A Reactive Broadcasting Protocol for Video on Demand

Jehan-François Pâris[a], Steven W. Carter[b] and Darrell D. E. Long[b*]

[a]Department of Computer Science, University of Houston, Houston, TX 77204-3475

[b]Department of Computer Science, Jack Baskin School of Engineering
University of California, Santa Cruz, CA 95064

ABSTRACT

Broadcasting can reduce the cost of video on demand services by distributing in a more efficient fashion videos that are simultaneously watched by many viewers. Rather than waiting for individual requests, broadcasting schedules repeated broadcasts of the various segments of a video ahead of time. Hence it performs best when it is used to distribute videos that are watched concurrently by hundreds or thousands of viewers.

We propose a reactive broadcasting protocol that addresses the problem of distributing moderately popular videos in a more efficient fashion. Like all efficient broadcasting protocols, reactive broadcasting assumes that the customer set-top box has enough local storage to store at least one half of each video being watched. Unlike other broadcasting protocols, reactive broadcasting only broadcasts the later portions of each video. The initial segment of each video is distributed on demand using a stream tapping protocol.

Our simulations show that reactive broadcasting outperforms both conventional broadcasting protocols and pure stream tapping for a wide range of video request rates.

Keywords: video on demand, video broadcasting, stream tapping.

1. INTRODUCTION

Despite the appeal of the concept and its potential to revolutionize the whole home entertainment business, video on demand (VOD) has yet to succeed on the marketplace. The reason behind this situation is simple: as its name indicates, a VOD service lets its customers watch the video of their choice at the time of their choice. If each request were to be served in an independent fashion, the total bandwidth requirements of the service would be equal to the average video bandwidth times the number of concurrent users. Handling such bandwidths would require costly upgrades of the existing communication infrastructure and equally costly servers to handle all the necessary I/O traffic. As a result, VOD services are still too costly to compete with either video rentals or pay-per-view television.

Broadcasting is one of several techniques that aim at reducing these bandwidth requirements.[15] It only applies to videos that are likely to be simultaneously watched by many viewers. Rather than responding to individual requests, broadcasting uses a proactive approach and executes frequent retransmissions of these "hot" videos. The bandwidth savings can be considerable since the top ten or twenty most popular videos are likely to account for over forty percent of the total demand.[4-5]

What distinguishes broadcasting from other VOD distribution methods is that the number of viewers watching a given video does not affect its bandwidth requirements. Hence, broadcasting is especially suited to the distribution of very popular videos over a large metropolitan area. Conversely, methods that transmit videos on demand, such as *piggybacking*,[6] *bat*ching[5], and stream tapping,[2-3,8] will require less bandwidth to distribute the videos that are not in such heavy demand.

Here we address the problem of distributing in an efficient fashion the videos that fall between these two groups. These moderately popular videos will be typically watched by, say, five to fifteen simultaneous users. The solution we propose takes advantage of the fact that nearly all broadcasting protocols require much less bandwidth per minute to broadcast the later parts of a video than its first few minutes. Our *reactive broadcasting* protocol combines the reactive and the proactive approaches by partitioning each video into n segments of equal duration. We then distribute the first segment of each video

* Correspondence: Email: **paris@acm.org**, **carter@cse.ucsc.edu** and **darrell@cse.ucsc.edu**.
WWW: **http://csl.cse.ucsc.edu/projects/video-on-demand**

Part of the IS&T/SPIE Conference on Multimedia Computing and Networking 2000
San Jose, California • January 2000
SPIE Vol. 3969 • 0277-786X/00/$15.00

in a reactive fashion using Carter and Long's *stream tapping* protocol[2-3] while using a modified version of our *pagoda broadcasting* protocol[12] to distribute the $n-1$ remaining segments.

2. PREVIOUS WORK

Given the large number of video distribution techniques that have been discussed in the literature, we will focus our discussion on the two approaches that are combined in our reactive broadcasting protocol, namely broadcasting and stream tapping.

