Desktop video recording and streaming

COMP4801 Final Year Project

Hu Yucheng (Patrick) 3035029299

Abstract

This project is going to develop low CPU consumption desktop video recording and streaming applications. With these two applications, manager can record and see what their employees are doing during their work time. Similar techniques have been broadly used in game broadcasting websites like, Panda TV, Twitch.
# TABLE OF CONTENTS

1 BACKGROUND .................................................................................................................. 4
   1.1 Online live steaming ................................................................................................. 4
   1.2 Monitoring software ............................................................................................... 5
2 OBJECTIVES ..................................................................................................................... 6
3 PROBLEM STATEMENT .................................................................................................... 7
   3.1 High time consumption in developing ................................................................... 7
   3.2 Use of open source library ..................................................................................... 8
   3.3 High requirement of efficiency ............................................................................. 8
4 DELIVERABLES ............................................................................................................... 8
5 SCOPE ............................................................................................................................. 9
6 METHODOLOGIES ......................................................................................................... 10
   6.1 Register Sequence ................................................................................................. 10
   6.2 Working Sequence ............................................................................................... 11
   6.3 Termination Process ............................................................................................ 11
   6.4 Plug-in Architecture ............................................................................................. 12
   6.5 Application Frame ............................................................................................... 14
   6.6 Open Source Solution FFmpeg ............................................................................ 15
   6.7 Screen Capture Technique .................................................................................. 16
   6.8 Compression Technique ....................................................................................... 16
7 CHALLENGE .................................................................................................................. 18
   7.1 Find a suitable method to achieve the objective .................................................. 18
   7.2 Understand the architecture chosen for this project .......................................... 18
   7.3 Use of the open source library ............................................................................. 18
7.4 Implement the playback technique .......................................................... 19

8 TASK AND PROJECT SCHEDULE .......................................................... 19

LIST OF FIGURES

Figure 1 A screenshot of DouyuTV ............................................................... 5
Figure 2 The mapping of monitoring software market ................................... 6
Figure 3 Monitor blueprint ........................................................................... 9
Figure 4 Register sequence diagram ............................................................ 10
Figure 5 Working sequence diagram ............................................................ 11
Figure 6 Termination sequence diagram ....................................................... 12
Figure 7 Plug-in architecture ....................................................................... 13
Figure 8 Compression algorithm contrast graph ......................................... 17
1 BACKGROUND

This technique can be applied in situations. Analyzing both cases can help find the market entry point for the project.

1.1 ONLINE LIVE STREAMING

The first one is the online live streaming. In 2014, Twitch came into people’s sight, and the broadcaster used the technique provided by Twitch to broadcast their game. This new communication has not only attracted millions of young people to the website but also created tons of opportunities to both the players and the watchers. Meanwhile, China’s live-streaming websites, such as DouyuTV and ZhanqiTV, also started their business in China by initially focusing on the game live streaming. At present, they have already become a burgeoning cottage industry, which offers money-making opportunities and even stardom to their mostly female hosts. Also, this is a new alternative for millions of viewers to online dramas. The technique used by these websites is the same as what we implement in this project. The difference is that we have no need to worry about scaling at architectural and organizational level.

What is the relation between this project and online live streaming? Similar techniques have been used. A screenshot of DouyuTV (see figure 1) can give us some intuitive feelings. The host’s computer is a recorder, while the audience’s computer is like a monitor. The movement on the host’s computer can be recorded and transmitted to the website. Then, the audiences can see what the host is doing on his desktop via the website.
In this case, since the industry is currently well-developed and all of these websites have a mature technical team to maintain the operation, there is no market entry point for the project.

1.2 Monitoring Software

The second one is the monitoring software. The companies install a desktop video that records application on every computer as well as a streaming application on the manager’s computer. Then, the managers can see what the employees are doing during their work time. Indeed, this is a quite efficient way in restricting the manners of the employees.

The existing monitoring software and their main features can be shown in the following Figure 2.
The first row lists the monitoring software in the market, and the first column lists the features. Nearly all of the software has many features. Besides, it can be seen that the price of all the software is really high. In some user cases, however, only one or two features are needed. Thus, the users are unwilling to pay so much for the needless features. Maybe, there is a need for the monitoring software that can be customized. Hence, users can choose and pay for the features they need. This refers to a great market entry point for this project. After finding this, I set the goal of my project to develop a user-friendly and highly customized monitoring software.

