A variable bandwidth broadcasting protocol for video-on-demand

Jehan-François Pâris[a1], Darrell D. E. Long[b2]

[a]Department of Computer Science, University of Houston, Houston, TX 77204-3010
[b]Department of Computer Science, University of California, Santa Cruz, CA 95064

ABSTRACT

We present the first broadcasting protocol that can alter the number of channels allocated to a given video without inconveniencing the viewer and without causing any temporary bandwidth surge. Our *variable bandwidth broadcasting* (VBB) protocol assigns to each video a minimum number of channels whose bandwidths are all equal to the video consumption rate. Additional channels can be assigned to the video at any time to reduce the customer waiting time or retaken to free server bandwidth. The cost of this additional flexibility is quite reasonable as the bandwidth requirements of our VBB fall between those of the *fast broadcasting* protocol and the *new pagoda broadcasting* protocol.

1. INTRODUCTION

Broadcasting protocols offer the best solution for the successful deployment of metropolitan video-on-demand (VOD) services because they provide a way to distribute very popular videos in an efficient fashion and these so-called "hot" videos are expected to account for the majority of customer requests. Rather than reacting to individual viewer requests, broadcasting protocols distribute the contents of videos according to a fixed schedule guaranteeing that all customers will receive these contents on time. As a result, the number of customers watching a given video does not affect the server workload.

All recent VOD broadcasting protocols derive in some way from Viswanathan and Imielinski's *pyramid broadcasting protocol*[12] and, like it, they assume that most, if not all, users will watch each video in a strictly sequential fashion. They also require customers to be connected to the service through a "smart" set-top box (STB) capable of (a) receiving data at rates exceeding the video consumption rate and (b) storing locally the video data that arrive out of sequence. In the current state of storage technology, this implies having a disk drive in each STB, a device already present in the so-called digital VCR's offered by TiVo[10], Replay[9] and Ultimate TV[11].

We can classify VOD broadcasting protocols into two broad groups according to the way they broadcast the various segments of each video. Harmonic broadcasting protocols[4, 6] assign a separate data stream to each segment of a video. The very low bandwidth requirements of these protocols result from the fact that they transmit each segment at the minimum bandwidth required to ensure its on-time delivery to a viewer watching the video in a sequential fashion. The second group of broadcasting protocols is comprised of protocols that broadcast each video over a fixed number k of data channels whose bandwidths are normally equal to the video consumption rate b. As we will see, they achieve their bandwidth savings through the use of time division multiplexing.

Research on VOD broadcasting protocols has been focused on two main axes, namely, (a) designing protocols that require less bandwidth to achieve the same customer waiting time, and (b) designing protocols that switch to a reactive approach when the number of user requests is too low to justify maintaining a deterministic broadcasting schedule. So

[1] paris@cs.uh.edu; supported in part by the Texas Advanced Research Program under grant 003652-0124-1999 and the National Science Foundation under grant CCR-9988390.
[2] darrell@cse.ucsc.edu; supported in part by the National Science Foundation under grant CCR-9988363.

Multimedia Computing and Networking 2003, Ragunathan Rajkumar, Editor, Proceedings of
SPIE-IS&T Electronic Imaging, SPIE Vol. 5019 (2003) © 2003 SPIE-IS&T · 0277-786X/03/$15.00

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

Figure 1. The first three channels for the FB protocol

far, scant attention has been given to the problem of dynamically altering the bandwidth allocated to a popular video to reflect changes in bandwidth availability. For instance, the operator of a video service might want to reduce the number of channels allocated to some videos to make space for a new offering or to increase the number of channels allocated to a video when extra bandwidth is available. Decisions of this kind are analogous to the movie scheduling decisions made daily by managers of theater multiplexes: they may want sometimes to show the same movie on two or three different screens or have at other times two movies shown at alternate times in the same auditorium.

