# The Multimedia Conference Recorder

Lambros Lambrinos, Vicky Hardman and Peter Kirstein

Computer Science Department

University College London

{L.Lambrinos/V.Hardman/P.Kirstein @cs.ucl.ac.uk}

## Abstract

The ability to archive Internet multimedia collaboration traffic is required in most applications. This paper presents the design and implementation of the Multimedia Conference Recorder, a system capable of recording and playing back multimedia data. The MMCR has been designed to have a client-server architecture; logical independence between components simplifies development and component replication. The MMCR includes real-time multimedia indexes to facilitate fast data access, allowing efficient fast-forward, rewind and random access operations. Where possible, the network delay variance is eliminated to improve playback and stream synchronisation.

## 1. Introduction

Multimedia facilities are available on a variety of platforms for desk-top multimedia conferencing; users interact with each other in real time using audio, video and shared workspace. The ability to archive multimedia data is a requirement in most applications. Recording a multimedia session enables anyone who could not originally participate to find out the content of the discussion or seminar. Another requirement is the ability to playback a clip in an on-going conference. Multimedia servers with multicast capabilities along with recording, playback and editing facilities must be an integral part of the emerging multimedia computing infrastructure.

The Multimedia Conference Recorder (MMCR) is a client-server system designed for recording and playback of multicast multimedia conferences. The MMCR addresses the recording and playback characteristics of the various media and has a flexible architecture, allowing easy addition of new components. The MMCR includes real-time multimedia indexes to facilitate fast data access, allowing efficient fast-forward, rewind and random access operations. Where possible, the network delay variance is eliminated to improve playback and stream synchronisation.

The paper starts with a description of the applications of a multimedia recorder and some general information on multimedia conferencing. The requirements for a multimedia recorder are then presented and existing systems studied. In section 4 the design of the MMCR is described followed by the current implementation. The last section describes future enhancements to the MMCR.

## 2. Multimedia Conferencing and Presentations

The availability of low-cost multimedia-capable networks and cheap personal computing facilities that can cope with real-time multimedia are the two major factors which led to the current widespread use of desktop multimedia conferencing.

### 2.1 Applications

A multimedia conferencing session usually consists of multi-way real-time audio, video and shared workspace; the interactive use of continuous media makes the application delay intolerant.

Multimedia conferencing tools are publicly available from developers in Europe and the US. The most widely used tools are 'vic'[1] for video, 'rat'[2], 'vat'[3] for audio and 'wb'[4], 'NTE'[5] for shared workspace. The separate tools allow users to use a combination of the media. 'SDR'[6] is a tool for announcing sessions and inviting users to join a session.

Multimedia conferencing technology can be used by a variety of more specialised applications, such as distance learning (students at remote sites listen to and watch the teacher and the material presented – they can also interact with the teacher), major event broadcasts (e.g. NASA missions), trading activities (link trading rooms) and medical training/consultancy.

The ReLaTe [7] project pilots remote language teaching. Small group tutorials have been run over the last two years using geographically distributed real language teachers and students. This type of application needs the ability to review past lessons i.e. there is a need for a conference recorder. The previously mentioned applications may also require access to stored material.

### 2.2 Multicast data distribution

Multimedia conferencing is based on IP multicast [8], which allows a source to send data to multiple receivers, without the need to send to each receiver explicitly. Data is delivered via the MBone [9] a virtual network layered over high bandwidth parts of the Internet. To receive material, one joins the multicast group (the address/port combination) to which the data is transmitted. Due to the amount of data involved in the media streams (especially video) applications require a large amount of compute, storage and network bandwidth.

Multimedia data is sent using UDP (Universal Datagram Protocol) which provides a lighter weight service than TCP and does not guarantee delivery. Applications transmitting real-time data (i.e. audio and video) use the RTP [10] (real-time transport) protocol on top of UDP. RTP provides information such as the type of encoding, sender id and the data timestamp. RTP is used in conjunction with a control protocol RTCP [10] that allows data delivery monitoring and provides information on the session participants. Neither of RTP, RTCP guarantees quality of service; either regarding guaranteed delivery or bounded delay.

Other multicast tools use a reliable multicast mechanism [11] (again on top of UDP) to guarantee that the data will eventually reach all participants. Lost packets are requested and retransmitted. Some tools are not real time; they ensure that data is sent eventually; examples include presentation material which is present to be cached locally. Other tools are near real-time, such as shared workspace. Here both speed of response and reliable delivery are required, but some speed of response is sacrificed for more reliable delivery.

