Design of a System for Multimedia Transmission with Low Compression using JAVA and JMF

Dr. Kajal Sharma*
Dr. Sung-gaun Kim**
Dr. Manu Pratap Singh***

Abstract

The technology management and technology changes in IT industry are very critical for achieving the surviving competitiveness. To remain competitive in the global era in IT Company has to look for new application and new optimization technique or designing tools to support the operational technology mix in the global arena. This paper discusses the Develop optimization technique to secure transmission and use low compression coding, band width optimization, resulting in high quality video with high interactive streaming. Real-time Transport Protocol (RTP) is used for streaming the video. Technique critically evaluates an open and extensible presentation system for synchronized, Internet-based multimedia to support the technology technique for effective operational requirements. An algorithm has using to develop the system and implementing using a specific set of emerging technologies with Java and Java Media Framework (JMF). This paper analyze the optimization technique to support the operational technology in the IT environment.

Keywords **-** Video transmission, Java Media Framework (JMF), Real-time Transport Protocol (RTP)

1. Introduction

The Synchronization in multimedia systems refers to the temporal relations between media objects in multimedia systems (Figure 1). In future multimedia systems synchronization may also refer to spatial, content as well as temporal relationships. Synchronization between media objects comprises relationships between time-dependent media objects as well as time-independent media objects. If the presentations units of all units of a time-dependent media object are equal, it is called a continuous media object. Multimedia refers to the integration of text, images, audio, and video in a variety of application environments. These data can be heavily time-dependent, such as audio and video in a movie, and can require time-ordered presentation during use. The task of coordinating such sequences is called multimedia synchronization (Figure 2) [1].

*Department of Mech. & Automotive Eng., Kongju National University, Chonan, SOUTH KOREA
**Department of Mech. & Automotive Eng., Kongju National University, Chonan ,SOUTH KOREA
***Department of Computer Science, Dr. B. R. Ambedkar University, Uttar Pradesh, INDIA

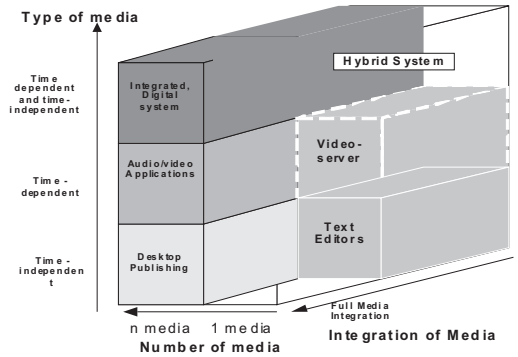

Fig 1: Multimedia Systems

Synchronization may need to occur at different levels in a multimedia system, consequently synchronization support is typically found in the operating system, communication system, databases, multimedia documents, and the application [2].

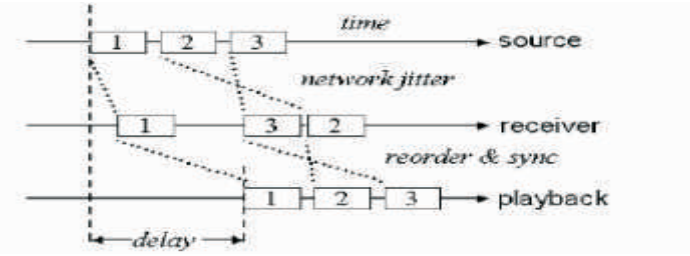

Fig 2: Illustration of synchronization mechanism

2. Media Quality of Services (QOS)

The QOS of media items is mainly related to how well an item can be presented according to its internal temporal structure, that is, its sampling rate or frame rate. The requirements on the network and presentation system become more severe as the information content or fidelity of the transmitted signal is increased. For time independent media types this discussion does not directly apply, since the presentation of such items typically involves retrieving the complete item before presenting it.

2.1 Audio

Audio signals are extremely sensitive to transients such as inaccurate sample values or variations in the playback rate. For example, the omission of a single sample during the playback of an audio stream may introduce a sharp transient in the rendered audio waveform.

2.2 Video

Human perception of video signals is not time-critical to the same extent as for audio. In particular it is difficult for a human to detect the exact frame rate of a video signal. It is impossible, for example, to distinguish between the frame rates of 25 frames/s and 24 frames/s.

3. Dimulation and Inter Object Synchronization

Intra-object synchronization: Intra-object synchronization refers to the time relation between various presentation units of one time-dependent media object. An example is the time relation between the single frames of a video sequence. For a video with a rate of 25 frames per second, each of the frames must be displayed for 40ms (Figure 3).

