Bamba—Audio and video streaming over the Internet

by M. H. Willebeek-LeMair
K. G. Kumar
E. C. Snible

The World Wide Web has become a primary means of disseminating information, which is being presented increasingly through multiple media. The ability to broadcast audio and video information is becoming a reality with the advent of new media-streaming technologies. Most of the emerging streaming systems require high-bandwidth connections in order to deliver audio and video of suitable quality. In this paper we present a media-streaming system, called Bamba, that delivers audio and video over low-bandwidth modem connections with the use of standard compression technologies. Bamba offers high-quality audio and video over low-bit-rate connections and can operate using a standard HTTP server. The Bamba video is enhanced with special provisions for reducing the effect of errors in a lossy-network environment. Bamba adheres to existing standards wherever possible. Finally, Bamba has been fully implemented and deployed both internally at IBM and externally.

1. Introduction

The World Wide Web (WWW) has become a primary means of disseminating information. Initially, the type of information distributed was primarily in the form of text and graphics. Later, images and stored audio and video files emerged. These audio and video files are downloaded from a server and stored at the client before they are played. Most recently, streamed audio and video have become available from both stored and live sources on the Web. Audio and video streaming enables clients to select and receive audio and video content from servers across the network and to begin hearing and seeing the content as soon as the first few bytes of the stream arrive at the client. Streaming technology involves audio and video compression, schemes for stream formatting and transmission packetization, networking protocols and routing, client designs for displaying and synchronizing different media streams, and server designs for content storage and delivery. In this paper we present a system for audio and video streaming (with code name Bamba) developed at the IBM Thomas J. Watson Research Center. Bamba has been deployed within IBM and was demonstrated externally on the official Web site of the 1996 Olympics. It has since been made available for free download from the IBM AlphaWorks* Web site.[1]

Today's computer-network infrastructures, including the Internet, were not designed with streaming in mind. Streaming media requires that data be transmitted from a server to a client at a sustained bit rate that is high enough to maintain continuous and smooth playback at the receiving client station. A primary objective in developing Bamba is to stream audio and video across the Web through very-low-bit-rate connections. Audio is

[1] *http://www.AlphaWorks.ibm.com*

0018-8646/98/$5.00 © 1998 IBM

sufficiently compressed to stream over modem connections at 14.4 Kb/s, and video at 28.8 Kb/s. The system that has been developed not only achieves the low-bit-rate goal, but can also be extended to support higher-bit-rate streams to provide higher-quality streaming over intranets or higher-bandwidth Internet connections. Furthermore, when streaming is not possible because of congestion or insufficient bandwidth availability, the Bamba player (client software) at the receiving client automatically calculates how much data to preload in order to maintain continuous playback. This allows clients connected via low-bit-rate connections to fall back to a download-and-play mode and still receive the higher-bit-rate content.

Existing audio and video streaming technologies

In recent years, there has been much research and development in the areas of audio and video streaming as well as videoconferencing. Videoconferencing differs from audio and video streaming in that the communication is bidirectional, and end-to-end delays must be very low (<200 ms) for interactive communication. In fact, videoconferencing standards are quite mature and have emerged from the International Telecommunication Union (ITU) in the form of the H.3xx standards [1, 2], and from the Internet Engineering Task Force (IETF) in conjunction with the multicast backbone (MBone) [1, 3, 4]. In general, the two camps use the same audio and video compression standards (defined by the ITU) but differ in their networking protocol specifications.

