An Empirical Study of Harmonic Broadcasting Protocols

Cyrus Dara Vesuna[1], Jehan-François Pâris[2]

Department of Computer Science
University of Houston
Houston, TX 77204-3010

Tel. (1) 713-743-3341 Fax (1) 713-743-3335 Email paris@cs.uh.edu

ABSTRACT

Harmonic broadcasting protocols provide an efficient way to distribute on demand videos that are simultaneously watched by many viewers. Despite this, they were too often considered to be of marginal importance, due to the large numbers of independent data streams they require. We built a TCP/IP based client-server pair to measure the performance of the Cautious Harmonic Broadcasting protocol on conventional Fast Ethernet networks and found that a server with 512 MB of RAM could support up to 2,400 video streams and achieve a waiting time of less than six seconds for a three-hour video.

Keywords:
broadcasting, harmonic broadcasting, video-on-demand.

I. INTRODUCTION

Video broadcasting protocols provide an efficient way to distribute videos that are likely to be watched by more than ten to twenty viewers at a given time. These protocols operate by breaking up videos into segments and repeating these segments according to a fixed schedule over multiple data streams. Harmonic broadcasting protocols [1-3] are particularly interesting because they require less total bandwidth than other video distribution protocols, to guarantee a given maximum delay to the customers wanting to watch a given video.

The excellent performance of harmonic protocols comes at a price: they allocate a separate data channel to each individual segment of a given video. A video server broadcasting videos according to a harmonic protocol would thus have to manage several hundred independent data streams per video. In addition, each of these data streams would have a different bandwidth. As a result, harmonic protocols were often considered to be too difficult to implement to be of any practical value. Several recent broadcasting protocols [4-7] have indeed attempted to reproduce the performance of harmonic protocols while using much fewer streams.

To test the feasibility of harmonic broadcasting protocols in a LAN environment, we built a dedicated server simulating the behavior of a VOD server using the *Cautious Harmonic Broadcasting* protocol [3]. This server transmitted packets over the network at the exact data rates required by each stream. A client process on another workstation decoded these packets and kept track of their timeliness. We found that a single personal computer with 512 megabytes of main memory could transmit on up to 2400 simultaneous harmonic channels on a Fast Ethernet network, with only 0.1 percent of the data packets arriving more than 0.5 seconds late. This translates to a maximum waiting time less than five seconds for a three-hour video.

The remainder of this paper is organized as follows. In Section 2 we survey relevant previous work on broadcasting protocols for VOD. In Section 3, we describe our experimental setup, including hardware testbed, software methodology and network parameters. Our findings are discussed in Section and Section 5 has our conclusions.

II. PREVIOUS WORK

Broadcasting protocols for VOD aim at delivering efficiently "hot" videos—that is, videos that are likely to be watched by many viewers. Rather than transmitting one separate data stream to eac

[1] Supported in part by a grant of the USENIX Association.
[2] Supported in part by the National Science Foundation under grant CCR-9988390.

J. Díaz de León, G. González, J. Figueroa (Eds.): Avances en Ciencias de la Computación, pp. 351-358, 2003.

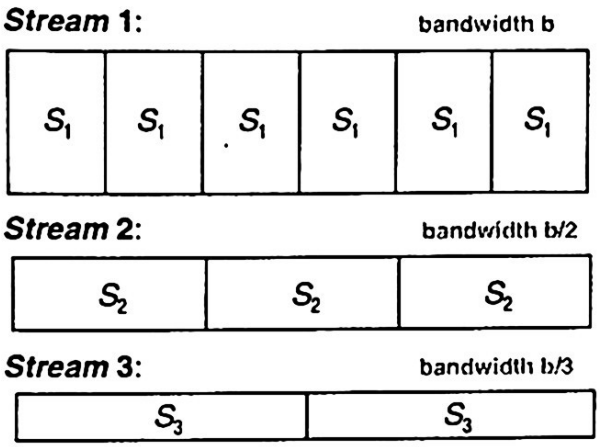

Figure 1. The first three streams for the HB protocol.

