Stream-Bundling Approach to Provide Video Services Over Broadband Networks

S.-H. Ivan Yeung[a] and S.-H. Gary Chan[b]

[a]Department of Electrical and Electronic Engineering,
The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong
[b]Department of Computer Science, The Hong Kong University of Science and Technology,
Clear Water Bay, Kowloon, Hong Kong

ABSTRACT

In traditional near-VOD (NVOD), the number of streams required is still high if the user delay goal is low (say, 2 minutes). In this paper, we study the use of client buffering to reduce the bandwidth requirement given a certain maximum user delay goal. Driven by the observation that broadband networks to the home has become a reality, we propose a broadcasting scheme termed "stream-bundling," which proves to be effective for popular movies. The scheme groups (i.e., "bundles") the server streams into channels of incrementally increasing bandwidth. Such high-speed bundled channels are used to deliver the beginning segment of a video to the client so that it can merge with an on-going multicast stream quickly. By comparing with other previously proposed broadcasting schemes such as Pyramid Broadcasting, Skyscraper Broadcasting and Harmonic Broadcasting, stream-bundling is shown to achieve similar level of performance with much lower complexity (without many channels to manage and to hop). We then obtain the minimum request rate of a movie under which such stream-bundling technique should be used over the one-stream-per-request approach. By considering a video system, we show that the bandwidth requirement of a system can be reduced significantly as compared with the traditional NVOD (by more than 50%), with the cost of only little client buffering ($\leq$ 20% of the movie length).

Keywords: Video service, client buffering, multicasting, data broadcasting, broadband networks

1. INTRODUCTION

Advances in computing technology and high-speed networking has made the delivery of on-demand video service (video-on-demand, VOD) to the home a reality.[1] In true-VOD, each user is assigned with its own dedicated unicast stream.[2] Such a technique is appropriate for those unpopular movies. For those more popular movies with many concurrent users, a more scalable and cost-effective technique is to use multicasting, whereby a single multicast stream is used to serve multiple requests at the same time. Such technique is referred to as near-VOD (NVOD) in previous literatures.[3–9] Clearly in NVOD, users cannot enjoy flexible interactivity with the videos. Furthermore, users have to wait for a certain amount of time before they are served. In order to offer a good quality of service, such user delay should be kept below a certain value. In this paper, we consider the design of video systems with a maximum user delay D_{max} while minimizing (server or network) bandwidth requirement.

Though multicasting technique is effective, NVOD still requires many streams if D_{max} is low. Consider a movie of length T_h minutes. The movie can be played in a staggered manner once every D_{max} minutes to satisfy the delay goal. The total number of streams required is then $\lceil T_h/D_{max} \rceil$, which increases when D_{max} decreases. The bandwidth requirement can be reduced with some cost of client buffer. Using the buffer, a client can prefetch future video segments from other bandwidth channels.* Note that a client can concurrently buffer data from multiple channels while playing its own stream in this system.

Traditionally, the aforementioned client buffering technique is usually applied for those hot movies and based on *streaming approach*, in which movies are delivered to the clients in a streaming fashion. An example of which is the pyramid broadcasting for hot movies, which is discussed by Viswanathan et al.[10] The scheme, however, requires

Send correspondence to S.-H. Gary Chan: E-mail: gchan@cs.ust.hk

*We distinguish "channel" from "stream" here: a channel is associated with an arbitrary bandwidth while a stream is associated with a bandwidth equal to the streaming rate of a movie.

a large client buffer size (close to the movie size). Another scheme, the harmonic broadcasting,[11] achieves a lower buffer requirement at the cost of complexity. Another scheme, skyscraper broadcasting,[12] achieves lower complexity at the cost of some bandwidth and buffer requirement. Other schemes based on streaming approach with client buffering are also discussed previously.[13–19] A client in all of these schemes has to manage two or more channels at the same time, and hop and join specific channels at specific time and for a specific duration. This leads to the complexity of the schemes. Moreover, the previous schemes are generally discussed in the context of "hot" or popular movies without quantifying the request rate under which a movie can be regarded as "hot," and specifying how a system should operate when there are some less "hot" movies.

In this paper, we consider client buffering techniques for video services. In particular, we address, given the movie request rate of a video system, how the movies should be delivered to minimize (server and network) bandwidth requirement. Certainly, when the arrival rate is "low," we are justified to use true-VOD, and for "high" arrival rate (those "hot" movies), we should use some kind of broadcasting scheme. We propose here a simple broadcasting scheme termed "stream-bundling." The design of such a scheme is based on the observation that broadband networks such as fiber or cable networks (of bandwidth beyond the streaming rate) to the home has become a reality. As a result, previous video broadcasting schemes based on the streaming approach does not necessarily have to be applied. In fact, we show that a much simpler system can be designed by grouping or "bundling" a number of server streams into channels with incrementally increasing bandwidth, and delivering movie segments using these channels. A client in our *stream-bundling* scheme only needs to join two channels for a deterministic D_{max} minutes at the start of a movie, and for the rest of time it needs only one stream. Since there is no channel hopping, the scheme is simple in design and implementation. Delivering videos at a higher rate was also discussed by Poon et al.[20] We consider a different technique here and use channels with incrementally increasing speed, as opposed to channels of a fixed streaming rate presented there.

