Proc. 10th International Conference on Computer Communications and Networks, Oct. 15-17, 2001, pp. 418-423.

A Fixed-Delay Broadcasting Protocol for Video-on-Demand

Jehan-François Pâris[1]
Department of Computer Science
University of Houston
Houston, TX 77204-3475
paris@cs.uh.edu

***Abstract*—Broadcasting protocols reduce the cost of video-on-demand services by distributing more efficiently videos that are likely to be simultaneously watched by many viewers. Rather than answering individual customer requests, they broadcast the contents of each video according to a fixed schedule.**

We present a *fixed-delay pagoda broadcasting* protocol that requires all users to wait for a small fixed delay before watching the video they have selected. The protocol uses this delay to reduce the bandwidth required to transmit the first minutes of each video. As a result, our protocol provides the lowest waiting times of all protocols using segments of equal duration and channels of equal bandwidth. In addition, its performance is not very far from the theoretical minimum. We also show how to modify our protocol to restrict the set-top box receiving bandwidth to two times the video consumption rate.

I. INTRODUCTION

The main reason for the lack of success of video-on-demand (VOD) is its high cost relative to its two more entrenched rivals, namely, pay-per-view and videocassette rentals.

This situation has led to numerous proposals aiming at reducing the cost of providing video-on-demand (VOD) services. Many, if not most, of these proposals have focused on finding better ways to distribute the top ten or twenty so-called "hot" videos in a more efficient fashion. Broadcasting protocols [2] were introduced for that purpose. Rather than answering individual customer requests, they distribute the contents of each video according to a fixed schedule that is not affected by the presence–or the absence–of requests for that video. Hence the number of viewers watching a given video does not affect their bandwidth requirements.

Broadcasting protocols have two major advantages. First they scale up extremely well. Second they have very modest bandwidth requirements: the best broadcasting protocols require less than six times the video consumption rate to ensure that no customer will wait more than 42 seconds for a two-hour video [9].

We present a broadcasting protocol that has even lower bandwidth requirements. Like the *polyharmonic broadcasting* protocol [8] and the GEBB protocol [6], our *fixed-delay pagoda broadcasting* (FDPB) protocol requires all users to wait for a small fixed delay before watching the video they have selected. This small delay allows for a much more efficient transfer of the first few minutes of the video. Unlike polyharmonic broadcasting and GEBB, our FDPB protocol uses fixed-size segments and assigns them to a few fixed-bandwidth channels. It is thus much easier to implement than its two predecessors are.

We compared the bandwidth requirements of the FDPB protocol with those of pagoda broadcasting and polyharmonic broadcasting. We found that maximum customer delays for a given number of channels are typically 40 to 50 percent less than those achieved by the new pagoda broadcasting protocol.

We also present a modified version of the FDPB protocol that restricts the STB receiving bandwidth to two times the video consumption rate and show that it requires less server bandwidth than the *skyscraper broadcasting* protocol.

II. PREVIOUS WORK

The simplest video broadcasting protocol is *staggered broadcasting* [11]. 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.

The past five years have seen the development of many more efficient broadcasting protocols [2]. Most of these 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. The protocols in the first group are based on Viswanathan and Imielinski's *pyramid broadcasting* protocol [10]. They include Aggarwal, Wolf and Yu's *permutation-based pyramid broadcasting* protocol [1], Hua and Sheu's *skyscraper broadcasting* protocol [3] and Juhn and Tseng's *fast broadcasting* protocol [5].

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 [4, 8]. Harmonic protocols divide each video into n segments of duration $d = D/n$ where D is the duration of the video. With the original harmonic broadcasting protocol [4], each

[1] Supported in part by the Texas Advanced Research Program under grant 003652-0124-1999 and the National Science Foundation under grant CCR-9988390.

Slot	0	1	2	3	4	5
Channel 1	S_1	S_1	S_1	S_1	S_1	S_1
Channel 2	S_2	S_4	S_2	S_5	S_2	S_4
Channel 3	S_3	S_6	S_8	S_3	S_7	S_9

Figure 1. How pagoda broadcasting maps nine segments into three channels.

segment S_i is broadcast repeatedly on its own stream whose bandwidth is equal to b/i, where b is the consumption rate of the video).

