

IMPLEMENTATION OF A WIRELESS STREAMING SYSTEM FOR  
UNIVERSAL MULTIMEDIA ACCESS

A THESIS SUBMITTED TO  
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES  
OF  
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

HALİM ÜNSEM ÜNAL

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF  
MASTER OF SCIENCE

IN  
THE DEPARTMENT OF ELECTRICAL-ELECTRONICS ENGINEERING

AUGUST 2003

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

---

Ünsem Ünal

## Approval of the Graduate School of Natural and Applied Sciences

---

Prof. Dr. Canan Özgen

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science

---

Prof. Dr. Mübeccel Demirekler

Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

---

Assoc.Prof.Dr. Gözde Bozdağı Akar

Supervisor

---

Assoc.Prof.Dr. A.Aydın Alatan

Co-Supervisor

Examining Committee Members

Prof.Dr. Semih Bilgen

Assoc.Prof.Dr. Gözde Bozdağı Akar

Assoc.Prof.Dr. A. Aydın Alatan

Assist.Prof.Dr. Nail Akar

Assist.Prof.Dr. Cüneyt Bazlamaçcı

## **ABSTRACT**

# **IMPLEMENTATION OF A WIRELESS STREAMING SYSTEM FOR UNIVERSAL MULTIMEDIA ACCESS**

Ünal, Ünsem

M.Sc., Department of Electrical and Electronics Engineering

Supervisor: Assoc.Prof.Dr. Gözde Bozdağı Akar

August 2003, 67 pages

This thesis describes a universal multimedia access system and its implementation details. In the context of this thesis, universal multimedia access means accessing multimedia content over ubiquitous computer networks, using different computing platforms. The computer networks involve both wired and wireless networks, and computing platforms involve wired PC's, mobile PC's and personal digital assistants (PDA). The system is built on client/server architecture. Video data is H.263 coded and carried over RTP. Java Media Framework is utilized and its capabilities are extended with special plug-ins when necessary.

**Keywords:** Multimedia Access, Mobile Computing, Wireless Networks, Real-time Video, Personal Digital Assistant

## ÖZ

# GENEL ÇOKLU ORTAM ERİŞİMİ İÇİN BİR KABLOSUZ AKIŞ SİSTEMİ GERÇEKLEŞTİRİMİ

Ünal, Ünsem

Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü

Tez Yöneticisi: Doç.Dr. Gözde Bozdağı Akar

Ağustos 2003, 67 sayfa

Bu tez çokluortam verisine genel erişim sağlayan bir sistemi ve gerçekleştirim detaylarını anlatmaktadır. Tez kapsamında çokluortam verisine erişmek, çokluortam içeriğine her yerde olan bilgisayar ağları üzerinden farklı bilgisayar platformları kullanarak erişmek anlamına gelmektedir. Bahsedilen bilgisayar ağları hem kablolu hem kablosuz ağları; bahsedilen bilgisayar platformları kablolu PC'leri, taşınabilir PC'leri ve kişisel sayısal asistanları kapsamaktadır. Sistem istemci/sunucu mimarisi üzerine kurulmuştur. Görüntü verileri H.263 kodlanmış ve RTP üzerinden taşınmaktadır. Java Media Framework'den yararlanılmış ve gereken durumlarda özel eklentiler ile yetenekleri genişletilmiştir.

Anahtar kelimeler: Çokluortam Erişimi, Gezici Hesaplama, Kablosuz Ağlar, Gerçek-Zamanlı Görüntü, Kişisel Sayısal Asistan

## TABLE OF CONTENTS

ABSTRACT .....	iv
ÖZ.....	v
TABLE OF CONTENTS .....	vi
LIST OF TABLES .....	viii
LIST OF FIGURES.....	ix
LIST OF ABBREVIATIONS.....	xi
INTRODUCTION.....	1
1.1. Problem Definition .....	1
1.2. Scope of the Thesis.....	2
1.3. Outline of the Thesis .....	3
BACKGROUND.....	4
2.1. Real-time Transport Protocol .....	5
2.2. Java™ Media Framework .....	7
2.3. H.263 [17] .....	9
2.3.1. Fundamentals of Video Compression .....	11
2.4. Bluetooth .....	13
2.4.1. Bluetooth Profiles [13] .....	14
2.4.1.1. LAN Access Profile.....	14
2.5. Microsoft Windows CE 3.0 for PocketPC 2002 [14].....	15

	vii
2.6. Unified Modeling Language [15].....	17
ARCHITECTURE.....	20
3.1. Software Architecture.....	20
3.2. User-Interface Samples .....	34
EXPERIMENTS.....	38
4.1. System Configuration.....	38
4.2. Experiment Configurations .....	39
4.3. Experiment Results.....	40
4.4. Comparison .....	49
CONCLUSION .....	51
5.1. Results .....	51
5.2. Future Work .....	52
REFERENCES.....	54

## LIST OF TABLES

Table 1 Picture Formats Supported by H.263 .....	10
Table 2 Descriptions of Architectural Views .....	21
Table 3 Icons for Analysis Classes.....	23
Table 4 Logical Package Descriptions .....	29
Table 5 Output Bit rate Of Encoder vs Source Frame Rate .....	40
Table 6 Calculated Values Of The Experiments .....	46



## LIST OF FIGURES

Figure 1 Technology Relations .....	4
Figure 2 RTP Header Fields[3] .....	5
Figure 3 Recording, processing and presenting time-based media [11] .....	7
Figure 4 High Level JMF Architecture [11] .....	8
Figure 5 Components of an intra frame.....	11
Figure 6 Relations between I-Frame and P-Frame.....	12
Figure 7 Bluetooth Profiles [13].....	14
Figure 8 PocketPC Hardware [14] .....	16
Figure 9 Architectural Views .....	20
Figure 10 Use-case View Of The System Architecture .....	22
Figure 11 Analysis Model Class Diagram .....	24
Figure 12 Login Realization Sequence Diagram .....	25
Figure 13 Watch Movie Realization Sequence Diagram .....	26
Figure 14 Deployment View Of The Architecture.....	27
Figure 15 Logical View of the Architecture .....	28
Figure 16 Server Package Contents.....	30
Figure 17 Client Package Contents .....	31
Figure 18 UI Package Contents.....	32
Figure 19 Common Package Classes .....	32

	x
Figure 20 Native Decoder Contents .....	33
Figure 21 Process View Of The System Architecture .....	33
Figure 22 GUI Of LoginForm .....	34
Figure 23 GUI Of OptionList.....	35
Figure 24 GUI Of MovieList.....	35
Figure 25 GUI Of MoviePlayer 1.....	36
Figure 26 GUI Of MoviePlayer 2.....	36
Figure 27 GUI Of MoviePlayer 3.....	37
Figure 28 Simplified Block Diagram .....	38
Figure 29 5 FPS Source Rate Experiment Results .....	41
Figure 30 10 FPS Source Rate Experiment Results .....	42
Figure 31 15 FPS Source Rate Experiment Results .....	43
Figure 32 20 FPS Source Rate Experiment Results .....	44
Figure 33 25 FPS Source Rate Experiment Results .....	45
Figure 34 Difference of Souce And Displayed Frame Rate ( $\Delta$ FPS).....	47
Figure 35 Variance Of Displayed Frame Rate ( $\sigma^2$ ).....	48

## LIST OF ABBREVIATIONS

CSRC	Contributing Source Identifier
DCT	Discrete Cosine Transform
DLL	Dynamic Link Library
FPS	Frames Per Second
GOB	Group Of Blocks
GSM	Global System for Mobile Communications
IP	Internet Protocol
IR	Infrared
JMF	Java Media Framework of SUN Microsystems
JNI	Java Native Interface
KFR	Key Frame Rate (number of P-Frames between I-Frames)
MB	Macroblock
MFC	Microsoft Foundation Classes
PDA	Personal Digital Assistant
PPP	Point-To-Point Protocol
RFCOMM	The transport service defined in Bluetooth specifications

RMI	Remote Method Invocation
RTCP	Real-Time Transport Control Protocol
RTP	Real-Time Transport Protocol
SSRC	Synchronization Source Identifier
TCP	Transmission Control Protocol
UML	Unified Modeling Language
Wi-Fi	IEEE 802.11 Wireless LAN
Win32	Windows API provided by Microsoft
XMI	XML Metadata Interchange

# CHAPTER I

## INTRODUCTION

### 1.1. Problem Definition

The rapid growth of wireless communication and access, together with the success of Internet, has brought a new era of mobile/wireless multimedia applications and services. The convergence of Internet, wireless, and multimedia has created a new paradigm of research and development that enables multimedia content to move seamlessly between Internet and mobile wireless networks. With the benefit of increase in bandwidth in wireless networks, new access capabilities including mobile visual phones, personal digital assistants, and video streaming are pervading people's everyday life. Such mobile web services provide us an enhanced ability to access to Internet content anytime, anywhere, and from any device.

However, the converging technologies are the results of individual uncoordinated studies. Their design requirements and criterias are not related to each other's. Hence, most of the time there is plenty of work to be done while connecting different kinds of networks. For example, Internet is based on TCP/IP transmission protocol, which provides reliable connection oriented transmission mechanisms. Applications utilizing TCP are not affected by the floating bandwidth capacity of the worldwide internet network. On the other hand, multimedia content delivery is very much dependent on the capacity and the quality of the connection between two end points. Therefore, it is difficult to carry multimedia data using TCP on the Internet [1]. There are a number of solutions that address this problem [2].

When wireless/mobile networks come into the picture, other problems arise due to the nature of communication in noisy channels [4]. Problems mainly fall into two categories: coding techniques and transmission mechanisms. Some of them are, multi-path fading, shadowing, inter-symbol interference, and noise disturbance. There are many solutions to the defined problems of wireless/mobile networks [5] [6] [7].

On the other hand, the advances in computer architecture, display devices, power supplies and VLSI techniques opened a new dimension [8]. Computing devices are at pocket size now [9] [10]. These devices cannot be thought without Internet connectivity. Nowadays, there are some devices even with more than one network access, such as GSM and wi-fi or GSM, IR and Bluetooth. The multimedia capabilities of these devices are becoming comparable to the desktops. Web services of multimedia are within the reach of small devices now.

Rapid growth of the mobile device market and wireless/mobile networks pushes us to think about delivering multimedia web services to the mobile world.

## **1.2. Scope of the Thesis**

With the idea of bringing multimedia web services to the mobile world, it is necessary to construct such a system that enables both fixed and mobile users to access content. It is our requirement to design a system so that content is available from any device anywhere.

In this thesis, a multimedia web service is proposed as a universal multimedia access system. The system is able to deliver live or stored video to mobile and fixed clients. Both wired and wireless networks can be used to access the video content.

The proposed system is also implemented. Experiments regarding the playback performance of the system are obtained.

Although there are a number of similar systems in the market, our implementation differs from them in a number of points. First of all, a customizable and extendable architecture is required. Portability of object code has to be maintained since there will be different target clients that run operating environments of different vendors. Moreover, in order to measure the performance of the system, we should be able to access the inner parts of the implementation and have access to the source code.

Another contribution of the thesis is in UML. All the system has been analyzed and designed according to the object-oriented software engineering techniques. UML is the inevitable choice of modeling language in this situation.

In the context of this thesis, you will be reviewing an object based; multimedia web-service that is customizable, expandable and portable such that different targets running different operating environments can be clients.

### **1.3. Outline of the Thesis**

In the next chapter, technical background is given in order to make the text easier to read. Background information covers the basic concepts of video streaming, existing and used technologies.

In Chapter III, the architecture of the system is elaborated from the hardware and software point of views; design criteria, and algorithms are presented.

In Chapter IV, information about the experiments and the results are given. Comparison between this system and other available commercial systems are included.

In the last chapter, conclusions and possible future extensions to this work are discussed.

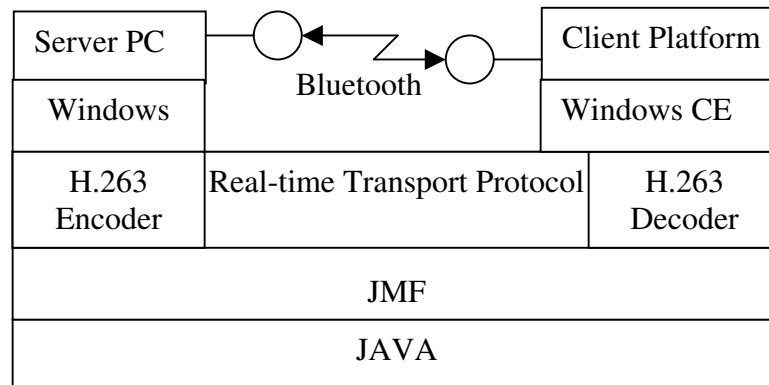
## CHAPTER II

### BACKGROUND

Accessing information anywhere anytime is a complex goal. It is required to unburden what we have in hand, melt the available in the same pot and cultivate such that the blend of the outcome is satisfying.

In order to design and implement such a system, complete understanding of the available is indispensable. This chapter, hereby, shall unburden the available. It, of course, is far from a complete, detailed description of the existing technology and tools. Instead, the information provided here can make it easy to understand the further chapters.

The following figure expresses the coordination of technologies used.



**Figure 1 Technology Relations**



## 2.1. Real-time Transport Protocol

Real-time Transport Protocol is being widely used for delivery of multimedia data in streaming applications [3]. RTP provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or experiment data, over multicast or unicast network services. RTP does not address resource reservation and does not guarantee quality-of-service for real-time services. The data transport is augmented by a control protocol (RTCP) to allow monitoring of the data delivery in a manner scalable to large multicast networks, and to provide minimal control and identification functionality. RTP and RTCP are designed to be independent of the underlying transport and network layers. The protocol supports the use of RTP-level translators and mixers.

The following figure describes the fixed header of an RTP packet.

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
V=2		P	X	CC			M	PT					Sequence Number																		
Timestamp																															
Synchronization Source (SSRC) Identifier																															
Contributing Source (CSRC) Identifiers																															
...																															

**Figure 2 RTP Header Fields[3]**

The first twelve octets are present in every RTP packet, while the list of CSRC identifiers is present only when inserted by a mixer.

