Light Microscopy:

A Prototype For A Remote Image Diagnosis System

by

Alexander B. Zakharov

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Computer Science and Engineering

and Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 21, 1997

Author: ....
Department of Electrical Engineering and Computer Science
May 21, 1997

Certified by: ....
C. Forbes Dewey, Jr.
Thesis Supervisor

Approved by: ....
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Light Microscopy:
A Prototype For A Remote Image Diagnosis System

by
Alexander B. Zakharov

Submitted to the
Department of Electrical Engineering and Computer Science

May 21, 1997

In Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in Computer Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

Telemedicine provides doctors with access to remotely-located resources and empowers diagnosis and consultation over a network connection. Key objectives of telemedicine are to decrease the cost and increase the efficiency of medical service. This thesis presents an example of an advanced telemedicine application. A prototype system for remote image diagnosis using an optical microscope was designed and implemented. This system allows for remote control of a light microscope in the 10Mbps switched ethernet environment. The design uses H.261 video streaming technology and a high-resolution video camera for image transfer. To manipulate the microscope remotely, two separate TCP/IP sockets over the same ethernet connection are used: one for image transfer, the other to control the different components of the microscope. This system can also operate with improved real-time images if higher-speed connections such as ATM are available.

Thesis Supervisor: C. Forbes Dewey
Title: Professor, MIT Mechanical Engineering Department

Acknowledgements

First and foremost I would like to thank Professor C. Forbes Dewey. His ideas, support, and guidance were crucial to making this project a reality. Without a doubt, both professionally and personally Professor Dewey is the best supervisor I have ever worked with. I am honored to have had the opportunity to experience his influence and intelligence during the past year.

I would also like to thank my friends, particularly Dave, Hyung, Nate, Noah, and college women, for sharing with me social and cultural experiences of Boston and surrounding areas, which provided for a natural balance between academia and leisure.

Of course, my family was also quite supportive, although they have no idea as to what I have been doing for the last five years at MIT.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

During the past decade the world has witnessed the explosive growth of communication networks and distributed applications. One of the advantages of that growth is that today it is possible to perform a variety of business and science-related tasks remotely. Examples of such remotely-accomplished tasks include video-conferencing, telemedicine, telecommuting, distance learning, shopping, and many others.

This thesis is an example of an advanced telemedicine application. The ideas behind telemedicine have been in existence for over thirty years [1], however the actual implementation of such ideas was limited by inadequate technology. In the last few years, the technology has finally caught up with some of the telemedicine requirements. Not surprisingly, telemedicine today is a topic associated with active research and heated discussions. There are a number of works assessing technical feasibility of the distributed telemedicine applications [2], [3], general design and evaluation of such applications [4], as well as economic feasibility of telemedicine [5].

Naturally, telemedicine is not merely an interesting, hypothetical proposition. Aside from theoretical approaches and speculations, a number of telemedicine projects were implemented and installed at different medical facilities all around the world [1], [6], [7], [8], [9], [10].

The majority of the implemented systems focus on providing researchers and doctors with access to remotely-located medical data. The basic mode of operation of such systems allows a researcher or a doctor to download data from a central storage facility, analyze that data, and send the results back at a later time. The time difference between acquiring information and sending the results of the analysis back is usually quite significant and is commonly measured in minutes or hours.

Telemedicine becomes truly useful when acquisition and analysis of information is performed in real-time. In fact, some of the existing projects do address the real-time mode of operation. However, as of now, the capabilities provided by such systems are still quite limited. In particular, some scenarios allow a researcher to perform a real-time analysis of a single stable image. Another common application allows a group of researchers to discuss a previously acquired image by utilizing one of the existing video conferencing systems. In such applications a copy of the image to be analyzed is downloaded and displayed at each of the remote sites and a group of doctors discuss the image by sending annotations across the network which are displayed on the pre-loaded image. Alternatively, the participants can use low-resolution videoconferencing to discuss the pre-loaded image.

The purpose of this thesis is to illustrate how an image diagnosis, a procedure which is generally accomplished on site, can be performed remotely and in real-time. In particular, the thesis will focus on microscopy and will present the design and implementation of a prototype system for a remote image diagnosis using a state-of-the-art light microscope.

There are a number of reasons why light microscopy has been chosen for this project. Most importantly, microscopy-oriented applications are ubiquitous in both medicine and science. One example of a microscopy application in medicine is the analysis of pathology images, which in itself is an important and well-developed field. Science makes heavy use of microscopy in such areas as semiconductor industry and research. In addition, choosing microscopy for the prototype, provides us with a solid base for extending the scope of the project to such applications as X-ray analysis, mammography, and electrocardiography.

Chapter 2 Background

2.1 Expectations

Before presenting the implementation details of the remote image analysis system, it is useful to describe what should be expected of such a system. In this section a number of factors that had a significant influence on the design of the prototype are discussed.

Time: To accomplish a paradigm shift from local image analysis to remote operations, it is necessary for the remote manipulation of the microscope to be performed in real-time. Defining what real-time video means from the technical perspective requires evaluation of two parameters: the latency and the frame-rate. In this project a researcher or doctor should be able observe the changes in the image of a specimen under the microscope while he is controlling the microscope from a remote location. Thus, ideally, the project requires a video-streaming system with a frame-rate of over 15fps, and a latency of less than .1 second.

Resolution: The resolution of images obtained from the micrsocope is another major factor in the design of the system. Ideally, we want to provide a user of with the capability of displaying high-resolution (e.g. 1280x1040) 24-bit color images in real-time; realistically such performance is unattainable with today's technology. Today it is possible to either acquire single high-resolution (over 1040x960) images which typically incurs a significant latency, or transmit and display a real-time lower resolution (e.g 352x288) video stream. We will present a more detailed discussion of the resolution requirements in later sections.

