

Streaming Media Middleware is more than Streaming Media

Lawrence A. Rowe
Computer Science Division – EECS
University of California
Berkeley, CA 94720-1776
1-510-642-5117

Rowe@BMRC.Berkeley.EDU

ABSTRACT

Middleware for streaming media requires services other than media capture, encoding/decoding, network transmission, and presentation. Specifically most streaming media applications are distributed applications so they require the services being developed to support client/server and peer-to-peer applications. They also require multicast application services such as soft-state announce/listen protocols, reliable multicast protocols, and publish/subscribe multicast protocols. Some applications require dynamic user-interface definition and support for multimedia authoring and media processing.

1. INTRODUCTION

Streaming media middleware is more than code to support streaming media, that is, routines to capture, encode, decode, and play media, to frame, send, and receive network packets, to convert different media representations, and to play continuous media. While these abstractions are important, most streaming media applications require many more services. Example applications are distributed collaboration (e.g., video conferencing, multiple-player games, IP telephony, etc.), Internet webcasting, and multimedia authoring. Some of the services required by these applications are general-purpose distributed programming services (e.g., directory, process management, and communication services) and some services are specific to multimedia applications (e.g., media recording, indexing, playback, and editing).

This paper discusses the middleware services used in an Internet webcasting production system being developed at U.C. Berkeley and uses this system to illustrate the middleware services needed for a particular class of distributed streaming media applications.

Section 2 briefly describes the architecture of the webcast production system. The next section lists the middleware services required to implement this application and discusses other services that are needed to support the development of multimedia applications.

2. INTERNET WEBCASTING

The Berkeley Multimedia, Interfaces, and Graphics (MIG) Seminar is a regularly scheduled seminar that has been webcast worldwide on the Internet since January 1995. The seminar and webcast are a test bed for research on webcasting. In the early days the webcast was composed of one audio and one video stream transmitted on the Internet Mbone using the Mbone tools (i.e., *sdr*, *vat*, and *vic*). A manually controlled camera, which was aimed at the speaker or projector output (e.g., overhead transparencies, computer-based presentation, or VCR), and a wireless mic were used to capture the seminar. The webcast was produced by running the Mbone tools on a computer located in the classroom.

Many changes have been made over the last six years to improve webcast quality, increase the audience, and reduce the cost and effort required to produce the webcast. Quality improvements include: 1) webcasting two streams (e.g., speaker and content), 2) adding more cameras and mics to the classroom (e.g., audience camera and mics, wide-angle stage camera, etc.), 3) incorporating video effects processing and stored material playback, and 4) experimenting with remote speakers and questions.

We added multiple transmissions of the webcast to allow more viewers to watch the seminar and to experiment with higher quality webcasts (i.e., higher bit rate). Over the years we have had a difficult time delivering the webcast to viewers who were not at Universities or industrial research laboratories because few commercial organizations support multicast on their internal networks. Moreover, the transition from the DVMRP-based Mbone to a PIM-based Mbone further reduced the audience. To solve both problems, we added a Real Networks (RN) transmission. Although RN provides a lower quality webcast, it is designed to work through security firewalls and run on any platform. The development of experimental high-speed networks such as Internet2 allowed us to produce higher quality versions of the seminar that required more bandwidth. This past semester three transmissions were produced: 1) a low-bit rate Mbone webcast (200 Kbs), 2) a medium-bit rate Mbone webcast (800 Kbs), and 3) a RN webcast (50/100/250 Kbs multi-rate). During the last seminar of the semester we experimented with a new production TV quality streaming system, called RTPtv [9], which sends full-sized 60 field/second 4:2:2 video streams encoded using MJPEG and stereo 16 KHz Linear 16 audio streams which required an aggregate bandwidth of 15 Mbs.