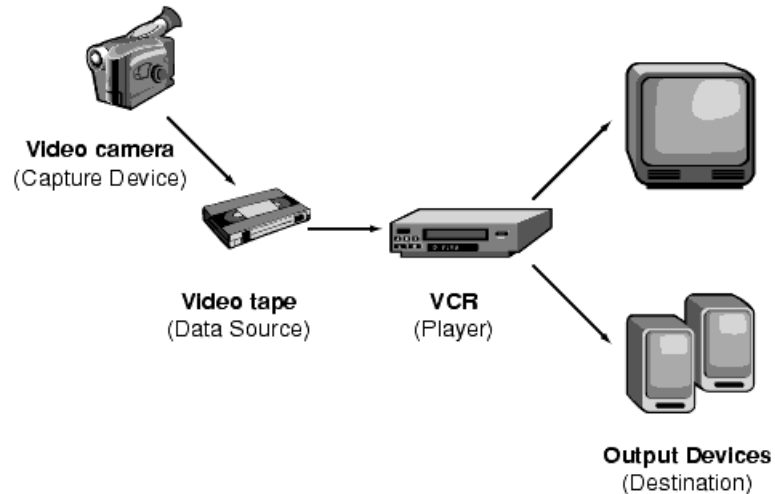
The fields have the following meaning:

- Version (V): 2 bits: This field identifies the version of RTP.

- Padding (P): 1 bit: If the padding bit is set, the packet contains one or more additional padding octets at the end, which are not part of the standard payload.
- Extension (X): 1 bit: If the extension bit is set, the fixed header is followed by exactly one header extension that defines the profiles.
- CSRC count (CC): 4 bits: The CSRC count contains the number of CSRC identifiers that follow the fixed header.
- Marker (M): 1 bit: The interpretation of the marker is defined by a profile. It is intended to allow significant events, such as frame boundaries to be marked in the packet stream.
- Payload type (PT): 7 bits: This field identifies the format of the RTP payload and determines its interpretation by the application. A profile specifies a default static mapping of payload type codes to payload formats.
- Sequence number: 16 bits: The sequence number increments by one for each RTP data packet sent, and may be used by the receiver to detect packet loss and to restore packet sequence.
- Timestamp: 32 bits: The timestamp reflects the sampling instant of the first octet in the RTP data packet. The sampling instant must be derived from a clock that increments monotonically and linearly in time to allow synchronization.
- SSRC: 32 bits: The SSRC field identifies the synchronization source.
- CSRC list: 0 to 15 items, 32 bits each: The CSRC list identifies the contributing sources for the payload contained in this packet.

## 2.2. Java™ Media Framework

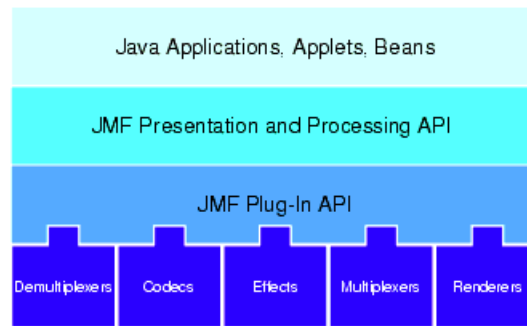
The Java™ Media Framework (JMF) [11] is an application-programming interface (API) for incorporating time-based media into Java applications and applets. JMF provides a unified architecture and messaging protocol for managing the acquisition, processing, and delivery of time-based media data. JMF supports most standard media content types, such as AIFF, AU, AVI, GSM, MIDI, MPEG, QuickTime, RMF, and WAV.



**Figure 3 Recording, processing and presenting time-based media [11]**

JMF uses the above basic model. A *data source* encapsulates the media stream much like a videotape and a *player* provides processing and control mechanisms similar to a VCR. Playing and capturing audio and video with JMF requires the appropriate input and output devices such as microphones, cameras, speakers, and monitors.

Data sources and players are integral parts of JMF's high-level API for managing the capture, presentation, and processing of time-based media. JMF also provides a lower-level API that supports the seamless integration of custom processing components and extensions. This layering provides Java developers with an easy-to-use API for incorporating time-based media into Java programs while maintaining the flexibility and extensibility required to support advanced media applications and future media technologies.



**Figure 4 High Level JMF Architecture [11]**

JMF has many media processing abilities. Among some of the features that JMF supports are

1. Capturing time based media
2. Transcoding between different media formats
3. Streaming time based media
4. Editing time based media
5. Plug-in architecture

Especially, the Plug-In architecture of JMF is important. Via custom plug-ins, it is possible to extend the capabilities of JMF. For example, by implementing one of the JMF plug-in interfaces, the media associated with a `Processor` can be directly accessed and manipulated:

1. Implementing the `Demultiplexer` interface enables you to control how individual tracks are extracted from a multiplexed media stream.
2. Implementing the `Codec` interface enables you to perform the processing required to decode compressed media data, convert media data from one format to another, and encode raw media data into a compressed format.
3. Implementing the `Effect` interface enables you to perform custom processing on the media data.

4. Implementing the `Multiplexer` interface enables you to specify how individual tracks are combined to form a single interleaved output stream for a `Processor`.
5. Implementing the `Renderer` interface enables you to control how data is processed and rendered to an output device.

Custom `Codec`, `Effect`, and `Renderer` plug-ins are available to a `Processor` through the `TrackControl` interface. In order to make a plug-in available to a default `Processor` or a `Processor` created with a `ProcessorModel`, one should register it with the `PlugInManager`. Once you have registered your plug-in, it is included in the list of plug-ins returned by the `PlugInManager` `getPlugInList` method and can be accessed by the `Manager` when it constructs a `Processor` object.

JMF has different performance and cross-platform implementations. On desktop Windows platforms, JMF has performance packs in which efficient audio and video codecs are implemented. On the other hand, cross-platform implementation had been implemented in pure Java and does not require any operating system dependent functionality. However the cross-platform implementation cannot satisfy high performance requirements for audio/video codecs when CPU processing power is limited.

In this thesis work, the plug-in mechanism has been used to port JMF to PocketPC. A native decoder has been implemented in C++ and added to JMF in order to obtain better performance.

### **2.3. H.263 [17]**

H.263 has been standardized in 1996. It was designed for low bit rate communication, early drafts specified data rates less than 64 Kbits/s, however this limitation has now been removed. It is expected that the standard will be used for a wide range of bit rates, not just low bit rate applications.

In order to understand how H.263 is used for video compression, it is necessary to briefly talk about digital video as a first step. Digital video is composed of a sequence of pictures, or frames, that occur at a certain rate. Each picture is composed of a number of samples. These samples are often referred to as pixels (picture elements) or simply pels. Each pixel contains a luminance component, denoted as Y, and two chrominance components  $C_B$  and  $C_R$ . For example, “black” is represented by  $Y=16$ , “white” is represented by  $Y=235$ , and the range of  $C_B$  and  $C_R$  is between 16 and 240, with 128 representing zero color difference (i.e. a shade of gray). The chrominance components however, typically have lower resolution than luminance in order to take advantage of the fact that human eyes are less sensitive to chrominance than the luminance. In H.263, the  $C_B$  and  $C_R$  components have half the resolution both horizontally and vertically, of the Y component. This is commonly referred to as the 4:2:0 format. Each  $C_B$  or  $C_R$  sample lies in the center of four neighboring Y samples.

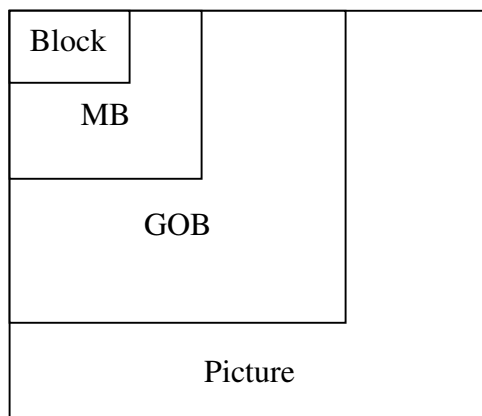
H.263 supports five resolutions. In addition to QCIF and CIF that were supported by H.261, there is SQCIF, 4CIF, and 16CIF. SQCIF is approximately half the resolution of QCIF. 4CIF and 16CIF are 4 and 16 times the resolution of CIF respectively. The support of 4CIF and 16CIF means the codec could then compete with other higher bit rate video coding standards such as the MPEG standards.

Picture format	Luminance pixels	Luminance lines	H.261 support	H.263 support	Uncompressed bit rate (Mbit/s)			
					10 frames/s		30 frames/s	
					Grey	Color	Grey	Color
SQCIF	128	96		Yes	1.0	1.5	3.0	4.4
QCIF	176	144	Yes	Yes	2.0	3.0	6.1	9.1
CIF	352	288	Optional	Optional	8.1	12.2	24.3	36.5
4CIF	704	576		Optional	32.4	48.7	97.3	146.0
16CIF	1408	1152		Optional	129.8	194.6	389.3	583.9

**Table 1 Picture Formats Supported by H.263**

In video coding, typically, an entire picture is not encoded at once. Instead, it is divided into blocks that are processed one by one, both by the encoder and by the decoder, most often in a raster scan order. This approach is often referred to as block-based coding.

In H.263, a block is defined as an 8x8 group of samples. Because of the downsampling in the chrominance components as mentioned earlier, one block of  $C_R$  samples correspond to four blocks of Y samples. The collection of these six blocks is called a *macroblock*. An MB is treated as one unit in the coding process. A number of MBs are grouped together and called a *group of blocks* (GOB).



**Figure 5 Components of an intra frame**

A frame of picture that is composed of GOBs is called an *intra frame* (I-Frame).

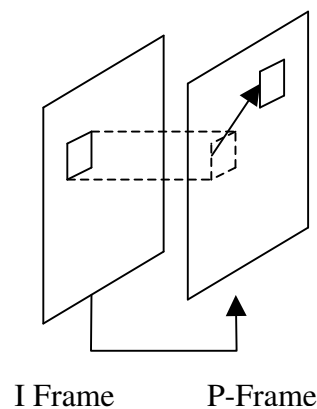
### **2.3.1. Fundamentals of Video Compression**

Compression of video data is typically based on two principles: reduction of spatial redundancy and reduction of temporal redundancy. H.263 uses a discrete cosine transform to remove spatial redundancy and motion compensation to remove temporal redundancy.

The *discrete cosine transform* (DCT) is a frequency transform similar to the *discrete Fourier transform* (DFT), but using only real numbers. It is equivalent to a DFT of roughly twice the length, operating on real data with *even* symmetry (since the Fourier transform of a real and even function is real and even), where in some variants the input and/or output data are shifted by half a sample.

Discrete cosine transform is applied to each block of an intra frame. On the other hand, a frame can be predicted from a previously coded intra frame. Using motion compensation methods, a displacement vector is calculated for each block, and a new frame can be generated depending on the previous frame. This kind of frame is called a P-Frame.

The following figure represents the relations between an I-Frame and a P-Frame.



**Figure 6 Relations between I-Frame and P-Frame**

There are four optional negotiable options included to improve performance: Unrestricted Motion Vectors, Syntax-based arithmetic coding, Advanced prediction, and forward and backward frame prediction similar to MPEG called P-B frames.

In the thesis work, encoder is not using the mentioned optional modes.



## 2.4. Bluetooth

Bluetooth technology eliminates the need for numerous and inconvenient cable attachments for connecting computers, mobile phones, mobile computers and handheld devices. Moreover, it enables connections between devices, which were not communicating before (i.e. refrigerator and microwave oven)

A Bluetooth chip is built into digital devices. By the help of this chip, all connections are instant and wireless. It's fast and secure for both voice and data transmission.

All Bluetooth enabled devices communicate simultaneously to offer three major advantages:

1. Voice/data access point

Access to other networks are made easier. Bluetooth is a common medium for connecting different nets. (i.e. Mobile phones can use Bluetooth connection to a PC which is hardwired to the Internet)

2. Cable replacement

Troublesome cable attachments are eliminated.(i.e. Mobile computers can send e-mail via mobile phones even when the devices are not within line of sight)

3. Personal ad-hoc nets

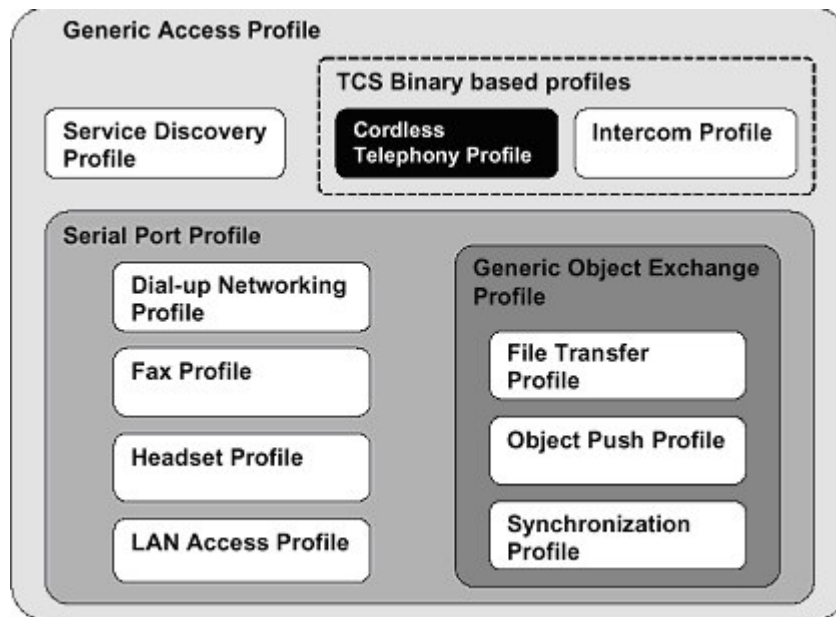
All Bluetooth enabled devices can be setup to automatically exchange information and synchronize with one another.(i.e. Appointments arranged on mobile devices while you are traveling are automatically accepted in your desktop PC as soon as the two devices are within range of each other)

The Bluetooth technology is designed to be fully functional even in very noisy radio environments, and its voice transmissions are audible under severe conditions. The technology provides a very high transmission rate and all data are protected by advanced error correction methods, as well as encryption and authentication routines for the user's privacy.

### 2.4.1. Bluetooth Profiles [13]

Bluetooth uses profiles to ensure interoperability between products and brands.

In Figure 7, the Bluetooth profile structure and the dependencies of the profiles are depicted. A profile is dependent upon another profile, if it reuses parts of that profile, by implicitly or explicitly referencing it. Dependency is also illustrated in the same figure. A profile has dependencies on the profile(s) in which it is contained directly and indirectly.



