2007

DYNAMAC media distribution system

Luis A. Caceres Chong

DYNAMAC MEDIA DISTRIBUTION SYSTEM

BY

LUIS A. CACERES CHONG

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
TELECOMMUNICATIONS ENGINEERING TECHNOLOGY

ROCHESTER INSTITUTE OF TECHNOLOGY
March 2007

MASTER OF SCIENCE THESIS

OF

LUIS A. CACERES CHONG

APPROVED:

Thesis Committee

____

Chair - Dr. Chance M. Glenn Sr.

____

Dr. Warren L. G. Koontz

____

Mark Indelicato

ROCHESTER INSTITUTE OF TECHNOLOGY

March 2007

Library Rights Statement

In presenting the thesis DYNAMAC Media Distribution System in partial fulfillment of the requirements for an advance degree at the Rochester Institute of Technology, I agree that permission for copying as provided for by the Copyright Law of the U.S. (Title 17, U.S. code) of this thesis for scholarly purposes may be granted by the Librarian. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my written permission. I hereby grant permission to the RIT Library to copy my thesis for scholarly purposes.

Luis A. Cáceres Chong

Date

Abstract

This thesis describes the initial development of a multiplatform client-server based system capable of distributing content derived from a new format over an IP network. The system will also provide digital rights management and secure transmission of video content utilizing the DYNAMAC video compression/decompression algorithm. The work presented here is part of the Digital Media Research Group, which is focused on the use of the DYNAMAC algorithm for digital rights management and distribution of high definition media content in real time, while achieving better efficiency on bandwidth usages. Three components were created with this initial development: a player capable of rendering video content compressed with the DYNAMAC algorithm, a server that distributes the compress video content over an IP network, and a protocol specially design to communicate and transfer the video content from the server to the client. Since the development was based on the Java Media Framework, the client and the server of the distribution system will be able to be installed and run on any computer platform with a Java Runtime Environment. The fact that the system is written in Java also means that any changes can be made to single components without having to change the entire system. In this paper I intent to describe the approach taken to develop the system and the results obtain, were several videos are transmitted demonstrating the functionalities of the DYNAMAC algorithm and the DYNAMAC media distribution system.

Keywords

video compression, IPTV, DYNAMAC algorithm, Java Media Framework, digital rights management, telecommunication

Acknowledge

First of all I would like to thank God for allowing me to get this far in life and for giving me the strength to do all the things I have done.

Secondly, I would like to thank my country Honduras for letting me grow up surrounded by so much beauty and for allowing me to be a part of a great people full of strength and hope. They taught me the value of a having to work hard for what we want.

I would also like to thank my family and Sara. My mother and father for always being there pushing me to be a better person everyday. Sara for never losing hope on me and always telling me that no matter what happens everything is going to be fine.

Thanks to Dr. Chance M. Glenn, for letting me be a part of his work and having faith in me that I wouldn't let him down.

Also, thanks to Dr. Warren Koontz, for always having his door open and for always listening to me when I had a problem or a doubt.

Last, but not least, I would like to thank my friends, for believing that I'm better than what I really am, and forcing me to live up to their ideal.

Table of Contents

Chapter 1

1. Introduction

From its beginnings of providing telegraph services and plain old telephone services (POTS), telecommunications have evolved into a whole new era, giving the world services such as voice over internet protocol (VoIP), internet protocol television (IPTV), video-on-demand, Internet access, mobility, and many other applications.

Thanks to the convergence of voice, video, and data, the telecom industry can now take advantage of the networks already in place and use them to help people communicate with the rest of the globe. We now don’t just read about current events happening on the other side of the planet, but we can also watch and listen to these events as they unfold live. If you already have access to the Internet, you can talk to your loved ones for free. Sitting in front of their computers at home, people can search for movies or music videos from anywhere on Earth and view them in a matter of seconds.

Services such as Video on Demand and IPTV are achieved thanks to streaming. Streaming offers a whole new approach to media on the Internet[1]. Instead of waiting for the whole file to download to a user's computer before playback begins, streaming media playback occurs as the file is being transferred. The data travels across the Internet, is played back and then discarded. Streaming is also considered real-time because you may view an event as it happens. The primary factor that makes a stream real-time is that there is no intermediate storage of the data packets[2]. There may be some short buffers, like frame stores in the decoder, but the signal essentially streams all the way from the camera to the player.

Although the capacity of modern networks is very high, streaming puts more demand on the network's bandwidth. In order to be able to provide these services, compression is required. Compression is the art and science of squeezing out unneeded information in a picture, or a stream of pictures (a movie) or sound before sending or storing it[3]. In transmission systems, compression allows a reduction in bandwidth which will generally result in a reduction in cost. This may make possible a service which would be impracticable without it. If a given bandwidth is available to an uncompressed signal, compression allows faster than real-time transmission in the same bandwidth. If a given bandwidth is available, compression allows a better-quality signal in the same bandwidth.

Also as new technologies appear, ill-intentioned persons try to find ways to use those technologies to their advantage regardless of the laws they break or the people they hurt. In this case it could be the piracy of digital entertainment: music, movies, or games[4]. Some Internet piracy is in the domain of the lone hacker, who treats unauthorized access as a challenge. A bigger problem is piracy for commercial gain. Therefore, digital assets have to be wrapped securely to prevent access to all but authorized users that have been properly authenticated. File security and digital rights management becomes more of an issue when content is delivered over the public Internet. Digital rights management allows owners, distributors, and providers of content to deliver content securely to consumers as part of an e-commerce system.

1.1 DYNAMAC Algorithm

For some of the reasons mention above, and others, the DYNAMAC algorithm was created[5]. The DYNAMAC algorithm is based in nonlinear systems theory. It is able to achieve significant compression of audio and image data while maintaining good signal quality. Because of the way the compression/decompression algorithm works, it has an aspect of encryption useful for digital rights management and secure transmission. The basic foundation of the DYNAMAC process lies in the realizations that (a) chaotic oscillators are dynamical systems that can be governed by mathematical expressions, and (b) chaotic oscillators are capable of producing diverse waveform shapes. The premise is this: a segment of a digital sequence, such as that derived from image data, can be replaced by the initial conditions of a chaotic oscillation that matches it within an acceptable error tolerance. If the size of the data needed to specify the conditions needed to reproduce the chaotic oscillation is smaller than the size of the digital sequence, compression is achieved.