In the early days, the seminar averaged 35-50 remote viewers. These viewers were synchronous, meaning that they watched the

seminar as it happened. During the transition from DVMRP to PIM, the number of remote viewers dropped to less than ten. The addition of the RN transmission and the development of Internet2 raised the average number of viewers to over 150. However, most viewers watch the webcast asynchronously.¹

Over the years we have developed software and systems to reduce the time and cost of producing a webcast and to improve the production (e.g., media quality and interaction). Some examples are:

Broadcast Manager. As more services were added to the production, the number of processes required to produce a webcast increased. Starting these applications on the appropriate machine and supplying the correct parameters using a command-line interface was both time consuming and error prone. At one point, it took the student producing the webcast over 30 minutes to start all the processes. An application was developed to store webcast configurations in a database and launch them automatically which simplified the process and reduced the time required and errors [27].

Director's Console. The introduction of more cameras and the addition of more streams to the webcast meant that we needed a tool for the webcast producer/director to control the sources that were mapped to each webcast stream. We also needed an interface that allowed the director to control the remote equipment (e.g., pan/tilt cameras, audio/video equipment configurations, etc.). This application, called *dc*, uses a service discovery protocol to locate the services required by the webcast and to add the appropriate interface abstractions to control the services [30].

Parallel Software-only Video Effects Processing System. The broadcast TV industry has known for years that special effects, both visual effects (e.g., titling, dissolves, composition, chroma key, etc.) and audio effects (e.g., background music), improve the quality of a program. A software-only video effects system was developed to exploit low-cost commodity hardware [15]. In addition, control interfaces were designed and implemented that allowed effects processing to be incorporated into *dc* [16].

Virtual Director. The cost of webcast production can be reduced by using software automation rather than human operators. The Virtual Director system adds services to the webcast production system and extensions to *dc* to automate camera switching. Inputs include rules designed to switch cameras according to an aesthetic model and recognizing when local audience members ask a question. This functionality is similar to the features provided by AutoAuditorium [8] and Microsoft [11] but it uses different input events, decision logic, and stream mappings.

Transcoding Gateways. Over the years, many gateways have been implemented to route streams to different sessions (e.g., from one multicast session to another or from multicast to a

unicast connection to a specific user), to transcode to different formats (e.g., Mbone to RN), and to implement bandwidth adaptation for the streams in a webcast [2,3]. We have recently completed a new tool, called *tgw*, which is integrated with the webcast production model we are developing.

Question Monitor. The most challenging problem in distributed collaboration applications is to support seamless interaction with remote participants. This problem involves more than just floor control because seamless interaction requires that the speaker and local and remote participants see each other, called "sense of presence." We have experimented with floor control tools [14] and visualizing remote participants, but work remains to make remote collaboration as effective as local collaboration.

Figure 1 shows the webcast production system architecture. The S_i processes represent different a/v sources for the webcast (e.g., cameras, etc.). The Studio Mbone network cloud represents two multicast sessions (i.e., audio and video) used to send streams between the webcast production processes. The Webcast Mbone network cloud represents the session with the streams that will be webcast. The Studio and Webcast Mbone streams use high bandwidth media formats that are convenient for production processing. Lines with a single shaft imply one stream. Lines with double shafts imply multiple streams (e.g., all Studio Mbone streams are read by *dc/vd*). And lastly, lines with dashed shafts represent unicast transmissions that might use UDP, TCP, or HTTP as supported by the RN transport protocol. The figure illustrates a webcast with three transmissions: 1) low bit rate Mbone, 2) high bit rate Mbone, and 3) RN. *tgw* copies packets from the Webcast Mbone to the appropriate destination network and modifies them if necessary. In practice, *dc/vd* output the webcast into the Internet2 Mbone and *tgw* joins that session to receive the webcast streams.

This figure shows the general architecture, but the system is composed of many processes that run on different computers. For example, figure 2 shows only a part of the system relating to the Director's Console and two capture computers in a studio classroom. The shaded box is an analog routing switch to which all audio and video devices in the room are connected. A separate embedded computer, manufactured by AMX, controls the switch.



Figure 1. Webcast Production System Architecture

¹ This result mirrors what happened with the lecture webcasting system we developed at Berkeley [22]. During this past semester fifteen classes were webcast, including several large undergraduate classes, and over 19,000 lectures were played each month. Approximately 90% of the lectures were played asynchronously with the heaviest usage just before examinations.

