Yima*: Design and Evaluation of a Streaming Media System for Residential Broadband Services

Roger Zimmermann, Kun Fu, Cyrus Shahabi, Didi Yao, Hong Zhu

Abstract—We describe and evaluate the implementation of a streaming media system called Yima, which consists of a scalable continuous media server and client components. We report on the real-life experiences that we gained from streaming near NTSC quality video and audio to residential locations within a metropolitan area. We investigated the feasibility of such streaming services with current broadband technology. We describe our experimental setup and the results, which indicate that streaming applications, such as movie-on-demand, are not only technically feasible but also may be economically viable in the near future.

Keywords— Multimedia delivery, distributed architecture, scalability, variable bit rate media, video servers

I. Introduction

MOST currently deployed networks, such as the Internet and corporate Intranets, are based on IP. Furthermore, isochronous media types, such as video and audio, are becoming ubiquitous and need to be disseminated via these IP networks. Yima is a complete end-to-end system that addresses the issues of (a) storing and retrieving isochronous media types, (b) delivering such media types with their real-time constraints intact, and (c) rendering the media at the client location. Yima relys on commodity off-the-shelf (COTS) hardware components, such as standard personal computers, to provide a cost-effective solution. It is designed to support a wide range of media delivery bit rates from several hundred Kb/s (e.g., MPEG-4) to more than 20 Mb/s (e.g., HDTV streams compressed with MPEG-2).

Many of todays isochronous media streams are compressed at variable bit rates (VBR), for example according to the MPEG-4 industry standard. Yima supports the streaming of VBR media via a simple yet flexible flow control paradigm. With this mechanism, the sender (server) does neither need to know about the compression scheme that is used for a specific stream, nor does it need to understand the details of a stream's file format. Therefore, new streamable media types, such as real-time haptic information [1], can easily be added.

This study presents an evaluation of Yima for video-on-demand type applications (e.g., distance learning, movie-on-demand) via currently available broadband connections to residential homes. There have been a number of reports of trials and deployments of video-on-demand services. However, little to no information is available about the architectural details and performance tradeoffs for these systems. For this study we have setup a client-server testbed that employs industry standards such as MPEG-4, ADSL (asynchronous digital subscriber line), RTP (real-time protocol [2]) and RTSP (real-time streaming protocol) over IP to evaluate the feasibility and performance of near NTSC quality video delivery to the home. Fig. 1 illustrates the experimental setup.

Section II introduces the Yima system architecture while Section III discusses the performance evaluation. Section IV contains our conclusions.

II. System Architecture

An important component of delivering isochronous multimedia over IP networks to end users and applications is the careful design of a multimedia storage server. The task of such a server is twofold: (1) to efficiently store the data and (2) to schedule the retrieval and delivery of the data precisely before it is transmitted over the network.

A. Data Placement and Scheduling

Magnetic disk drives have established themselves as the storage device of choice for CM servers because of their high performance and moderate cost. To achieve the high bandwidth and massive storage required for multi-user CM servers, disk drives are commonly combined into disk arrays to achieve many simultaneous I/O requests [3]. To efficiently store each individual multimedia object and to aggregate the bandwidth of multiple disks without requiring data replication, an object X is commonly striped into n equi-sized blocks: $X_0, X_1, \ldots, X_{n-1}$ [4], [5]. Both, the display time of a block and its transfer time from the disk are a function of the display requirements of an object and the transfer rate of the disk, respectively. The display requirements of multimedia objects encoded with many of the popular compression algorithms, e.g., MPEG-4, vary

* In ancient Iranian religion, Yima is the first man, the progenitor of the human race, and son of the sun.

The authors are with the Integrated Media Systems Center, 3740 McClintock Avenue, Suite 131, Los Angeles, CA 90089-2561. E-mails: [rzimmerm, kfu, cshahabi, didiyao, zhu]@usc.edu. This research has been funded in part by NSF grants EEC-9529152 (IMSC ERC) and ITR-0082826, and unrestricted cash/equipment gifts from IBM, Intel and SUN.