**Figure 7 Bluetooth Profiles [13]**

Among the many other profiles, LAN Access Profile is the only one necessary to be mentioned in this background section. Detailed information related to others can be found in the reference documents.

#### 2.4.1.1. LAN Access Profile

This profile defines LAN Access using PPP over RFCOMM.

1. PPP is a widely deployed means of allowing access to networks.

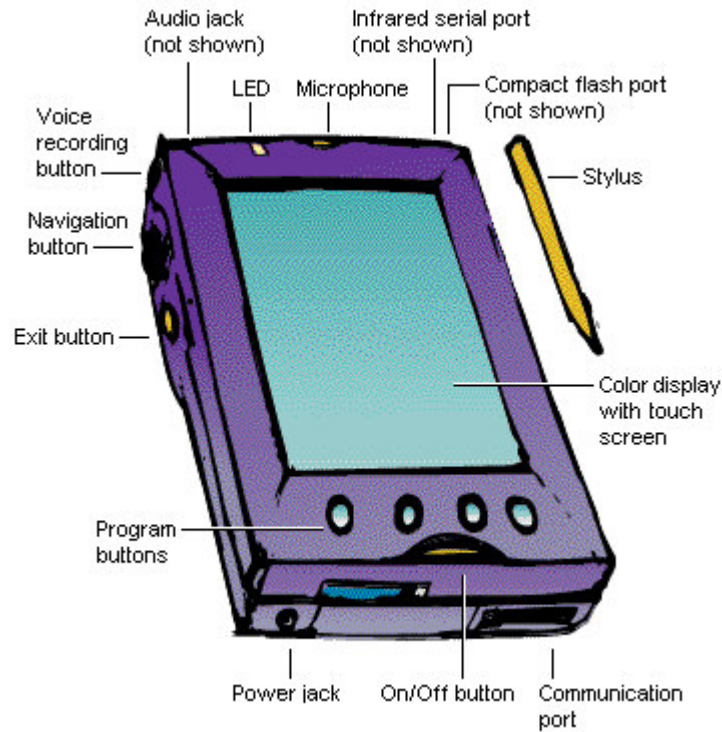
2. PPP provides authentication, encryption, data compression and multi-protocol facilities. PPP over RFCOMM has been chosen as a means of providing LAN Access for Bluetooth devices because of the large installed base of devices equipped with PPP software.
3. PPP is capable of supporting various networking protocols (e.g. IP, IPX, etc.). This profile does not mandate the use of any particular protocol.
4. This profile does not deal with conferencing, LAN emulation, ad-hoc networking or any other means of providing LAN Access.

This profile defines how PPP networking is supported for the following situations.

1. LAN Access for a single Bluetooth device.
2. LAN Access for multiple Bluetooth devices.
3. PC to PC (using PPP networking over serial cable emulation).

## **2.5. Microsoft Windows CE 3.0 for PocketPC 2002 [14]**

The Pocket PC is a personal companion for mobile device users. The following figure shows possible hardware of a PocketPC device.



**Figure 8 PocketPC Hardware [14]**

Apart from the illustrated hardware functionality, some devices employ built-in or plug-in wireless network interfaces, such as Bluetooth or IEEE 802.11 (wi-fi) secure digital cards.

Microsoft Windows CE™ is an embedded operating system designed for PocketPC devices. Similar to the PC operating system Microsoft Windows™, Microsoft Windows CE™ provides a windows graphical user interface, kernel functionality, multi-threaded application framework and TCP/IP protocol stack.

Some versions of the PocketPC device contains embedded Java virtual machine that is required for our system.

In the thesis work, a PocketPC with an integrated Bluetooth chip has been used. Using embedded visual tools of Microsoft systems, a DLL has been developed in order to implement the native decoder.

## 2.6. Unified Modeling Language [15]

Large enterprise applications must be more than just a bunch of code modules. They must be structured in a way that enables scalability, security, and robust execution under stressful conditions, and their structure - frequently referred to as their architecture - must be defined clearly. Of course a well-designed architecture benefits any program, and not just the largest ones. Another benefit of structure is that it enables code reuse: Design time is the easiest time to structure an application as a collection of self-contained modules or components. Eventually, enterprises build up a library of models of components, each one representing an implementation stored in a library of code modules. When another application needs the same functionality, the designer can quickly import its module from the library. At coding time, the developer can just as quickly import the code module into the executable.

Modeling is the designing of software applications before coding. Modeling is an essential part of large software projects, and helpful to medium and even small projects as well. A model plays the analogous role in software development that blueprints and other plans (site maps, elevations, physical models) play in the building of a skyscraper. Using a model, those responsible for a software development project's success can assure themselves that business functionality is complete and correct, end-user needs are met, and program design supports requirements for scalability, robustness, security, extendibility, and other characteristics, before implementation in code renders changes difficult and expensive to make. Surveys show that large software projects have a huge probability of failure - in fact, it's more likely that a large software application will fail to meet all of its requirements on time and on budget than that it will succeed. If you're running one of these projects, you need to do all you can to increase the odds for success, and modeling is the only way to visualize your design and check it against requirements before coding.

Unified Modeling Language™ (UML™) [15] helps to specify, visualize, and document models of software systems, including their structure and design, in a way that meets all of these requirements. UML can be used for business modeling and modeling of other non-software systems too.

One can model just about any type of application, running on any type and combination of hardware, operating system, programming language, and network, by using UML. Its flexibility lets one model distributed applications that use just about any middleware. It's a natural fit for object-oriented languages and environments such as C++, Java, and the recent C#, but one can use it to model non-object oriented applications as well in, for example, Fortran, VB, or COBOL. UML Profiles (that is, subsets of UML tailored for specific purposes) help one model transactional, real-time, and fault-tolerant systems in a natural way.

One can do other useful things with UML as well: For example, some tools analyze existing source code and reverse-engineer it into a set of UML diagrams. Another example: In spite of UML's focus on design rather than execution, some tools on the market execute UML models, typically in one of two ways: Some tools execute your model interpretively in a way that lets you confirm that it really does what you want, but without the scalability and speed that you'll need in your deployed application. Other tools (typically designed to work only within a restricted application domain such as telecommunications or finance) generate program language code from UML, producing most of a bug-free, deployable application that runs quickly if the code generator incorporates best-practice scalable patterns for, e.g., transactional database operations or other common program tasks.

The process of gathering and analyzing an application's requirements, and incorporating them into a program design, is complex one. The industry currently supports many methodologies that define formal procedures specifying how to manage it. One characteristic of UML - in fact, the one that enables the widespread industry support that the language enjoys - is that it is methodology-independent. Regardless of the methodology that you use to perform your analysis and design, you can use UML to express the results. And, using XMI (XML Metadata Interchange), you can transfer your UML model from one tool into a repository, or into another tool for refinement or the next step in your chosen development process.

UML defines 12 types of diagrams, divided into 3 categories: 4 diagram types represent static application structure; 5 represent different aspects of dynamic behavior; and 3 represent ways you can organize and manage your application modules.

Structural Diagrams include the Class Diagram, Object Diagram, Component Diagram, and Deployment Diagram.

Behavior Diagrams include the Use Case Diagram (used by some methodologies during requirements gathering), Sequence Diagram, Activity Diagram, Collaboration Diagram, and Statechart Diagram.

Model Management Diagrams include Packages, Subsystems, and Models.

Among the mentioned UML diagrams, class diagrams, use-case diagrams, sequence diagrams and packages have been used in this thesis work.

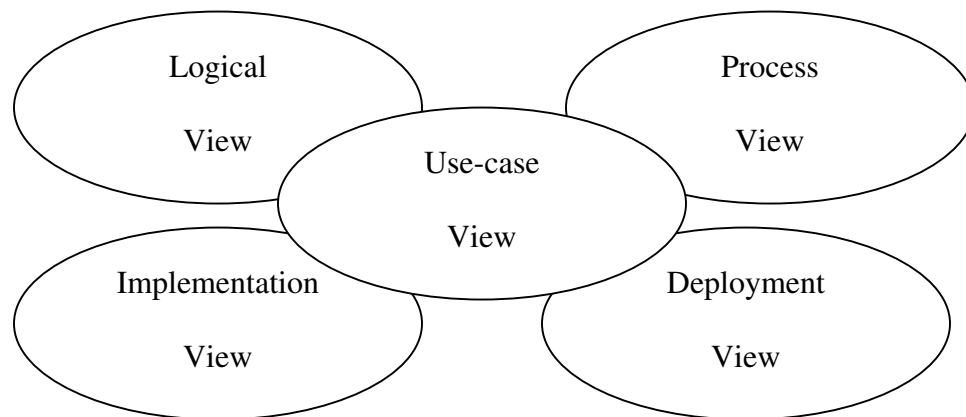
## CHAPTER III

### ARCHITECTURE

#### 3.1. Software Architecture

The system is modeled in Unified Modeling Language (UML) [15]. The diagrams shown in this chapter are UML diagrams that define the architecture. All of the system is developed according to Object Oriented Analysis and Design techniques.

The software architecture is defined by five main views of the system.



**Figure 9 Architectural Views**

The architectural views define a software system from different perspectives. Following is a short description of the architectural views.

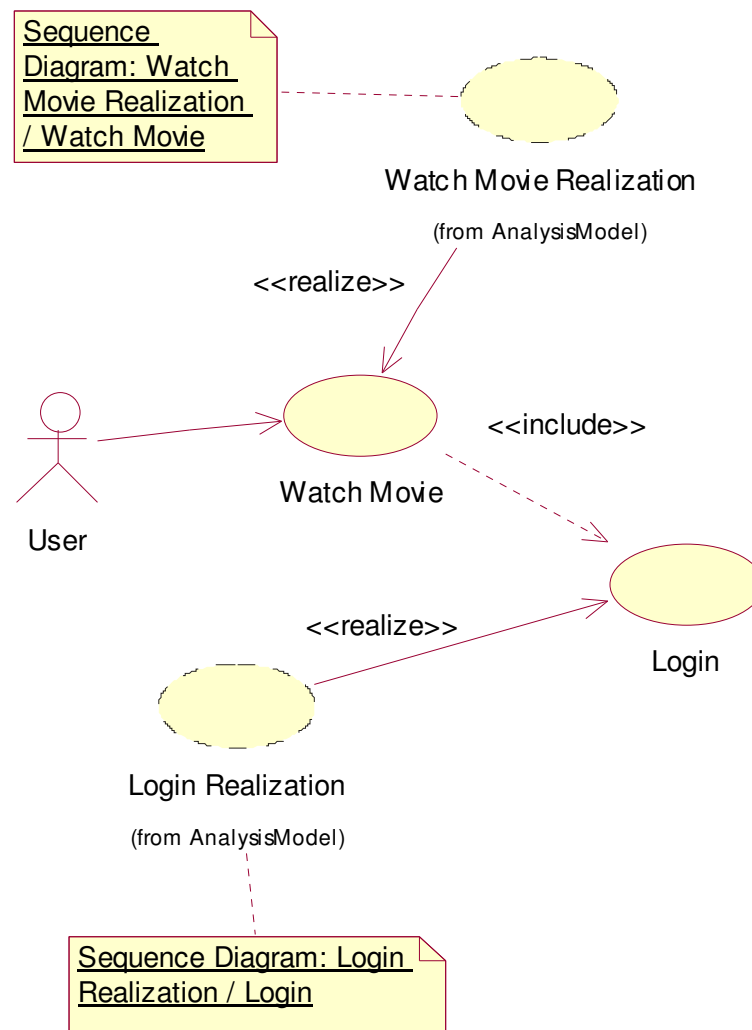
Use-case View	Defines the system from the user perspective. Identifies the scenarios, external actors and their relations.
Logical View	Defines the system from the logical class organization perspective. The classes, their relations, sub-system



	perspective. The classes, their relations, sub-system organizations and dependency relations are shown in this view.
Process View	Displays the run-time organization of the system. Processes, threads, their relations are shown.
Deployment View	Displays the deployment of software modules on to the hardware, connections and device configurations are shown.
Implementation View	Displays how logical entities defined in logical view are mapped onto the real world physical components.

**Table 2 Descriptions of Architectural Views**

The following is the use-case of the system. The User is the only actor that is interacting with the system. User can login and watch a movie.



**Figure 10 Use-case View Of The System Architecture**

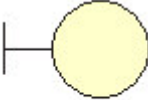


During analysis a number of analysis classes are found. Analysis classes are divided into three groups and they are stereotyped according to their group: Boundary class, Control class and Entity class.

Boundary classes interfaces with actors, implements the protocols and GUI's defined for a particular actor.

Control classes implements the business logic of the use-cases.

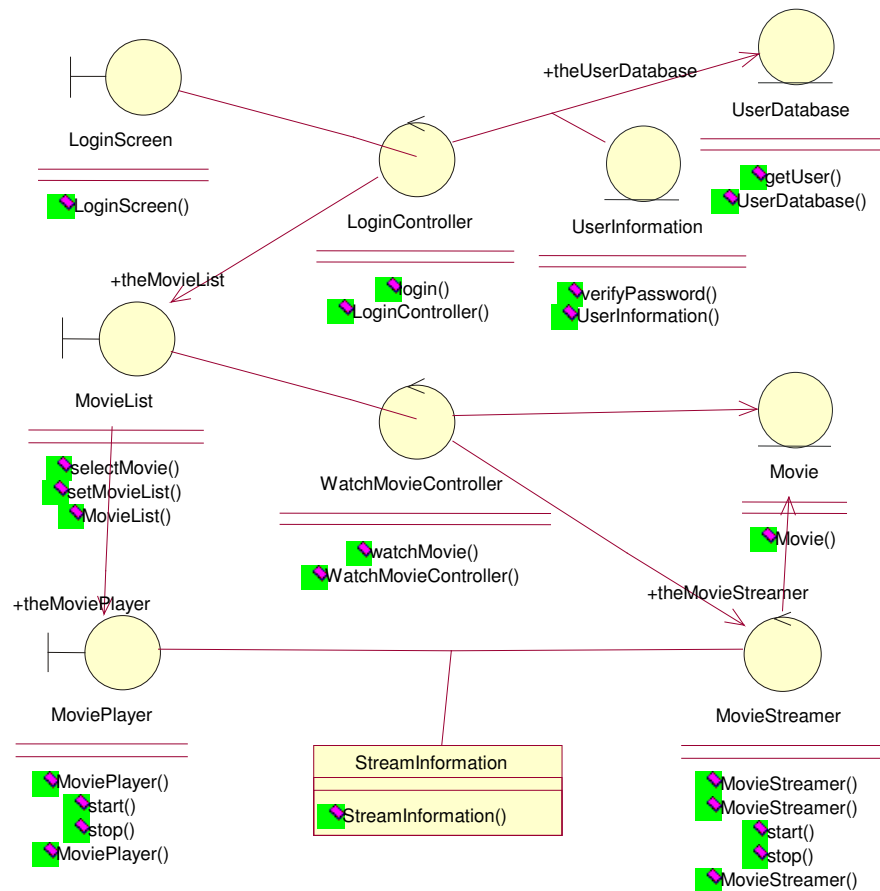
Entity classes are used to store persistent information handled by the system.