The customer must receive all streams at once, which means that the server and the customer STB must support a bandwidth of

$$B_{HB} = \sum_{i=1}^{n} \frac{b}{i} = bH(n)$$

for each video, where $H(n)$ is the harmonic number of n.

Unfortunately, harmonic broadcasting does not always deliver all data on time [8], but two variants have been developed which solve that problem without imposing much additional waiting time on the customer [8].

Like harmonic broadcasting, *polyharmonic broadcasting* (PHB) [8] breaks each video into n segments of equal duration d. It requires however all customers to wait for a fixed time interval $w = md$ where m is some integer $m \geq 1$ and uses this time interval to start downloading the n segments of the video. As a result, segment S_i needs only to be transmitted once every $w + (i-1)d$ time units. The bandwidth required to distribute the video is thus equal to

$$B_{PHB} = \sum_{i=1}^{n} \frac{b}{m+i-1} = H(n+m-1) - H(m-1).$$

Polyharmonic broadcasting requires m times more segments than harmonic broadcasting to achieve the same maximum customer waiting time. Its bandwidth requirements are lower than those of harmonic broadcasting as long as $m > 1$.

The multitude of streams that all harmonic protocols require complicates the task of the STB's and the servers. Like HB, *pagoda broadcasting* (PB) [9] uses fixed-size segments. It assigns to each video k video channels whose bandwidths are all equal to the video consumption rate and partitions these k channels into slots of equal duration.

Figure 1 shows how PB can pack nine segments into three channels. Each channel is partitioned into *slots*, whose duration is equal to the duration of a segment. Channel 1 continuously repeats segment S_1 to ensure that it is repeated once every slot. Channel 2 broadcasts segment S_2 once every two slots and segments S_4 and S_5 once every four slots. Even though it was not stated in the original description of the PB protocol, channel 2 is subdivided into two subchannels of equal bandwidth using time-division multiplexing. The first of these subchannels, let us call it *subchannel* 0, contains all even slots of channel 1 and uses them to broadcast segment S_2 at half the channel bandwidth b. The second subchannel (*subchannel* 1) contains all odd slots of channel 1 and uses them to broadcast segments S_4 and S_5. Channel 3 is similarly subdivided into three subchannels with subchannel 0 broadcasting segment S_3, subchannel 1 broadcasting segments S_6 and S_7, and subchannel 2 broadcasting segments S_8 and S_9. As Figure 2 shows the same arrangement is repeated for all subsequent channels. All odd-numbered channels, but channel 1, are subdivided into three subchannels of equal bandwidth while all even numbered channels are similarly subdivided into two subchannels. As a result, pagoda broadcasting can pack 49 segments into 5 channels, which means that the segment size will be equal to 1/49 of the duration of the video. Hence no client would ever have to wait more than two minutes and half for a two-hour video. 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 [9].

Channel	Subchannels	Segments
1	–	S_1
2	0	S_2
	1	S_4 and S_5
3	0	S_3
	1	S_6 and S_7
	2	S_8 and S_9
4	0	S_{10} to S_{14}
	1	S_{20} to S_{29}
5	0	S_{15} to S_{19}
	1	S_{30} to S_{39}
	2	S_{40} to S_{49}

Figure 2. How pagoda broadcasting maps 49 segments into 5 channels

The GEBB protocol [6] improves upon the polyharmonic protocol by using channels of equal bandwidth $b' < b$ and increasing the size of successive segments rather than decreasing the channel bandwidths. As a result, these channels are much easier to multiplex.

III. THE FIXED-DELAY PAGODA BROADCASTING PROTOCOL

The *fixed-delay pagoda broadcasting* (FDPB) protocol differs from previous pagoda protocols in two fashions. First, it implements a fixed-delay policy that results in lower bandwidth requirements than other pagoda protocols. Second, it uses a much simpler segment-to-channel mapping.