We study the performance of stream-bundling broadcasting in terms of its buffer and bandwidth requirements. We quantify under what arrival rate the scheme should be used over true-VOD. By considering an M-movie system, we illustrate how this approach can be used to achieve minimum server bandwidth requirement with little client buffering. Our contributions in this paper is hence three-fold: i) we propose a "stream-bundling" broadcasting approach which is suitable for today's high-speed networks; ii) we analyze and optimize the scheme in terms of its bandwidth and buffer requirements; and iii) given movie request rates, we illustrate how the scheme can be used jointly with true-VOD in a video system to achieve minimum server bandwidth requirement.

This paper is organized as follows. We first discuss some previous work in Sect. 2. In Sect. 3, we describe and analyze stream-bundling broadcasting scheme. In Sect. 4, we present some illustrative examples and comparisons between the scheme and some other traditional broadcasting schemes. We then present in Sect. 5 how a movie system can be designed combining the scheme with true-VOD to minimize the total bandwidth requirement. We conclude in Sect. 6.

2. PREVIOUS WORK

Most of the previous work on the use of client buffering in broadcasting popular videos are based on streaming approach. We describe in more detail three of the more representative schemes here, namely, the pyramid broadcasting,[10] the harmonic broadcasting,[11] and the skyscraper broadcasting,[12] which provide a basis for our comparisons of our stream-bundling approach.

In pyramid broadcasting, each video is divided into K segments, with the size of the ith segment being α times of the $(i-1)$th segment. The total server capacity is accordingly divided into K different streaming channels with the same bandwidth. The ith segment of all videos are delivered using the ith channel sequentially. Since the first segment is the smallest, it is broadcast more frequently, leading to a low user start-up delay. While receiving a segment, a client can prefetch other segments at the same time using its local buffer. Given a certain user maximum delay requirement, such broadcasting scheme is found to achieve low bandwidth requirement. However, the buffer requirement of this scheme is usually large (more than 75% of the movie length). The broadcasting scheme is also quite complicated in the sense that a client needs to monitor multiple channels closely and prefetches the data at the right time in order not to miss the segments that it needs.

In harmonic broadcasting, each movie is divided into N segments of equal size of S minutes. Let S_i be the size of the ith segment (in minutes), and hence the whole movie length is $T_h = S_1 \cdot S_2 \cdot \ldots \cdot S_N$, where $(\cdot)$ is concatenation. The ith segment is further sub-divided into i sub-segments of equal size. For segment i, a stream is sub-divided into

i bandwidth channels (sub-streams) with bandwidth b_0/i, where b_0 is the bit rate of a stream. The sub-segments take turn in using the channel much similar to a TDM system, and the schedules of the sub-streams are staggered. As a result, the number of concurrent streams used by a movie is $H_N = \sum_{n=1}^{N} \frac{1}{n}$. When a client begins accepting S_1, it also prefetches segments $S_2, \ldots, S_N$ into its buffer. In this scheme, we can reduce the user delay by increasing the number of segments of a movie. Unfortunately, this also increases the system complexity. Though the bandwidth requirement of this scheme is slightly higher than pyramid broadcasting, its buffer requirement is much lower (but still larger than 30% of the movie length). However, this scheme is difficult to be implemented, and the number of channels the client and server have to manage increases with N. A client also needs to download data from N different channels at the same time, hence complicating the I/O issues.

In skyscraper broadcasting, each video is also divided into K different fragments. The fragmentation method, however, is different and follows the following recursive function:

$$f(n) = \begin{cases} 1, & n = 1; \\ 2, & n = 2, 3; \\ \left(2 + 2\lfloor \frac{n}{2} \rfloor - n\right) f(n-1) + & \\ \left(1 + 2\lfloor \frac{n}{2} \rfloor - n\right) \times & \\ \left(1 + \lfloor \frac{n - 4\lfloor \frac{n}{2} \rfloor}{2} \rfloor\right), & \text{otherwise,} \end{cases}$$

where $f(n)$ is proportional to the size of the nth fragment (in minutes). For example, the size of the first segment is one unit; the second segment is two units; and so forth. There is a parameter W (called the "width") in the scheme which restricts the size of the largest segment (those segments larger than W is forced to be W). By adjusting W, the buffer and bandwidth requirement can be traded off with each other: A smaller W reduces the buffer requirement but increases the bandwidth requirement, and *vice versa* (when $W = 1$, the scheme reduces to the traditional NVOD system without client buffering).

Several other schemes have also been proposed, namely, permutation-based pyramid broadcasting,[13] fast broadcasting[15,16] and pagoda broadcasting.[17] All these schemes are based on similar approaches with differences in how videos are segmented and how these segments are scheduled. Since the performance of these schemes are similar to the aforementioned schemes, we only concentrate in pyramid, harmonic and skyscraper broadcasting in this paper.