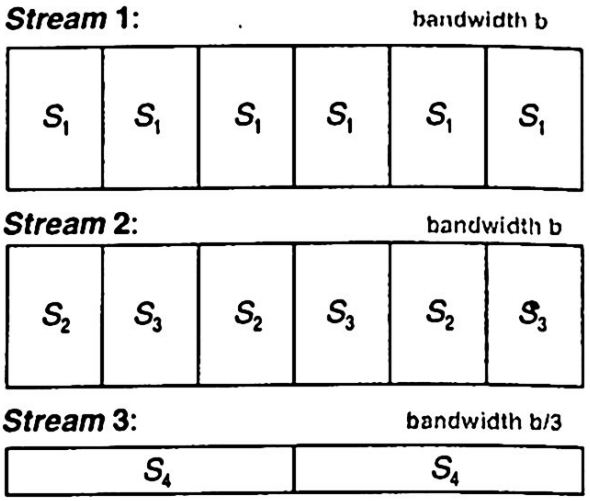

Figure 2. The first three streams for the CHB protocol.

customer requesting a given video, these protocols repeatedly broadcast the video over several data streams in such a way that no customer will have to wait more than a few minutes before being able to start watching the video.

Two factors make broadcasting protocols especially important to the success of VOD services. First, they will handle a large percentage of the user requests, as it can conservatively be estimated that at least 40 percent of the viewers would order the same 10 to 20 popular videos [8-9]. Second, potential customers of a VOD services will not be ready to pay much more for a VOD than they pay now for a videocassette rental or a pay-per-view program. Hence, any reduction in the cost of providing a given quality of service through video broadcasting is likely to have a direct impact on the commercial viability of VOD services.

As a result, video broadcasting protocols have become an emerging area of research. The last eight years have seen the development of several broadcasting protocols starting with the pioneer work of Hollmann and C. D. Holzscherer [10] and Viswanathan and Imielinski's *Pyramid Broadcasting* protocol [11].

All these broadcasting protocols require a user set-top box (STB) capable of storing all the video data that arrive ahead of time, which could amount to 40 to 50 percent of the video. In the current state of the storage technology, this implies that the STB must have a local disk.

Harmonic Broadcasting (HB) [1] divides each video into N segments S_i with $i = 1, 2, \ldots, N$ of equal duration d. Each of these N segments is continuously broadcast on a separate data stream and the bandwidth of these channels is proportional to the harmonic series. As seen on Figure 1,segment S_1 is broadcast at bandwidth equal to the video consumption rate b, segment S_2 is broadcast at half this bandwidth, segment S_3 is broadcast at bandwidth $b/3$ and so on. More generally, segment S_i with $1 \le i \le N$ is broadcast by stream i at bandwidth b/i.

When customers order a video, their STB waits for the beginning of a new instance of segment S_1 on stream 1. Since the first segment is broadcast at full bandwidth, the customers can start watching it. Meanwhile the STB starts receiving and buffering video data from the remaining $N-1$ streams. Hence the maximum customer waiting time is equal to the duration of a segment $d = D/N$ where D is the duration of the video.

The total bandwidth used by the video is given by

$$B_{HB} = \sum_{i=1}^{i=N} \frac{b}{i} = b \sum_{i=1}^{i=N} \frac{1}{i} = bH_N \qquad (1)$$

where b is the video consumption rate and H_N is the harmonic number of N.

The original HB protocol suffers from a fundamental flaw: it cannot always provide in-time delivery of all video data [2]. Pâris, Carter and Long proposed two variants of the harmonic broadcasting protocol, the *Cautious Harmonic Broadcasting* (CHB) and the Quasi-Harmonic Broadcasting (QHB) protocols, which fully remedy the limitations of the original protocol.

As shown in Figure 2, the CHB protocol broadcasts its two first streams at full bandwidth. Stream 1 repeatedly broadcasts segment S_1 while stream 2 broadcasts segments S_2 and S_3. Stream 2 is the sole stream broadcasting more than one segment as subsequent segments are broadcast on separate streams at decreasing bandwidths. In particular, segment S_i with $4 \le i \le N$ is broadcast by stream $i-1$ at bandwidth $b/(i-1)$. This new organization guarantees that the customer will either receive a segment at full bandwidth when it is needed, or have the entire segment already in its buffer before it is needed.

The bandwidth requirements of the CHB protocol are given by:

$$B_{CHB} = b + b + \sum_{i=3}^{N-1} \frac{b}{i} = \frac{b}{2} + \sum_{i=1}^{N-1} \frac{b}{i} = \frac{b}{2} + bH_{N-1} \quad (2)$$

where b is the video consumption rate, N is the total number of segments and H_{N-1} is the harmonic number of $N-1$.