2.1 Broadcasting

The simplest video broadcasting protocol is *staggered broadcasting*.[15] A video broadcast under that protocol is continuously retransmitted over k distinct video channels at equal time intervals. The approach does not necessitate any significant modification to the set-top box (STB) but requires a fairly large number of channels per video to achieve a reasonable waiting time. Consider, for instance, a video that lasts two hours, which happens to be close to the average duration of a feature movie. Guaranteeing a maximum waiting time of five minutes would require starting a new instance of the video every five minutes for a total of 24 full-bandwidth streams.

The past three years have seen the development of many more efficient broadcasting protocols. All these better protocols assume that the client set-top box has enough local storage to store at least one half of each video being watched. We can subdivide these protocols into two groups. Protocols in the first group are all based on Viswanathan and Imielinski's *pyramid broadcasting* protocol.[14] They include Aggarwal, Wolf and Yu's *permutation-based pyramid broadcasting* protocol,[1] Hua and Sheu's *skyscraper broadcasting* protocol[7] and Juhn and Tseng's *fast broadcasting* protocol.[10]

While these protocols require less than half the bandwidth of staggered broadcasting to guarantee the same maximum waiting time, they cannot match the performance of the protocols based on the *harmonic broadcasting* (HB) protocol.[9, 11] These protocols divide videos into many equally sized segments and allocate a separate data stream to each segment. In addition, they require the STB to start to read all these streams when the customer starts watching the first segment of a video. As a result, each segment can be broadcast at the minimum bandwidth required to ensure on time delivery of its data.

Even though the total bandwidth requirements for HB and its variants are quite small, the multitude of streams these protocols involve complicates the task of the STB's and the servers. Like HB, *pagoda broadcasting*[12] uses fixed-size segments. Rather than lowering the bandwidth of the later segments of the video, pagoda broadcasting broadcasts them less frequently. Given a fixed number k of full bandwidth streams, pagoda broadcasting divides each stream into fixed-size *slots* of equal duration. It then tries to find the segment mapping that maximizes the number n of segments that can be packed into these k streams while satisfying the condition that segment S_i, for $1 \le i \le n$, is repeated at least once every i slots. For example, a segment mapping using three streams would be:

First Stream	S_1	S_1	S_1	S_1	S_1	S_1
Second Stream	S_2	S_4	S_2	S_5	S_2	S_4
Third Stream	S_3	S_6	S_8	S_3	S_7	S_9

Since the protocol can pack nine segments into the three streams, the segment size will be equal to one ninth of the duration of the video. Hence no client would ever have to wait more than 14 minutes for a two-hour video. The original pagoda broadcasting protocol used a fairly regular algorithm mapping segments to successive pairs of streams. A more recent version of the protocol, the *new pagoda broadcasting* protocol uses more complex segment to stream mappings and packs more segments into the same number of data streams to achieve even lower maximum waiting times.[13]

2.2 Stream Tapping

In order to use stream tapping[2-3] or its variants,[8] clients must have a small buffer on their STB. The buffer allows them to "tap" into streams of data on the VOD server originally created for other clients, and then store the data until it is needed. In the best case, clients can use all of the data from the existing stream, and greatly reduce the amount of time they need their own stream.