## 2 Objectives

The objective of this project is to develop a user-friendly monitoring software which has:

- High customization
- High scalability
- Low CPU consumption
• Low delay

**Customization:** The users can choose the features they need and download the corresponding versions. The intermediate goal of this project is to have two features, including real time recording and remote recording, in the feature pool. This will be done by next April. The ultimate goal of this project is to have more than 20 features in the feature pool. Then, the users can choose the feature they need and download the corresponding version of software.

**Scalability:** The first version of the project will be finished before the next April. Later, the new features will be added to the first one. New features can be added to the existing version easily. Also, the new features added will not influence the previous ones.

**Low CPU consumption:** The software will have low CPU consumption. Regarding the CPU consumption, no specific standard is set up now. To lower the CPU consumption, continuous testing will be carried on in February.

**Low delay:** The monitor will receive the real-time movement on the recorder’s desktop. Due to the limitation of network and hardware, there are some delays. However, it will be controlled to less than 0.25 seconds.

### 3 PROBLEM STATEMENT

#### 3.1 HIGH TIME CONSUMPTION IN DEVELOPING

Assume that there are 10 features in the feature pool, there would be more than 100 different combinations of different features. If the application is developed in a traditional way like the previous work did, it would consume large amounts of resources. If the cost to develop such a comprehensive
system is initially too high, or if the timeline to deployment is too long, the beautiful thing about
custom software will no longer be beautiful. Thus, the project needs a fast developing path.

3.2 USE OF OPEN SOURCE LIBRARY

The recorder and monitor will be implemented based on the open resource FFMpeg and its subtool
FFplay. Since this is the first time of working with open source, a good understanding of the open
resources will be needed.

3.3 HIGH REQUIREMENT OF EFFICIENCY

Indeed, this is a project that focuses more on implementing than researching. There are already some
similar applications in the market, such as airplay, chrome cast, and twitch, which do really good job.
The meaning of this project lies in the sharpening of desktop programming skills and self-learning
skills. With all the other applications developed by a big team, the biggest problem of the project is the
efficiency. The first version with real-time recording and remote recording should come out before
February. Furthermore, to optimize the functionality of the software, the tests would be taken in the last
two months. No chance for constructional mistakes is allowed. Before the implementation, a good user
case study and a clear product design will be needed.

4 DELIVERABLES

The deliverables have two applications. To be specific, one is installed on the monitor’s computer to
serve as the monitor, while the other one can be installed on several computers to serve as the recorder.
The monitor can set connection with these recorders simultaneously. It can switch among different
recorders to see the desktop activities (see Figure 3).
FIGURE 3 MONITOR BLUEPRINT

5 SCOPe

The programming language of this project is C++, which is a requirement of this project. The programming platform is Virtual Studio, and it can offer a set of helpful tools for the developing.

This project, which is a combination of existing techniques, takes the mature and popular plug-in architecture. Other fundamental functionalities are implemented with the help of open source library FFmpeg and its subtool FFplay. Also, it includes making improvement to these techniques so as to let them work better with frame.

These two applications support live broadcasting whose delay would be controlled to less than 25 milliseconds. However, the specific time interval delay broadcasting is not supported. Also, the video streaming application will support all the main video formats. Besides, the resolution ratio can be
adjusted according to the user’s preference. Most of all, this project supports one-to-many monitoring, meaning that one recorder can monitor unlimited recorders.

6 METHODOLOGIES

6.1 REGISTER SEQUENCE

The first process involved in a new connection is the register process, and it can help set up a stable connection between the monitor and a particular recorder. The process can be shown in figure 3.

![Diagram of Register Sequence](image)

**FIGURE 4 REGISTER SEQUENCE DIAGRAM**

This diagram gives the sequence of the register process. It is triggered by the recorder on clicking the starting register button. After the click of button, the initialization can be done. Then, it will send the register signal to the monitor via the TCP long connection. After receiving the register signal, the monitor will automatically assign a port number to this request and send the number back to the recorder. In the end, the register process will succeed.
6.2 Working Sequence

After the success of register, the monitor and recorder will start the working procedures. This process can be shown in Figure 4.

Following a successful register, the monitor will start monitoring. Then, the recorder will generate FFMpeg launch parameters and create a FFMpeg process. After the success of process creation, the recorder will obtain RTP information and send the record working signal to the monitor. Meanwhile, the FFMpeg process will start recording the screen and do the encoding. After receiving the recorder working signal, the monitor will generate the RTP description file and create the FFplay process, which will receive the video transferred from the recorder.