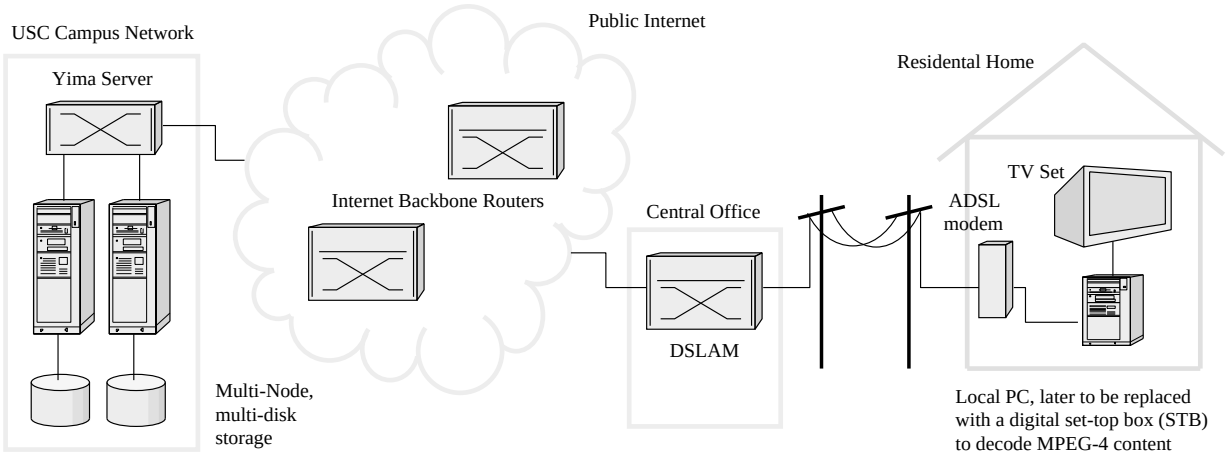

Fig. 1. Components of our experimental streaming media system setup.

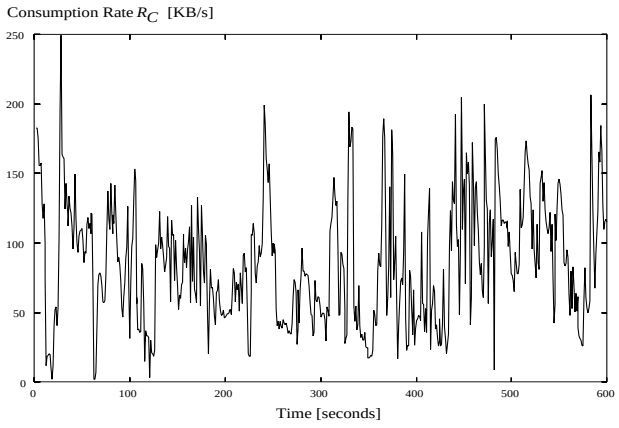

Fig. 2. Variable consumption rate of a 10-minute segment of a typical MPEG-4 movie.

as a function of time. Fig. 2 shows the display bandwidth requirement of a 10-minute segment for one of our test movies. The blocks of such variable bit rate (VBR) multimedia objects can be stored with two basic approaches: (1) the size of each block is varied to keep the display time per block constant, or (2) the size of each block is constant which results in a variable display time per block. The first approach simplifies the retrieval scheduling but the storage system becomes more complex. For the second approach the tradeoffs are reversed. The Yima framework is based on constant block sizes and our solution to VBR scheduling and delivery is presented in Section II-C.

There are two basic techniques to assign the data blocks to the magnetic disk drives that form the storage system: in a *round-robin* sequence [6], or in a *random* manner [7]. Traditionally, the round-robin placement utilizes a cycle-based approach to scheduling of resources to guarantee a continuous display, while the random placement utilizes a deadline-driven approach. In general, the round-robin/cycle-based approach provides high throughput with little wasted bandwidth for video objects that are retrieved sequentially (e.g., a feature length movie). Block retrievals can be scheduled in advance by employing optimized disk scheduling algorithms (such as *elevator*) during each cycle. Furthermore, the load imposed by a display is distributed evenly across all disks. However, the initial startup latency for an object might be large under heavy load because the disk on which the starting block of the object resides might be busy for several cycles. The random/deadline-driven approach, on the other hand, allows fewer optimizations to be applied, potentially resulting in more wasted bandwidth and less throughput. However, the startup latency is generally shorter, making it more suitable for interactive applications.