"Chaos is a name for any order that produces confusion in our minds."
—G. Santayana, Philosopher, Poet, Humanist and Critic (1863-1952)

The concept behind the DYNAMAC algorithm is that the binary sequences used by computers to store and represent images and sound can be represented by waveforms generated by a chaotic oscillator.

To implement this concept we could sit around and wait an infinite amount of time for a chaotic oscillator to randomly produce in just the right order the waveforms needed to represent a specific image (somewhat like the Infinite Monkey Theorem[6]), or we could work our way from the image and arrive to the waveforms, just like the DYNAMAC algorithm does. It divides the image into sets of digital sequence and searches inside a combined chaotic oscillation matrix to find the closest match between a waveform and the digital sequence. The waveforms inside the matrix are identified by a 40-bit sequence or d-bite. For example, if the waveform selected represents a 256-bit sequence, the DYNAMAC algorithm has compress the digital sequence from 256 bits to 40 bits.

To access a waveform inside the combined chaotic oscillation matrix, the DYNAMAC algorithm requires an ordering key. The index of this ordering key is part of the 40-bit d-bite use to identify a waveform. That's how the DYNAMAC algorithm produces digital rights management. If the wrong key is used in the decompression process, the result will be a scramble image.

1.2 Objective

The objective of this thesis is to design and develop a multiplatform client-server based system capable of distributing content derived from a new format over an IP network. The system will also provide digital rights management and secure transmission of video content utilizing the DYNAMAC video compression/decompression algorithm.

To accomplish this purpose, a new network protocol will be created with the purpose of establishing the communication standard for the client and server. The new protocol will be designed specifically to transfer streaming video compressed by the DYNAMAC algorithm. The user interface will be designed in the JAVA programming language.

An added bonus is that the system will prove to be a valuable tool for The Laboratory of Advanced Communications Technology (LACT) and the Digital Media Research Group (DMRG). This system will be a platform for further technology development.

1.3 Thesis Road Map

This chapter explains the scope of the thesis. A small background and a description of the DYNAMAC algorithm have been provided. The rest of the thesis is organized as follows:

Chapter 2 provides the approach taken to develop the distribution system.

Chapter 3 discusses the results obtained by following the approach mention on the previous chapter.

Chapter 4 proposes some future work that may be required to improve the distribution system and the DYNAMAC Project.

Chapter 5 summarizes the conclusions obtain from the work done in this thesis.

Chapter 2

Approach

Although the DYNAMAC algorithm has been proven to achieve compression and encrypt data to preserve digital rights[7], still there isn't a system capable of demonstrating the algorithm's potential to its fullest. The system should be capable of streaming the compressed video content over an IP network to serve several clients, as well as preventing a client without the proper authentication mechanism viewing the video content.

The first step is to create a DYNAMAC video client interface capable of decompressing the video content and displaying it on screen. The client should be able to read DYNAMAC video files or dyv files as well as connect to a DYNAMAC video server using TCP/IP and receive the compressed data. The client is also required to read a combined chaotic oscillation matrix and the matrix ordering key, which are needed for the decompression process.

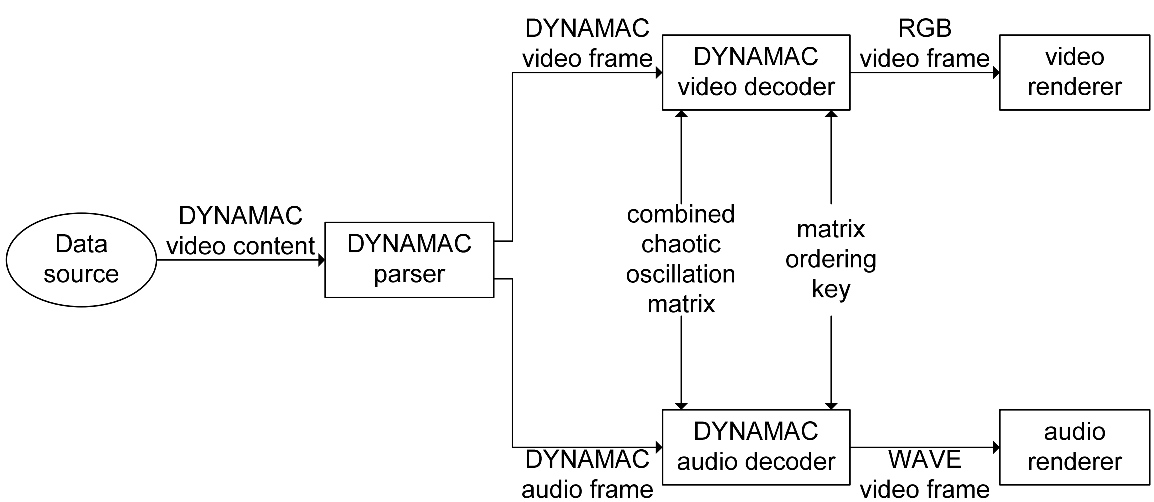


Figure 1: DYNAMAC video decompression and rendering process

The dyv file consists of an interleaving between DYNAMAC compressed video frames and DYNAMAC compressed audio frames. The first step is to split the image and audio content of the file or incoming stream. The decoders receive the DYNAMAC compressed frame and decompress it using the combined chaotic oscillation matrix and the ordering key. Finally the raw video and audio streams are rendered through the screen and speakers respectively.

The next step is to create a DYNAMAC video server capable of handling several TCP/IP request by the clients trying to connect to him. The server will have a repository where dyv files are going to be stored. When a client requests a specific file, the server reads the dyv file and starts streaming the compressed video content back to the client.

