ADAPTATION OF APPLETS AND LIVE VIDEOS TO MOBILE DEVICES

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree of
Master of Technology

Juturu Manoj Kumar
under the guidance of
Prof. Sridhar Iyer

Department of Computer Science and Engineering,
Indian Institute of Technology, Bombay

Dissertation Approval Certificate

Department of Computer Science and Engineering

Indian Institute of Technology, Bombay

The dissertation entitled **"Adaptation of Applets and Live videos to Mobile Devices"**, submitted by **Juturu Manoj Kumar** (Roll No: **08305052**) is approved for the degree of **Master of Technology** in **Computer Science and Engineering** from **Indian Institute of Technology, Bombay**.

Prof. Sridhar Iyer
CSE, IIT Bombay
Supervisor

Prof. Purushottam Kulkarni	**Dr. Vijay Raisinghani**
CSE, IIT Bombay	**HoD NMIMS College, Mumbai**
Internal Examiner	**External Examiner**

Prof. Abhay Karandikar
EE, IIT Bombay
Chairperson

Place: IIT Bombay, Mumbai
Date: 28th **June, 2010**

Declaration

"I declare that this written submission represents my ideas in my own words and where others ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action as per the rules of regulations of the Institute."

Manoj Kumar Juturu
IIT Bombay
June 29, 2010

Acknowledgements

I would like to thank my guide, Prof. Sridhar Iyer for his continuos support and guidance. All this work and its successful completion would not have been possible without him. His advice on possible directions in my work, and his guidance regarding critical decisions has helped me in weighing the options wisely.

I would also like to thank Prof. Purushottam Kulkarni for his anaylysis of the various aspects of the project and his suggestions during group meetings, gave me an idea of what more could be done in my work.

J Manoj Kumar
Computer Science and Engineering
08305052

Abstract

Many educational organizations provide E-learning content which is in the form of animations and lecture videos. Showing these content on mobile devices has many problems. Mobile devices work on J2ME technology which is incompatible for showing these animations in the form of Applets, where as Streaming of Live videos on mobile devices is difficult because mobiles devices have limited memory resources. Also Live videos require large network bandwidth, so they cannot be shown over mobile device connections like GPRS which has low network bandwidth.

The aim of this thesis is to adapt animations and live videos to mobile devices. Different approaches for porting Java Applets on to Mobile devices like J2ME MIDlets and flash files have been explored with their relative merits and demerits. It also presents the differences in terms of feasibility, complexity in converting J2SE Applets in to J2ME and Flash.

Similarly different possible ways to adapt live videos have been explored. An implementation of Live Video Streaming of CDEEP lecture video's in a cost effective and adaptive way for mobile devices has been provided, which is integrated into the system proposed by the supporting thesis named study element based adaptation.

Contents

List of Figures

Chapter 1

Introduction

1.1 OSCAR Applet Repository

Project OSCAR (Open Source Course ware Animations Repository) aims to create a repository of web-based, interactive animations for teaching various concepts and technologies. This could be useful not only for classroom teaching but also for enabling independent-learning and distance education. It contains repository of animations on concepts at Undergraduate and Postgraduate levels.
Each animation is typically a Java Applet that focuses on one concept and provides the following through a platform-independent, web-interface:

- A brief description of the concept, including relevant references.
- An in-built animation, to explain the concept in detail.
- An interactive animation, wherein the user defines the parameters.

1.2 Adaptation of Applets to Mobile devices

Applets are animations shown in web browsers, created using J2SE technology, but animations in mobile devices are created using J2ME. Both are Java based only but they have different libraries. So applets can not be viewed directly on mobile devices. Two approaches have been proposed

Conversion To J2ME – This approach describes about automation of J2SE applets int to J2ME MIDlets. The difficulties in automation of conversion process are summarized below.

1. **Architectural Differences in J2SE and J2ME** — J2SE has a Hierarchical architecture, while J2ME has a Flat-file architecture. Conversion of Hierarchical structure to flat-file structure is a cumbersome process.

2. **Text Adaptation** — The amount of text in OSCAR animation is very high, which cannot be displayed on mobile devices. Re-planning of Text Adaptation cannot be automated.

3. **Canvas Size and Image co-ordinates** – The co-ordinates of images in applet designed for desktops. Replanning of image co-ordinates cannot be automated.

The possibilities in automation process are summarized below.

1. **Copy of paint method** – Both J2SE and J2ME use the same graphics class. So the paint method in J2SE can be copied into J2ME.

2. **Buttons to Key mapping** – J2ME uses mobile device keys to accept input instead of buttons. So buttons in J2SE can be mapped to mobile device keys.

3. **LWUIT** – LWUIT solves the hierarchical problem in J2ME and allows use of swings in J2ME.

Conversion to Flash – This approach discusses about the difficulties in converting applets to flash. The difficulties are summarized as

1. **Architectural differences between J2SE and flash** – The architecture of flash is different from that of J2SE, causing porting a cumbersome process

2. **No Multithreading** — Java is multithreaded environment where as Flash is a single threaded environment, it lacks support of multithreading. Multithread to a single thread environment conversion cannot be automated.

Syntax conversion – Java and Action script both have different syntax. Java syntax can be converted into Action script syntax J2AS3.

These conversion methods are explained in detail in the consecutive chapters. In conclusion, the automation of Java Applets into J2ME and Flash files is difficult and not feasible.

1.3 Adaptation of CDEEP live videos

Centre for Distance Education and Engineering Programme (CDEEP) is distance learning initiative by IIT Bombay. It provides class lectures as offline and live videos to remote locations. These CDEEP videos typically have a bit rate of 800 K bps - 1200 K bps. But GPRS connection has low bit rate of 40 k bps, causes huge delays. Also mobile users are charged on basis of data transferred, in the range of 10paise per 10 KB. Hence, for a viewing a video of size 600 MB of a one hour lecture video, the cost of viewing would be enormous. Hence Live videos have to be adapted to suite low network bandwidth of mobile device connection in real time.

There is a supporting thesis Study element based adaptation which adapts pre-recorded CDEEP lecture videos to Mobile Devices. This method has been adapted to support Live video Streaming of CDEEP lecture Videos.

