# A UNIVERSAL DISTRIBUTION PROTOCOL FOR VIDEO-ON-DEMAND

*Jehan-François Pâris*
Department of Computer Science
University of Houston
Houston, TX 77204-3475

*Steven W. Carter* *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 request arrival rates and do not perform well beyond that range. We present a universal distribution protocol based on Juhn and Tseng's *fast broadcasting* protocol. Our protocol performs as well as the best reactive protocol at low to moderate request arrival rates and reverts to the fast broadcasting protocol at high arrival rates.

**Keywords:** video-on-demand, broadcasting protocols, stream tapping.

## 1. INTRODUCTION

Recent years have seen numerous proposals aimed at reducing the bandwidth of video-on-demand (VOD) services. All these proposals can be broadly classified into two groups.

The first group consists of *reactive* protocols that assume that the video server will merely answer individual customer requests without trying to anticipate them. The second group of proposals takes a different approach: it anticipates customer demand and distributes 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*.

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 given video remains below, say ten to twenty requests per hour. Proactive protocols follow a deterministic broadcasting schedule that is not affected by the request arrival rate for a given video. Hence they perform best at very high request arrival rates and should not be used to distribute videos that are in low demand. The overall consensus so far has been that broadcasting protocols are the best technique to distribute the ten or twenty most popular videos over a very large customer base while reactive protocols perform better in all other cases, namely, less popular videos or smaller customer bases.

This viewpoint fails to take into consideration that the popularity of a video is likely to vary over time. 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.

None of the existing distribution protocols can effectively handle the distribution of these videos. While reactive protocols would 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. Conversely, proactive protocols would perform very well when the video is in high demand but waste a large fraction of their bandwidth in all other cases. The solution we propose is a universal distribution protocol that performs *fairly well* for any request arrival rate. More precisely, our protocol performs as well as the best reactive protocol at low to moderate request arrival rates and reverts to the fast broadcasting protocol at high arrival rates. As a result, its bandwidth requirements always remain within 30 percent of the most efficient broadcasting protocols.

## 2. PREVIOUS WORK

For brevity sake, we will focus our discussion on the techniques that are directly relevant. These include the fast broadcasting protocol [5], on which the universal distribution protocol is based, and four other protocols, which we will use as benchmarks.

*Fast broadcasting* (FB) [5] allocates to each video to be broadcast $k$ 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 set-top box (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 $S_{i-1}$.

0-7803-6536-4/00/$10.00 (c) 2000 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

The only serious drawback of the FB protocol is its requirement that the customer STB should have enough buffer space to store up to 50 percent of the duration of the video being watched. In the current state of storage technology, this implies that the STB must include a hard drive.

The *new pagoda broadcasting* (NPB) [6] 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*** | $S_2$ | $S_4$ | $S_2$ | $S_5$ | $S_2$ | $S_4$ |
| ***Third Stream*** | $S_3$ | $S_6$ | $S_8$ | $S_3$ | $S_7$ | $S_9$ |

**Figure 2.** The first three streams for the NPB protocol

*Skyscraper broadcasting* (SB) [3] differs from both FB and NPB by its emphasis on reducing both the size of the STB buffer and the number of data streams the STB has to receive at any given moment. 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 have recently improved skyscraper broadcasting by making the protocol dynamic [2]. Since their *dynamic skyscraper broadcasting* protocol abides by the same restrictions on client bandwidth as the original SB protocol, it also requires a higher server bandwidth than other broadcasting protocols.

| | | | | |
|---|---|---|---|---|
| ***First Stream*** | $S_1$ | $S_1$ | $S_1$ | $S_1$ |
| ***Second Stream*** | $S_2$ | $S_3$ | $S_2$ | $S_3$ |
| ***Third Stream*** | $S_4$ | $S_5$ | $S_4$ | $S_5$ |

**Figure 3**. The first three streams for skyscraper broadcasting

Unlike the four protocols we have reviewed, *stream tapping* [1] and its variants [4], take a purely reactive approach. To use stream tapping, 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 get most of their data from existing streams, which greatly reduces the amount of time they need their own stream.