We present a broadcasting protocol that can alter the number of channels allocated to a given video without inconveniencing any customer and without causing any temporary bandwidth surge. Our *variable bandwidth broadcasting* (VBB) protocol assigns to each video a minimum number k_{min} of channels whose bandwidths are all equal to the video consumption rate b. Additional channels can be assigned to the video at any time in order to reduce the customer waiting time or withdrawn in order to free server bandwidth. The cost of this additional flexibility is quite reasonable as the bandwidth requirements of our VBB fall between those of the *fast broadcasting* protocol[5] and the *new pagoda broadcasting* protocol[7].

The remainder of the paper is organized as follows. Section 2 reviews relevant previous work on broadcasting protocols. Section 3 defines the problem we want to address and introduces our variable bandwidth broadcasting protocol. Section 4 compares its performance to those of extant protocols and Section 5 has our conclusions.

2. PREVIOUS WORK

Earlier video distribution protocols attempted to reduce server bandwidth either by batching together several requests[2] or by accelerating the video playback rate of new requests to let them catch up with previous transmissions[3]. Viswanathan and Imielinski[12] proposed in 1996 a better solution. Their *pyramid broadcasting* protocol required special customer set-top boxes (STBs) (a) capable of receiving data at data rates exceeding the video consumption rate and (b) having enough buffer space to store one hour of video data. Their original proposal has been followed by several more recent schemes requiring less server bandwidth to achieve the same customer waiting times. We will only mention here those protocols that are directly relevant to our work. The reader interested in a more comprehensive review of broadcasting protocols for VOD may want to consult reference[1].

The simplest broadcasting protocol is Juhn and Tseng's *fast broadcasting* (FB) protocol[5]. The FB protocol allocates to each video k data channels whose bandwidths are all equal to the video consumption rate b. It then partitions each video into $2^k - 1$ segments, S_1 to $S_{2^{k-1}}$, of equal duration d. As Figure 1 indicates, the first channel continuously rebroadcasts segment S_1, the second channel transmits segments S_2 and S_3, and the third channel transmits segments S_4 to S_7. More generally, channel 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 that segment while their STB starts downloading data from all other channels. Hence the maximum customer waiting time is equal to the duration of a segment. Define a *slot* as a time interval equal to the duration of a segment. To prove the correctness of the FB protocol, we need only to observe that each segment i with $1 \le i \le 2^{k-1}$ is rebroadcast at least once every i slot. Then any client STB starting to receive data from all broadcasting channels will always receive all segments on time.

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

Figure 2. An NPB protocol with three channels

Channel	***First Segment***	***Last Segment***
C_1	S_1	S_{12}
C_2	S_{13}	S_{42}
C_3	S_{43}	S_{116}

Figure 3. The first three channels for the FDPB protocol with $m = 9$

The *new pagoda broadcasting* (NPB)[7] protocol improves upon the FB protocol by using a more complex segment-to-channel mapping. As seen on Figure 2, the NPB protocol can pack nine segments into three channels while the FB protocol can only pack seven of them. 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.

Neither the FB protocol nor the NPB protocol require customer STBs to wait for any minimum amount of time. As a result, there is no point in requiring customer STBs to start downloading data while customers are still waiting for the beginning of the video. The newer *fixed-delay pagoda broadcasting* (FDPB) protocol[8] requires all users to wait for a fixed delay *w* before watching the video they have selected. This waiting time is normally a multiple *m* of the segment duration *d*. As a result, the FDPB protocol can partition each video into much smaller segments than either FB or PB with the same number of channels. Since these smaller segments can be packed much more effectively into the *k* channels assigned to the video, the FDPB protocol achieves smaller customer waiting times than FB and PB protocols with the same number of channels.

Figure 3 summarizes the segment-to-channel mappings of a FDPB protocol requiring customers to wait for exactly nine times the duration of a segment. This allows the protocol to map 116 segments into three channels and achieve a deterministic waiting time of 9/116 of the duration of the video, that is, slightly less than ten minutes for a two-hour video.

3. THE VBB PROTOCOL

Our design objectives were straightforward. First, we wanted to design a flexible broadcasting protocol that could increase or decrease the number of channels allocated to a given video without causing any interruption of service and without requiring any temporary bandwidth surge. In other words, the transition between the old and the new segment to channel mapping had to be seamless. We also wanted our protocol to make an efficient use of its bandwidth and achieve minimum customer waiting times comparable to those of the most recent broadcasting protocols.

