be synchronized with the client playback. Since $0 < (t - t_{m-1}) < (T_r - \delta) < L$, we can see that the unicast channels are occupied for much shorter duration when compared to TVoD. This reduction in service time allows more requests to be served by the unicast channels.

2. BATCHING

Batching is first proposed by Dan *et al.* [3] and later also investigated by Aggarwal *et al.* [5], Almeroth *et al.* [6] and Shachnai *et al.* [7]. The principle of batching is to group users waiting for the same movie at a video server and serve them by a single multicast channel. This batching technique can reduce resource requirement both at the server and in the network. However, the user cannot perform interactive VCR controls such as pause-resume, fast-forward, and fast-backward. To tackle this limitation, one can set aside some contingency channels to serve those users performing interactive controls as proposed and studied in Dan *et al.* [3], Almeroth *et al.* [6], and Li *et al.* [8].

There are several algorithms in which waiting users can be scheduled for service in batching. For example, in FCFS batching [3], arriving users all join a single queue. Once a free channel becomes available in the server, the user at the head of queue will be served. Moreover, other queued users with the same movie selection will also be served together by the same multicast channel.

Another algorithm called Maximum Queue Length (MQL) [3], maintains a separate queue for each movie for the arriving users. Once a free channel becomes available, the movie with the maximum number of waiting users will be selected for service. This algorithm can improve batching efficiency at the expense of fairness as users waiting for unpopular movies are likely to experience longer waiting time than users waiting for popular movies. There are other more sophisticated batching algorithms and the interested readers are referred to the study by Shachnai *et al.* [7] for details.

Reconsidering the UVoD architecture, we observe that the unicast channels within UVoD operates in the same way as a TVoD system. The only difference is that a unicast channel in UVoD is not occupied for the entire duration of the movie. The channel will be released once the user is merged back to a multicast channel. This motivates us to investigate applying batching to these unicast channels to further improve the system efficiency – batched UVoD.

3. BATCHED UVOD

In batched UVoD, the unicast channels are replaced by batching channels while the multicast channels remain the same. Specifically, using the same admission controller, the system will assign an arriving user to wait for a free batching channel (admit-via-batching) if the waiting time for the upcoming multicast cycle is longer than the admission threshold δ . Waiting users are then served according to the FCFS discipline when a free channel becomes available. Otherwise, the system will assign the user to wait for an upcoming multicast channel (admit-via-multicast) instead.

Unlike the original batching schemes, the channel-occupation time for a batching channel in UVoD is shorter than the movie length. To see why, consider a batching channel serving n admit-via-batching users. Let a_i be the time user i arrives at the system and the nearest multicast cycle starts at t_m and t_{m+1} respectively, where $t_m < a_1 < a_2 \dots < a_n < (t_{m+1} - \delta)$. Then user i will leave the batching channel to merge back to the multicast channel after a service time of $(a_i - t_m)$ as shown in Fig. 3. The batching channel can be released once all n users are merged back to the multicast channel. Therefore the holding time of the batching channel is simply the maximum of $(a_i - t_m)$, $i=1,2,\dots,n$.

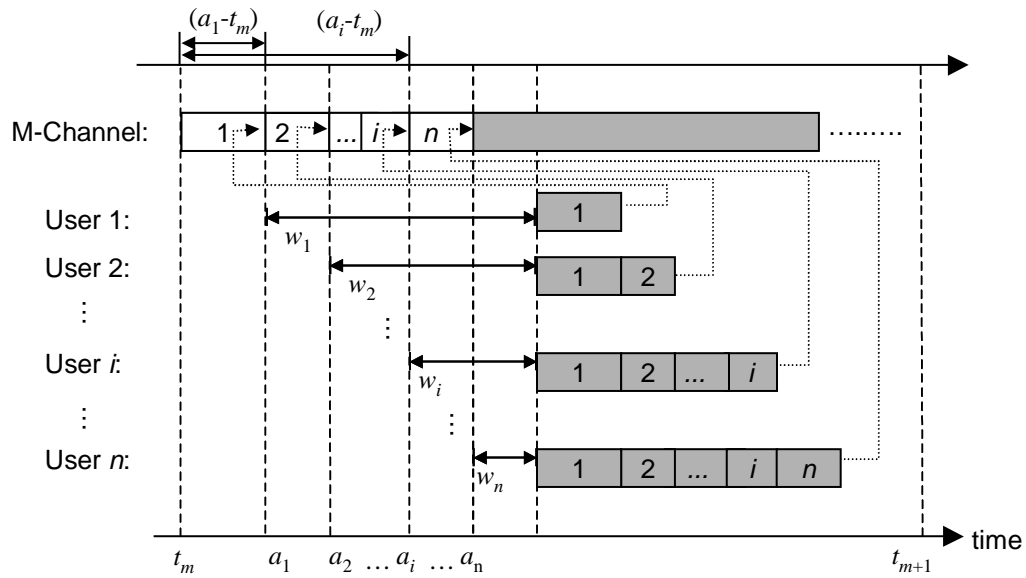
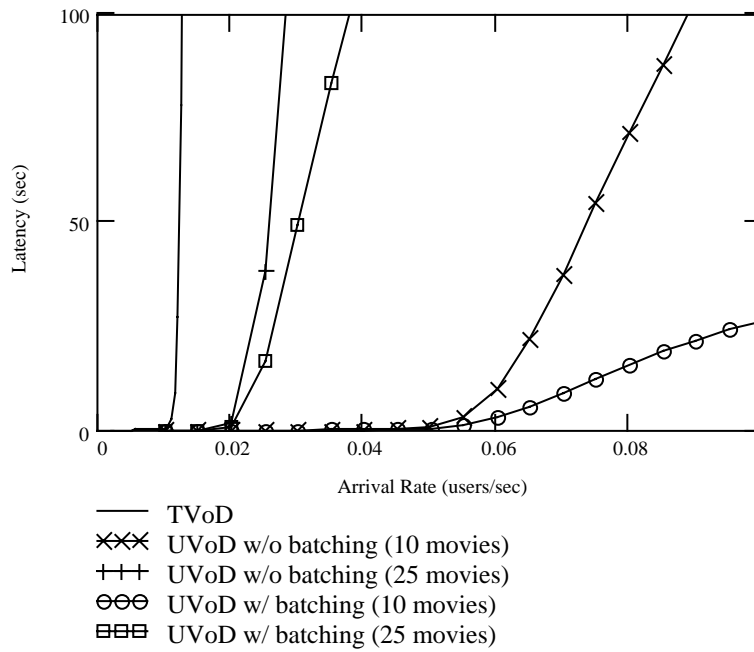