All of the previous broadcasting schemes are based on using a server stream "as is" (or subdivide a stream as in the harmonic broadcasting). Since a video segment is transmitted in the channel with different frequency, a client has to keep track of the starting and ending time of each segment and buffer it accordingly. Consequently, it has to keep "hopping" from one channel to another at exact times. This leads to design and implementation complexity. Motivated by the availability of high-speed networking of today, we propose a much simpler scheme in which the server streams are aggregated (i.e., bundled) to form high-speed channels, and such channels are held only for a brief period of time at the start of the movie. As a result, there is no "channel" hopping, leading to much simpler design and implementation. We show that our scheme, as compared to the other schemes, does not compromise simplicity with performance.

Another scheme called "Patching" is proposed previously.[21–23] It is similar to our proposed scheme such that a patch stream is used to patch the missing portion of a movie to the user and a multicast stream is used to multicast the video stream to a number of clients. The difference of patching with our scheme is that the patch streams are slotted into some mini-slot and a maximum latency of the slot size exists in our scheme. Moreover, the main feature of patching is providing true-VOD using multicast, while our scheme concentrate on providing NVOD with easy implementation. Some early merging media-on-demand policies like hierarchical merging schemes are also proposed by Eager et al.[24]

3. SCHEME DESCRIPTION & ANALYSIS

3.1. Scheme description

In our scheme, since the operation of a movie should be independent to the others; therefore, we can discuss the schemes by considering a particular movie. Recall that b_0 bits/minute is the streaming rate of the movie and T_h minutes be its duration, and D_{max} minutes is the maximum start-up delay of the users in the system.

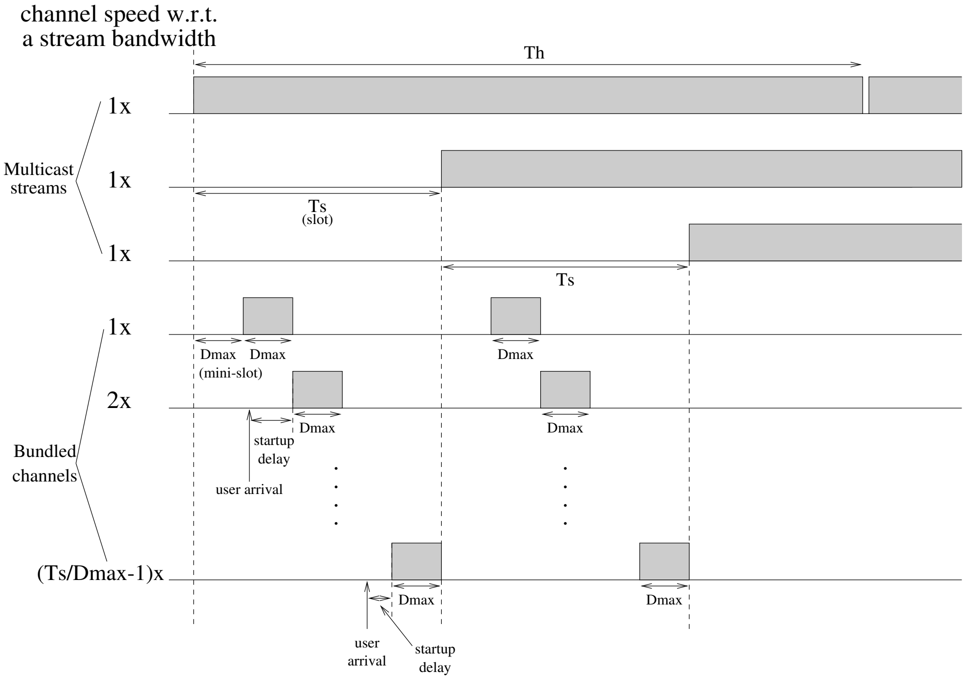

Figure 1. Operation of the stream-bundling broadcasting scheme.

The server streams are organized (i.e., bundled) into multicast/broadcast "channels" of increasing bandwidth with increment of b_0. These channels are used to push video to a client (and hence video data is not "streamed" over the channels). A "1×" channel is a channel with bandwidth b_0 (i.e., a stream), while a "2×" channel is a channel "bundled" with 2 streams (i.e., of bandwidth $2b_0$), and so on. We show in Fig. 1 the operation of the scheme. As in NVOD, the movie is multicast using multicast streams in a staggered fashion, with the offset time being T_s minutes (therefore, the number of multicast channels allocated for this purpose is given by $\lceil T_h/T_s \rceil$). The slot interval T_s is in general larger than D_{max}, and is further subdivided into "mini-slots" of length D_{max}. Each mini-slot is used by a specific bundled channel much similar to time-division-multiplexing to "download" the beginning portion of the movie, with the first mini-slot using 0× channel, the second one using 1× channel, the third one using 2×, and so on. Obviously, the number of bundled channels is given by $\lceil T_s/D_{max} \rceil$.

