# IMPLEMENTATION OF A WIRELESS/MOBILE VIDEO DELIVERY SYSTEM

By

KEVIN BIRKETT

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF ENGINEERING

UNIVERSITY OF FLORIDA

2001

To Kelly, and all my friends and family who supported me.

ACKNOWLEDGMENTS

I would like to thank Dr. Sumi Helal for all his work and dedication as chairman of my committee. I would also like to thank Dr. Wilson and Dr. Dankel for their commitments. Also, I would like to thank Dr. Randy Chow for substituting for Dr. Dankel at the last minute.

I thank Dr. Ron Landis for the use of hardware necessary for testing purposes. Without his cooperation much of the testing for this research would not be complete.

I would like to issue a special thanks to my wife, Kelly, for her patience while I completed my education. Also, I would like to thank my family for all their dedication and support.

TABLE OF CONTENTS

page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Engineering

IMPLEMENTATION OF A WIRELESS/MOBILE VIDEO DELIVERY SYSTEM

By

Kevin Birkett

December 2001

Chairman: Dr. Sumi Helal
Major Department: Computer and Information Science and Engineering

With the widespread distribution of portable computing devices, more users operate in a pervasive manner. As users migrate to mobile technology, they also expect to receive services similar to those available at a desktop computer. The service we look to provide in this thesis is the ability to deliver multimedia content to mobile users connecting with a wide variety of client devices and network types.

Playback of multimedia content on computationally poor devices has become more prevalent in recent years. Streaming video to these computationally poor devices over unreliable and slow networks is difficult and presents many challenges that must be overcome.

In this thesis we examine ways in which we can provide the best quality video to a particular user based on the device and network in use. We propose a client-server architecture to accomplish this task, and implement most of the features in the proposed architecture for testing and evaluation purposes.

To handle several different client and network modalities, a matrix structure is used to define the clients and networks supported by the server. Along with the matrix comes a list of appropriate ways to reduce video content based on specifications of both the client and network. As a request for video is received, the matrix is consulted to see which version of the video should be sent to the requesting client device.

# CHAPTER 1
# INTRODUCTION

Multimedia content is becoming a large part of everyday life for many computer users. Whether watching the local newscast, viewing movie trailers, or watching a how-to series for work, the desire for this type of content is overwhelming. These types of services are currently available to desktop users that connect over high-bandwidth networks. However, with the advent of the personal digital assistant (PDA), mobility is high on the priority list of today's computer users. Our goal in this research is to provide a means for the delivery of multimedia content to these mobile users.

As users migrate from their high-powered desktop systems, to the computationally poor mobile devices, the ability to receive multimedia data becomes more difficult. Not only are we limited by the inability of the mobile devices, but also by the networks in which they are connected. If we attempted to deliver the same video stream to the mobile device as we do for the desktop system, the response time could be so undesirable that the user will not use the service. Even worse, the device may not be able to receive, decode, and playback all the information that is transmitted.

To support the needs of the mobile users, we examine an architecture that focuses on reducing the content of a video stream to make it suitable for a particular device and network bandwidth combination. As the capabilities of the device and network diminish, we reduce the content of the video in such a way that the response time seen by the user is significantly reduced. Without this reduction, playback of multimedia data would be difficult, if not impossible.

When we attempt to deliver video to a mobile device over a wireless network, several issues must be addressed. We must first decide which, if any, current video standard we would like to support. While following standards is not always beneficial, it does insure compatibility with other similar systems. Chapter 2 examines the current standards and the standard chosen for our research.

Our goal from the beginning of this research project has been to provide the capability to deliver video content to a mobile device over a wireless network. Chapter 3 looks at the work of our predecessors, and how we propose to improve upon existing work or completely design a new system. While we focus most of our attention on the client side, we must still create a relatively complex server to handle the requests of mobile clients. The design of our server architecture is discussed in detail in Chapter 5, with the implementation details of both the client and the server revealed in Chapter 6.

At the heart of our video delivery system is the idea of the video reduction matrix. This matrix provides a mapping between devices, networks, and appropriate reductions for a given device-network pair. Our research is limited to a simple implementation of this idea, but the groundwork for future work has been presented, and future changes should be straightforward.

While our main goal was to provide video content to mobile devices, we also had the desire to do so in a timely manner. Simply providing the content is not enough to declare this project a success, and, therefore, we examine the significant performance gains seen by using our system. These results are discussed in detail in Chapter 7.

# CHAPTER 2
# OVERVIEW OF VIDEO DELIVERY SYSTEMS AND STANDARDS

If asked to pick a video delivery system for your company, you would find that the number of choices facing you could be overwhelming. There are numerous formats in use today each having advantages and disadvantages associated with them. One particular concern in the area of video delivery systems is that lack of a common standard from which to build all possible systems.

In this thesis, we focus on the video compression standards laid out by the Moving Picture Expert Group (MPEG) [1]. The reasons for this decision are made clear in the remaining sections of this chapter, as well as Chapter 3.

## 2.1 MPEG Video Overview

The basic MPEG video structure consists of the video sequence, group of pictures, picture, and slice layers [2]. The MPEG video compression standards are frame-based, meaning that the video data are stored as individual frames. This quality proves useful to us as we look at various video reduction techniques.

One inherent characteristic of video data is the large number of bits needed to represent it. If, for example, we look at displaying a video with a resolution of 640 pixels per line x 480 lines, color resolution of 24-bits per pixel, a frame rate of 30 frames per second, and a length of 60 seconds. Such a video, in uncompressed format would require:

((640 * 480) * 24 * 30 * 60) = 1,327,104,000 bits of storage space

or roughly 1.66 gigabytes (GB). Even with today's rapidly increasing hard drive prices, clearly we must find a way to represent such a video using less space. The MPEG formats achieve this compression by performing reductions in both the temporal and spatial domains of a video [3].

### 2.1.1 Temporal Compression Techniques

Reduction in the temporal domain is achieved by taking advantage of the close relationship between adjacent frames. Generally, an object in a video does not appear and then re-appear instantaneously. It is usually the case that an object in the video simply moves around the screen, and some objects may remain stationary. MPEG standards typically examine this motion in 16 x 16 blocks of pixels, which form a macroblock. Using this fact, MPEG defines three basic frame formats that are used to encode a video stream—I-frames, P-Frames, and B-Frames.

I-frames, or intra-coded frames, are frames encoded without reference to any other frame [3]. In other words, the frame in the video that we wish to encode is coded without any motion prediction. These frames serve as a starting point, and also help to remove any errors that may occur with motion prediction as the stream is encoded. Since I-frames do not use motion prediction, they are the largest frames in an MPEG video stream.

P-frames, or forward-predicted frames, are encoded based on the information in the previous I or P frame. Figure 2.1 shows the redundancy between two successive frames. Since we have the two consecutive frames that are to be encoded, it is necessary to search each frame for similarities in 16 x 16 blocks of pixels. Since these P-frames only encode the differences between the current frame and a previous frame, the required number of bits to represent this frame is less than that of the I-frame.

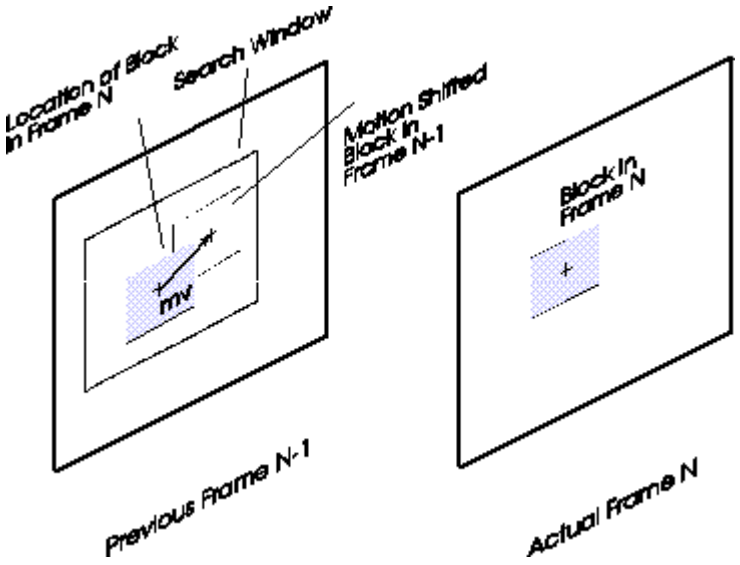


Figure 2.1. Motion vector calculation [4]

B-frames, or bi-directionally predicted frames, are encoded on information stored in the previous frame, successive frame, or both [5]. Shanableh and Ghanbari [6] claim that B-frames help to hide errors that may occur between frames. One reason for this is the fact that the B-frames can be encoded from both a previous and a future frame. This allows for the use of the average motion vector, and can be thought of as an average of two different frames. Since it consists of an average and contains information from the two video frames, it will have the effect of making the video appear smoother.

With both the P and B frames, algorithms are used to determine the best motion vector prediction for the current frame. With the P and B frames each macroblock is treated individually and, if no appropriate prediction can be made, then the block is intra-coded like an I-frame [3].

The Berkeley Multimedia Research Center (BMRC) suggests a typical playback sequence similar to Figure 2.2. Since the B-frames could be referencing a future and a past frame, we must transmit the P-frame before the two B-frames [3].

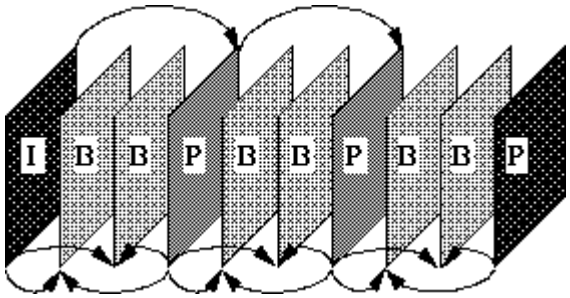


Figure 2.2. Typical sequence of frames during playback [7]

2.1.2 Spatial Compression Techniques

The previous section outlined the nature of video data and the redundancies between adjacent frames. To achieve greater compression, the MPEG standards also exploit similarities in adjacent pixels. Spatial compensation is applied on Intra-coded frames, as well as non-intra coded frames after motion prediction has taken place. A formula known as the Discrete Cosine Transforms (DCT) organizes these blocks spatially [3]. By using a DCT we can represent information in a more concise manner, which is desirable when attempting to achieve maximum compression. If we consider a 16 x 16 block of pixels, the color values of these pictures are generally related. The MPEG coding scheme takes advantage of this by converting the brightness of each pixel in the block into the frequency domain by using the DCT formula in Figure 2.3. Figure 2.4 shows the Inverse Discrete Cosine Transform (IDCT) used to decode the video stream on the receiver side.

$$t(i,j) = c(i,j) \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} s(m,n) cos \frac{\pi(2m+1)i}{2N} cos \frac{\pi(2n+1)j}{2N}$$

Figure 2.3. The DCT formula [8]

$$s(m,n) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} c(i,j) t(i,j) cos \frac{\pi(2m+1)i}{2N} cos \frac{\pi(2n+1)j}{2N}$$

Figure 2.4. The inverse DCT formula [8]

## 2.2 MPEG Standards

Currently there are five main video standards that have been released by MPEG—MPEG-1, MPEG-2, MPEG-4, MPEG-7, and MPEG-21. Since our implementation will focus on the MPEG-1 and MPEG-2 standards, we will focus our discussion on these areas. The information presented in the previous section applies to both the MPEG-1, and MPEG-2 standards. Other authors [9,10] give detailed descriptions of MPEG-4, MPEG-7, and MPEG-21, and therefore will not be discussed in great detail in this thesis.

### 2.2.1 MPEG – 1

MPEG-1 was the first video standard released by MPEG in the early 1990's [1]. MPEG-1 is targeted to provide VHS-quality video, and is optimized for 352 x 240 pixels, 30 frames per second, and approximately 1.5 megabits per second (Mbps) [2]. MPEG-1 video only supports progressive frames, so it is not ideally suited for television, which uses interlaced frames.

### 2.2.2 MPEG –2

MPEG-2 was finalized in 1994 [5]. The need for a new digital television broadcast standard prompted the development of MPEG-2. This standard provides the ability to encode interlaced frames, which are necessary for broadcast television. In

addition, MPEG-2 provides a much greater video quality than does MPEG-1. However, as with most situations in computer science, there is a tradeoff between quality and size.

As with any compression technique, if we wish to obtain a higher quality picture, we must sample the data more frequently. One side effect, of this, is the necessity for a greater number of bits to represent that data accurately. The new High Definition Television (HDTV) system provides high quality pictures to the user's television set. MPEG-2 has been adopted as the HDTV compression standard, and due to the high quality requires a bandwidth of 12-20 Mbps [5]. There is currently a debate over which standard to adopt for Digital Video Disc (DVD) technology, with MPEG-2 being the most prevalent technology.