A fourth harmonic protocol, *Polyharmonic Broadcasting* (PHB) [4] implements a *fixed-delay* policy. Like other harmonic broadcasting protocols, PHB breaks a video into N segments of equal duration. Each of these N segments is allocated a separate stream and segment S_i with $1 \le i \le N$ are broadcast by stream i at bandwidth $b/(i+m-1)$. As a result, any customer having waited m/N times the duration of the video can receive all the video segments by the time they are needed. The bandwidth requirements of the PHB protocol are given by:

$$B_{PHB} = \sum_{i=1}^{N} \frac{b}{i+m-1} = b(H_{N+m-1} - H_{m-1}) \quad (3)$$

All harmonic broadcasting protocols share a common limitation. They use rather large numbers of data streams for each video they broadcast. While CHB only requires six times the video consumption rate to guarantee a maximum waiting time of less than 80 seconds for a three-hour video, it allocates that bandwidth to 136 distinct data streams that will transmit data at individual bandwidths varying between 1 and 1/136 times the video consumption rate. PHB only requires 5.2 times the video consumption rate to achieve the same customer delay but would distribute this bandwidth among 548 data streams.

In addition, smaller maximum waiting times require even more data streams. PHB requires only around six times the video consumption rate to guarantee a maximum waiting time of 30 seconds for the same three-hour video but must distribute its bandwidth among at least 1,428 channels.

III. EXPERIMENTAL SETUP

The most straightforward implementation of a harmonic broadcasting protocol would use the same packet size for all streams and have the low-bandwidth streams transmitting their packets less frequently than the high-bandwidth streams.

If we use this solution, there could be instances where a large number of streams will be scheduled to send packets at exactly the same time. Server and network bandwidth limitations will force most of these packets to be delayed. Small delays are not likely to cause any problems that cannot be resolved by adding some extra buffering. We were wondering whether there would be times when the number of packets scheduled to be transmitted will be so large that some packets would experience unacceptable delays.

To evaluate the feasibility of harmonic broadcasting protocols in a LAN/WAN environment, we needed to design a test bed client/server system where:

- The server could send packets over the network at very specific rates and
- Server and clients could provide accurate readings of packets sending and arrival times.

Since time would be measured, the ability to coordinate the server clocks periodically through SNTP [12] was also necessary.

A. System Selection

Though a lot of applicable platforms for implementation of the server were possible, we settled for using an off-the-shelf OS Linux [13] built upon kernel versions 2.2 and 2.4.

Both client and servers were written under Linux Red Hat [14] version 6.0, subsequently upgraded up to version 7.2. There were multiple reasons for using this OS, namely, the advantages offered by open source code, the high reliability of the Red Hat distribution and its very wide support in the developer community. It also allowed the incorporation of POSIX compliance, which provided queuing of signals, real time signal dispatching and handling, with only minor modifications to the kernel. As of writing this paper, the Linux kernel has already incorporated a large number patches towards POSIX.4 compliance.

Timer interrupts for a standard Linux kernel (i386) normally occur 100 times per second. This translates to a ten-millisecond resolution for the timer, which was insufficient for our measurements. We increased the number of interrupt per seconds to 1,024 to allow theoretical resolutions of up to one millisecond. A simple kernel recompilation and installation was used to put these changes into effect.

B. Software Architecture

Since the primary focus of our work was to measure the performance of an actual video broadcasting protocols, considerable attention was paid to the system architecture of the client and server software, so that they do not introduce aberrations or distortions into the monitored parameters.

We selected a pre-forked transmitting design for our server. The number of video streams the server is to broadcast is passed as a parameter to the server on invocation. The server then forks one separate process for each video stream.

Advantages include scalability of architecture for multi-platform or multiprocessor support and modularity. In addition, separate process executions insure protection from errant processes and a more robust application.

The client uses an auto-forking scheme. When the client accepts a stream on its listening port, it forks a new process and lets the process handle that data stream. It then again begins doing a blocked accept( ) on the socket for the next incoming stream. This is very similar to way the inetd super daemon operates on Unix.