A user arriving into the system is served in the next mini-slot (and hence a maximum delay of D_{max}), at which it is served by the corresponding "high-speed" channel while concurrently buffering the on-going multicasting stream started at the beginning of the slot. Since the beginning portion of the video is downloaded with a rate at least as fast as the normal playback rate b_0, a user can start playing the movie right at the beginning of the mini-slot. With the use of the high speed channel and client buffering, a user can playback its movie from its own buffer at the end of the mini-slot and hence merges back to the multicast stream. From then on, the user just receives the multicast stream into one end of its buffer while playing it from the other end. (Note that we can as well use a 1× channel delivering for a time kD_{max} to replace the kD_{max} channel here to achieve the same effect.) To put the operation in a more formal way, a user arriving at a time between $(k-1)D_{max}$ and kD_{max} from the beginning of a slot ($1 \leq k \leq \lceil T_s/D_{max} \rceil$) will use a $k\times$ bundled channel for a duration of D_{max} minutes to download the first kD_{max} minutes of video segment while concurrently prefetching data from the normal multicast stream from time kD_{max} onwards. Clearly, we see that the buffer requirement of a client is not more than T_s minutes of video time. We note that in this scheme, a client needs to receive only two channels (one bundled channel and one multicast stream) for only a fixed duration of D_{max} at the beginning of the movie (while playing one stream). Clearly, there is no channel hopping in our scheme, which leads to the simplicity of the scheme.

3.2. Scheme analysis

We are interested in the following performance measures:

- The number of streams required for a movie, S: Here we are interested in the bandwidth used per movie in terms of the number of streams used; and
- The amount of client buffer in video minutes, B: Clearly, we are interested in a small B (e.g., $\leq 50\%$ of the movie length).

The number of streams required is the sum of the multicast streams ($= T_h/T_s$) and the streams in the bundled channels given by $\sum_{i=1}^{T_s/D_{max}-1} iD_{max}/T_s = (T_s/D_{max} - 1)/2$. Therefore,

$$S = \frac{T_h}{T_s} + \frac{T_s}{2D_{max}} - \frac{1}{2}. \tag{1}$$

Clearly, S achieves its minimum of

$$S^* = \sqrt{\frac{2T_h}{D_{max}}} - \frac{1}{2} \tag{2}$$

when

$$T_s^* = \sqrt{2D_{max}T_h}. \tag{3}$$

Note that for true-VOD, the (average) stream requirement is given by λT_h, where λ req/min is the request rate for the movie. Therefore, stream-bundling achieves lower bandwidth than true-VOD if $S^* \leq \lambda T_h$, i.e., when

$$\lambda \geq \sqrt{\frac{2}{T_h D_{max}}} - \frac{1}{2T_h} \equiv \lambda_{\text{CO}}. \tag{4}$$

This shows that when the request rate is higher than the "cross-over" rate λ_{CO} (those "hot" movies), we should use stream-bundling; otherwise, true-VOD is more cost-effective.

The client buffer should be at least large enough to buffer the data from the bundled channel and the multicast stream. The largest buffer requirement occurs for arrivals getting served at the last mini-slot in a slot, and therefore,

$$B = T_s - D_{max}. \tag{5}$$

Substituting T_s^* into Eq. (5), we have the buffer requirement of $B^* = \sqrt{2D_{max}T_h} - D_{max}$, which attains its maximum at $\hat{B} = T_h/2$ when $D_{max} = T_h/2$. Therefore, the buffer requirement would not be larger than 50% of the movie length.

4. ILLUSTRATIVE NUMERICAL EXAMPLES & COMPARISONS

We present in this section the performance of our proposed scheme and compare it with the other broadcasting schemes proposed previously, namely, the pyramid, harmonic and skyscraper schemes. We consider in the following a baseline system in which $T_h = 100$ minutes and $D_{max} = 2$ minutes.

We show in Fig. 2 the server streams used per movie S with respect to T_s. As T_s increases, S decreases quickly at the beginning and then rises gradually. This is expected since if T_s is low, the number of the multicast streams is high; while if T_s is high, many high-speed bundled channels (and hence streams) would be needed. The number of streams achieve its minimum at a certain value of T_s^* given by Eq. (3) ($T_s = 20$ minutes and $S^* = 9.5$ with the parameters used).

We show in Fig. 3 the trade-off between S^* and D_{max}, and the trade-off between B^* and D_{max}. Clearly, S^* decreases as D_{max} increases. As D_{max} increases, B^* (given by $T_s^* - D_{max}$) also increases. However, as mentioned before, B^* would not increase indefinitely, but reaches a maximum of 50% of movie length (not shown). Also shown is the NVOD case. Obviously, given a maximum user delay, the number of streams required in our scheme is much lower than NVOD.