While MPEG-2 video does provide high quality images, such bit-rates are not practical in our system involving wireless networks. However, since most videos are encoded with the MPEG-2 standard, we would like to make our system capable of handling these files. Since our client will be based on the MPEG-1 standard, we will use publicly available utilities to convert from MPEG-2 encoding to MPEG-1 encoding. This is discussed further in Section 7.1.6.

### 2.2.3 MPEG – 21

While most previous MPEG standards involved defining a syntax for the compression of video, the MPEG-21 standard looks at a different area of need in the multimedia environment. Currently, the different video delivery formats and standards are too numerous to count. Different vendors develop proprietary formats that suit the needs of their systems. With all these systems, there is no clear understanding of how these elements are related and how they could work together [10]. MPEG-21 is a

standard to define a *multimedia framework*, for the presentation of multimedia information [10].

The MPEG-21 standard defines seven key elements that are needed to properly implement a multimedia framework [10]. Of the seven, we explore three, namely Digital Item Identification and Description, Intellectual Property Management and Protection, and Terminals and Networks.

2.2.3.1 Digital item identification and description

From live newscasts to interactive online golf lessons, the amount of multimedia data available is increasing at a staggering rate. However, as this content continues to increase, it becomes more difficult to determine which content you would like to view. The MPEG-21 standard defines the means by which the content of the multimedia data can be described, and then viewed by the user. The group may choose to use a derivative of the MPEG-7 standard, which aims to make multimedia data as searchable as text data. MPEG-21 lists several goals for identification and description [10], including:

- A common description and identification scheme,
- Standard methods of accessing information, and
- Inter-operability with existing standards.

2.2.3.2 Intellectual property management and protection

Recently, there has been a debate over the highly controversial company, Napster. Copyright groups are outraged at the ability of individuals to share music files in violation of copyright laws. In light of this revelation, it has become apparent that some steps must be taken to insure that the intellectual property of the creator is preserved. The MPEG-21 standard defines several ways to facilitate the protection of the owners of multimedia data. Since most multimedia data today is in a digital format, it can be shared

with others. The MPEG-21 standard states the need to "specify a framework for the enforcement of the management and protection of digital items [10:8]. Unfortunately, this is an enormous undertaking and MPEG gives no mention of how this is to be accomplished.

2.2.3.3 Terminals and networks

Of interest in our research is a mechanism by which we can deliver multimedia data across varying network and device modalities. We would like for the users to be oblivious to any change in network quality and would like to supply zero-configuration where device information is gathered dynamically. Bormans and Hill state, "The vision for MPEG-21 is to define a multimedia framework to enable transparent and augmented use of multimedia resources across a wide range of networks and devices used by different communities." [10:1] It appears that the members of MPEG have recognized the growing collection of multimedia collections, as well as the varying modalities of today's users. MPEG-21 could prove useful in a system such as ours, however, MPEG-21 is quite new, and probably will not be finalized for several years. Therefore, we will not use it for our system, but it could prove useful for future modifications.

## 2.3 Multimedia Streaming

It is apparent that multimedia content is becoming a part of every day life for many computer users. It is not uncommon to watch your local news, or even music videos on your computer. For some time it has been possible to play videos stored on a local hard drive or CD-ROM drive. With the advent of the Internet, and high-speed networks, we have seen a shift to multimedia content providers supplying information on websites or other servers. With information at a remote site, we have two ways in which we can handle it—download the entire file and then begin playback, or playback the

information as it is downloaded. The second choice, known as streaming, has several positive attributes.

### 2.3.1 Streaming Advantages

One of the advantages of streaming technology is that it eliminates the need to store large video files on the local hard drive. If the video is stored on a remote server, we can request that video and then play the video as the information is received. The only storage space we require is a buffer to offset a slow network with a relatively fast video playback.

The second advantage of streaming video is the broad range of multimedia content available. If we limit ourselves only to videos stored on the client device, we have a limited number of choices. With the wide array of content available across the Internet today, the choices in multimedia content are virtually endless.

If we desired to have up-to-date information, local storage would not be an option. Consider a breaking news story involving a wildfire threatening your home. It is not possible to have this video stored on your local device, since the story is taking place at the same moment. If we wish to view that newscast we would be required to stream it from the content provider.

### 2.3.2 Streaming Difficulties

With the advantages we have discussed, one may be curious as to why this technology has not overtaken the marketplace. The bottom line of why streaming technology has not dominated the market is simple—poor quality [11]. The streaming of multimedia data is extremely sensitive to any errors in delivery. If an error occurs, it is immediately visible to the user in the form of jerky motion, or in the worst case, a complete loss of video. In most data delivery systems, if a data transmission error occurs,

the information is simply retransmitted. However, if an error in a video stream occurs, it does not make sense to retransmit that data since successive frames have already been played.

Another problem with streaming media is the large bandwidth required to deliver multimedia content. As we discussed in Section 2.1, obtaining VHS quality video with MPEG-1 requires around 1.5 Mbps. An enormous amount of network bandwidth is utilized for one video, not to mention if several videos are playing over the same network simultaneously. While corporate networks have the capabilities to handle some streaming, since most Internet traffic is routed through one gateway, the performance of the rest of the network can suffer.

### 2.3.3 Streaming in a Wireless Environment

One inherent quality of multimedia content is its bulky nature. With today's powerful desktop computers and high-speed networks, these videos can be handled with relative ease. However, we have seen a shift in the modality of computer users. More and more individuals are becoming mobile users with the advent of the Personal Digital Assistant (PDA). While the powerful desktop computers may handle a video with ease, playing the same video on a PDA may be difficult or even impossible.

Current wireless technologies have several problems. The most notable of these are low bandwidth, high latency, and high bit-error rates. As we discussed in the previous section, any errors in the transmission of a video are immediately visible to the user and cannot be fixed by retransmission. There is ongoing research in the area of quality of service (QoS) in wireless video systems [12], which try to predict any errors that may occur during video transmission. If errors can be predicted, we may be able to

correct them before the user sees them. If the wireless connection is very limited, we might have to inform the user that video cannot be played over the current link.

## 2.4 Reduction Strategies

In our research, we wish to provide multimedia content to a wide variety of client devices over a wide range of network capabilities. With this wide range of network and device modalities one video cannot possibly provide the appropriate content. For example, let us look at providing a 640 x 480 video with 256 colors, and a frame rate of 30 frames per second (fps). If we have a client device with a screen size of 240 x 120, we would not want to transmit the original video to the client, because we would waste valuable network resources that may or may not be available. To accommodate the various combinations of devices and networks, we must devise ways to reduce the content in videos so they are suitable for the client's network and client modalities.

### 2.4.1 Size Reduction

As the popularity of PDA's continues to increase, we are forced to accommodate the screen size limitations of such devices when we attempt to deliver video content to them. These devices usually have a screen capable of displaying about 240 pixels per line, with 120 lines on the screen. This is much smaller than the 1024 x 768 resolution used on most desktop systems. With this idea under consideration, it would benefit us to provide a reduction that will allow a video to fit the size of our screen. While most video decoders can compensate for images larger than the screen by scaling the video, we waste network bandwidth and might possibly send more data to the device than it is capable of handling.

The idea of size reduction is fairly basic. We need to reduce the number of pixels used to represent the images. If we can reduce the number of pixels in the images, we

can reduce the overall size of the video, providing the appropriate size video for a specified device while using as few network resources as possible. To reduce the number of pixels, we can take the average of neighboring pixel values, and make them the new pixel value for the reduced value. While this reduction can be done at the pixel level, reductions at the block level should provide reduction without noticeable quality reduction. Algorithms have been devised to combine four 8 x 8 blocks of pixels into one 8 x 8 block of pixels [13].

### 2.4.2 Color Reduction

In the same way a mobile device may not have the same resolution capabilities of a desktop system, there will most likely be a difference in the amount of color that can be displayed on the mobile device. Most mobile devices do not have the ability to display full color (24 bits per pixel). Therefore, it is important for us to reduce the amount of color information stored in the image. Reducing the number of colors in the video similarly reduces the number of bits required to represent that video. In some extreme cases, it may be necessary to display a video on a black and white device. If we attempt to transmit a full color video to a black and white device, we have sent superfluous information to the client and thereby wasted valuable network and computational resources.

### 2.4.3 Frame Dropping

We have discussed two reductions that work to fit the content of a video into the target device. We will now examine a frame dropping reduction, which does not change the size of the stored video, but rather regulates how much information is transmitted to the client device.

2.4.3.1 Frame dropping scenarios

As we have seen, mobile devices have limited computational abilities. With this limitation comes the restriction on the number of video frames that we can play per second. Most videos today are encoded at 30 fps. This rate is optimized for desktop systems, whereas most handheld devices cannot support such a high frame rate. To combat this, we provide a method to reduce the number of frames transmitted to the client device. If we can prevent extra frames from getting sent to the device, we prevent the client from being overwhelmed with data, while at the same time easing the burden on the network.

Frame dropping can be used in more situations than just the simple fact of the player not being able to present frames quickly enough. Certain conditions may arise where the quality of the network connection degrades. This degradation will limit the amount of data we can transmit to the client device. To accommodate this situation we can drop frames if we detect the network quality reduction warrants it.

Similar to the reduction of network quality, is the situation where the client receives more data than it can handle at one time. If the client is unable to process the frames quickly enough, frames passing through the network may be lost due to timeout errors. It would be convenient for the client to have a mechanism by which it could ask the server to send less information for a while to allow the client to catch up. The only realistic way the server can reduce the amount of data transmitted is to drop frames within the video sequence. The alternative is to stop transmitting data, but this could cause serious problems including complete loss of video.

2.4.3.2 Selection of frames to drop

After we have made the decision that frame dropping needs to occur, we would like to do so in such a way that limits the e ffect on the user's perceived quality of video. If we examine a video of a baseball game, we notice that there are long lapses in the action, followed by short bursts of activities—pitches, hits, and stolen bases. If we had information about the video that specified which portions contained heavy motion, and which contained relatively little motion, we could choose to drop frames within sequences containing small amounts of motion. If the scene is not changing very rapidly, then the dropping of a few frames may not be perceivable by the user. The MPEG-7 standard deals with content description of video and is discussed in further detail by Sampath et al. [9].

# CHAPTER 3
# WIRELESS AND MOBILE VIDEO DELIVERY

With high-speed networks and fast processors, delivering video to a fixed-network workstation is almost trivial. When we attempt to deliver this same video to a mobile device over a wireless network, we are presented with several challenges not present in the fixed network scenario.

## 3.1 Device Characteristics

With the advent of the Personal Digital Assistant (PDA), users of computers have become more mobile. Capabilities that once were limited to the desktop can now be provided to devices that fit in your hand. However, with the reduction of size in these devices comes a reduction in performance relative to a workstation. Limitations of these devices include:

- Screen size,
- Color depth,
- Memory, and
- Processor speed.

All these factors must be considered when evaluating the ability of a device to play multimedia data. Table 3.1 shows the relationship between several of the handheld computing devices currently on the market.

Table 3.1. Summary of device specifications [14]

| | iPAQ 3600 | Cassiopeia EM-500 | Jornada 540 | Handspring |
|---|---|---|---|---|
| **Processor** | 206 MHz | 150 MHz | 133 MHz | 33 MHz |
| **Memory** | 32 MB RAM | 16MB RAM | 32MB RAM | 8MB RAM |
| **Resolution** | 240 x 320 | 240 x 320 | 240 x 320 | 160 x 160 |
| **Colors** | 4086 | 65536 | 4086 | 256 |

At the current time, an average desktop system contains a 1.0 GHz processor, 256 MB RAM, and 40 GB of storage space. Comparing these numbers with those found in Table 1 shows the large difference in performance between mobile devices and desktop systems. While these devices are powerful enough for typical business applications, the playback of multimedia data requires large amounts of memory and storage space, which are limited on these devices. Therefore, because of the limited computational power of these devices, we must consider reducing the video content to better suit the needs of these mobile devices.

The limiting factor of the perceived video quality on these devices is determined by several factors. The major limitation is the resolution of the screen. If the device has a screen resolution of 240 x 320, we cannot display video with a 640 x 480 resolution. The second major factor is the number of colors the device is capable of displaying. Certain mobile devices are only capable of displaying black and white images, and the perceived video quality would therefore be less than a device with a full color display. The final factor in video quality is the frame rate the player can display the video. The higher the frame rate, the less jerky the video appears. The higher the frame rate, the less

sensitive the video is to changes in motion. The value of the frame rate is dependant upon the processor speed, memory size, and the bit-rate at which the video was encoded.