To reduce the effect of disk scheduling and I/O limitations, we bypassed the disk subsystem and generate a dummy video file in memory on the server side. The server reads this dummy file from the disk before it starts its operation. The file is of the segment size of the CHB protocol and is previously generated as a gzipped file with maximal compression to reduce the effects of network compression done by the intermediary network hardware (TCP/IP stack, network card, switch) between the server and the client.

By doing so, we eliminate the disk subsystems from our results and thereby presenting timing analysis results which reflect the real performance of the protocols being tested, with limitations of the disk subsystems bypassed almost completely. Even then, there was a large amount of logging for packet timing measurements on both the client and the server.

C. The Transport Layer

The selection of a transport layer protocol is the cornerstone of the network design; a considerable amount of discussion and experimental work has gone into the TCP versus UDP for multimedia [15]. The well-known advantages of TCP include reliable delivery and congestion control. Disadvantages include the time taken for setting up and tearing down TCP connections, congestion control and retransmissions that may add to jittering. UDP has the advantage of speed of connection though it lacks a mechanism for sequencing packets and reliable delivery of the packets. This in an important consideration because the loss of I-frames in a MPEG broadcast can prove disastrous.

D. Network Packet Size and TCP Window Size

An important parameter of network throughput using TCP is the *bandwidth delay product*, that is, the product of the network bandwidth (100 Mb/s) times the delay between the server and client. This

product helped us select the right TCP windows size for maximum throughput of the TCP/IP stack.

For packet size selection for our test, we ran multiple network throughput tests using IPERF [16] using different size packets and TCP window sizes of 2KB through 64 KB, we finally settled on using a 64 KB packet size with a 64 KB TCP window which we selected at runtime using the setsockopt( ) and getsockopt( ) functions from the UNIX socket library.

E. Hardware Configurations

We used various server and client configurations to test our system. The server configurations included a relatively fast system with an Athlon XP 1600 CPU, 512 MB of RAM and an IDE disk subsystem as well as slower systems with 600+ MHz CPU's and 512 MB of RAM. Client configurations included a system with an Athlon XP 1600P CPU and 512 MB RAM, dual 733 Intel processor systems with SCSI disk subsystems, and generic PC's with 256 MB RAM and 600+ MHz clock speeds.

F. Broadcasting Protocol Selection

We simulated the Cautious Harmonic Broadcasting protocol for a standard video of length $D = 180$ minutes. Each video was partitioned into N equal-size segments to be broadcast over $N-1$ distinct data streams.

The average bandwidth of the first two streams were fixed to $b = 5$ Mb/s, which is a reasonable approximation of the average bandwidth of an MPEG-2 video. As the CHB protocol specifies, the bandwidths of the other streams followed an harmonic series with the bandwidth of stream i equal to b/i for $3 \le i \le N-1$.

The maximum customer waiting time for a video of duration D is given by D/N.

G. Protocol implementation

Each data stream was managed by a separate server process continuously transmitting the contents of a specific segment. The client also consisted of a series of forked processes, each listening to a specific stream.

When the server broadcasts a 64 KB packet, we record its size and its timestamp. These packets arrive at the client broken up into numerous smaller packets. When the client receives one of these smaller packets, it records the arrival time and the size of the packet in a log. After the experiment is terminated, we add up the client packets sizes until they match the size of the original packet sent by the server, and thereby calculate the timing delay of each packet sent by the server. All send and received times are calculated in reference to the first packet send or received by the client. This negates the absolute time difference between the two systems, but does not compensate for time drift.

H. Experimental Results

Figures 3 and 4 summarize our findings. Both figures display histograms of packet arrivals delays with the x-axis representing packet delays in seconds and the y-axis representing the percentage of packets having experienced a specific delay. The top graphs of the two figures include both packets that arrive before the deadline and after it. Arrival times of packets that arrived before the deadline are represented as *negative delays* while late packets have positive delays. The bottom graph of each figure only shows late packets.

As one can see, most packets arrive within a five-second interval before the deadline. Some packets are exceptionally early, and can be buffered by the client until playback time arrives.

Packets that miss the deadline tend to miss it by less than a fraction of a second. We observed some very late packets, with one of them arriving 38.3 seconds late, but found out that packets arriving more than 1.5 seconds late never constituted more than 10^{-2} percent of the total number of packets and represent a very small fraction of the whole video. In almost all cases, there are hardly any packets that are delayed more than 0.8 seconds. As Figure 4 shows, our server was able to support up to 2,400 concurrent streams, which would allow to broadcast a video partitioned into 2,401 segments a guarantee a maximum customer waiting time of 5.3 seconds for a three-hour video.