Study element based System — In existing Study element based system, images are extracted from a stored video.The Extracted Images are classified into study elements as instructor, white paper and presentation. Images are sent to mobile devices at intervals based on user experience. This system works for CDEEP stored videos only.

Live Video Adaptation In Live Video Adaptation method, sets of 300 images are extracted from live video stream. The extracted images are classified into study elements by Tagging Mechanism. Audio is also captured from live stream and stored chunks of mp3 files. The extracted images and audio are sent to mobile user at different intervals based on user experience provided.

Image extraction Different approaches for Image extracting using VideoLAN (VLC), FFmpeg, and Mplayer have been explored. In VLC, image extraction has high cpu utilization and delay. In FFmpeg, image extraction has high CPU utilization. Audio is extracted using FFmpeg. In Mplayer the image extraction process is done.

Integrating into Existing System The Live Video adaptation module has been successfully integrated into Existing Study element based system. A comparison to existing system is also made in detail.

In conclusion, Live Video adaptation provides a cost effective and adaptive way for live streaming of CDEEP videos.

1.4 Organization of Thesis

The Thesis is Organized as follows.

1. Chapter 1 – Introduction and Aim of thesis.
2. Chapter 2 – Discusses about Adaptation of Applets to mobile devices.
3. Chapter 3 – Discusses about Adaption of CDEEP live video streaming to mobile devices.
4. Chpater 4 – Discusses about Implementation of Adaption of Live videos and comparison to Existing system
5. Chapter 5 – Conclusion and Future Work

Chapter 2

Applet Adaptation Methodology

Applets are animations shown in web browsers, created using J2SE technology, but animations in mobile devices are created using J2ME. Both are Java based only but they have different libraries. So applets can not be viewed directly on mobile devices.

2.1 J2ME based adaptation

2.1.1 J2SE

Java Technology is both a platform and language. Java can be used to create two types of programs: Applications and Applets. An Application is a program that runs on any computer, under the operating system of the computer. Java is not much different from any other computer language. Rather it is Java's ability to create applets that makes it important. An applet is an application designed to be transmitted over Internet and executed by a Java-compatible web browser. Java is simple, secure, portable, Object-oriented, Robust, Multi-threaded, and dynamic language.

2.1.2 Overview of J2ME

J2SE requires high processor memory and speed but mobile devices like mobiles and PDAs do not have such high RAM and speed. So, Mobile phones are installed with different version of Java Platform Micro Edition (J2ME).

General Architecture of J2ME

J2ME uses configurations and profiles to customize the Java Runtime Environment (JRE). As a complete JRE, J2ME is comprised of a configuration, which determines the JVM used, and a profile, which defines the application by adding domain-specific classes.

The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. The profile defines the application; specifically, it adds domain-specific classes to the J2ME configuration to define certain uses for devices. J2ME also provides technology specific APIs that extends the capabilities of a Java application environment.

J2ME architecture doesnt replace the operating system of a small computing device. Instead, J2ME architecture consists of layers located above the native operating system. The following Figure 2.1 depicts the relationship between the different virtual machines, configurations, and profiles.

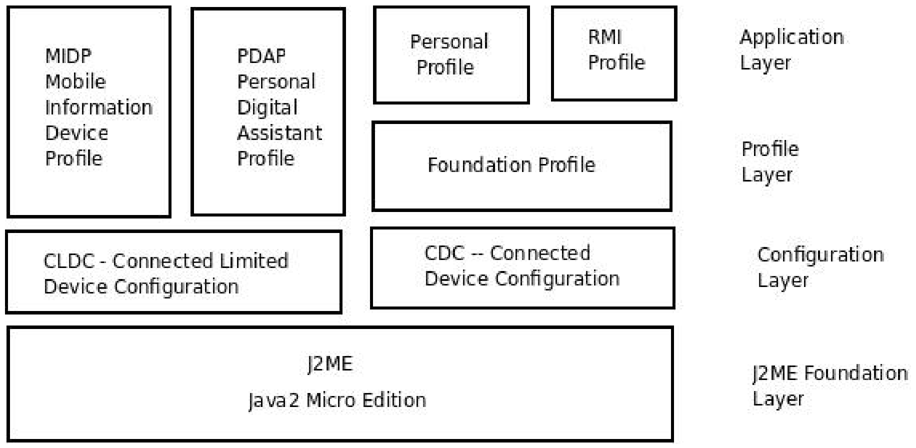

MIDP Mobile Information Device Profile
PDAP Personal Digital Assistant Profile
Personal Profile
RMI Profile
Application Layer
Foundation Profile
Profile Layer
CLDC - Connected Limited Device Configuration
CDC -- Connected Device Configuration
Configuration Layer
J2ME Java2 Micro Edition
J2ME Foundation Layer

Figure 2.1: Architecture of J2ME

The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. Currently, two configurations exist for J2ME, they are:

Connected Device Configuration (CDC)

Connected Device Configuration (CDC) has been defined as a stripped-down version of Java 2 Standard Edition (J2SE) with the CLDC classes added to it. Therefore, CDC was built upon CLDC, and as such, applications developed for CLDC devices also run on CDC devices.

CDC was designed for devices with minimum 512KB of read-only memory (ROM), 256KB of random access memory (RAM), and some kind of network connection. Example of CDC devices are smart phones, two-way pagers, PDAs, home appliances, point-of-sale terminals, and car navigation systems. These devices run a 32-bit microprocessor and have more than 2 MB of memory, which is needed to store the C virtual machine and libraries. While the K virtual machine supports CLDC, the C virtual machine (CVM) supports CDC. CDC is associated with the Foundation Profile.

Connected Limited Device Configuration (CLDC))

CLDC is aimed at smaller devices than those targeted by the CDC. It encompasses mobile phones, pagers, PDAs and other devices of smaller size. CLDC is designed for devices with 160KB to 512KB of total memory including a minimum of 160KB of ROM and 32KB of RAM. The reference implementation of CLDC is based on small JVM called KVM, and its KJava profile run on top of CLDC. KVM cant do everything a JVM does in J2SE world. CLDC outlines the most basic set of libraries and Java virtual machine features required for each implementation of J2ME on highly constrained devices.

CLDC Requirements CLDC defines the following requirements:

- Full Java language support (except for floating pointer support, finalization, and error handling).
- Full JVM support.
- Inherited classes – all classes not specific to CLDC must be subsets of J2SE 1.3 classes.
- Classes specific to CLDC are in javax.microedition package and subpackages.