## 3.2 Network Characteristics

With the proliferation of fiber optics, wired networks can move data at literally the speed of light. A Gigabit Ethernet is capable of transmitting 1 billon bits per second. As users become more mobile the typical fixed network connection is no longer an option, and wireless connectivity is needed to stay connected to the network. Wireless connectivity ranges from short-range, high-bandwidth to long-range, low-bandwidth.

### 3.2.1 Global System for Mobile Communications (GSM)

GSM is the technology associated with 2$^{nd}$ generation wireless networks. GSM is a digital signal based system, and is used throughout the world for wireless phone service. Since this technology was designed with carrying voice signals, the amount of data that can be transmitted is very limited. Data speeds are limited to 9.6 kbps in current GSM systems [15]. Typical devices associated with this class of network are cellular phones. Recently phones have become more capable devices, and some are even capable of video playback. However, video playback on such a network requires large reductions in the video content and a large buffer time. With this type of network it is imperative that any video stream be reduced as much as possible before transmission.

### 3.2.2 General Packet Radio System for GSM (GPRS)

GPRS is a system used to allow packet data transmission to existing GSM cellular networks. There are 8 timeslots for each time division multiple access (TDMA) frame, from which each user can be allocated one or more slots [16]. Depending on the number of timeslots allocated to each user, data rates of up to 150 kbps are possible [16]. While

data rates can be as high as 150 kbps, it is rare that one user be given all 8 timeslots for their data. The advantage of a packet-based system is that the network is in use only when data needs to be transmitted. This differs from a circuit switched network designed for voice communications that remains connected even if no data is in need of transmission. Because there is no need for a constant connection with GPRS, service providers typically charge based on the amount of data sent or received by a particular user. Our video delivery system could be very useful in such an environment where the minimum amount of data must be transmitted. Reducing the content of the video to be delivered not only minimizes the network bandwidth utilization, but also could save money for users of the GPRS devices.

### 3.2.3 Bluetooth

Bluetooth is a network designed for mobile devices concerned with power consumption issues. A Bluetooth network can provide data rates of up to 1Mbps. As a consequence of the low power consumption, the useful range of Bluetooth is limited to about 10 meters [17]. Recent worries about interference between Bluetooth and 802.11 networks have caused a delay in the widespread distribution of Bluetooth networks. More information on this technology can be found on the Bluetooth website [18]. When we get to speeds associated with Bluetooth networks, video delivery becomes more practical. The amount of reduction required is not as severe as with GSM and GPRS networks.

### 3.2.4 802.11 Wireless LAN

The IEEE 802.11 standard is an in-room technology capable of 11Mbps with a usable range around 300 meters [17]. While this type of network does provide a high bandwidth, this comes with the price of limited mobility. The user is restricted in the

amount they are able to roam from the access point before a handoff must occur. Currently handoffs in this type of network are very costly in terms of network disconnection time.

When connected with a wireless LAN, we have the freedom to make decisions about video delivery based more on the device's capabilities rather than the network capabilities. The reason for this is that the limitations of the mobile device to receive, decode, and display video outweigh the amount of data that can be transmitted by the network.

While this network is quite fast compared with the other networks discussed in this thesis, it is not without is pitfalls. As with any wireless communication wireless LANs are error prone and have high latency values. Since the data is visible to the user, the user sees any error in transmission on the screen. Additionally, if there is an error in transmission, a retransmission of the data is usually impractical because of the temporal nature of video transmission. In other words, if there is an error in a particular frame, resending the frame is pointless since the time for that frame to display has already passed.

# CHAPTER 4
# RELATED WORK

Currently, there are several video players available for personal computers. RealPlayer, Windows Media Player, and QuickTime are just a few of the products used to play multimedia data. While these technologies are available for most desktop systems, the number of products available for mobile devices is somewhat more limited. This chapter examines some of the technology available for mobile devices as well as research in the area of providing a variety of video content to various devices.

## 4.1 Windows Media Player 7.1

Microsoft's Windows Media Player is a freely available software package [19]. To play video files locally, or stream them from remote locations, the video must be encoded in Microsoft's proprietary file format. To accomplish this task, Microsoft provides the Windows Media Encoder package, allowing content providers to convert various video formats to the Windows Media format. One nice feature of this system related to our work is the ability to encode video with a specific target platform in mind. Currently, profiles for the Compaq iPAQ, HP Jornada, and the Casio Cassiopeia are available as add-ons to the Windows Media Encoder software [19].

While the Media Player is a nice software package, the one limitation is the proprietary file format. In Microsoft's attempt to control every market, they avoid the standards set forth by MPEG. While this may not appear to be a problem, it does present

compatibility issues between users of other video systems. If you have an MPEG video stream, it will play on any player claiming to be MPEG compatible.

Another limitation of the Windows Media Player is that it is closed source. From an educational perspective, it is nice to look at the code and see how the software is working. Research can then be done to find better ways to perform certain tasks and to customize the software package for specific research or commercial needs. Most MPEG players are open source, and this is one main reason we choose to use MPEG standards in this thesis.

## 4.2 Packet Video

PacketVideo is a relatively new company dedicated to providing streaming video content over a wide range of networks. In their own words, "PacketVideo, the global leader in wireless multimedia, develops MPEG-4 compliant software that enables the delivery, management and viewing of video and audio over current wireless networks to mobile information devices such as cellular telephones and personal digital assistants" [20:1]. PacketVideo also makes the claim that video encoded with their software can be adapted to many different bit-rates without recoding. This is accomplished with help from the facilities provided in the MPEG-4 standard, as well as the Dynamic Rate Control system developed by PacketVideo [20].

Overall, the PacketVideo software is the most advanced commercial product available for streaming video over limited wireless connections. It is the opinion of the authors that while PacketVideo does address the problem of limited network bandwidth, they fail to address the second half of the problem when streaming to mobile devices—the actual limitations of the device. PacketVideo does claim their software takes into

account devices with limited processing power and display sizes [20]. However, they seem to compensate for the general principle of the portable devices being smaller, rather than adjusting for each class of device that may connect to their system.

## 4.3 Emblaze

Emblaze is a recent company that has emerged from the wireless video industry. In their own words " Emblaze provides the first commercial end-to-end solution for streaming video over existing and future wireless networks and devices." [21:1] The Emblaze system is one that involves a server and a thin-client with the option of a special middleware for network transport. The company claims support for mobile phones as well as PDA's running the Windows CE operating system. While at first glance this appears to be a nice solution, special hardware is required to properly serve video to an Emblaze client program. It is the opinion of the author that support should be provided to work on existing network infrastructures without the need for specialized server, client, or network hardware. Additionally, the emblaze system makes no mention of video reductions. Without reducing the original content of the video stream, it is difficult, if not impossible, to provide the best quality service given any network and device modalities.

# CHAPTER 5
# REFERENCE SYSTEM ARCHITECTURE

The architecture for our video delivery system was developed with modularity and scalability as two of the main underlying goals. To realize a system capable of streaming video to a wide variety of clients, several factors must be considered. Perhaps the main difficulty with such a system is the diversity of devices and their corresponding network modalities. Designing a system to support video delivery to fixed network workstations is not as challenging as our system, which must adapt video data to a wide range of client devices.

The clients in our system are limited in the amount of data they can receive as well as the amount of data they can display. For instance, if a client device only supports 4-bit color, it makes no sense for us to deliver a video with 24-bit color. Sending such a bit-stream would waste valuable network resources, and may even overload our limited client in such a way that it cannot process the video stream. Similarly, sending a 640 x 480 video image to a device with a screen size of 120 x 100 has similar consequences to the excess color delivery. We must be careful not to deliver too much information, while also trying to deliver the richest possible content to the client device. Because we try to give the client a near-zero configuration, most of the computation will be done on the server.

Our server consists of several modules each perform an integral task in properly delivering video content. By splitting the system into modules, we make each task independent of the next, which allows for greater flexibility in the system. Figure 5.1

shows a high-level view of the system, with the flow of a request for video indicated by arrows.

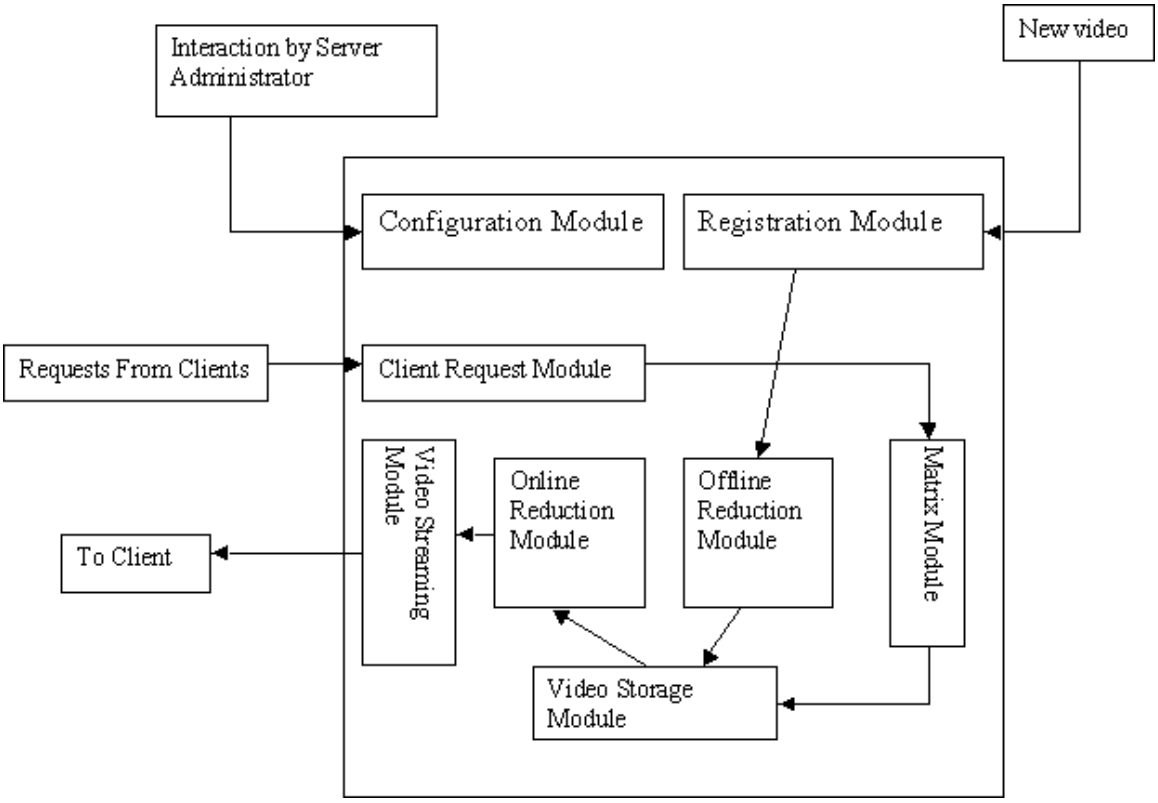


Figure 5.1. Overview of the video delivery architecture

## 5.1 Client Request Module

The client request module is responsible for handling the request for video sent by the client device. In this stage a number of actions must be taken before the remaining modules can be invoked.

One of the key features of this system, is the near zero configuration required on the client side. With current products in use today, users are required to enter the type of network connection present on their device. While this works fine for users with fixed workstations and network connections, pervasive users may change connection

modalities several times during the course of one day. For instance, a user of a Personal Digital Assistant (PDA) may have a 10Mbs wireless network connection in the office, but while traveling outside the building, perhaps only a 14.4 kbps connection is all that is available. In this situation the user would be required to change settings on the client when changing network modalities. Further complicating this issue is the fact that the user may not know when the network conditions changed, as is the case with vertical network handoff scenarios. The client request module aims to ease the burden of the user so they do not have to enter network parameters by hand. To dynamically determine the status of the network connection several issues could be considered.

Bandwidth, latency, and bit-error rate are all metrics used to determine the capabilities of a network connection. It is our opinion that these metrics would be difficult to measure in a real-time environment, so a compromise must be made. The end result of all the discussed metrics is the amount of data received by the client per unit time, or the throughput. The underlying concept of this module's functionality is to measure the throughput of the client device, and then determine the category of network based on the results of the network test.

As the module receives the request, it must first determine the capabilities of both the client and the network between them. Along with the request for video, the client sends information about itself to the server to aide the server in selecting the reductions appropriate for the specific device. The information to be sent is no less than:

1. Operating system name,
2. System memory size,
3. Screen size, and

4. Processor type and speed.

As the above information is received, it is stored locally by the server and will be available to other modules in the system as needed. After the request is received the module then initiates the network throughput test. When the client receives the first byte of information it starts a timer, which records the elapsed time from the start of the transmission to the end of transmission. This time is then returned to the server and the throughput calculation is made by dividing the amount of data transmitted by the time it took for the client to receive all the data. Once the throughput has been determined, the server is ready to activate the Matrix Module.

