A Dynamic Heuristic Broadcasting Protocol for Video-on-Demand

Scott R. Carter
Jehan-François Pâris
Saurabh Mohan
Department of Computer Science
University of Houston
Houston, TX 77204-3475

Darrell D. E. Long
Department of Computer Science
Jack Baskin School of Engineering
University of California
Santa Cruz, CA 95064

ABSTRACT

Most existing distribution protocols for video-on-demand are tailored for a specific range of video access rates and perform poorly beyond that range. We present a dynamic heuristic broadcasting protocol that performs as well as stream tapping with unlimited extra tapping at low video access rates and has the same average bandwidth requirements as the best existing broadcasting protocols at high video access rates. We also show how our protocol can handle compressed video and adapt itself to the individual bandwidth requirements of each video.

1. INTRODUCTION

In the current state of the technology, video-on-demand (VOD) is too expensive to effectively compete against cheaper rivals such as pay-per-view and video-cassette rentals. The major reason behind the high cost of VOD is the extremely high bandwidths it requires to service individual customer requests. Handling such bandwidths would require major outlays for upgrading the existing communication infrastructure and building servers capable of handling the resulting I/O traffic.

This situation has resulted in many proposals aimed at reducing the bandwidth requirements of VOD services. Despite all their differences, most of these proposals fit into one of two groups. The first group of proposals follows a *reactive* approach. These proposals assume that the video server will merely answer individual customer requests without trying to anticipate them. Whenever several user requests for the same video arrive in close succession, the server will try to transmit only once all the data that can be shared by two or more requests.

Proposals in the second group take a different approach: they anticipate customer demand and distribute the various segments of each video according to a deterministic schedule. These distribution protocols are said to be *proactive* and are grouped under the common name of *broadcasting protocols* [20].

Each of these two approaches has its own advantages and disadvantages. Reactive protocols perform much better than broadcasting protocols as long as the request arrival rate for a two-hour video remains below, say ten to fifteen requests per hour [15]. Because proactive protocols follow a deterministic broadcasting schedule, their bandwidth requirements are not affected by the request arrival rate for a given video. Hence they are the best protocols for distributing videos that are in very heavy demand.

The real problem is that the frequency of requests for any given video is likely to vary widely with the time of the day. Child-oriented fare will always be in higher demand during the day and early evening hours than at night. Conversely, videos appealing to older viewers are likely to follow an opposite pattern. No conventional distribution protocols can effectively handle the distribution of these videos. While reactive protocols will perform very well when the video is in low demand, they run the risk of overloading the server when the video is in high demand. On the contrary, broadcasting protocols will perform very well when the video is in high demand but will waste a large fraction of their bandwidth in all other cases.

One possible solution to this problem is to design better reactive protocols that can handle high request arrival rates. These protocols include Eager and Vernon's *dynamic skyscraper broadcasting* (DSB) protocol [5], their more recent *hierarchical multicast stream merging* (HMSM) protocol [6] and Gao, Zhang and Towsley *selective catching* (SC) [8]. Another solution is to combine the reactive and the proactive approaches as in our *universal distribution* protocol (UD) [17]. While all four protocols perform very well at low to medium request arrival rates, their performance at high request arrival rates is not as good as that of the best broadcasting protocols.

The *dynamic heuristic broadcasting* protocol does not suffer from this limitation. It performs reasonably well at all request arrival rates and can be easily tailored to the specific bandwidth requirements of any given compressed video.

1063-6927/01 $10.00 © 2001 IEEE

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

Figure 1. The first three streams for fast broadcasting

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

Figure 2. The first three streams for the NPB protocol

2. RELATED WORK

The first video distribution protocols attempted to reduce bandwidth either by batching together several requests [3, 4] or by accelerating the video playback rate to let new requests catch up with previous transmissions [10]. This situation changed when Viswanathan and Imielinski [19] proposed to add to the customer set-top box (STB) enough buffer space to store between, say, thirty minutes and one hour of video data. This allowed the STB to receive most video data out of sequence and thus resulted into the first efficient broadcasting protocol. We will only mention here those broadcasting protocols that are directly relevant to our work.