In addition to the javax.microedition package, the CLDC API consists of subsets of the J2SE java.io, java.lang, and java.util packages. For example, the java.lang in J2ME (CLDC 1.1 version) has 17 classes and 1 interface, compared with the java.lang of Java SE (JDK 1.6 version), which has 35 classes and 8 interfaces. J2ME also possesses many features which are specific and limited to J2ME itself. The table 1 shows the core Java API's supported by CLDC 1.1 [9].

1. Native methods cannot be added at runtime. All native functionality is implemented in KVM only.
2. The KVM only includes a subset of the standard byte code verifier. This means that the task of verifying classes is split between the CLDC device and some external mechanism.

CLDC Vs CDC

The following figure 2.2 depicts the relationship between CDC and CLDC. It also illustrates their relationship to the full J2SE API.

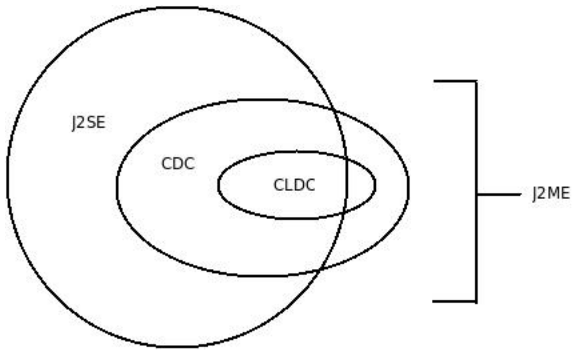


Figure 2.2: Relation between CLDC, CDC, and J2SE

Profile

A profile defines the type of device supported. The Mobile Information Device Profile (MIDP), for example, defines classes for cellular phones. It adds domain-specific classes to the J2ME configuration to define uses for similar devices. Two profiles have been defined for J2ME and are built upon CLDC: KJava and MIDP. Both KJava and MIDP are associated with CLDC and smaller devices.

Profiles are built on top of configurations. Because profiles are specific to the size of the device (amount of memory) on which an application runs, certain profiles are associated with certain configurations. A skeleton profile upon which you can create your own profile, the Foundation Profile, is available for CDC.

MIDP is geared toward mobile devices such as cellular phones and pagers. The MIDP, like KJava, is built upon CLDC and provides a standard run-time environment that allows new applications and services to be deployed dynamically on end-user devices.

MIDP is a common, industry-standard profile for mobile devices that is not dependent on a specific vendor. It is a complete and supported foundation for mobile application development.

MIDP contains the following packages, the first three of which are core CLDC packages, plus three MIDP-specific packages. They are: java.io, java.util, javax.microedition.io, javax.microedition.lcdui, javax.microedition.midlet, javax.microedition.rms.

2.1.3 J2ME vs Applets - Conversion Difficulties

Incompatability of Architectures of J2SE and J2ME

The Architecture of Applets in J2SE is different from a J2ME MIDlet. The below figure shows general architecture of Applets and MIDlets.

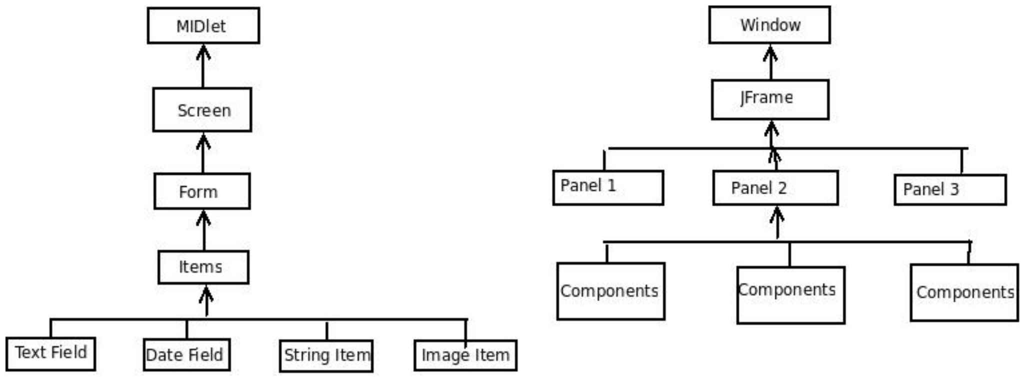


Figure 2.3: Architecture of J2ME MIDlet and J2SE applets

From the Figure 2.3 it can be observed that a MIDlet consists of a screen which is used to display content. Only one form can be displayed on screen and all the items are added to this form. MIDlet has a flat-file architecture. The screen cannot be divided in to multiple panels.

But in-case of J2SE applets, a window is divided into multiple panels. Each panel holds multiple components. So it has hierarchical architecture. The process of converting this hierarchal architecture to flat-file architecture is cumbersome process.

Amount of text on screen

The following code shows portion of a sample OSCAR animation.

```
public void init(){
```

```
ta = new TextArea("", 31, 47);
ta.setText("\n\t Energy Transaction in Biological Systems\n"+
"\nIn this animation you will learn the concepts of ATP Metabolism.\n"+
"ATP is the energy currency of the cell and transfers energy from
chemical bonds to endergonic (energy absorbing) reactions within the cell.\n"+
"\nATP stands for Adenosine Tri Phosphate which is a Nucleotide."+
"\nThis nucleotide consists of a Ribose, three Phosphate groups and a Nitrogenous
base Adenine.\n"+ "\n ADP + Energy + Pi  ----->  ATP\n"+"\nADP or Adenosine
Di Phosphate is formed by the removal of one phosphate from an ATP molecule.\n"+
"\nThe formation of ADP from ATP releases usable metabolic energy.");
}
```

The amount of text displayed in one applet screen is designed desktop resolution. This cannot be fit in one mobile screen. So the amount of text that has to be put per screen has to be replanned. This replanning cannot be automated.

Canvas size and co-ordinates of Image object

The Co-ordinates of various Image objects in an OSCAR animation screen has been designed for desktop resolution. For example the following code shows paint method in an OSCAR animation with the co-ordinates set manually at the coding time of the program.

```
public void paint(Graphics G){
g.drawImage(Image img1, 480, 600,this);
g.drawImage(Image img2, 530, 400,this);
g,drawImage(Image img3, 60, 370, this);
}
```

The position of image objects has to be replanned for mobile device resolution. This replanning cannot be automated.