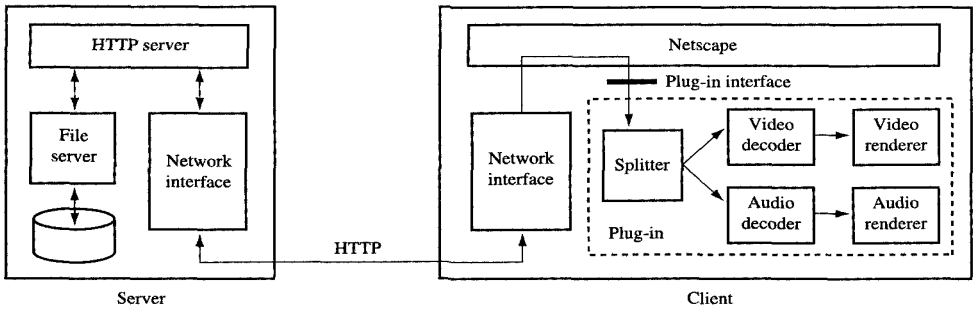
Audio and video streaming differs technically from its videoconferencing counterpart in that it can afford greater flexibility in end-to-end delays when the data is transmitted across a network and in the fact that stored content may be manipulated off-line with additional processing. These begin to merge when one considers live audio and video streaming applications (e.g., Internet, radio, and TV). The most relevant of the ITU standards is H.323, which defines audio/visual services over LANs for which quality of service cannot be guaranteed [5]. This standard specifies a variety of audio and video coders and decoders (CODECs) as well as signaling protocols to negotiate capabilities and set up and manage connections [6]. The underlying transport specified is the Real-time Transport Protocol (RTP) [7]. This protocol, defined by the IETF, is intended to provide a means of transporting real-time streams over Internet Protocol (IP) networks. A new protocol, the Real Time Streaming Protocol (RTSP), just proposed to the IETF, more directly addresses the issues of delivering and managing multimedia streams [8]. Clearly, this area is still evolving as new protocols are being defined and refined to satisfy a wide range of emerging networked multimedia applications.

There are a large number of audio and video streaming systems available in the market today [9]. These include VDOLive**,[2] StreamWorks**,[3] Vosaic**,[4] VivoActive**,[5] InterVU**,[6] and RealAudio**.[7] VDOLive, Streamworks, Vosaic, and RealAudio are based on proprietary client–server systems that transport their audio and video streams by means of User Datagram Protocol (UDP/IP) connections. This unreliable transport does not retransmit lost packets and is blocked by most firewalls unless they are specially reconfigured. The others use HTTP (based on TCP/IP) [10]. VDOLive employs a proprietary hierarchical compression technique that allows the server to adapt the video-stream bandwidth to the available network connection bandwidth. StreamWorks, Vosaic, and InterVu are based on MPEG** [11], while Vivo uses H.263 [12]. In general, these systems are designed to work over higher-bandwidth LAN connections and not at modem speeds. At modem speeds, the MPEG-based systems revert to slide-show-type video.

Bamba is a streaming system that was designed to run over existing computer network infrastructures. In particular, it is versatile in dealing with the heterogeneous nature of this environment and the unpredictable congestion behavior of today's network traffic. In the Bamba system, audio and video are compressed into a Bamba file. This file is specially formatted to interleave the audio and video content and may even be extended to include other data types. The Bamba file is placed on a server. A client equipped with the appropriate Bamba software is able to communicate with the server and receive the Bamba audio/video file. If the network conditions are suitable (sufficient sustained bandwidth is available), this file, streaming across the network, is played at the client immediately. Otherwise, the file is played once uninterrupted playback can be ensured.

The Bamba streaming system has several key features. The first of these is the quality of the audio and video, where the audio is set at a constant 6.3 Kb/s and the video ranges from very low bit rates of tens of kilobits per second to hundreds of kilobits per second. The second is the fact that both the audio and video compression are based on standard algorithms and can be performed by standards-compliant decoders. Third, the Bamba streaming system uses either a standard HTTP server or an enhanced video server running RTP over UDP/IP. In the HTTP case, no special server software is required to store and send Bamba clips, and the transmitted streams can traverse firewalls with no special firewall configuration requirements. In the case of the video server running RTP

[2] *http://www.vdo.net*
[3] *http://www.xingtech.com*
[4] *http://www.vosaic.com*
[5] *http://www.vivo.com*
[6] *http://www.intervu.com*
[7] *http://www.realaudio.com*



Bamba system block diagram.

over UDP/IP, additional functionality is provided by means of a control protocol between the client and server. This functionality includes pacing of the transmission stream at a target bit rate as well as specific start and end times of transmission within an audio or video file. Finally, the Bamba player has been implemented either as a helper application, which runs outside a Web browser, or as a browser plug-in, which enables application developers to embed audio and video clips easily within an HTML document or as a Java** applet, which can be downloaded directly from a Web server containing Bamba clips without requiring special software installation at the client.