The following icons are used to describe analysis classes.

Boundary Class	 Boundary
Control Class	 Control
Entity Class	 Entity

**Table 3 Icons for Analysis Classes**

The following diagram represents the relations of analysis classes and responsibilities assigned to the analysis classes.



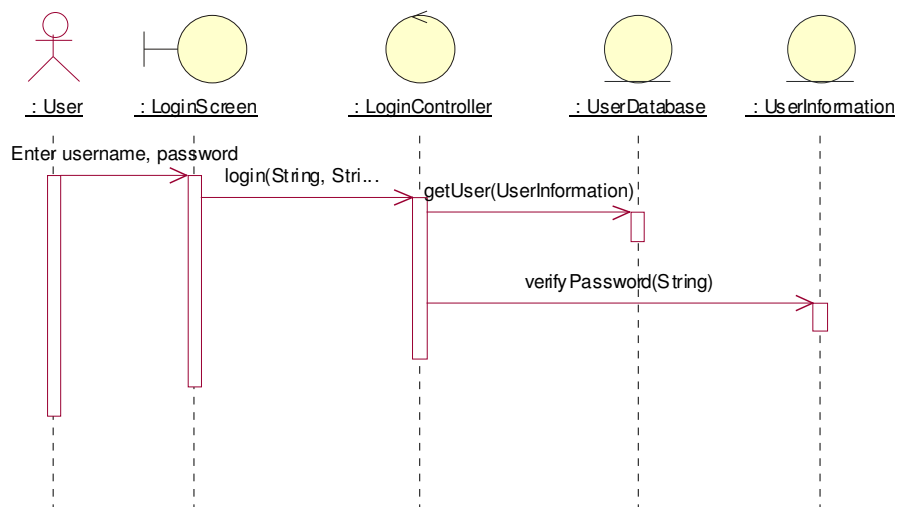
**Figure 11 Analysis Model Class Diagram**

Let's give brief explanation of the analysis classes.

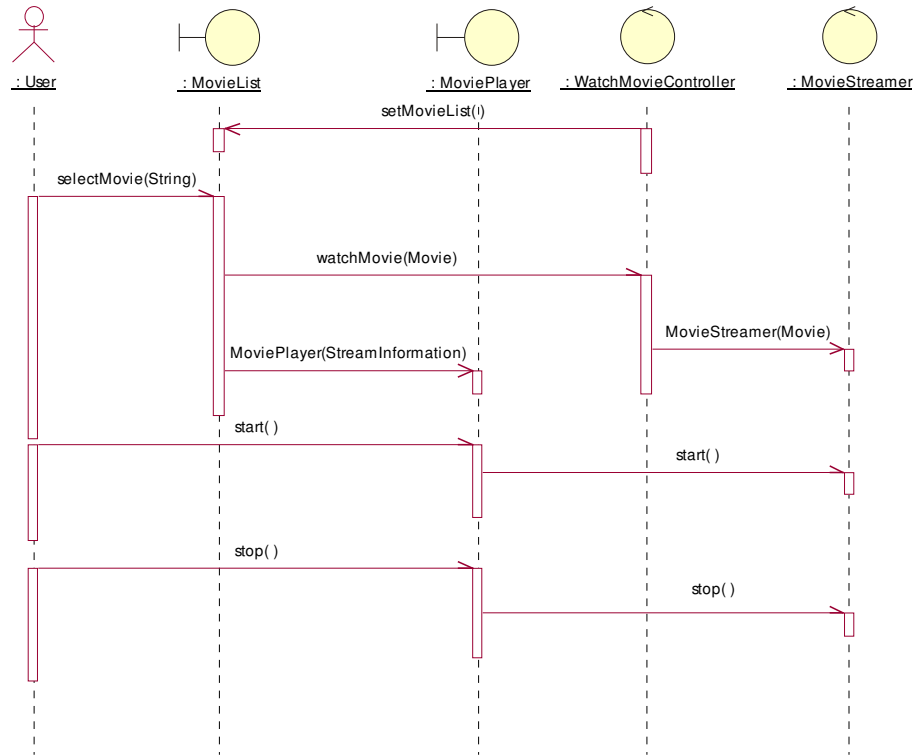
1. Login Screen: Inputs username, password and server address.
2. LoginController: Controls the login use-case realization. Simply checks the username and password provided with the stored information.
3. UserDatabase: Maintains user information for all the users in the system.
4. UserInformation: Handles information for a single user.
5. MovieList: Displays a list of available movies in the server and helps user select a movie.

6. WatchMovieController: Controls the watch movie use-case realization. Gathers the information of all movies and directs user commands.
7. Movie: Handles the information of single movie.
8. MoviePlayer: Plays the movie for the user. Inputs user commands (start/stop) for the movie and passes the commands to the associated MovieStreamer.
9. MovieStreamer: Streams the movie to the client. Accepts commands from the MoviePlayer. Coordinates the streaming operation with MoviePlayer.
10. StreamInformation: Models the information of a single stream. This information is passed between MovieStreamer and MoviePlayer.

After the analysis of use-cases, use-case realizations are obtained. Login use-case is realized by LoginRealization and Watch Movie is realized by WatchMovieRealization. Following sequence diagrams describe how login and watch movie use-cases are realized in the system.

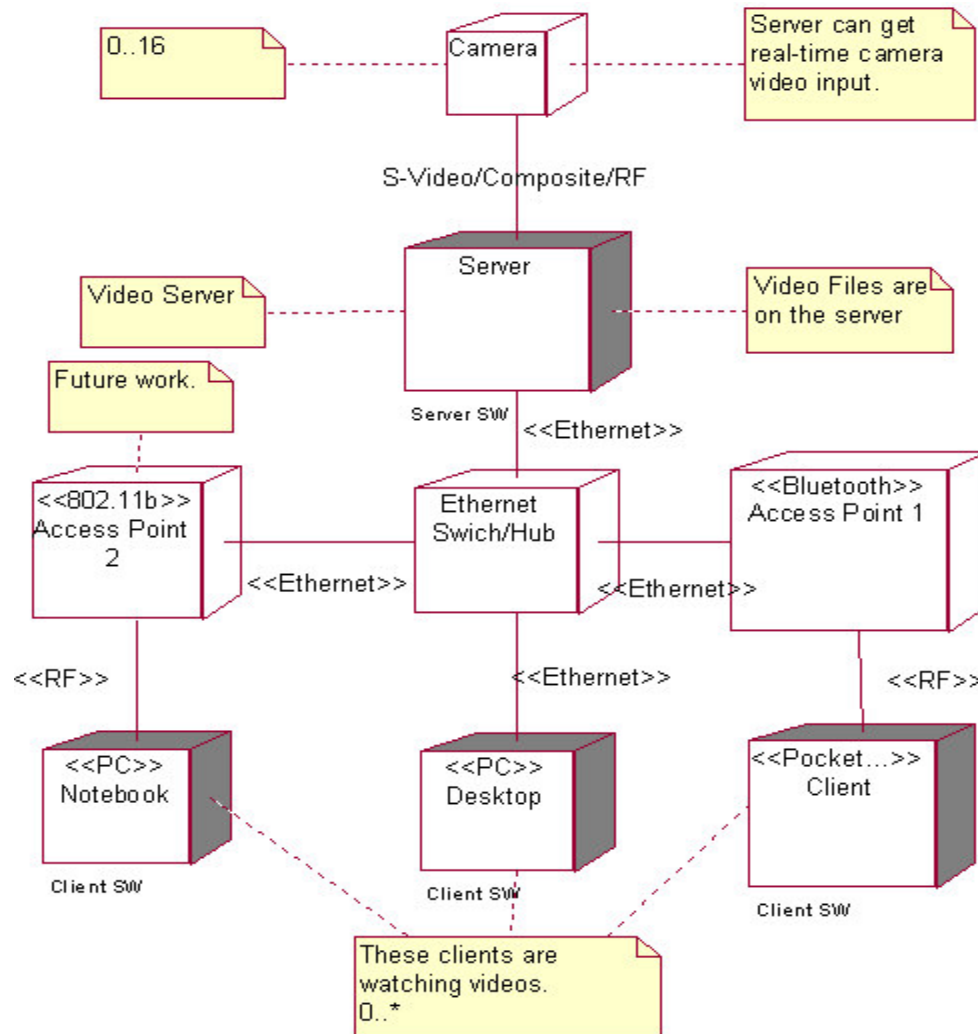


**Figure 12 Login Realization Sequence Diagram**



**Figure 13 Watch Movie Realization Sequence Diagram**

Figure 14 illustrates the Deployment View of system architecture.



**Figure 14 Deployment View Of The Architecture**

The system has a client/server architecture. Server node runs the server process and client process is distributed over different nodes.

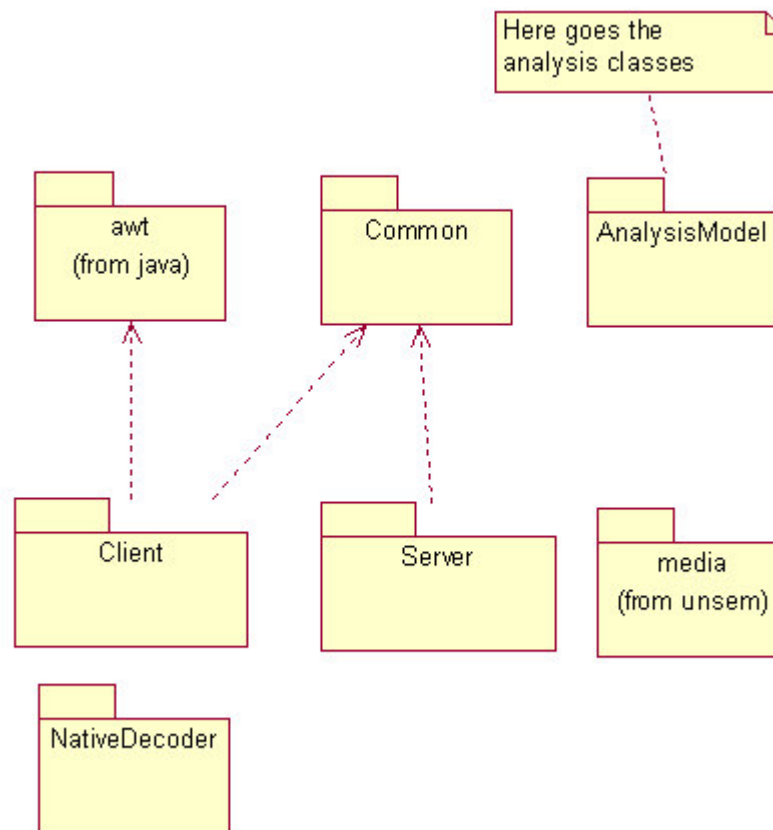
Desktop and notebook nodes have Intel x86 CPU's whereas PocketPC has Intel SA1110 CPU. The model of PocketPC is Compaq iPaq 3870. Notebook and desktop can run Windows 98, Windows 2000 or Linux. PocketPC runs Windows CE operating system. Thus, the client software is required to run on different CPU architectures and different operating systems. In order to avoid developing different client programs and then porting the code, we have selected Java™ [16] as the development language. Java™ (write once run anywhere) eliminates the porting of client code.

The communication between client and server is carried out over Remote Method Invocation (RMI) [16]. This service is provided by the Java platform.

The video is encoded in H.263 format and RTP is used for streaming media from the server to the client.

The PocketPC has limited processing power and it is not sufficient to efficiently decode H263 coded video streams in real-time. Thus, a native, custom, efficient H263 decoder plug-in for JMF has been implemented in thesis study. The plug-in had been developed as a Win32 DLL in Embedded Visual C++ and it's been attached using Java Native Interface (JNI) [16].

The following figure represents the logical package architecture of the system.



**Figure 15 Logical View of the Architecture**

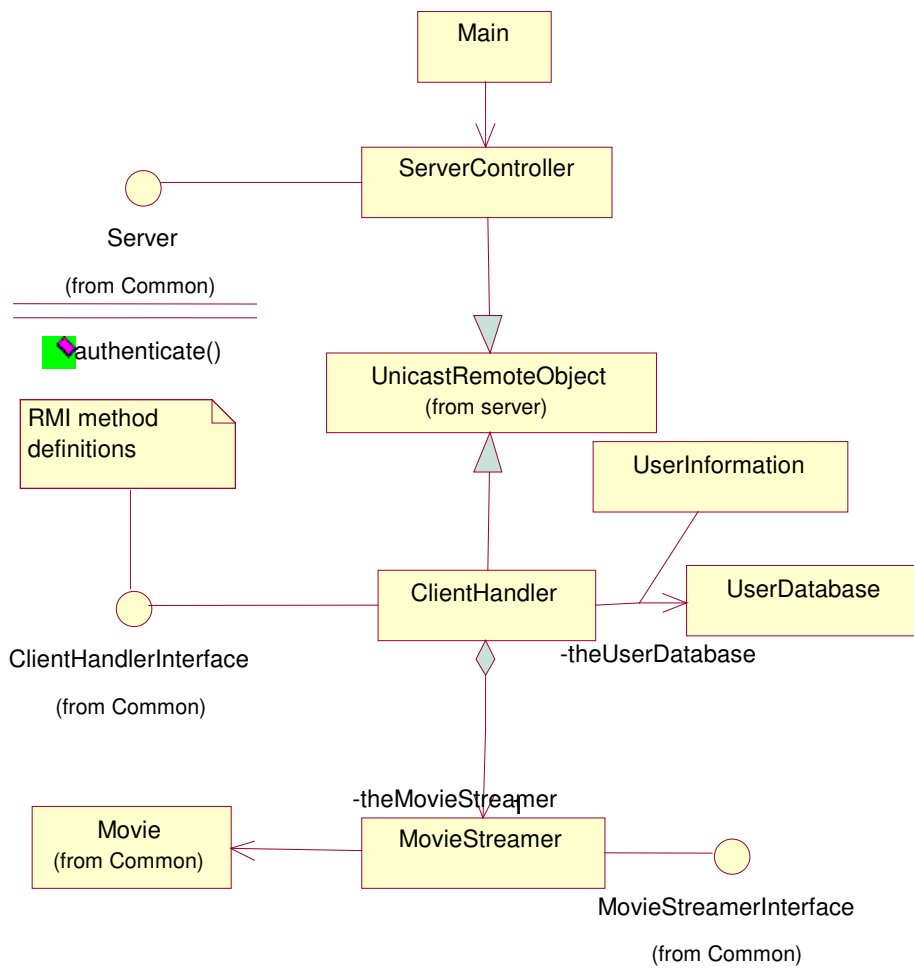
The following table describes the packages.