We now compare our scheme with the other broadcasting schemes proposed previously (The parameters α and β for the pyramid broadcasting are chosen to be 2.7 and 20 respectively,[10] while the width W for skyscraper broadcasting is chosen to be 5 so that the buffer requirement is similar to that of stream-bundling). We show in Fig. 4 S^* versus D_{max} for these schemes. Clearly, S^* decreases with D_{max}, and compared with the NVOD case, all

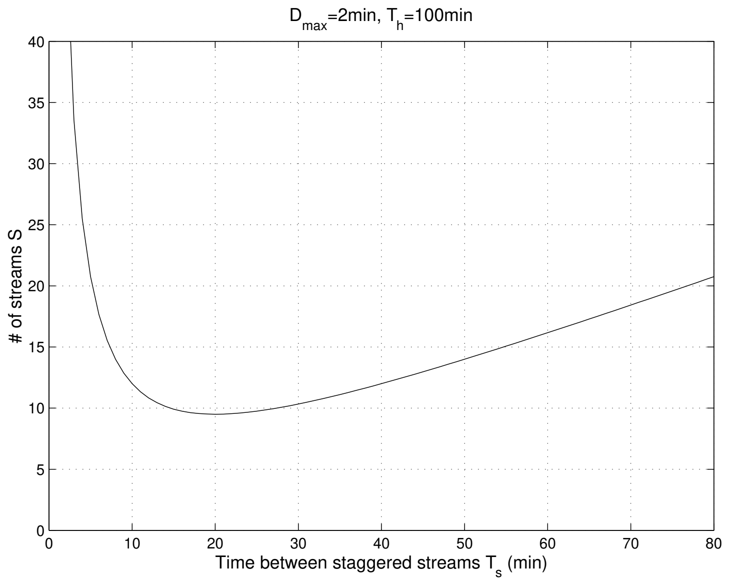

Figure 2. S vs. T_s for stream-bundling broadcasting.

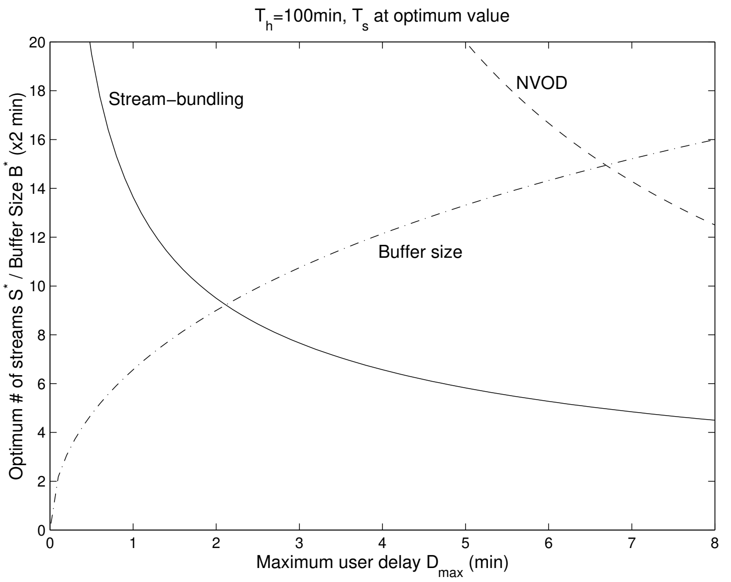

Figure 3. S^* vs. D_{max}, and B^* vs. D_{max} for stream-bundling.

the schemes achieve much lower bandwidth requirement at the cost of client buffer. Both pyramid and harmonic broadcasting consume similar bandwidth, while skyscraper consumes slightly higher bandwidth than ours (especially at low D_{max}).

We show in Fig. 5 the buffer requirement with respect to D_{max} of these schemes. Even though pyramid broadcasting has the lowest S^*, it requires the largest buffer that is close to the whole movie length. The buffer size for the harmonic scheme is relatively high and quite independent of D_{max}. Our scheme achieves low buffer requirement (with the maximum buffer requirement at 50% of the movie). The buffer requirement for the skyscraper broadcasting is proportional to the user delay, and depends on the parameter W.

From Figs. 4 and 5 we see that stream-bundling achieves similar trade-off between S^* and B^* as compared with the previously proposed schemes. However, it has the lowest complexity as the client does not need to hop from one

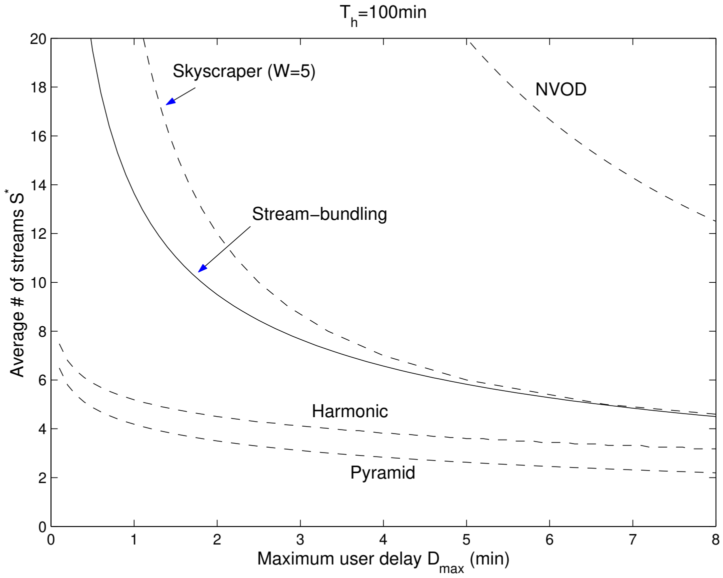

Figure 4. Comparison of S^* of different broadcasting schemes.

