International Journal of Multidisciplinary Engineering in Current Research - IJMEC
Volume 8, Issue 8, August-2023, http://ijmec.com/, ISSN: 2456-4265

DISTRIBUTED VIDEO ENCODING FOR WIRELESS LOW-POWER SURVEILLANCE NETWORK

Dr.Bhargav Gangadhara
*Senior Technical lead/ Director, Jack Henry and Associates, USA
Email: bhargavmtechmem@gmail.com

Abstract— Transmission of real time video over a network is a very bandwidth intensive process placing large demand both on the network resource and the end terminal computing points. To improve throughput rates and reduce bandwidth requirement video encoding is resorted to. This project addresses the video encoding process for wireless low-power surveillance networks.

Real time video encoding is a CPU intensive task. A number of algorithms exist to share the encoding process over multiple computer platforms. This is called as parallel processing. In this project I have attempted to split a video into multiple chunks and encode them separately in different computers and then merge them into a compressed video retaining all the major details but with reduced bandwidth occupancy. This project demonstrates setting up a distributed video encoder in a LAN network .To date this concept is prevalent only in high end grid computing platforms.

Furthermore, and of particular interest to distributed processing, input video files can be easily broken down into work-units. These factors make the distribution of video encoding processes viable. The majority of research on distributed processing has focused on Grid technologies, however, in practicality, Grid services tend to be inflexible and offer poor support for ad-hoc interaction. Consequently it can be difficult for such technologies to efficiently exploit the growing pool of resources available around the edge of the Internet on home PCs. This project attempts to overcome these deficiencies.

Keywords: - Real time video encoding process, parallel processing, and high end grid computing platforms.

1. INTRODUCTION

The Distributed Video Encoder is a utility application tool used to encode conventional video with the assistance of a distributed network of desktop workstations.

Our main objective in this project is to implement an encoder that can make use of the processors of the computers connected in the network to do the video compression .This will increase the speed of the encoding process by about N times (if there are N computers in the network) .We will implement this encoder on a specific video format to demonstrate the feasibility of the application and to derive results that will later be used for a wider range of video formats [1].

1.1 Introduction to the Problem Area

Video encoding is a lengthy, CPU intensive task, involving the conversion of video media from one format to another. Furthermore, and of particular interest to distributed processing, input video file scan be easily broken down into work-units. These factors make the distribution of video encoding processes viable. The majority of

DOI: https://doi-ds.org/doilink/12.2023-99435617

research on distributed processing has focused on Grid technologies, however, in practicality, Grid services tend to be inflexible and offer poor support for ad-hoc interaction. Consequently it can be difficult for such technologies to efficiently exploit the growing pool of resources available around the edge of the Internet on home PCs. It is these resources that the DVE uses to provide a distributed video encoding service [2].

1.2 Overview of Video Encoding

A Video File contains both useful and redundant information about the video that it contains. Video files are quite large in size compared to other kinds of files like text files or music files. Thus it would be advantageous to compress it for the purpose of efficient storage or for transmitting it faster over the network. Video Encoding works by removing the redundant part of the information from the video file. If a very high degree of compression is required then the quality of the video file can be tweaked to achieve higher compression.

A compression encoder works by identifying the useful part of a signal which is called the entropy and sending this to the decoder. The remainder of the signal is called the redundancy because it can be worked out at the decoder from what is sent. Video compression relies on two basic assumptions. The first is that human sensitivity to noise in the picture is highly dependent on the frequency of the noise [3].

The second is that even in moving pictures there is a great deal of commonality between one picture and the next. Data can be conserved both by raising the noise level where it is less visible and by sending only the difference between one picture and the next.

In a typical picture, large objects result in low spatial frequencies whereas small objects result in high spatial frequencies. Human vision detects noise at low spatial frequencies much more readily than at high frequencies. The phenomenon of large-area flicker is an example of this. Spatial frequency analysis also reveals that in many areas of the picture, only a few frequencies dominate and the remainder is largely absent.

For example if the picture contains a large, plain object, high frequencies will only be present at the edges. In the body of a plain object, high spatial frequencies are absent and need not be transmitted at all.

1.3 Overview of Distributed Computing

Over the past two decades, advancements in microelectronic technology have resulted in the availability of fast, inexpensive processors, and the advancements in communication technology have resulted in the availability of cost-effective and highly efficient computer networks. The net result of the advancements in these two technologies is that the price-performance ratio has now changed to favor the use of inter-connected, multiple processors in place of a single high-speed processor

Computer architectures consists of two types

1.3.1. Tightly Coupled Systems

In these systems, there is a single system wide primary memory (address space) that is shared by all the processors. Any communication between the processors usually takes place through the shared memory.

1.3.2 Loosely Coupled Systems

DOI: https://doi-ds.org/doilink/12.2023-99435617

In these systems, the processors do not share memory, and each processor has its own local memory. All physical communication between the processor is done by passing messages across the network that interconnects the processors.

Tightly coupled systems are referred to as parallel processing systems and loosely coupled systems are referred to as Distributed Computing Systems.

1.4 Problem Statement

To create a parallel video encoder which utilizes the processing power of the computers in a Local Area Network to complete the Encoding Process in a lesser duration of time when compared to a single encoder run on a single computer .