This second criterion excluded some simple solutions. Consider for instance the FB protocol described in the previous section. As seen on Figure 4, we could easily collapse its first two channels into a single channel broadcasting segments S_1, S_2 and S_3. This would however triple the customer waiting time, which is not acceptable.

A better approach is to start with a given number of channels k_{min}. We can then split each segment S_i into two segments, $S_{\backslash 1}$ and S_{i2}, of equal duration $d/2$ and add an extra channel that will continuously broadcast the half-segment S_{11}. This process can be repeated as many times as needed with each step adding one extra channel and halving the maximum customer waiting time.

Old Second Channel	S_1	S_2	S_3	S_1
Old Third Channel	S_4	S_5	S_6	S_7

Figure 4. Collapsing the first two channels of the FB protocol

First Channel	S_{11}	S_{12}	S_{11}	S_{12}	S_{11}	S_{12}	S_{11}	S_{12}
Second Channel	S_{21}	S_{22}	S_{31}	S_{32}	S_{21}	S_{22}	S_{31}	S_{32}
Third Channel	S_{41}	S_{42}	S_{51}	S_{52}	S_{61}	S_{62}	S_{71}	S_{72}

(a) Splitting each segment into two smaller segments

New Channel	S_{11}	S_{11}	S_{11}	S_{11}	S_{11}	S_{11}	S_{11}	S_{11}
Old First Channel	S_{11}	S_{12}	S_{11}	S_{12}	S_{11}	S_{12}	S_{11}	S_{12}
Old Second Channel	**S_{21}**	S_{22}	S_{31}	S_{32}	**S_{21}**	S_{22}	S_{31}	S_{32}
Old Third Channel	**S_{41}**	S_{42}	S_{51}	S_{52}	S_{61}	S_{62}	S_{71}	S_{72}

(b) Adding a new channel exclusively broadcasting S_{11}: redundant segment transmissions are represented in gray over white while segments that will not always arrive on time are in bold over gray

New Channel	S_{11}	S_{11}	S_{11}	S_{11}	S_{11}	S_{11}	S_{11}	S_{11}
Old First Channel	**S_{21}**	S_{12}	**S_{21}**	S_{12}	**S_{21}**	S_{12}	**S_{21}**	S_{12}
Old Second Channel	**S_{41}**	S_{22}	S_{31}	S_{32}	**S_{41}**	S_{22}	S_{31}	S_{32}
Old Third Channel	S_{81}	S_{42}	S_{51}	S_{52}	S_{61}	S_{62}	S_{71}	S_{72}

(c) Updating the segment to slot mapping of the protocol

Figure 5. Adding an extra channel to the FB protocol

Figure 5 describes one step of the process for the first three channels of the FB policy. As we can see, we will have to make some adjustments to the segment-to-channel mapping of the protocol. First, consider the old channel 1. Like all other channels, it is now partitioned into twice as many slots as before. Fifty percent of these smaller slots are now free, as their retransmissions of half-segment S_{11} are redundant. Second, some of the new segments will have to be broadcast more frequently. Consider for instance segment S_{21}. It belonged to the old segment S_2, which was rebroadcast once every two old slots, that is, once every four of the new slots. It is now the third segment of the video and has thus to be retransmitted once every three new slots. More generally each and every new segment S_{i1} will now have to be retransmitted at least once every $2i - 1$ new slots while each and every new segment S_{i2} will have to be retransmitted at least once every $2i$ new slots. This will result in a cascade of new allocations:

- Segment S_{21} will now be broadcast by the old channel 1 and occupy the slots that were previously broadcasting segment S_{11}. As a result, it will now be rebroadcast once every two new slots.
- The slots freed by that move will now broadcast segment S_{41}, which will now be rebroadcast once every four slots.
- The slots previously occupied by segment S_{41} are now free. If there was a fourth channel, we could use these free slots to broadcast the first segment of that fourth channel, that is segment S_{81} (represented in bold gray).