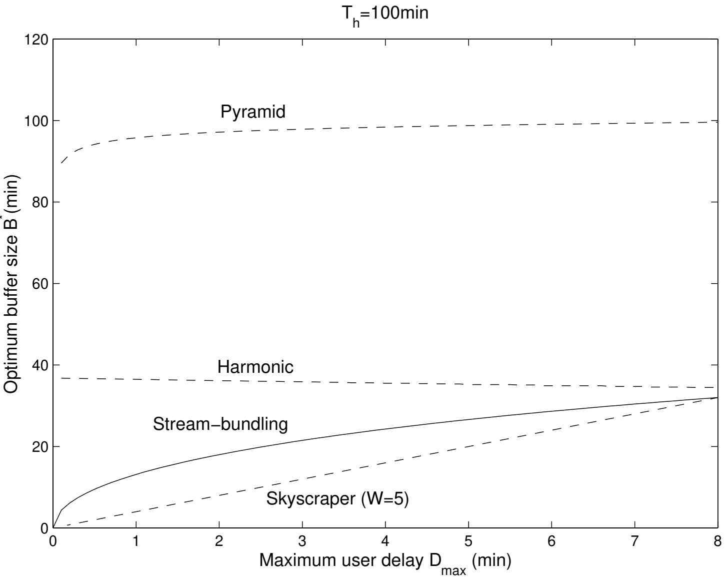

Figure 5. Comparison of B^* of different broadcasting schemes.

channel to another continuously at specified times as in the other schemes.

5. A HYBRID SYSTEM AND ITS DESIGN EXAMPLE

We show in Fig. 6 the number of streams required with respect to λ for stream-bundling and true-VOD. As expected, for very low arrival rate, true-VOD should be used as it requires lower number of streams (since stream-bundling has a constant multicast streams overhead), while stream-bundling is the best for high arrival rate. In order to minimize the bandwidth requirement, a movie should be served by one of the schemes depending on its request rate.

We show in Fig. 7 the crossover request rate, λ_{CO}, beyond which stream-bundling should be used in order to minimize the bandwidth requirement. As D_{max} increases, the crossover point decreases, showing that more movies are suitable for stream-bundling.

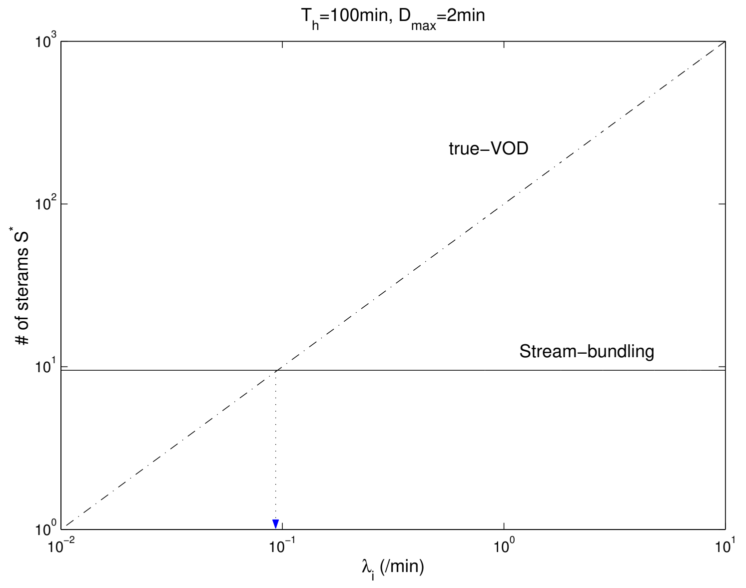

Figure 6. S^* vs. λ for true-VOD and stream-bundling broadcasting.

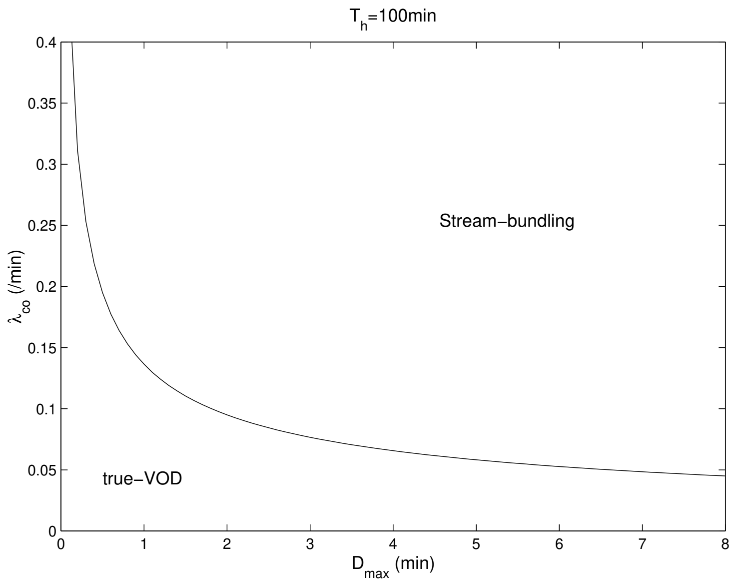

Figure 7. Crossover point between true-VOD and stream-bundling broadcasting.