### 2.3 Requirements for recording and playback

The recording and playback mechanisms must be flexible enough to support the variety of media in a multimedia conference. The possible scenarios of using a recording/playback system lead to the following requirements:

- ability to record multicast data from any of the conferees
- a single storage location where people can store/retrieve data; have access to a large archive space
- browsing of the data archive
- material recording independence techniques suitable for different kinds of media
- playback of all types of media based on their individual characteristics allowing user interactivity and random access to the clips
- playback recorded material to another multicast conference

## 3. Multimedia Recorders in the MBone environment

A few MBone multimedia recorders have been previously developed ranging from simple command line tools handling a single stream, to systems for recording and playback of a whole MBone session. None yet offers a complete system with the flexible architecture indicated by the requirements list.

### 3.1 Existing systems

The MBone VCR [12] is an application with a “VCR-like” interface capable of recording and playing back single MBone sessions. Sessions can consist of RTPv1 (the older version) audio and video (wb/nte are not supported). The MBone VCR is not a client – server system and one consequence is that the system can only be controlled locally. Additionally, users must find their own space for recording; considering the amount of data involved most people will have trouble recording.

The Interactive Multimedia Jukebox [13] uses a scheduling mechanism to distribute RTP audio and video over three channels. Its best characteristic is its flexibility in how programs are requested and scheduled for playback. The IMJ lacks in some areas and is unsuitable for our purpose. For example, recording is done off line and interactivity is not guaranteed – it depends on server resources.

The multicast Media On Demand system [14] supports recording and playback of all media. It allows random access during playback and uses a Web interface for starting and controlling sessions. mMOD uses per-media indexes to access data, but not per-source indexes.

The development work on MMCR [15] has started three years ago when even less was available. Even now we feel it provides a vehicle for the minimal facilities required and can be extended to provide much more.

## 4. The Multimedia Conference Recorder (MMCR)

MMCR is a system specifically designed for recording and playing back multicast multimedia conferences. It has a client - server architecture and consists of the client, the server, the player and the recorder; logical component independence simplifies development and component replication.

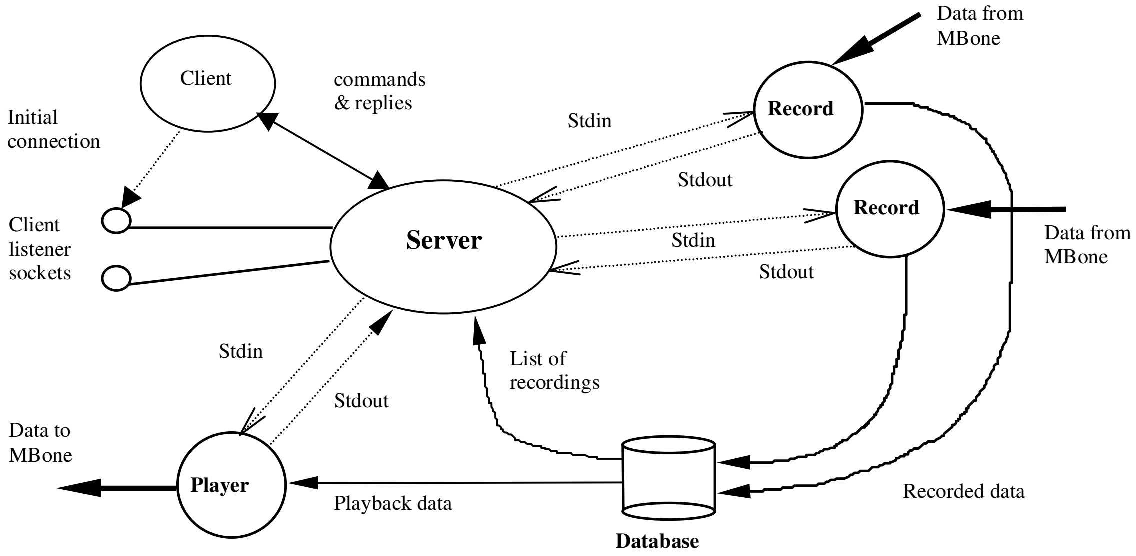


**Figure 1: The overall architecture of the MMCR**

All components have access to the database archive to store/retrieve recordings. Much research has considered ways of providing efficient storage/access mechanisms for Video On Demand (VOD) systems [16,17], which require high bandwidth delivery. However, the simple disk model used here is adequate considering the current bandwidth limitations on the MBone and an enhancement RAID array [17,18] can be integrated in the system if necessary.