Table 1. Segment-to-subchannel mapping of the VBB protocol

Channel	*Subchannel*	*Segments*
1	1	1
2	1	2
	2	4 and 5
3	1	3
	2	6 and 7
	3	8 and 9
4	1	10 to 12
	2	13 to 16
	3	17 to 21
5	1	22 to 25
	2	26 to 30
	3	31 to 36
	4	37 to 43
	5	44 to 51
6	1	52 to 58
	2	59 to 66
	3	67 to 75
	4	76 to 85
	5	86 to 97
	6	98 to 110
	7	111 to 125
7	1	126 to 136
	2	137 to 148
	3	149 to 161
	4	162 to 175
	5	176 to 190
	6	191 to 207
	7	208 to 225
	8	226 to 245
	9	246 to 267
	10	268 to 291
	11	292 to 317

More generally the slots that were previously broadcasting the first segment of every channel, that is, segments S_{11}, S_{21}, S_{41}, … will now broadcast the first segment of the *next* channel that is, segments S_{21}, S_{41}, S_{61}, …. In other words, the

lowest numbered segment of each channel will move up one channel and occupy the slots that were previously occupied by the lowest numbered segment of that channel.

This will be a very orderly process that will only involve one segment per channel. This would not be the case if we tried the same approach with the NPB protocol. First, the NPB protocol broadcasts many more segments at their minimum bandwidth than the FB protocol, which means that many more of the split segments would have to be reassigned. Second, the irregular segment-to-slot mappings used by the NPB protocol would make this task more complex.

One possible way to eliminate these segment reassignments would be not to transmit any segment at its minimum bandwidth and ensure instead that each segment S_i is repeated at least once every $i - 1$ slots. This would unfortunately reduce the number of segments that can be broadcast over the k_{min} channels allocated to the video and result in an unacceptable increase of the customer waiting time. We decided instead to build our *variable bandwidth broadcasting* protocol around a more efficient segment-to-slot mapping that only required the reallocation of five segments all located in the first three channels assigned to the video.

Table 1 describes in some detail this mapping for the first seven channels. As one can see, each of these seven channels is partitioned into a variable number s_j of subchannels. We will allocate to each subchannel an equal number of equally spaced slots of the channel it belongs to. Hence the first subchannel of channel 2 will occupy all even slots of that channel while the other subchannel will have all odd slots. More generally, subchannel x of channel j will contain all slots z such that $z \bmod s_j = x - 1$, that is $1/s_j$ of the slots of channel j.

The segment-to slot mappings of the first three channels are identical to these of a NPB protocol with three channels. As a result, segments S_1, S_2, S_3, S_4 and S_6 are broadcast at their minimum broadcasting frequency, that is, once every slot for S_1, once every two slots for S_2, once every three slots for S_3 and so forth. The remaining channels are mapped according to a scheme similar to that used by the FDPB[8] protocol.

Consider a channel j with $j \geq 4$ and assume that the lowest numbered segment to be broadcast by the channel is segment S_i. This channel will be partitioned into $s_j = \text{round}(\sqrt{i})$ subchannels that will be successively filled with the segments assigned to that channel, starting with subchannel 1 and ending with subchannel s_j.

While other VOD broadcasting protocols try to find the tightest segment-to-slot mapping guaranteeing that all segments will always arrive on time, the VBB protocol also ensures that adding an extra channel will never result in a segment reassignment in any channel j with $j \geq 4$. To meet this new requirement, each S_i with $i \geq 10$ will be broadcast at least once every $i - 1$ slots instead of once every i slots.

We will enforce this last condition by limiting the number of segments broadcast by each subchannel. Consider a channel j having s_j subchannel and let S_i be the lowest numbered segment to be broadcast by subchannel l. To ensure that S_i will always be repeated at least once every $i - 1$ slots, the number of segments n_l repeated by subchannel l will always satisfy the inequality $n_l < i/s_j$.