One disadvantage of random data placement is the need for a large amount of meta-data: the location of each block X_i needs to be stored and managed in a centralized repository (e.g., $\langle node_x, disk_y \rangle$). Yima avoids this overhead by utilizing a *pseudo-random* block placement. With random number generators, a seed is usually used to generate a sequence of random numbers. Such a sequence is pseudo-random because it can be reproduced if the same seed value is used. By placing blocks in a pseudo-random fashion, the next block in a sequence of blocks can always be found using the random number generator and the appropriate seed for that sequence. Hence, Yima needs to store only the seed for each file instead of a location for each block. Yima is configurable to operate in either round-robin or pseudo-random mode allowing us to measure and compare its performance under varying scenarios.

To achieve scalability, Yima servers are built as clusters of multiple server nodes. Each node may physically connect to one or more disk drives. A distributed file system provides a complete view of all the data on each node without the need to replicate individual data blocks (except as required for fault-tolerance [8]).

Hop #	Router
1	user-2iniv81.dialup.mindspring.com (165.121.125.1)
2	207.69.228.1 (207.69.228.1)
3	s4-1-1.lsanca1-cr1.bbnplanet.net (4.24.24.13)
4	p2-1.lsanca1-ba1.bbnplanet.net (4.24.4.5)
5	p0-0-0.lsanca1-cr3.bbnplanet.net (4.24.4.18)
6	s0.uscisi.bbnplanet.net (4.24.40.14)
7	usc-isi-atm.ln.net (130.152.128.2)
8	rtr43-18-gw.usc.edu (128.125.251.210)
9	rtr-gw-1.usc.edu (128.125.254.1)
10	zanjaan.usc.edu (128.125.163.158)

TABLE I

END-TO-END ROUTE FROM ONE YIMA CLIENT (LOCATED IN WEST COVINA, CONNECTED VIA AN ADSL LINE) TO THE YIMA SERVER (USC CAMPUS, LOS ANGELES). THE DISTANCE BETWEEN THE TWO LOCATIONS IS APPROX. 40 KM.

B. Real-Time Multimedia Delivery

In addition to the scheduling, data placement, and fault-resilience components, the Yima server software provides networking services to deliver data in real-time to end users. The stream data is transmitted via the industry standard networking protocols RTSP and RTP. These protocols are compatible with widely used industry standard technologies and hence provide compatibility and interoperability with other platforms in large-scale systems and the Internet, such as Apple Computer's QuickTime™.

RTP is used for the delivery of data packets because of the sensitive, real-time constraints of CM data. RTP uses the User Datagram Protocol (UDP) to ensure the quickest, although not necessarily the most reliable, packet delivery method. The UDP protocol features a low overhead but it delivers packets on a best-effort basis. Retransmitting packets is often considered inappropriate in the context of real-time streaming because it increases the latency between the sender and the receiver. This is especially critical for interactive and two-way applications. However, in a video-on-demand system some data buffering (and its effect of increasing the latency slightly) can often be tolerated if it improves the overall quality of the playback. In movie-on-demand systems the visual quality of the display is very important for the commercial success of such a service because end users can directly compare the playback with video displayed from a VCR or DVD player.

To study the effects of packet retransmissions we have implemented two policies in our Yima server: (1) no retransmissions and (2) selective retransmissions based on a low-overhead protocol [9]. The results for both approaches are described in Section III.

C. Client Buffer Management

The multimedia data that is streamed from the Yima server is consumed by an application at the client side. The client software implements the RTSP and RTP protocols to receive data transmissions from the server. A buffer module reassembles the RTP packets into multimedia blocks that are ready for consumption by the application.