## 4.1 The Server

The server acts as the single point of contact for recording, browsing and playback. It has a separate interface for each task and more interfaces can be added when required (e.g. editing interface). Depending upon the type of service requested by the client, the server starts one of the recording or browsing/playback mechanisms that receives and processes the remote client’s requests.

4.2 The Recorder

To record the media streams the recorder need not be an active part of the conference; it 'listens' to the specified multicast groups and collects the data storing it on a per-source basis. Information about each media (e.g. type, name) and each source (e.g. data location) recorded is saved in header files.

An important feature of the system architecture is the **real-time multimedia index.** For each source recorded there is an index created – the 'primary index'. The primary index contains an entry per data packet received. Each index entry consists of the time the packet was received at the recorder and the offset into the data file that the packet begins. This structure is used for efficient data access and index entries are independent of the data type.

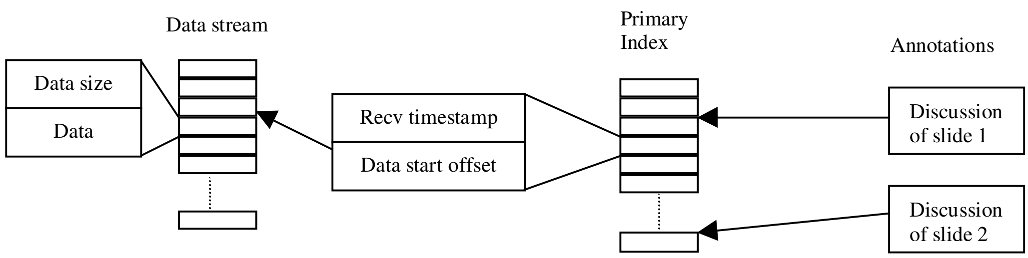


**Figure 2: The indexing mechanism and annotations**

The addition of annotations allows users to add meta-data to a recording describing particular sections in it. Each annotation points to the relevant index entry. The annotations mechanism (which can take the form of a book contents list) can be useful for rapidly accessing the sections in a media stream that are of interest to a user.

A potential use of the indexing mechanism is **Hierarchical indexing**. Instead of a data reference an index entry can point to another index entry. Based on this concept it is possible to create index hierarchies. These hierarchical structures can be created offline by special applications. For example an application may scan a video stream automatically generating an index entry every time the complete video image is transmitted. A special player will access the new index and playback only the indexed parts providing a fast perusal of the clip. As shown in figure 3 the new index stream contains references to the appropriate positions in the primary index stream.

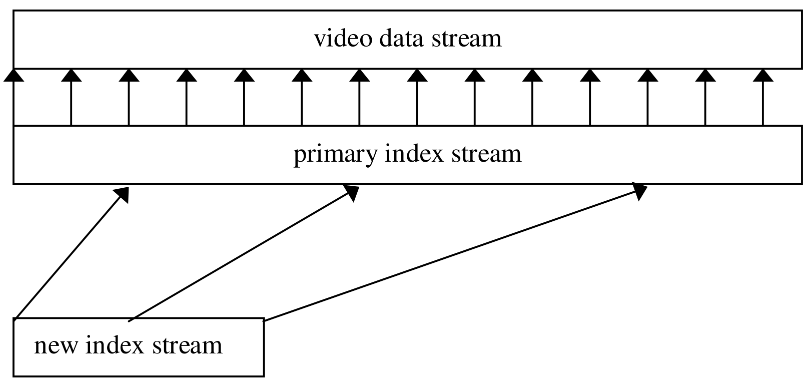


**Figure 3: Hierarchical Indexes**

Using the indexing mechanism, **editing facilities** can be provided. This is achieved by copying the required index segments and joining them to form another index.

4.3 The Player

The real advantage of storing data on a per source basis is that people can playback only the streams they are actually interested in - ignoring the rest. This allows utilisation of all kinds of networks as people with bandwidth limitations may choose to play only the audio which requires much less bandwidth than video.

Before starting a player, the browsing mechanism is used to identify the required session and its media/sources. These details are then passed to the player to access the corresponding data streams and send them on the specified multicast groups.

The player schedules real-time packet transmission based on the timestamp in the index entry. RTP compliant media provide additional information in the RTP header that can be used for providing smoother playback [see section 4.2]. Other media (e.g. shared workspace) packets are sent on the network based on their received timestamp (i.e. with the same inter-packet gap as they originally arrived).