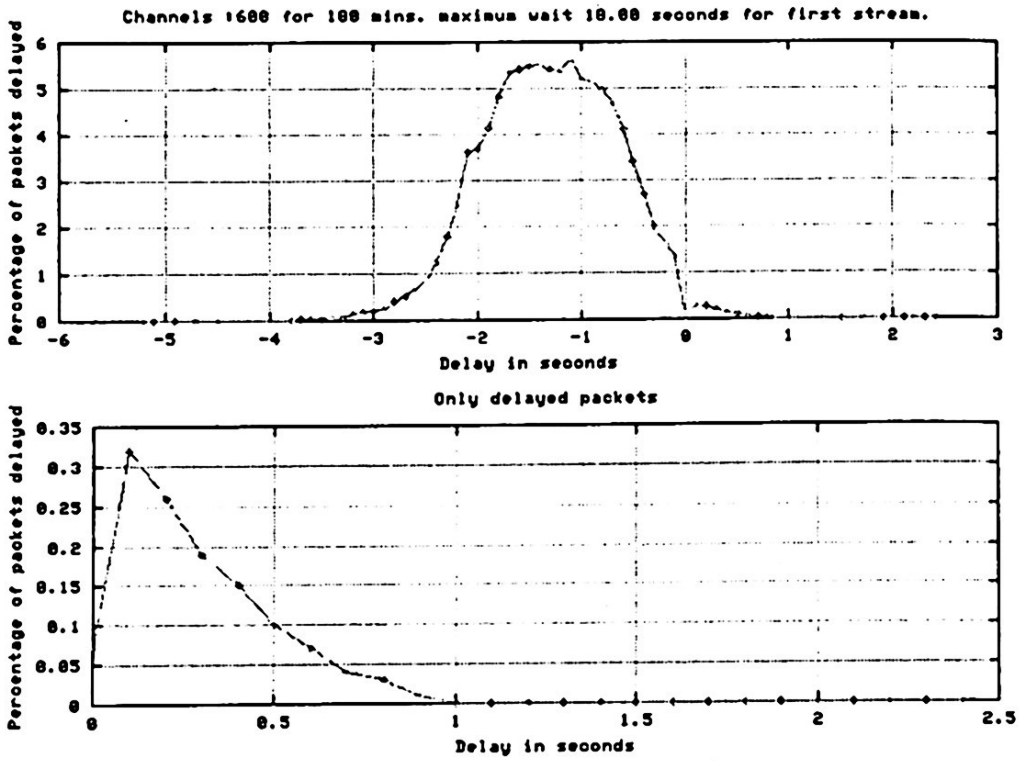

Figure 3. Packet delay histogram for a server broadcasting a three-hour video over 600 data streams to achieve a maximum customer waiting time of 18 seconds.