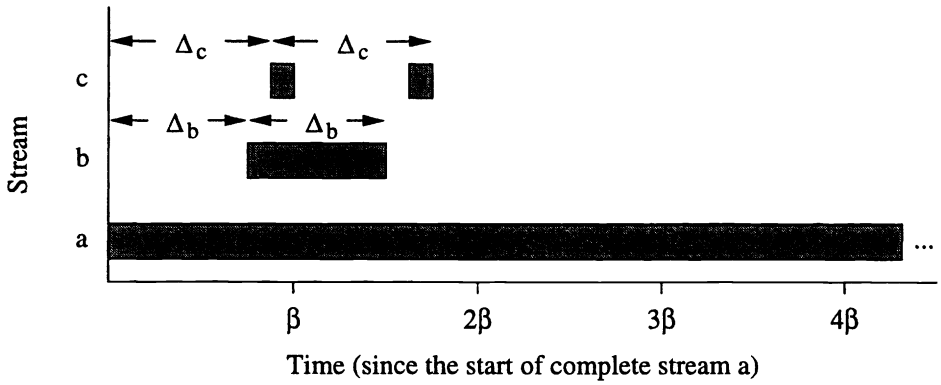

Figure 1: How stream tapping works

In particular, stream tapping defines three types of streams. *Complete streams* read out a video in its entirely. These are the streams clients typically tap from. *Full tap streams* can be used if a complete stream for the same video started $\Delta \leq \beta$ minutes in the past, where β is the size of the client buffer, measured in minutes of video data. In this case, the client can begin receiving the complete stream right away, storing the data in its buffer. Simultaneously, it can receive the full tap stream and use it to display the first Δ minutes of the video. After that, the client can consume directly from its buffer, which will then always contain a moving Δ-minute window of the video. Stream tapping also defines *partial tap streams*, which can be used when $\Delta > \beta$. In this case clients must go through cycles of filling up and then emptying their buffer since the buffer is not large enough to account for the complete difference in video position.

To use tap streams, clients only have to receive at most two streams at any one time. If they can actually handle a higher bandwidth than this, they can use an option to the protocol called *extra tapping*. Extra tapping allows clients to tap data from any stream on the VOD server, and not just from complete streams. Figure 1 shows some example streams from the VOD server's perspective. Stream *a* is a complete stream, and it must exist for the entirety of the video. Stream *b* is a full tap stream starting Δ_b minutes after stream *a*. It only has to exist for Δ_b minutes. Stream *c* is another full tap stream, but it is able to use extra tapping to tap data from stream *b*, and so its service time is much smaller than Δ_c.

3. THE REACTIVE BROADCASTING PROTOCOL

Looking back at all recent broadcasting protocols, we observe that they require much less bandwidth per minute to distribute the later segments of a video than its first segment. Consider, for instance, the case of pagoda broadcasting with three streams and recall that the protocol divides each stream into *slots* of duration $d = D/n$, where D is the duration of the video and n the number of segments. Since segment S_1 is retransmitted once every slot, one complete stream is dedicated to its transmission. Segment S_2, on the other hand, only needs to be retransmitted once every two slots and will only require half a stream. Later segments will require even less bandwidth.

Observe now what happens when the number of customer requests for the video is low and below an average of one or two customers per slot. From time to time there will be no customer consuming a particular instance of the segment S_1. Some bandwidth could be saved if this segment was sent *on demand* rather than according to a predefined schedule. Sending the first segment would have the additional benefit of providing instant access to the video instead of forcing the customer to wait on the average half a slot.

Sending on demand the other segments of the video, namely, segments S_2 to S_n, would not achieve similar savings because these segments require less bandwidth and each particular instance of a given segment is much less likely not to be consumed by at least one customer. We propose to continue to broadcast them according to a predefined schedule.

Given a video of duration D partitioned into n segments of equal duration $d = D/n$, our *reactive broadcasting* protocol with k broadcasting streams (RB-k) will:

a) distribute the first segment of each video *on demand*, that is, in a reactive fashion, and

b) transmit the remaining segments of the video on the k broadcasting streams according to a predefined schedule.

We will discuss in turn how these two functions are implemented.

We selected *stream tapping* to distribute the first segment of each video because it requires much less bandwidth than its alternatives.[2-3] The only modification we made to the protocol concerned the size β of the STB buffer. The original stream tapping protocol assumed a rather small buffer. Since the broadcasting protocol used to transmit segments S_2 to S_n assumes that the STB can store at least one half of each video being viewed, we assumed that the stream tapping process would never be limited by the size of the buffer. Hence, all stream taps would be full stream taps.