The rest of this paper is organized as follows. In Section 2, we describe the underlying Bamba technology. This includes a description of the video-compression algorithm as well as details related to the overall system design. In Section 3, we describe several enhancements made to the basic Bamba streaming system, such as increased robustness in lossy-network environments. A description of the Live Bamba architecture is given in Section 4. The paper is summarized in Section 5.

2. Bamba technology

A base requirement of the Bamba streaming system is to function within the WWW standard HTTP-based client–server architecture. In this section, we provide a description of the overall client–server architecture and present details concerning the compression algorithms. We also describe the Bamba file format and synchronization technique.

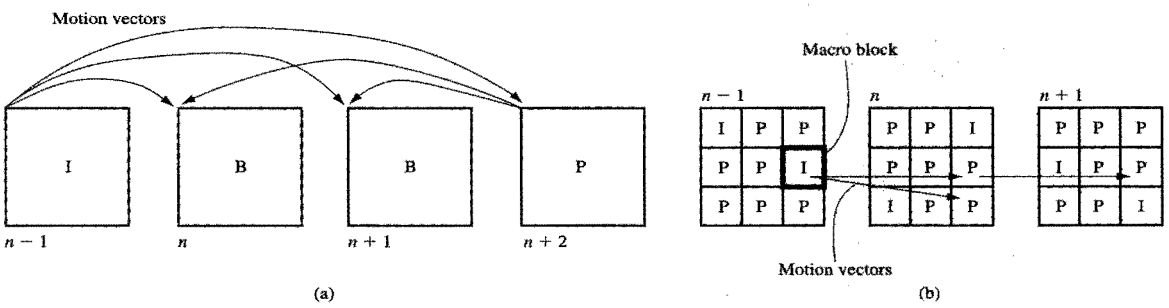
- *Bamba streaming architecture*

A block diagram of the Bamba streaming system is presented in **Figure 1**. The system consists of a client and a server component. The server is a standard HTTP Web server, which contains the stored Bamba audio and video files. The client consists of a Web browser and the Bamba audio and video plug-in software.

The Bamba plug-ins are implemented as a set of dynamic link libraries that interface with the Web browser through the Netscape-defined plug-in API. Netscape has defined a set of plug-in routines that are used to communicate between the plug-in and the browser. Each plug-in library contains an initialization routine within which is declared what Netscape plug-in routines are used by the plug-in. These routines include mechanisms to create and delete instances of a plug-in, manage the plug-in display window, control the flow of data streams to the plug-in, etc. In general, the plug-in is tightly integrated with the browser. Note that while Netscape was used in this example, the approach is similar for other browsers.

Bamba files may be embedded in HTML pages by means of a URL pointing to a file on an HTTP or video server. When the URL is requested, the server passes the metadata identifying the Bamba file and containing information about the file type to the client. The file type is used by the browser to launch the appropriate plug-in to play back the Bamba file.