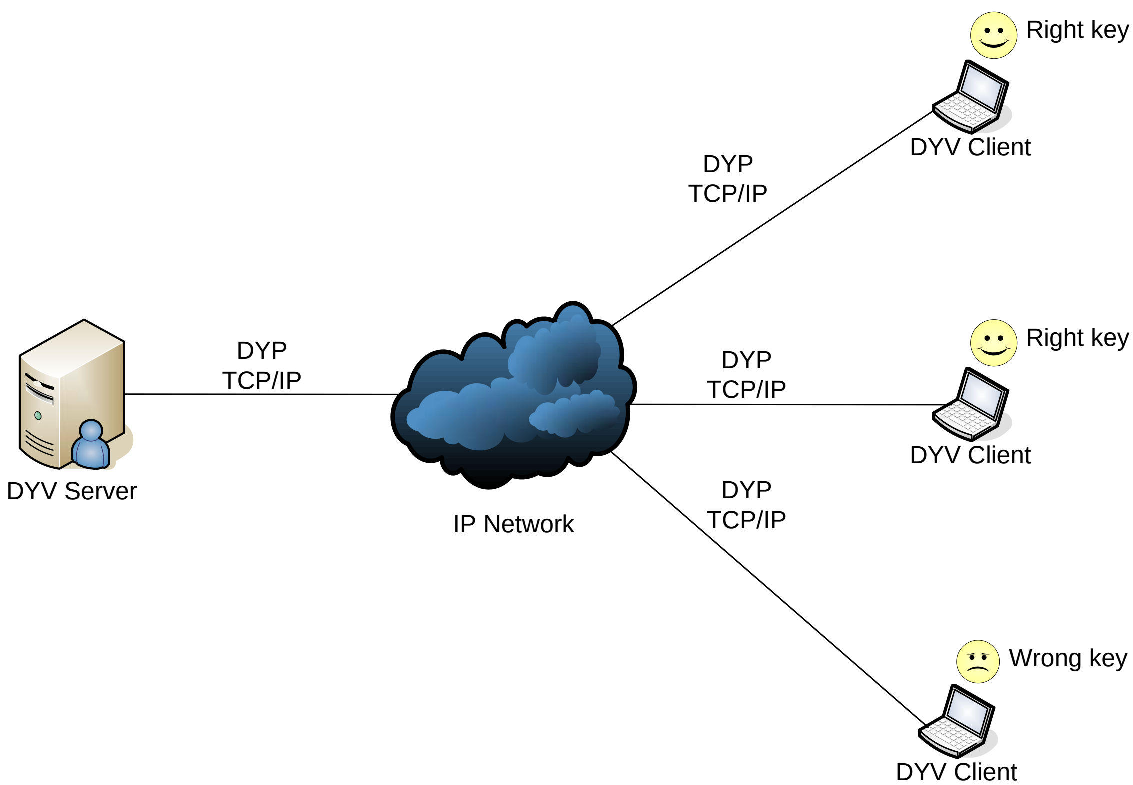


Figure 2: DYNAMAC distribution system network topology

All the clients and the server are going to be connected through an IP network. The clients are going to request the video from the server. The server is going to stream the dyv video. The clients are going to start decoding and rendering the video (as shown in Figure 1). If a client has a wrong matrix ordering key, the video is going to be displayed scramble.

The final step is to create the protocol, or DYNAMAC protocol (DYP), used by the server and the client to communicate with each other. Besides specifying the steps used to start video streaming, the protocol packet should indicate the number of d-bites in its payload, as well as the type of d-bite (image or audio).

2.1 DYNAMAC Video client

The DYNAMAC video client or DYNAMAC player consists of several components (see figure 1) required to process the video content. These components are use in four processing stages: accessing the video content, demultiplexing, decoding, and rendering. These four stages are executed every time a video is played, whether the video source is a file located on the local disk, or a media server located on the other side of the planet.

2.1.1 Accessing Video Content

The first stage of the DYNAMAC player is to access the video content. A *Datasource* component is required for each protocol use to access the media. It is the interface between the DYNAMAC player and the media repository.

The *Datasource* used depends on the protocol required to read the media content. For example, if the video file is located on a computer's local drive, the *Datasource* associated with the FILE protocol is loaded. To access DYNAMAC videos two *Datasource* were created: a *Datasource* for the FILE protocol, and a Datasource for the DYP protocol.

A *Datasource* knows how to connect and how to read the media content. It also knows the MIME type of the media content it can handle, but the *Datasource* is unaware of the actual structure of the video content. In our case, our *Datasources* know how to connect to the DYNAMAC videos depending on the access protocol, and they know that they handle "dynamac/video" content type, but the only thing they know about the file's inner format is the header structure.

The FILE protocol *Datasource* opens a dyv file located on the computer's hard drive. No further work is required other than opening the file and reading the bytes from the file. The DYP protocol *Datasource* is a little more complicated. It needs to know the IP address and the TCP port number of the DYNAMAC Media Server. It also needs to know the DYP policies that will allow him to communicated and obtain the video content from the server.

2.1.2 Demultiplexing

As mention earlier, a DYNAMAC video file holds both audio and video content. This interleaved media is access through the same *Datasource*, so a demultiplexer is required to extract both video and audio tracks. For this reason, the DYNAMAC parser was created. The DYNAMAC parser takes the interleaved media stream from the *Datasource*, extracts the tracks from the Datasource, and provides methods so the player can read an entire DYNAMAC video or DYNAMAC audio frame depending on the track.

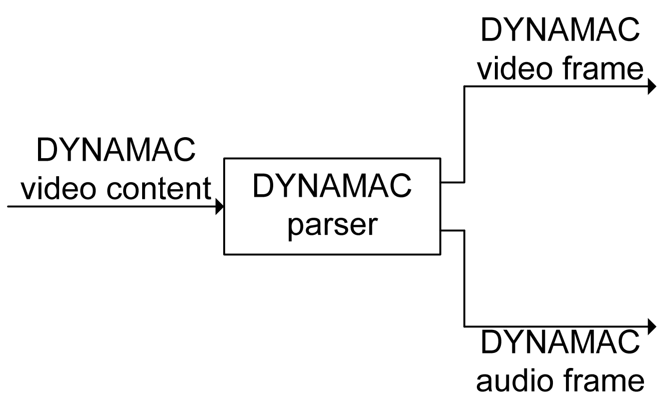


Figure 4: DYNAMAC parser data flow diagram

Unlike the *Datasource*, the DYNAMAC parser has to know the structure of the media content in order to split the video stream into image and audio.