## 3. THE UNIVERSAL PROTOCOL

The universal video distribution protocol was designed with three specific objectives in mind:

- The protocol should provide an acceptable performance over the widest possible range of request arrival rates.
- Our main optimization criterion should be the *average server bandwidth* required for achieving a given maximum waiting time.
- The instantaneous server and client bandwidths of the protocol should always remain bounded.

We included this last objective because a protocol that exhibits high surges of server or client bandwidth would be impractical.

There were also several issues that we decided not to consider:

- We did not try to provide zero-delay access to the videos and assumed that an average delay of up to one minute would be acceptable.
- We did not try to minimize the size of the STB buffer since the recent increases in disk drive capacity have made the issue much less pressing than a few years ago.
- Neither did we try to minimize the client bandwidth of the protocol since we found that the most basic disk drives would be able to handle the bit rates equal to five to eight times the video consumption rate.

The basic idea behind our universal distribution protocol is the same as that behind dynamic skyscraper broadcasting [2]: taking an existing proactive protocol and transforming it into a reactive protocol by broadcasting segments *on demand*. The approach guarantees a good performance for high request arrival rates, as the new protocol would behave exactly as its proactive parent. Note that our universal protocol is a *slotted* protocol, as all segments will always start at times that are multiples of the segment duration *d*.

The most critical decision we had to make was the choice of the broadcasting protocol on which to base our protocol. Since our objective was to minimize the server bandwidth, skyscraper broadcasting could be immediately eliminated, as it is one of the broadcasting protocols that requires the most bandwidth to guarantee a given maximum waiting time. On the other hand, the new pagoda protocol (NPB) was a very strong candidate given its low bandwidth requirements. We quickly found that NPB had one major drawback: its very precise segment-to-slot mapping would have resulted in a poor performance at low to moderate request arrival rates.

| *Slot* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| *1st Stream* | – | $S_1$ | $S_1$ | $S_1$ | $S_1$ | $S_1$ | $S_1$ | $S_1$ | $S_1$ |
| *2nd Stream* | – | – | $S_2$ | $S_4$ | $S_2$ | $S_5$ | $S_2$ | $S_4$ | $S_2$ |
| *3rd Stream* | – | – | – | $S_3$ | $S_6$ | $S_8$ | $S_3$ | $S_7$ | $S_9$ |

**Figure 4.** Segment-to-slot mapping of an incoming request for a hypothetical universal protocol based on the NPB protocol.

Consider, for instance, the case of a hypothetical universal protocol based on an NPB protocol with three data streams and nine 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 0. One transmission of segment $S_1$ has been scheduled during slot 1, one transmission of segment $S_2$ scheduled during slot 2, and one transmission of segment $S_3$ scheduled during slot 3. Note that the placement of the six remaining segments is constrained by the fact that these six segments have to occupy slots that are compatible with the NPB segment-to-slot mapping (represented here by the segments in gray). For instance, segments $S_4$ and $S_5$ can only occupy odd slots of stream 2 as all even slots are reserved for segment $S_2$. As a result, the whole nine segments are scheduled over eight slots, that is 8/9 of the duration of the video.