We now present a design example of how our scheme can be used in a VOD system with a maximum user delay. We consider a system with $M = 100$ movies with $T_h = 100$ minutes and maximum user delay $D_{max} = 2$ minutes. The aggregate arrival rate to the system is Λ req/min, and the arrival rate for movie i, where $1 \leq i \leq M$, is λ_i. The popularity of the movies is according to a Zipf distribution, with $\lambda_i \propto 1/i^{\zeta}$, where $\zeta = 0.78$ (from a video store data[4]). The client has limited buffer size of 20 minutes (i.e., 20% of the movie length).

We design the system given λ_i so as to minimize the bandwidth requirement. To achieve so, recall from Fig. 6 that when λ_i is less than λ_{CO} (0.95 req/min in this case), true-VOD should be used; otherwise, stream-bundling would be used. This is the hybrid system we are studying here.

We show in Fig. 8 the performance of the hybrid system. When Λ is low, most λ_i are low and hence most of the movies are delivered using true-VOD. As Λ gradually increases, more and more movies are delivered by stream-

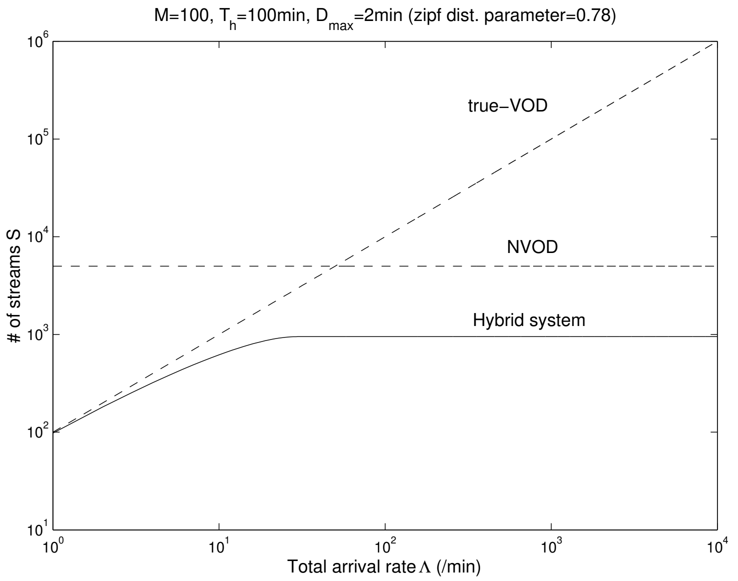

Figure 8. S^* vs. Λ for the hybrid system, true-VOD and near-VOD with $D_{max} = 2$ minutes.

bundling. The maximum number of streams is given by $M(\sqrt{2T_h/D_{max}} - 1/2)$, independent of Λ, when all of them are delivered by stream-bundling. Note that with only a small client buffer (less than 20 video minutes), the system achieves significantly lower bandwidth than the true-VOD, and achieves more than 50% reduction in bandwidth compared with the traditional NVOD case without client buffering (given by MT_h/D_{max}).

6. CONCLUSIONS

Advances in high-speed networking and computing technology have made video delivery to the home a reality. In traditional near-VOD, the number of streams in a server can still be high if the requirement in user delay is low. In this paper, we study a scheme termed "stream-bundling" broadcasting, which is suitable for high-speed networks and appropriate for movies with high request rate. In this scheme, the server streams are organized into channels with incrementally increasing bandwidth by grouping (i.e., bundling) a number of streams together. A movie is multicast in a staggered manner, while the "high-speed" bundled channels are used to deliver the beginning portion of the video so that they can merge with the multicast stream in short time. We show that such a technique can significantly reduce bandwidth requirement with little buffering (mostly less than 20% of the movie length) while achieving a low delay (less than 2 minutes). Our scheme, as compared with the other traditional broadcasting ones (such as pyramid, harmonic and skyscraper), achieves low complexity in terms of number of channels managed for each movie, the frequency of channel hopping, and the number of channels a client has to join at the same time.

We have analyzed the scheme and obtained the cross-over request rate beyond which such scheme can be used over true-VOD to achieve minimum bandwidth requirement. We show that by scheduling those "hot" movies according to this scheme, with the movies of low arrival rate scheduled according to true-VOD, the bandwidth requirement can be reduced by more than half as compared to the traditional near-VOD. This is achieved with only little cost in client buffering (less than 20% of the movie length).

ACKNOWLEDGMENTS

This work was supported, in part, by the Areas of Excellence (AoE) Scheme on Information Technology funded by the University Grant Council in Hong Kong, and by the Direct Allocation Grant of Hong Kong Research Grant Council.