2. REQUIREMENT ANALYSIS

The basic purpose of software requirement specification (SRS) is to bridge the communication between the parties involved in the development project. SRS is the medium through which the users' needs are accurately specified; indeed SRS forms the basis of software development. Another important purpose of developing an SRS is helping the users understand their own needs.

2.1. System Requirements Specification

2.1.1 Introduction

2.1.1.1 Purpose

This Software Requirements Specification provides a complete description of all the functions and constraints of the Distributed Video Encoder, Version 1, developed for commercial and industrial use.

2.1.1.2 Intended Audience

The document is intended for the developers and testers working on the project and other independent developers who are interested in plug-in development for DVE.

2.1.1.3 Project Scope

The DVE is used to encode uncompressed video data into a compressed format in the shortest time possible by harnessing available distributed computing power in a network. It facilitates a hassle free environment to encode large volumes of video data quickly and efficiently. Similar software technology exits for large server farms that contain hardware that is dedicated specifically to video processing which prove to be expensive for individuals and small organizations that prefer to use only a part of such technology on a short time basis. Clearly they would opt for a cost effective technology that can be used with commercially available hardware and minimal technical know how on video compression.

4.1.2 Overall Description

4.1.2.1 Product Perspective

The DVE is a technology bridge between standalone encoders on workstations and massively distributed parallel hardware encoders. It combines the requirements of the former and the computing power of the latter with minimal compromise on productivity.

ISSN: 2456-4265

DOI: https://doi-ds.org/doilink/12.2023-99435617

4.1.2.2 Product Features

The main features of the DVE are

- It can encode video files in parallel and hence produce results in a shorter duration of time that a standalone encoder is able to provide.
- It will consist of a GUI based interface for easy user interaction with the application. The GUI will be a Windows Application Form based that can be run directly on the Windows Platform.
- It will provide automated network connectivity which will display connections to systems that are ready to contribute resources to the encoding process.
- It will feature a software interface for external programs to run and perform the tasks asynchronously and provide means to retrieve results from the above processes.

4.1.2.3 Operating Environment

The project will be designed for the Intel 32 bit architecture range of workstations that requires Microsoft Windows NT based operating system coupled with the Microsoft .NET runtime environment.

2.1.2.4 Assumptions and Dependencies

The project requires having a fully working and scalable user network of workstations devoid of any issues regarding connectivity and usability. Tools regarding support for multimedia in the windows platform are present in the systems being used by the product. Up-to-date versions of .NET 2.0 runtime environment or higher must also be present. The project will utilize the features of the FFMPEG encoder and the MPGTX splitter that will provided along with the above mentioned.

The network on which the application will run is a LAN network which is completely connected over an Ethernet switch.

Limited resources will enable us to develop a distributed encoder for only a specific type of video file format known as MPEG -1. The encoder will convert the mentioned file format to that of the MPEG-4 standard format only.

2.2 System Features

System Feature 1: Network Information

Description: Contains the information regarding the network id of a participating workstation connected to the distributed network. It is used to map network address to a user given host name for a workstation. This feature is critical in nature and is required for the establishment of the network.

Functional Requirements: IP Addressing scheme, DNS naming scheme. If these schemes are not present or have not been enabled or are inaccessible to the user then this feature should intimate the user as an error and suggest necessary action to be performed. Since this is a critical part in establishing the network, the application must halt and not proceed any further.

DOI: https://doi-ds.org/doilink/12.2023-99435617

System Feature 2: Execute Command

Description: Will execute an external program that is required by the application for completing tasks such as splitting, joining and encoding of video. This feature is critical in nature and is necessary for a software interface to third party programs.

Functional Requirements: process name, process arguments. Error or failure of execution of the command must be checked and the cause of the issue must be identified and informed to the user.

System Feature 3: Send File

Description: A feature that utilizes the TCP protocol to send a file stored on a workstation across to any remote workstation accessible within the network directly.

Functional Requirements: Destination workstation, File to be transferred.

System Feature 4: Receive File

Description: A feature that utilizes the TCP protocol to receive an incoming file from a remote workstation in the network directly and store it on the specified location in the workstation.

Functional Requirements: Source workstation, File store location.

System Feature 5: File Information

Description: To store all the information regarding the details of a file

Functional Requirements: File Name, Directory Stored, Number of parts (if it is split)

System Feature 6: Split File

Description: To generate arguments to an external program to split pieces of files according to the given constraints.

Functional Requirements: File Information, Number of Workers connected mpgtx.

System Feature 7: Join File

Description: To generate arguments for an external program to combine multiple pieces of a single file which has been split and processed previously.

Functional Requirements: File Information, All parts of a single file entity, ffmpeg.

System Feature 8: Encode File

Description: To generate parameter arguments necessary for an external program to encode a given video file successfully.

Functional Requirements: Input file, Output file, input video type, output video type

System Feature 9: Helper Program Information

DOI: https://doi-ds.org/doilink/12.2023-99435617

Description: This utility feature can be used to set and access the stored location of external programs necessary for the working of the application.

Functional Requirements: Directory access to external programs.