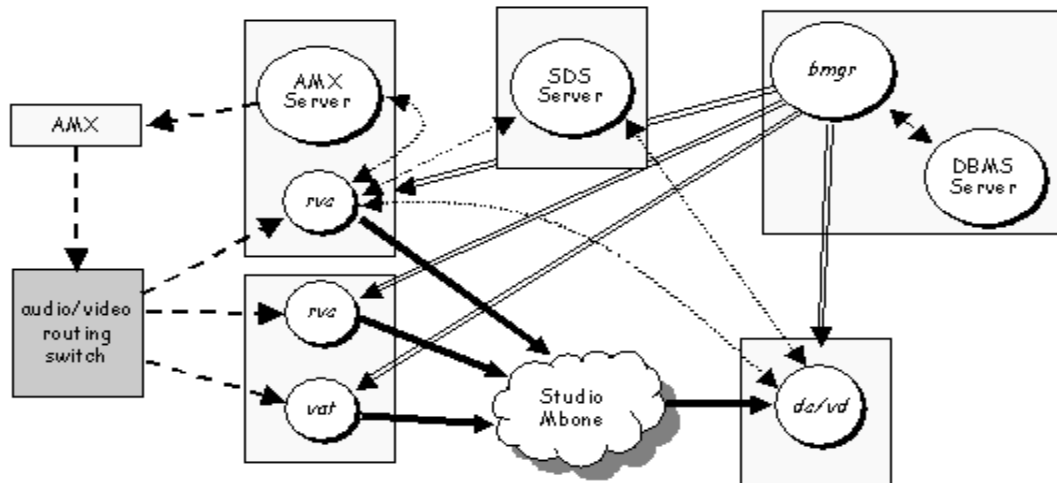


Figure 2. Process Architecture for a Subset of the Webcast Production System.

The AMX control computer has an RS232 interface that allows the webcasting system to control a/v equipment. The *AMX Server* process provides an RPC interface so other processes can execute equipment commands. The *rvc* and *vat* processes capture and transmit video and audio streams, respectively. The *dc/vd* process represents the Director's Console and the Virtual Director. The Virtual Director is actually composed of several processes that interact with *dc* and other webcast processes.

The *SDS Server* manages the session discovery service used by the webcasting system to locate and connect to webcast production services. Lastly, the Broadcast Manager (*bmgr*) is run when a webcast is initiated. Webcast configurations (e.g., what processes should be started on what machines) are stored in a database. The processes are launched either when a scheduled webcast is to be initiated or in response to a request entered by a producer/director.

Different line styles are used in the figure to denote different actions. The line styles are:

Solid black line – an RTP media stream.

Dash line – an external computer connection. For example, the routing switch is connected to a video capture board and an audio capture device.

Dotted line – an RPC call/return. For example, *rvc* executes an RPC on the *AMX Server* to switch the video source in the routing switch.

Double-shafted line – a process launch. For example, *rvc*, *vat*, and *dc/vd* are launched by *bmgr*.

Notice that all processes except the *AMX* and *SDS Servers* are launched by the *bmgr*. These two processes are continuously

running. They listen to well-known IP addresses for requests to execute operations.²

An example will clarify the implementation of the system. Suppose a producer/director wants to start a webcast. He or she runs the *bmgr*, selects a webcast configuration, and launches the processes. In the example above, the *dc/vd*, *rvc*, and *vat* processes are launched on the specified hosts. The *rvc* and *vat* processes register with the *Service Discovery Service*. *dc/vd* contacts the *SDS Server* in response to commands entered by the director to locate the desired webcast services. We use a dynamic service discovery protocol rather than the configuration database because different webcasts produced from the same room(s) might use different services.

The *SDS Server* responds to *dc/vd* with the information needed to contact the service process (i.e., host, port, and protocol). Each service returns data and code required to control the service in response to an open service request from *dc/vd*. In particular, media services return a Tcl command string that will instantiate interface widgets into a window allocated by *dc/vd* to provide the director interactive controls for the service. For example, the interface widget for *rvc* above includes a pulldown menu to change the video source by issuing a command to the routing switcher.