Other General Differences between J2ME and J2SE

The other general differences between J2ME and J2SE are

- **No floating point support in J2ME**: The CLDC does not support floating point numbers. Therefore, no CLDC-based application can use floating point numbers or types such as float or double. This is mainly because CLDC target devices do not have floating point support in their hardware.

- **No finalize method** : The CLDC APIs do no include the Object.finalize method so final cleanup operations can not be performed on object data before the object is garbage collected.
- **No Java Native Interface (JNI)** : A Java virtual machine supporting the CLDC does not implement the Java Native Interface (JNI) primarily for security reasons. Also, implementing JNI is considered expensive, given the memory constraints of CLDC target devices.
- **No user-defined class loaders** : A Java virtual machine supporting the CLDC must have a built-in class loader that cannot be overridden or replaced by the user. This is mainly for security reasons.
- **No thread groups or daemon threads** : While a Java virtual machine supporting the CLDC implements multithreading, it cannot support thread groups or daemon threads. So to perform thread operations for groups of threads, use collection objects to store the thread objects at the application level.

2.1.4 J2SE Vs J2ME – Conversion Possibilities

Copy of paint method

J2SE uses Graphics class to show animation in applet. J2ME also uses the same Graphics class with some minor changes. So the paint method can be copied as it is during conversion. The following code shows paint method in J2SE animation.

```
In applet

public void paint( Graphics g)
{
g.drawString("Sending",150,75);
g.drawRect("40,50,10,20);
g.drawImg(Image img,20,30,center);
g.drawArc(30,40,10,30,60,120);
}
```

These methods are common for J2SE and J2ME from the above example

- drawString
- drawImage
- drawArc
- drawRect

Buttons to Key Mapping

Though J2ME doesn't display visual buttons, it uses mobile device keys to accept input. OSCAR animations on the other hand use visual buttons for accepting input like start, stop of animation. These visual buttons in the applets can be mapped to mobile device keys to get the same effect.

LWUIT

Light Weight User Interface Toolkit (LWUIT) is a UI library provided by Sun under GPLv2. It offers advanced UI capabilities and a clean API that is inspired by swing. It removes the necessity to write device specific code for different screen sizes. This UI library solves the hierarchical problem present in J2ME. LWUIT allows user to have multiple panels in one screen.

2.2 Flash Based Adaptation

2.2.1 Overview of Flash and Action script

Adobe Flash is a multimedia platform language used to create animations, advertisements. Flash has support for an embedding scripting language called Action Script (AS), which is based on ECMA script. Flash animations can be controlled, provides interaction with animation. Flash movies are easier to create, more stable on web browsers than to Java applets. Action Script has its own rules for grammar and punctuation that determine which characters and words are used to create meaning and in which order they can be written. Action Script has evolved in to an Object Oriented Script which has the capability similar to Java Script. Action Script in flash lets to do anything with animations and also things which cannot be done using animation techniques. Action

Script can be used to create almost all of the interactivity seen in many flash applications. Flash movies are saved as Shock Wave Files with .swf file extension, which can be played in Flash Player. Recently Flash has also provided support for XML to render rich content in the browser. This technology has been made a formal approach called Adobe Flex, which uses Flash runtime to build Rich Internet Applications.

Adobe Flash Lite is a highly optimized implementation of Flash runtime for mobiles, consumer electronic devices, and Internet connected digital home devices. Flash technology on mobile phones provides the same advantages as using flash on desktop. Rich, interactive, files can be built providing a quicker time to market. Flash Lite has the ability to send and receive data over HTTP. This enables user to load data (and SWFs) into your application from web browser, giving benefit of dynamic content in an installed application.

2.2.2 Java to Action Script – Conversion Difficulties

No Multi-Threading

Multi-Threading – A multithreaded program contains two or more parts that can run concurrently. Each part of such a program is called a thread,and each thread defines a separate path of execution. Thus, multithreading is a specialized form of multitasking. Java is multithreaded environment.

Flash is a single threaded environment, means it does not have support of threads. Threading is essential ability to have the code do something in background while the UI is doing something else. Because action script is single-thread environment it spends a lot of time doing heavy computation, the UI can not be updated. If the action script does not stop running, flash will kill after 60 seconds.

The following code shows a part of OSCAR animation with thread class. This cannot be implemented incase of action script.

```
class Timer extends Thread {
public  int myTime = 0;
public static int myTime1 = 0;
Network myNetwork;
```

```
Component myComponent;
int myInterval;
void set time(){
....
}
void sleep(){
....
}
```

No Function Overloading

Action Script doesn't allow programmer to overload a functions. For example in Java a default overloading will be like

```
public void defaultSetter(int iValue) {
    setter(iValue, false);
}

public void setter(int iValue, boolean bForce) {
    // whatever
}
```

In Action script it is written as

```
public function setter(iValue:int, bForce:Boolean = false):void {
    // whatever
}
```

This overload can be done for one function. But if there are multiple methods that take different types, then there will be a lot funtions like setFormString() and setFormInt(), etc. This is fine but if the constructor is the one being overloaded, it is a cumbersome process to porting java to action script.

Other General Difficulties in Action script compared to Java are

- **Limited Data Structures**: Action script only supports Arrarys and Dictionary, no queues, no linked list, no stack compared to Java. It would cause a trouble if porting is performed to action script from Java classes. It is better to re-implement the data structures that are wrapped.

- **No Yielding or Blocking** There is no yielding or blocking in Actionscript. If the next line of code is supposed to run, user cannot prevent the next line of code from running. That means that when you call *Alert.show()*, the next line of code following that runs right away. In many other runtimes, the Alert window has to be closed before the next line of code continues. The application has to be refactored into event driven.

Syntax Conversion from Java to Action Script

Action Script is much similar to Java Script with a little change in syntax. For example a Java function looks as

```
public ReturnType methodName(ParamType1 param1, ParamType2 param2)
```

Action Script Equivalent is

```
public function methodName(param1:ParamType1, param2:ParamType2):ReturnType
```

Syntax is easy to change. So, Java Script to Action Script 3 (J2AS3) [13] converter was developed. It is an open source application written in Adobe AIR that can use to convert Java Files to Action Script 3 files. It only does Syntactic conversion of functions, data types of variables. Manually change other parts of the code has to be done in order to build an action script file. The rest of the conversion process cannot be automated.