2.3 External Interface Requirements

2.3.1User Interface:

2.3.1.1 Worker UI

Description: It is used to provide a user interface to a workstation. It contains all the details regarding on connecting to an existing server in the network. It must be implemented as a Windows Application Form

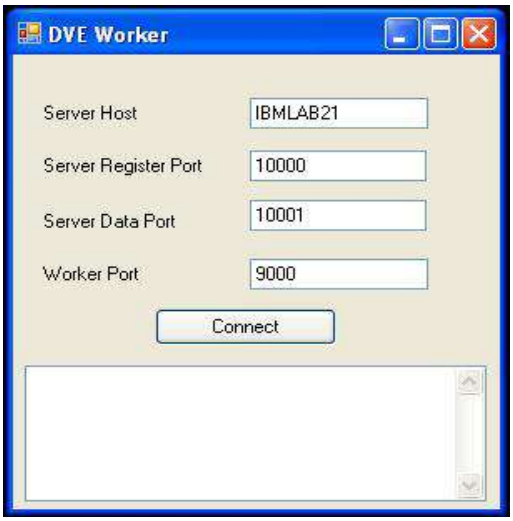

Fig 1 Worker User Interface

2.3.1.2 Server UI

Description: It is used to provide a user interface to a server. This will contain all the details regarding on connection initialization, Video file encoding and preview and also to manage external software (mentioned in the software interface) that will be required for the application.

DOI: https://doi-ds.org/doilink/12.2023-99435617

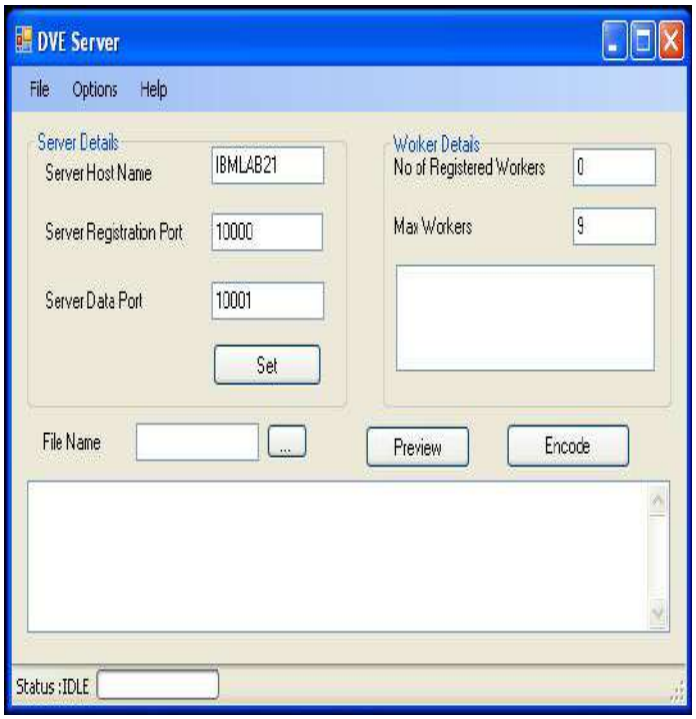

Fig 2 Server User Interface

2.3.1.3 Interface for selecting the video file

Description: A file dialog has been provided for selecting the video file for encoding. This dialog enables the user to choose a file in a manner convenient to the user as this dialog is a standard for all windows systems

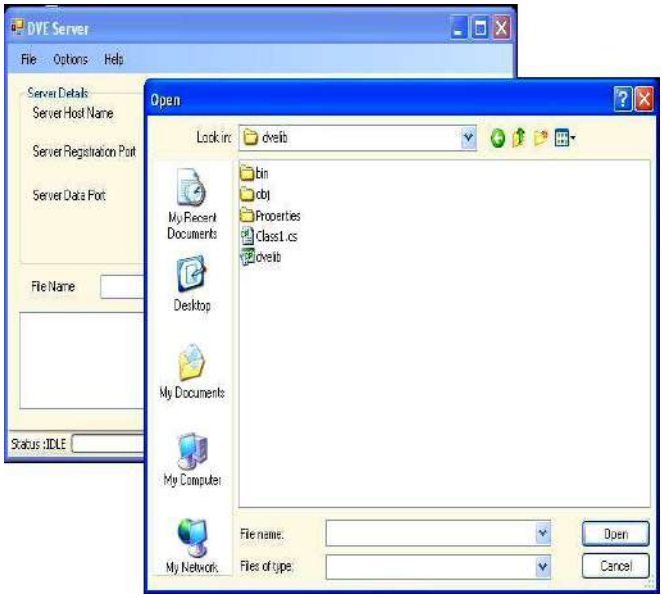

Fig 3 Video File Selection Dialog

DOI: https://doi-ds.org/doilink/12.2023-99435617

2.3.1.4 Interface for selecting the location of the utility programs

Description: A form with open dialog boxes have been provided to locate the utility programs. A filter has also been provided so that only executable files are selected.