We selected a pagoda-like protocol to distribute the remaining segments of each video because these protocols do not require much more bandwidth than the best harmonic protocol and are much easier to implement. The only significant change we had to bring concerned the minimum frequencies at which each segment had to be broadcast. Since segment S_1 is now sent on demand, segment S_2 must be ready to be viewed after a time delay d. Transmitting the segment S_2 every two slots will not suffice because the customer STB would not always receive on time the beginning of S_2. The simplest way to guarantee that the STB will always receive the beginning of S_2 in time is to ensure that the STB will always receive the *whole segment* S_2 by the time the customer finishes watching the first segment of the video. Thus will have to broadcast S_2 at least once every slot instead of once every two slots. By induction, we can show that the same argument applies to all subsequent segments of the video. Hence the server will have to broadcast segment S_i with $1 < i \leq n$ once every $i-1$ slots.

Consider first the case of a reactive broadcasting protocol where only one single stream is used for broadcasting (RB-1). Since S_2 has to be retransmitted once every slot, it will require the whole stream and the video will have two segments. For a two-hour video, this would mean that we would distribute on demand the first hour of the video and have one channel continuously transmitting the second hour of the video.

Allocating two streams for broadcasting (RB-2) would allow us to partition the video into four segments using the segment mapping:

First Stream	S_2	S_2	S_2	S_2	S_2	S_2
Second Stream	S_3	S_4	S_3	S_4	S_3	S_4

For a two-hour video, this would mean that we would distribute on demand the first 30 minutes of the video and have two channels continuously rebroadcasting the three segments constituting the remaining 90 minutes of the video. Note that the mapping is not particularly efficient as segment S_4 is rebroadcast once every two slots while it only needs to be rebroadcast once every three slots. Allocating one extra stream for broadcasting (RB-3) would greatly improve the situation as we could now partition the video into ten segments using the segment mapping:

First Stream	S_2	S_2	S_2	S_2	S_2	S_2
Second Stream	S_3	S_5	S_3	S_6	S_3	S_5
Third Stream	S_4	S_7	S_9	S_4	S_8	S_{10}

Going back to our example, the protocol would distribute on demand the first 12 minutes of a two-hour video and use three channels to distribute the remaining 108 minutes of the video.

4. DISCUSSION

Comparing the bandwidth requirements of our reactive broadcasting protocol to those of a conventional broadcasting protocol presents one difficulty because reactive broadcasting provides instant access to the video while all other broadcasting require the customer to wait up to the duration of slot. We decided that waiting no more than a few seconds would be equivalent to a zero wait for all practical purposes. Hence we selected as benchmark the *new pagoda broadcasting* (NPB) protocol with seven streams as this configuration guarantees that no customer would have to wait more than 17 seconds for a two-hour video.[13]

Figure 2 compares the average bandwidth requirements of our reactive broadcasting protocol with those of stream tapping and NPB for request arrival rates varying between 0 and 200 customers per hour. The bandwidth values for stream tapping and the reactive part of our protocol were computed using the same discrete simulation model as in our previous papers on stream tapping.[2-3] We assumed a standard video duration of two hours. All bandwidths are expressed in full