Thus adaptation of applets to mobile devices is difficult and not feasible, so Adaptation of Live videos has been explored in next chapter.

Chapter 3

Live video Adaptation

3.1 Video Transcoding and its limitations

3.1.1 Video Transcoding

Video Transcoding is the operation of converting a video from one format into another format. A format is defined by characteristics like bit rate, frame rate, spatial resolution, coding syntax and content. A multimedia system may consist of various devices like PCs, Laptops, PDAs, mobile phones, etc interconnected via heterogeneous wireline and wireless networks. Each client in Heterogeneous system has diverse capabilities like screen resolution, speed, memory, decoders, etc. Transcoding enables Heterogeneous systems to exchange content on the Internet. Transcoding process is shown in Figure 3.1.

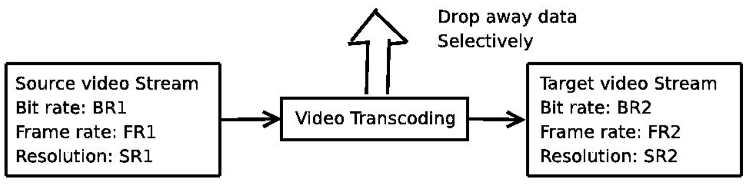


Figure 3.1: Transcoding process

The bit rate is the main determinant of the quality of encoded video file — the more the bits, the better in general the video will look. But, some types of images like JPG require fewer bits to look good. Each picture is called a frame and multiple frames are required to create the illusion of moving picture. The frame rate defines how many frames are displayed per second and this value usually ranges from about 24 to 60. The minimum frame rate needed for motion is about 24 frames per second (fps) and this is standard used for movies. Resolution is number of pixels used to display a frame. The bandwidth required to display is approximately equal to resolution multiplied by frame rate. Thus video transcoding is one of the essential components for providing universal access.

3.2 Existing System – Study Element Based Adaptation

Centre for Distance Education and Engineering Programme (CDEEP) is an distance initiative learning by Indian Institute of Technology Bombay. CDEEP provides Video Lectures and Live Streaming to remote locations. The instructor uses mainly two modes of instructions namely presentational slides and explanation on a white paper element. One of the Methods of adaptation is Study Element Based Adaptation [10].

The existing system using Study Element Based Adaptation for pre-recorded CDEEP Live videos to Mobile Devices. The Architecture of Existing system is shown in Figure 3.2.

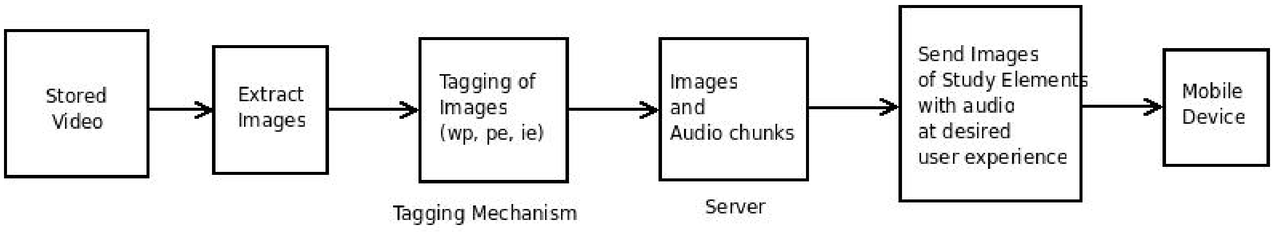

Stored Video
Extract Images
Tagging of Images (wp, pe, ie)
Tagging Mechanism
Images and Audio chunks
Server
Send Images of Study Elements with audio at desired user experience
Mobile Device

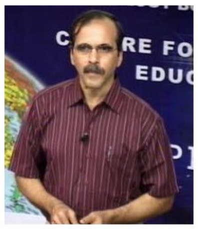
Figure 3.2: Existing Study Element Based Adaptation for stored videos

In the Existing system, the videos are pre-recorded and stored in repository viz. entire video is available for processing. The images extracted from stored video are sent to tagging mechanism. In the Tagging Mechanism, the images are classified in to Study elements. After classification, the images and extracted audio are stored in the server manually. The server sends images and audio to mobile user based on user experience and network bandwidth. This system works only stored CDEEP live videos. All the extracting and tagging mechanism are performed offline.

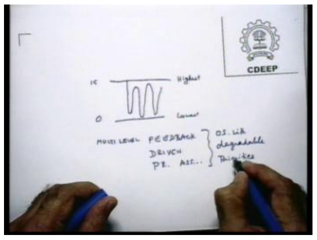
Three types of study elements are categorized according to [10]. They are

1. **Presentation Element** - Portion of video that shows one slide of a presentation
2. **White Paper Element** - Portion of video that shows white paper on which instructor is writing
3. **Instructor Element** - Portion of video that shows instructor talking

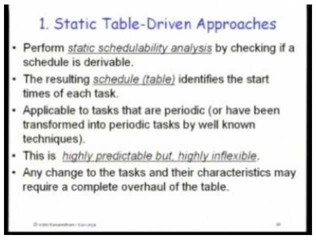
Sample images of the elements are shown in Figure 3.3



(a) Instructor Element



(b) White Paper Element



1. Static Table-Driven Approaches

- Perform *static schedulability analysis* by checking if a schedule is derivable.
- The resulting *schedule (table)* identifies the start times of each task.
- Applicable to tasks that are periodic (or have been transformed into periodic tasks by well known techniques).
- This is *highly predictable but, highly inflexible*.
- Any change to the tasks and their characteristics may require a complete overhaul of the table.

(c) Presentation Element

Figure 3.3: Types of Study Elements

The study element based adaptation procedure has 3 basic steps

1. **Video Tagging**
2. **Building User experience**
3. **Finding Sending Intervals**

Building User experience and Finding sending Intervals can be referred from [10]. In this thesis, Video Tagging method is exploited to generate the specific results. Video Tagging Technique uses feature based tagging to identify the boundaries of the study element. This is used identify the Starting time, Ending time and Type of Element, details are stored in an XML file.