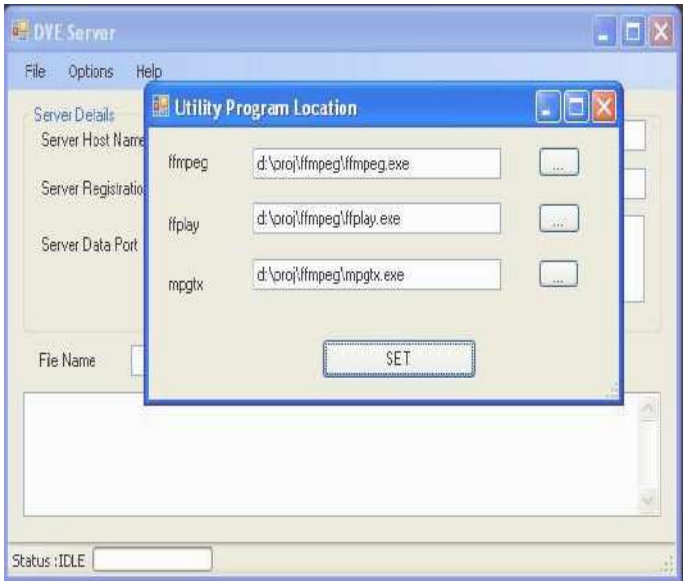

Fig 4 Utility Program Location

2.3.1.5 Hardware Interface: None

2.3.1.6 Software Interface: MPGTX

Description: It is command line MPEG audio/video/system file toolbox that slices and joins audio and video files, including MPEG1, MPEG2 and MP3. It is used as a single independent executable that can be run as a process. It is an Open Source application released under the GPL. Its current version is 2.0.

2.3.1.7 FFMPEG

Description: FFmpeg is a complete solution to record, convert and stream audio and video. It includes libavcodec, the leading audio/video codec library. FFmpeg is developed under Linux, but it can compile under most operating systems, including Windows. Is is used as a single executable that uses the POSIX threads for Windows library. It is an Open Source application released under the GPL. Its current version is 0.4.9

FFPlay

Description: FFplay is a player that can be used o play MPEG videos files in the absence of Window Media Codec's. This application is an extra utility for the project and is not necessary for the functioning of the application. This player is bundled along with the FFMPEG encoder.

2.3.1.8 Communication Interface

TCP

ISSN: 2456-4265

DOI: https://doi-ds.org/doilink/12.2023-99435617

The network that the application will run on is the IEEE 802.3 LAN Technology. All systems in the network will be connected over a single switch. The communication will occur as transmission of raw bytes over the TCP Send/Receive protocols. It is a connection oriented protocol and can guarantee the transmission of data from source to destination.

2.4 Hardware Requirements

- Pentium 4 2.0Ghz or above processor nodes
- 10 / 100 mbps Network
- Network card in the nodes
- Graphics Processors in nodes(Separate or Integrated)
- Sound Card

2.5 Software Requirements

- Windows XP with SP2
- Microsoft .NET Framework 2.0
- Ffmpeg encoder
- Mpgtx splitter
- Network Drivers and Support in Win XP
- Microsoft Visual Studio .NET 2005

3. IMPLEMENTATION

The purpose of this chapter is to give a brief description about the various implementation tools like the programming language, the modules created and the details of each module implementation. The goal of the implementation phase is to translate the design of the system produced during the design phase into code in a given programming language. This involves the details of the methods and interfaces used and their integration.

3.1 Modules

Module Name: Execute Command

This module is used for running an external program or a command. Our project consists of three helper programs which are used for the purpose of splitting the files, encoding the video files and for playing or previewing the file. This module is used for executing these programs. The module takes in the process name and arguments as input. Options for displaying the executing process have also been provided for monitoring its progress [4].

Algorithm 1: Execute Command

Input: Command, Arguments

Step1: Check if arguments present

Step 2: If present GOTO Step 2.1 else GOTO Step 3

Step 2.1: Start new process with command and arguments

DOI: https://doi-ds.org/doilink/12.2023-99435617

Step 2.2: Execute Process and wait till end of process

Step 2.3: Complete execution and GOTO Step 3

Step 3: End

Module Name: FileEncoder

This module acts as a helper module for the Execute Command module. This modules function is to generate arguments for the file encoder (ffmpeg) and then pass this argument to the Execute Command process for execution [5].

Algorithm 1: FileEncoder

Input: Filename

Step1: Copy Filename to InputFIleName

Step 2: End

Algorithm2: Generate Argument

Step 1: Concatenate the filename; format required (".avi") and the output file name

Step 2: Return the above file as an argument.

Step 3: End.

Module Name: FileJoiner

This module also acts as a helper module for the Execute Command module. This modules function is to generate arguments for the copy command which is used for concatenating all the split pieces. The second function of this module is to generate the transcoding arguments for ffmpeg for generating the output file.

Algorithm1: Generate Argument

Input: NoofFiles, Filename

Step 1: Generate the filename; format required (".avi")

Step2: Do Step 1 for every piece of the file. If all pieces are done, GOTO Step 3

Step3: Return the above generated text as an argument.

Step 4: End.

Module Name: FileSplitter

This module is used for generating arguments for the file splitter (mpgtx). It takes the number of pieces as input. This data is obtained from the number of workers connected to the server.

Algorithm1: Generate Argument

Input: Filename

Step 1: Generate the filenames for each peice; format required (".avi")

Step2: Do Step 1 for the whole file. If EOF is reached, GOTO Step 3

Step3: Return the above generated text as an argument and the number of pieces.

Step 4: End.

Module Name: NetworkInfo

This module is responsible is responsible for providing network related information like IPAddress and the Host Name of the computer.

DOI: https://doi-ds.org/doilink/12.2023-99435617

Algorithm 1: Get Network Information

Input: None

Step1: Get IP Address of the system and store as List

Step2: Get the Host name of the system as store as text

Step 3: Done

Module Name: NodeList

This module is used to manage the worker nodes. Functions for adding, deleting nodes are provided. Each node stores information about a specific worker. This information is then used by the server to communicate with the workers.

Algorithm 1: Add Node

Input: Node

Step 1: Add node to an array of Nodes

Step 2: Done

Algorithm2: Delete Node

Input: Node

Step 1: Search for the given node in the array of Nodes and delete it.