User Interface: The system is targeted towards users with a primarily medical background and users who do not necessarily possess an extensive computer knowledge. Consequently, the complexity of the system should be hidden behind a user-friendly interface.

Adaptability: Ideally, we would like the system to be network- and platform-independent. This would allow for an easier integration with future technologies and higher-speed networks such as those based on the Asynchronous Transfer Mode (ATM) protocols.

Scalability: Finally, we will strive to have the capability to extend, as seamlessly as possible, the scope of the applications of the system to such areas as X-rays, electrocardiography or echocardiography.

2.2 Available Technology

This section describes how the state of today's technology allows us to address the expectations listed above.

2.2.1 Microscopy

Naturally, any system for the remote control of a medical apparatus is worthless unless the apparatus itself can provide high-quality data to the researchers or doctors who need it. Also, such a system is virtually impossible to implement, if the apparatus does not allow for a non-manual control. Today, there a number of providers who deliver highest quality computer-controlled microscopy-oriented equipment.

For example, one of the more eminent companies in the medical field is Carl Zeiss (Germany). In the field of microscopy, Zeiss offers a series of products for the following areas: light microscopy, microscopes for the microelectronics industry, scanning probe microscopy, fluorescence correlation spectroscopy, histology, and electron microscopy. Most of the products offer both manual and high precision computer control and are flexible enough to accommodate for a variety of applications.

2.2.2 Video Cameras

The introduction of digital cameras based on Charge Coupled Devices (CCD) technology [11] marked a major step in the development of high-quality video products. Digital video cameras capable of producing 1280x1040 24bit color images are easily located in today's market. One can also obtain digital cameras capable of achieving 1600x1200 resolution. Admittedly, the high end of the curve necessitates a serious monetary investment, but as the time progresses the maximum resolution will keep increasing and affordability of the products will improve.

2.2.3 Computer Networks

By definition telemedicine applications involve a transfer of acquired images to a remote viewing location where the analysis of these images is performed. Computer networks have become the media of choice for an efficient and reliable data delivery. The crucial factor affecting the speed of the transfer of image data is the network's bandwidth. This bandwidth becomes particularly important when the transfer is done in real-time.

For the purposes of this discussion it is necessary to distinguish between packet-switched and circuit-switched networks. The major factor that differentiates these two types of networks is bandwidth control. In particular, a circuit-switched network is capable of providing point-to-point connections with guaranteed bandwidth, allocated on a per-connection basis. The bandwidth allocated for each connection depends on the type of application and current bandwidth capacity of the network. A packet-switched network, on the other hand, does not provide any sophisticated bandwidth management and as such does not guarantee the availability of any particular bandwidth for communicating parties. Thus, in the packet-switched network the communicating parties can not predict or request any particular bandwidth for their connection.

The most commonly used network standard today is a non-switched ethernet, which supports up to 10Mbps transfer rates. Realistically, non-switched ethernet, does not even come close to providing 10Mbps bandwidth; in fact the real-world rate of such network can be as low as 100 bytes/second. Such performance is inadequate for the majority of modern distributed real-time image analysis applications (see Video Compression subsection). The emerging networking standards which have been gaining popularity are ATM [13] and Fast Ethernet [14]. Both of these standards are based on switched point-to-point technology. ATM supports up to 155Mpbs point-to-point connections, while Fast Ethernet supports up to 100Mbps of dedicated bandwidth.

As mentioned above, both ATM and Fast Ethernet provide point-to-point connections, however, ATM's bandwidth control is superior to that of Fast Ethernet. In particular, ATM allows to allocate specific dedicated amounts of bandwidth for different types of applications. Thus, in ATM, the network traffic is controlled on a per-application basis, where each application can run over the same or different physical connections.

2.2.4 Video Compression

The high-performance point-to-point networks like ATM and Fast Ethernet are still not powerful enough to provide sufficient capacity handle such bandwidth-intensive applications as real-time video. Table 1 illustrates sample bandwidth requirements for a real-time video transmission at different resolutions. Examining the specifications in the table makes it clear that effective compression and decompression techniques are necessary to accommodate real-time video streaming.

Image Resolution	Minimum Bandwidth
350 x 280	67.2 Mbps
640 x 480	210.4 Mbps
800 x 600	329.6 Mbps
1024 x 768	540 Mbps
1280 x 960	844 Mbps
1280 x 1024	900 Mbps

Table 1: The bandwidth required for real-time (30 fps) transfer of full color (24 bit) images.

There are a number of standards for video compression and transmission available today. These standards define technology that make the scenario depicted in Figure 1 possible.

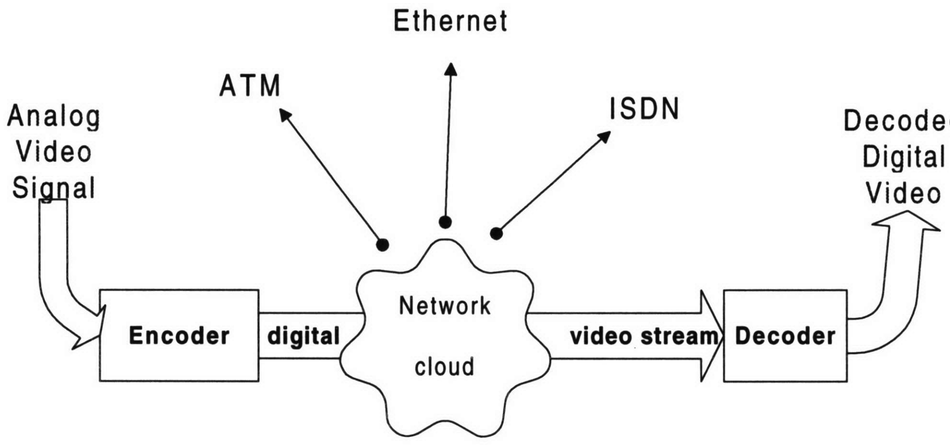

Figure 1: A generic video streaming scenario.