This outcome has the major drawback of reducing the potential overlap between successive requests and thus decreasing the number of segment transmissions that can be shared between successive requests. Hence, any universal protocol based on the NPB protocol would perform much worse than stream tapping at low to moderate request arrival rates. This is clearly unacceptable since the average distribution cost per request under any distribution policy will always be highest at low request arrival rates. Thus we believed it was more important to have the best possible protocol performance for low request arrival rates than for high arrival rates.

| *Slot* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| *1st Stream* | – | $S_1$ | $S_1$ | $S_1$ | $S_1$ | $S_1$ | $S_1$ | $S_1$ |
| *2nd Stream* | – | – | $S_2$ | $S_3$ | $S_2$ | $S_3$ | $S_2$ | $S_3$ |
| *3rd Stream* | – | – | – | – | $S_4$ | $S_5$ | $S_6$ | $S_7$ |

**Figure 5.** Segment-to-slot mapping of an incoming request for a universal protocol based on the FB protocol.

We decided therefore to base our universal protocol on the *fast broadcasting* (FB) protocol, whose segment-to-slot mapping is easier to manage. As Figure 5 shows, an incoming request finding an empty system will now have its seven segments scheduled over seven slots, that is, over the whole duration of the video. We found that this property was not sufficient to guarantee a good protocol performance.

| *Slot* | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| *1st Stream* | $S_1$ | $S_1$ | $S_1$ | $S_1$ | $S_1$ | $S_1$ | $S_1$ |
| *2nd Stream* | $S_3$ | $S_2$ | $S_3$ | $S_2$ | $S_3$ | $S_2$ | $S_3$ |
| *3rd Stream* | $S_5$ | $S_6$ | $S_7$ | $S_4$ | $S_5$ | $S_6$ | $S_7$ |

**Figure 6.** An inefficient allocation of the second stream.

Consider now the scenario represented on Figure 6 where a first request arriving during slot 0 is followed by two other requests respectively arriving during slots 3 and 4. Focussing on the way the protocol handles the second request, we can notice one clear inefficiency. In order to remain consistent with the current segment-to-slot map, the protocol must schedule one transmission of segment $S_2$ during slot 4, and one transmission of segment $S_3$ during slot 5 even though these two segments are only consumed during slots 5 and 6. As a result, the protocol will have to schedule another transmission of segment $S_2$ during slot 6 for the benefit of the third request.

| *Slot* | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| *1st Stream* | $S_1$ | $S_1$ | $S_1$ | $S_1$ | $S_1$ | $S_1$ | $S_1$ |
| *2nd Stream* | $S_3$ | $S_2$ | $S_3$ | $S_2$ | $S_3$ | $S_2$ | $S_3$ |
| *3rd Stream* | $S_5$ | $S_6$ | $S_7$ | $S_4$ | $S_5$ | $S_6$ | $S_7$ |

**Figure 7.** A better allocation of the second stream.

There is, however, a simple solution to this problem. Since slot 5 and 6 were still free when the protocol started handling the second request, the protocol could have started a mapping on that segment. This is the solution depicted in Figure 7, where the protocol respectively allocates slots 5 and 6 to segments $S_2$ and $S_3$. As a result, both segment transmissions can now be shared with the third request.

The final version of our universal protocol can be summarized as follows:

- Each video to be broadcast is given a maximum bandwidth allocation that will always be an integer multiple $k$ of the video consumption rate.
- The video is then partitioned into $2^k$-1 segments of equal duration $d$. These $2^k$-1 segments will be grouped into $k$ logical streams with segments $S_{2^{j-1}}$ to $S_{2^j-1}$ being assigned to stream $j$.
- The time interval during which the video is distributed is divided into fixed-size slots whose duration $d$ is equal to the duration of a segment.
- Each stream $j$ has a start slot $b_j$ whose initial value is *undefined.*
- When a request arrives during slot $i$, the server first looks at the current segment distribution schedule, stream by stream:

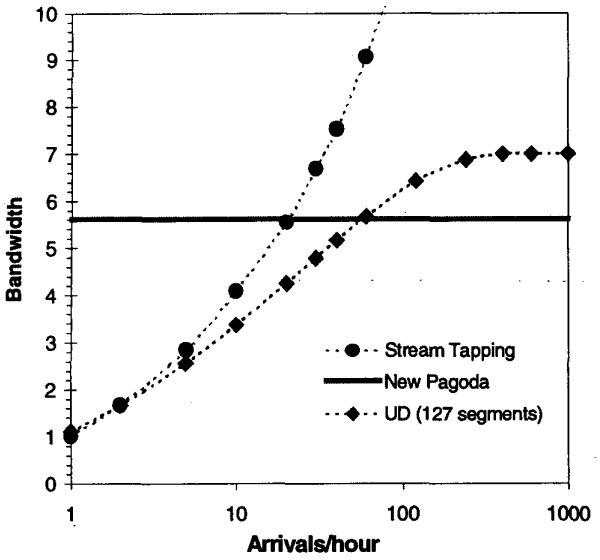


**Figure 8.** Compared bandwidth requirements of stream tapping, new pagoda broadcasting and the universal protocol with 127 segments.

a) if the last scheduled segment transmission for stream $j$ is before slot $i+2^{j-1}$ then $i+2^{j-1}$ becomes the new start slot $b_j$ for stream $j$, and

b) if no transmission of segment $S_l$ of stream $j$ has been already scheduled for any slot greater than $i$ then the server will schedule a new transmission of $S_l$ in slot $b_j+(l-2^{j-1})$

As a result, our universal distribution protocol never allocates more than one segment per stream in any given slot. Hence the instantaneous server and client bandwidths of the protocol will always remain less than or equal to the number of streams $k$.

## 4. DISCUSSION

Figure 8 compares the bandwidth requirements of our universal distribution protocol (UD) with 127 segments with the bandwidth requirements of stream tapping and new pagoda broadcasting (NPB) with 127 segments. These two protocols were selected as benchmarks because of their low bandwidth requirements within their range of customer arrival rates. Note that stream tapping allows instant access to the video while the UD and NPB protocol with 127 segments guarantee that no customer will ever wait more than 1/127 of the duration of the video, that is no more than 57 seconds for a two-hour video.

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, an exponential distribution of all interarrival times, and an unlimited buffer size for stream tapping.

We can immediately see that the new UD protocol outperforms both stream tapping and NPB when the request arrival rates remain between 5 and 55 arrivals per hour. Stream tapping performs slightly better than UD at one arrival per hour while NPB bests UD at all request arrival rates above 60 arrivals per hour. Stream tapping performs the worst of all three in that range of arrival rates but that should be expected from a protocol providing instant access to the video

The modest performance of UD at high request arrival rates was to be expected since it is based on a broadcasting protocol that does not use bandwidth as efficiently as the NPB protocol. As we said earlier, we were willing to trade a much better performance for lower arrival rates for a somewhat lower performance at very high arrival rates.

Additional simulations led us to a more interesting finding, namely that the number of segments has no effect on the bandwidth requirements of our UD protocol as long as the request arrival rate is lower than 20 requests per hour. Thus, it would be economically feasible to distribute videos partitioned into 255 segments if we wanted to further reduce the customer waiting times.

## 5. CONCLUSIONS

Most distribution protocols for video-on-demand are tuned to a specific range of video request arrival rates. Outside of that range they tend to perform rather poorly. We have presented a universal distribution protocol that performs fairly well for any request arrival rate. At low to moderate request arrival rates, our protocol performs as well as stream tapping. It reverts to the fast broadcasting protocol at high arrival rates where its bandwidth requirements remain within 30 percent of those of the most efficient broadcasting protocols.

## ACKNOWLEDGMENTS

This research was supported in part by the Texas Advanced Research Program under grant 003652-0124-1999, and by the National Science Foundation under grant PO-10152754.

## REFERENCES

[1] S. W. Carter and D. D. E. Long. Improving video-on-demand server efficiency through stream tapping. *Proc. IC3N'97 Conference*, pages 200-207, Sep. 1997.

[2] D. L. Eager 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.

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

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

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

[6] J.-F. Pâris. A Simple low-bandwidth broadcasting protocol for video on demand, *Proc. ICCCN'99 Conference*, pages 690–697, Oct. 1999.