Step 2: Done.

Algorithm 3: Count

Input: None

Step 1: Count the number of nodes in the array

Step 2: Return the count as an integer

Step 3: Done.

Algorithm 4: Pop Node

Input: None

Step 1: Remove the topmost node in the array of Nodes

Step 2: Return the node from the array and delete the node

Step 3: Done.

Module Name: ReceiveFile

This module is used for receiving a file which has been sent from a remote node. A connection is first established with the remote node. Then Information about the file is received, this is used to create a new file. The file contents are then received.

Algorithm 1: Receive File

Input: port, noofFiles

Step 1: Start a TCP Client to receive files. Initialize count to 0

Step 2: If count < noofFiles, then continue else GOTO Step 3

Step 2.1: Accept the incoming Node information as fname

Step 2.2: Accept the incoming data with a buffer of 2048 bytes.

Step 2.3: Write the data to a file as mentioned in the fname object

DOI: https://doi-ds.org/doilink/12.2023-99435617

Step 2.4: Increment count and GOTO Step 2.

Step 3: Close the TCP client.

Step 4: Done.

Module Name: SendFile

This module is used for sending a file to a remote computer. This module is used for sending file chunks to the worker and for sending the compressed files back to the server.

Algorithm 1: Send

Input: FileName, Node

Step 1: Get the file name and the node information

Step 2: Start a TCP Client and connect to the system that has to receive the file

Step 3: Transmit the Node directly over the network stream

Step 4: Do the steps till the EOF is reached else GOTO Step 5

Step 4.1: Copy contents of Filename to a 2048 byte size buffer

Step 4.2: Transmit buffer data as a byte stream to the system over a socket.

Step 4.3: Confirm transfer and GOTO Step 4.

Step 5: Close the File and Disconnect from the system

Step 6: Done.

4. STRUCTURAL TESTING

Structural testing - This testing is based on knowledge of the internal logic of an application's code. Also known as Glass box Testing. Internal software and code working should be known for this type of testing. Tests are based on coverage of code statements, branches, paths, conditions [6].

4.1 Test Cases

Test Case 1

Module Name	Network
Function	Set Port Number
Input	Input entered as a number greater than 65000
Expected Results	Do not accept the number as a valid port number
Actual Results	An error message is displayed and the port field is reset to the default value
Remarks	Only valid port numbers are accepted

Test Case 2

Module Name	Network
Function	Set Port Number

DOI: https://doi-ds.org/doilink/12.2023-99435617

Input	Input entered consisted of a string of characters
Expected Results	Do not accept the string as a port number
Actual Results	Show an error message and reset the field
Remarks	Only valid port numbers are accepted

Test Case 3

Module Name	Network
Function	Connect to Server
Input	Provide Incorrect server port and then attempt to connect.
Expected Results	Show a user friendly message about the error
Actual Results	A Message is displayed that the server is not listening at that specified port.
Remarks	Only those ports that the server is listening at can be used for connecting.

Test Case 4

Module Name	Video
Function	Select Video File
Input	Choose a file that is not a MPEG 1/2 video file
Expected Results	Inform the user that a valid file has not been selected and provide an option to choose the video file again.
Actual Results	An error message is displayed after which the open dialog prompts to select a new file
Remarks	Only MPEG files can be selected as the input file

Test Case 5

Module Name	Video
Function	Preview
Input	Choose a file that is not a MPEG 1/2 video file
Expected Results	Inform the user that a valid file has not been selected and hence do not execute FFPLAY
Actual Results	The file is not accepted and an error message is displayed
Remarks	Only valid video files can be previewed

Test Case 6

Module Name	Video
Function	Encode

ISSN: 2456-4265

DOI: https://doi-ds.org/doilink/12.2023-99435617

Input	Encode when only 1 worker is present
Expected Results	The splitting and joining phases are skipped. The video file is encoded directly
Actual Results	The file is encoded in the single system
Remarks	Encoding works even if there is only one worker

Test Case 7

Module Name	Video
Function	Encode
Input	Encode when no workers are present
Expected Results	The Encoding process will not occur
Actual Results	The system shows a message informing the user that at least 1 worker must be present for the encoding to take place.
Remarks	A minimum of 1 worker is needed for execution

Test Case 8

Module Name	DVE
Function	Encode
Input	MPEG 1 video file
Expected Results	Compressed MPEG 4 Video file
Actual Results	Compressed MPEG 4 Video file
Remarks	The encoder can compress MPEG 1 files

Test Case 9

Module Name	DVE
Function	Encode
Input	MPEG 2 video file
Expected Results	Compressed MPEG 4 Video file
Actual Results	Compressed MPEG 4 Video file
Remarks	The encoder can compress MPEG 2 files

Test Case 10

Module Name	DVE
Function	Encode

DOI: https://doi-ds.org/doilink/12.2023-99435617

Input	AVI compressed File
Expected Results	Error Message
Actual Results	DVE does not accept the video file
Remarks	Only MPEG files are currently supported