² The *AMX Server* should be configured as a launch on demand process using *inet.d* or a similar service, but the TclDP tools we use for RPC do not support this mechanism [Perham97]. And, the *AMX Server* was the first service developed for the webcasting system. It is used by several different applications (e.g., the location-based services for mobile clients work [Hodes97]) so we have to maintain the current implementation.

This section described various features of the webcast production system being developed and used at U.C. Berkeley. The next section discusses middleware services required to implement the system.

3. MIDDLEWARE SERVICES

The middleware services used to implement the webcast production system include many services other than the services required to capture, encode/decode, transmit, store/fetch, and process media. These services can be decomposed into three categories: 1) distributed programming, 2) user-interfaces, and 3) media services. This section describes each category in turn.

3.1 Distributed Programming

Many conventional distributed programming services are required for distributed streaming media applications. These services include at least the following.

Directory and Naming. Services to locate, name, and access various services.

Process Management. Services to launch, manage, and kill remote processes. These services might also include services to spawn servers on different hosts on-demand to balance computational load. Recovery or restart procedures are needed to insure that critical services are always available.

Remote Procedure Call. Services to communicate with remote processes and objects. The RPC services must include call/return, continuation, text and structured binary argument data transfer, and unicast and multicast communication.

Soft-state Announce/Listen Protocol. A commonly abstraction used in distributed streaming media applications is a soft-state announce/listen protocol. Rather than having a service maintain critical data about a remote service, called hard-state, the service maintains a data structure with entries that time-out if they are not re-registered by the remote service. All services announce periodically that they exist, and everyone listens to the communication. The Active Services Architecture implements such abstractions [3].

Distributed Object Services. Object-oriented programming has been shown to be a better principle for organizing some programs because the software abstractions mirror real-world objects and behaviors. Many research groups and companies over the past two decades have argued that objects should be primitive entities in distributed programming, file systems, and databases (e.g., CORBA [28], DCOM [25], OODB [4], etc.).

Scalable Naming and Addressing. Very large distributed applications require services that might not be needed if the application involves a smaller number of objects and or locations. Raman and McCanne have developed a protocol that allows participants to produce and consume objects reliably and efficiently for very large distributed applications [21]. This service, called the Scalable Naming and Addressing Protocol (SNAP), uses multicast and soft-state protocols to reduce communication costs and improve reliability. We have used SNAP in the parallel video effects system [16] and the webcast service architecture [30]. This service is similar to the multicast publish/subscribe service developed by Tibco and researchers at Cornell [5].

Some of these services are traditional distributed programming services, but an important feature of the webcast production system is the dynamic nature of both processes and services. We are continuously improving the system, which means prototyping new features and testing them in the production system. It is essential that we be able to define new classes, objects, and methods dynamically, that is, while the system is running. Moreover, the use of a scripting language, like Tcl, has been a significant research advantage. It is relatively easy to experiment with new specification languages and to implement and test new services and features. While the primary concern of many distributed programming systems is performance, the webcasting system, like all streaming media applications, requires some components to be efficient (e.g., media coding/decoding) and some components to be flexible and easy to implement (e.g., rule-driven control interfaces).

3.2 User Interfaces

The user interfaces used in the webcast production system are essentially client/server interfaces. The original Mbone tools (i.e., *sdr*, *vat*, *vic*, etc.) on which much of the system is dependent (i.e., Mash [17] and the Open Mash Consortium [19]) bundled an interface with code to capture, encode, transmit, receive, decode, and display streaming media. The development of distributed collaboration systems, such as the Access Grid [1], Virtual Rooms Videoconferencing System [29], and the Berkeley Webcast Production System, has lead to the recognition that client/server tools that can be managed either by a GUI interface or a remote program are needed. Consequently, the development of client/server tools is an important change being made by the Open Mash Consortium.

The development of client/server interfaces means that a client interface must be configured dynamically to provide the controls needed by the remote service. As described above for *dc/vd*, a client program must be able to configure the interface dynamically when given widget definitions and function bindings by the service. The prototyping nature of the research work implies that statically compiling all interface abstractions into the client applications is impractical. Scripting languages and interpretive languages (e.g., Visual Basic, Java, and Lisp) are natural candidates for such development. However, the toolkits and runtime environments must support dynamic definition and instantiation of classes and objects.