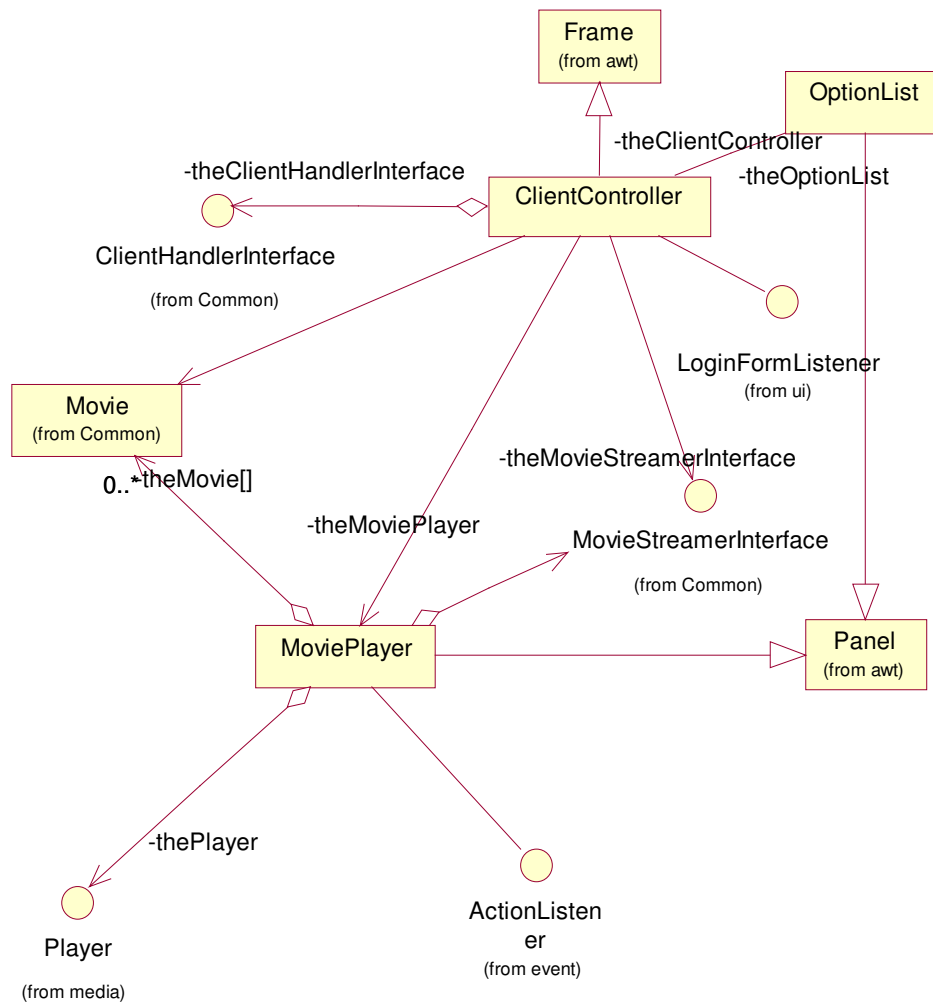
<b>Package</b>	<b>Description</b>
AWT	The user interface classes of Java.
Common	Components common to both Client and Server.
Client	Contains the client components
Server	Contains the server components
Media	Common media processing components
NativeDecoder	Native C++ implementation of custom Renderer plug-in
Analysis Model	Contains analysis classes

**Table 4 Logical Package Descriptions**

Following diagrams express what is available in each logical package.



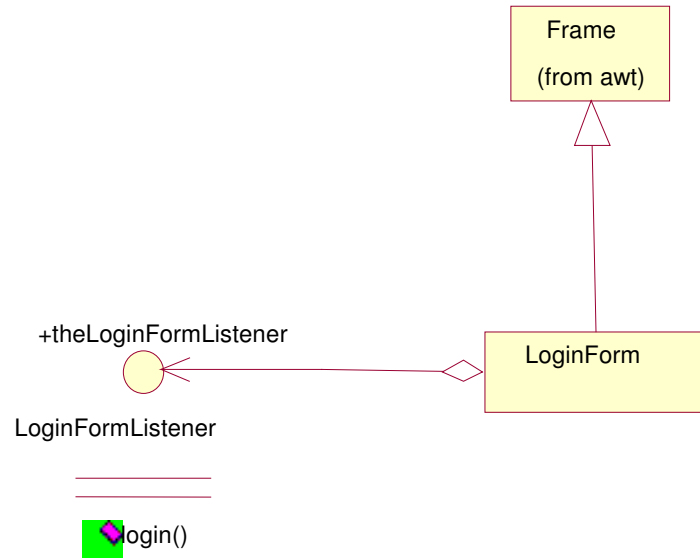
**Figure 16 Server Package Contents**



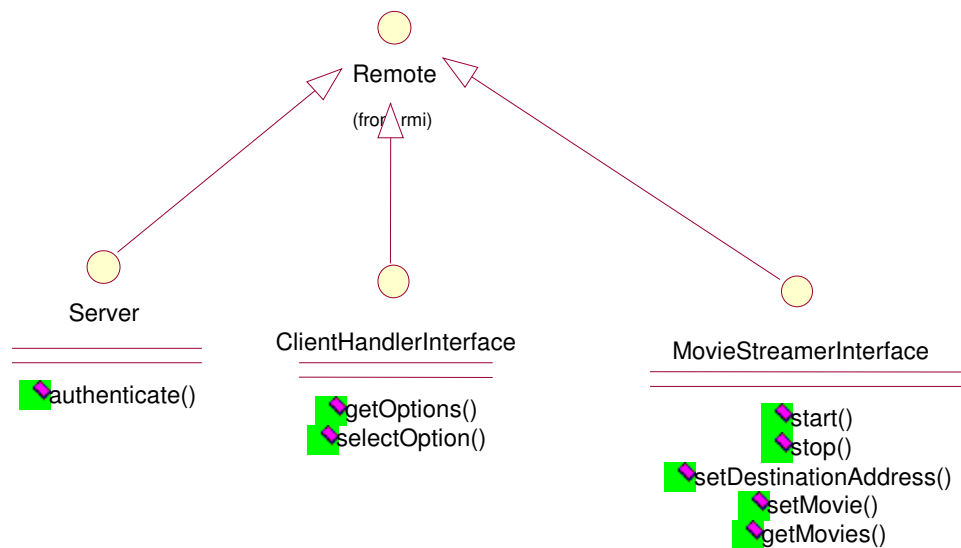
**Figure 17 Client Package Contents**

Common package consists one sub-package UI.

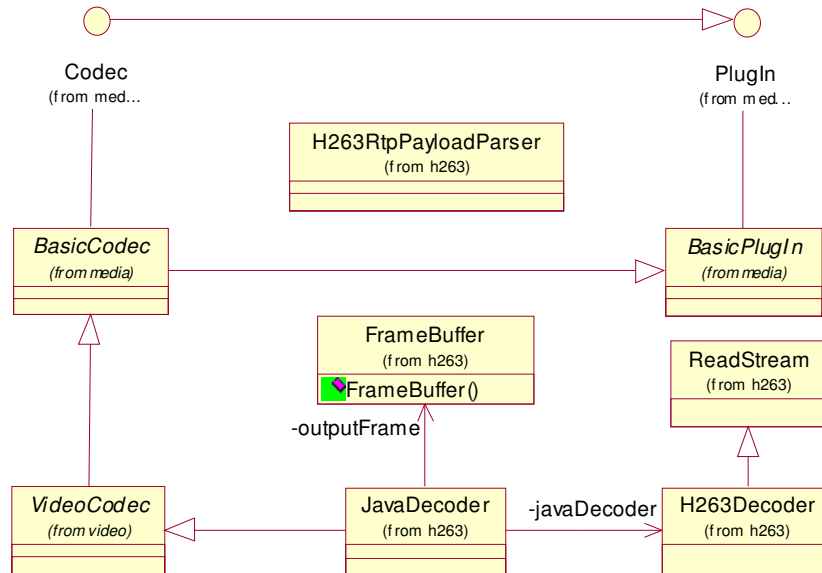
This package contains reusable GUI components.



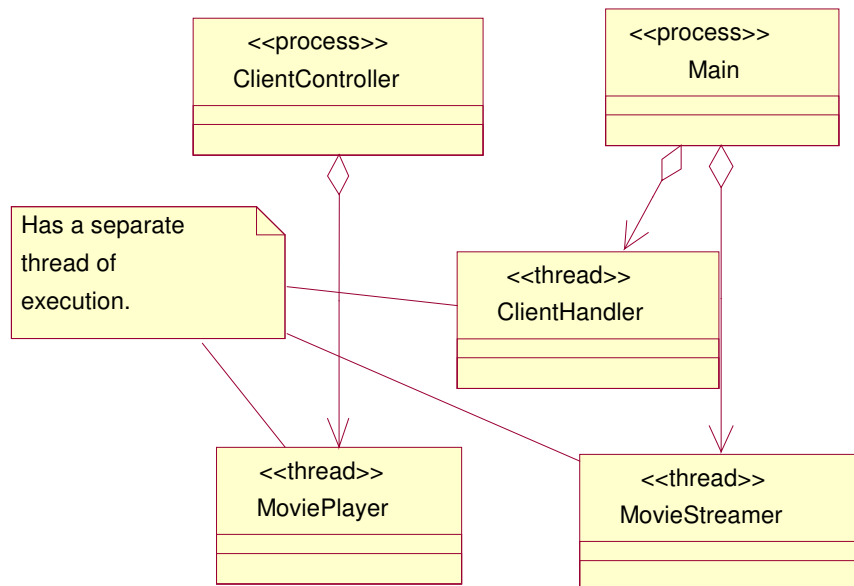
**Figure 18 UI Package Contents**



**Figure 19 Common Package Classes**



**Figure 20 Native Decoder Contents**

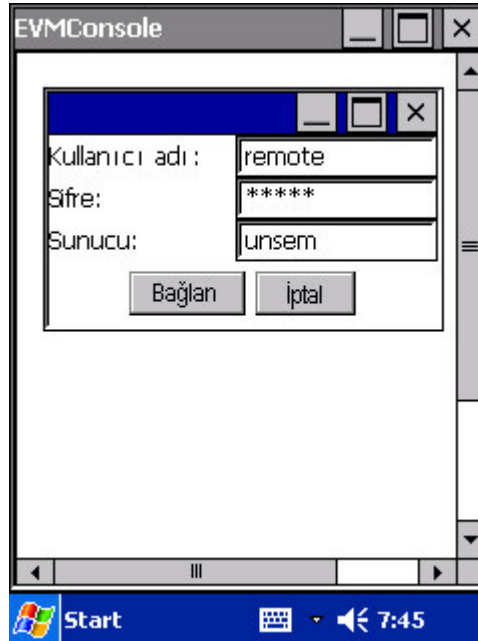


**Figure 21 Process View Of The System Architecture**

The above diagram describes the process view of the system. Each client runs a ClientController which is a process. When the user requests a movie to be played, a MediaPlayer is created. MediaPlayer has its own thread of execution. On the other hand, a Main process runs on the server. For each connected ClientController a separate ClientHandler is created. ClientHandler processes the requests of associated ClientController. Also, a MovieStreamer implements the server side of movie streaming. A separate MovieStreamer is created for each movie request of client. MovieStreamer coordinates the streaming operation with the associated MediaPlayer.

### 3.2. User-Interface Samples

The following figures represent the output of boundary classes.