Classifying Images into Elements

As mentioned above Study Element Based Tagging uses feature based tagging. Here three elements are classified based on some features present in element.

1. Instructor element — feature used is Face Recognition
2. White paper element — feature used is CDEEP logo image

3. Presentation Element — No specific feature, categorized based on background color (since it doesn't change)

The downside of this method is that there can be misclassification because some times features could not recognized correctly like face recognition. In real time these errors can not be corrected, corrections cause huge delay.

3.3 Proposed system

A supporting thesis Study element based adaptation which adapts pre-recorded CDEEP lecture videos to Mobile Devices. This method has been adapted to support Live video Streaming of CDEEP lecture Videos. The Block Diagram is shown in Figure 3.4.

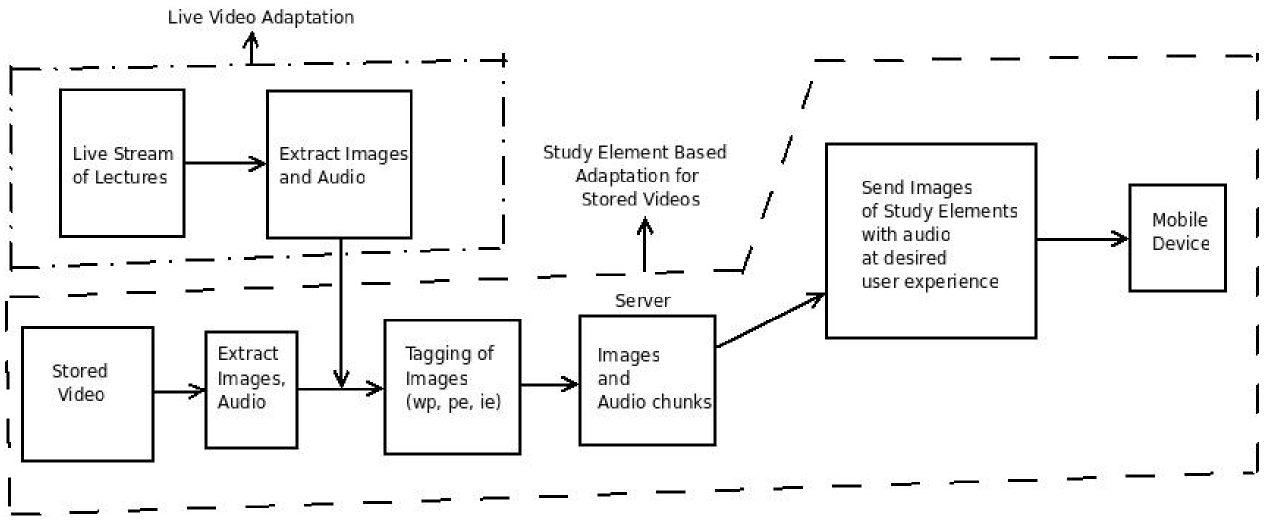

Live Video Adaptation
Live Stream of Lectures
Extract Images and Audio
Study Element Based Adaptation for Stored Videos
Send Images of Study Elements with audio at desired user experience
Mobile Device
Server
Stored Video
Extract Images, Audio
Tagging of Images (wp, pe, ie)
Images and Audio chunks

Figure 3.4: Block Diagram of Study Element Based Adaptation with Live video Adaptation

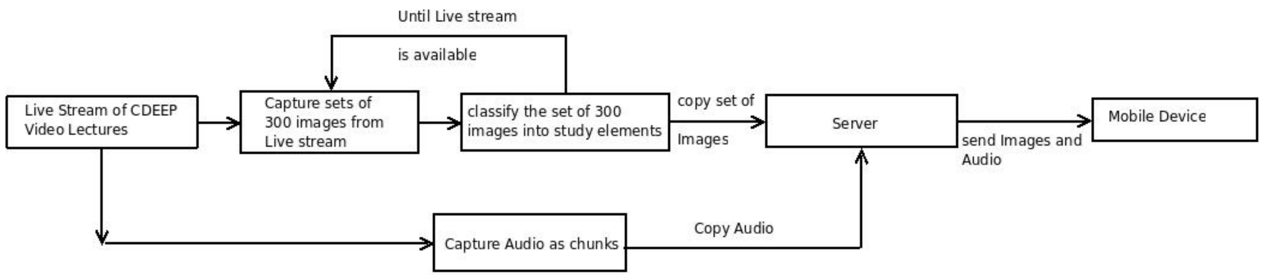

Until Live stream
is available
Live Stream of CDEEP Video Lectures
Capture sets of 300 images from Live stream
classify the set of 300 images into study elements
copy set of
Images
Server
send Images and Audio
Mobile Device
Capture Audio as chunks
Copy Audio

Figure 3.5: Live video Adaptation Process

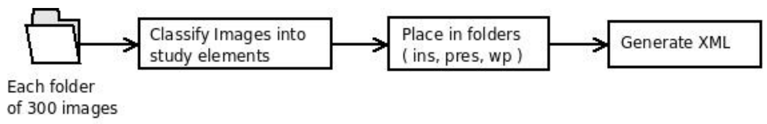

Each folder of 300 images
Classify Images into study elements
Place in folders (ins, pres, wp)
Generate XML

Figure 3.6: Tagging process

In this system, Live video Adaptation has been integrated as shown in above diagram 3.4, marked as Live video Adapatation module. In Live video adaptation module figure 3.5, Images are extracted from Live video Stream. The existing system can extract images from stored videos only because the entire video is available completely. In Live stream entire video is not available. So different approaches for extraction of images are discussed in later sections. Audio is extracted and saved as chunks. The Extracted images sent to Tagging mechanism to classify images as study based elements. After Tagging, Images and Audio chunks are stored dynamically in real-time in server and sent to mobile device at desired experience.

The Tagging process generates an XML file. This process for live video streaming is shown in Figure 3.6. The process is used for every 300 images for the total video. Each time new XML file is appended to old XML file. Thus making it to provide a live video stream on mobile devices.