Consider for instance the segment-to-slot mapping of the fourth channel. Since S_{10} is the lowest numbered segment to be broadcast by that channel and $\sqrt{10} \approx 3$, we will partition it into 3 subchannels, each having one third of the channel slots. The first subchannel will continuously repeat segments S_{10} to S_{12} ensuring that each segment will be repeated once every 9 slots, the second subchannel will do the same with segments S_{13} to S_{16} ensuring that each segment will be repeated once every 12 slots and the third subchannel will handle segments S_{17} to S_{21} ensuring that each segment will be repeated once every 15 slots.

Let us consider now what will happen when we add an extra channel and split each segment S_i into two smaller segments S_{i1} and S_{i2}. Since all segments S_i with $i \geq 10$ are broadcast at least once every $i - 1$ slots, the segment-to-slot mappings of the channels broadcasting these segments will not be affected by the process. We only need to consider the changes to be brought to the first three channels.

As Figure 6 shows, the sole segments whose mappings will be modified are segments S_{11}, S_{21}, S_{31}, S_{41} and S_{61}:

- The new channel will continuously broadcast segment S_{11}.

First Channel	S_{11}	S_{12}	S_{11}	S_{12}	S_{11}	S_{12}	S_{11}	S_{12}	S_{11}	S_{12}	S_{11}	S_{12}
Second Channel	S_{21}	S_{22}	S_{41}	S_{42}	S_{21}	S_{22}	S_{51}	S_{52}	S_{21}	S_{22}	S_{41}	S_{42}
Third Channel	S_{31}	S_{32}	S_{61}	S_{62}	S_{81}	S_{82}	S_{31}	S_{32}	S_{71}	S_{72}	S_{91}	S_{89}

(a) Splitting each segment into two smaller segments

New Channel	S_{11}	S_{11}	S_{11}	S_{11}	S_{11}	S_{11}	S_{11}	S_{11}	S_{11}	S_{11}	S_{11}	S_{11}
Old First Channel	S_{11}	S_{12}	S_{11}	S_{12}	S_{11}	S_{12}	S_{11}	S_{12}	S_{11}	S_{12}	S_{11}	S_{12}
Old Second Channel	**S_{21}**	S_{22}	**S_{41}**	S_{42}	**S_{21}**	S_{22}	S_{51}	S_{52}	**S_{21}**	S_{22}	**S_{41}**	S_{42}
Old Third Channel	**S_{31}**	S_{32}	**S_{61}**	S_{62}	S_{81}	S_{82}	**S_{31}**	S_{32}	S_{71}	S_{72}	S_{91}	S_{92}

(b) Adding a new channel exclusively broadcasting S_{11}: redundant segment transmissions are represented in gray over white while segments that will not always arrive on time are in bold over gray

New Channel	S_{11}	S_{11}	S_{11}	S_{11}	S_{11}	S_{11}	S_{11}	S_{11}	S_{11}	S_{11}	S_{11}	S_{11}
Old First Channel	S_{21}	S_{12}	S_{21}	S_{12}	S_{21}	S_{12}	S_{21}	S_{12}	S_{21}	S_{12}	S_{21}	S_{12}
Old Second Channel	S_{31}	S_{22}	S_{61}	S_{42}	S_{31}	S_{22}	S_{51}	S_{52}	S_{31}	S_{22}	S_{61}	S_{42}
Old Third Channel	S_{41}	S_{32}	—	S_{62}	S_{81}	S_{82}	S_{41}	S_{32}	S_{71}	S_{72}	S_{91}	S_{92}

(c) Updating the segment to slot mapping of the protocol

Figure 6. Adding an extra channel to the VBB protocol.

- The slots that segment S_{11} previously occupied in the old first segment will be reassigned to segment S_{21}, which will now be rebroadcast once every two slots.
- The slots that segment S_{21} previously occupied in the old second segment will be reassigned to segment S_{31}, which will now be rebroadcast once every four slots.
- The slots that segment S_{31} previously occupied in the old third segment will be reassigned to segment S_{41}, which will now be rebroadcast once every six slots.
- The slots that segment S_{41} previously occupied in the old second segment will be reassigned to segment S_{61}, which will now be rebroadcast once every eight slots.
- The slots that segment S_{61} previously occupied in the old third segment will remain free.