## 5.2 Matrix Module

The Matrix Module houses the key component in the video delivery system, the reduction matrix. The reduction matrix is the core of our video delivery architecture, as the entries in this table define which devices will receive which video. While the reduction matrix is the core of the system, this module's functionality is quite simple. The Client Request Module is responsible for gathering information from the client regarding network and client capabilities. With this information the Matrix Module needs only look up the correct entry in the table corresponding to those values.

### 5.2.1 Device Categorization

This module contains a mapping to certain reductions based on a function of the device capabilities and network quality/speed. Based on the type of device requesting the video, and the available bandwidth of the network connection, predetermined reductions are found in the corresponding entry in the matrix. This information is computed in the Request for Video Module and is available to this module when it is invoked. Using this

information appropriate reductions are selected from the matrix and passed to the Video Storage Module. A generic form of the matrix is shown in Table 5.1.

Table 5.1. The video reduction matrix

| | Device Type 1 | Device Type 2 | Device Type 3 | … |
|---|---|---|---|---|
| Network Type 1 | reduction x | reduction yy | reduction zz | |
| Network Type 2 | reduction xy | reduction yz | reduction zy | |
| Network Type 3 | reduction z | reduction xz | reduction yx | |
| … | | | | |

5.2.2 Selection of Appropriate Reductions

The selection of appropriate reductions is made simple in some cases, but in others can be a difficult decision to make. If, for example, we have a column in the matrix representing a device with a 4-color display, it would not make sense to transmit a 24-bit color video to that device. Therefore, it would make sense to have color reduction as an appropriate reduction for that particular classification of devices. Similarly, if we have a column in the matrix representing a device with a screen size of 120 x 110, we should not transmit a video with 640 x 480 resolution to such a device, because network bandwidth would be wasted and the device may not be able to interpret the larger video file. These concepts are discussed further in Chapter 2.

While the server will examine the matrix and make a decision based on the nature of the client, and the quality of the network, user intervention is still required by a knowledgeable administrator. This administrator must fill the matrix with the appropriate reductions. The server program can then make a determination as to which reduction or reductions should be applied to the particular video before transmitting it to the client. In

the case of our server, when this module selects the appropriate reduction, any offline reductions have already taken place at registration time, so the video file can be selected and streamed to the client.

### 5.2.3 Choosing the Ideal Reduction

After we determine a set of appropriate reductions, it is necessary to choose the reduction or reductions that make the most sense for a given video. Determining which of these reductions should be used is a difficult task, and is one that will not be fully examined in this thesis. To achieve the ideal reduction, each video must be examined for content, such as color and motion, and a decision must be made based on that content.

When deciding on the ideal reduction, the system benefits if it has information about the requested video available. If we have descriptive information about the requested video, then we can make a more educated decision about the type of reduction or reductions to perform. For example, let us examine the frame dropping reduction. Frame reduction can cause a loss of perceived video quality, especially in videos that contain rapid scene changes. In other words, a video of a football game contains rapid scene changes and dropping frames can cause unsatisfactory results. However, a video of commentators talking about a football game contains relatively few movements, and, therefore, dropping frames would not affect the perceived quality of the video delivered. The MPEG-7 standard provides content descriptors that describe various aspects of a video including motion and color. Therefore, if we have knowledge of the amount of motion and the richness of the color in the video, we can make a reasonable attempt to provide the reduction that least affects the perceived quality by the user. However, these observations are made for videos that fall within the computational capabilities of the

device. Even if we know that the video contains rich color content, the requesting device may not support full color and, therefore,a reduction still must be performed.

## 5.3 Video Storage Module

The Video Storage Module is responsible for a few operations in the overall process. The first functionality of this module is to maintain references to all of the video files stored on the server. These references are used when the appropriate version of the selected video needs to be retrieved for preparation to send to the client.

The second purpose of this module is to examine the list of appropriate reductions and make a decision as to which reduction or reductions should be used. To choose the ideal reduction, several factors need to be considered.

### 5.3.1 Multiple File Versions

Since our system deals with servicing clients with varying network and device capabilities, one version of a video cannot suit the needs of all devices. To allow all clients to receive the maximum quality video, we must provide different versions of each video, each with different properties.

Based on the settings provided by the server administrator, this module will store appropriate versions of each video registered with the system. The versions stored correspond to the offline reductions that were chosen for the system. If there are three offline reductions defined, then there will be at least four versions of the video stored (the original, plus the three versions corresponding to the reduced versions). There may be files that have more than one reduction applied to the video. For example, we might want to have color and size reduction on one video. Since neither of these can be performed online, we must store a version of the video with both reductions already performed.

This method of video storage is needed because of the inherent compute intensive algorithms used to reduce the video. While the storage overhead involved with such a system is large, it is necessary since the technology for most reductions does not permit real-time results. These techniques are described in further detail later in this chapter.

### 5.3.2 Interaction with Other Modules

For the Video Storage Module to operate correctly, it must interact with the rest of the system. The main purpose of this module is to take information from the Matrix Module and the Offline Reduction Module to provide a method to select appropriate video files. The input from the Matrix Module provides information about which reductions should be performed, and then the Video Storage module selects the video based on the reductions performed in the Offline Reduction Module.

## 5.4 Offline Reduction Module

As the name implies, the Offline Reduction Module is responsible for producing all the static reductions requested by the Registration Module. The offline reductions that will be explored in this thesis are color reduction, size reduction, and rate reduction. The details of these reductions are discussed in Chapter 2.

Because these reductions generally require that the video be decoded, reduced, and then re-encoded, it is not feasible to perform these reductions in a real-time environment. Therefore, as a new video is registered with the server, the Offline Reduction Module is informed and all static reductions are performed before the video is available to clients.

Since the all reductions are performed upon video registration, a large amount of time can elapse between the time the video is registered, and the time the video is

available for the client to view. However, this process will only happen one time, and there may be unlimited accesses by the clients. Therefore, we tradeoff a longer delay at registration time for quicker access to a particular video by the client. If each client had to wait for all the offline reductions to complete, several minutes could elapse between the request and the actual streaming of the requested video. By computing the reductions statically, we also eliminate the problem of the same reductions being repeated by several different clients.

## 5.5 Online Reduction Module

The Online Reduction Module has the responsibility of handling any reductions that need to be applied as the video stream is being transmitted. As we have discussed previously, most reductions must be performed offline, since the computational resources required to decode, reduce, and then encode the video are too great to allow this to happen in real-time. We will, however, examine the one reduction in our system that can be performed as the video is being streamed to the client.

One of the great advantages to using MPEG video streams is that they are in a frame-based format. By this we mean that all information is broken up into frames, with each frame containing some portion of the video. In our Online Reduction Module, we are able to determine when the start of the next frame is ready for transmission, and can choose not to transmit that frame. The benefits of not transmitting these frames are discussed in detail in Chapter 2.

The Online Reduction Module will have to be notified that it is to perform the frame dropping reduction. This information is received from the Matrix Module when it decides which reductions should be performed on a particular video file. This module

could also be notified to perform frame dropping if the client sends a message to the server indicating that too much information is being sent over the network. If the quality of the network degrades from the time the throughput is calculated, the server may send too much data for the client to handle. If the client becomes overloaded, it can simply drop the frames itself, but we would like to notify the server so that the network bandwidth will not be wasted.

## 5.6 Video Streaming Module

This module serves the purpose of delivering the appropriate bit-stream to the requesting client. Frames from the selected video are passed received by this module and are sent directly out of the server to the client.

In addition to transmitting the video frames, this module also listens for messages from the client. If the client becomes overloaded with data, it will send a message to the Video Streaming Module, and then frame dropping can take place in order to prevent overloading the client. Another message from the client indicating that it is not overloaded will halt any additional frame dropping that was initiated because of the overload condition.

## 5.7 Configuration Module

There are several aspects of our system that must be controlled by a knowledgeable server administrator. The Configuration Module is the tool that this administrator will use to manage the video delivery server.

The main feature that the server administrator must control is the structure of the reduction matrix. It is the responsibility of the administrator to ensure that the proper reductions are placed in the appropriate locations of the matrix, so the clients will receive

the proper video for their particular system. This module will allow for flexibility in the system, and the ability for new device and network modalities to be added to the matrix. For example, if a new class of devices is released that supports 16-bit color, a new entry in the matrix can be made for such devices so the quality of video delivered can be maximized for that device. This feature of added functionality will allow for the system to change based on emerging device and network technologies.

## 5.8 Registration Module

The Registration Module is another interface to be used by the server administrator. When a new video is to be added to the system, any offline reductions must be performed before the video is published for use by the client. In order for this module to determine which reductions need to be performed, it must look at the settings the administrator set in the Configuration Module. If there are references to offline reductions in the reduction matrix, then these reductions must be performed at registration time. As each of the reductions is performed, the Registration Module notifies the Video Storage Module of the reductions and their respective file names corresponding to the new video's name.

For the Video Storage Module to select the appropriate file for a given video, we devise a naming scheme that will allow us to identify the reductions performed on each video. We use a simple scheme that appends a letter to the end of the filename representing which reduction or reductions have been performed on the specific file.

# CHAPTER 6
# IMPLEMENTATION

While many topics are discussed in this thesis, it was not practical to implement everything. Work was done in the past [9] on a system similar to this research. One problem with the previous research is the fact that it was only implemented in a simulated environment. Our job is to realize this system for use on an actual mobile device.

Since the main goal was to get a system that worked on a mobile platform, a great deal of emphasis was placed on the client side of the system. This differs from the original research, which placed most of the emphasis on the server side of this problem. However, our research did require a fairly complicated server to provide the desired services to the requesting clients. We now discuss the details of both the client and the server.

## 6.1 Server Implementation

The server plays a key role in the idea of a video streaming system. It is the responsibility of the server to receive requests, and service those requests accordingly. Additionally the server must maintain all the videos in such a way that they may be identified and retrieved easily by both humans as well as the software. While this server excludes some sophisticated features, it does provide a basic framework to which additional features may be added later.

6.1.1 General Execution Flow

We start with a broad description of what happens when the server program is executed. Figure 6.1 shows the various states in which the server may operate. From the main screen, if either the matrix setup or register buttons are pressed, the program displays dialogs to gather information from the administrator and display any appropriate output to the user. If the start button is pressed, the server begins the process of waiting for requests and no further setup may be done without stopping the server.

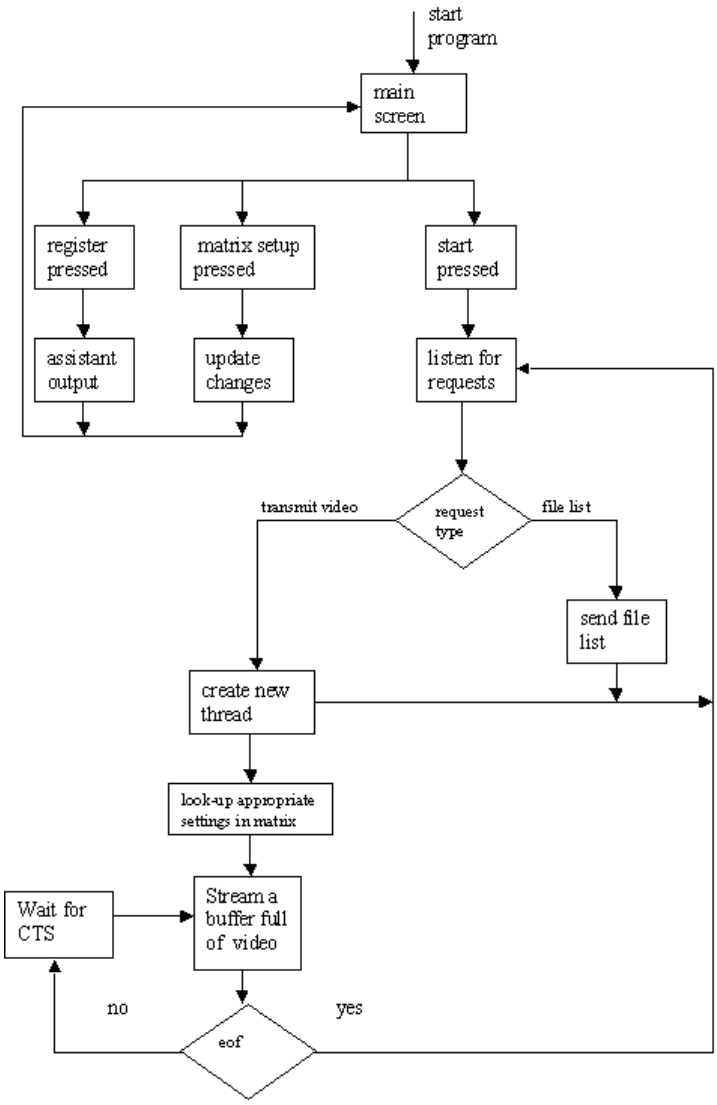


Figure 6.1. Server program flow

### 6.1.2 Matrix Implementation

As discussed previously, the matrix is the heart of the video delivery system. Without the matrix we have no way of determining the appropriate reductions to apply to each device and network. One important consideration with the matrix is the ability to allow change in the future. The implementation of the matrix has in place all entries necessary for the three major reductions—size reduction, color reduction, and frame dropping. Due to limitations of existing reduction algorithms, size reduction is the only one considered for this implementation.