Reason for choosing 300 images The method implemented may some times cause errors, because the Tagging procedure used has a constraint that it should start only with a Instructor Element. But after every 300 images it can not be sure that there is an instructor element. The tagging of 300 images took approximately 90 sec from analysis. Therefore, by reducing the no of bulk images processed at one time, it may result in more errors. If the number images processed at every time is increased, then initial delay will become high. So it better to take a value which is not very higher or very lesser.

3.4 Image Capture Methodologies

The first part of process is getting images from buffered Live Stream. Many media players can be used to extract images from stream, among them Video LAN (VLC) [16], FFmpeg [2], and Mplayer [5] are used as part of experiment. Many media players do not have the support to get images from Live Stream especially.

3.4.1 Approach 1: VideoLAN(VLC)

In Live video stream, the entire stream is not available. So initial approach was to record/save a small part of live stream say 5 minutes, and generate images from video. Using VLC, the video was recorded from live stream and also transcoded into the specified format.

Recording Live Stream to Video File

```
vlc --run-time=300 url :sout=#transcode{vcodec=mp2v,vb=3072,scale=1,acodec=mpga,
ab=128,channels=2} :duplicate{dst=std{access=file,mux=ps,dst="C:\VLC\myTestMovie.M
```

The above command records a video from url, transcodes the video into specified MPG format. Once recorded, the images were extracted from saved file by take a screen shot of VLC window using the following command.

Extracting images from recorded file

```
vlc -V image --image-out-format jpg --image-out-ratio 24 --image-out-prefix snap
test.mpg vlc:quit
```

Disadvantages

- This process requires high CPU utilization because of transcoding.
- Delay between actual video and seeing video increases for each iteration as transcoding of video file takes time.

Extraction from MJPEG stream

Since incoming stream could not be edited directly, another alternate idea was to transcode live video stream to MJPEG [1] stream (sequence of images) and chain stream it. These MJPEG images are separated by delimiter, which might help in extracting images from stream directly.

Problems with MJPEG stream

In this process transcoding itself has high CPU utilization. At the receiver side the extracted images with out delimiters could not displayed, only VLC was able to display them. These images can only be rendered on a browser with a special multimedia type.

3.4.2 Approach 2: FFmpeg

In this approach, the images were extracted from Live Stream with out saving the video. Here FFmpeg [2] tool is used to extract images from live stream directly.

Extracting Images

```
ffmpeg -i url -an -r 1 -f image -s 320x240 video%d.jpg
```

Images are saved in sequential order at 1 second interval in a single file. A group of images say every 300 images are sent to Tagging Mechanism. The tagging mechanism classify images in to study elements. Audio is recorded from live stream in small audio chunks using ffmpeg.

Extracting Audio chunks

```
ffmpeg -i url -t 00:00:20 -acodec libmp3lame -ab 24 -ar 8000 audio.mp3
```

The Audio chunk of every 20 seconds is recorded and saved in the mp3 format with bit rate of 24 Kbits and sampling rate of 8000 Hz.

3.4.3 Approach 3: Mplayer

In this approach, the extracting and grouping mechanism are combined in to one statement. Mplayer can capture images directly from the live stream and could separate as a set of images in separate directories which is easier to process.

Extracting Images

```
mplayer -nosound -vf framestep=25 -vo jpeg:subdirs=img:maxfiles=300 url
```

The above command captures an image every 25 frames, store every group of 300 images in to a single directory [5] with sequential numbering. The directories are given to Tagging Mechanism. This approach is currently used in this process for extracting of images. The Audio is generated as explained in approach 2 using ffmpeg

3.5 Comparison with Existing System

- Supports Automatic Creation of offline copy of live video. In the existing system, the process of creating the offline copy has to be done by video author using automated tools shipped as part of it.
- Enables Viewing of live video as soon as it starts. The existing system works only for pre-recorded videos and hence after completion of lecture.
- In the existing system, any misclassification of images is corrected by the video author. But incase of Live video Streaming misclassifications can not be corrected.

Chapter 4

System Implementation

The system is meant to provide content to mobile device user. There are certain issues that arouse during implementation on mobile device which are briefly discussed below.

4.1 Port Blocking and NAT in Mobiles

Mobile device IP address bases on the NAT (Network Address Translation) performed by the gateway of the service provider. This will result a ”masquerading” IP address, for which reason the mobile device is not visible to the outside world with that IP. Such IP address is called ”private”. As a result, the mobile device web browser in a ”masqueraded network” can browse a outside website, but this does not work the other way. It depends on the network administrator if the translation table entries will be configured for permanent use, often referred to ”static NAT” or port forwarding, thus allowing traffic from outside network to access into that ”hidden” network.

The proxy server implemented by service provider does not allow direct TCP socket connections to any port except for HTTP ports 80 and 8080, all other ports are blocked by service provider. This is tested by a small program that simply listens on a specific port say 1234. The program would display whenever the remote host is connects and would display anything the remote is sending. When mobile host tried to connect to server through GPRS, the connection could not be made.

Hence a web server along with Java servlets has been used as a server to which all client mobile application connect. Since all requests goes through HTTP, do not have port block issue. Port blocking has direct impact on streaming, the stream can not be send directly to mobile client. Instead streaming has to be RTSP based, simply as audio on demand with a RTSP server. There are issues in deploying using RTSP, because mobile devices have small memory and computing power which makes it difficult to buffer stream and play it at the same time. It uses progressive download technique only, it will only play after the entire the stream is downloaded.

4.2 System Diagram

The system has 3 components.

1. **Web Server** — Contains all images and audio files, Java servlet which handles all requests.
2. **Mobile Client** — A J2ME application used to connect to server and display content.
3. **user**

The Fig 4.1 shows the sequence of commands involved in playing video. Initial and default settings of speed and user experience are set, but these can be changed according to user demands.