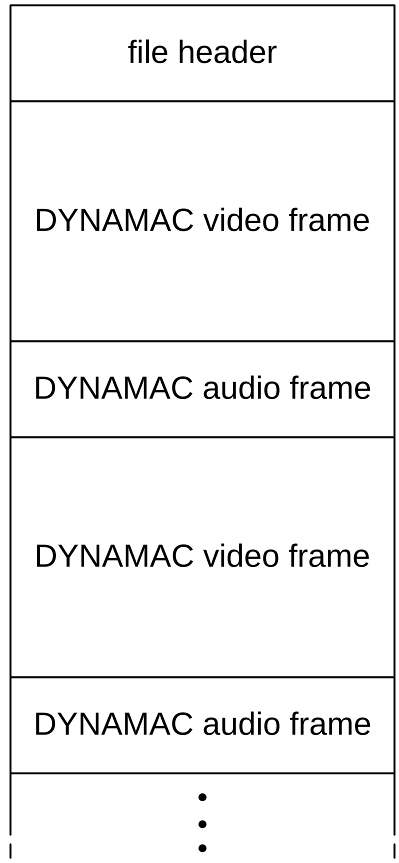


Figure 3: DYNAMAC video file structure

The video file structure consists in three parts: the file header, a video frame, and an audio frame.

2.1.2.1 File Header

byteorder	# frames	framerate	sample per second	# image channels	# audio channels	image Ns	audio Ns	iX	iY
1 byte	4 bytes	1 byte	2 byte	1 byte	1 byte	1 byte	1 byte	2 bytes	2 bytes

Figure 4: File header structure

The file header consists of 16 bytes divided into 10 fields. The fields on the file header are used to determine the amount of d-bites in a video frame and the amount of d-bites in an audio frame. The first field is the byteorder which indicates the byte order (big endian or little endian) used while writing the file. The # frames field is the number of frames contain in the file or stream. The sample per second field indicates the sampling rate of the audio frames. The # image channels field indicates if the video frame is based on a 8 bit pixel or 24 bit pixel. The # audio channels field indicates if the audio track is mono or stereo.

The image Ns and audio Ns fields represent the DYNAMAC transform factor used for compression. Finally the iX and iY fields are the number of width and length pixels of the video frames.

2.1.2.2 DYNAMAC video frame

blue d-bite	green d-bite	red d-bite	blue d-bite	green d-bite	red d-bite		...

Figure 5: d-bites inside a video frame

A usual video frame consists of 24-bit pixels, where every 8 bits represent a color. The different combinations of red, green, and blue provide and wide range of colors. A DYNAMAC video frame is no different. The DYNAMAC video frame is build up of several d-bites representing the red, green, and blue components of the original video frame. So the combination of blue, green, and red d-bites represents a sequence of pixels in the frame.

2.1.2.3 DYNAMAC audio frame

left d-bite	right d-bite	left d-bite	right d-bite		...

Figure 5: d-bites inside an audio frame

A wave sound usually consists of one or two channels. When a wave sound has only one channel it is call mono, and when a wave sound has two channels it is call stereo. So a stereo song has a left channel and a right channel. And just like a DYNAMAC video frame resembles a RGB video frame, the DYNAMAC audio frame resembles a wave audio frame by compressing the left channel and the right channel into d-bites. So each pair of d-bites represents the left channel and the right channel of a sequence of stereo samples in the audio frame.

2.1.3 Decoding

Nstart	Ntype	Nm	IMin	IMax
11 bits	5 bits	8 bits	8 bits	8 bits

Nstart	Ntype	Nm	AMin	AMax
11 bits	5 bits	8 bits	16 bits	16 bits

Figure 6: video and audio d-bite structure

As mention earlier, the information required by the DYNAMAC algorithm to match specific waveforms on the chaotic matrix with the digital sequence that it represents on a video or audio frame is call a d-bite. Regardless that the d-bite may be an audio d-bite or a video d-bite, it is made up of five fields. The Nstart field represents the X position in the chaotic matrix of the waveform that resembles a bit sequence of the original frame. The Ntype field is used as the index of the ordering key. The value of the ordering key represents the Y position in the chaotic matrix of the waveform mention before. The length of the waveform is determined by the Ns value in the file header and the Nm value of the d-bite. After normalizing and processing the waveform in relation to the Min and Max fields of the d-bite, the resulting bit sequence is used to recreate the media frame.

So after the player has demultiplexed the media stream into an audio track and a video track, each track needs to be decoded. That’s exactly what the DYNAMAC video decoder and DYNAMAC audio decoder do.

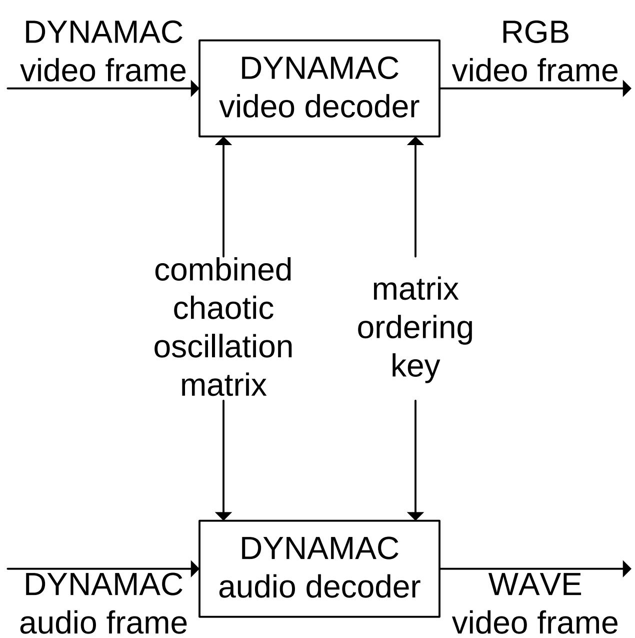


Figure 7: DYNAMAC decoder data flow diagram

Each decoder processes on a frame by frame basis. The DYNAMAC video decoder starts by receiving the DYNAMAC video frame. Once the frame is ready to be processed, the video decoder reads a red, green, and blue set of d-bites to get the waveforms inside the chaotic oscillation matrix. These waveforms are then normalized and combine to create the sequence of pixels compressed by the DYNAMAC algorithm. The process is repeated until all the d-bites are read and the RGB video frame is full.