Bamba was designed to stream clips from standard HTTP Web servers without special streaming software on the server. As such, Bamba is limited to the communication mechanisms provided by the HTTP protocol. This approach has certain advantages, the greatest of which is that it is simple and maps gracefully into the existing Web browsing architecture. As a result, content creators can easily produce Bamba audio and video clips and embed them in standard HTML [13] pages, which are then loaded onto and accessed from a standard HTTP server. Since the underlying transport protocol used by HTTP is TCP/IP, which provides reliable



Video-compression algorithms: (a) MPEG I-, P-, and B-frame compression dependencies; (b) H.263 I and P macro-block dependencies.

end-to-end network connections, no special provisions are required for handling packet loss within the network. In essence, a Bamba audio or video clip is treated like any other HTTP object, such as an HTML or JPEG [14] file. If selected, the Bamba clip is transferred to the client (browser station) as fast as TCP/IP can move it, and the client begins decoding and displaying the Bamba file as soon as the first few bytes arrive.

Since Bamba uses TCP/IP as the underlying communication protocol, the streams can traverse firewalls with no special configuration requirements. In general, systems based on UDP/IP cannot traverse firewalls without explicit permission changes in the firewall to allow passage to the UDP/IP packets. This is because UDP/IP packets are easier to imitate than TCP/IP packets, since the UDP/IP protocol involves no end-to-end handshakes or sequence numbers [15].

• *Bamba audio and video technology*

The audio and video technology used in Bamba is based on standard algorithms originally defined within the ITU H.324 standard for video telephony over regular phone lines [16]. The audio standard, G.723, specifies two bit rates: 5.3 Kb/s and 6.3 Kb/s [17]. Bamba uses the higher-bit-rate CODEC, which compresses an 8-kHz input of 16-bit samples to a fixed 6.3Kb/s stream. This audio algorithm is optimized to represent speech at high quality over low-bit-rate connections. It encodes speech into 30-ms frames by means of linear predictive analysis-by-synthesis coding [17]. The input signal for the higher-bit-rate coder is Multipulse Maximum Likelihood Quantization (MP-MLQ) [17].

The Bamba video CODEC complies with the H.263 video compression standard [12], which uses an approach based on the discrete cosine transform (DCT). This is similar to the technology used for MPEG. Unlike MPEG, which uses intrapicture frames (I-frames), predicted frames (P-frames) and bidirectional predicted frames (B-frames), H.263 does not define I- and P-frames, but rather I- and P-blocks—8-pixel by 8-pixel subregions of a frame. **Figure 2(a)** illustrates the MPEG dependencies among I-, P-, and B-frames, while **Figure 2(b)** illustrates the partitioning of H.263 frames into I- and P-blocks and the dependencies between blocks. Representing frames as collections of I- and P-blocks reduces the size variance between frames and adds flexibility in selecting the refresh distance between I-blocks for different regions of the video image. To maximize compression based on temporal redundancy, there may be long intervals between I-blocks for regions in the image that are not changing.

The H.263 algorithm is designed to deliver video over very low-bit-rate (<64 Kb/s) dedicated connections. In this low-bit-rate range, H.263 has been demonstrated to outperform its predecessor, H.261 [18], by a 2.5:1 ratio [2] (i.e., at the same bandwidth, the signal-to-noise ratio of H.263 is 2.5 times higher than that of H.261. H.263 can also be easily extended to higher bit rates, in the 100–200-Kb/s range. These rates are suitable for streaming over ISDN or intranet LAN-type connections. The H.263 video compression algorithm uses a planar YVU12 format, which contains three components: luminance (Y) and two chrominance planes (V and U). The sizes of these planes vary as a function of the video resolution. Two of the resolutions supported by Bamba are the Common Intermediate Format (CIF) and the Quarter Common Intermediate Format (QCIF) [2]; the formats are presented in **Table 1**. Smaller and intermediate-size resolutions are also supported. The resolution and target bit rate are selected at compression time. The compression target bit rate may be set anywhere between 10 and 356 Kb/s.