Recall that the Yima multimedia delivery framework supports variable bit rate media. Numerous techniques have been proposed in the literature to transmit VBR media (for an overview see [10]). Most of the techniques *smooth* the consumption rate R_C by approximating it with a number of constant rate segments. Such smoothing algorithms need complete knowledge of R_C as a function of time to compute each segment length and its appropriate constant bit rate. In our implementation we reduce the smoothing technique into a binary variation: a stream is sent at a rate of either R_N or zero megabits per second. R_N is fixed at a value between the average and maximum consumption rate of a stream (the correct value depends on the client buffer size). A disadvantage of this technique is that it may temporarily require a higher link transmission rate than other smoothing techniques. Advantages include its simple design and that it can be applied dynamically to streams that are in progress without precomputation if, for example, only the maximum consumption rate is known.

Our technique is based on a simple flow control mechanism. A circular buffer of size B accumulates the media data and keeps track of two watermarks: buffer overflow WM_O and buffer underflow WM_U. Data is consumed from the same buffer by the decoding application. If the data in the buffer reach WM_O the data flow from the server is paused. The playback will continue to consume media data from buffer. When the buffer reaches the underflow watermark WM_U, the delivery of the stream is resumed from the server. If R_N is set correctly then the buffer will not underflow while the stream is resumed.

In an ideal case the server could be paused and resumed instantaneously and the watermarks could be set as follows: $WM_O = B$ and $WM_U = 0$. However, in a real world implementation the processing of pause and resume results in some delays (Yima uses the RTSP commands PAUSE and PLAY for these purposes). The accurate values of WM_O and WM_U are calculated as follows. After a pause request is sent out, the server is still sending data during a short time T_d. The data accumulated during T_d is $(R_N - R_C) \times T_d$. Hence, the buffer overflow watermark must be set to

$$WM_O \leq B - (R_N - R_C) \times T_d \quad (1)$$

to avoid a buffer overrun. Similarly, the buffer underflow watermark must be set to

$$WM_U \geq R_C \times T_d \quad (2)$$

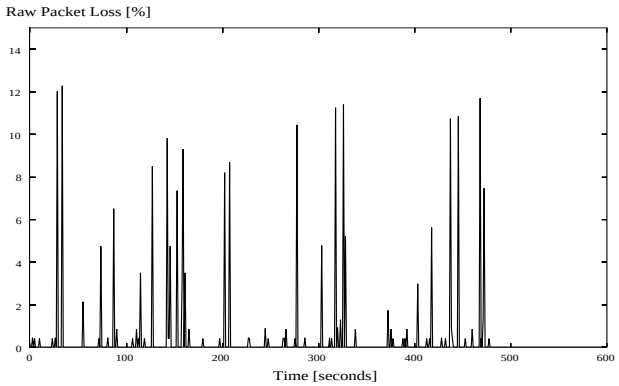

Fig. 3. Transmission (raw) packet loss for a MPEG-4 encoded movie segment of 10 minutes. The average is 0.365%.

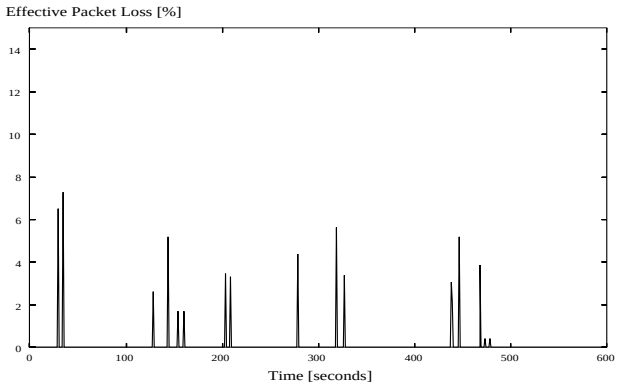

Fig. 4. Effective packet loss for a MPEG-4 encoded movie segment of 10 minutes. The average is 0.098%.

Finally, the following condition must hold: $WM_O \geq WM_U$. If this condition is not met then the buffer size B is not large enough for this operating environment.

The delay time T_d depends on the network delay between the client and the server as well as the server request processing time and must be empirically obtained.