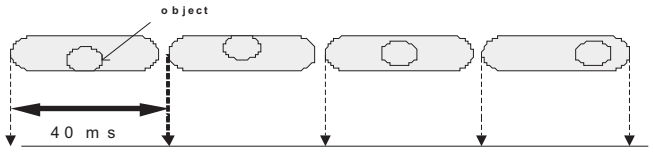

Fig 3: Intra –object Synchronization

Inter-object synchronization: Inter-object synchronization refers to the synchronization between media objects. Figure 4 shows that the time relations of a multimedia synchronization that starts with an audio/video sequence, followed by several pictures and an animation that is commented by an audio sequence [1].

Fig 4: Inter-object Synchronization

3.1 live and synthetic synchronization

The goal of live synchronization is to exactly reproduce at a presentation the temporal relations as they existed during the capturing process. In the case of synthetic synchronization, the temporal relations are artificially specified.

Model I: [Sync. by location alignment]
1. loop { /* playback a media-unit */
2. estimate video-waiting-time; /* Eq. (1) or Eq. (2) */
3. retrieving-and-playback-audio-segment();
4. retrieving-and-playback-videoframe();
5. waiting (video-waiting-time);
6.} until end-of-playback

$$\text{video-waiting-time} = \frac{\text{audiosegement-size}}{\text{audio-sampling size}} \quad (1)$$

$$\text{video-waiting-time} = \frac{\text{audiosegement-size} + \text{systemoverheads}}{\text{audio-sampling size}} \quad (2)$$

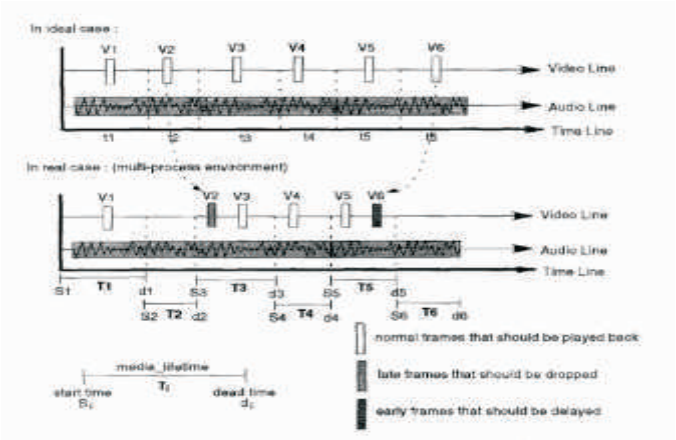

Fig 5: Media lifetime concept and the delay-or-drop policy.

In Model I, media-unit playback and synchronization control are embedded in a loop structure. The basic idea of the proposed approach is not to perform both the intra-media and the inter-media synchronization within a loop structure, but distribute the jobs of synchronization from only one process to many individual processes.

A good synchronization algorithm must guarantee both inter-media and intra-media synchronization within the given tolerable precision which is application dependent (Figure 5). At the same time, some video frames (V1,V2…..Vn) the associated audio segment in the same time interval should also be played back precisely under the enforced time-constraint to insure the intermedia synchronization. Some primitive functionality such as multiprocessors/multithreads creation, termination and intercommunication have been assumed for multitasking have been supported by the operating system. The synchronization can be achieved because the audio buffer is always filled and the audio data in the buffer is consumed at a constant rate, i.e., the audio can be played out smoothly along with the time axis and no audio discontinuity will occur [3, 4].

Definition 1: [Process Definition]: Two kinds of processes are defined in this synchronization model: the media-process (child process) and the control process (parent process). The media-process is responsible for both intra-media and inter-media synchronization. The control process is responsible for controlling the user interactive operations (such as rewind, pause) and some management functionalities of media process (such as generate or terminate a process). The main functions of the control-process include:

1. To pre-calculate the temporal information (such as video-waiting-time) for synchronization.
2. To generate (fork) other media-processes when media playback is started.
3. To terminate (kill) other media-processes when media playback is stopped.

Definition 2: [intra-media synchronization]: Intra-media synchronization must be achieved individually.

Definition 3: [inter-media synchronization]: Inter-media synchronization can be achieved cooperatively by each media-process.

According to the proposed model each child media process must undertake both the intra-media and the inter-media synchronization. In order to avoid the audio break phenomenon, the audio-process is designed to keep writing the sequence of audio segments to the audio device in a loop way. On the other hand, video-process adopts a delay-or-drop policy to deal with the out-of synchronization problem, as shown in Table 1, based on the definition of playback time interval (the so-called media-lifetime).

Table 1: Pseudo codes of the video and audio process.