The DYNAMAC audio decoder does a similar process. After receiving the DYNAMAC audio frame, the audio decoder reads the left and right d-bites to get the waveforms inside the chaotic oscillation matrix. The waveforms are then normalized and combine to create the samples of wave sound compressed by the DYNAMAC algorithm. This process is repeated until all the d-bites are read and the WAVE audio frame is full.

2.1.4 Rendering

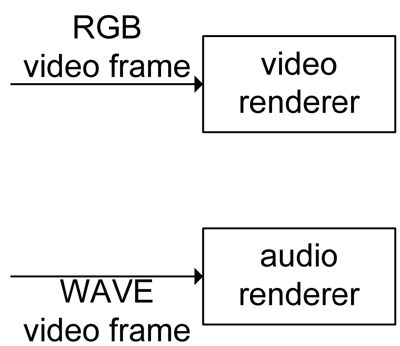


Figure 8: Final step of the DYNAMAC player

Rendering is the process of showing the media content in an output device. Once the DYNAMAC frame is decompress, the final step is to render it. The output of the DYNAMAC video decoder is render on screen, while the output of the DYNAMAC audio decoder is render on the computer speakers.

2.2 DYNAMAC Media Server

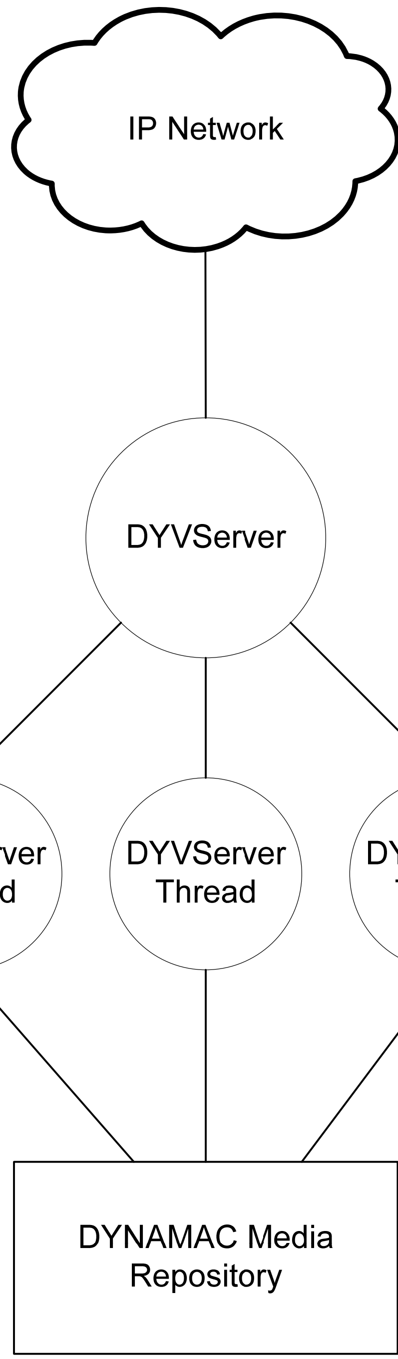


Figure 9: DYNAMAC Media Server Component Diagram

The DYNAMAC Media Server consists of three components required to provide several clients with DYNAMAC media content at the same time. The DYVServer is the main component in charge of accepting the request for connection that comes from the IP network.

Once the connection has been established, the DYVServer creates a new DYVServerThread to handle that new media request. The DYVServerThread starts communicating with the client using the DYNAMAC protocol and access the media content located on the DYNAMAC Media Repository.

The DYNAMAC Media Repository is made up of all the DYNAMAC media files located on the server, and a list of these files.

2.3 DYNAMAC Protocol

As mention earlier, the DYNAMAC protocol is the standard followed by the DYNAMAC client and the DYNAMAC media server to communicate with each other. With the DYNAMAC protocol they are able to exchange the information and d-bites required to play the media content.