**Figure 22 GUI Of LoginForm**

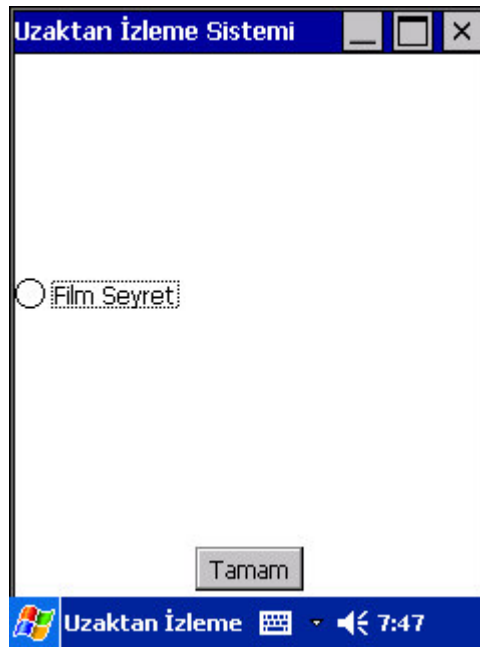


Figure 23 GUI Of OptionList

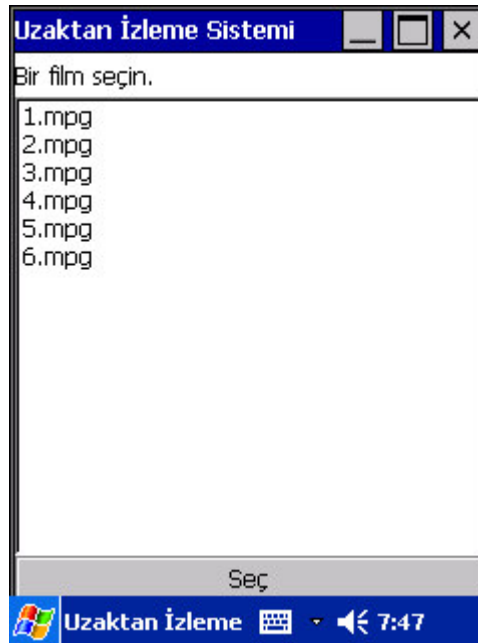


Figure 24 GUI Of MovieList

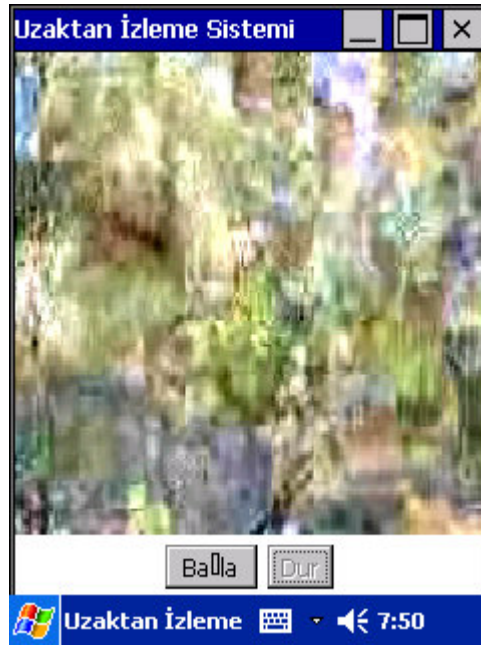


Figure 25 GUI Of MoviePlayer 1

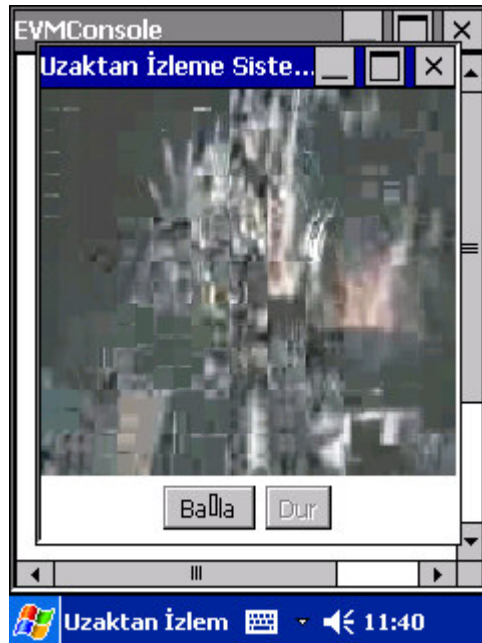
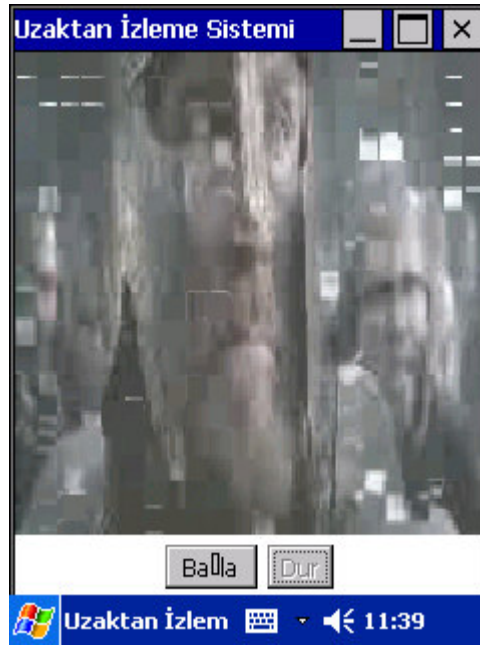


Figure 26 GUI Of MoviePlayer 2





**Figure 27 GUI Of MoviePlayer 3**

## CHAPTER IV

### EXPERIMENTS

In order to find out the system parameters that yield most satisfactory system performance, a number of experiments have been done. In those experiments, the difference of frame rates and the variance of the difference between an ideal decoder and our decoder are measured.

#### 4.1. System Configuration

The following figure represents a simplified system block diagram. It excludes the software complexity, and focuses on video data flow in the system.



**Figure 28 Simplified Block Diagram**

Video source is the input to the encoder. Video source can be a real-time source, i.e. web-cam, or stored video. It can also be raw or previously encoded media. From the experiment point of view, the only parameter of the video source is its frame rate, because the encoder cannot change the frame rate. The frame rate of the video source is denoted as source rate in the remaining parts of this thesis.

H263/RTP Encoder converts the input media to H263 coded media and also it's been multiplexed into the RTP stream. Encoder has an intra frame rate parameter that denotes the rate of I frames in the stream.

Bluetooth network carries the data between the encoder and the decoder. It not only consists of Bluetooth elements, but also Ethernet components. In fact, Bluetooth components can sustain a continuous throughput of 115 Kbps and Ethernet, at least, 10 Mbps.

H263/RTP Decoder converts the input bit stream into raw media. During these experiments, samples have been taken at this block of the system. The following parameters have been measured for each frame

1. Frame number: The order of frame in the stream.
2. Frame processing time: How long it takes to decode current frame.
3. Instantaneous frame rate: Reciprocal of frame processing time.

Renderer has no parameters, it just renders the input media to the screen.

## **4.2. Experiment Configurations**

The experiments have been run in order to find out the ideal source rate and intra frame rate.

The default parameters for all the experiments are as follows:

1. Video sequence duration is 60 seconds.
2. Source frame size is 288x176.
3. Encoder output frame size is 128x96.
4. Color depth of the source is 24 bits (16,7 Million different colors.)

The source is down sampled from 288x176 to 128x96 frame size in the encoder.

In order to run the experiments under controlled input source, a recorded video has been used. The source rate has been adjusted to 5, 10, 15, 20 and 25 fps.

Experiments are done in order that the experiments are controllable. Thus, all but one parameter is set to a default value and instantaneous frame rate versus frame number graphs of the decoder are obtained. Also, the variance of instantaneous frame rate and the difference between average instantaneous frame rate and source frame rate is calculated. Based on the variance and the difference, the most suitable intra frame rate and source rate are found. For an ideal decoder, both the variance and the difference between source rate and display rate shall be zero; the larger the difference, the slower shall the decoder be.

<b>Source Frame Rate</b>	<b>Output Bit rate Of Encoder</b>
5 FPS	12,288 bps
10 FPS	24,576 bps
15 FPS	36,863 bps
20 FPS	49,152 bps
25 FPS	61,440 bps

**Table 5 Output Bit rate Of Encoder vs Source Frame Rate**

The above table represents the bit rate at the encoder output. All the values obtained are less than the maximum bit rate available for the Bluetooth connection (115,200 bps). Thus, the network is not congested and we can assume that no packet loss shall occur due to network congestion.

### **4.3. Experiment Results**

The following are the results of the experiments. Each graph shows the displayed frame rate at the decoder output with constant source rate and changing intra frame rates (KFR: key frame rate). Each experiment is run five times and the graphs show the average values of the five experiments. The legend on the graphs identifies the intra frame rate, and the source rate is indicated on the graph title.

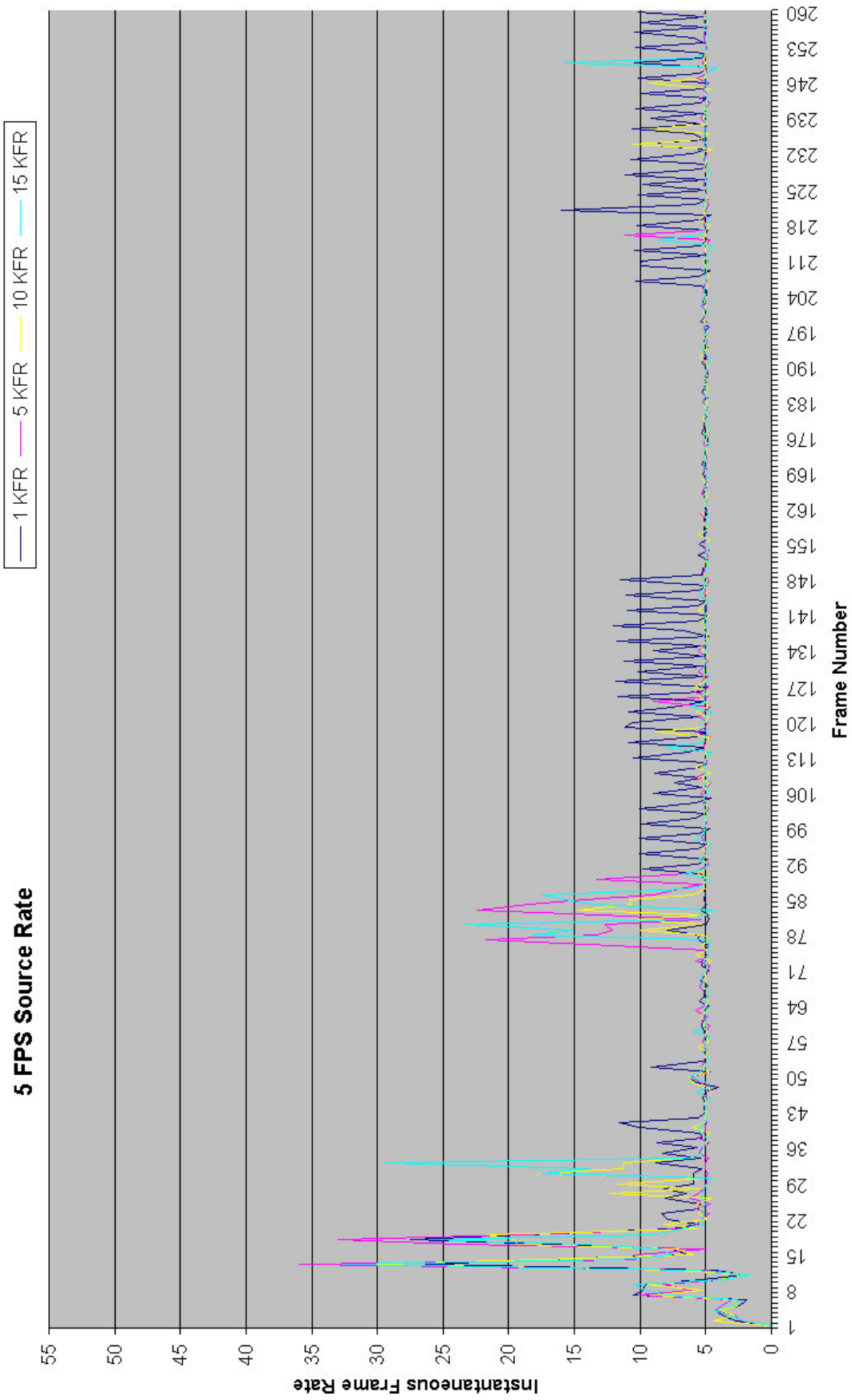
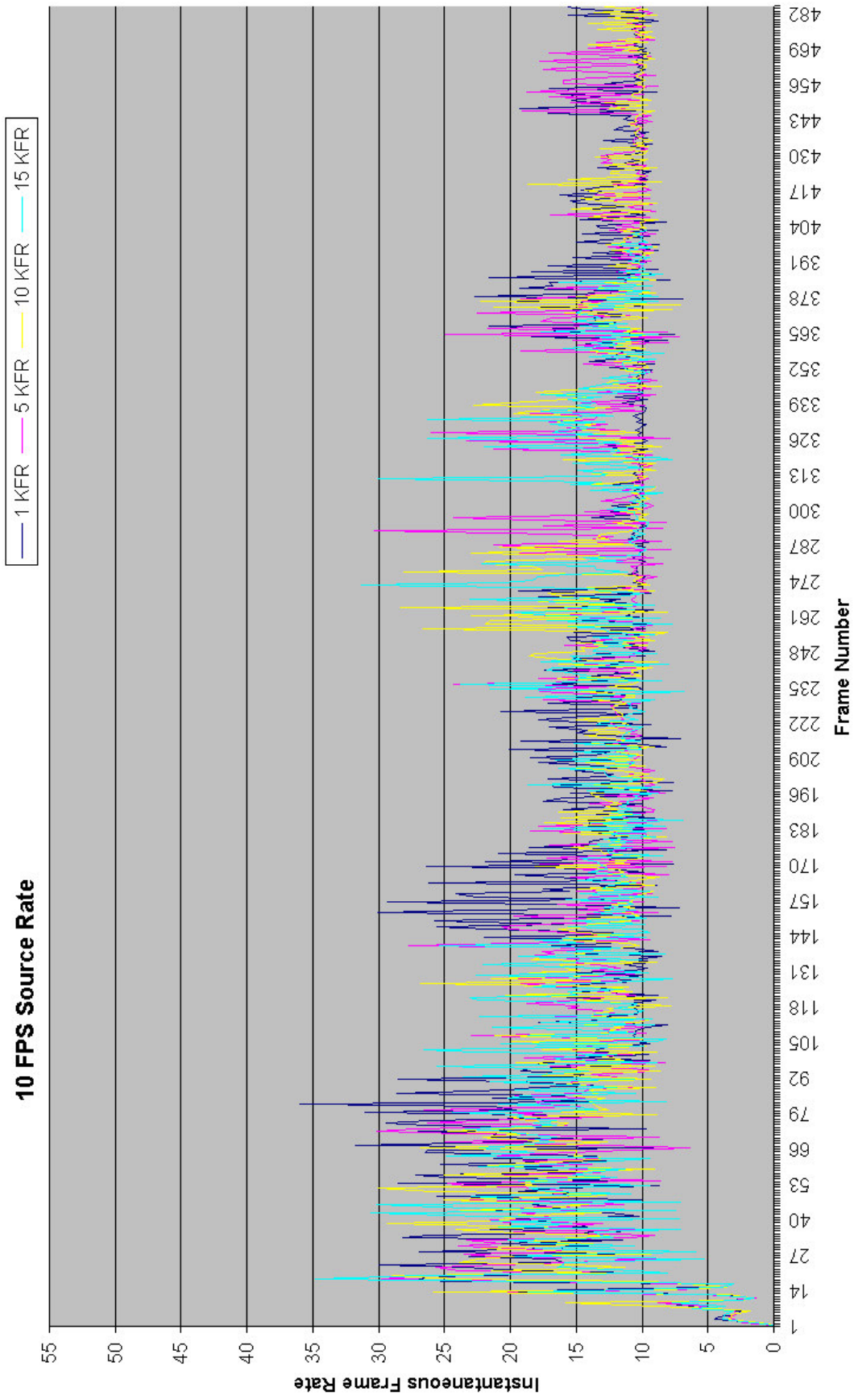