5. RESULTS

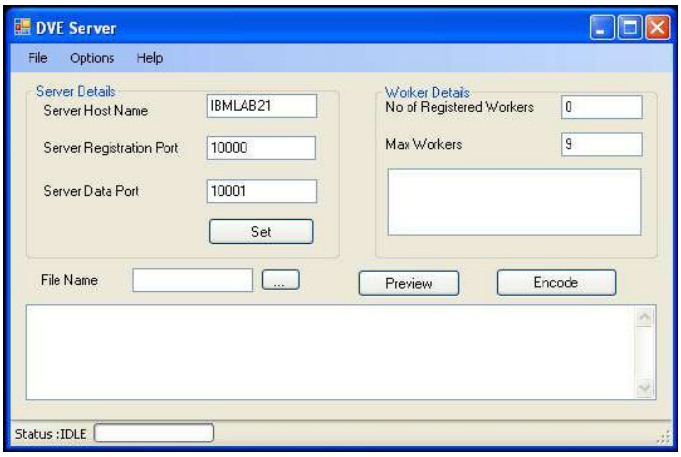

Fig 5 Server Main Screen

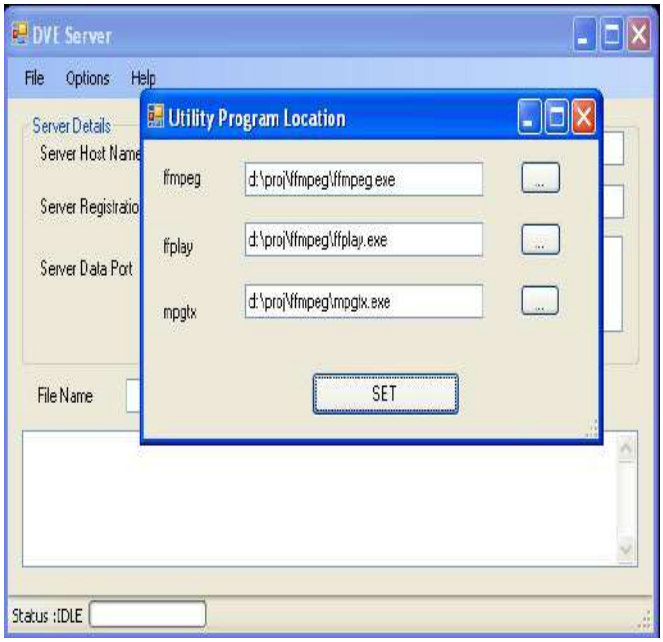

Fig 6 Server Utility Program Selection

DOI: https://doi-ds.org/doilink/12.2023-99435617

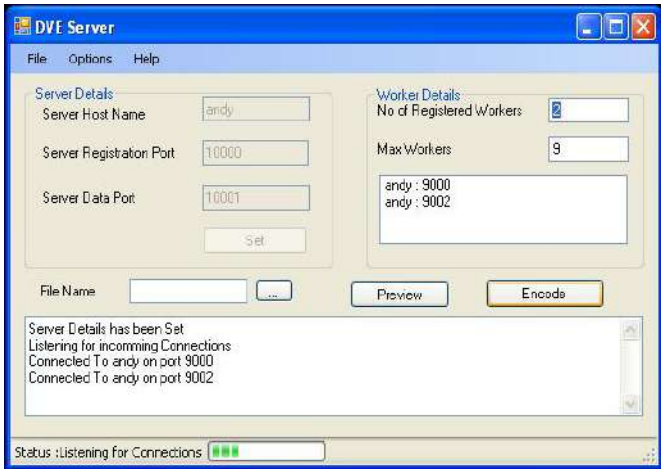

Fig 7 Server Accepting 2 workers on ports 9000 and 9002

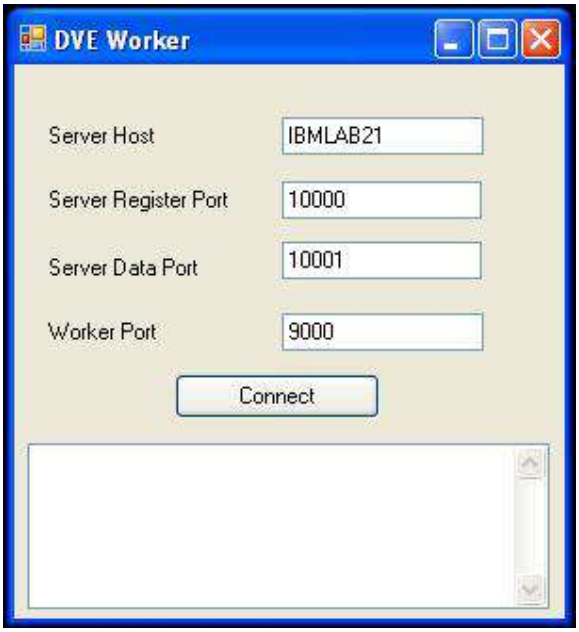

Fig 8 setting the server register, data ports and the worker port.