At the heart of the matrix implementation is a simple two-dimensional array of matrix elements. The size of the array is based on the number of device and network combinations that are to be represented. Each entry in the array represents the settings for a given device-network modality. The matrix elements contain the following values:

- Width,
- Height,
- Bit-rate,
- Number of colors, and
- Whether or not to allow frame dropping

The two primary times the matrix structure is used are during video registration and during a request for video transmission. During registration the server uses the matrix to see all possible video formats that will be needed by the devices using the system. The settings for each matrix entry are retrieved, and data regarding the settings for each file are deduced from this information. When a request for video is received, the

server must look up the device-network modality combination in the matrix to decide which file to transmit to the client.

With new networks and devices coming to market every month, adaptability is paramount in any software system. In order to allow for changing environments, updating the matrix structure was one design feature. In order to update the number of devices or networks supported, few changes need to be made to the server. The matrix array size must be increased accordingly, and new matrix elements with the desired settings must be added to the array. Finally the user interface must be updated to reflect these changes in the matrix.

### 6.1.3 Request Handling

Once the server has been started, the program initiates a blocking receive on a socket connected to a port known by all clients. This port number may be any port not in use by the computer system. In this simplified version of the server, only two types of requests are supported—send file list, and transmit file.

#### 6.1.3.1 Send file list request

Before the remote client can select a video to stream, it must first know which videos are available on the server. It accomplishes this task by sending a request for the list of files currently registered on the server.

As a new video is registered on the server, the filename is added to a file containing all other registered videos. When the server receives the request for the file list, it simply reads in the comma-separated file and transmits the contents to the remote client.

6.1.3.2 Transmit file request

After the client has requested the file list, it now has the option to choose one of those files for streaming. This is accomplished by sending a request for a file to be sent. Several pieces of information must be transmitted along with this request. The name of the video is transmitted as well as the device and network types. The device and network types are used as indices into the matrix. Once we find the appropriate entry in the matrix, we can extract the relevant settings that are expected for this particular device and network combination. This information is combined with the name of the video to produce the filename to be transmitted back to the client. A discussion of how the filenames were chosen is found in Section 6.1.4. After the appropriate filename is determined, the server reads the file and transmits the data to the client.

6.1.4 File Metadata

One difficult part of the server was devising a strategy to identify the contents of each file stored on the server. Future implementations of this project may be able to utilize the functionality of the MPEG-7 standard, but since the standard has not been finalized at this time it is not used in this project.

As a workaround, we chose to use a file naming strategy to provide this metadata. Any descriptive information about the file is found in the filename. This way, a glance at a particular filename will reveal all the encoding settings used on that video. The key feature of this naming scheme is that it must be compatible with our matrix implementation. The matrix provides us with the settings required for the requesting client, and we need a way to find the file with those exact settings.

The naming strategy chosen is quite simple. To form the name of the content reduced video, we take the name of the original video file, and then append the settings

used for the encoding of that video. For example, let us look at the registration of the topgun.mpg video for a particular entry in the matrix with desired height 240, width 120, and bit-rate of 200000. The resulting filename for the reduced video would be topgunw240h120b200.mpg. One glance at the filename reveals all relevant settings used to encode the video. Also, the server easily locates the file by looking up the values of the matrix and appending them to the name of the file being requested.

A nice feature of this naming scheme is the expandability. If new reduction settings are created, the filename can be extended, and the server's functionality will not be disrupted. For example, if we wished to implement a color reduction scheme, we could simply add the number of colors present in the video represented by this file.

### 6.1.5 User Interface

An important feature of any software is the interface that it provides to the user. In our case, the user is a knowledgeable server administrator, but we still must provide intuitive ways for them to configure the server appropriately.

#### 6.1.5.1 Main screen

Figure 6.2 shows the main interface provided to the server administrator. The main screen allows the administrator to set the IP address and port for the server to listen for requests. Typically this value is fixed, but for research and testing purposes this feature is provided. From this screen it is easy to access both the video registration page and the matrix setup page. These features are discussed later in this section. The final option the administrator has from this page is to start the server operation. Once the start button has been pressed, the server is operational and is listening for client requests. Details on how the server serves requests are discussed in Section 6.2.2.2.

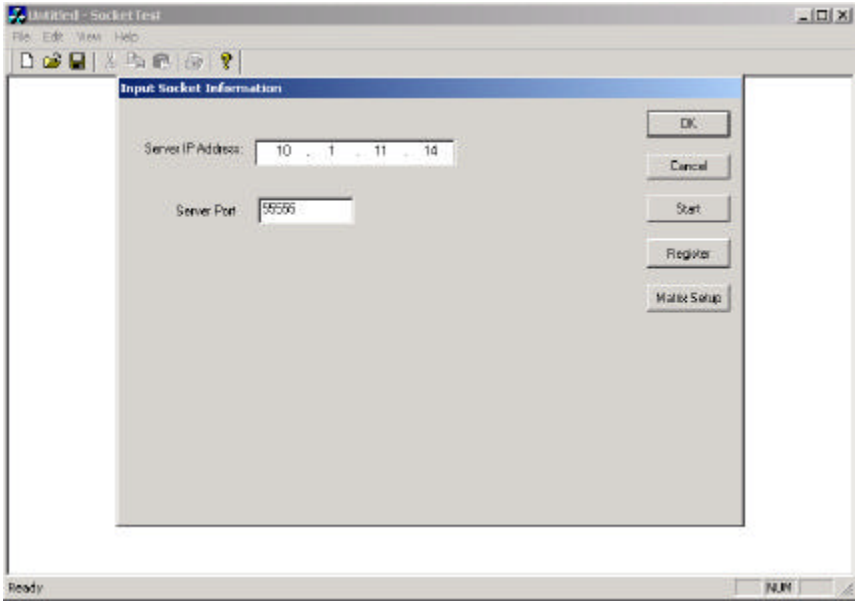


Figure 6.2. The server's main screen

6.1.5.2 Registration screen

To introduce new videos into the system, the administrator must have some way of notifying the system a new video should be added. Figure 6.3 shows this screen and its features. One thing about this screen is simplicity. There really is not too much that must be done here other than specify the file name and categorize the amount of motion in the video. The amount of motion in the video is purely subjective on the part of the administrator, and this feature is not fully implemented in this thesis. In future research, it will be used to help determine the ideal reductions that should be performed on a particular video stream.

From this screen there are only two options from which to choose. One may choose to register a video by entering the filename and amount of motion, or one may cancel the operation to return to the main screen. In the event that the filename entered is not valid, the user is notified and asked to enter a valid filename.

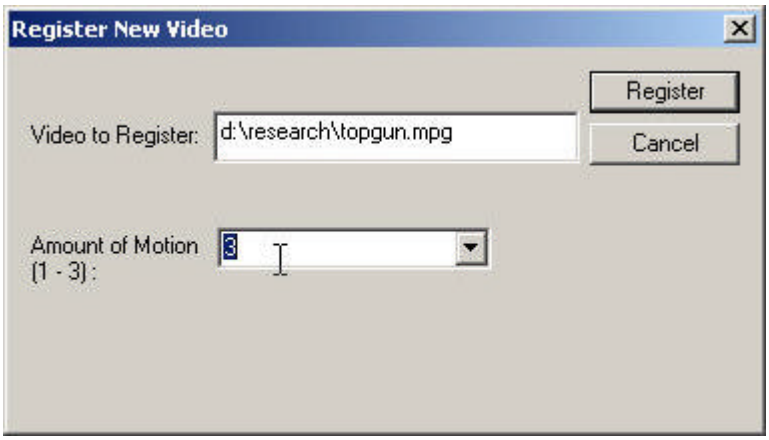


Figure 6.3. Video registration screen

6.1.5.3 Registration assistant screen

Pressing the Register button from the Registration screen notifies the server that a new video should be added to the list of available streams. Since we are serving several different clients we must provide reduced video to them, which require the storage of several version of the file. Since this thesis does not explore how to reduce the content of video, we use the publicly available TMPGEnc [22] tool to achieve video reduction. With several versions of each file, all with different video settings, we had to devise a naming scheme which would give enough information to the administrator to use the proper settings for each video file. Figure 6.4 shows the Assistant screen and a sample output.

Examining the output shown in Figure 6.4 there are three issues to consider. The first part of the filename is the name and path of the original file that is to be registered. Following the filename is the character “w” standing for the width of the new file. The numeric value immediately following is the desired width of the new file. Next is the “h” character, which is the indicator that the height value is to follow. The numeric value

immediately following is the desired height of the new file. The last character in the filename is “b,” which notes the bit-rate. The numeric value following is the desired bit-rate of the new file in kilobits per second. By simply looking at the name of each file to be created, the administrator can use the TMPGEnc utility to produce the necessary files for proper server operation.

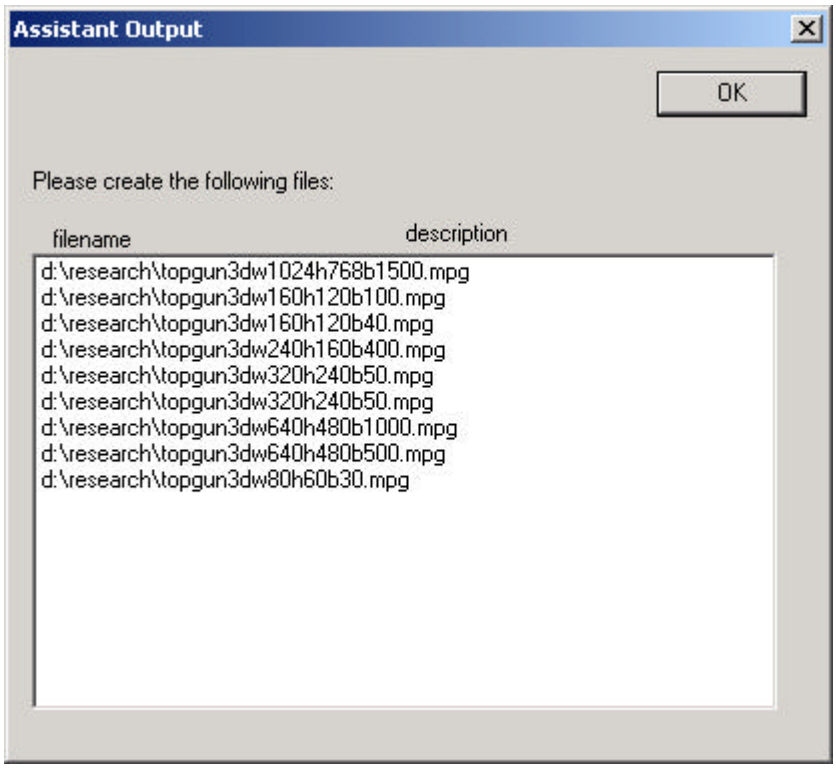


Figure 6.4. Assistant dialog output

6.1.5.4 Matrix setup screen

The Matrix setup screen allows for easy viewing and updating of the reduction matrix. For our implementation this screen is directly tied to the TMPGEnc utility, however changes could easily be made to accommodate a different matrix setup. From this screen the administrator has the ability to change settings they feel are appropriate for the given network and device modality. If they find that the given settings are not

adequate, the dropdown menus provide quick access to appropriate values. Once values have been changed, the update button must be pressed to store the changes to the matrix. One restriction for this project is that the matrix must be configured before the server is started. Once the server begins receiving requests, changes to the matrix are not allowed. The rational behind this is that in our research setting we do not want to go back and redo all the static reductions performed on previously registered videos. If this product were to be marketed commercially, appropriate changes to the software could be explored. Figure 6.5 shows the matrix setup dialog.