As depicted, a generic video streaming system takes an analog video signal as its input (commonly NTSC). That video signal is digitized and compressed by an encoder which is typically implemented in hardware. The output of the encoder - compressed digital video - is streamed across one of the existing network standards. A large percentage of today's videoconferencing products work only in the ISDN environment, but products capable of supporting other types of networks are also available. Finally, the digital video stream arrives at the decoder end where the signal is decompressed, and usually displayed on a computer screen.

In the next three subsections we will discuss the three most prominent encoding/decoding schemes that drive the video-conferencing/streaming industry: MPEG, MJPEG, and H.261.

2.2.4.1 Motion Picture Experts Group (MPEG)

MPEG[12] was originally designed to store sound and motion-video data on standard audio Compact Discs. This standard is very suitable for write-once read-many applications, since compression under MPEG is considerably more complicated then the decompression. Although the design of MPEG was oriented towards write-once read-many applications, MPEG toady is also used for video streaming/teleconferencing related applications. However, the practicality of MPEG based teleconferencing is somewhat limited by MPEG's relatively high bandwidth requirements.

There are a number of different versions for the MPEG standard. Regardless of the version, successful MPEG-based video transmission strongly depends on the availability of a sufficiently high bandwidth in the underlying network. In particular, MPEG-1 allows for a data rate of up to 1.86Mbps and is capable of supporting a 352x240 resolution at 30 frames per second. At the maximum resolution the data rate of as low as ~1Mbps can be used, provided that audio is abandoned, but the quality deteriorates noticeably for lower rates. MPEG-2, on the other hand, operates in 2 - 15Mbps range and can accommodate a resolution of 704x576 at 30 frames per second. A new generation of MPEG standard (MPEG-4) is currently is development.

MPEG compression algorithm makes heavy use of interframe dependency and motion prediction. In an MPEG video stream three types of frames can be distinguished: reference frames and two kinds of difference or delta frames. A reference frame is coded by itself, without any motion prediction, or dependency on any other frame in a stream. The delta frames on the other hand are coded with a reference to either past frames (P type) or past and future frames (B type). It is due to this frame dependency that a relatively high latency in an MPEG based transmission is inevitable, regardless of the type of network, encoding board, or CPU being used for a project.

2.2.4.2 Motion Joint Photographic Experts Group (MJPEG)

The MJPEG compression scheme emerged as a simpler alternative to MPEG technology. This alternative treats each frame of a video stream independently and performs a separate JPEG compression on each picture.

JPEG [13,14] defines a means of digitally compressing still pictures by breaking up a picture into 8x8 blocks. The JPEG compression algorithm consists of four main stages: a transformation stage, a lossy quantization stage, and two lossless coding stages. At the heart of the algorithm (transformation stage) is a mathematical process known as a Discrete Cosine Process (DCT) which is applied to each of the 8x8 blocks. Ironically, the application of a DCT actually increases the size of the data on which it operates. The quantizer causes a controlled loss of information, and the two coding stages further compress the data. Typically, the data, resulting from the application of the four main stages, is further compressed with some conventional compression techniques such as Huffmann coding. JPEG files are commonly characterized by the JPEG compression ratio, which quantifies the ratio between data before and after JPEG encoding scheme was applied.

There are a number of advantages to using MJPEG as compared to using MPEG. Most importantly, MJPEG is considerably simpler and achieves fast, low latency results. In addition, MJPEG provides a fairly uniform bit-rate and can be followed by a conversion to MPEG if necessary.

There are disadvantages to using MJPEG as well. In particular, MJPEG files are considerably larger than MPEG files. Furthermore, MJPEG files exhibit poorer video quality than MPEG at the same overall compression ratio.

2.2.4.3 H.261 Standard

H.261[16], published by International Telecom Union (ITU) in 1990, is a widely-used international video compression standard for teleconferencing and videophone applications. Originally it was designed for an ISDN-based environment, and as a result H.261 operates with datarates which are multiples of 64Kbit/s. Today, companies tend to provide H.261-based teleconferencing products in both ISDN and LAN environments.

H.261 supports two resolutions: Quarter Common Interchange Format (QCIF) at 176x244, and Common Interchange Format (CIF) at 352x288. Given a reasonable bandwidth (e.g. 10Mbps) and a well-designed implementation of the standard, frame rates of up to 25 frames per second can be achieved at CIF resolution.

The coding algorithm itself is a hybrid of inter-picture prediction, transform coding, and motion compensation. The algorithm has characteristics similar to those of MPEG, but a few important differences exist. For example, just like MPEG, H.261 exhibits some interframe dependency, but all motion estimations in H.261 are based on previously coded and already-transmitted pictures. The major contributor to MPEG's latency is its reliance, among other things, on future frames for motion predictions. H.261, on the other hand, was designed specifically for real-time video conferencing applications and its latency was reduced to a minimum.

Chapter 3 Development Environment

The first part of this section describes an underlying network on which the remotely-controlled microscope is installed and configured. The second part illustrates how the microscope is integrated into the network. In particular, the technology necessary for video transmission, microscope control, and image acquisition will be discussed in detail.