We will consider a video of duration D to be broadcast over k channels C_j with $1 \leq j \leq k$. The bandwidths of these k channels will all be equal to the video consumption rate b. The total bandwidth required by the protocol will thus be equal to kb. Like other pagoda protocols, the FDPB protocol will partition each video into n equal-size segments of

duration $d = D/n$. These n segments will be broadcast at different frequencies over the k channels, each segment transmission occupying a *slot* of duration d.

Unlike previous pagoda protocols, the FDPB protocol requires all customers wanting to watch a video to wait for a fixed time interval $w = md$, where m is some integer $m \geq 1$. The protocol will use this delay to stretch the reception of the n segments of the video over a longer time interval. Previous pagoda protocols required segment S_i to be repeated at least once every i slots to ensure the continuity of the video. With the FDBP protocol, segment S_1 needs to be transmitted at least once every m slots to be always received before the customer starts watching the video. More generally, segment S_i will need to be transmitted at least once every $m + i - 1$ slots.

As shown on Figure 3, the FDPB protocol partitions each channel C_j into s_j *subchannels* in such a way that slot j of channel C_j belongs to its subchannel $j \pmod{s_j}$. Each subchannel has thus $1/s_j$ of the slots and $1/s_j$ of the bandwidth of channel C_j

The FDPB protocol also differs from previous pagoda protocols in the way it maps segments to channels. Unlike previous pagoda protocols, the FDBP protocol maps segments into subchannels in a strict sequential fashion. Thus the first segments of the video are mapped into subchannel 0 of channel C_1, the next segments into subchannel 1 of the same channel and so on until all s_1 subchannels of channel C_1 have been used. The process repeats itself for the subchannels of channels C_2 to C_k. As a result, the whole segment-to-channel mapping can be derived from its $k + 1$ parameters, namely

a) the number k of channels allocated to the video,
b) the ratio m between the customer waiting time and the segment duration d, and
c) the numbers $s_1, s_2, \ldots s_k$ of subchannels for each of the k channels.

We quickly found that the optimal number of subchannels for a given channel C_j depended on the periodicity at which the segments assigned to that channel had to be retransmitted. Let be the first segment assigned to channel C_j. As we saw earlier, segment S_i needs to be rebroadcast at least once every $m + i - 1$ slots. By trial and error, we found that the best mappings were always achieved when channel C_j was partitioned into $\sqrt{m+i-1}$ subchannels. Hence, it is convenient—but not necessary for the correctness of the protocol—to assume that the ratio m between the duration of the waiting period w and the duration d of a segment is a perfect square.

Consider for instance the case when $m = 9$. As Figure 3 indicates, channel C_1 will be partitioned into 3 subchannels. The first segment to be broadcast is segment S_1. Since $m = 9$, S_1 needs to be repeated at least once every 9 slots. Let us assign it to subchannel 0. Since subchannel 0 has one third of the slots of channel C_1, we can map up to three segments into it while ensuring that each of these three segments will be repeated once every 9 slots. These three segments will be segments S_1 to S_3.

Slot	**0**	**1**	**2**	**3**	**4**	**5**
Subchannel 0	✔			✔		
Subchannel 1		✔			✔	
Subchannel 2			✔			✔

Figure 3. A channel partitioned into 3 subchannels

Subchannel	**0**	**1**	**2**
First Segment	S_1	S_4	S_8
Last Segment	S_3	S_7	S_{12}

Figure 4. The first channel for $m=9$.

Subchannel	**0**	**1**	**2**	**3**	**4**
First Segment	S_{13}	S_{17}	S_{22}	S_{28}	S_{35}
Last Segment	S_{16}	S_{21}	S_{27}	S_{34}	S_{42}

Figure 5. The second channel for $m=9$.

The first segment to be transmitted by subchannel 1 will be segment S_4, which needs to be repeated at least once every $9 + 4 - 1 = 12$ slots. As a result, we will map four segments into subchannel 1. The first segment to be transmitted by subchannel 2 will thus be segment S_8. Since S_8 needs to be repeated every $9 + 8 - 1 = 16$ slots, we will map five segments into subchannel 2. As a result, channel C_1 will transmit a total of *twelve* segments.