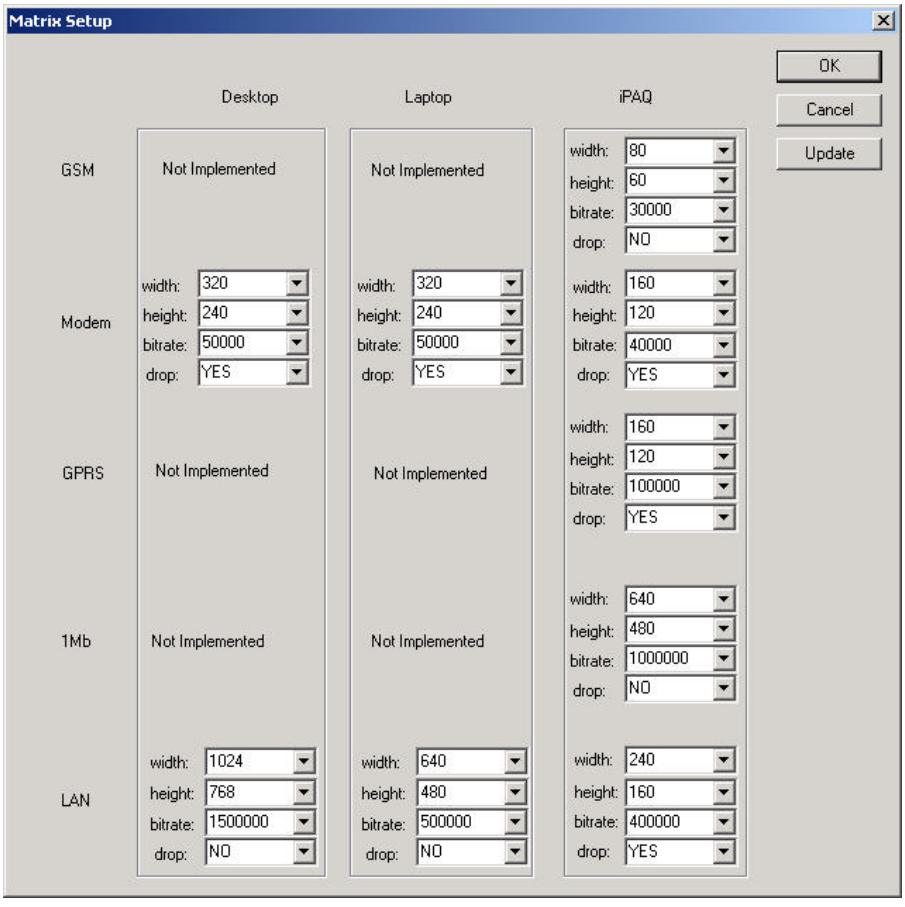


Figure 6.5. Matrix setup screen

### 6.1.6 The TMPGEnc Tool

Due to the limits on the size and scope of this thesis, writing algorithms to alter video streams was beyond the definition of this problem. However, some form of video reduction was needed to test the implementation. To achieve reduction in video, the publicly available tool TMPGEnc[22] was used for all video reductions.

TMPGEnc provides an excellent interface to the user who wishes to specify the settings for a particular video stream. User adjustable settings include resolution, bit-rate, frame-rate, and audio bit-rate. With just these settings, we were able to produce video with significantly reduced size. Of course, with the reduction in size, also comes the reduction in video quality. However, this reduction is necessary to fit the device and network modalities.

Overall, the TMPGEnc utility served its purpose for this thesis. Ideally, we would like to have a program that can perform all types of reductions discussed in this thesis. This is a project that can be considered for future research.

## 6.2 Client Implementation

The main goal of this research was to provide streaming video to a mobile device over a wide variety of networks. Similar research has already been done in this area, but was limited to a simulation environment. Therefore, it was critical to get some form of streaming video on an actual client device, which has been accomplished in this research.

### 6.2.1 PocketTV Player

To achieve our end result of playing video on a remote device, some software to interpret the MPEG video stream is required. Instead of writing our own code, which

would have taken much longer than we had, we choose to use a freely available open source program called PocketTV [23].

In its raw form, PocketTV is capable of playing back local files and is also capable of streaming from remote locations. One problem with the built in streaming functionality is that is uses the hypertext transfer protocol (HTTP), which adds a significant overhead to the data being transmitted. Since the streaming functionality of this product is inadequate for our purposes, intense modifications were made to the existing program.

6.2.1.1 Stream input handlers

As documented by the PocketTV website [23], if any protocol other than http is to be supported, the developer must implement a stream input handler (SIH) for this purpose. Since we chose to use the User Datagram Protocol (UDP), we had to implement the code to handle this new type of stream. The reasons for choosing UDP are discussed further in Section 6.3. Necessary portions of this code must include methods for retrieving data from the remote location and supplying that information to the player. The PocketTV player makes procedure calls to the SIH, and the appropriate data is returned from those calls. This portion of the project was no easy task, as not many people have attempted this type of development.

6.2.1.2 System development kit (SDK)

Also included with PocketTV is an SDK to allow for developers to write their own programs using the PocketTV decoding engine. Since our research required only simple features, and is not being marketed as a commercial product, we chose to modify an existing portion of code. The code we modified is a simplified version of PocketTV, and is called the simple player. Named appropriately the simple player allows for the

playback of streaming video, without all the features of the commercial PocketTV player. For our purposes, the simple player is more than adequate.

To make this program work for us, we had to add one main screen to the player, as well as a progress bar. The purpose of the progress bar is to indicate the percentage of the streamed file received by the device. The main screen is for the user to select the type of device and network, as well as to retrieve the list of available files and initiate streaming.

### 6.2.2 Client Execution Flow

Figure 6.6 shows the general flow of execution on the client side of our system. There is minimal processing with this system until the user presses the open button, at which time the file download is initiated. Once the streaming has been initiated, the main screen disappears, and the user may make no further changes.

#### 6.2.2.1 Get file list

For the user to select a file for streaming, that user must be presented with the files available on the server. Since the number of files may change from one access to another, the client must request a list of the currently available files on the server. To achieve this, the client opens a socket connection with the server and sends the string "`getFileList.`" The client then performs a blocking receive on the socket waiting for the server's response. The server response is a comma-separated list of filenames that are currently available. The client parses this list and displays the names in a user selectable window.

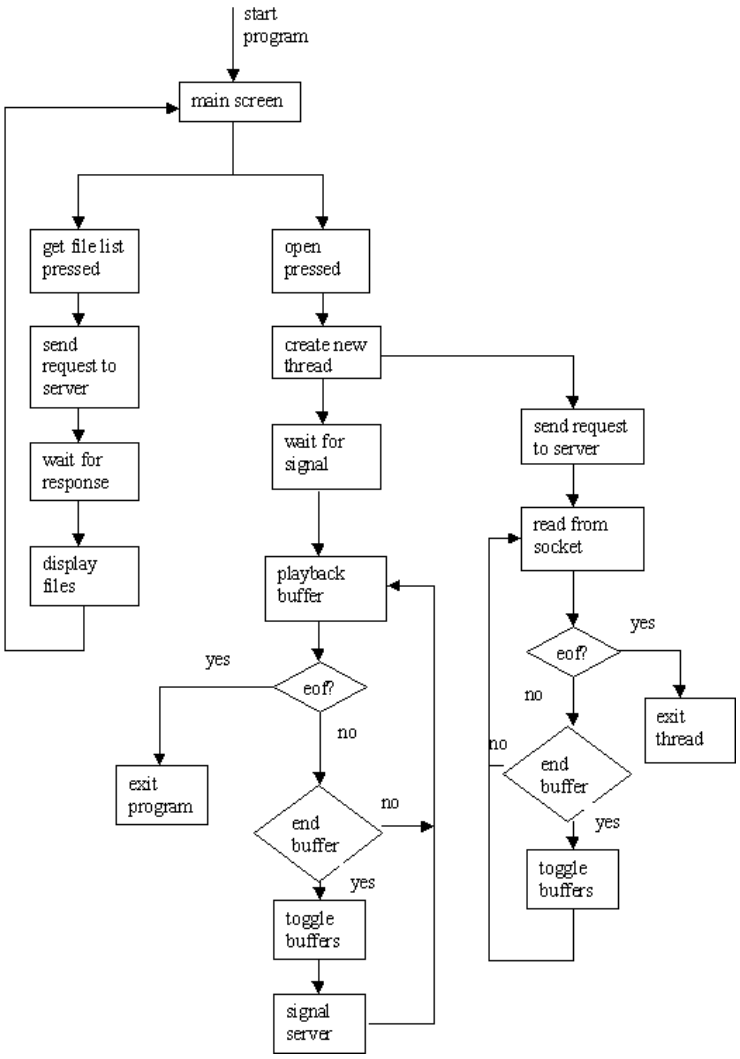


Figure 6.6. Flow of execution for the client application

6.2.2.2 Open file stream

After the list of available files has been retrieved, the user may now select any of the available files from the file list window. As the user presses the open button, several issues must be addressed before video playback can begin.

The first issue that must be addressed concerns the streaming and playback of video simultaneously. To accomplish this task, threading had to be used. One thread is used to download the file from the server, while another thread is responsible for the

playback of that file. In most cases, some buffering of the file is required, and therefore there must be some communication between the two threads. The download thread must notify the playback thread when enough bytes have been buffered to ensure proper playback.

The second issue to deal with is the amount of data we should buffer before playback can begin. This is calculated by subtracting the amount of time to playback the file from the amount of time to download the file. For example, if a file takes 50 seconds to download, and 30 seconds to playback, we need to buffer for 20 seconds to ensure the player does not run out of data.

The final issue deals with the device and network modality of our client device. For the server to provide the client with the appropriate video stream, the client has to notify the server of these parameters. When the user presses the open button, the string "transmitFile <filename> <deviceType> <networkType>" is sent to the server. Upon receipt of this message the server can parse the filename device and network type from this information, ensuring the proper video will be transmitted to this client.

### 6.2.3 User Interface

The target platform for deployment of our system is the Compaq iPAQ, running the PocketPC operating system. Development for this type of platform differs mainly in the user interfaces that can be developed. Since we have a small screen, and no keyboard we have fewer options than with a desktop development. To combat the size problem, drop-down boxes and scrollable windows are used to conserve space and make operation simpler for the user. Figure 6.7 shows the main screen of our client.

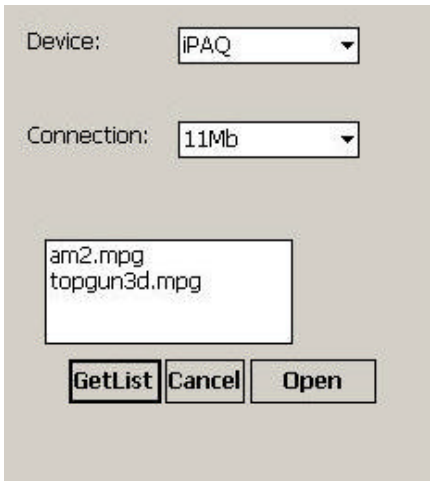


Figure 6.7. Client main window

The user interface presented during video playback was designed with simplicity in mind. The user should not be cluttered with superfluous information, so we provide the video as well as the download progress of the video. The download progress meter can help the user judge if they have selected the appropriate network connection. If the download does not complete before the video is done, the network bandwidth is not adequate to stream the requested video. This may be a problem with the server settings, or perhaps problems with the wireless network. Figure 6.8 shows a snapshot of the client during video playback.

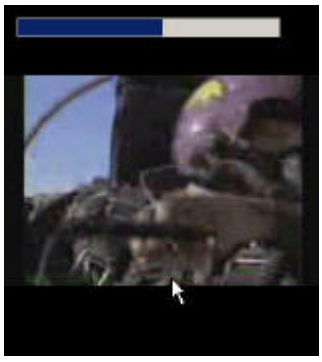

Figure 6.8. Client during video playback

## 6.3 Transport Protocol

For this thesis, we had the choice of transmission control protocol (TCP) or user datagram protocol (UDP). The PocketTV system we used as the basis for our implementation supports either HTTP or FTP transmission, both of which are TCP based protocols. Since TCP is a connection oriented delivery system, there exits some overhead in the setup of network communications. The payoff for this overhead is reliable service. If a packet is transmitted, it is guaranteed to reach the destination host. However, in a video delivery system time is critical. If there was an error in transmitting a video frame, retransmission of that frame may come after the frame was to be displayed causing unsatisfactory results.

Since our implementation is directly using networks that are limited in bandwidth, we need every speed advantage we can get. Therefore, we chose to use UDP for the transmission protocol in our client-server system. UDP packets are not guaranteed to reach the destination, but as discussed previously, if the packet does not arrive on time it is useless anyway. So, we tradeoff guaranteed packet delivery for lower overhead and faster speed.

# CHAPTER 7
# PERFORMANCE MEASUREMENTS

The research and implementation of the system designed in this thesis provide a good theoretical basis. However, this work would not be complete without demonstrating the benefits of using this video delivery system. If the system did not show significant advantages over traditional video delivery, then much more work would be required. While all details of the architecture were not implemented, and thereby reducing the effectiveness of the system, dramatic results were still noted with our system implementation.