3.1 Network Configuration

Figure 2 illustrates the network environment in which the microscope is configured. The architecture of the network is an example of a star topology centered around an ATM switch provided by Whittaker/Xyplex. This ATM switch provides a maximum bandwidth of 155Mbps. In addition to ATM support, the switch also has a capacity to handle switched ethernet traffic. In principal, if only two switched ethernet boards are used, the switch will support up to fifteen machines attached to it through switched ethernet interface. Any two of the fifteen machines can establish a point-to-point connection at a dedicated 10Mbps. The current implementation of the remote-controlled microscope utilizes switched ethernet. Finally, the switch is connected to the MIT campus network through one of its ethernet ports.

The current implementation of the remote-controlled microscope utilizes switched ethernet; however, a migration to ATM networks will be possible in the future (see Chapter 6 “Future Directions”).

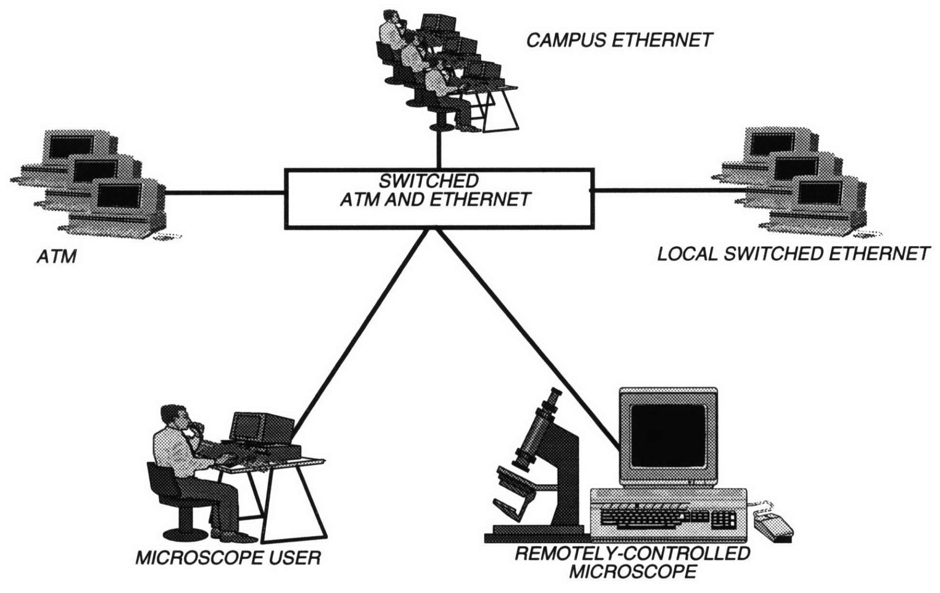

Figure 2: The network environment of the remote microscopy project.

3.2 Microscope Configuration

3.2.1 Optical Microscope

The optical microscope used for this project is Axioplan 2, manufactured by Zeiss. The Axioplan 2 is a state-of-the-art microscope allowing for both computer-based and manual control. The information about different features of Axioplan 2 can be found in

Appendices A.1, A.2, and A.3. For additional information consult the manuals [17, 18] that came with the microscope.

The microscope has a dedicated Windows95 PC attached to it via an RS232 port. The PC attached to the microscope can run a Zeiss application [18] that allows the user to control Axioplan 2 locally. Table 2 shows different software-controlled functions that are available. Naturally, all of these functions can be performed manually if so desired.

Microscope component/function	Mode of control
Objective	Selection of an objective turret position by clockwise/anticlockwise indexing.
Focus	Actuation of the motorized focusing (Z) drive.
Work/Load	Fast lowering and raising of the stage for specimen loading (the focus is preserved).
Condenser	Opening/Closing the aperture diaphragm and clockwise/anticlockwise indexing of the revolving condenser wheel.
Filter Magazine	Clockwise/anticlockwise indexing of the four positions each of filter wheel 1 and 2, and display of the filter transmittances.
Diaphragm	Control of the transmitted light, by the luminous field diaphragm
Lamp Intensity	Control of brightness of the 100W halogen lamp.

Table 2: The computer-controlled functions of the microscope.

Zeiss has also developed a set of RS232 codes [19] on which their application is based. It is the execution of these codes that accomplishes the control functions described in Table 2. To utilize this type of control, one has to gain access to the RS232 interface via which Axioplan 2 is attached to the PC. Typically, the RS232 interface can be controlled through an associated COM port. Any operating system provides means of reading from and writing to the system's COM ports. In technical terms, the port is considered "open" when one can read from it and write to it. Thus, to control the microscope via RS232 codes, one has to open a COM port and directly send the codes to the microscope by writing to that COM port. The responses from the microscope's hardware can be read from the same port.

3.2.2 Stage

The microscope comes with a separately-controlled stage [20], also manufactured by Zeiss. This stage can be moved at different speeds in the x-y plane. Like the microscope, there are three ways to control the stage: manual, through stand-alone application provided by Zeiss, and directly by sending RS232 codes to a different (from the one the microscope is attached to) COM port.

As mentioned earlier, Zeiss supplied the control application [15] for the microscope. The software for controlling the stage is integrated with the software that controls the microscope. Thus, by running the application provided by Zeiss on the attached PC, a user can control both the microscope and the stage. In order to control the stage, either through Zeiss software or by directly sending RS232 codes to it, it has to be connected via an RS232 cable to one of the remaining COM ports of the attached PC.

3.2.3 Camera

A dedicated high-definition image capture digital camera (FUJIX HC-2000 [21]) was mounted on the Axioplan 2. The HC-2000 is an outstanding digitizing camera with three

2/3-inch 1,300,000-pixel CCDs which allows for the capturing of 1280x1000 24 bit color images. In addition to a standard SCSI output through which high resolution images are captured, the camera has a regular NTSC output, thus enabling real-time viewing on a TV monitor. The NTSC output is limited to 352x240 resolution.