Video Process	Audio Process

1. loop{
2. retrieving_video_frame(i);
3. get current_time from system clock;
4. if(current_time>dead_time((i)){
5. drop this video frame i;
6. jump to next appropriate frame i;
7. }
8. if(current_time<start_time((i)){
9. delay until(current_time >= start_time((i));
10. }
11. playback_video_frame(i);
12. }until end_of_playback
	1.loop{
	2.retrieving_audio_segment(i);
	3.playback_audio_segment(i);
	4.}until end_of_playback

Delay-or-drop policy: The corresponding video frame should be:Rule (1): dropped if the current-time is ahead of the dead-time.Rule (2): delayed if the current-time lags behind the start-time.Rule (3): played back if the current-time is within the media lifetime (i.e. in between the start-time and the dead-time)(Table 1).

4. Dynamic Reordering Mechanism

Dynamic reordering mechanism invokes a certain amount of buffer which accumulates packets before forwarding them to the next stage. At the same time, it reorders the packets according to the sequence number field in order to reduce the occurrences of packet out-of-order to a certain degree depending on the buffer size. In addition, if the packet which should arrive on time is not received before the time it is supposed to be passed to the next stage, we treat such a situation as a packet loss no matter it encounters a real loss or just a short-run delay. Note that such kind of packets can be recovered in the next stage [5, 6].

In order to put each incoming packet with a sequence number, we apply a circular data structure on the reordering buffer. Assuming that the system provides the buffer size N, we take the sequence number n of incoming packets and divide by N to get the remainder n mod N in order to decide the position for each packet. The procedure of dynamic reordering is described below.

4.1 Procedure DR: Procedure of dynamic reordering

Step 1. Initialize all slots of the reordering buffer.
Step 2. Assuming that the sequence number of the first packet is m, we set an expected sequence number E as m+1 and immediately send to the next stage a repackaged packet with the necessary information, including the sequence number, the timestamp, the payload type and the payload data. The expected number E is an indicator to decide the operation for each incoming packet by checking their sequence numbers.
Step 3. Wait for the next incoming packet. Once receiving the new packet, we check its sequence number n.
Step 3.1 If n is smaller than E, it implies that the packet is obsolete. Skip this packet and return to Step 3.
Step 3.2 If n equals to E, meaning that this packet is what we expect, we check the slots from n mod N to n+N-1 mod N of the

reordering buffer sequentially until we encounter an empty slot. If slot k mod N ($n \leq k \leq n+N-1$) is not empty, pop out the repackaged packet to the next stage and set E as k+1. Otherwise stop the checking process and then return to Step 3. This step will flush out all continuous packets starting from n.

Step 3.3 If n falls between E and E+N, it implies that the packet we expect has not arrived yet. We shall wait for the expected packet to reorder incoming packets. Thus, we store the necessary information of the incoming packet in slot n mod N and then return to Step 3.

Step 3.4 If n is equal to or larger than E+N; it implies that the packet we expect has not arrived yet. However, in this case we cannot wait any longer because of the buffer overflow. Hence, we treat these packets with sequence number from E to n-N as lost packet and clear correlative slots. In addition, we store the necessary information of this incoming packet in slot n mod N. Finally, check the slots from n-N+1 mod N to n mod N of the reordering buffer sequentially until we encounter an empty slot. If slot k mod N ($n-N+1 \leq k \leq n$) is not empty, pop out the repackaged packet to the next stage and set E as k+1. Otherwise stop the checking process and return to Step 3.By procedure DR, the buffer accumulates dynamically only if incoming packets encounter the situation of packet out-of-order or packet loss. If incoming packets are received sequentially, we can pass them to the next stage in time without any extra processing delay [7, 8].

5. Java Media Framework (JMF)

The Java Media Framework (JMF) is an application programming interface (API) for incorporating time-based media into java programs. It provides support for media playback, capturing and storing media data and performing custom processing on media streams. It supports media data reception and transmission using RTP and RTCP. The JMF RTP APIs are designed to work seamlessly with the capture, presentation and processing capabilities of JMF. Players and processors are used to present and manipulate RTP media streams just like any other media content (Figure 6) [9, 10].

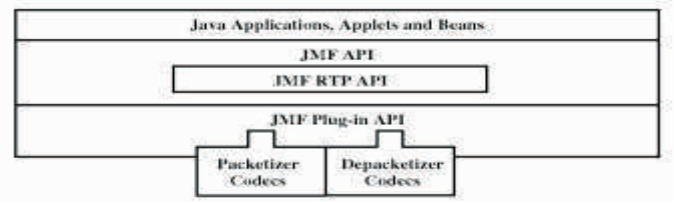

Fig 6 : JMF RTP Architecture

6. Transmission of RTP Media Streams

Following steps are needed to create a send stream to transmit data from a live capture source: Create, initialize, and start a Session Manager for the session.