## 7.1 Methodology

Determining the testing procedures for this project was quite challenging. Several issues had to be taken into consideration. These issues include which device to test, which networks to test, and which video reductions to use.

### 7.1.1 Test Device

For the device in our system we chose to test the Compaq iPAQ 3650 equipped with the PocketPC operating system from Microsoft. This device was chosen for its relatively powerful computational abilities as well as its widespread availability and popularity.

Since our implementation did not deal with the color reductions, a full color device was chosen as to avoid transmitting full color video to a black and white device.

We also wanted to deliver high quality video for testing and evaluation by other individuals, and the iPAQ provided the ability to do so.

### 7.1.2 Networks Tested

While our architecture should support any number of network modalities, we were forced to limited the scope of these networks for testing and evaluation purposes. Our choice of networks to test was quite simple—use the networks available to us with the iPAQ. In our case, we test an 802.11b wireless LAN, 56 kbps modem, 28.8 kbps modem, and a 9.6 kbps modem. Ideally we would like to have tested a network somewhere in-between the wireless LAN and the 56 kbps modem, but unfortunately we did not have a network connection available to us fitting that criterion.

The wireless card used for testing purposes was a Lucent Technologies (now Orinoco) WaveLAN Silver Turbo. The modem was the Pretec CompactFlash Modem with adjustable baud rate. The adjustable baud rate was a key feature needed for us to limit the connection speed of this modem.

### 7.1.3 Video Reductions

As discussed previously, the reduction strategies were not fully implemented in this system. In fact, an external program, TMPGEnc, was used to create the reduced video files. Since the number of reductions was limited we had to carefully choose how to reduce each video stream for each device and network modality. However, since only the iPAQ was tested, we only needed to focus on each network modality.

In general when choosing the appropriate settings for each video, the throughput of each network was examined. From that value, we could determine a value for the bit-rate suitable for that network. For the wireless LAN connection, the bit-rate chosen was significantly less than the network speed, and this was because the iPAQ cannot handle

both a high bit-rate and a high frame rate. Section 7.2.1 deals with some of these issues. With the lower bandwidth connections, it was impossible to reduce the bit-rate enough to match the throughput, which means longer delay times for the user. Experimentation showed that the bit-rate could not be reduced any further than 30 kbps for video, and 32 kbps for audio. Therefore, if we wish to provide audio (and all of our tests do), then the slower connections will be forced to buffer data for longer periods of time.

## 7.2 Performance Results

In this section we discuss all the pertinent results gathered by our tests. These tests show that our system indeed does provide a significant performance increase over traditional video streaming techniques.

### 7.2.1 Standalone iPAQ tests

Before we could make appropriate decisions about encoding settings for our video streams, we had to first determine what the iPAQ was capable of on a standalone basis. We performed tests that showed the relationship between the bit-rate of the encoded video, and the frame rate achieved by the iPAQ for these different bit-rates. Table 7.1 shows the frame rate achieved for various resolution settings. To optimize the video for the iPAQ, we needed to choose the resolution appropriately so as to provide the best quality video. After testing, the 240 x 160 resolution was chosen because of its overall size and frame rate. While we could have used the 160 x 120 setting and achieved a higher frame rate, the picture is much smaller and less desirable in our opinion. Therefore, the 240 x 160 resolution was chosen as the best-case resolution for our video system. We deliver this video to the iPAQ when the wireless LAN card is connected, and slower connections are given appropriately lower quality video streams.

Another important issue for the iPAQ is the bit-rate at which the video is encoded. The device is only capable of displaying a video of a certain quality, and no matter how much more data when send to the device, the quality will remain the same. After several tests, it was found that no noticeable difference in video quality is seen above 400 kbps. In fact, with some resolution combinations, performance can actually degrade after this point. Therefore, we chose the bit-rate of 400 kbps to use in conjunction with the 240 x 160 resolution.

Table 7.1. iPAQ resolution vs. frame rate

| Resolution | Display rate |
|---|---|
| 320 x 240 | 10 frames/sec |
| 240 x 160 | 20 frames/sec |
| 160 x 120 | 30 frames/sec |

7.2.2 Startup Delay Tests

Perhaps the most critical feature of any video streaming system is the delay from the time the user sends the request to the time video playback begins. When this time is too great, the user can become frustrated and not use the system. While quality is also important, our system strives to provide the fastest response time possible to the user of the streaming video system.

To perform this test, we took our four network connections and streamed a video using our system with and without video reductions. The video without reductions was 640 x 480 and had a bit-rate of 1Mbps, settings typical for a desktop system. Intuitively the video without any reductions should take much longer than the videos reduced with our system. As expected, our system reduced the user's startup delay in a significant fashion. Figure 7.1 shows the results of these tests.

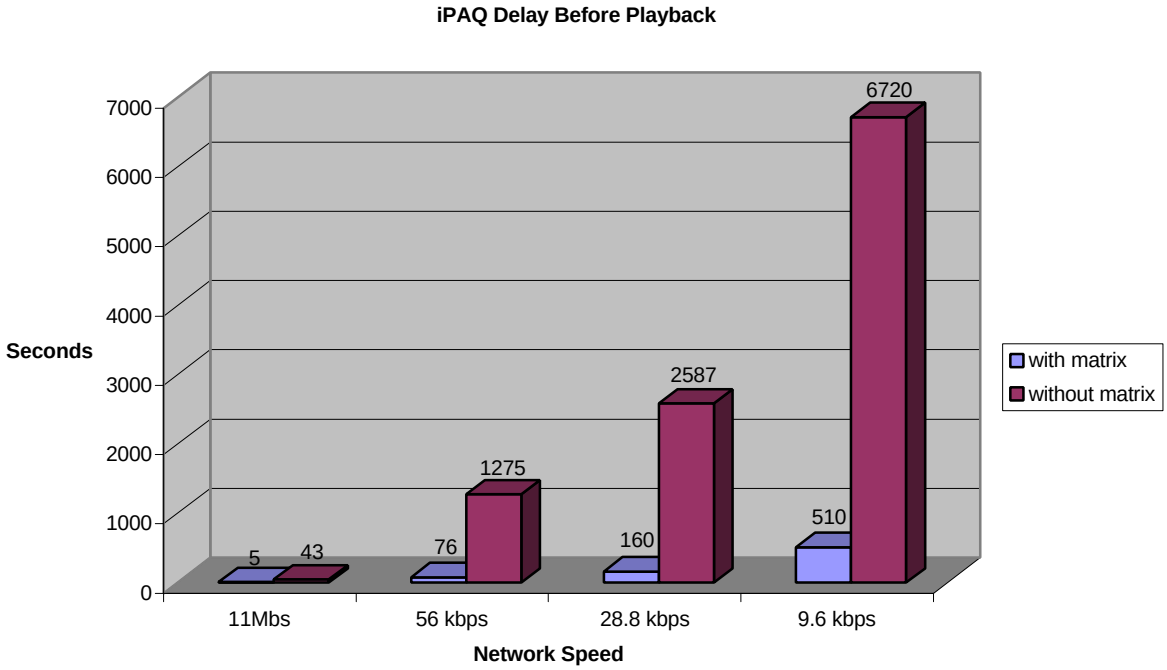


Figure 7.1. Delay before playback begins on the iPAQ

As shown in Figure 7.1, the time savings associated with our delivery system are substantial. While not as noticeable with the 11 Mbps network, the differences become much more apparent with the lower bandwidths. This is due to the enormous reduction in bit-rate the videos undergo before transmission across the slower bandwidth lines.

### 7.2.3 Perceived Quality Tests

While the reduction in startup delay time is quite impressive, it does not come without a price. To reduce the time to download the video, we must reduce the number of bits transmitted to the client and, therefore, reduce the quality of the video stream. Measuring just how much quality has been lost is a difficult metric to obtain. It is a highly subjective matter and, therefore, requires users to examine video and give an assessment of the video quality.

For our tests, we chose to measure three main aspects of the video stream that we feel summarizes the overall video playback experience quite well. The three aspects measured were video smoothness, overall picture quality, and audio quality. While we did not explicitly change the audio quality in any of our tests, network slowdowns and processor loading affects the quality of sound delivered to the user. Additionally these tests only consider the quality of the video. They do not account for the startup delay, as that was tested previously. Table 7.2 summarizes these results.

Table 7.2. User quality ratings

| | Original | | | 11 Mbps | | | 56/28.8 kbps | | | 9.6 kbps | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S | P | A | S | P | A | S | P | A | S | P | A |
| User 1 | 3 | 5 | 4 | 5 | 5 | 4 | 4 | 3 | 4 | 3 | 2 | 3 |
| User 2 | 3 | 5 | 5 | 5 | 5 | 5 | 4 | 2 | 4 | 4 | 1 | 4 |
| User 3 | 2 | 3 | 3 | 5 | 4 | 4 | 3 | 3 | 2 | 3 | 2 | 4 |
| User 4 | 2 | 4 | 3 | 5 | 5 | 4 | 4 | 3 | 3 | 3 | 1 | 3 |
| Average | 2.5 | 4.25 | 3.75 | 5 | 4.75 | 4.25 | 3.75 | 2.75 | 3.25 | 3.25 | 1.5 | 3.5 |

- S - Smoothness, P - Overall Picture Quality, A - Audio Quality
- 1: lowest rating, 5: highest rating

A glance at Table 7.2 shows the quality ratings given by each of 4 unique users of our video delivery system. One interesting part of this data is that the video without any reductions actually received the lowest score for video smoothness. This is due to the high resolution and bit-rate which together only allowed the iPAQ to display about 4 frames per second. One other note is the video stream used for the 56 kbps and the 28 kbps connections were identical. This is due to the inability of the encoder to reduce the video beyond 30 kbps for the video stream. Since we needed to reserve this bit-rate for the 9.6 kbps connection, we chose to encode the 56 kbps and the 28.8 kbps at 40 kbps.

Figures 7.2 – 7.4 display each of the three individual measurements for each of the videos.