DOI: https://doi-ds.org/doilink/12.2023-99435617

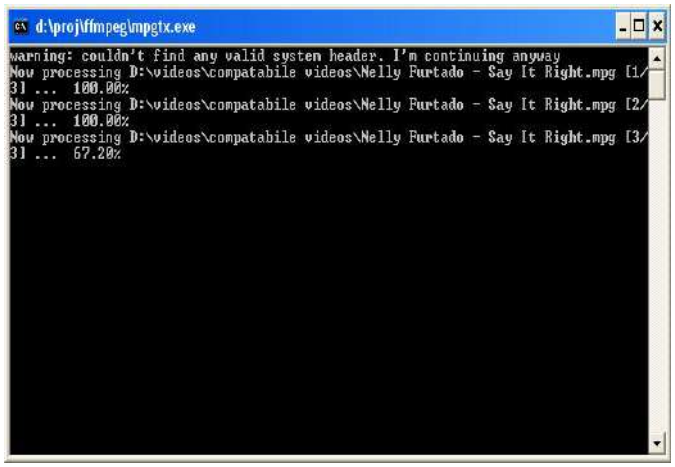

Fig 9 MPGTX splitting the video file into 3 pieces

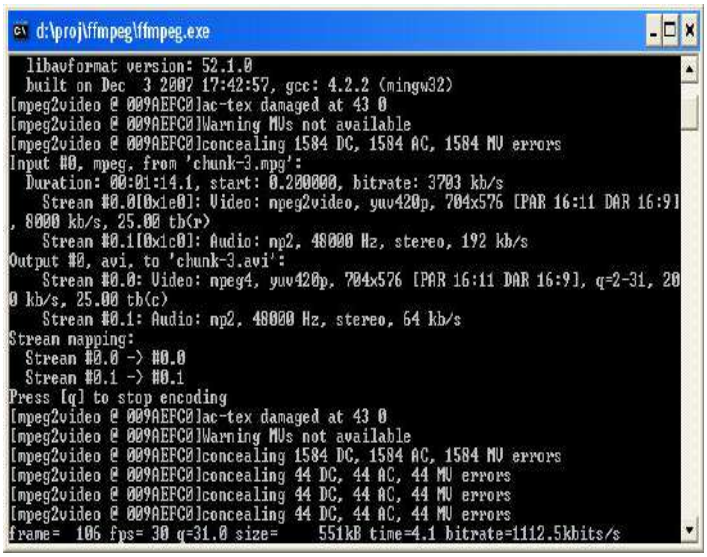

Fig 10 FFMPEG : Encoding the video chunk

DOI: https://doi-ds.org/doilink/12.2023-99435617

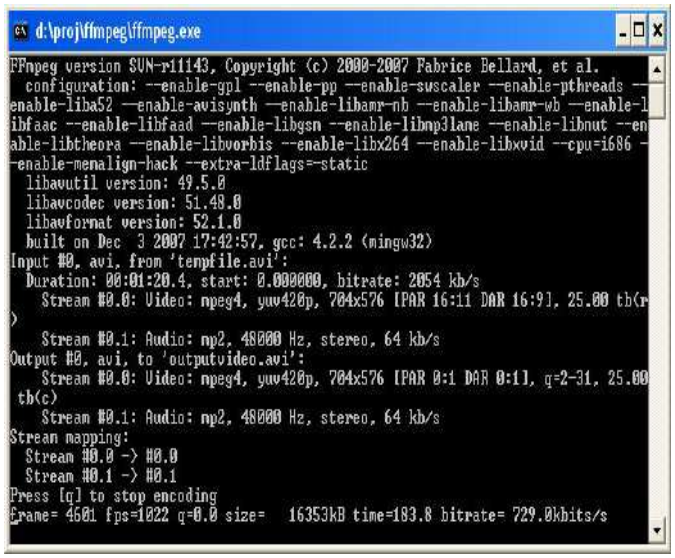

Fig 11 FFMPEG : Used for Transco ding the Joined Chunk

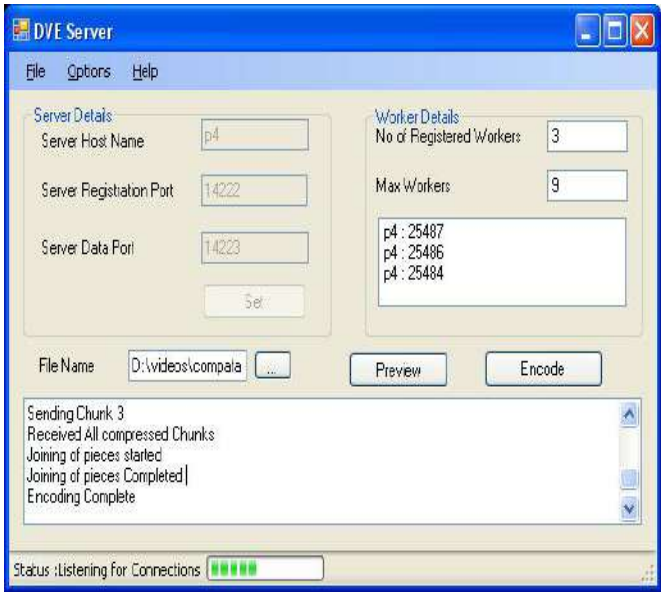

Fig 12 Server Info Box Showing Status after the conversion is done

DOI: https://doi-ds.org/doilink/12.2023-99435617

Fig 13 Worker after finishing encoding 3 pieces of video

Fig 14 FFPLAY - previewing video files before encoding

6. FUTURE ENHANCEMENTS

- The DVE is capable of handling further additional features that can improve the quality of encoding and also provide a wider spectrum of options for the application. The following enhancements are listed below as follows [7].
- The application can be made compatible for Real Media, QuickTime Video and H.264 video codecs which also can support parallelization.
- File formats such as Matroska (.mkv), Real format (.rm), Vobris (.ogg) can be used as a more efficient alternative for AVI file structures.