1. Construct a Processor using the appropriate capture Data Source.
2. Set the output format of the Processor to an RTP-specific format. An appropriate RTP packetizer codec must be available for the data format we want to transmit.
3. Retrieve the output Data Source from the Processor.
4. Call create Send Stream on the session manager and pass in the Data Source.

We can control the transmission through the SendStream start and stop methods.

6.1 Real-time Transport Protocol (RTP) and Real-Time Control Protocol (RTCP)

The Real-Time Transport Protocol (RTP) handles transport issues specifically related to real-time data. RTP includes another protocol, Real-Time Control Protocol (RTCP), for managing RTP sessions. Some of the main responsibilities of RTP/RTCP are: packet sequencing, synchronization, payload identification, QOS feedback and encryption [11, 12].

Real-time Streaming Protocol (RTSP) is a protocol for initiating and controlling the delivery of both stored and live multimedia streams over the Internet. The main concept of RTSP is providing a session-like abstraction for delivering one or more media streams to a single client or multicast destination. The main responsibilities of RTSP are initiating a session, controlling a session while active signaling the end of a session.

7. Real Time Multimedia Processing and Session Manager

Applications that use RTP can be categorized into RTP servers (applications that need to send data over the network) and RTP clients (those that need to receive data from the network). However, some applications, such as teleconferencing, establish RTP sessions to capture and transmit, as well as receive data. Session Manager keeps track of the session participants and the streams being transmitted, as well as handles the RTCP control channel and supports RTCP for both senders and receivers.

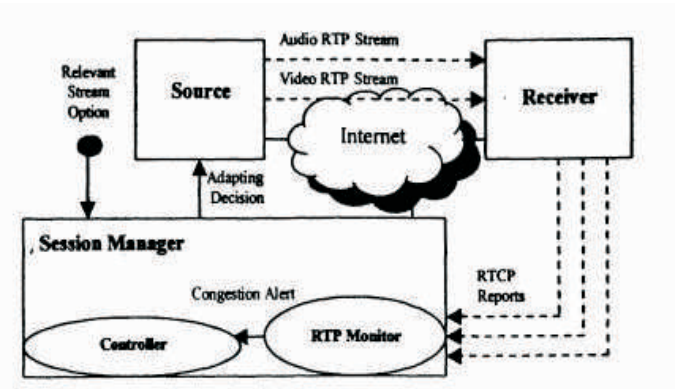

Fig 7 : System Architecture

A typical Internet video streaming system is shown in figure 7 and it works as follows: the system is composed of a source and a receiver, or a group of receivers, which all become members of a multicast group by joining this group using a multicast IP address and a given port number. The source sends to the receiver its compressed live audio and video content for the streamed session each on a separate RTP stream. The receiver is capable then of receiving the RTP streams and playing them back. During the session time each receiver issues a series of RTCP reports for each received stream periodically. These reports help in identifying the most recent receiving status of this receiver mainly regarding jitter, the number of packets lost, and its fraction from the sent packets. The simplest way to transmit RTP data is to use a Media Locator with the RTP session parameters to construct an RTP Data Sink by calling the Manager

object's createDataSink() method. Session Manager can also be used to create send streams for the content and control the transmission. A Player is created using the Manager object's createPlayer() method, passing a Media Locator as the argument [13, 14,15].

8. Results & Discussion

The audio and video data is transmitted in unicast and multicast network. It is first captured and then transmitted through network and then received on the receiving side. The synchronization of these audio and video streams is done by using buffering techniques and then played on the receiving side (Figure 8 & 9). Then the same method is applied on real time media streams by capturing streams and use Real Time Protocol (RTP) and then synchronization is performed.

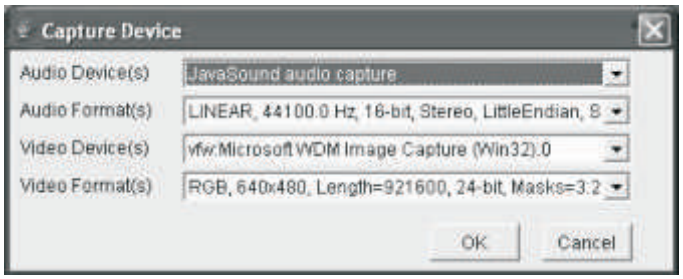

Fig 8 : Capture Devices and device formats

Fig 9 : Reception of media streams on receiving side

8.1 CPU requirement

We analyze the CPU and memory requirement of the client application by simulating the environment of different number of participant in the conference on a P-IV computer with 2.4 GHz speed and 512 MB RAM (Table 2).