2.3.1 DYNAMAC Protocol Process

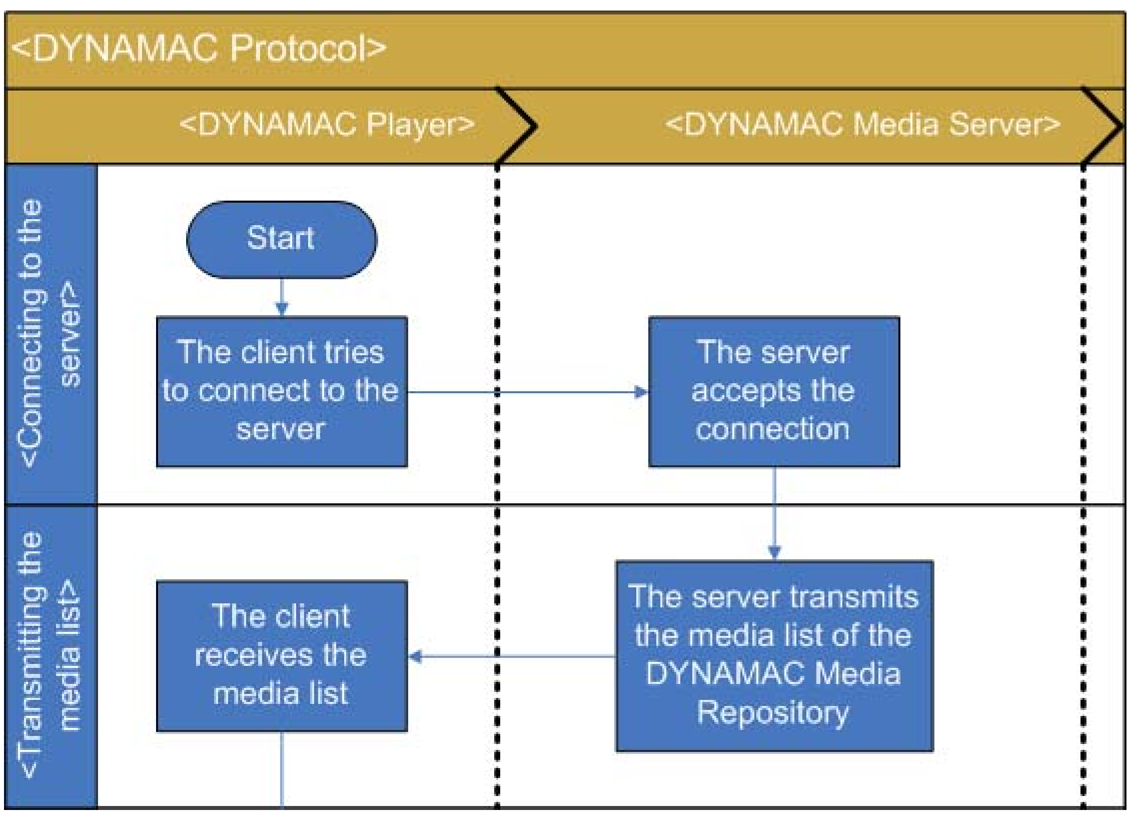


Figure 10: DYNAMAC protocol process (part 1)

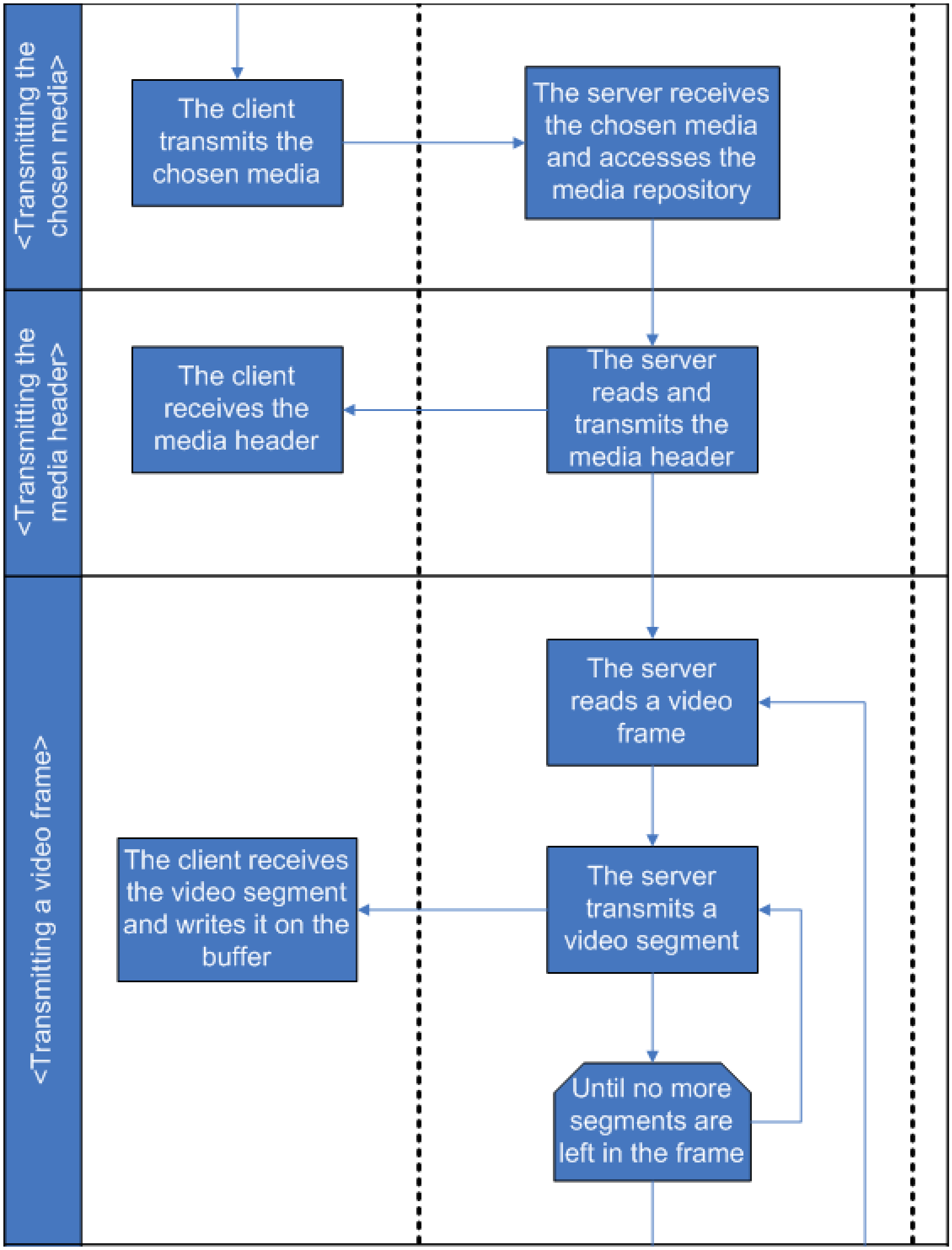


Figure 10: DYNAMAC protocol process (part 2)