DOI: https://doi-ds.org/doilink/12.2023-99435617

- Features such as HTTP, FTP and various application level protocols can be used for control and data transfer of video.
- Custom made plug-ins for splitting video files and joining them can be written for various file formats that can be supported as mentioned above.
- The application can be mapped and re-implemented on to a Linux operating system and thus can be made to run in the UNIX environment giving more flexibility and freedom for development [8].
- System and Network Diagnostics can be implemented to automate the workload distribution across the entire system and can provide tracking results and work progress back to the operator.
- Decentralization of the entire model can be implemented on larger scaled applications with higher grade network management features such as session handling and metadata in feed-forward cycles across the network.
- The whole application can be scaled beyond the LAN network to the WAN and other networks to cover a larger region of users who need the services of the above application [9].
- The application can be plugged into a Cloud computing architecture that can enable a user to utilize the feature from a remote location. This also opens business prospects for large software and multimedia based companies.

7. CONCLUSION

With the above seen results one can clearly guarantee that we have accomplished all the necessary objectives we initially laid out for the Distributed Video Encoder application. Academically, we have also managed to understand the latest technology available and have successfully used it to design and develop the application.

We have also spoken about how the product can be enhanced further and this project should lay a cornerstone for other developers and designers who are interested to work and develop on the related field. I hope that this paper serves as a beacon of light to those who want to venture along the path of technological development and progress for the betterment of mankind.

8. APPENDIX

1 Glossary

AVI	Audio Video Interleaved. Microsoft format for digital audio and video playback from Windows 3.1. Somewhat cross-platform, but mostly a Windows format. Has been replaced by the ASF format but is still used by some multimedia developers.
CODEC	A software or hardware device to encode and decode video, audio or other media. The encoder and decoder for a particular compression format. The word is contracted from Coder-

DOI: https://doi-ds.org/doilink/12.2023-99435617

	Decoder. The codec refers to both ends of the process of squeezing video down and expanding it on playback. Compatible coders and decoders must be used and so they tend to be paired up when they are delivered.
DCT	Discrete Cosine Transform algorithms are used in MPEG encoders to compress the video.
DIVX	A popular standard for compressing video to a reasonably good quality at low bit rates. A 2-hour movie compresses to a size that comfortably fits on a CD-ROM. This is derived from the MPEG- 4 standard with some additional enhancements that make it noncompliant in some minor respects.
JPEG	Joint Photographic Experts Group.
LAN	Local Area Network. Usually Ethernet running at 10 Megabits or 100 Megabits. Sometimes 1000 Mbits are deployed and referred to as Gigabit Ethernet.
Macroblock	Video pictures are degenerated to blocks of 16 × 16 or 8 × 8 pixels so that similarities can be found.
MPEG	Motion Picture Experts Group. A working group within the ISO/IEC standards body. This group is concerned with the standardization of moving-picture representations in the digital environment.
MPEG 1	The original MPEG video standard designed for CD-ROM applications and other similar levels of compression. This is a legacy format. Still useful for some solutions but new projects should use a later codec.
MPEG 2	Widely used for digital TV and DVD products. A much improved codec that supersedes MPEG-1 in many respects. It has been popularized by DVD content encoding and DVB transmission.
MPEG 4	A huge leap forward in the description of multimedia content.MPEG-4 is the essence container.
QuickTime	A platform for developing and playing rich multimedia, video, and interactive content. It is not a codec but a platform. It has some unique codecs that are not available anywhere else but you should be using open-standard codecs rather than these. When it is referring to a video format, it is talking about a codec jointly developed by Apple and Sorenson. It gets improved from time to time as successive versions of QuickTime are released.

ISSN: 2456-4265

DOI: https://doi-ds.org/doilink/12.2023-99435617

	This is a popular name for the Sorenson codecs.
TCP / IP	Transmission Control Protocol/Internetworking Protocol describes the message format and connection mechanisms that make the Internet possible. It is built on top of UDP. It contains ack/nak (acknowledge/negative acknowledge) protocols to ensure that all packets have arrived It does this at the expense of timeliness.

REFERENCES

[1] Dr. Bhargav Gangadhara, "Optimizing Cloud - Based Manufacturing: A Study on Service and Development Models", International Journal of Science and Research (IJSR), Volume 12 Issue 6, June 2023, pp. 2487-2491, https://www.ijsr.net/getabstract.php?paperid=SR23626155823

[2] Dr.Bhargav Gangadhara , "Optimize Utility in Computing-Based Manufacturing Systems Using Service Models and Development Models", Presented and published at International Journal of Computer Science and Information Security, IJCSIS Editorial Board, Vol -14, Page 5.

[3] Dr.Bhargav Gangadhara,"Optimize utility in cloud-Based manufacturing systems using service models and development models", Presented and published at International Journal of Innovative Research in Advanced Engineering.IJIRAE Editorial Board, August 11 - 12, 2016.

[4] A Practical Guide to Audio and Video Compression – Cliff Wooton

[5] Video Demystified - Keith Jack

[6] MPEG encoding basics – Snell and Wilcox

[7] C# and the .net Framework – Andrew Tolstein

[8] Microsoft Developer Network (MSDN)

[9] Software Engineering – Ian Sommerville

[10] Object Oriented Analysis and Design – Ali Bahrami

[11] http://www.setiathome.com

[12] http://folding.stanford.edu

DOI: https://doi-ds.org/doilink/12.2023-99435617