The first segment to be broadcast by channel C_2 is segment S_{13}, which needs to be repeated at least once every $9 + 13 - 1 = 21$ slots. Since 20 is not a square and the closest square, $25 = 5^2$, channel C_2 will be partitioned into 5 subchannels. As Figure 5 shows, subchannel 0 will continuously retransmit segments S_{13} to S_{16} ensuring that each segment is repeated exactly once every 20 slots. Subchannel 1 will transmit segments S_{17} to S_{21} ensuring that segment S_{17} is repeated at least every $9 + 17 - 1 = 25$ slots. Subchannel 2 will transmit segments S_{22} to S_{27} to ensure that segment S_{22} is repeated at least every 30 slots and subchannel 3 will transmit segments S_{28} to S_{34} and subchannel 3 will transmit segments S_{28} to S_{34} ensuring that segment S_{28} is repeated at least every 36 slots. Finally subchannel 4 will repeat segments S_{35} to S_{42} ensuring that segment S_{35} is repeated at least every 43 slots. Hence, channel C_2 will broadcast 30 segments.

Table 1 summarizes the segment-to-channel mappings for up to seven channels. Allocating six channels to a video allows partitioning it into 2046 segments. The waiting time for the video will then be equal to 9/2046 of its duration, that is, less than 32 seconds for a two-hour video.

This is much better than the maximum waiting time of 44 seconds that can be achieved by the new pagoda broadcasting with the same number of channels. Broadcasting the same video over seven channels would reduce the waiting time to less than 12 seconds instead of 17 seconds for the new pagoda broadcasting protocol.

Table 1. Summary of the mappings for $m = 9$

Channel	Number of Subchannels	First Segment	Last Segment
C_1	3	S_1	S_{12}
C_2	5	S_{13}	S_{42}
C_3	7	S_{43}	S_{116}
C_4	11	S_{117}	S_{292}
C_5	17	S_{293}	S_{770}
C_6	28	S_{771}	S_{2046}
C_7	45	S_{2047}	S_{5477}

Table 2. Summary of the mappings for $m = 100$

Channel	Number of Subchannels	First Segment	Last Segment
C_1	10	S_1	S_{156}
C_2	16	S_{157}	S_{565}
C_3	26	S_{566}	S_{1650}
C_4	42	S_{1651}	S_{4563}
C_5	68	S_{4564}	S_{12418}
C_6	112	S_{12419}	S_{33684}
C_7	184	S_{33685}	S_{91321}

We could even reduce this delay by increasing m. The only problem with this approach is that this would partition each video into larger and larger numbers of smaller and smaller segments. As shown in Table 2, a FDPB protocol with $m = 100$ could pack 33,783 segments into six channels. The waiting time for the same two-hour video would then be given by $7200 \times 100/33{,}783 = 21.4$ seconds, that is less than half of the maximum waiting time for a new pagoda protocol with the same number of channels. Partitioning the video into 33,783 segments implies that each segment would now last 7,200/33,783 = 0.213 second. Assuming an average bandwidth of 5Megabits/second, this means that each segment would contain around 130 kilobytes of data. This still remains a reasonable record size and would not affect the performance of the disk subsystem of the video server.

We can derive a lower bound for the waiting time w of the FDBP protocol by computing the limit of this waiting time when m goes to infinity and k remains constant. Consider a video of duration D and assume that all customers are willing to wait w time units between the time they have ordered the video and the time they can start watching it. Let b represent the video consumption rate and Δt a small time interval at a location t within the video. Assuming that each customer STB starts downloading video data from the moment the video is ordered, the contents of this time interval will have to be broadcast at a minimum bandwidth $b/(t+w)$ where b is the video consumption rate.