FUJIX provides a software package that allows a rudimentary control of the camera through a SCSI interface. With this software package, images of different resolution can be previewed on a computer screen, captured and saved to a hard disk.

Chapter 4 Implementation

4.1 High-Level Design

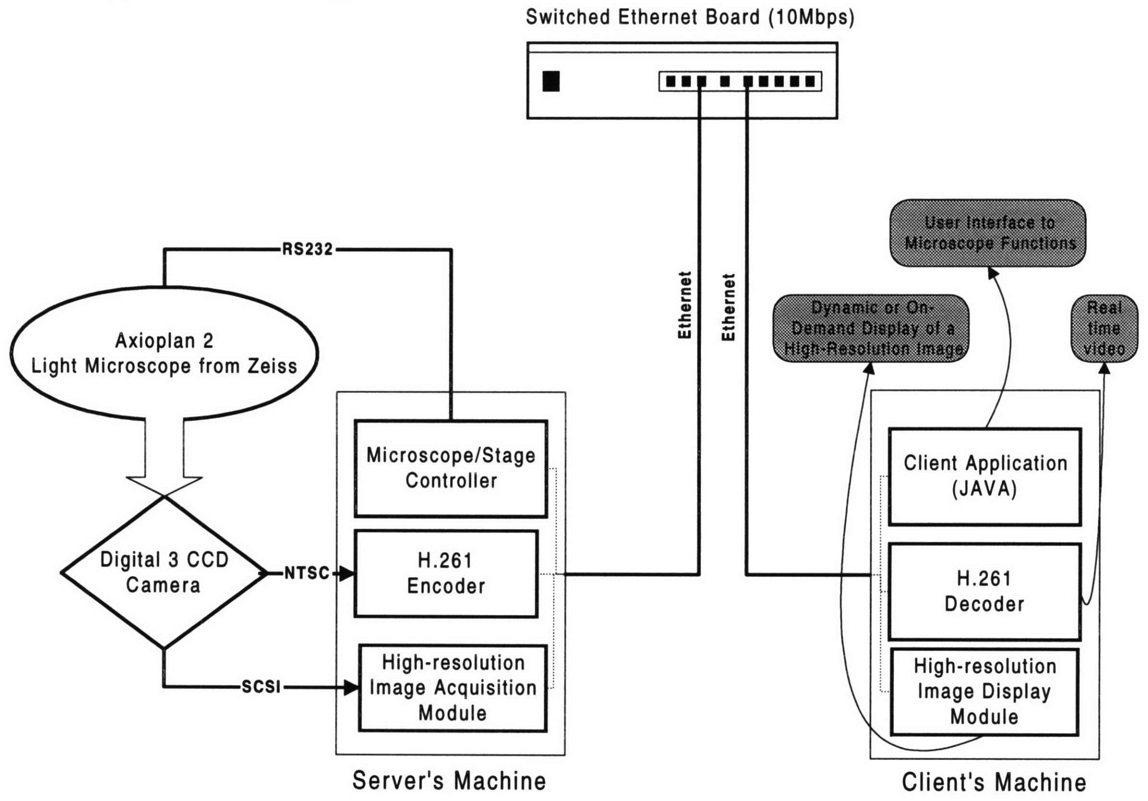
Switched Ethernet Board (10Mbps)
Ethernet
Ethernet
RS232
Axioplan 2 Light Microscope from Zeiss
Digital 3 CCD Camera
NTSC
SCSI
Microscope/Stage Controller
H.261 Encoder
High-resolution Image Acquisition Module
Server's Machine
User Interface to Microscope Functions
Dynamic or On-Demand Display of a High-Resolution Image
Real time video
Client Application (JAVA)
H.261 Decoder
High-resolution Image Display Module
Client's Machine

Figure 3: Controlling the microscope and stage: high-level schematic view.

Figure 3 illustrates the scenario for the remote control of the microscope. At a remote location a user will specify the functions for the microscope to execute, and will be able to observe a real-time low-resolution (352x288) video signal coming from the microscope through the video camera. Hence, the operator as she performs different control functions, can see the changes in image on her computer screen.

It is important to realize that the remote control of both the microscope and stage is done in parallel with video-streaming. In fact, two separate TCP/IP sockets (associated with the same physical ethernet connection) will be used: one for the microscope and stage control, the other for video-streaming. The socket for the microscope control will require very low bandwidth and will not affect the quality of the video stream.

As shown in the figure, the camera also has a high-resolution (1280x1000) SCSI output, through which high-resolution still images can be acquired. These still images can be delivered to the user at different points of the program execution.

The implementation of this system consists of three separate well-defined subparts: control of the microscope and stage functions, control of video-streaming, and control of acquisition and delivery of high-resolution still images. From a technical perspective, acquisition and delivery of still images is the most straightforward part of the project; however, it is the only module that was not implemented. Chapter 6 presents the reasons why this module was not implemented and delineates how it can be implemented in the future. What follows is a detailed discussion of the microscope control and video control.

4.2 Microscope and Stage Control

4.2.1 Overview

The remote control of the microscope and the stage is based on a client-server paradigm. Both the client and the server are written in JAVA. The programming language JAVA was chosen because it is a high-level platform-independent object-oriented language which is aggressively used by many programmers today and will be utilized for many more projects tomorrow. In order to actually control the stage and the microscope, the server has to go through a hardware controller written in C. The details of the server will be discussed in a greater detail shortly.