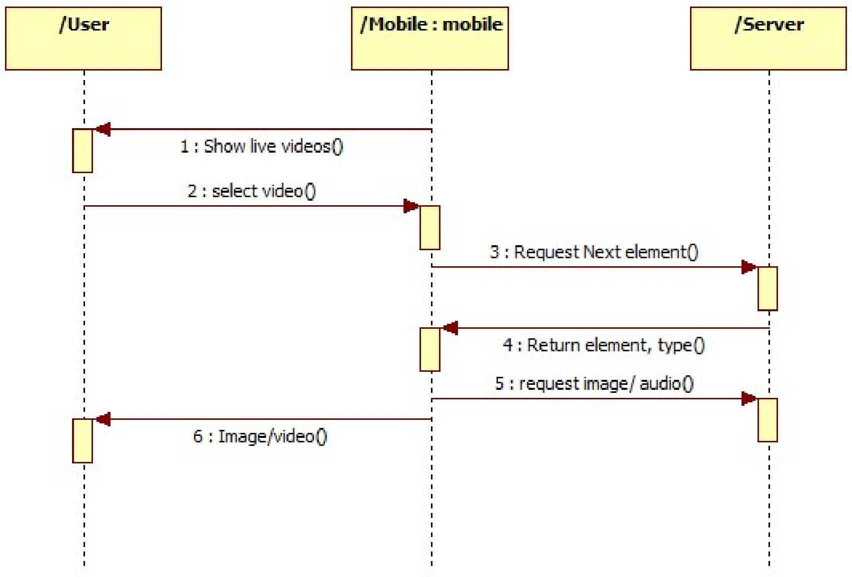
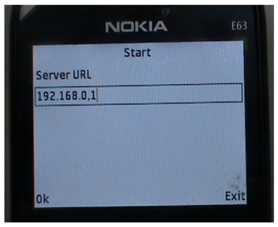


Figure 4.1: Sequence Diagram

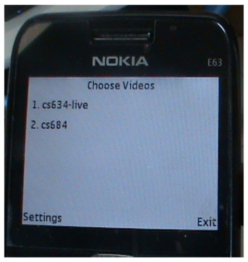
The user requests for current list of video's. The server returns with list of current Live video's video's. Once user selects the video, the images are sent based on user experience input, one after the other. In the background the audio file is played on mobile device using progressive download technique i.e., when is first chunk is being played, the second chunk is being downloaded and so on. Every time 300 images are inserted in the server after the tagging process in real time. Before the first 300 images can be sent, the next 300 images will be kept in server with updated XML file.

4.3 Watching Live video

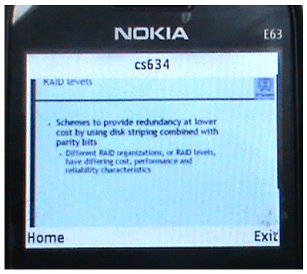
The below images 4.2 show live video's being streamed on mobile phones. User inputs enters the http URL of the server. He is shown the list of Live video's being streamed, user selects the Live video. According to the user settings, images are downloaded one after the other with specified user interval. While the Images are being downloaded, new images are added dynamically to the server. So, there will be no delay once the Live Video Stream is started.



(a) Start Page



(b) Video List


NOKIA
E63
cs634
RAID levels
Schemes to provide redundancy at lower cost by using disk striping combined with parity bits
Different RAID organizations, or RAID levels, have differing cost, performance and reliability characteristics
Home
Exit

(c) Slideshow of Images

Figure 4.2: Seeing a Video

In Figure 4.2, the Start page shows user input screen for URl. The Video list screen shows the current list of Live and Stored CDEEP Videos in the repository. Once the video is selected, the images are downloaded to mobile device and shown as Slideshow of Images. The Audio is played along with the slideshow.

Chapter 5

Conclusion and Future Work

In conclusion Java applets cannot be ported on to mobiles directly. J2SE code transformation to J2ME or Action script is very difficult and not feasible. It requires large resources. It is easy to start from scratch in both J2ME and Action Script to build the similar applet. The present Study Based Based Element Adaptation to Live CDEEP video lectures can be adapted to Mobile devices with low memory and networks with low bandwidth as a cost effective method.

As Future, there are certain area's that can be improved in Live Streaming of CDEEP video to make it more efficient. They are

1. **Tagging Efficiency** – The study based element adaptation is prone to errors in classification of images, improving this method by better classification methods is necessary.
2. **Text Improvement** – The presentation and white paper elements have much text content. Improving text quality in these elements
3. **Synchronization of Images and Audio** – There is no synchronization between audio and images. Implement synchronization will give better user experience.

Bibliography

[1] Mjpeg tools. `http://mjpeg.sourceforge.net`.

[2] Fabrice Bellard. Ffmpeg codec manual. `http://www.ffmpeg.org/ffmpeg-doc.html`.

[3] Hojung Cha, Jongmin Lee, Jongho Nang, Sungyong Park, Jin-Hwan Jeong, Chuck Yoo, and Jin-Young Choi. A video streaming system for mobile phones: Practice and experience. *Wireless Networks*, 11(3):265–274, 2005.

[4] Adobe Flex. Programming action script 3.0 manual. `http://livedocs.adobe.com/flex/3/progAS_flex3.pdf`.

[5] Free Software Foundation. Mplayer manual. `http://www.mplayerhq.hu/DOCS/HTML/en/index.html`.

[6] H.264. H.264 format. `http://ati.amd.com/products/pdf/h264_whitepaper.pdf`.

[7] ISO/IEC. Mpeg-1 coding of moving pictures and associated audio for digital storage media at up to about 1,5 mbit/s. *ISO/IEC 11172*, 1993.

[8] ISO/IEC. Mpeg-2 generic coding of moving pictures and associated audio information. *ISO/IEC 13818*, 1996.

[9] Jonathank Knudsen and Sing Li. *Beginning J2ME: From Novice to Professional*. Apress, third edition, 2005.

[10] G.N. Murthy and S. Iyer. Study element based adaptation of lecture videos to mobile devices. In *Communications (NCC), 2010 National Conference on*, pages 1–5, 29-31 2010.

[11] OSCAR Project. Overview. `http://oscar.iitb.ac.in/`.

[12] Herbert Schildt. *Java 2 the complete reference*. Osborne/McGraw-Hill, fourth edition, 2001.

[13] Java to Action Script Converter. Home. `http://code.google.com/p/j2as3/`.

[14] Transcoding. Internet transcoding for universal access. `http://www.research.ibm.com/networked_data_systems/transcoding/`.

[15] A. Vetro, C. Christopoulos, and Huifang Sun. Video transcoding architectures and techniques: an overview. *IEEE Signal Processing Magazine*, 20(2):18–29, mar 2003.

[16] VLC. Video lan project. `http://www.videolan.org/`.