The simplest broadcasting protocol is Juhn and Tseng's *fast broadcasting* (FB) protocol [13]. FB allocates to each video k data streams whose bandwidths are all equal to the video consumption rate b. It then partitions the video to be broadcast into 2^k-1 segments S_1 to S_{2^k-1} of equal duration d. As Figure 1 indicates, the first stream continuously rebroadcasts segment S_1, the second stream transmits segments S_2 and S_3, and the third stream transmits segments S_4 to S_7. More generally, stream j with $1 \le j \le k$ transmits segments $S_{2^{j-1}}$ to S_{2^j-1}.

When customers want to watch a video, they wait until the beginning of the next transmission of segment S_1. They then start watching the video on the first stream while their STB starts downloading data from all other streams. By the time the customer has finished watching segment S_1, segment S_2 will either be already downloaded or ready to be downloaded. More generally, any given segment S_i will either be already downloaded or ready to be downloaded by the time the customer has finished watching segment S_{i-1}.

The *universal distribution* (UD) protocol [17] is a dynamic broadcasting protocol based upon the FB protocol. Segments are transmitted only on demand, which saves a considerable amount of bandwidth when the request arrival rates remains below 100 requests per hour. Above 200 requests per hour, all channels become saturated and the UD reverts to a conventional FB protocol.

The *new pagoda broadcasting* (NPB) [14] protocol improves upon the FB protocol by using a more complex segment-to-stream mapping. As seen on Figure 2, the NPB protocol can pack nine segments into three streams while the FB protocol can only pack seven segments. Hence the segment size will be equal to one ninth of the duration of the video and no customer would ever have to wait more than 14 minutes for a two-hour video.

First Stream	S_1	S_1	S_1	S_1	S_1	S_1
Second Stream	S_2	S_3	S_2	S_3	S_2	S_3
Third Stream	S_4	S_5	S_4	S_5	S_4	S_5

Figure 3. The first three streams for skyscraper broadcasting

Hua and Sheu's *skyscraper broadcasting* (SB) [11] differs from both FB and NPB by never requiring the STB to receive more than two streams at the same time. As Figure 3 shows, the result is a segment-to-stream mapping that packs fewer segments per stream. Hence SB will always require more server bandwidth than NPB and FB to guarantee the same maximum waiting time d. Eager and Vernon's *dynamic skyscraper broadcasting* (DSB) [5] is a reactive protocol based upon the SB protocol. Since it abides by the same restriction on client bandwidth as the original SB protocol, it also requires a higher server bandwidth than the UD protocol. Their more recent *hierarchical multicast stream merging* (HMSM) protocol requires less server bandwidth than DSB to handle the same request arrival rate. Its bandwidth requirements are indeed very close to the theoretical minimum for a reactive protocol that does not require the STB to receive more than two streams at the same time [6].

Stream tapping [2] and *patching* [12] take a purely reactive approach. To use one of these methods, 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 they are needed. In the best case, clients can get most of their data from existing streams, which greatly reduces the duration of their own stream.

Selective catching combines both reactive and proactive approaches. It dedicates a certain number of channels for periodic broadcasts of videos while using the other channels to allow incoming requests to catch up with the current broadcast cycle. As a result, its bandwidth requirements are $O(\log(\lambda_i L_i))$ where λ_i is the request arrival rate and L_i the duration of the video [8].

3. OUR PROTOCOL

While the UD protocol performed as well as the best reactive protocols at low request arrival rates, its performance at high request arrival rates was less satisfactory as it was outperformed there by the NPB protocol [17]. We wanted to design a better dynamic broadcasting protocol that would not be outperformed by any other distribution protocol at any request arrival rate.