3.3 Media Services

The theme of this paper is that streaming media middleware is more than streaming media. Nevertheless, streaming media services beyond those provided by typical toolkits are required. The following lists some examples of desired services.

Media Storage. RTSP defines a protocol for controlling playback of RTP streaming media [24]. However, it does not define a standard for media storage in a file or the metalanguage for describing a multimedia title or presentation. SMIL2/XML might be appropriate for the metalanguage, but middleware services are needed to create, parse, and edit these descriptions.

Distributed Recording/Playback. A distributed conference or performance cannot be recorded at one location. Shuett

describes a robust collaboration archiving system that can fulfill some of the recording and playback requirements [23].

Multimedia Authoring. Many research groups are working on multimedia titles such as titles for class lectures and seminars [18]. But these are simple titles composed of static images from a presentation and streaming audio/video. Rich multimedia content titles must include dynamic branching, searchable content and indexing, interactive animations and simulations, and other interactive media. While many organizations and companies have attempted to define a standard for multimedia authoring and many companies and groups have tried to create authoring/editing tools, an open public domain standard has not been developed. Today, most authors use either a conventional programming language (e.g., C/C++, Java, or Visual Basic) or a proprietary authoring language (e.g., Macromedia Authorware, Asymetrix Toolbook, etc.). The web standards (i.e., HTML, XML, SMIL, etc.) offer the best alternative for a standard authoring language. But, many toolkits and services are needed to enable use of these standards.

Content Processing. All streaming media toolkits provide services to compress and decompress media. However, few provide services to process the data (e.g., produce key frames, salient still images [26] or video abstracts [6], analyze video [31] or audio [7], manipulate images, video or audio, etc.).

It is easy to see that much work remains to provide appropriate middleware tools for distributed streaming media applications.

4. SUMMARY

This paper argues that distributed streaming media applications require many services other than the media capture, coding/decoding, sending/receiving, and playback services found in a conventional streaming media toolkit. Too often the focus in these toolkits is to support the most popular media codecs and relatively simple applications (e.g., on-demand playback). Most interactive, distributed streaming media applications require significant support for distributed computing. And, multimedia applications that involve media editing and processing (e.g., content analysis and query) are poorly served by current toolkits.