III. Performance Evaluation

We have implemented a prototype of Yima to show (1) the flexibility of the Yima framework and (2) the feasibility of implementing the architectural design. First, we describe our experimental setup and then report the results.

A. Experimental Setup

Fig. 1 illustrates the details of our current system. For our Yima server setup, we are using two low-cost, commodity Pentium II 450MHz PCs with 384 MB of memory. Each PC is connected to an Ethernet switch (model Cabletron 6000) in our lab via a 100 Mb/s interface. Movies are striped over two 18 GB Seagate Cheetah disk drives (one per server node). The disks are attached through an Ultra2 low-voltage differential (LVD) SCSI connection that can provide 80 MB/s throughput. Red Hat Linux 6.0 is used as the operating system for each Yima server PC.

The data sent out from the Yima servers[1] in our lab are transported via the USC campus network to the public Internet. Table I shows the data route between one of our test client locations and the Yima servers. The geographical distance between the two end points is approximately 40 kilometers. The client was set up in a residential apartment and linked to the Internet via an ADSL connection. The raw bandwidth achieved end-to-end between the Yima client and servers was approximately 1 Mb/s[2].

[1] Note that the Yima architecture is scalable and we can easily expand the hardware to more PCs and/or disk drives.

[2] The ADSL provider did not guarantee any minimum bandwidth but stated that 1.5 Mb/s will not be exceeded.

For the client setup, we used low-cost, commodity Pentium III 600MHz PCs. The Yima player software runs under Linux or Windows and uses a variety of decoders to display different media types. Examples are MPEG-1 (1.5 Mb/s), MPEG-2 at DVD-quality (3-8 Mb/s) and at HDTV-quality (20 Mb/s), MPEG-4 ($<$ 100 Kb/s to $>$ 1 Mb/s), and Apple QuickTime™ formats (up to approximately 2 Mb/s). For our tests we chose an MPEG-4 software decoder called "DivX;-)" [11]. The high compression ratio of MPEG-4 allows for near NTSC quality video transmissions at less than 1 Mb/s bandwidth requirement, suitable for a residential ADSL connection. For example, our test stream was encoded with a frame size of 720 $\times$ 576 pixels and 25 frames per second (fps). The stream required an average of 105 KB/s (840 Kb/s) bandwidth for both the video and audio layers (audio was encoded in MPEG-1 Layer 3 format), shown in Fig. 2. This compares favorably with MPEG-1 which would require 1.5 Mb/s for a lower video resolution of 320 $\times$ 240 pixels at 30 fps.

B. Results

We have conducted several end-to-end video streaming playback tests with the physical setup described in the previous paragraphs.

Since the data transfer is based on RTP/UDP/IP, a data packet could arrive out of order at the client or could not arrive at all since UDP does not guarantee packet delivery. The characteristics of real-time data places rigid constraints on packet delivery and may be sensitive to packet loss. If a packet arrives after the decoder needs it, then it is considered a lost packet.

To achieve the best visual and aural quality possible at the client side we implemented the following two techniques: *(a)* The decoding of a movie was started with a delay after an initial amount of data had arrived at the client side buffer.

(b) To reduce the number of RTP data packets lost between the server and client locations we implemented a selective retransmission protocol [9].
The initial buffer size was determined so that no buffer underflow starved the MPEG-4 decoder. The selective retransmission protocol was configured to attempt at most one retransmission of a lost RTP packet if it seemed likely (based on the round-trip packet delay) that the retransmitted packet would arrive in time for consumption by the decoder. With this technique the *raw* transmission data loss $P_0 = p$ can be reduced to an *effective* data loss of

$$P_1 = p \times (q + ((1 - q) \times p)) \quad (3)$$

where p is server-to-client and q is the client-to-server data loss probability. Eqn. 3 assumes a different loss rate for both link directions since most broadband technologies (such as ADSL and cable modems) provide asymmetric throughput for upstream and downstream data flow. If the loss probability is identical in both link directions, then Eqn. 3 reduces to

$$P_1 = p^2 \times (2 - p) \quad (4)$$

Figs. 3 and 4 illustrate the reduction in lost data from an average of 0.365% to 0.098% with this protocol in our test system. Note that the loss characteristic of a real network link is very bursty (as illustrated in Fig. 3) as compared with the uniform approximations of Eqns. 3 and 4. Hence, the real reduction of the loss rate is less dramatic than would be expected from the analytical equations. However, in practice the protocol worked very well and some of the loss peaks are completely eliminated while others are reduced significantly.