We first experimented with a dynamic version of the NPB protocol. As we expected, it bested the UD protocol at moderate to high access rates because its bandwidth requirements never exceeded those of NPB. Unfortunately, its performance lagged behind that of both UD and stream tapping whenever there were less than 40 to 60 requests per hour.

We then decided in favor of a different approach. Our new *dynamic heuristic broadcasting* (DHB) protocol would not be based on any existing broadcasting protocol. It would instead use a more flexible heuristic approach.

The DHB protocol partitions each video into an arbitrary number n of segments S_i of equal duration $d = D/n$, where D is the duration of the video. These segments will be broadcast on demand on any of k identical data streams whose bandwidth is equal to the video consumption rate b. DHB is a *slotted* protocol as all segments will always start at times that are multiples of the segment duration d.

When customers request a video, their STB sends a message to the video server, which prepares a transmission schedule starting at the next slot. Thus the segment duration d is also the maximum time any customer will ever wait before starting to watch a video.

To reach peak performance and to achieve a minimum average bandwidth, we need to obtain maximum sharing of each video segment among all overlapping requests. To realize maximum sharing, each segment must be delayed as long as possible to allow a maximum number of future requests to share each segment. If a transmission schedule starting at slot $i + 1$ cannot share its j-th segment S_j with any previous transmission schedule, it should attempt to schedule it in slot $i + j$ but never later than that. Note that each segment S_i has to be scheduled at a unique minimum frequency $1/id$ (or a maximum period id). Thus the first segment must be scheduled at least once every slots, and so on. The DHB protocol achieves minimum bandwidth by scheduling each segment on demand according to its minimum frequency.

Slot	1	2	3	4	5	6	7	8	9
1st Stream	–	S_1	S_2	S_3	S_4	S_5	S_6	–	–
2nd Stream	–	–	–	–	–	–	–	–	–

Figure 4. Transmission schedule of an incoming request arriving into an idle system.

When a request arrives at the server, the DHB protocol first checks the transmission schedules of all current requests to find which segments are already scheduled in some future slot. If this is the case then there is already a timely transmission of the segment and no new transmission has to be scheduled. The protocol then schedules transmissions of all remaining segments, each in the furthest possible slot for that segment.

Consider, for instance, the case a video that has been partitioned into six segments. Figure 4 represents the segment-to-slot mapping that would have resulted from the arrival of an incoming request into an idle system during slot 1. Since there were no previously scheduled transmission for any of the six video segments, the protocol will schedule one transmission of segment S_1 during slot 2, one transmission of segment S_2 scheduled during slot 3, and, more generally, one transmission of segment S_i during slot $i + 1$ for all $1 \le i \le 6$.

Slot	1	2	3	4	5	6	7	8	9
1st Stream	–	S_1	S_2	S_3	S_4	S_5	S_6	–	–
2nd Stream	–	–	–	S_1	S_2	–	–	–	–

Figure 5. Combined transmission schedules of two overlapping requests for the same video.

Consider now what would happen if a second request for the same video arrived during slot 3. The new request could share segments S_3 to S_6 with the first request. As Figure 5 shows, the protocol would only schedule one new transmission of segment S_1 during slot 4 and one transmission of segment S_2 scheduled during slot 5.

Note that the protocol will never schedule more than one instance of segment S_i once every i slots because all requests arriving within $i - 1$ slots after a request already having a transmission of that segment in its schedule will be able to share that transmission.

Assumptions:

slot k already contains m_k segment instances
video contains n segments
new video request arrives during slot i

Algorithm:

for $j := 1$ **to** n **do**
search slots $i+1$ to $i+j$ for an already scheduled instance of S_j
if not found **then**
let $m_{min} := \min \{m_k \mid i+1 \le k \le i+j\}$
let $k_{max} := \max \{k \mid i+1 \le k \le i+j \text{ and } m_k = m_{min}\}$
schedule one instance of S_j in slot k_{max}
end if
end for loop

Figure 6. The Dynamic Hybrid Broadcasting protocol.