5. REFERENCES

- [1] Access Grid, Web Pages, <http://www.accessgrid.org/>, June 2001.
- [2] E. Amir, S. McCanne, and H. Zhang, An Application-level Video Gateway, *Proceedings of The Third Annual ACM International Multimedia Conference*, November 1995.
- [3] E. Amir, S. McCanne, and R. Katz, An Active Service Framework and its Application to Real-time Multimedia Transcoding, *Proceedings of ACM SIGCOMM '98*, Vancouver, British Columbia, September 1998.
- [4] M. Atkinson, F. Bancillon, D. DeWitt, K. Dittrich, D. Maier, and S. Zdonik. The Object-oriented Database System Manifesto, *Proc. of the First Int. Conf. on Deductive and Object-Oriented Databases*, Kyoto, Japan, 1990.
- [5] K. Birman, et.al., Middleware Support for Distributed Multimedia and Collaborative Computing. *Software - Practice and Experience*, Vol. 29, No. 14, 1999, pp 1285-1312.
- [6] J. Boreczky, et.al., An Interactive Comic Book Presentation for Exploring Video, *Human Factors in Computing Systems ACM SIGCHI Proceedings*, The Hague, Holland, April 2000.
- [7] G.J. Brown and M. Cooke, Computational Auditory Scene Analysis, *Computer Speech and Language*, Vol. 8, August 1994, pp 297-336.
- [8] Foveal Systems, AutoAuditorium, <http://www.autoauditorium.com/>, November 1999.
- [9] M. Delco and L.A. Rowe, Production Quality Television over the Internet, Technical Report, Berkeley Multimedia Research Center, U.C. Berkeley, June 2001.
- [10] T.D. Hodes, R.H. Katz, Composable Ad hoc Location-based Services for Heterogeneous Mobile Clients, *ACM Wireless Networks Journal, special issue on mobile computing: selected papers from MobiCom '97*, Vol. 5, No. 5, October 1999, pp. 411-427.
- [11] Q. Liu, et.al., Automating Camera Management for Lecture Room Environments, *Proc. ACM SIGCHI'01*, Seattle, WA, March-April 2001.
- [12] E. Machnicki, AMXd Server, Web Page, Berkeley Multimedia Research Center, U.C. Berkeley, <http://bmerc.berkeley.edu/~machnick/amxd/index.html>, September 2000.
- [13] E. Machnicki and L.A. Rowe, Virtual Director: Automating a Webcast, submitted for publication, June 2001.
- [14] R. Malpani and L.A. Rowe, Floor Control for Large-Scale Mbone Seminars, *Proceedings of The Fifth Annual ACM International Multimedia Conference*, Seattle, WA, November 1997, pp 155-163.
- [15] K. Mayer-Patel, A Parallel Software-Only Video Effects Processing System, Ph.D. Dissertation, Computer Science Division – EECS, U.C. Berkeley, 1999.
- [16] K. Mayer-Patel and L.A. Rowe, A Multicast Control Scheme For Parallel Software-only Video Effects Processing, *Proceedings of The Seventh Annual ACM International Multimedia Conference*, October 1999.
- [17] S. McCanne, et.al., Toward a Common Infrastructure for Multimedia-Networking Middleware, *Proc. Seventh Intl. Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '97)*, St. Louis, Missouri, May 1997.
- [18] S. Mukhopadhyay and B. Smith, Passive Capture and Structuring of Lectures, *Proceedings of The Seventh Annual ACM International Multimedia Conference*, October 1999.
- [19] Open Mash Consortium, <http://www.openmash.org/>, March 1999.
- [20] M. Perham, B.C. Smith, T. Jánosi, I. Lam, Redesigning Tcl-DP, *Proceedings of the Fifth Annual Tcl/Tk Workshop*, Boston MA, July 1997.
- [21] S. Raman and S. McCanne, Scalable Data Naming for Application Level Framing in Reliable Multicast, *Proceedings of The Sixth Annual ACM International Multimedia Conference*, Bristol, UK, September 1998.

- [22] L.A. Rowe, et.al., BIBS: A Lecture Webcasting System, Technical Report, Berkeley Multimedia Research Center, U.C. Berkeley, June 2001. Available at <http://bmrc.berkeley.edu/papers/bibs-report.html>.
- [23] A. Schuett, Active Services for Archive Applications, Ph.D. Dissertation, Computer Science Division – EECS, U.C. Berkeley, 2000.
- [24] H. Schulzrinne, A. Rao, and R. Lanphier, Real Time Streaming Protocol (RTSP), RFC 2326, Internet Engineering Task Force, April 1998.
- [25] R. Sessions, *COM and DCOM: Microsoft's Vision for Distributed Objects*. John Wiley & Sons, New York, NY, 1997.
- [26] L. Teodosio and W. Bender. Salient video stills: Content and context preserved. *Proc. First ACM International Conference on Multimedia*, Anaheim, CA, August 1993.
- [27] D. Wu, A. Swan, and L.A. Rowe, An Internet Mbone Broadcast Management System, *Multimedia Computing and Networking 1999, Proc. IS&T/SPIE Symposium on Electronic Imaging: Science & Technology*, San Jose, CA, January 1999.
- [28] S. Vinoski, CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments, *IEEE Communications Magazine*, Vol. 14, No. 2, February 1997.
- [29] Virtual Rooms Videoconferencing System, Web Page, <http://www.vrvs.org/About>, June 2001.
- [30] Tai-Ping Yu, D. Wu, K. Meyer-Patel, and L.A. Rowe, dc: A Live Webcast Control System, *Multimedia Computing and Networking 2001, Proc. IS&T/SPIE Symposium on Electronic Imaging: Science & Technology*, San Jose, CA, January 2001.
- [31] H. J. Zhang, C. Y. Low, S. W. Smoliar and J. H. Wu, Video Parsing, Retrieval and Browsing: an Integrated and Content-based Solution, *Proceedings of The Third Annual ACM International Multimedia Conference*, San Francisco, November 1995.