Note that the resulting segment-to-slot mapping will not contain any instance of a segment S_i with $i > 1$ that is not broadcast at least once every $i - 1$ slots. Hence further additions of extra channels will not require the modifications of the mappings of *any* segment but segment S_{11}.

A great advantage of this property is that customers who request the video while a new channel is being added will see much more quickly the new segment-to-slot mapping and will have their maximum waiting time reduced much faster. Channel removals will also take less time.

Setting up the protocol for a given video will require specifying the minimum number of channels k_{min} that will be permanently assigned to the video. For instance, selecting $k_{min} = 4$ will mean that video will always be broadcast on at

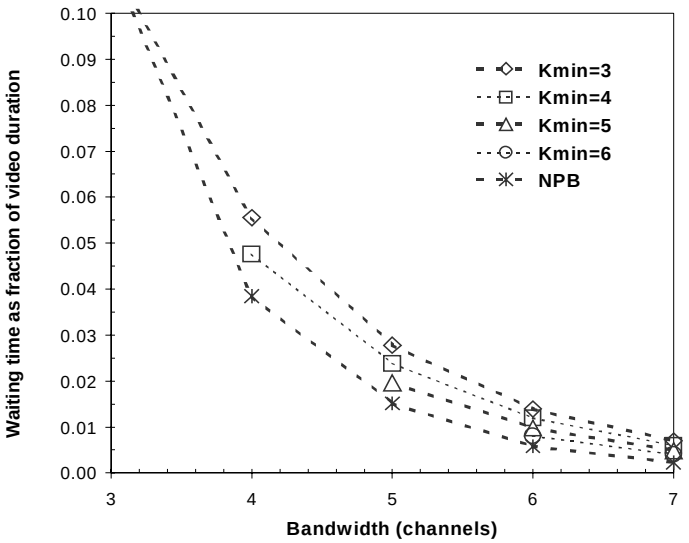

Figure 7. How the VBB protocol compares with other broadcasting protocols

least 4 channels and partitioned into at least 21 segments. As a consequence, the maximum customer wait will be equal to 1/21 of the video duration, that is, slightly less than six minutes for a two-hour video. Each extra channel added to the four base channel will halve this maximum waiting time. Hence we will need three additional channels to bring this maximum waiting time below 45 seconds.

There is a clear tradeoff in the selection of the best k_{min} for a video: while low values of k_{min} will give the service provider more flexibility in assigning channels to the video, mappings based on higher values of k_{min} will use the available bandwidth more efficiently. Going back to our previous example, we can see that selecting $k_{min} = 6$ would allow us to partition the video in at least 125 segments and achieve a waiting time of slightly less than a minute for a two-hour video with only six channels.

Let us briefly describe the procedure for taking away a channel. Two cases have to be considered. First, if the video has two or more channels over its minimum channel allocation k_{min}, we can always remove the channel transmitting the first segment of the video without affecting the segment-to-slot mappings of the other channels. Second, if the video only has one additional channel above its minimum channel allocation, we will need to proceed by stages:

a) The server will stop first scheduling instances of segment S_{11}in the slots of the first channel that do not correspond to an instance of S_{21} in the second channel. This will immediately increase the maximum customer waiting time to two slots.

b) Eight slots after that, the STB will move its transmissions of segment S_{61} to the free slots of the fourth channel; as a result, segment S_{61} will now be retransmitted every twelve slots.

c) When this is done, the STB will move its transmissions of segment S_{41} to the slots of the third channel that were previously transmitting segment S_{61}; as a result, segment S_{41} will now be retransmitted every eight slots.

d) The STB will move its transmissions of segment S_{31} to the slots of the fourth channel that were previously transmitting segment S_{41}; as a result, segment S_{31} will now be retransmitted every six slots.

e) The STB will move its transmissions of segment S_{21} to the slots of the third channel that were previously transmitting segment S_{31}; as a result, segment S_{21} will now be retransmitted every four slots.

f) The STB will move its transmissions of segment S_{11} to the slots of the second channel that were previously transmitting segment S_{21}.

g) The STB will then release the first channel of the video.