6.3 Termination Process
A termination process will be generated to terminate the monitoring. The detailed sequence of this process can be shown as follows.

![Termination Sequence Diagram](image)

**FIGURE 6 TERMINATION SEQUENCE DIAGRAM**

The monitor sends the signal of terminating monitoring to the recorder. After receiving the signal, the recorder will terminate the FFMpeg process and return the terminate status to the monitor. Then, the monitor will terminate the FFplay process and refresh the status of the recorders. Under monitoring, the corresponding record will disappear from the table of recorders.

### 6.4 Plug-in Architecture

For achieving the goal of high customization and scalability, the project will develop a plug-in architecture. From a high-level view (see figure 5), we can regard that the plug-in architecture is formed by four parts:

- Application framework
An application framework is a software library that provides the fundamental structure to support the development of applications for a specific environment. An application framework acts as the skeletal support to build an application. During the development of applications, it can lessen the general issues. Through the use of code that can be shared across different modules of an application, this is achieved. It acts like a commander. Even though it knows which step the program will take, it does not know what each step is.

Plug-in interface, also known as plug-in contract, exists in the form of a service interface. All the plug-ins of the system will implement the unified interface specification of the system framework. As a result, this can help the application facilitate an effective organization and management of the plug-ins.
Plug-in is the software component that adds features to the program. It is loaded to the application under the requirements of plug-in interface.

Public function library contains the functions used by both of the framework and the plug-ins.

In this architecture, each system function module, common user interface, and icon can be developed as plug-ins to enhance the versatility of functional modules. The modification of functional modules will not affect the normal operation of the other plug-in modules, reduce the difficulty of system maintenance, and improve the system scalability. Also, when the features are implemented in the form of plug-ins as separate components, it can be added and removed from the application without influencing the other features.

6.5 APPLICATION FRAME
The application frame chosen for this project can be developed by a coder for personal use. I asked him for the permission to use his frame. This frame is composed of four modules:

- Transport Module
- Session Module
- Service Module
- Application Module

Transport module is based on Asio library and it takes asynchronous mode. The main function is to deal with the establishment and monitoring of connections, as well as the data’s transmission and reception.
Session module is mainly responsible for the maintenance of the connection state, including heartbeat, reconnection, and message routing.

Service module is primarily responsible for the abstract decoupling of the application, as well as the encoding and decoding of the message.

Application module mainly implements a set of plug-in system specifications. Also, it is responsible for the maintenance of plug-in’s management and the application’s life cycle.

6.6 OPEN SOURCE SOLUTION FFmpeg

FFmpeg is a set of open source audio and video tools for recording, converting, and streaming multimedia content. Moreover, it supports nearly every digital format and codec known, from the old and obscure to the cutting edge. The toolset is highly portable - available on most operating systems and platforms.

The FFmpeg project distributes four major applications:

- **FFmpeg** - A command-line utility that can be utilized to process, convert, or manipulate media. This tool refers to the foundation of the other applications.

- **FFplay** - A simple media player.

- **FFserver** - A streaming media web server.

- **FFprobe** - A stream analysis tool.

FFmpeg is built with multiple self-contained libraries, which provide discreet functionalities that can be included into the other applications. These features include codec encoding and decoding, compression, image scaling, re-sampling, and format conversion.
6.7 SCREEN CAPTURE TECHNIQUE

There are a couple of methods in implementing the screen capture functionality. I choose the mostly used three methods to test for a best one:

- **GDI way:** This mechanism is based on the simple principle that the desktop is also a window. Namely, it has a window Handle (HWND) and a device context (DC). If the device context of the desktop can be captured, then those contents can be normally transferred to the application defined device context.

- **DirectX way:** Every DirectX application contains a front buffer, which holds the video memory related to the desktop contents. By accessing the front buffer, it can capture the contents of the screen at the moment.

- **Windows Media API:** Windows Media 9.0 supports screen captures using the Windows Media Encoder 9 API. It includes a codec named *Windows Media Video 9 Screen codec* that has been specially optimized to operate on the content produced through screen captures.

These three methods are all provided by the open source library FFmpeg. During the optimization period, every method will be tested. Finally, the one that gives the most stable and fast experience will be chosen.