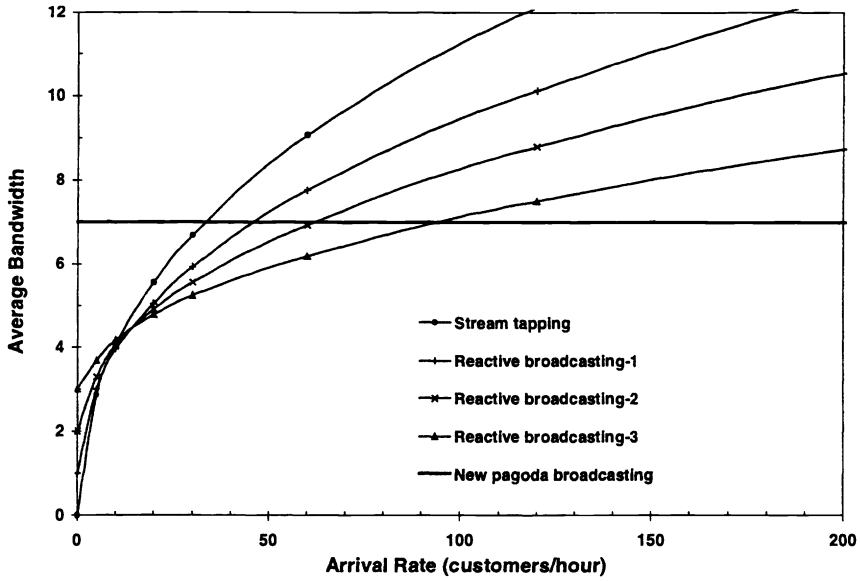

Figure 2: Comparing the average bandwidth requirements of reactive broadcasting with those of stream tapping and conventional broadcasting

bandwidth streams. The number of broadcasting streams used by each version of our reactive broadcasting protocol is identified in the corresponding graph label. For instance, the label *Reactive broadcasting-3* refers to the version of the reactive broadcasting protocol with 3 broadcasting streams (and 10 segments of 12 minutes each)

As one can see, all versions of the reactive broadcasting protocol require less bandwidth than conventional broadcasting as long as long as the customer arrival rate stays below 50 arrivals per hour or one customer every 72 seconds. The versions of the protocol using more broadcasting streams perform better than those using less broadcasting channels when the customer arrival rate is above 12 arrivals per hour. In particular, reactive broadcasting with 3 broadcasting streams (RB-3) outperforms the NPB protocol as long as the customer arrival rate remains below 92 customers per hour. While the NPB protocol outperforms the RB-3 protocol or higher arrival rates, RB-3 bandwidth requirements stay within 25 percent of those the NPB protocol as long as the customer arrival rate stays below 200 customer arrivals per hour.

We can distinguish three ranges of customer arrival rates:

a) *low customer arrival rates*, that is, customer arrival rates below 12 requests per hour, where stream tapping outperforms all other methods; reactive broadcasting is second best, beating NPB by a margin of at least 39 percent;

b) *medium customer arrival rates,* that is, customer arrival rates between 12 and 92 requests per hour, where RB-3 outperforms all other methods;

c) *high customer arrival rates,* that is, customer arrival rates above 92 requests per hour, where NPB outperforms all other methods and RB-3 is the second best.

One major advantage of reactive broadcasting over stream tapping and conventional broadcasting protocols is that its bandwidth requirements remain very reasonable all over the spectrum of customer arrival rates. Even though it is not always the best protocol, it is never the worst. This is not the case for either stream tapping, which does not handle very well high customer arrival rates or conventional broadcasting, which performs poorly for low customer arrival rates. Reactive broadcasting, especially in its RB-3 form, clearly constitutes the safest choice for all videos that are not known to be either extremely popular or very unlikely to be simultaneously watched by, say, more five to ten customers.