3. PERFORMANCE ANALYSIS

Figure 7 compares the maximum customer waiting times achieved of our VBB protocol for k varying between four and seven channels with those achieved by three other VOD broadcasting protocols. The protocols against which we compared the bandwidth requirements of the VBB protocol are:

a) the *fast broadcasting* (FB) protocol,

b) the *new pagoda broadcasting* (NPB) protocol, and

c) a *fixed-delay pagoda broadcasting* (FDPB) with $m = 3$.

All waiting times are expressed as fractions of the video duration D. So, a value of 0.05 would correspond to a maximum waiting time of six minutes for a two-hour video.

As expected, the performance of the VBB protocol is strongly affected by its minimum bandwidth parameter k_{min}. Lower values of k_{min} always result in longer waiting times for the same video bandwidth. A VBB protocol with $k_{min} = 3$ only achieves waiting times slightly below those achieved by the FB protocol while VBB protocols with $k_{min} > 3$ achieve better waiting times. In particular, selecting k_{min} equal to 6 will bring the performance of the VBB protocol very close to that of the NPB protocol.

We can also see that the FDPB protocol always produces the lowest maximum customer waiting time. This is because this protocol requires all customers to wait for the same fixed delay before watching the video. Hence its maximum and its average customer waiting time are the same. We compared maximum waiting times because they matter the most to customers. A comparison of average waiting times would have led to a different ranking of the protocols.

Another issue to be considered is how the protocol would fare against an optimal protocol achieving the minimum waiting time for a given server bandwidth. To compute this minimum, let us consider a video of duration D being broadcast in such a way that all customers requesting the video wait for w time units before starting the video. Let Δt represent a small time interval starting at a location t within the video. To avoid STB underflow, the contents of this time

interval must be broadcast at a minimum bandwidth $b/(t+w)$ where b is the video consumption rate. Summing over all intervals as Δt approaches 0, we see that the bandwidth required to transmit the video is given by

$$\int_0^D \frac{b}{t+w} dt = b(\ln(D+w) - \ln w) = -b \ln \frac{D+w}{w}$$

Assume now that we want this bandwidth to be equal to a fixed multiple k of the video consumption rate b. The waiting will then be the solution of the equation

$$-b \ln \frac{D+w}{w} = kb$$

and we will have

$$w = \frac{D}{e^k - 1} \tag{1}$$

Figure 8 compares the maximum waiting times achieved by our protocol with those achieved by the optimal protocol. While the relative gap between the waiting times achieved by two protocols increase with k, their absolute differences are also becoming less relevant.

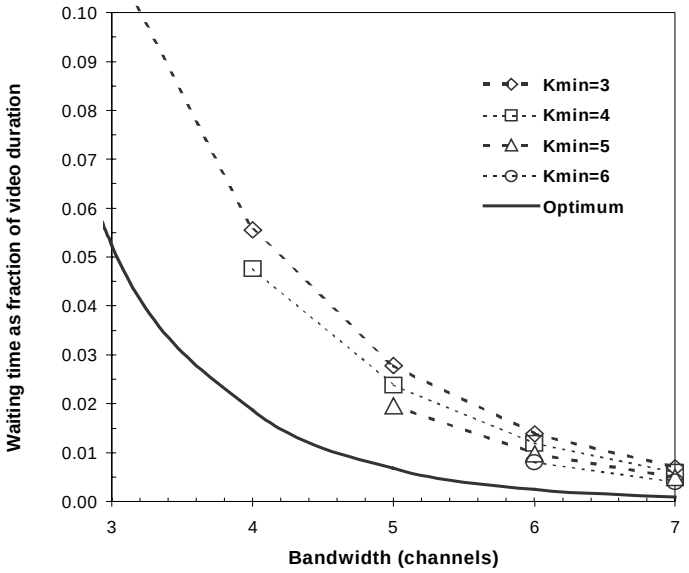

Figure 8. How the VBB protocol compares with the theoretical optimum

A last aspect of the performance of a broadcasting protocol is its maximum disk storage cost. To derive this, we will assume that the STB uses an *eager fetch* policy and always downloads the first instance of any new segment even when the segments will be repeated before being actually viewed by the customer.