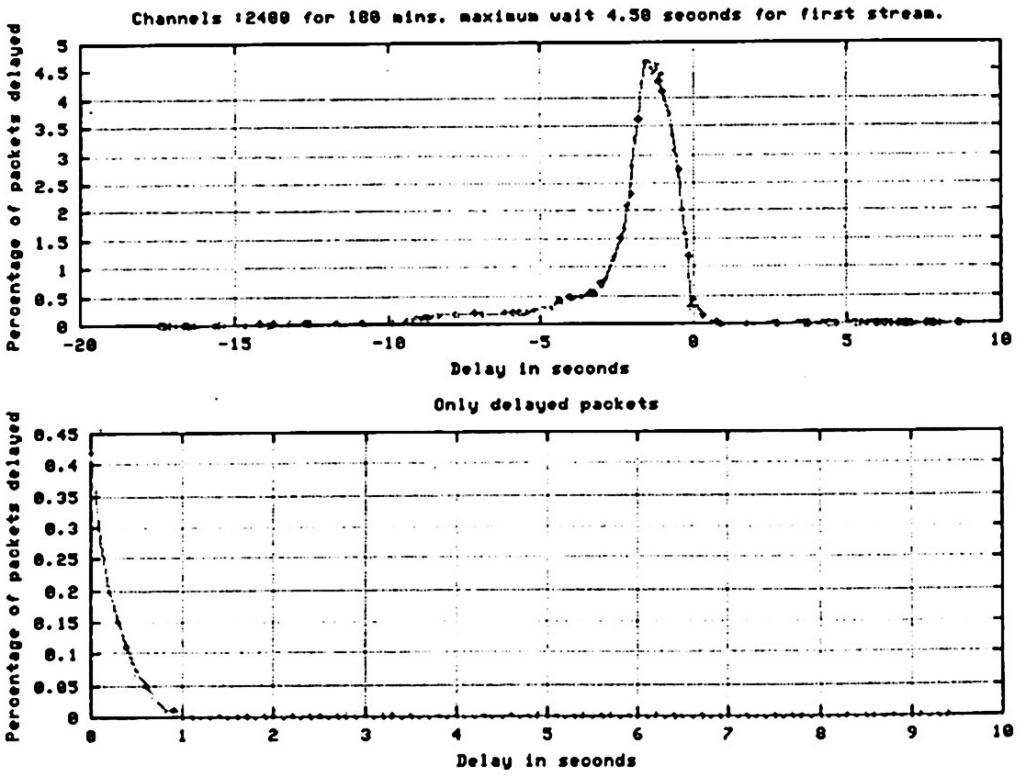

Figure 4. Packet delay histogram for a server broadcasting a three-hour video over 2,400 data streams to achieve a maximum customer waiting time of

Attempts to increase the number of segments above than 2,400 lead to a sudden decrease in system reliability, and very large packet delays. This situation was mostly attributable to system saturation since there were over 2400 processes competing for the server main memory.

We did not observe any noticeable differences between the performances of the two server architectures we investigated, namely a 600 MHz Intel Pentium III and an AMD Athlon XP1600 both with 512 MB of RAM. While the CPU utilizations were higher on the older architecture, the newer architecture could not support more concurrent processes. This apparent paradox can be explained by the fact that our server did not use the CPU for doing any video rendering or any other task where CPU speed would make a difference. Its two major functions were managing a large number of memory-resident processes and transmitting video packets to its clients. Since both functions are memory intensive, we should not be surprised to see that systems with the same amounts of RAM performed almost identically.

IV. DISCUSSION

We can draw two direct conclusions from our measurements. First, harmonic broadcasting protocols are a strong contender for distributing popular videos to large audiences over the Internet. This is especially true of the PHB protocol whose bandwidth requirements are very close to the theoretical minimum [3]. Second, the number of concurrent video streams a server can manage is limited by the memory requirements of the processes required to manage these streams.

There are two main approaches to address this bottleneck. First, we could add more RAM to the server, which would then increase the maximum number of memory-resident processes the server would be able to manage. Nowadays, the very low cost of RAM makes this solution a very attractive proposition.

The second approach is even cheaper. It consists of reducing the number of processes required to broadcast the N segments of a video. One of the simplest ways to achieve this goal is to transmit more than one segment per video channel.

Observe first that the first few segments of a video consume a very high fraction of the total bandwidth requirements of any harmonic broadcasting protocol. Conversely, the segments in the second half of the video contribute very little to the total bandwidth. Consider for instance a video broadcast over 719 data streams by a CHB protocol in order to achieve a maximum waiting time of 30 seconds for a three-hour video. As equation (2) shows, the total bandwidth required by the first 17 segments of the video is given by

$$B_{CHB} = \frac{b}{2} + bH_{16} = 3.881b, \qquad (4)$$

while the total bandwidth required to broadcast the whole video is given by

$$B_{CHB} = \frac{b}{2} + bH_{719} = 7.656b \qquad (5)$$

In other words, the first 17 segments of the video consume more than 50 percent of the total bandwidth. Conversely the segments in the second half of the video, that is, segments S_{361} to S_{720} only require $b(H_{720} - H_{360}) = 0.694b$ units of bandwidth, that is, less than 9 percent of the total bandwidth. Hence, we do not need to distribute these segments as efficiently as the first segments of the video.

Consider, for instance a very naïve modification of the CHB protocol that would repeatedly broadcast segments S_{361} to S_{720} on a single stream of bandwidth equal to the video consumption rate b. Each of these 360 segments would occupy 1/360 of the stream bandwidth and the layout would guarantee that any client could always receive these 360 segments before any of them needs to be displayed. Doing so would replace the 360 processes that were required to broadcast segments S_{361} to S_{720} by a single process managing the new stream, thus halving the number of processes required to distribute the video.