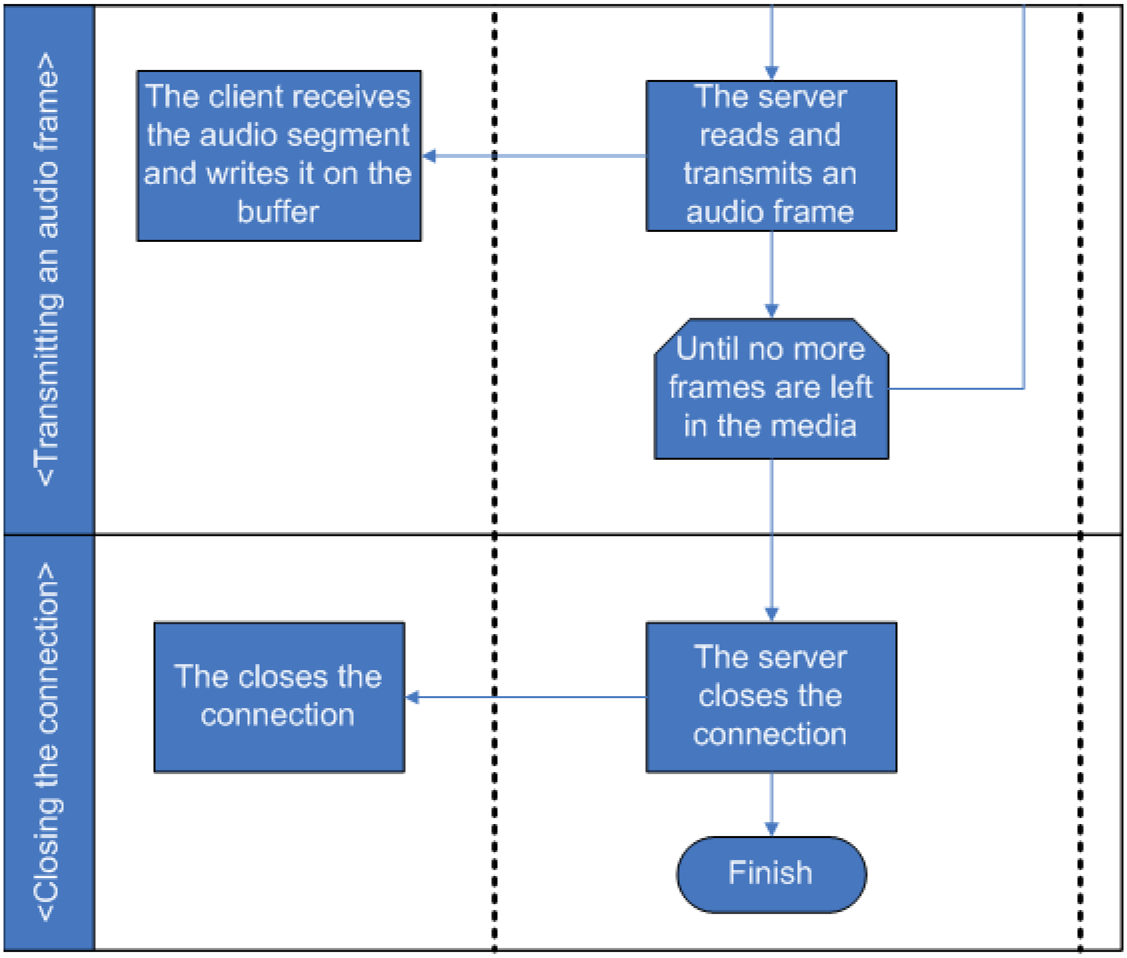


Figure 10: DYNAMAC protocol process (part 3)

The protocol starts when the client attempts to make a TCP/IP connection with the DYNAMAC media server. The media server is constantly listening for any incoming connections, when the client tries to make the connection, the media server accepts it. Once the connection has been established, the media server transmits a list of all the media content residing in the DYNAMAC Media Repository on the server's side. Once the media list is on the client side, a media is selected and transmitted to the media server. The server then opens the media content and sends the header (see Figure #4) to the client. Both the server and the client use the media header to calculate the amount of frames that are going to be transmitted, the amount of segments inside a video frame, and the size and order of each packet. Finally the media frames are transmitted.

Since the media content is interleaved video and audio content, the same thing happens in the transmission. First a video frame is sent and then an audio frame is sent. Because of the size of a video frame, the frame is divided into segments. Every segment is transmitted in a single packet, until the entire video frame has been sent. Then an entire audio frame is transmitted. These steps are repeated until the entire media content has been sent.

To finish the protocol process, the media server closes the connection with the client.

2.3.2 DYNAMAC Protocol Packet

type	frame #	segment #	video payload
1 byte	4 bytes	2 bytes	variable length

type	frame #	audio payload
1 byte	4 bytes	variable length

Figure 11: DYNAMAC protocol media packet structure

As mention in the protocol process, the size of the packets used to send the actual media content are decided at run time by the media header. Using the header, the DYNAMAC player and media server know the amount of d-bites in a video and audio frame, and the width of a video frame used to divide the frame into segments.

The packet also contains information required to reassemble the media content on the other side. For instance, the video packet transmits the frame number and the segment number of the video frame the d-bites in the payload belong to. The same goes for the audio packets; they contain the frame number of the frame the audio d-bites belong to. Both packets contain a type field that identifies them. A type 1 indicates a video packet and a type 2 indicates an audio packet.

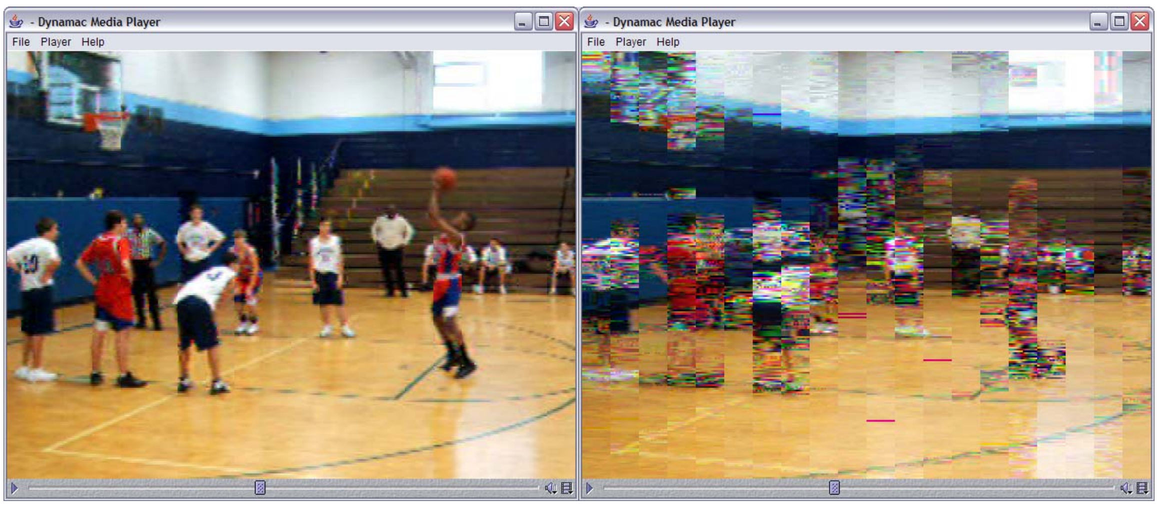
The length of both video packets and audio packets is calculated in order to determine the buffer size required by the client and the server. If the video packet is larger than the audio packet, then the buffer size is equal to the video packet, otherwise the buffer size is equal to the audio packet.