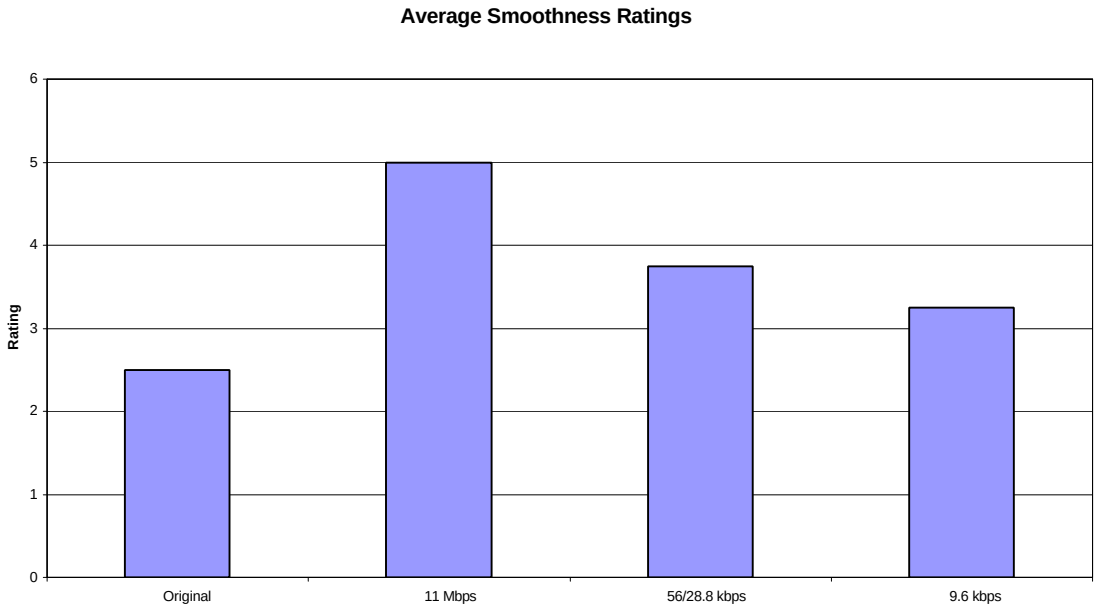


Figure 7.2. Average video smoothness ratings

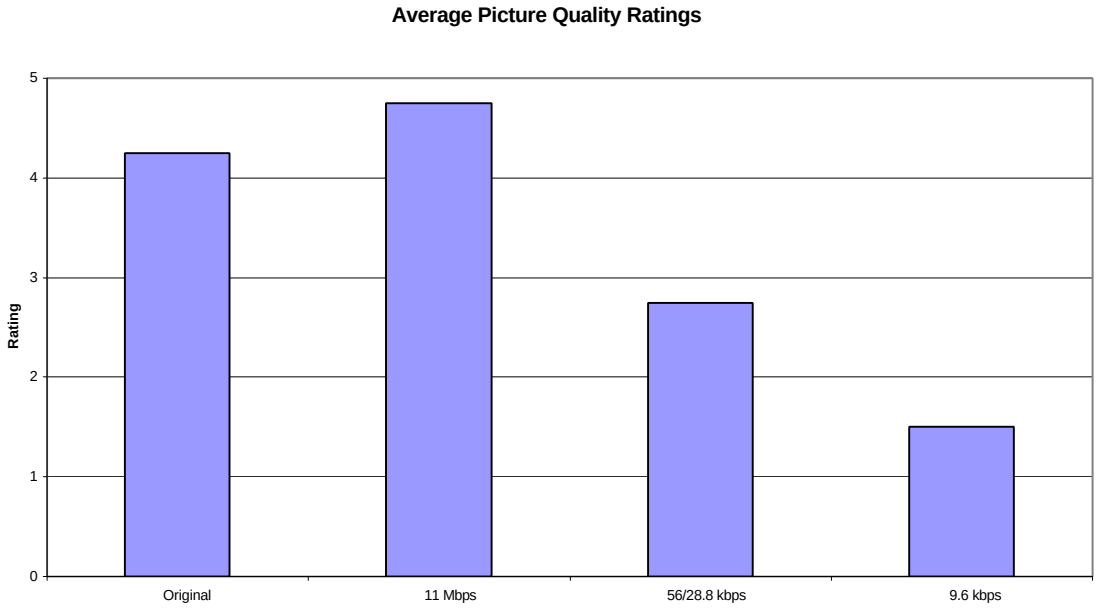


Figure 7.3. Average picture quality ratings

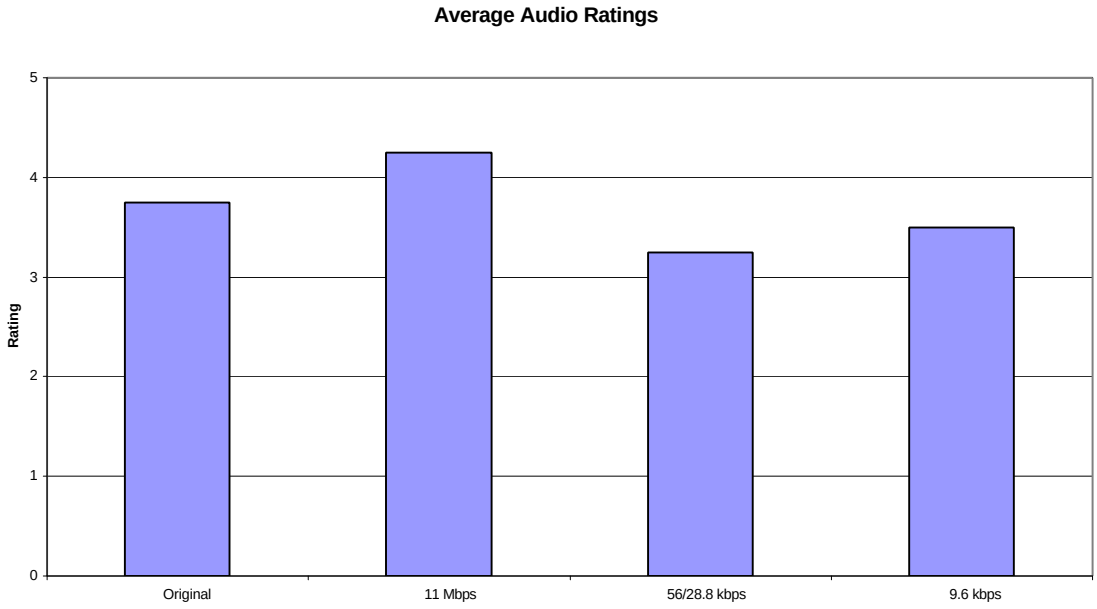


Figure 7.4. Average audio quality ratings

Combining all three of the metrics discussed in this section, we can determine an overall rating for each of the videos. This score represents the video playback experience each user received, and we feel is a good representation of overall video quality. Figure 7.5 summarizes the overall video quality.

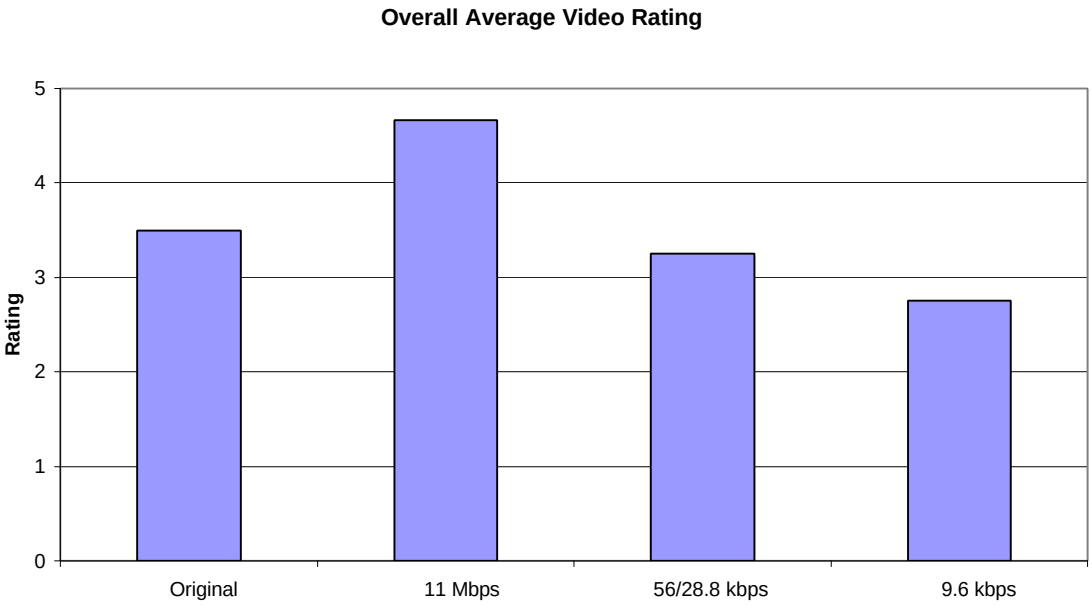


Figure 7.5. Overall video ratings

# CHAPTER 8
# CONCLUSIONS AND FUTURE WORK

## 8.1 Summary of Work Completed

From the beginning of this research project, the desired result was to have a video streaming system available on a mobile platform with varying network capabilities. As the research concludes for this thesis, that task has been accomplished.

While many research projects in this field tend to be theoretical, our system is taken through the implementation stage. There are several projects that have simulated video playback on a computationally poor device with limited bandwidth, but few have implemented such ideas on a real device. Our achievements provide future researchers the ability to focus on other areas of research without having to focus on the actual delivery of the video from server to client.

Perhaps the greatest achievement resulting from this research is the great reduction in the startup delay exhibited on the iPAQ. This reduction allows for multimedia content to be delivered to mobile devices in such a way that the users do not become frustrated with the slow speeds of typical streaming applications. There have been studies showing that users with faster computers are much more productive in a work environment. We feel that with faster video streaming systems the productivity will come in the form of greater use and, therefore, greater profit for content providers.

## 8.2 Future Work

While great strides were taken during this research effort, there is still much to be done to provide the best quality video to the end user. Most of our attention was placed on the delivery of the video, and less on the methodology behind the actual content of the video streams.

One area of research that must be completed to improve this project is video reduction. It was beyond the scope of this research to design and code customized algorithms to reduce the content of the video in our server. Therefore, future efforts need to focus on the format of MPEG video streams and determining ways to reduce the number of bits in the stream. Needed algorithms include color reduction, size reduction, and rate reduction. In our research, we found the need to accomplish all these tasks offline before streaming the video to the client. However, it may be possible to do some of these reductions on the fly, thereby reducing redundant storage on the server. Needless to say, there are several issues involved with this topic that would probably substantiate an entire thesis.

For the reductions to be done correctly, we feel more emphasis should be placed on the MPEG-7 standard, which provides descriptions of the video content. If certain attributes of the video were known—color depth, amount of motion, frame rate, etc.—better decisions about which reductions to deploy could be made.

In addition to the research on how to reduce the content of a video stream, there needs to be research in the area of which reductions are ideal for which situations. In our research we made some assumptions and performed some tests to see which video streams worked well in different situations, but work is needed in this area. In addition to figuring out which reduction or reductions are ideal for a given entry in the matrix, there

may be advantages to doing one reduction before the other. Again, this topic has many avenues that have yet to be explored and could greatly enhance the system set forth in our research.

LIST OF REFERENCES


[1] MPEG homepage. http://www.cselt.it/mpeg/, January 2001.

[2] Wiseman, John. An Introduction to MPEG Video Compression. http://members.aol.com/symbandgrl, January 2001.

[3] Berkeley Multimedia Research Center. MPEG-1 Video. http://bmrc.berkeley.edu/research/mpeg/faq/mpeg1.html, January 2001.

[4] Sikora, Thomas. MPEG-1 and MPEG-2 Digital Video Coding Standards. http://wwwam.hhi.de/mpeg-video/papers/sikora/mpeg1_2/mpeg1_2.htm, January 2001.

[5] Agnew, Grace. Digital Video for the Next Millennium. http://sunsite.utk.edu/video/1.shtml, February 2001.

[6] Shanableh, Tamer and Mohammed Ghanbari. The Importance of the Bi-Directionally Predicated Pictures in Video Streaming. IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, No. 3, March 2001

[7] Simon Fraser University Computer Science Department. Video Compression. http://www.cs.sfu.ca/undergrad/CourseMaterials/CMPT479/material/notes/Chap4/Chap4.2/Chap4.2.html#MPEG, February 2001.

[8] Schroeder, Mark D. JPEG Algorithm and Associated Data Structures. http://people.aero.und.edu/~mschroed/jpeg.html#SECTION2, January 2001.

[9] Sampath, L., A. Helal, and J. Smith. UMA-based Wireless and Mobile Video Delivery Architecture, Proceedings of the SPIE Voice, Video and Data Communications Symposium, Volume: 4210, October 2000, Pages: 103 – 115.

[10] Bormans, Jan, Keith Hill. MPEG-21 Overview. http://www.cselt.it/mpeg/standards/mpeg-21/mpeg-21.htm, March 2001

[11] Wallace, Chris. Is Streaming Video Dead? http://www.zdnet.com/filters/printerfriendly/0,6061,2697806-2,00.html, March 2001.

[12] Robert, P. Max, Ahmed M. Darwish, and Jeffrey H. Reed. MPEG Video Quality Prediction in a Wireless System. Vehicular Technology Conference, 1999 IEEE 49th , Volume: 2, 1999, Pages: 1490 –1495.

[13] Dugad, Rakesh, Narendra Ahuja. A Fast Scheme for Downsampling and Upsampling in the DCT Domain. IEEE Transactions on Circuits and Systems for Video Technology, Volume: 11 Issue: 4, April 2001, Pages: 461 – 474.

[14] PocketPC homepage. http://www.microsoft.com/mobile/pocketpc/default.asp, January 2001.

[15] Scourias, John. Overview of GSM. http://kbs.cs.tu-berlin.de/~jutta/gsm/js-intro.html, April 2001.

[16] Kari, Hannu H. General Packet Radio System. http://www.ee.oulu.fi/~fiat/gprs.html, April 2001.

[17] Helal, Abdelsalam, Bert Haskell, Jeffery L. Carter, Richard Brice, Darrell Woelk, Marek Rusinkiewicz. Any Time, Anywhere Computing. Kluwer Academic Publishers, Boston 1999, Pages: 18 – 25.

[18] Bluetooth homepage. http://www.bluetooth.com, April 2001.

[19] Windows Media Technologies homepage. http://www.microsoft.com/windows/windowsmedia/EN/default.asp, April 2001.

[20] PacketVideo homepage. http://www.packetvideo.com, April 2001.

[21] Emblaze Systems hompage. http://www.emblaze.com/serve/main_page.asp, April 2001.

[22] TMPGEnc homepage. http://www.tmpgenc.com/, April 2001.

[23] PocketTV homepage. http://www.mpegtv.com/wince/pocketty, February 2001.

# BIOGRAPHICAL SKETCH

Kevin Birkett was born January 1, 1977, in Gainesville, Florida, where he has remained a resident his entire life. While pursuing his undergraduate degree at the University of Florida, he enrolled in the combined bachelor's and master's program. He received his bachelor's in computer engineering in May 2000 and finishes his master's in engineering in December 2001.

Besides his schoolwork, Kevin is a commercial pilot with multi-engine and instrument ratings. He enjoys all outdoor activities, especially hunting, fishing, golf, football, and basketball.