Figure 29 5 FPS Source Rate Experiment Results



**Figure 30 10 FPS Source Rate Experiment Results**

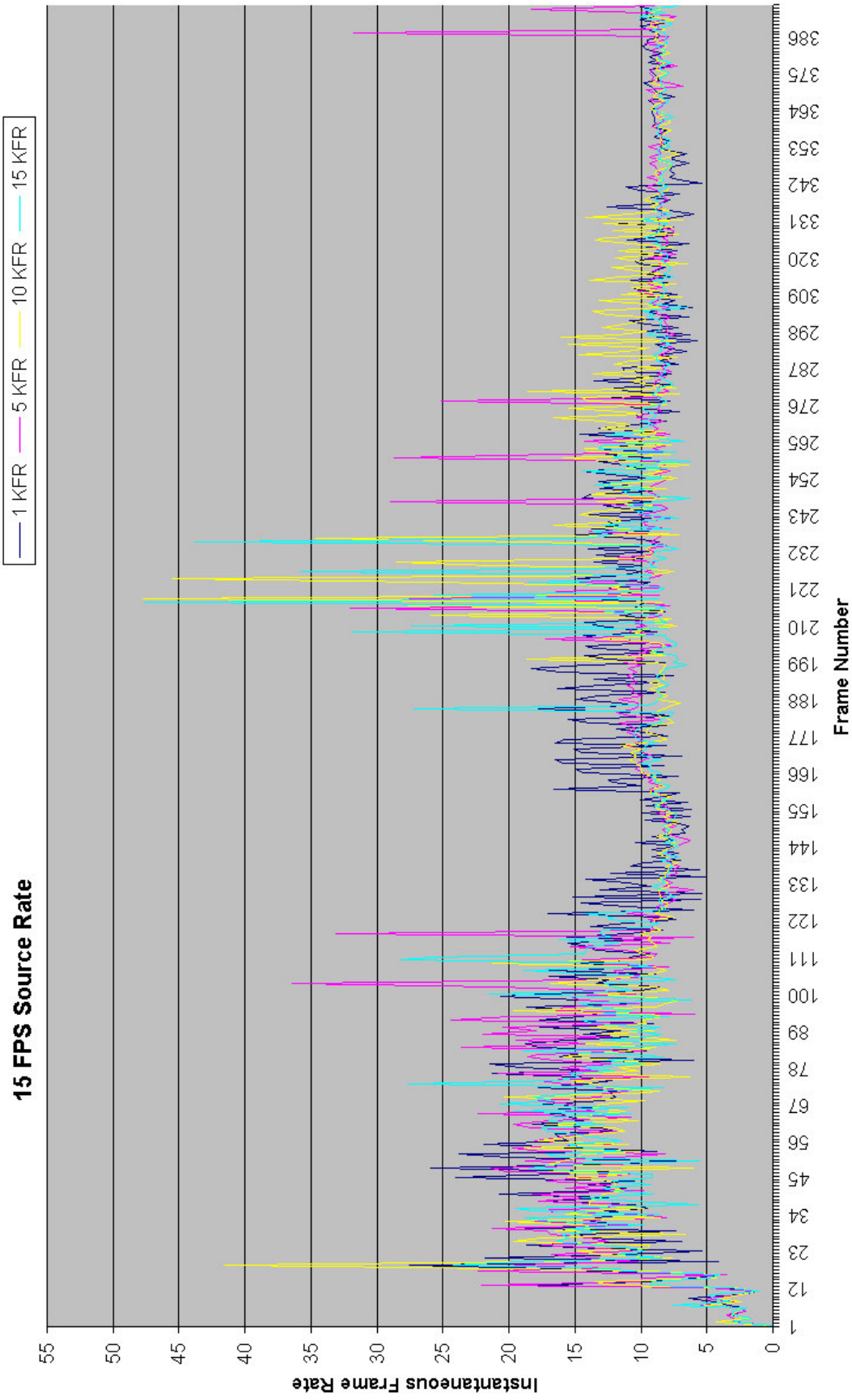


Figure 31 15 FPS Source Rate Experiment Results

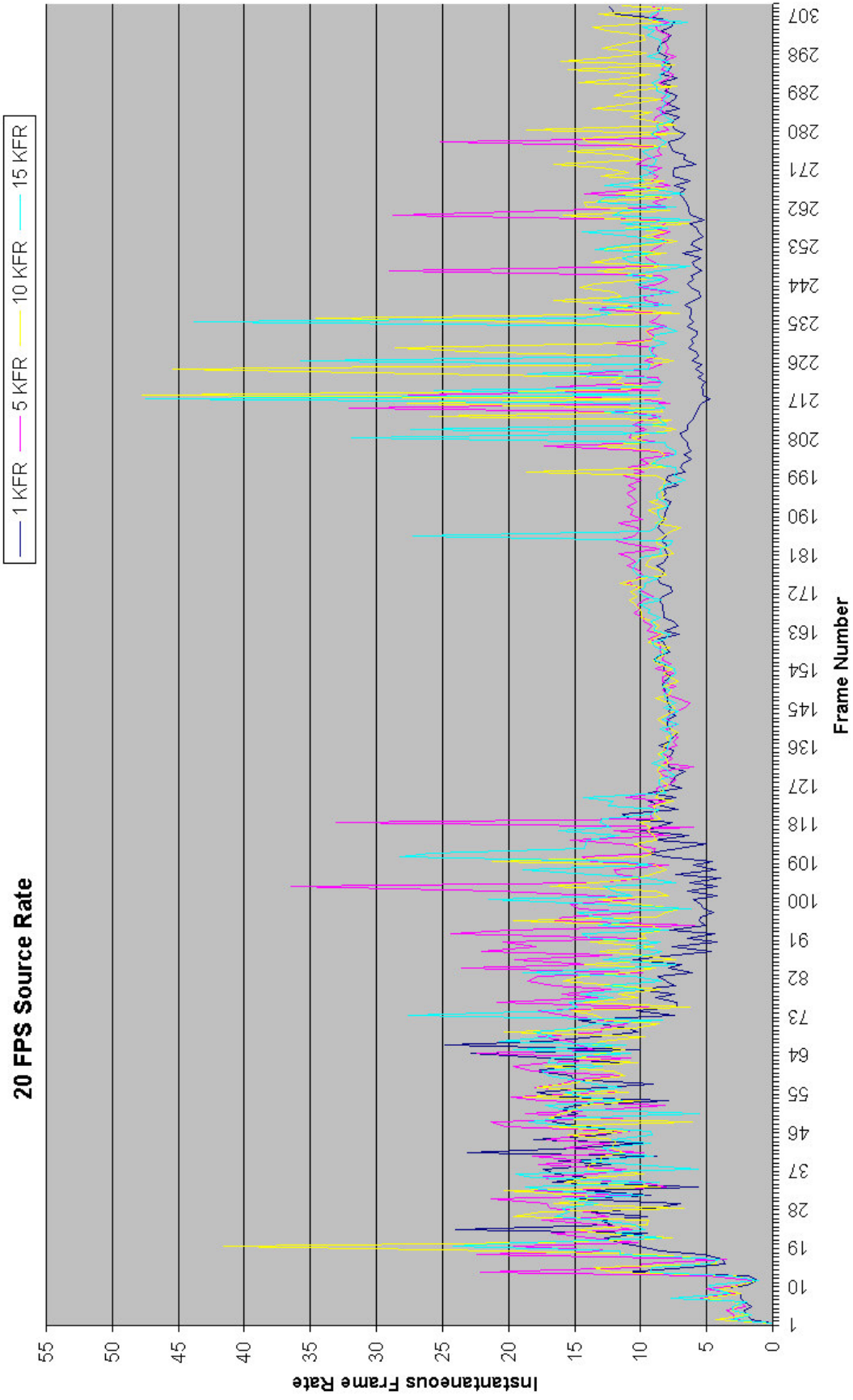
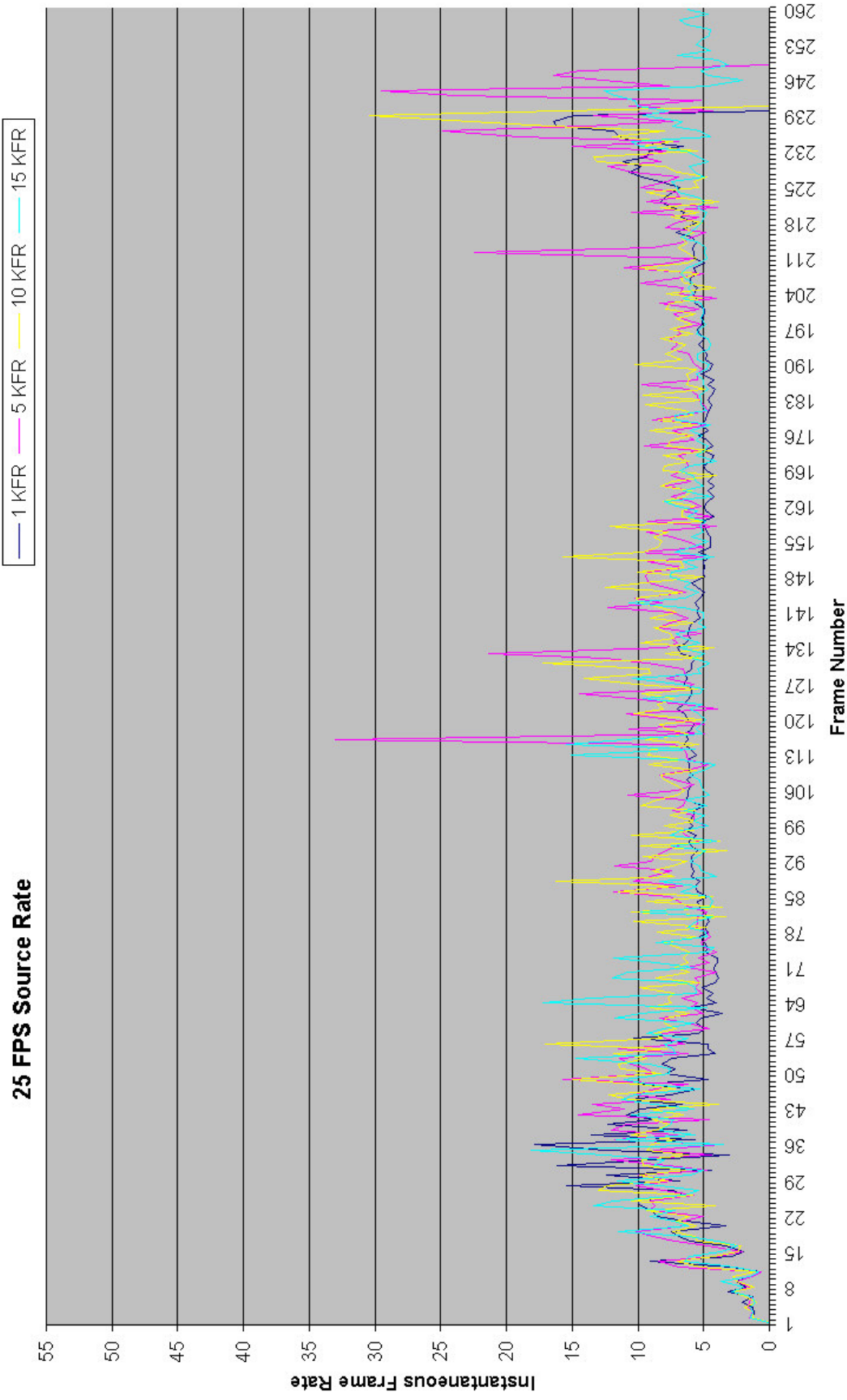


Figure 32 20 FPS Source Rate Experiment Results





**Figure 33 25 FPS Source Rate Experiment Results**

Key Frame Rate is the number of P-Frames between each I-Frame.

The following tabulates the resulting values that are obtained during the experiments.

		<b>Intra frame Rate</b>			
		<b>1 KFR</b>	<b>5 KFR</b>	<b>10 KFR</b>	<b>15 KFR</b>
<b>Source Frame Rate</b>	<b>5 FPS</b>	$\Delta \text{FPS} = 1,58$ $\sigma^2 = 9,69$	$\Delta \text{FPS} = 0,92$ $\sigma^2 = 13,61$	$\Delta \text{FPS} = 0,75$ $\sigma^2 = 8,09$	$\Delta \text{FPS} = 0,88$ $\sigma^2 = 12,47$
	<b>10 FPS</b>	$\Delta \text{FPS} = 3,41$ $\sigma^2 = 27,36$	$\Delta \text{FPS} = 3,00$ $\sigma^2 = 21,19$	$\Delta \text{FPS} = 3,01$ $\sigma^2 = 20,01$	$\Delta \text{FPS} = 3,47$ $\sigma^2 = 24,54$
	<b>15 FPS</b>	$\Delta \text{FPS} = 4,08$ $\sigma^2 = 15,25$	$\Delta \text{FPS} = 4,27$ $\sigma^2 = 21,48$	$\Delta \text{FPS} = 4,37$ $\sigma^2 = 23,65$	$\Delta \text{FPS} = 4,64$ $\sigma^2 = 23,60$
	<b>20 FPS</b>	$\Delta \text{FPS} = 11,78$ $\sigma^2 = 12,63$	$\Delta \text{FPS} = 9,27$ $\sigma^2 = 21,48$	$\Delta \text{FPS} = 9,37$ $\sigma^2 = 23,65$	$\Delta \text{FPS} = 9,64$ $\sigma^2 = 23,60$
	<b>25 FPS</b>	$\Delta \text{FPS} = 19,95$ $\sigma^2 = 5,03$	$\Delta \text{FPS} = 17,75$ $\sigma^2 = 11,09$	$\Delta \text{FPS} = 17,56$ $\sigma^2 = 7,36$	$\Delta \text{FPS} = 18,62$ $\sigma^2 = 7,35$

**Table 6 Calculated Values Of The Experiments**

In the above table  $\Delta \text{FPS}$  denotes the absolute difference between the source frame rate and mean of displayed frame rate and  $\sigma^2$  denotes the variance of displayed frame rate. Also, the below two diagrams represent the table values as figures in order to ease the visualization of values.