The different media characteristics affect the fast forward and rewind operations. Audio and video are continuous media and therefore moving to a random point in the stream simply involves skipping intermediate parts and restarting at the new position. Additionally, the RTP header of the packets must be modified to maintain the continuity in timestamps and sequence numbers. For non-continuous media, such as shared workspace (wb/nte) fast-forward should involve the transmission of intermediate parts so that the data set is complete.

## 5. The implementation of MMCR

MMCR is currently under development as part of the MERCI project [19]. The client software and the server (runs on Unix workstations) are written in Java and the playback, recording mechanisms are written in Objective C. The client is totally independent of the server and can run on Unix workstations and PCs.

### 5.1 Features implemented

The system supports recording and playback of any type of media sent over the

Mbone (e.g. RTPv2 audio/video, shared workspace etc). For RTPv2 compliant audio and video, we extract information from the data and use it for a more efficient playback. Encrypted conferences can also be recorded and played back.

Partial recording of a conference is possible both as to the streams (e.g. only record the audio) and the duration (e.g. record for 1 hour). The facility to 'pause' and 'resume' recording is also provided.

Part of the server is a browsing mechanism which allows users to view the contents of the archive and obtain more information on particular recordings e.g. the media in conference Y and the actual participants[1].

Playback occurs on the multicast addresses specified by the users who must also start the appropriate conferencing tools to receive the data. The functionality provided consists of play, pause, fast forward, rewind and random access operations. Fast forwarding has provisions for all the various media requirements as specified in 4.3. Rewinding shared workspace data does not delete or resend (as receivers will ignore it) previously sent data.

## 5.2 The retrieval of sender timestamps for audio and H261 video

The 'best effort' model for data delivery relies on the current network conditions. As a result, packets may arrive at their recipients out of order, with a variable delay or even get lost. Based on information provided in the RTP header (only applies to audio and video) we try to correct some of the errors immediately after recording to make

[1] The participant's name can only be identified in RTP streams-for other streams an IP address is kept

playback smooth and 'network friendly'.

The real time representation contained in the RTCP packets (the NTP timestamp) combined with the timestamps in the RTP headers could be used to retrieve the actual time the packets were sent. There is no guarantee though, that the clocks of the machines from the original conference were either synchronised with each other or with the server. Unsynchronised clocks will give wrong results making the streams totally unsynchronised. To avoid this error, we do not attempt to calculate the original sender timestamps. Instead, we calculate the minimum network delay and eliminate the variation in the network delay. This method is based on a combination of the RTP timestamp in the data packets and the received timestamp in the index (section 4.2). Due to different characteristics between audio and video a different strategy is used for each media.

For **audio**, we calculate the difference between the RTP timestamp in the first packet of the talkspurt and a few subsequent packets. We convert this difference in time units and compare it with the difference between the received timestamps of the packets. If it is greater, the first packet was actually delayed more than the packet it is compared to; its received timestamp is adjusted accordingly to make the difference between the received timestamps in the packets equal to the difference between their RTP timestamps.

| Packet number | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Packet's RTP timestamp | 23456960 | 23457280 | 23457600 | 23457920 | 23458240 |
| Packet's received timestamp | 238ms | 265ms | 301ms | 330ms | 372ms |
| RTP difference from first * | ------- | 40ms | 80ms | 120ms | 160ms |
| REC difference from first | ------- | 27ms | 63ms | 92ms | 134ms |
| new start REC timestamp ** | 210ms | ------- | ------- | ------- | ------- |
| New REC timestamp *** | 210ms | 250ms | 290ms | 330ms | 370ms |

* the difference in milliseconds derived by using the RTP timestamps and the audio frequency
** the smallest result of (Packet's received timestamp – RTP difference from first)
*** the result of (new start REC timestamp + RTP difference from first)

By performing the calculation for the first few packets of the talkspurt (so that we get the likely smallest network delay for the packets – tests showed that after the first 6-7 packets we could only improve by 1-2 milliseconds) we calculate when the talkspurt start packet was sent. Based on this timestamp and the RTP timestamp in the header of each packet the rest of the talkspurt's packets are sent. As shown in the table above the amount of jitter varies and we eliminate this variation (e.g. for packet 2 we removed 15 milliseconds). We have to re-calculate at every new talkspurt to prevent the drift/skew of the sampling clock from becoming significant.

**Video** is more complicated as all packets that belong to a frame have the same RTP timestamp. Therefore to estimate when the first packet of the frame was sent we need to read ahead a few frames and use the same method applied to the audio using the first packet of each frame. We also measure the number of packets in the frame and the time difference from the next frame to calculate the time gap between packet transmissions. For example: frame 5 has 3 packets and calculated timestamp of 350ms and frame 6 has calculated timestamp of 410ms. The difference is 60ms divided by 3 gives a gap of 20ms. Therefore the first packet of frame 5 is sent at 350ms the next at 370ms and the last at 390ms. Data is sent on the network smoothly.