This very simple approach is not very different from that used in [6] to derive a lower bound on the server bandwidth for delivery techniques that provide immediate real time service to client. It cannot work because intolerable bandwidth peaks would occur whenever too many segments are scheduled to the same slot. Consider for instance a two-hour video that has been partitioned into 120 segments to guarantee a maximum waiting time of one minute. Assume that the video is in high demand and that there is at least one request arriving during each slot, starting with slot 1. Since each segment S_i with $1 \le i \le 120$ has to be scheduled once every i slots starting with slot i, slot 120! will contain one transmission of each and every segment of the video. The result will be a bandwidth peak equal to 120 times the video consumption rate.

As shown in Figure 6, the DHB protocol avoids this problem thanks to a simple scheduling heuristic. Consider a request arriving during slot i and assume that the request requires a new transmission of segment S_j. Our protocol will search slots $i+1$ to $i+j$ to find the slot having the minimum number m_{min} of scheduled transmissions and schedule a new transmission of segment S_j during that slot. If two or more slots are found to have the minimum number of scheduled transmissions, the protocol always picks the slot k_{max} with the longest delay. Note that the heuristics never affects the customer waiting time as segment S_j can only be transmitted on time (when $k_{max} = i+j$) or ahead of schedule ($k_{max} < i+j$).

To evaluate the performance of our protocol we wrote a simple simulation program assuming that requests for a particular video were distributed according to a Poisson law. The results of our simulations are summarized in Figures 7 and 8.

Figure 7 compares the average bandwidth requirements of our DHB protocol with 99 segments with the average bandwidth requirements of the universal distribution protocol (UD), stream tapping and new pagoda broadcasting (NPB)l. Request arrival rates are expressed in arrivals per hour and bandwidths are expressed in multiples of the video consumption rate. We assumed a video duration of two hours and an unlimited buffer size for stream tapping.

Note that both stream tapping allows instant access to the video while the three other protocols only guarantee that no customer will ever wait more than 1/99 of the duration of the video, that is no more than 73 seconds for a two-hour video. Observe also that the bandwidth requirements of NPB do not vary with the request arrival rate because NPB distributes video segments according to a deterministic schedule that is not affected by the timing of incoming requests.

We can immediately see that the new DHB protocol requires less average bandwidth than its four rivals do for all request arrival rates above two requests per hour. Stream tapping performs slightly better than DHB at one request per hour but is outperformed by the four other protocols at higher request arrival rates above the same two requests per hour. This should be expected from a reactive protocol offering zero-delay access to all videos. Similar considerations would apply to selective catching.

We were particularly happy to observe that DHB had lower average bandwidth requirements than NPB at all request arrival rates because NPB is the most efficient broadcasting protocol that uses (a) fixed-sized segments and (b) a limited number of equal bandwidth data streams. DHB does not waste more bandwidth scheduling segments on the fly than NPB does by using a segment-to-slot mapping that fills completely each data stream.

Figure 8 compares the maximum bandwidth requirements of our DHB protocol with 99 segments with those of the UD and NPB protocols with the same number of segments. As one can see, NPB has the smallest maximum bandwidth and DHB the highest but the difference between these two protocols never exceeds twice the