Figure 3 - Batching of multiple users in batched UVoD

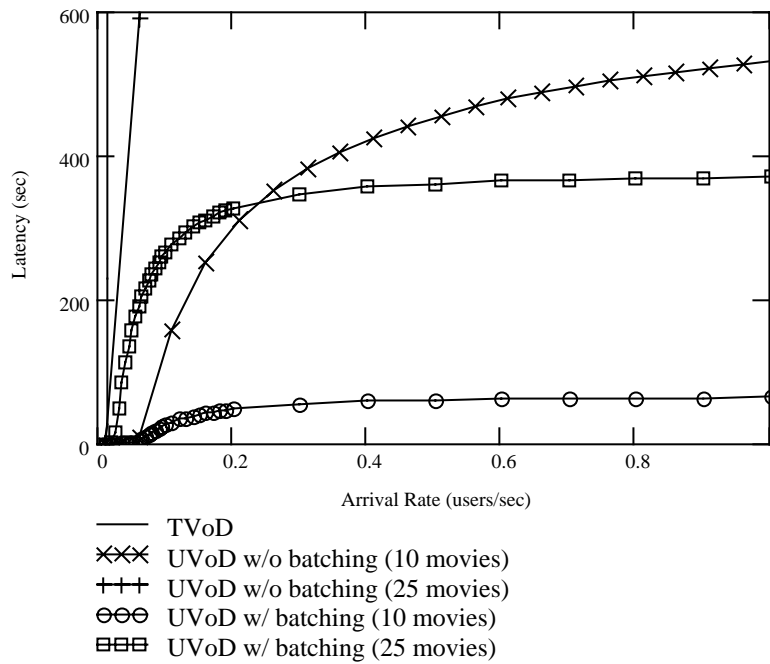
This batching algorithm enables a unicast channel in UVoD to serve multiple users instead of just a single user. As a channel is allocated to waiting users once it becomes available, there is no additional delay incurred to waiting users by using this batching algorithm. Similar to the original batching scheme, the effect of batching increases with more waiting users. Therefore we expect the performance to improve more under heavy load. In the next section we present detail simulation results to compare batched UVoD with non-batched UVoD.

4. PERFORMANCE EVALUATION

To study the performance of batched UVoD, we conduct simulations using CNCL [9] to quantify and compare batched UVoD with the original UVoD. Each run simulates a duration of 1440 hours (60 days) with the first 24 hours of data skipped to reduce initial condition effects. We use movie length of 120 minutes for all movies and assumes the movie popularity is Zipf distributed [10]. There are a total of 100 channels, where 50 are multicast channels and 50 are batching channels (unicast channels in case of UVoD). All 100 channels are unicast in case of TVoD. In this study, we do not consider user interactions and simply assume users all playback the entire movie from start to finish.