The sole drawback of the modification would be modest increase of the total bandwidth requirements of the protocol as segments S_{361} S_{720} would now require b units of bandwidth instead of the $0.694b$ units they required before. In other words, the cost of halving the number of processes needed by the server to broadcast the video would be a modest 4% increase of the total bandwidth.

More generally, consider a video partitioned into $2m$ channels being broadcast over $2m-1$ distinct video streams. Replacing the m streams used by segments S_{m+1} to S_{2m} by a single stream, would halve the number of processes required to broadcast and result in an increase Δ of the total bandwidth requirements of the protocol with

$$\Delta = b - b(H_{2m} - H_m) \qquad (6)$$

V. CONCLUSIONS

Despite their very low bandwidth requirements, harmonic broadcasting protocols were too often considered to be of limited practical interest due to the large numbers of independent data streams they require. To investigate this claim, we built a TCP/IP based client-server pair simulating the behavior of a video server using the Cautious Harmonic Broadcasting protocol to broadcast a single video over a Fast Ethernet network. We found out that a single PC with a Pentium III class CPU and 512 MB of RAM could manage up to 2,400 independent video streams and guarantee a maximum customer waiting time of 5.4 seconds for a three-hour video. This surprisingly good performance indicates that harmonic broadcasting protocols need to be seriously considered when selecting a video distribution protocol for an Internet-based video-on-demand service.

More work is still needed to investigate the benefits of switching to a connectionless network protocol and reducing the number of processes required to implement the CHB protocol.

REFERENCES

[1] L. Juhn and L. Tseng, Harmonic broadcasting for video-on-demand service. *IEEE Transactions on Broadcasting*, 43(3):268–271, Sept. 1997.

[2] J.-F. Pâris, S. W. Carter, and D. D. E. Long, Efficient broadcasting protocols for video on demand. *Proc. 6th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems* (MASCOTS '98), pages 127–132, July 1998.

[3] 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.

[4] J.-F. Pâris, S. W. Carter and D. D. E. Long, A Hybrid Broadcasting Protocol for Video on Demand, *Proc. 1999 Multimedia Computing and Networking Conference* (MMCN '99), pages 317–326, Jan. 1999.

[5] A. Hu, I. Nikolaidis and P. van Beek, On the design of efficient video-on-demand broadcast schedules, *Proc..7th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems* (MASCOTS '99), pp. 262-269, Oct. 1999.

[6] J.-F. Pâris, A Fixed-Delay Broadcasting Protocol for Video-on-Demand, *Proceedings of the 10th International Conference on Computer Communications and Networks* (ICCCN '01), pages 418–423, Oct. 2001.

[7] Y.-C. Tseng, M.-H. Yang, and C.-H. Chang, A Recursive Frequency-Splitting Scheme for Broadcasting Hot Videos in VOD Service, *IEEE Transactions on Communications*, 50, 8:1348-1355, Aug. 2002.

[8] A. Dan, D. Sitaram, and P. Shahabuddin, Scheduling policies for an on-demand video server with batching. *Proc. 1994 ACM Multimedia Conference*, pages 15–23, Oct. 1994.

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

[10] H. D. L. Hollmann and C. D. Holzscherer, Presentation system for messages that provide information on demand and transmitter station and receiver station for use in such presentation system, US Patent 5,524,271, June 1996.

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

[12] D. Mills, Simple Network Time Protocol (SNTP), Request for Comment RFC- 2030, IETF, October 1996

[13] L. Torvalds, *The Linux Operating System*, at http://www.kernel.org.

[14] RedHat Inc., *RedHat Linux*, at http://www.redhat.com/, Raleigh, North Carolina

[15] W. Bai, P. Zarros, M Lee, T. Saadawi, Design, Implementation and Analysis of Multimedia Conference System Using TCP/UDP," *IEEE International Conference on Communications*, pp. 432–438, 1994.

[16] M. Gates, A. Warshavsky, *Iperf version 1.1.1, Bandwidth Testing Tool*, at http://dast.nlanr.net/Projects/Iperf/, NLANR Applications, Feb. 2000.