Passing to the limit when Δt goes to 0, we see that the minimum bandwidth required to transmit the video is be given by

$$B_{\min} = \int_0^D \frac{b}{t+w}\,dt = b \log \frac{D+w}{w} \qquad (1)$$

From this equation, we can also derive the minimum waiting time that can be achieved when the broadcasting bandwidth is equal to k times the video consumption rate

$$w_{\min} = \frac{D}{e^k - 1} \qquad (2)$$

Hence the minimum waiting time that can be achieved with a bandwidth equal to six times the video consumption rate is given by

$$w_{\min} = \frac{7200}{e^6 - 1} = 17.9\text{s} \qquad (3)$$

Figure 6 shows the waiting times achieved by the FDBP for selected values of m between 4 and 100. All bandwidths are expressed as multiples of the video consumption rate b and all waiting times are expressed as fractions of the video duration D. The dotted curve at the top represents the maximum waiting times achieved by the new pagoda protocol while the tick solid curve at the bottom represents the lower bound of equation (2).

As one can see, the FDPB protocol achieves lower maximum waiting times than new pagoda broadcasting at every bandwidth. In particular, with $m = 100$, a bandwidth equal to five times the video consumption suffices to bring the waiting time under 0.81 percent of the duration of the video, that is, less than one minute for a two-hour video.

Our FDPB protocol does not have the same advantage over the polyharmonic broadcasting protocol and the GAB protocol. As shown on Figure 7, the polyharmonic broadcasting protocol with $m = 16$ (that is, $w = 16d$) provides waiting times that are very close to the lower bounds derived from equation (1). This is not the case for the FDPB protocol, which performs significantly worse. The superior performance of the polyharmonic broadcasting comes however at a price: achieving a waiting time of 20 seconds for a two-hour video requires partitioning the video into 5,760 segments and broadcasting each of these segments on a separate channel.

IV. Restricting the Client Bandwidth

Like most other broadcasting protocols, the FDBP protocol assumes that the set-top box (STB) can and will simultaneously receive data from the k channels on which the various segments of the video are broadcast. This requirement complicates the design of the STB and increases its cost.

One possible approach to this problem is to restrict the STB receiving bandwidth to a given multiple $k' < k$ of the video consumption rate. For instance, the *skyscraper broadcasting* protocol [3] never requires the customer STB to receive data from more than two channels at the same time.

This approach has a major drawback, namely a very significant increase in the server bandwidth required to

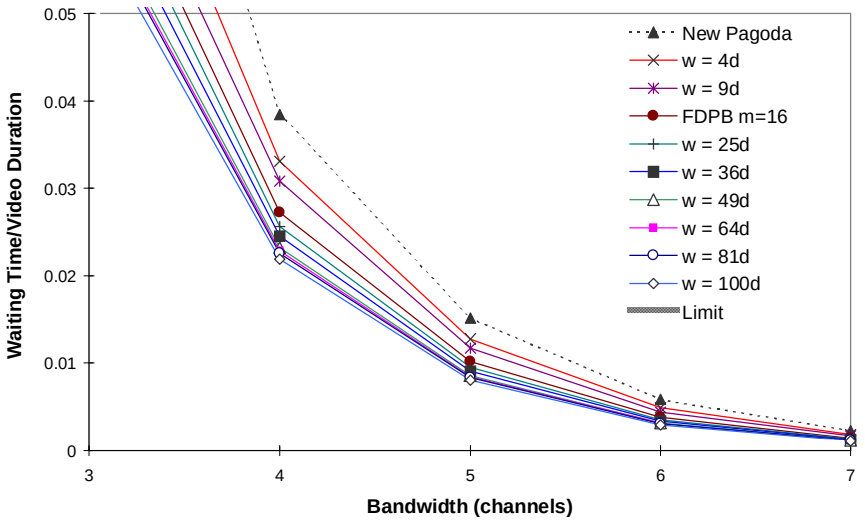

Figure 6. Waiting times achieved by the FDPB protocol for different values of *m*.