(a) Lighter load ranges (up to 0.1)



(b) Heavier load ranges (up to 1)

Figure 4 - Latency comparison of UVoD with and without batching

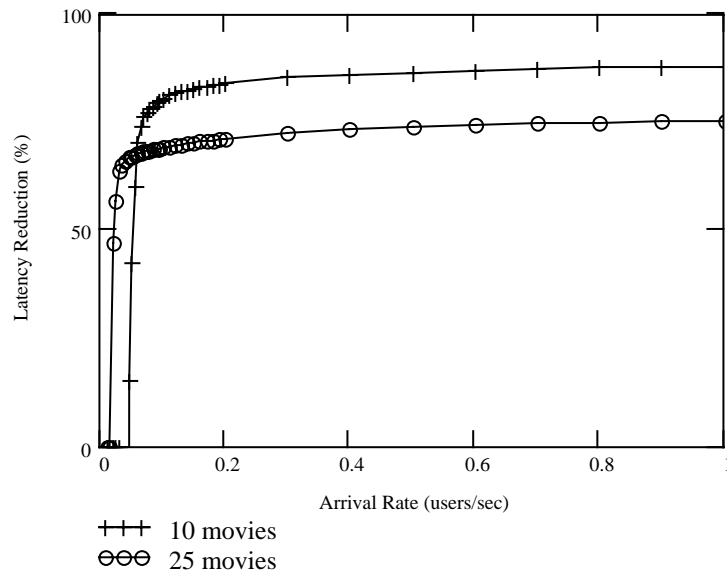


Figure 5 - Latency reduction by batched UVoD compared to Original UVoD

To compare the performance of UVoD with and without batching, we plot the latency versus arrival rate in Fig. 4. For lighter load (arrival rate up to 0.1 users per second) ranges in Fig. 4a, batched UVoD starts to outperform UVoD from a load of 0.04 and 0.025 users per second for the 10-movies and 25-movies configuration respectively. The latency reduction increases with the load. For example, at an arrival rate of 0.1 users per second, the latency of batched UVoD is only 26 seconds while that in the original UVoD is 132.4 seconds for the 10-movies configuration. Similar latency reduction is observed for the 25-movies configuration.

Fig. 4b shows the simulation result for heavier load ranges (up to 1 user per second). We observe that the latency of batched UVoD further increases with the load until at over 0.4 users per second where it levels off and approaches 66 seconds for the 10-movies configuration and 375 seconds for the 25-movies configuration. Comparing to the original UVoD, batched UVoD achieves a much shorter latency at heavy load (e.g. 65.8 seconds for batched UVoD versus 625.2 seconds for UVoD, for the 10-movies configuration).

Fig. 5 plots the normalized latency reduction versus arrival rate for the two architectures. For both 10-movies and 25-movies configurations, the trends are similar where reduction increase with the arrival rate and then levels off at heavy load. Latency reduction of up to 85% and 75% are observed for the 10-movies and 25-movies configuration respectively. On the other hand, latency reduction is insignificant at light load because the latency of UVoD in such light load (e.g. up to 0.05 users per second in 10-movies configuration) is almost equal to zero (0.8 seconds) and hence batching occurs only sparingly.

5. CONCLUSION

In this paper, we investigate the application of batching to the UVoD architecture. Specifically, the unicast channels in the original UVoD architecture are replaced by batching channels serving users according to the FCFS discipline. Similar to the original batching studies, we achieve further latency reduction with the new batched UVoD architecture. Results show that the latency reductions are substantial at heavy load and are insensitive to the movie-to-channel ratio. Similar to the original UVoD architecture, batched UVoD can also cope with infinite load while the resultant latency levels off to a much smaller value. This batched UVoD architecture opens a new way to provision near-TVoD service at a fraction of the cost compared to TVoD.

REFERENCES

- [1] S.L. Tsao and Y.M. Huang: "An efficient storage server in near video-on-demand system", *IEEE Transactions on Consumer Electronics*, vol.44 (1), 1998, p.p.27–32.
- [2] J.Y.B Lee: "UVoD – A Unified Architecture for Video-on-Demand Services", *IEEE Communications Letters*, vol.3, 1999, p.p.277–279.
- [3] A. Dan, D. Sitaram, and P. Shahabuddin: "Dynamic batching policies for an on-demand video server", *ACM Multimedia Systems*, vol.4, 1996, p.p.112–121.
- [4] W. Liao and V.O.K. Li: "The split and Merge protocol for interactive video-on-demand", *IEEE Multimedia*, vol.4(4), 1997, p.p.51-62.
- [5] C.C. Aggarwal, J.L. Wolf, and P.S. Yu: "On optimal batching policies for video-on-demand storage servers", *Proc. IEEE 3rd International Conference on Multimedia Computing and Systems*, June 1996, p.p.253-258.
- [6] K. C. Almeroth and M.H. Ammar: "The use of Multicast Delivery to Provide a Scalable and Interactive Video-on-Demand Service", *IEEE Journal of Selected Areas in Communications*, vol.14(6), Aug 1996, p.p.1110-1122.
- [7] H. Shachnai and P.S. Yu: "Exploring Waiting Tolerance in Effective Batching for Video-on-Demand Scheduling", *Proc. 8th Israeli Conference on Computer Systems and Software Engineering*, Jun 1997, p.p.67-76.
- [8] V.O.K. Li, W. Liao, X. Qiu, and E.W.M. Wong: "Performance Model of Interactive Video-on-Demand Systems", *IEEE Journal of Selected Areas in Communications*, vol.14(6), Aug 1996, p.p.1099-1109.
- [9] ComNets Class Library and Tools: <http://www.comnets.rwth-aachen.de/doc/cncl.html>.
- [10] G. Zipf, *Human Behaviour and the Principle of Least Effort*. Addison-Wesley, 1994.