To compute the maximum storage, we observe that the STB will receive data from the video server at a variable rate while the customer will consume the data at a fixed rate. Moreover the rate at which the STB will receive data from the server is a decreasing function of time as the STB will download data from less and less channels as the video progresses. The maximum storage requirement will occur when the STB stops receiving more data than it is consuming, which happens when the STB is done with the next to last channel. At that time, the STB will have in its buffer:

a) all the segments of the last channel it has already received and

b) the segments of the next to last channel that have not yet been viewed by the customer.

By adding these two terms, we obtain the maximum disk storage cost of the protocol. Consider for instance the case of a VBB protocol with $k = k_{min} = 7$. As we can see from Table 1, the protocol will broadcast 317 segments over its 7 channels and segment S_{125} will be the last segment broadcast by channel 6. Observing that this last segment is broadcast exactly once every 105 slots, we know that the protocol maximum disk storage cost will be 105 + (125 – 105) = 125 segments, that is 39.4 percent of the video. We found that this percentage remained rather stable when k varied between 4 and 8 channels as the VBB protocol never required the customer STB to hold more than 43 percent of the video in its local buffer. This is slightly less than the storage costs of most other protocols, which typically require the STB to hold up to 50 percent of each video in its local buffer.

4. CONCLUSIONS

We have presented a first broadcasting protocol that can alter the number of channels allocated to a given video without inconveniencing the viewer and without causing any temporary bandwidth surge. Our *variable bandwidth broadcasting* (VBB) protocol assigns to each video a minimum number k_{min} of channels whose bandwidths are all equal to the video consumption rate. we have seen how we could assign additional channels to the video at any time to reduce the customer waiting time and retaken later to free server bandwidth. We found the cost of this additional flexibility to be

quite reasonable as the bandwidth requirements of our VBB fall between those of the *fast broadcasting* protocol and the *new pagoda broadcasting* protocol. In addition, our VBB protocol never requires the customer STB to hold more than 44 percent of the video in its local buffer, which is slightly less than most other broadcasting protocols.

Several extensions of the VBB protocol can be contemplated. The most attractive one is a dynamic version of the protocol that would skip segment broadcasts that are not needed by any incoming requests. Another one is to develop a variant of the protocol that would never require the customer STB to receive data from more than two channels at the same time.

REFERENCES

1. S. W. Carter, D. 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, pages 179–189.
2. A. Dan, D. Sitaram, and P. Shahabuddin. "Dynamic batching policies for an on-demand video server." *Multimedia Systems*, **4**(3):112–121, June 1996.
3. 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
4. L. Juhn, and L. Tseng, "Harmonic broadcasting protocols for video-on-demand service," *IEEE Transactions on Broadcasting*, **43**(3):268–271, Sep. 1997.
5. 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.
6. J.-F. Pâris, S. W. Carter and D. D. E. Long. "A low bandwidth broadcasting protocol for video on demand." *Proc. 7th International Conference on Computer Communications and Networks* (ICCCN '98), pages 690–697, Oct. 1998.
7. J.-F. Pâris. "A simple low-bandwidth broadcasting protocol for video on demand," *Proc. 7th International Conference on Computer Communication and Networks*, pages 690–697, Oct. 1999.
8. J.-F. Pâris. "A fixed-delay broadcasting protocol for video-on-demand," *Proc. 10th International Conference on Computer Communications and Networks* (ICCCN '01), pages 418–423, Oct. 2001.
9. ReplayTV. http://www.replay.com/.
10. TiVo Technologies. http://www.tivo.com/.
11. UltimateTV. http://www.ultimatetv.com/.
12. S. Viswanathan and T. Imielinski. "Metropolitan area video-on-demand service using pyramid broadcasting." *Multimedia Systems*, **4**(4):197–208, 1996.