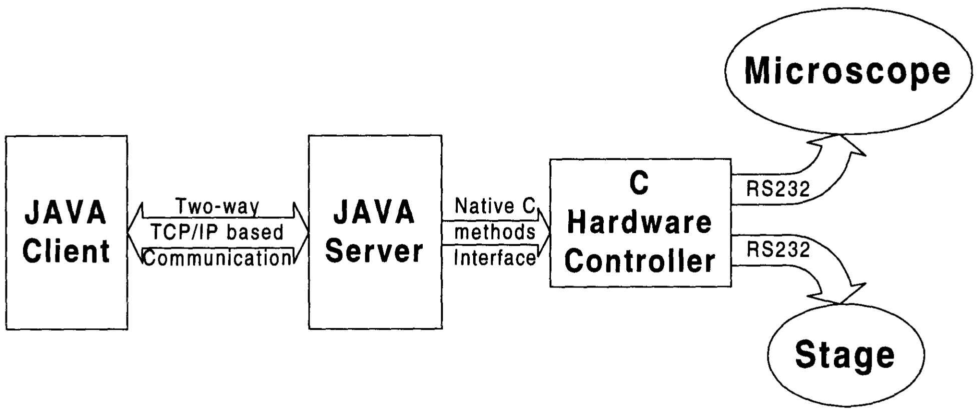

Figure 4: Controlling the microscope and stage: specifics.

Figure 4 illustrates a general control flow of the system. Ideally, the client can run on any computer with JAVA support and a network connection. Realistically, the client's underlying hardware and operating system is limited by the video transmission technology that is used for the project. Please refer to the video control section for more information on video control and transmission. The client provides a graphic user interface (GUI), that gives the user an ability to establish a network connection with the computer attached to the microscope, and specify various microscope and stage functions. Once a particular function is specified the client sends an appropriate code to the server.

From a high level perspective, the server loops indefinitely, waiting for a client to connect. Once the connection is established and certain hardware checking is successfully completed, the server waits for the client to send codes. These codes specify different functions that a user is trying to perform with the microscope or the stage. The server examines each code sent by the client and executes the corresponding function on the microscope or stage. Depending on the specified function, the server may send a reply code to the client, which typically uses it to update the GUI.

The server, like the client, can run on any computer with JAVA support and a network connection, and its underlying hardware is limited by the video transmission system used for the project. However, there are further limitations imposed on the server. In particular, the server needs to be able to write to and read from the COM ports, to which the stage and the microscope are attached. The COM port support differs among various computer architectures, and thus is platform dependent. JAVA, on the other hand, is a high-level platform-independent programming language, that does not have an explicit support for the COM port interface. In this implementation of the project, the COM port interface is written in C and the JAVA server has to call native C methods when it actually needs to send instructions to either the stage or the microscope.

4.2.2 Client-Server Protocol

The remote control of the microscope and the stage requires an agreed-upon set of communication codes between the client and the server. In addition, we developed a protocol defining the actions that take place on both the client and the server in response to various codes. This protocol covers three major areas: initialization, microscope control, and stage control.

4.2.2.1 Initialization and Hardware Checking

Status Codes	Description
AxioCode.IS_SERVER_ALIVE	The server receives this code from the client and sends confirming response back to the client (see the next table entry).
AxioCode.SERVER_ALIVE	The server sends this code to the client in response to AxioCode.IS_SERVER_ALIVE
AxioCode.MICRO_OK	Once the client connects, the server tries to execute simple test funcitons on the microscope. If successful, it sends this code to the client.
AxioCode.MICRO_COM_ERROR	Once the client connects, the server tries to open a COM port associated with the microscope, if unsuccessful, it sends this code to the client.
AxioCode.MICRO_ERROR	Once the client connects, the server tries to execute simple test functions on the microscope, if unsuccessful, it sends this code to the client.
AxioCode.STAGE_OK	Once the client connects, the server tries to execute simple test funcitons with the stage, if successful, it sends this code to the client.
AxioCode.STAGE_COM_ERROR	Once the client connects, the server tries to open a COM port associated with the stage, if unsuccessful, it sends this code to the client.
AxioCode.STAGE_ERROR	Once the client connects, the server tries to execute simple test functions with the stage, if unsuccessful, it sends this code to the client.

Table 3: The client server communication codes during the initialization.

Hardware checking becomes an important feature in the remote mode of operation, since a medical researcher needs a way to verify that the stage and the microscope are actually present and connected at the remote site. This checking occurs when the client connects to the server. The client checks for three factors: presence of the server, status of the stage, and status of the microscope. Table 3 provides the specifics of the initialization protocol.

The above error codes are handled in the following manner. The operation of the program will not continue if the server is not responding. If the server is on-line, the status of the microscope and the stage are to be checked. If this check reveals that, for whatever reason, the microscope is not operational, the server terminates the connection with the client with an appropriate error message. However, if the microscope is operational while the stage is not, the program will still continue, and client's functionality will not include the movement of the stage in the x-y plane.

4.2.2.2 Microscope Codes

Once the connection between the client and the server has been established and the usability of the microscope was confirmed, the client can execute the microscope's functions remotely by sending the appropriate codes to the server. These codes, together with the corresponding actions of the server, are presented in Table 4.