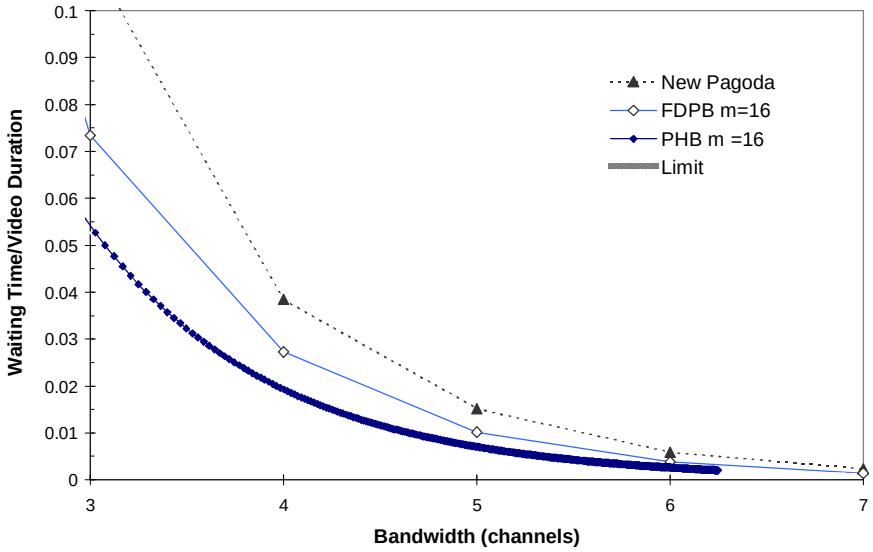

Figure 7. How the FDPB protocol compares to new pagoda broadcasting protocol and polyharmonic broadcasting

distribute the videos. Hence, the potential savings in STB costs achieved by skyscraper broadcasting would require bigger, more expensive video servers and a costlier network infrastructure.

We propose here a less radical implementation of the same concept, namely, reducing the client bandwidth requirements of an existing protocol to two or three concurrent channels. As we will see, this approach will result in very moderate increases of the server bandwidth. Consider the case of a FDPB protocol with m = 100 that restricts the server client bandwidth to 2 channels. As shown in Table 3, the segment-to-subchannel mappings of the two first channels are unchanged. The first mappings to be affected are those of channel C_3 as the STB must now wait until it has received all data from the first channel before starting to receive data from the first channel. The last segment broadcast by the first channel is segment S_{156}. It is broadcast along with segments S_{134} to S_{155} by subchannel 0 once every 230 slots because their broadcasting period must be a multiple of the number of subchannels in the first channel. The first segment broadcast by channel C_3 is segment S_{566}. Recalling that the customer waiting time is equal to 100 slots, we see that segment S_{566} must now be broadcast at least once every 566 + 99– 230 =435 slots. Similarly segment S_{567} has now to be broadcast at least once every 567 + 99– 230 =436 slots and so on. As a result, channel C_3 will now be partitioned into $\sqrt{435} \approx 21$ subchannels and will broadcast segments S_{566} to S_{1268}.

Figure 8 presents the waiting times achieved by a FDPB protocol restricting the STB bandwidth to two channels (k' = 2)

Table 3. Summary of the mappings for a FDBP limiting the STB bandwidth to two channels (m = 100)

Channel	Number of Subchannels	First Segment	Last Segment
C_1	10	S_1	S_{156}
C_2	16	S_{157}	S_{565}
C_3	21	S_{566}	S_{1268}
C_4	27	S_{1269}	S_{2486}
C_5	36	S_{2487}	S_{4617}
C_6	47	S_{4618}	S_{8298}
C_7	62	S_{8299}	S_{14595}

and compares them to those achieved by skyscraper broadcasting and new pagoda broadcasting. As before, all bandwidths are expressed as multiples of the video consumption rate b and all waiting times are expressed as fractions of the video duration D. We can see that our protocol performs much better than skyscraper broadcasting but significantly worse than the new pagoda broadcasting protocol. Given a server bandwidth equal to six times the video consumption rate, skyscraper broadcasting can only achieve a maximum waiting time of 4 minutes and 27 seconds for a two-hour video. Our FDPB protocol can reduce this delay to 87 seconds, that is, slightly more than twice the 42 seconds achieved by the new pagoda broadcasting protocol. Both protocols perform significantly worse than the unrestricted version of the FDPB, which can achieve a waiting time of 21 seconds with the same server bandwidth.