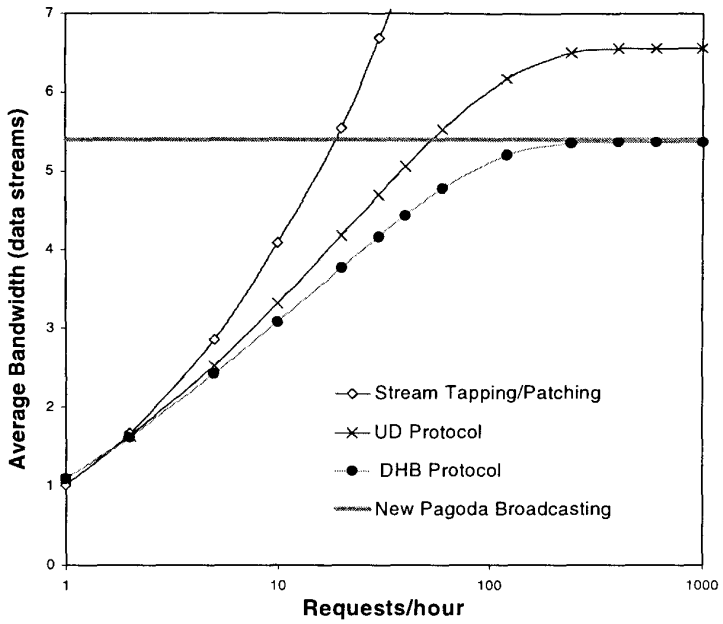

Figure 7. Compared average bandwidth requirements of stream tapping, NPB, UD and DHB protocols with 99 segments.

video consumption rate. We believe it is a very reasonable price to pay for the better average performance of our new protocol.

Another cost factor to consider is the cost of scheduling segments on the fly rather than according to a predefined mapping. Going back to our example of a video partitioned into 99 segments, we can see that each incoming request will result in the separate scheduling of 99 possible new segment instances. Fortunately for us, the actual complexity of the task will be greatly reduced at high arrival rates because most of the segment instances required by a particular request would have been already scheduled by some previous request.

4. HANDLING COMPRESSED VIDEOS

Most broadcasting protocols assume that videos will have a fixed bandwidth corresponding to a fixed video consumption rate. This assumption is incorrect because all video servers will broadcast compressed videos whose bandwidth requirements will depend on the rate at which the images being displayed change [1, 9, 7]. To ensure jitter-free delivery of video in a system allocating a fixed bandwidth to each video, the VOD server will have to set the broadcasting bandwidth to the maximum bit rate required by the most rapidly changing moments of the fastest paced scenes of the video.

Two existing broadcasting protocols can handle compressed videos. They are *polyharmonic broadcasting with partial preloading* (PHB-PP) and the *Mayan temple broadcasting* protocol (MTB) [16], but these two protocols have never been tested on a real video. In addition, these two protocols are much more complex to implement than FB or NPB as PHB-PP requires a large number of small bandwidth data streams and MTB uses segments of increasing durations.

To evaluate how our DHB would handle compressed videos, we analyzed a DVD format version of the movie *The Matrix*. We selected that format because it uses the MPEG compression algorithm and is widely available. The video lasts 8170 seconds, that is, 2 hours 16 minutes and 10 seconds. The maximum bandwidth over a period of one second is 951 kilobytes per second and the average bandwidth 636 kilobytes per second.

Assume that we wanted to distribute the video and guarantee a maximum waiting time of one minute. The simplest solution would be to partition the video into 137 segments and allocate to each data stream 951 kilobytes per second of bandwidth. This is our base solution; we will refer to it as solution *DHB-a*.

A much more economical solution would be to require each segment to be always completely downloaded by the STB *before* the customer finishes watching the previous segment of the video. This will require all customers to wait for exactly the duration of one segment. The average