As with many compression standards, the H.263 standard specifies the format of the video so that any standards-compliant decoder can successfully decode the video stream. Typically, this leaves much flexibility in the actual encoding technique and implementation. The H.263 encoding used for Bamba uses an innovative algorithm to trade frame rate for frame quality [19]. The art in video compression lies in the decision of how best to apportion a few bits to different components in the compression process so that the compressed stream, once decoded and displayed, produces the highest quality as perceived by the end user. Quality is highly ambiguous and is perceived differently by different users. A typical tradeoff is between frame rate and frame quality (pixel quantization). For the same number of bits, it is possible to create two very different standards-compliant streams. One stream may have a higher frame rate, while the other may have a finer quantization of the frame pixels, obtaining a sharper image.

The Bamba video implementation incorporates a dynamic frame-rate-control algorithm, which trades frame rate for frame quality (bits per frame) while maintaining a constant average bit rate. This approach allows the video to balance between the two extremes and deliver smoother motion or sharper images as appropriate, depending on the content and scene changes in the video. The algorithm behavior is illustrated in **Figure 3**. A video sequence with dynamically changing content is used to illustrate the algorithm's adaptable frame rate. The original clip is approximately 30 seconds long, captured at 15 frames per second for a total of 445 frames. It was compressed at a target bit rate of 20 Kb/s and resulted in a total of 332 frames. Typically, larger frames are followed by a drop in frame rate in order to maintain the constant bit rate. The spikes in the figure correspond to larger frames, generated when the scene changes or the amount of motion in a scene is significant. These spikes are typically followed by several frame periods in which no data is transmitted at all.

The Bamba H.263 implementation includes special motion-estimation techniques [20] and fast DCT algorithms [21, 22], which result in very efficient implementations.

• *Framing structure*

A simple framing technique for smooth playback was implemented. Audio and video are interleaved into a single file to simplify the server function. Essentially, the server treats a Bamba file as any other data file. Audio and video data are interleaved proportionately to maintain a synchronous playback of both streams at the client. Bamba frames consist of a 240-byte segment of audio and a $240\beta/\alpha$-byte segment of video, where α is the audio rate and β is the video rate.

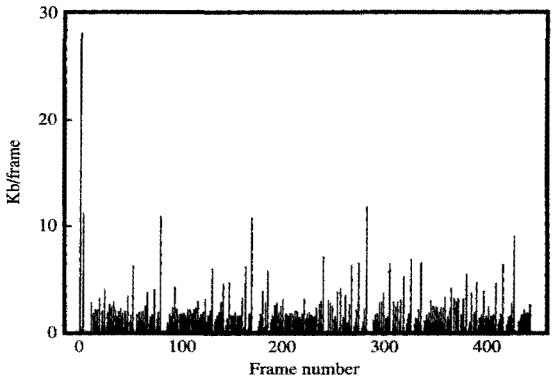


Figure 3

Illustration of Bamba video compression with dynamic-frame-rate-control algorithm. The number of bits transmitted is shown for each frame of a sample video sequence.

Table 1 CIF and QCIF planar YVU12 formats.

	Pixels/line	*Lines/frame*
CIF luminance (Y)	352	288
CIF chrominance plane (V)	176	144
CIF chrominance plane (U)	176	144
QCIF luminance (Y)	176	144
QCIF chrominance plane (V)	88	72
QCIF chrominance plane (U)	88	72

• *Streaming-control algorithm*

When the Web is accessed, the actual connection speed between a client and a server in the network varies depending on the access method (e.g., modem or LAN), the network load, the server load, and even the client load. Hence, it is rarely possible to guarantee performance in this "best-effort" environment, where processing and bandwidth resources are typically evenly distributed among all competing applications. Consequently, when an audio and/or video clip is accessed over the network, there is no guarantee that the resources (bandwidth and processing) are available to play the clip smoothly. To handle this situation, Bamba has a built-in rate monitor that dynamically evaluates the effective data-transfer rate (σ) of a selected audio or video clip and compares this to the specified bit rate ($\alpha + \beta$) for the clip, which is contained in the clip header. If the specified rate is less than the measured rate, the clip can be played immediately. If, on the other hand, the specified rate exceeds the measured rate ($\alpha + \beta > \sigma$), a fraction of the clip is buffered

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.