Using the RTP timestamps we eliminate the variation in the jitter imposed by the network during recording. That results in an acceptable level of inter-packet synchronisation as the data is sent to the network with the same order and inter-packet gap as the original sender intended. Packets are sent to the network smoothly avoiding bursts that may cause further losses and will arrive to the recipients with only the jitter variation imposed during playback. This makes the task of lip synchronisation [20] easier for receivers who expect packets to arrive with a logical jitter variation.

5.3 Results

MMCR is currently running on a Sun Sparc5 machine running Solaris. To date, the system has been successfully used for recording/playback of project meetings (8-12 participants each using audio, video and shared workspace), seminars and a high quality (video bandwidth ~500Kbps) surgical workshop. MMCR is also being used in remote language teaching by recording the lessons when they occur, making them available for future reference.

## 6. Conclusion and further work

There is currently a lot of interest in the field and a new protocol (RTSP) [21] for control of myltimedia servers is under development. MMCR will be made compliant with RTSP. The fact that the client is independent from the server allows users to supply their own RTSP clients. The facility to 'invite' the server to record/playback using SIP [22] will also be provided.

Other issues which will be examined in further work include:

a) distributed recording using local recorders at sites participating in the session
b) distributed playback - transcoders can be used in at the user's site in conjunction with the player to provide a variety of encodings or limit the bandwidth
c) decryption of encrypted streams to be able to extract RTP timestamps
d) provision of editing facilities and hierarchical indexes (as shown in 4.2)

This paper has presented the requirements and design/implementation of a multimedia recorder. The system is not complete, but its flexible architecture makes it quite promising.

## REFERENCES


[1] S.McCanne-‘vic’ manual page– Lawrence Berkeley Labs, University of California
[2] http://www.cs.ucl.ac.uk/mice/rat
[3] V.Jacobson, S.McCanne – “vat – X11 based audio teleconferencing tool” - vat manual page - Lawrence Berkeley Labs, University of California
[4] V.Jacobson, S.McCanne – “Using the LBL network whiteboard” - Lawrence Berkeley Labs, University of California
[5] M.Handley - Network text (nt) – “A scalable shared text editor for the Mbone” – Dept. of Computer Science – University College London – 1995
[6] M.Handley – “SDP: Session description protocol” – Internet draft – work in progress
[7] http://www-mice.cs.ucl.ac.uk/relate.html
[8] S.Casner – MBone frequently asked questions
[9] H.Eriksson – “Mbone: The multicast backbone” – Communications ACM, vol. 37, Aug. 1994
[10] H.Schulzrinne, S.Casner, R.Frederick, V.Jacobson – “RTP: A Transport Protocol for Real–time applications” – RFC 1889
[11] S.Floyd, V.Jacobson, S.McCanne – “A Reliable Multicast Framework for light-weight Sessions and Application Level Framing” – IEEE/ACM transactions on networking – Nov. 96
[12] W.Holfelder – “MBone VCR – Video Conference Recording on the MBone” – Proc. ACM Multimedia 95
[13] K.Almeroth, M.Ammar – “The Interactive Multimedia Jukebox (IMJ)” – Position paper for NOSSDAV 97
[14] P.Parnes, M.Mattson, K.Synnes, D.Schefstrom – “mMOD: the multicast Media-on-Demand system” – Submitted to WebNet97
[15] P.Kirstein, M.Handley, A.Sasse, S.Clayman – “Recent activities in the MICE project” – Proc. INET 95
[16] M.Buddhikot – “Design of a large scale multimedia server” – Proc. INET 94
[17] K.Nishikawa – “High performance VOD server aims” – Globecom 95
[18] P.Rangan–“Multimedia storage servers: a tutorial and survey”–IEEE Computer 95
[19] http://www-mice.cs.ucl.ac.uk/mice/merci
[20] I.Kouvelas, V.Hardman – “Lip synchronisation over the Internet: Analysis and implementation” – Globecom 96
[21] H.Schulzrinne, A.Rao, R.Lanphier – “Real Time Streaming Protocol (RTSP)” – Internet draft, IETF, Aug. 1997, Work in progress
[22] M.Handley, H.Schulzrinne, E.Schooler – “SIP: session invitation protocol” – Internet draft – Mar. 97 – Work in progress