Two factors can explain the large gap between the performances of skyscraper broadcasting and the restricted version of our protocol. First, the FDPB broadcasting protocol uses much more efficient segment-to-slot mappings than skyscraper broadcasting for its first two channels. As these mappings remain unchanged when the STB bandwidth gets limited to two times the video consumption rate, this gives a definitive edge to our protocol. Second, the skyscraper broadcasting has the objective of reducing both the STB bandwidth and the size of the STB buffer. These were both important objectives when the skyscraper broadcasting was proposed in 1997, as disk drives capable of storing large amounts of video data were still expensive. Nowadays, it is virtually impossible to buy a new disk drive that cannot contain at least four hours of video data. Hence reducing the size of the STB buffer is not as important today as it was then.

V. CONCLUSION

We have presented a new pagoda broadcasting protocol for VOD that requires all customers to wait for the same amount of time before watching the video they have ordered. Our fixed-delay pagoda broadcasting protocol (FDBP) uses this delay to reduce the bandwidth required to transmit the first minutes of each video. As a result, it

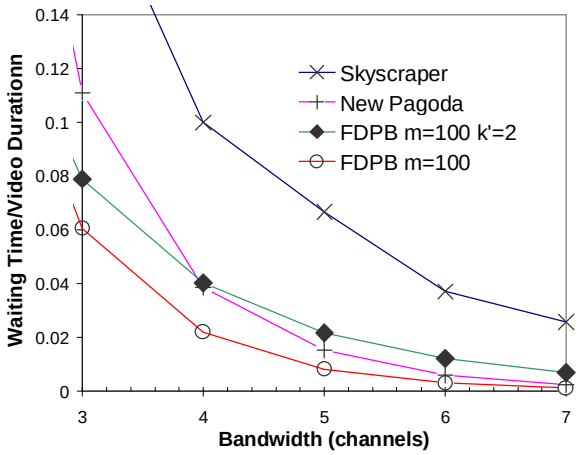

Figure 8. How a FDPB protocol restricting STB bandwidth to two channels compares to the new pagoda broadcasting protocol and skyscraper broadcasting.

provides the lowest waiting times of all protocols using segments of equal duration and channels of equal bandwidth. In addition, its performance is not very far from the theoretical minimum. We have also shown how we can modify our protocol to restrict the STB receiving bandwidth to two times the video consumption rate.

More work is needed to improve the performance of the protocol when it is used to transmit variable bit-rate video using a constant transmission rate [7] and adjust the transmission frequencies of the segments that always arrive one or more slots ahead of time because of the constant transmission rate.

REFERENCES

[1] Aggarwal, C. C., J. L. Wolf, and P. S. Yu, "A permutation-based pyramid broadcasting scheme for video-on-demand systems," *Proc. ICMCS Conf.*, pp. 118–126, June 1996.

[2] Carter, S. W., D. E. Long and J.-F. Pâris. "Video-on-demand broadcasting protocols," In *Multimedia Communications: Directions and Innovations* (J. D. Gibson, Ed.), Academic Press, San Diego, 2000, pp. 179–189.

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

[4] Juhn, L., and L. Tseng, "Harmonic broadcasting protocols for video-on-demand service," *IEEE Trans. on Broadcasting*, 43:268–271, Sep. 1997.

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

[6] Hu, A., I. Nikolaidis, and P. van Beek. "On the design of efficient video-on-demand broadcast schedules," *Proc. 7th Int. MASCOTS Symp.*, pp. 262–269, Oct. 1999.

[7] McManus, J. M., and K. W. Ross, "Video-on-demand over ATM: constant rate transmission and transport," *IEEE Journal on Selected Areas in Communication*, 14(6):1087–1098, 1996.

[8] Pâris, J.-F., S. W. Carter and D. D. E. Long, "A low bandwidth broadcasting protocol for video on demand," *Proc. 7th ICCCN Conf.*, pp. 690–697 Oct. 1998.

[9] Pâris, J.-F. "A simple low-bandwidth broadcasting protocol for video on demand," *Proc. 87th Int. ICCCN Conf.*, pp. 690–697, Oct. 1999.

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

[11] Wong, J. W., "Broadcast delivery," *Proc. IEEE*, 76(12):1566–1577, Dec. 1988.