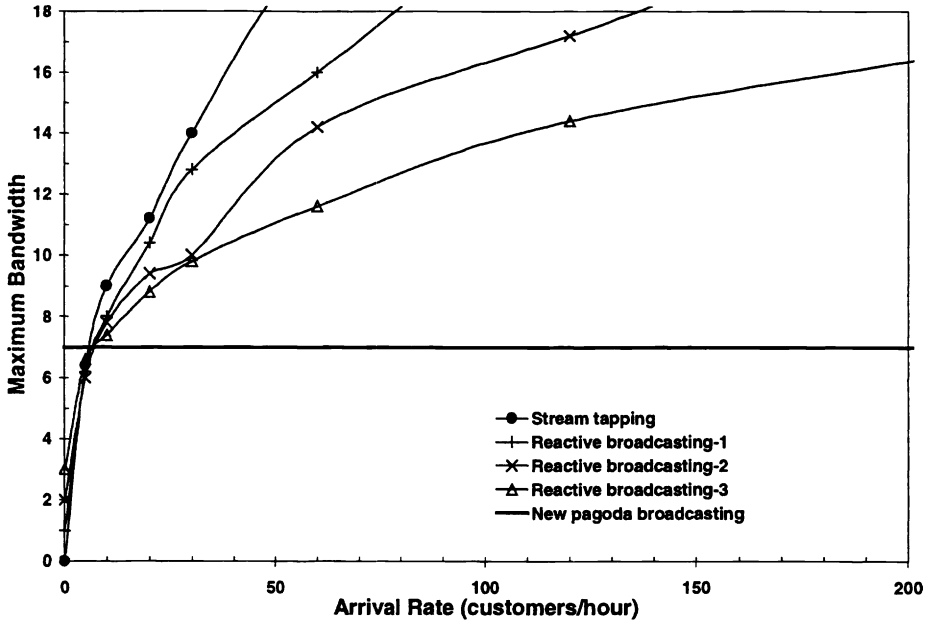

Figure 3: Comparing the maximum bandwidth requirements of reactive broadcasting with those of stream tapping and conventional broadcasting

The situation is quite different when we compare the maximum bandwidth requirements of the protocols. Like all conventional broadcasting protocols, NPB uses a predefined static schedule for all its broadcasts. Hence, its maximum bandwidth requirements coincide with its average bandwidth requirements. The same is not true for stream tapping and for reactive broadcasting. We observe that RB-3 performs the least poorly of the lot probably because 9 of the 10 segments making each video are transmitted using a static schedule. More work is still needed to devise methods limiting the bandwidth fluctuations inherent to all reactive distribution protocols for VOD. One option could be to increase the number of broadcasting streams to four or even five streams. This would increase the number of segments and hence reduce their size. Since the protocol would transmit less data on demand, it would be less affected by the unavoidable fluctuations in the actual request arrival rates. The sole drawback of the approach is that it would also reduce the potential benefits of using a reactive protocol.

5. POSSIBLE EXTENSIONS

In this section, we sketch two possible extensions to our reactive broadcasting protocol. Both extensions aim in different ways at reducing the bandwidth requirements of the protocol. Our first extension allows the customer STB's to snoop on the stream tapping traffic and preload the first segment of some of the videos being broadcast. A second extension provides better segment mappings by distributing on demand the two first segments of each video.

5.1 Reducing bandwidth consumption through snooping

As disk drive capacities have been doubling every year for the last three years, we expect that many customer STB's will have a significant amount of unused space. Any STB having enough free space could snoop on the stream tapping traffic until it encounters a complete segment S_1 and preload as many of these segments at it can. If the customer wants to watch one of the videos whose first segment has been preloaded, the STB can proceed without requesting a stream from the server. This would lower both the server workload and the network traffic.

Snooping will work best for versions of the reactive broadcasting protocol using at least three broadcasting streams as these versions have much smaller segments. For instance, an STB having enough free space to preload 60 minutes of video signal could preload the initial segments of 5 two-hour videos distributed through a RB-3 protocol. If the same videos were distributed through a RB-2 protocol, that STB could only preload the initial segments of 2 videos.

Note that the process can remain entirely optional. Some customers may elect not to participate in it. Any STB needing to reclaim some of the space occupied by these preloaded segments can do it without affecting its ability to play any of the

videos being programmed. In addition, the technique does not require a large number of successful preloads to have a significant impact upon the overall performance of the protocol. Consider for instance a two-hour video that is watched by an average of 60 customers an hour. As one can see on Figure 2, the average bandwidth required to service these requests through an RB-3 protocol is 6.17 times the video consumption rate. Assuming that 33 percent of the requests can use a preloaded segment, the server would only have to handle 40 requests per hour. Hence, the average bandwidth required to service the original 60 requests would be reduced to that required to serve 40 requests, that is 5.7 times the video consumption rate.