Chapter 3

Results

By following the approach mention before and developing the system, several video files are played to demonstrate the functionalities of the DYNAMAC algorithm and the DYNAMAC media distribution system.

3.1 Digital Rights Management



right key wrong key

Figure 12: Example of digital right management

As mention earlier, the decoding process requires a special key to determine the specific waveform represented by a d-bite. This special key is the digital rights management feature embedded in the DYNAMAC algorithm.

right key	tV9E0WSoYb56NkqpvVaYeH8nOTg
wrong key	tV9E0WSNkqpvVaYeH8nOTg4kfjh

In Figure 12 it can be seen what happens when the wrong key is used to play a DYNAMAC video. The images look fuzzy because the wrong waveforms are extracted from the chaotic oscillation matrix. Even with similar keys as long as the right key isn't used, the video is going to look scramble.

3.2 Content Distribution

Content distribution is achieved by transferring videos from the server to the client using the DYNAMAC protocol over TCP/IP. By examining the packet format of the DYNAMAC protocol, we can calculate the maximum length of a video, and the maximum size of a single frame to demonstrate the capabilities of the protocol.

With a 32 bit number representing the frame number, we have that

$$2^{32} – 1 = 4,294,967,295$$

meaning that we can transfer up to 4,294,967,295 frames in a single connection. In other words a video being transfer may have up to 4,294,967,295 frames long. To know how long this value means in time we calculated based on the following premise: 30 frames represent a second of video content

$$\begin{aligned} 4,294,967,295 \text{ frames x (1 second / 30 frames)} &= 143,165,576 \text{ seconds} \\ &= 2,386,092 \text{ minutes} \\ &= 39,768 \text{ hours} \end{aligned}$$

The protocol also has 16-bits that represents the sequence number inside the frame. Since the sequence number is based on the width of the video frame,

$$2^{16} – 1 = 65,535$$

it means that a video frame may have a width of 65,535 pixels. Currently a HDTV video frame has 1,920 pixels in length x 1,080 pixels wide.

Estimating that a movies like Lord of the Rings and Kingdom in Heaven are approximately two and one half hours long, by our calculations we can state that the DYNAMAC protocol is capable of handling any video content with HDTV quality.

Chapter 4

Future Work

The first steps have been given in order to provide a system capable of distributing high definition images anywhere in the world while having the ability to manage digital rights of the media content being shared.

But still some work needs to be done. Normal video decoding on the player side usually has a processing factor of 30 frames per second, but because of the current decoding algorithm, the processing factor of the DYNAMAC player is 10 frames per second. This processing factor produces a slow motion effect when rendering a video.

Since the DYNAMAC player was developed using the Java Media Frameworks, there is an issue when it comes to sound rendering. The video and sound rendering components describe on Chapter 2 are inherited from the Java Media Frameworks. On a Windows computer these components rely on the DirectX software that comes with the operating system. Because of the DirectSound cooperative level used by the Java Media Frameworks,[8] the sound card buffer discards the sampling rate of the media and always renders it a 22.05 kHz. This causes audio sound to have a “chipmunk” effect because the sound is sampled at 44.1 kHz and it is played at 22.05 kHz.

Currently the DYNAMAC media distribution system is develop to only support video distribution, so a new *DataSource* that supports dynamac/audio content type and a modification in the DYNAMAC protocol are needed to support audio content.

Finally the DYNAMAC media distribution system is still in an developing stage so more work is required to have a fully functional commercial system.

Chapter 5

Conclusion

By following the approach mention on this paper we have created a system capable of distributing media content on an IP network. We have demonstrated the features of the DYNAMAC algorithm when it comes to digital rights management and data compression in a practical scenario. We demonstrated the capability of the DYNAMAC protocol to handle any kind of video content wit the highest quality that currently exists. We have concluded that more work needs to be done on the compression/decompression algorithm in order to achieve faster video processing time and therefore real time motion. We foresee that the steps taken today will help create a better future were media content on demand will be available for everyone, not just for entertainment but for educational purposes like online learning and even in the enterprise world were people won't be limited to send email messages, but audio and even video messages.

List of References

[1] Mack, S., “Streaming Media Bible”, Hungry Minds, Inc., 2002

[2] Austerberry, D., “The Technology of Audio and Video Streaming, Second Edition”, Focal Press, 2005.

[3] Newton, H., Horak, R., “Newton’s Telecom Dictionary, 20th Edition”, CMP Books, 2004.

[4] Austerberry, D., “Digital Asset Management: How to Realize the Value of Video and Image Libraries”, Focal Press, 2004.

[5] Glenn, C, M., Eastman, M., and Paliwali, G., “A New Digital Image Compression Algorithm Based on Nonlinear Dynamical Systems”, International Association for the Development of Advances in Technology, 2005.

[6] http://en.wikipedia.org/wiki/Infinite_monkey_theorem, “Infinite Monkey Theorem”

[7] Glenn, C, M., Eastman, M., and Curtis, N., “Digital Rights Management and Streaming of Audio, Video, and Image Data Using a New Dynamical Systems Based Compression Algorithm”, International Association for the Development of Advances in Technology, 2005.

[8] http://msdn2.microsoft.com/en-us/library/bb147234.aspx, “DirectSound Normal Cooperative Level”