B.1 Visual Results

The visual and aural quality of an MPEG-4 encoded movie at less than 1 Mb/s is surprisingly good. Our test movie, encoded at almost full NTSC resolution, displayed some blocking artifacts during complex scenes that can be attributed to the high compression ratio. If a highly compressed movie is streamed over a network and packets are lost, then severe artifacts may be visible. In our setup, because of the low packet loss rate, almost no degradation of the movie was noticeable compared with its local playback.

IV. Conclusions and Future Work

In this study we presented an evaluation of Yima which addresses the complete end-to-end issues of storing, retrieving, and delivering isochronous media types over IP networks. We described the architecture that is based on a scalable multi-disk, multi-node server and an MPEG-4 capable client. We introduced a mechanism to stream variable bit rate media in a simple yet flexible way via the industry standard protocols RTP and RTSP. We demonstrated the feasibility of streaming near NTSC-quality video and audio (compressed via MPEG-4 and MPEG-1 Layer 3 algorithms) to residential locations over current broadband connections (ADSL).

In the near future, we plan to scale up our prototype to more nodes and evaluate its scalability and fault-tolerance with a large number of different clients and media types as well as enable multiple retransmissions.

Acknowledgments

The authors would like to acknowledge the many suggestions and implementation help of Christos Papadopoulos and Rishi Sinha from the University of Southern California.

References

[1] C. Shahabi, G. Barish, B. Ellenberger, M.R. Koladouzan N. Jiang, S.-R. Nam, and R. Zimmermann, "Immersidata Management: Challenges in Management of Data Generated within an Immersive Environment," in *Proceedings of the Fifth International Workshop on Multimedia Information Systems (MIS'99)*, Indian Wells, CA, October 21-23 1999.

[2] H. Schulzrinne, "RTP: A Transport Protocol for Real Time Applications," 1996, URL: http://ww.itef.org/rfc/rfc1889.txt.

[3] E. Chang and H. Garcia Molina, "Efficient Memory Use in a Media Server," in *Proceedings of the International Conference on Very Large Databases*, 1997.

[4] V.G. Polimenis, "The Design of a File System that Supports Multimedia," Technical Report TR-91-020, ICSI, 1991.

[5] F.A. Tobagi, J. Pang, R. Baird, and M. Gang, "Streaming RAID-A Disk Array Management System for Video Files," in *Proceedings of the First ACM Conference on Multimedia*, Anaheim, CA, August 1993, pp. 393–400.

[6] S. Berson, S. Ghandeharizadeh, R. Muntz, and X. Ju, "Staggered Striping in Multimedia Information Systems," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1994.

[7] J. R. Santos and R. R. Muntz, "Performance Analysis of the RIO Multimedia Storage System with Heterogeneous Disk Configurations," in *ACM Multimedia Conference*, Bristol, UK, 1998.

[8] R. Zimmermann and S. Ghandeharizadeh, "Continuous Display Using Heterogeneous Disk-Subsystems," in *Proceedings of the Fifth ACM Multimedia Conference*, Seattle, Washington, November 9-13, 1997, pp. 227–236.

[9] Ch. Papadopoulos and G. M. Parulkar, "Retransmission-based Error Control for Continuous Media Applications," in *Proceedings of the 6th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 1996)*, Zushi, Japan, April 23-26 1996.

[10] J. Al-Marri and S. Ghandeharizadeh, "An Evaluation of Alternative Disk Scheduling Techniques in Support of Variable Bit Rate Continuous Media," in *Proceedings of the International Conference on Extending Database Technology (EDBT)*, Valencia, Spain, March 23-27, 1998.

[11] J. Hibbard, "What the $%@# is DivX;-)?," *Red Herring Magazine*, January 2001.