REFERENCES

1. T. Little and D. Venkatesh, "Prospects for interactive video-on-demand," *IEEE Multimedia Magazine* , pp. 14–24, Fall 1994.
2. G. H. Petit, D. Deloddere, and W. Verbiest, "Bandwidth resource optimization in video-on-demand network architectures," in *Proceedings of the 1st IEEE International Workshop on Community Networking*, pp. 91–97, (NY, New York), July 1994.
3. V. O. K. Li and W. Liao, "Distributed multimedia systems," *Proceedings of the IEEE* **85**, pp. 1063–1108, July 1997.
4. A. Dan, D. Sitaram, and P. Shahabuddin, "Dynamic batching policies for an on-demand video server," *Multimedia Systems* **4**, pp. 112 – 121, June 1996.
5. V. O. K. Li, W. Liao, X. Qiu, and E. W. M. Wong, "Performance model of interactive video-on-demand systems," *IEEE Journal on Selected Areas in Communications* **14**, pp. 1099–1109, August 1996.
6. S.-H. G. Chan and F. A. Tobagi, "On achieving profit in providing near video-on-demand services," in *Proceedings of the 1999 IEEE International Conference on Communications (ICC'99)*, pp. 988–994, (Vancouver, Canada), June 1999.
7. S.-H. G. Chan, F. A. Tobagi, and T.-M. Ko, "Providing on-demand video services using request batching," in *Proceedings of the 1998 IEEE International Conference on Communications (ICC'98)*, pp. 1716–1722, (Atlanta, Georgia), June 1998.
8. S.-H. G. Chan, F. A. Tobagi, and T.-M. Ko, "Bandwidth planning in near video-on-demand," in *Proceedings of the 1998 IEEE International Symposium on Circuits and Systems (ISCAS'98)*, pp. 61–64, (Monterey, CA), June 1998.
9. A. D. Gelman and S. Halfin, "Analysis of resource sharing in information providing services," in *Proceedings of IEEE Globecom'90*, pp. 312–316, 1990.
10. S. Viswanathan and T. Imielinski, "Metropolitan area video-on-demand service using pyramid broadcasting," *Multimedia Systems* **4**, pp. 197–208, August 1996.
11. L.-S. Juhn and L.-M. Tseng, "Harmonic broadcasting for video-on-demand service," *IEEE Transactions on Broadcasting* **43**, pp. 268–271, Sept 1997.
12. K. A. Hua and S. Sheu, "Skyscraper broadcasting: a new broadcasting scheme for metropolitan video-on-demand systems," *ACM Computer Communication Review* **27**, pp. 89–100, Oct 1997.
13. C. Aggarwal, J. L. Wolf, and P. S. Yu, "On optimal piggybacking merging policies for video-on-demand systems," *Performance Evaluation Review* **24**, pp. 200–209, May 1996.
14. L. Gao, J. Kurose, and D. Towsley, "Efficient schemes for broadcasting popular videos," in *Proceedings of NOSSDAV'98*, (Cambridge, UK), July 1998.
15. L.-S. Juhn and L.-M. Tseng, "Fast broadcasting for hot video access," in *RTCSA'97*, pp. 237–243, Oct 1997.
16. L.-S. Juhn and L.-M. Tseng, "Fast data broadcasting and receiving scheme for popular video service," *IEEE Transactions on Broadcasting* **44**, pp. 100–105, Mar 1998.
17. J.-F. Pâris, S. W. Carter, and D. D. E. Long, "A hybrid broadcasting protocol for video on demand," in *Proceedings of the 1999 IS&T/SPIE Conference on Multimedia Computing and Networking*, pp. 317–326, (San Jose, CA), Jan 1999.
18. J. Y. B. Lee, "UVOD — a unified architecture for video-on-demand services," *IEEE Communication Letters* **3**, pp. 277–279, Sept 1999.
19. W. Liao and V. O. K. Li, "The split and merge protocol for interactive video-on-demand," *IEEE Multimedia* , pp. 51–62, October-December 1997.
20. W.-F. Poon and K.-T. Lo, "New batching policy for providing true video-on-demand (T-VOD) in multicast system," in *Proceedings of IEEE ICC*, pp. 983–987, (Vancouver, Canada), 6-10 June 1999.
21. K. A. Hua, Y. Cai, and S. Sheu, "Patching: A multicast technique for true video-on-demand services," in *Proceedings of ACM Multimedia 98*, pp. 191–200, (New York, USA), Sept 14-16 1998.
22. Y. Cai, K. A. Hua, and K. Vu, "Optimizing patching performance," in *Proceedings of Multimedia Computing and Networking (SPIE Vol. 3654)*, pp. 204–15, (San Jose, CA), Jan. 1999.
23. S. Sen, L. Gao, J. Rexford, and D. Towsley, "Optimal patching schemes for efficient multimedia streaming," in *Proceedings of NOSSDAV'99*, 1999.
24. D. Eager, M. Vernon, and J. Zahorjan, "Optimal and efficient merging schedules for video-on-demand servers," in *Proceedings of ACM Multimedia*, pp. 199–202, (New York, USA), 1999.