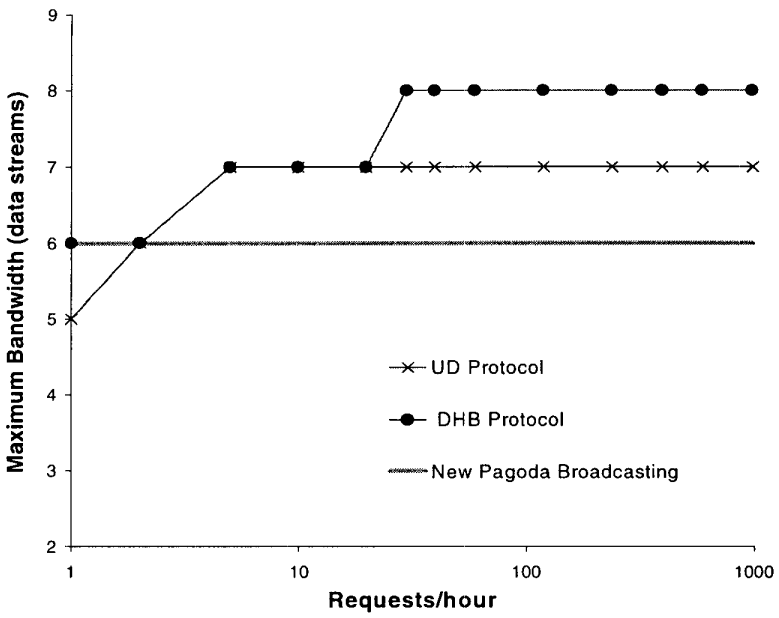

Figure 8. Compared maximum bandwidth requirements of NPB, UD and DHB protocols with 99 segments.

waiting time would thus be doubled while the maximum waiting time would remain the same. The great advantage of this solution is that the actual frame arrival times at the STB become totally irrelevant as long as all the frames arrive within the required time interval. The bandwidth required to ensure on time delivery of all segments will now be the maximum of the average bandwidths of all 137 segments, that is, 789 kilobytes per second. We will refer to this solution as solution *DHB-b.*

Even though it is much better than the base solution, solution *DHB-b* still does not use all the available bandwidth. A better solution is to make continuous use of all that bandwidth so that each one-minute segment would normally contain more than one minute of video data. This is what Salehi et al. call *smoothing by work-ahead* [18]. It would have two advantages. First, we would pack the contents of the 137 original segments into 129 segments of equal duration. Second, so much data would be received ahead of time that the bandwidth peaks occurring later in the video would be completely buffered. As a result, solution *DHB-c* needs to transmit less segments than solutions *DHB-a* and *DHB-b.* Since most bandwidth peaks have disappeared, the bandwidth at which each segment is transmitted is also reduced from 789 to 671 kilobytes per second.

One last optimization is possible. As many video data are now transmitted ahead of time, most segments will not need to be transmitted as frequently as before. Adjusting these minimum transmission frequencies will result in a further reduction of the average number of data streams that are needed to service a given number of customers per hour. We found, for instance, that the second segment of our video only needed to be broadcast every three slots and that nearly all other segments could be delayed by one to eight slots: the sole exceptions were segment S_1, which still had to be transmitted once every slot, and segment S_3 , which had still to be transmitted once every three slots. We will refer to this solution as solution *DHB-d.*

Figure 9 displays the average bandwidth requirements of implementations *DHB-a* to *DHB-d* protocol and compares them to these of a UD protocol. As one can see, more dramatic reductions in bandwidth can be achieved by tuning a specific broadcasting protocol to the requirements of a specific video rather than by replacing the protocol itself. Switching to a deterministic waiting time has the most impact on the average bandwidth requirements of the protocol (DHB-b) followed by adjusting the minimum segment frequency (*DHB-d*).

The very low bandwidth reduction that was achieved by merely reducing the number of segments being broadcast (*DHB-c*) should not surprise us. One of the major characteristics of any good broadcasting protocol for video-on-demand is that the broadcasting of high-numbered segments requires very little bandwidth.