5.2 Distributing more initial segments on demand

The performance of reactive broadcasting is strongly affected by the number of segments that we can pack into the k broadcasting streams subject to the condition that segment S_i with $1 < i \leq n$ must be broadcast at least once every $i-1$ slots. Mapping more segments into the same number of streams would reduce the segment sizes and thus the amount of information that has to be distributed on demand.

One way to obtain more efficient mappings is to distribute the two first segment of each video on demand so that the k broadcasting streams would only have to broadcast segments S_3 to S_n. Since the segment that has to be repeated the most frequently would be now segment S_3 instead of segment S_2, no segment would have to be repeated more frequently than once every two slots.

Consider first the case of a reactive broadcasting protocol using two broadcasting streams (RB-2). As we saw before, distributing on demand the first segment of the video allows us to pack three segments into the two broadcasting streams for a total of four segments. Hence 25 percent of the video is distributed on demand. If we decide instead to distribute on demand segments S_1 and S_2, we will be able to pack eight additional segments into the two broadcasting streams using the mapping:

First Stream	S_3	S_5	S_3	S_6	S_3	S_5
Second Stream	S_4	S_7	S_9	S_4	S_8	S_{10}

for a total of ten segments. Even though we would distribute on demand two segments instead of one, these two segments would only represent 20 percent of the video instead of the 25 percent we had to distribute before.

Similar savings could be achieved with the RB-3 protocol. Distributing on demand both segments S_1 and S_2 would allow us to pack 23 additional segments into the three broadcasting streams using the mapping:

First Stream	S_3	S_5	S_3	S_9	S_3	S_5	S_3	S_{10}	S_3	S_5	S_3	S_9	S_3	S_5	S_3	S_{10}	S_3	S_5	S_3	S_9
Second Stream	S_4	S_7	S_{13}	S_4	S_8	S_{14}	S_4	S_7	S_{15}	S_4	S_8	S_{13}	S_4	S_7	S_{14}	S_4	S_8	S_{15}	S_4	S_7
Third Stream	S_6	S_{11}	S_{16}	S_{19}	S_{22}	S_6	S_{12}	S_{17}	S_{20}	S_{23}	S_6	S_{11}	S_{18}	S_{21}	S_{24}	S_6	S_{12}	S_{16}	S_{19}	S_{25}

for a total of 25 segments. We would thus have to distribute on demand 8 percent of the video instead of the 10 percent we had to distribute before.

We should note that reductions in the size of the data to be transmitted by stream tapping will not have a proportional effect on the bandwidth requirements of the protocol. Because of the way stream tapping handles overlapping requests, any reduction of the duration of the video data to be distributed has the same effect as a proportional reduction of the customer arrival rate. Looking back at Figure 2, we can see that the three curves linking the average bandwidth requirements of our protocol with the customer arrival rates have very moderate slopes. Figure 3 shows us on the other hand that reductions in the customer arrival rate will have much more impact on the maximum bandwidth requirements of the protocol.

6. CONCLUSIONS

One of the most promising approaches to reduce the cost of video on demand services is to broadcast continuously the most frequently requested videos. The best broadcasting protocols assume that client set top boxes (or STB's) have enough local storage to store at least one half on each video being watched. This allows the protocol to transmit the later parts of the video either less frequently or at a lower bandwidth. While broadcasting is the best technique for distributing very popular videos over a large metropolitan area, it does not handle as efficiently moderately popular videos.

We have proposed a reactive broadcasting protocol that addresses the problem of distributing these videos in an efficient fashion. Like all efficient broadcasting protocols, reactive broadcasting assumes that the customer set-top box has enough local storage to store at least one half of each video being watched. Unlike other broadcasting protocols, reactive broadcasting only broadcasts the later portions of each video being distributed. The initial segment of each video is distributed on demand using a stream tapping protocol.