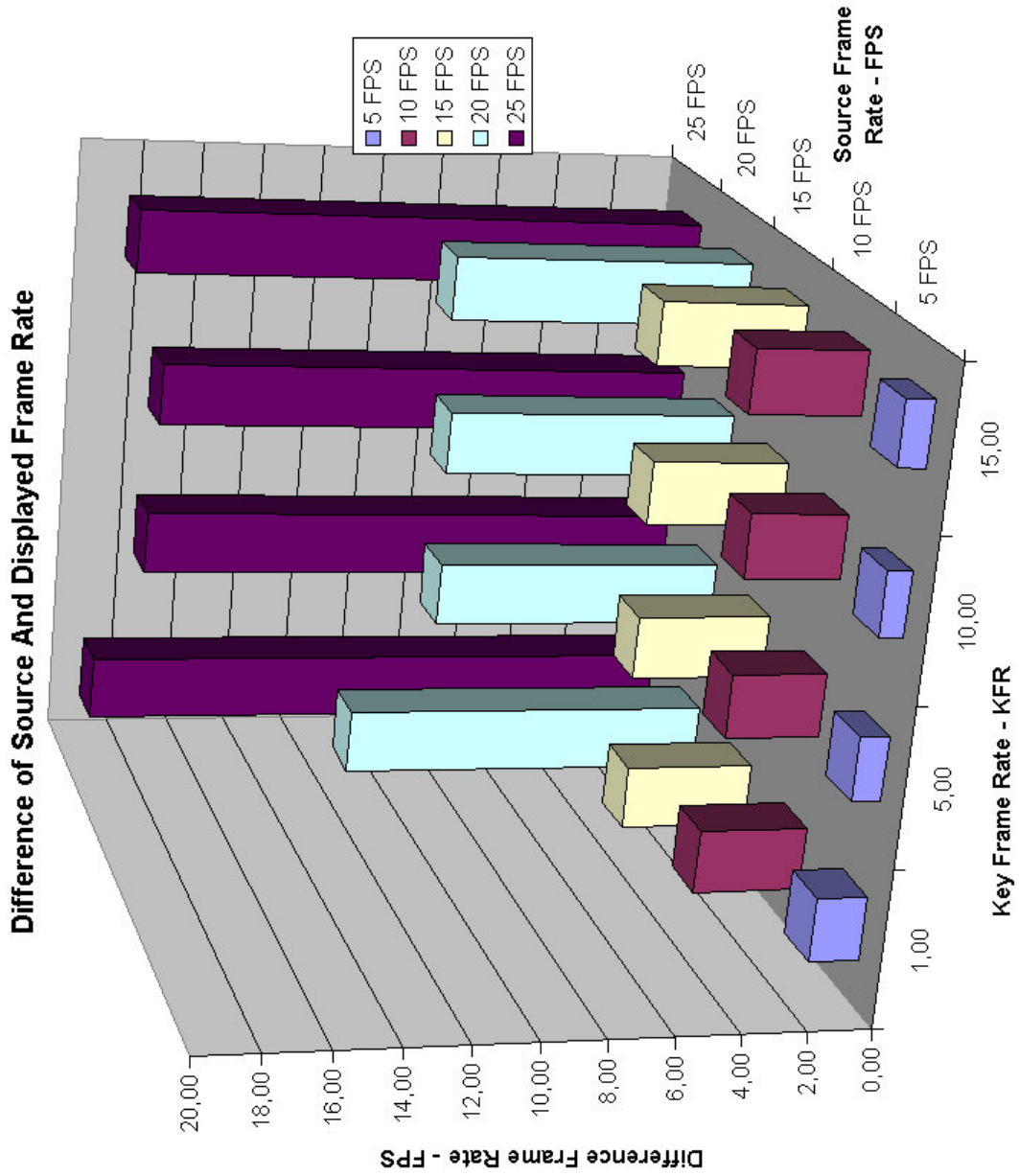


Figure 34 Difference of Source And Displayed Frame Rate ( $\Delta$  FPS)

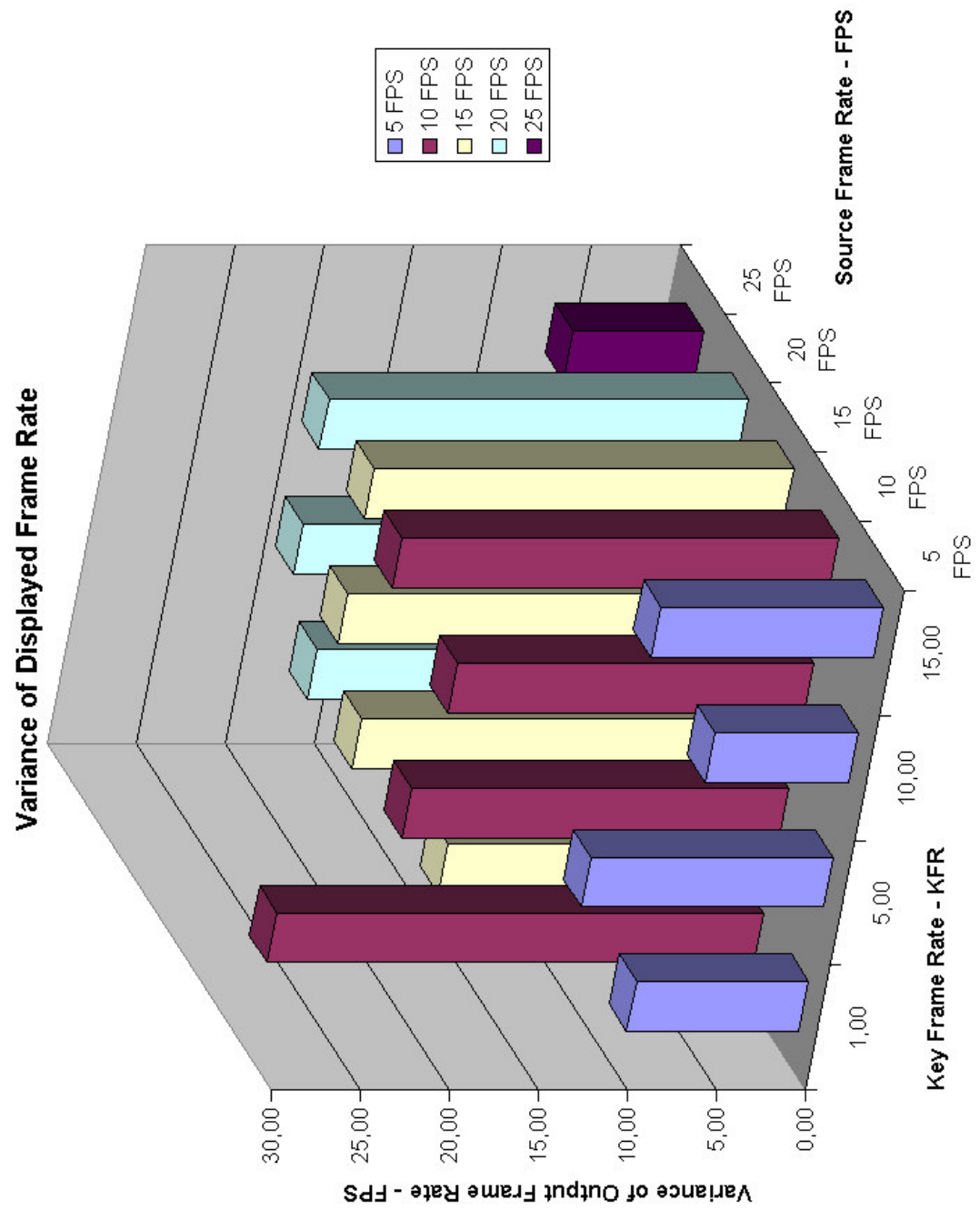


Figure 35 Variance Of Displayed Frame Rate ( $\sigma^2$ )

The following observations are made on the obtained results:

1. The decoder can catch-up with a source rate of approximately 5 FPS. When the source rate is increased, the difference between sources and displayed rates increases. The decoder lags the source rate and more frames are dropped.
2. Changing intra frame rate with a constant source rate does not affect the displayed frame rate. In other words; from the decoder point of view it is not important whether the input frame is I frame or P frame.
3. Changing source rate with a constant intra frame rate reduces the performance of decoder. The decoder cannot catch-up with the source when the source rate is increased.

According to the observations, the source rate is selected as 5 FPS. Slower source rates could also be used. However, the quality of viewing experience degrades significantly.

For the selected source rate, changing intra frame rate slightly changes the decoder performance. However, it seems, from the difference plots, that 15 intra frame rate performs slightly better. Thus, the intra frame rate is selected as 15.

#### **4.4. Comparison**

There are numerous other solutions for previously addressed problems. But as mentioned before, they are the results of uncoordinated efforts. On the other hand, they try to solve only one part of the problem; an integrated approach was the requirement.

There are some other integrated systems as well. These commercial systems employ only the native designs, i.e. blocks of software that target the host operating system and host processor. None of them is trying to provide access content via same executable software from different computing platforms.

These properties differentiate our system from others.

In order to investigate how our system performs regarding to other systems such as Windows Media Technologies and RealNetworks solutions [18], it is required to obtain their quantitative measures. However, it is not possible to measure the performance of these systems using the available tools and it is not possible to access their implementation details.

Although objective comparison is not possible, after evaluating our system and other systems, it can be said that results are comparable to the available systems in terms of visual quality. However, our system has the flexibility and its portable to various other platforms. These features are not available in commercial systems.

## CHAPTER V

### CONCLUSION

The aim of this thesis work is to implement a universal multimedia system, which enables its users to access multimedia content over wireless networks, using different computing platforms. A number of different technologies have been involved in the study. Namely, Java technology is selected as the middleware for the software, Bluetooth networks connect the mobile devices to the network and PocketPc devices are the platform of choice when mobility is required. Also, several experiments have been conducted in order to evaluate the performance of the implemented decoder of mobile devices.

#### **5.1. Results**

The system has been implemented and run according to the requirements given. Experiments have shown that even the small and limited (from the performance point of view) devices can be used as mobile stations to access multimedia content. Although, it is not possible to obtain outstanding results, current technology can satisfy the needs of someone with relax requirements.

Particularly, the best decoder performance is obtained when the source frame rate is 5 FPS and one intra frame is inserted in a 15-frame group. The difference between the output performance of an ideal decoder and our implementation is the basis for this evaluation. It is estimated that with the increase in the computing power available to mobile devices, the decoding performance will not be a matter of concern in the near future.

According to the results, wireless network performance is satisfying for a Bluetooth connection when there is a single mobile device. If a wired connection were to be used alone, no performance loss would have been occurred. In fact, there is already a wired connection –the Ethernet connection between the Bluetooth access point and the server-. The Ethernet connection can sustain orders of magnitude more bandwidth than the wireless. In order to see what the effect would be when the Bluetooth connection is congested, another set of experiments should be done.

## **5.2. Future Work**

The resulting system is far from complete to have commercial value. Moreover, building a commercial system is out of the scope of this study. A real universal multimedia access system requires robust handling mechanisms, user friendly interfaces, management functionality and embedded maintenance support. Hence, more work should be done in case the system is to be commercialized.

On the other hand, the system provides a good base for research. It provides easy to use interfaces for developing experimental software.

The following subjects are some of the research areas that can be experimented using the provided system.

The system has not been tested under heavy network load. Other experiments can be done in order to evaluate the performance of the system; such as using higher bit rate video streams or simultaneous access of different devices.

The system employs Bluetooth for network connection, which is not the only choice as a wireless connection. There is another wireless network that can be used: IEEE 802.11. The system can be modified in order that wi-fi is utilized. 802.11 has orders of magnitude more bandwidth than Bluetooth. Alternative configuration of the system is shown also on Figure 10. With this network, other kinds of experiments that evaluate the performance under heavy network load. Also, GSM/GPRS can be another way of connecting mobile devices to the network.



The system is built on top of unicast video transmission. Broadcasting can be an alternative. Multiple clients can be served via this configuration.

Changing the video codec can be another area that needs to be investigated. H.263 can be replaced by MPEG-4.

Another area can be using the mobile device as the source of content, and video conferencing between two mobile devices can be evaluated.

## REFERENCES

- [1] Mahbub Hassan, Alfandika Nayandoro, Mohammed Atiquazzaman, “Internet Telephony: Services, Technical Challenges, and Products”, IEEE Communications Magazine, April 2000
- [2] DapengWu, Yiwei ThomsHou, Wenwu Zhu, “Streaming Video over the Internet: Approaches and Directions”, pp. 282-300 IEEE Transactions On Circuits and Systems for Video Technology, Vol. 11, No. 3, Marc 2001.
- [3] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, “RTP: A transport protocol for real-time applications” IETF Audio/Video Transport Working Group, January 1996, RFC 1889.
- [4] C. Wen Chen, R. I. Lagendijk, A. R. Reibman, W. Zu, “Introduction to the Special Issue on Wireless Communication”, IEEE Transactions On Circuits and Systems For Video Technology, Vol. 12, No. 6, June 2002
- [5] M. Van der Schaar, H. Radha, “Scalable Video Source Coding”, IEEE Transactions On Circuits and Systems For Video Technology, Vol. 12, No. 6, June 2002
- [6] T.-C. Wang, H.-C. Fang, L.-G. Chen, “Low-Delay and Error-Robust Wireless Video Transmission for Video Communications”, IEEE Transactions On Circuits and Systems For Video Technology, Vol. 12, No. 12, December 2002

- [7] Q. Zhang, Z. Ji, W. Zhu, Y.-Q. Zhang, "Power-Minimized Bit Allocation for Video Communication Over Wireless Channels", IEEE Transactions On Circuits and Systems For Video Technology, Vol. 12, No. 6, June 2002
- [8] C. De Vleeschouwer, T. Nilsson, K. Denolf, and J. Bormans, "Algorithmic and Architectural Co-Design of a Motion Estimation Engine for Low-Power Video Devices", IEEE Transactions On Circuits and Systems For Video Technology, Vol. 12, No. 12, December 2002
- [9] PocketPC Magazine, <http://www.pocketpcmag.com>
- [10] Palm Products, Services, Company Information, <http://www.palm.com>
- [11] Java Media Framework, SUN Microsystems, <http://java.sun.com/jmf>
- [12] Bluetooth Technical Specification, <http://www.bluetooth.org>
- [13] Bluetooth Profiles Specification, <http://www.bluetooth.org>
- [14] Microsoft Corporation, <http://www.microsoft.com>
- [15] Unified Modeling Language, <http://www.uml.org>
- [16] Java Technology Web Site, <http://java.sun.com>
- [17] Atul Puri, Tsuhan Chen, "Multimedia Systems, Standards and Networks", Marcel Dekker Inc.
- [18] RealNetworks Corporate Web Site, <http://realnetworks.com>