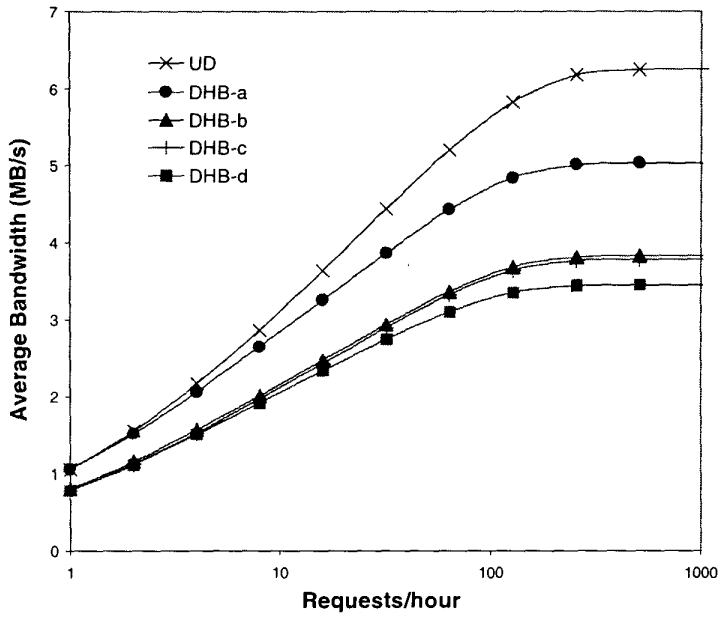

Figure 9. Compared average bandwidth requirements of the UD protocol and four implementations of the DHB protocol.

Hence reducing the number of segments to be broadcast from 137 to 129 could not have had any significant impact on the total bandwidth. Conversely, adjusting the minimum segment frequencies had a more significant impact because segment S_2 and S_4 were among the segments that were now transmitted less frequently.

The impact of these modifications on the general philosophy of the DHB protocol will be minimal. The protocol will first analyze each video to be broadcast to find the minimum bandwidth b_{min} that need to be allocated to it to guarantee on -time delivery of all frames. It will then select a channel bandwidth b that satisfies the condition $b \geq b_{min}$. Selecting a bandwidth b greater than b_{min} will increase the probability that the empty slots could be shared by other videos, which could have higher minimum bandwidth requirements.

Once an effective transmission bandwidth b is selected, the protocol will compute the number n' of segments that need to be scheduled with $n' \geq D/d$. It will then associate with each segment S_i a *maximum period* $T[i]$ representing the maximum number of slots by which the transmission of S_i can be delayed. We will necessarily have $T[1] = 1$ and, more generally, $T[i] \geq i$ for $i \geq 2$.

The protocol will otherwise remain unchanged with one sole exception. Whenever a request arriving during slot i will require a new transmission of segment S_j, the protocol will now search slots $i + 1$ to $i + T[j]$ to find the slot having the minimum number of scheduled transmissions rather than searching only slots $i + 1$ to $i + j$.

The cost of these modifications will also be minimal since each video to be broadcast will need to be analyzed only once.

5. CONCLUSIONS

Most existing distribution protocols for video-on-demand are tailored for a specific range of video access rates and perform poorly beyond that range. We have presented a dynamic heuristic broadcasting (DHB) protocol that performs as well as the best existing protocols at any access rate. Because of its flexibility, our protocol can adapt itself to the individual bandwidth requirements of each video. As it could be expected, considerable bandwidth savings can be achieved by (a) adopting a deterministic waiting time, (b) always using all available bandwidth and (c) never transmitting video data more frequently than required.

More work is still needed. We will first apply our DHB protocol to other videos in order to learn how its performance is affected by the individual characteristics of each video. We also want to investigate how we could reduce or eliminate bandwidth peaks without increasing the average video bandwidth. Finally, we would like to investigate dynamic heuristic broadcasting protocols that limit the client bandwidth to two or three data streams [6].

ACKNOWLEDGEMENTS

Scott R. Carter was partially supported by the University of Houston Scholars Program. J.-F. Pâris acknowledges the support of the Texas Advanced Research Program under grant 003652-0124-1999 and the National Science Foundation under grant CCR-9988390. Saurabh Mohan was partially supported by the Texas Advanced Research Program under grant 003652-0124-1999. Darrell Long acknowledges the support of the National Science Foundation under grant CCR-9988363.

REFERENCES

[1] Beran, J., R. Sherman, M. Taqqu, and W. Willinger. Long-range dependence in variable bit-rate video traffic. *IEEE Trans. on Communication*, 43:1566–1579, 1995.