Table 2: CPU and memory usage of the application

Number of Targets	Number of Receivers	CPU Usage	Memory Usage
0	0	8%	30 MB
1	1	38%	45 MB
5	5	55%	65 MB
10	10	87%	85 MB
12	12	92%	95 MB

8.2 Bandwidth requirement

To analyze the network bandwidth required for transmitting audio and video streams of single user we vary the bit rate of H.263 encoder and observe the quality on video stream at the receiver end (Table 3).

Table 3: Bandwidth requirement

Bit Rate in kbps	Frame Rate	Frame Size	Video Quality
300	15	352X288	As captured
200	15	352X288	Good
100	15	352X288	Still acceptable
50	15	352X288	Slightly degraded

8.3 Performance against packet loss

To observe how the quality of video effected with packet losses, we introduce some packet loss by arbitrarily dropping some fraction of packets in the RTP connector module and observe the video quality at the receiver end. For high packet losses we choose the key frame rate small so that receiver can resynchronize more quickly after a loss of packet. Up to five percent packet loss can be tolerated (Table 4).

Table 4: Performance against packet loss

Packet Loss (%)	Key Frame Rate	Video Quality
0.5%	15	No much effect
1.0%	12	Slightly disturbed
2.0%	10	More disturbed
5.0%	5	Poor

9. References

1. *G. Blakowski and R. Steinmetz; "A Media Synchronization Survey: Reference Model, Specification, and Case Studies," IEEE J. Selected Areas in Comm., vol. 14, no. 1, Jan. 1996, pp. 5-35.*
2. *Little, T. D. C; Ghafoor;"A. Synchronization and Storage Models for Multimedia Objects", IEEE Journal on Selected Areas in Communications, Apr., 1990.*
3. *Y. D. Chawathe. Scattercast;" An Architecture for Internet Broadcast Distribution as an Infrastructure Service", Ph.D. Thesis, University of California, Berkeley, December 2000.*
4. *Y. Chu, S. G. Rao, S. Seshan and H. Zhang;" Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture", In Proceedings of ACM SIGCOMM, pp.55-67, San Diego, CA, August 2001.*
5. *Peterson, L. Larry and Davie, S. Bruce; "Computer Networks: A*

Systems Approach", San Francisco, CA: Morgan Kaufmann Publishers, 2003. Third Edition.

6. *B. Zhang, S. Jamin, and L. Zhang; "Host multicast: A framework for delivering multicast to end users", In IEEE Infocom, 2002D.*
7. *Atushi Koguchi and Hidenori Nakazato and Hideyoshi Tominaga;" A Tree Routing Method on Multi-Tree Application Level Multicast Streaming System", 6 2005.*
8. *G. S. Blair, M. Papathomas, G. Coulson, P. Robin, J. B. Stefani, F. Horn, and L. Hazard; "A Programming Model and System Infrastructure for Real-Time Synchronization in Distributed Multimedia Systems", IEEE Journal on Selected Areas in Communications, Jan., 1996.*
9. *Sun Microsystems, Inc. JavaBeans API Specification. Version 1.01. Mountain View, CA, USA: Sun Microsystems,Inc., 1997. 114 p. Available www:http://java.sun.com/beans/docs/beans.101.pdf*
10. *Java Media Framework Home Page: java.sun.com/products/java-media/jmf/index.html http://www.javaworld.com/javaworld/jw-04-1997/jw-04-jmf.html*
11. *L. Berc, W. Fenner, R. Frederick, S. McCanne, and P. Stewart;"RTP payload format for JPEG-compressed video," Internet Engineering Task Force, RFC 2435, 1998.*
12. *Y. Chu, S. Rao, S. Seshan, and H. Zhang;"A case for end system multicast," Selected Areas in Communications, IEEE Journal on, vol. 20, no. 8, pp. 1456–1471, 2002.*
13. *C.Huang, H.Kung, and J.Yang; "Synchronization and flow adaptation schemes for reliable multiple-stream transmission in multimedia presentations", Journal of Systems and Software, V. 56, Issue 2, 1 March 2001, Pages 133-151.*
14. *The HyperCast project. http://www.cs.virginia.edu/hypercast.*
15. *P. Druschel, M. Castro, A.-M. Kermarrec, and A. Rowstron;" Scribe: A large-scale and decentralized application-level multicast infrastructure", IEEE Journal on Selected Areas in Communications, 20, 2002.*