6.8 COMPRESSION TECHNIQUE

These are the compression algorithm that will be used:

- **PAQ based context mixing algorithms**

- **FLIF**

The PAQ based context mixing algorithms use large numbers of independent context models to predict the next pixel in an image from neighboring pixels, followed by weighted averaging of the predictions, and arithmetic coding. They produce the smallest
output. However, the speed is a little bit slower. The following Figure shows the images sorted on compression ratio.

![Image of compression ratio comparison](image)

**FIGURE 8 COMPRESSION ALGORITHM SPEED GRAPH**

FLIF is a novel lossless image format, which outperforms PNG, lossless WebP, lossless BPG, lossless JPEG2000, and lossless JPEG XR concerning compression ratio. FLIF clearly beats other image compression algorithms. According to the compression experiments we have performed [older results here], FLIF files are on average:

- 14% smaller than lossless WebP
- 22% smaller than lossless BPG
- 33% smaller than brute-force crushed PNG files (using ZopfiPNG)
• 43% smaller than typical PNG files
• 46% smaller than optimized Adam7-interlaced PNG files
• 53% smaller than lossless JPEG 2000 compression
• 74% smaller than lossless JPEG XR compression.

Hence, the project will use the FLIF compression algorithm. Besides, other algorithms will be tested to obtain a clearer comparison with FLIF.

7 CHALLENGE

7.1 FIND A SUITABLE METHOD TO ACHIEVE THE OBJECTIVE
To accomplish a more challenging project, a high-level objective was set for the project. Although this final year project does not involve the scalability and customization part, a well-working structure must be firstly set up. Thus, it takes really hard work to find the suitable architecture.

7.2 UNDERSTAND THE ARCHITECTURE CHOSEN FOR THIS PROJECT
After the plug-in architecture is determined, the next job is to find a suitable frame for this project. I looked through the open source plug-in architecture. Due to the limited knowledge of plug-in architecture, it is hard to figure out the most suitable one for this project. Under such circumstance, I chose to ask the coders for help. Some useful suggestions had been offered. Also, a plug-in architecture developed specifically for similar projects was offered.

Since the text is quite limited, it is also a tough job to understand the frame developed by others. Currently, I remain at this stage and try to find a correct way for the following work.

7.3 USE OF THE OPEN SOURCE LIBRARY
Personally speaking, much time will be spent on understating the open source library FFmpeg in the upcoming work. Based on this open source library, the main techniques will be implemented. There is probably some understanding bias for my first work with open source library.

7.4 IMPLEMENT THE PLAYBACK TECHNIQUE
Another possible challenge is the implementation of the playback technique. After finishing the implementation of recorder and monitor, I will start to work on this part.

8 TASK AND PROJECT SCHEDULE

<table>
<thead>
<tr>
<th>Month</th>
<th>Deliverables of Phase 1:</th>
</tr>
</thead>
<tbody>
<tr>
<td>October 14</td>
<td>• Project Plan</td>
</tr>
<tr>
<td></td>
<td>• Project Website</td>
</tr>
<tr>
<td>November 14</td>
<td>Find the appropriate architecture</td>
</tr>
<tr>
<td></td>
<td>Learn the knowledge of plug-in architecture</td>
</tr>
<tr>
<td>December 14</td>
<td>Learn the knowledge of FFmpeg open source library</td>
</tr>
<tr>
<td></td>
<td>Design the working sequence of the application</td>
</tr>
<tr>
<td>January 14</td>
<td>Deliverable of Phase 2</td>
</tr>
<tr>
<td></td>
<td>• Demo application</td>
</tr>
<tr>
<td></td>
<td>• Interim report</td>
</tr>
<tr>
<td>December - February</td>
<td>Development of the plug-ins:</td>
</tr>
<tr>
<td></td>
<td>• Monitor</td>
</tr>
<tr>
<td></td>
<td>Recorder</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------</td>
</tr>
<tr>
<td><strong>February - April</strong></td>
<td>Optimization and Testing:</td>
</tr>
<tr>
<td></td>
<td>• Lower CPU consumption</td>
</tr>
<tr>
<td></td>
<td>• Lower delay</td>
</tr>
<tr>
<td><strong>April 21</strong></td>
<td>Deliverables of Phase 3</td>
</tr>
<tr>
<td></td>
<td>• Finalized implementation</td>
</tr>
<tr>
<td></td>
<td>• Final report</td>
</tr>
</tbody>
</table>