Code for Microscope Function	Server's Action
AxioCode.OBJECTIVE_LEFT	Rotate the objective nosepiece turret one position to the left. Get current values for the objective, the filter wheel 1, the filter wheel 2 and send those values to the client to update the GUI.
AxioCode.OBJECTIVE_RIGHT	Rotate the objective nosepiece turret one position to the right. Get current values for the objective, the filter wheel 1, the filter wheel 2 and send those values to the client to update the GUI.
AxioCode.WHEEL1_LEFT	Rotate filter wheel 1 turret one position to the left.
AxioCode.WHEEL1_RIGHT	Rotate filter wheel 1 turret one position to the right.
AxioCode.WHEEL2_LEFT	Rotate filter wheel 2 turret one position to the left.
AxioCode.WHEEL2_RIGHT	Rotate filter wheel 2 turret one position to the right.
AxioCode.LAMP_INC	Start increasing the brightness of the lamp.
AxioCode.LAMP_DEC	Start decreasing the brightness of the lamp.
AxioCode.DI_INC	Start increasing the luminous field diaphragm.
AxioCode.DI_DEC	Start decreasing the luminous field diaphragm.
AxioCode.COND_INC	Start increasing the transmitted light condensor.
AxioCode.COND_DEC	Start increasing the transmitted light condensor.
AxioCode.MOVEMENT_DONE	Stops previously started increase/decrease of the lamp's brightness, diaphragm, or condensor.
AxioCode.Z_UP	Wait for the client to send a value (an integer), specifying the speed at which the stage is to be moved. Once that value is received, start adjusting the focus by moving the stage up (closer to the objective) at the speed, specified by the value.
AxioCode.Z_DOWN	Wait for the client to send a value (an integer), specifying the speed at which the stage is to be moved. Once that value is received, start adjusting the focus by moving the stage down (farther away from the objective) at the speed, specified by the received value.
AxioCode.Z_STOP	Stop adjusting the focus.

Table 4: The microscope control codes received by the server from the client.

4.2.2.3 Stage Codes

Given that initialization process has been completed successfully and the server has confirmed that both the microscope and the stage are present, the client can start controlling the stage remotely. Table 5 presents the codes that accomplish this task.

Stage Control Code	Server's Action
AxioCode.X_LEFT	Wait for the client to send a value (an integer), specifying the speed at which the stage is to be moved. Once that value is received, start moving the stage along the X axis (negative direction) at the speed, specified by the received value.
AxioCode.X_RIGHT	Analogous to AxioCode.X_LEFT only for positive movement along the X axis.
AxioCode.X_STOP	Stop the movement of the stage along the X axis.
AxioCode.Y_DOWN	Analogous to AxioCode.X_LEFT only for the movement along the Y axis (negative direction).
AxioCode.Y_UP	Analogous to AxioCode.X_LEFT only for the movement along the Y axis (positive direction).
AxioCode.Y_STOP	Stop the movement of the stage along the Y axis.

Table 5: The stage control codes received by the server from the client.

4.2.3 Client-Side Implementation

4.2.3.1 Development Tools

The client side implementation is based on Java Development Kit (JDK 1.1) - Sun's version of JAVA [22]. In addition to extending JDK's classes, the client's implementation makes use of the Graphic Java Toolkit (GJT) [23] - a graphics enhancing software addition to Sun's standard JAVA kit. GJT extends Java Abstract Window Toolkit (AWT) and features over thirty custom components for GUI development. Notably, GJT is based on JDK 1.0; for this project GJT was slightly modified so as to conform to JDK 1.1.

4.2.3.2 User Interface

Figure 5 depicts what a user will see when the client is invoked on a remote workstation. As shown in the figure, a user can manipulate the following components of the microscope: the focus, the lamp, the diaphragm, the condenser, the objective nosepiece, the filter wheel 1, and the filter wheel 2. Also, the user can move the stage in the x-y plane. The components in the upper section of the GUI help to connect to the server, and exit the program. The very bottom of the GUI shows the current hardware status of the server.

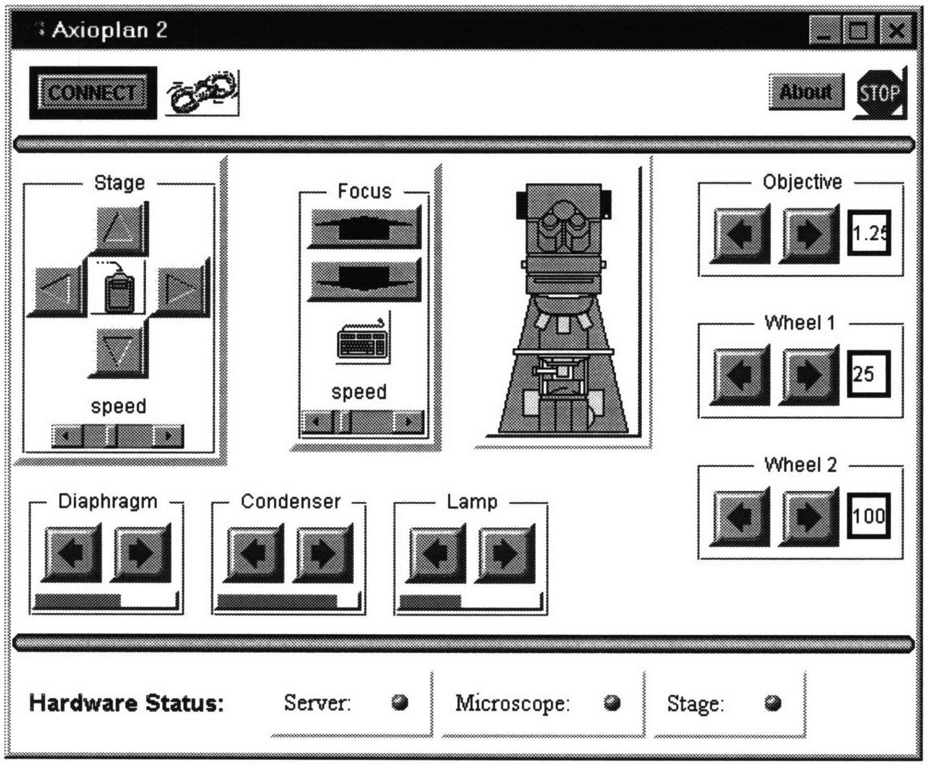

Figure 5: The Client's Graphic User Interface (GUI).