Our simulations indicated that reactive broadcasting with three broadcasting streams outperforms both stream tapping and Pagoda Broadcasting whenever customer interarrival times vary between 0.54 and 4 percent of the video duration, that is, between 12 and 92 arrivals per hour for a typical two-hour video. The bandwidth savings can exceed 20 percent over both methods. Even when it is not the optimal protocol, reactive broadcasting continues to perform in an acceptable fashion over a much wider range of customer arrival rates than its two competitors. It constitutes the safest choice for distributing all videos that are not known to be either extremely popular or very unlikely to be simultaneously watched by, say, more five to ten customers.

ACKNOWLEDGMENTS

This research was supported by the Office of Naval Research under Grant N00014-92-J-1807, by the National Science Foundation under grant PO-10152754 and by the USENIX Association. It was performed while the first author was visiting the Jack Baskin School of Engineering of the University of California, Santa Cruz.

REFERENCES

1. C. C. Aggarwal, J. L. Wolf, and P. S. Yu. A permutation-based pyramid broadcasting scheme for video-on-demand systems. *Proc. International Conference on Multimedia Computing and Systems*, pages 118–26, June 1996.
2. S. W. Carter and D. D. E. Long. Improving video-on-demand server efficiency through stream tapping. *Proc. International Conference on Computer Communication and Networks* (ICCCN '97), Las Vegas, NV, pages 200-207, Sep. 1997.
3. S. W. Carter and D. D. E. Long. Improving bandwidth efficiency on video-on-demand servers. *Computer Networks and ISDN Systems*, 30(1–2):99–111.
4. A. Dan, P. Shabuddin, D. Sitaram and D. Towsley. Channel allocation under batching and VCR control in video-on-demand systems. *Journal of Parallel and Distributed Computing*, 30(2):168–179, Nov. 1994.
5. A. Dan, D. Sitaram, and P. Shahabuddin. Dynamic batching policies for an on-demand video server. *Multimedia Systems*, 4(3):112–121, June 1996.
6. L. Golubchik, J. C. S. Lui, and R. R. Muntz. Adaptive piggybacking: a novel technique for data sharing in video-on-demand storage servers. *Multimedia Systems*, 4(3):140–155, June 1996.
7. K. A. Hua and S. Sheu. Skyscraper broadcasting: a new broadcasting scheme for metropolitan video-on-demand systems. *Proc. ACM SIGCOMM '97 Conference*, Cannes, France, pages 89–100, Sept. 1997.
8. K. A. Hua, Y. Cai, and S. Sheu. Patching: a multicast technique for true video-on-demand services. *Proc. 6th ACM Multimedia Conference*, Bristol, GB, pages 191–200, Sep. 1998.
9. L. Juhn and L. Tseng. Harmonic broadcasting for video-on-demand service. *IEEE Transactions on Broadcasting*, 43(3): 268–271, Sept. 1997.
10. L. Juhn and L. Tseng. Fast data broadcasting and receiving scheme for popular video service. *IEEE Transactions on Broadcasting*, 44(1):100–105, March 1998.
11. J.-F. Pâris, S. Carter and D. D. E. Long. Efficient broadcasting protocols for video on demand. *Proc. 6th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems* (MASCOTS '98), Montréal, Canada, pages 127–132, July 1998.
12. J.-F. Pâris, S. W. Carter and D. D. E. Long. A hybrid broadcasting protocol for video on demand. *Proc. 1999 Multimedia Computing and Networking Conference* (MMCN'99), San Jose, CA, pages 317–326, Jan. 1999.
13. J.-F. Pâris. A simple low-bandwidth broadcasting protocol for video on demand, *Proc. 8th International Conference on Computer Communications and Networks* (IC3N'99), October 1999, to appear.
14. S. Viswanathan and T. Imielinski. Metropolitan area video-on-demand service using pyramid broadcasting. *Multimedia Systems*, 4(4):197–208, 1996.
15. J. W. Wong. Broadcast delivery. *Proceedings of the IEEE*, 76(12), 1566–1577, Dec. 1988.