[2] Carter, S. W. and D. D. E. Long. Improving video-on-demand server efficiency through stream tapping. *Proc. 5th Int. Conf. on Computer Communications and Networks*. pages 200-207, Sep. 1997.

[3] Dan, A., P. Shahabuddin, 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.

[4] Dan, A., D. Sitaram, and P. Shahabuddin. Dynamic batching policies for an on-demand video server. *Multimedia Systems*, 4(3):112–121, June 1996.

[5] Eager, D. L. and M. K. Vernon. Dynamic skyscraper broadcast for video-on-demand. *Proc. 4th Int. Workshop on Advances in Multimedia Information Systems*, pages 18–32, Sep. 1998.

[6] Eager, D. L., M. K. Vernon and J. Zahorjan. Minimizing bandwidth requirements for on-demand data delivery. *Proc. 5th Int. Workshop on Advances in Multimedia Information Systems*, Oct. 1999.

[7] Feng, W. and J. Rexford. Performance evaluation of smoothing algorithms for transmitting prerecorded variable-bit-rate video. *IEEE Trans. on Multimedia*, September 1999, 302–313.

[8] Gao, L., Z.-L Zhang and D. Towsley. Catching and selective catching: efficient latency reduction techniques for delivering continuous multimedia streams. *Proc. 1999 ACM Multimedia Conf.*, pages 203–206, Nov. 1999.

[9] Garrett, M. and W. Willinger. Analysis, modeling and generation of self-similar VBR video traffic. *Proc. ACM SIGCOMM '94 Conference*, pages 269–280, Aug. 1994.

[10] Golubchik, L., J. Lui, and R. Muntz. Adaptive piggybacking: a novel technique for data sharing in video-on-demand storage servers. *Multimedia Systems*, 4(3): 140–155, 1996.

[11] Hua, K. A. and S. Sheu. Skyscraper broadcasting: a new broadcasting scheme for metropolitan video-on-demand systems. *Proc. ACM SIGCOMM '97 Conf.*, pages 89–100, Sept. 1997.

[12] Hua, K. A., Y. Cai, and S. Sheu. Patching: a multicast technique for true video-on-demand services. *Proc. 6th ACM Multimedia Conf*, pages 191–200, Sep. 1998.

[13] Juhn, L. and L. Tseng. Fast data broadcasting and receiving scheme for popular video service. *IEEE Trans. on Broadcasting*, 44(1):100–105, March 1998.

[14] Pâris, J.-F.. A simple low-bandwidth broadcasting protocol for video on demand, *Proc. 7th Int. Conf. on Computer Communications and Networks* (ICCCN'99), pages 690–697, Oct. 1999.

[15] Pâris, J.-F., D. D. E. Long and P. E. Mantey. A zero-delay broadcasting protocol for video on demand. *Proc. 1999 ACM Multimedia Conf.*, pages 189–197, Nov. 1999.

[16] Pâris, J.-F., S. W. Carter and D. D. E. Long. A reactive broadcasting protocol for video on demand. *Proc. 2000 Multimedia Computing and Networking Conf.*, pages 216–223, Jan. 2000.

[17] Pâris, J.-F., S. W. Carter and D. D. E. Long. A universal distribution protocol for video-on-demand. *Proc. Int. Conf. on Multimedia and Expo 2000*, July 2000.

[18] Salehi, J. D., Z.-L. Zhang, J. Kurose, and D. Towsley. Supporting stored video: reducing rate variability and end-to-end resource requirements through optimal smoothing. *Proc. 1996 ACM SIGMETRICS Conf.*, pages 222–231, May 1996

[19] Viswanathan, S. and T. Imielinski. Metropolitan area video-on-demand service using pyramid broadcasting. *Multimedia Systems*, 4(4):197–208, 1996.

[20] Wong, J. W. Broadcast delivery. *Proc. of the IEEE*, 76(12), 1566–1577, Dec. 1988.