4.2.3.3 Client's Class Structure

Figure 6 illustrates how the client's classes fit into an overall structure of JAVA classes, while Figure 7 shows which JAVA interfaces were implemented for this project. The client's core functionality consists of laying out the GUI components, listening and updating these components, and communicating with the server. We will discuss these tasks one at a time.

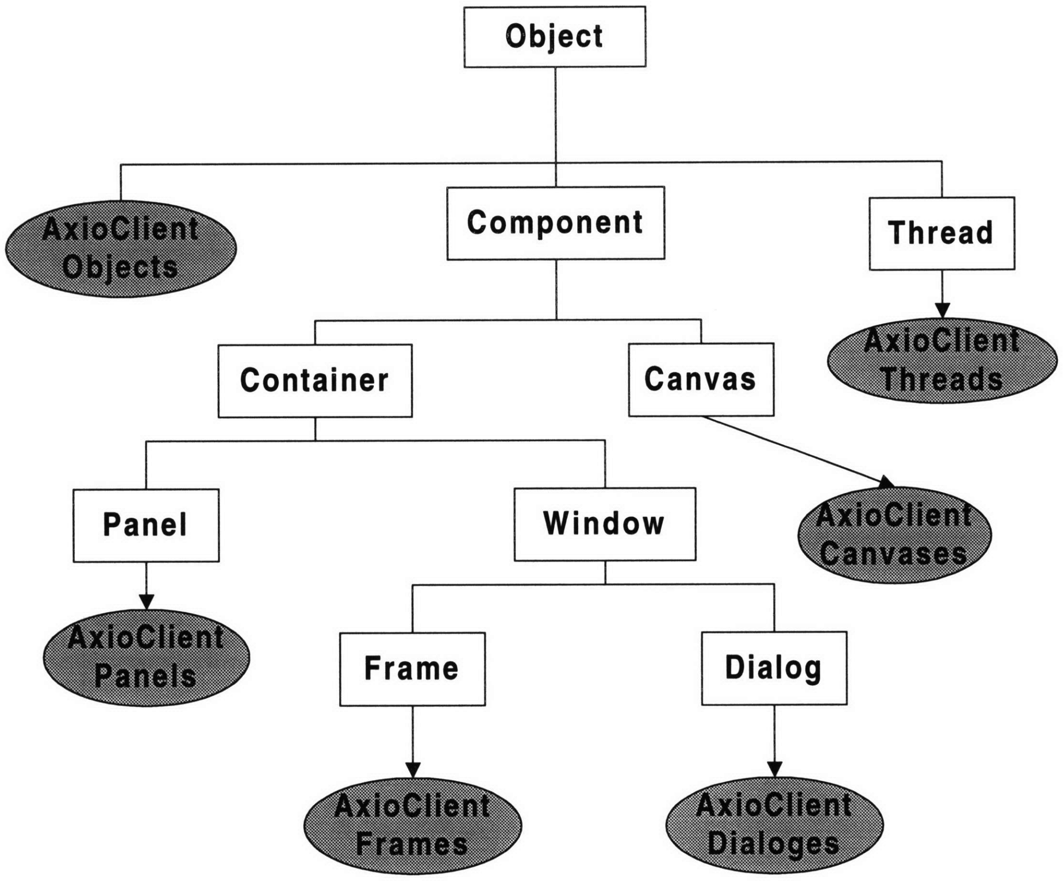

Figure 6: Client's classes in their relation to the Java classes.

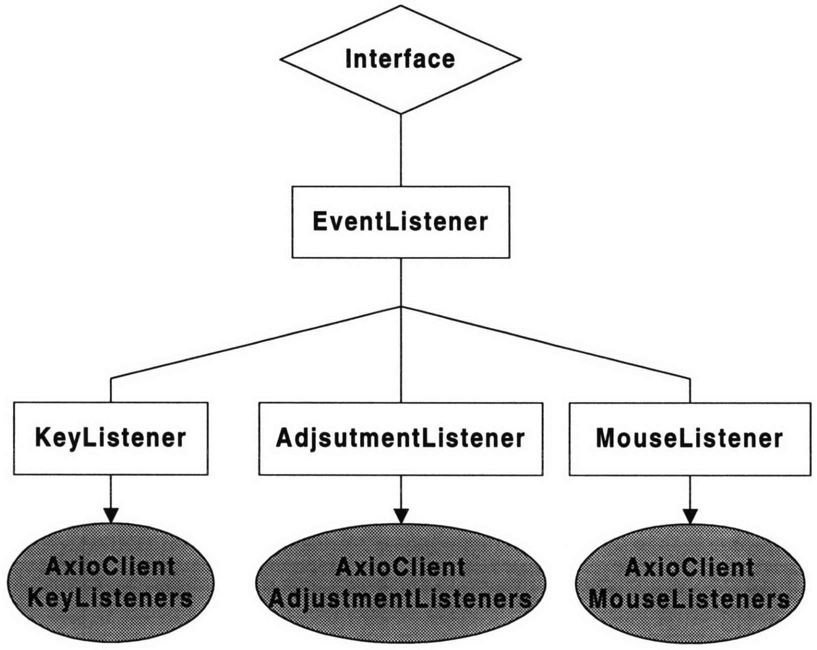

Figure 7: The interfaces needed for the implementation of the client.

4.2.3.4 Component Layout

The layout of the components is accomplished by the AxioClient panels, which extend the JDK's Panel class as shown in Figure 6. The panels implemented for this project are shown in detail in Figure 8. The names of the different panel classes are self explanatory, except that some clarification is needed regarding the XDirController panel. The XDirWithGauge is a class that lays out all components on the GUI that have a status gauge (the condenser, the diaphragm, the lamp) and the XDirWithValue class lays out all components whose current position is described by a discrete value (the objective nosepiece and both filter wheels).

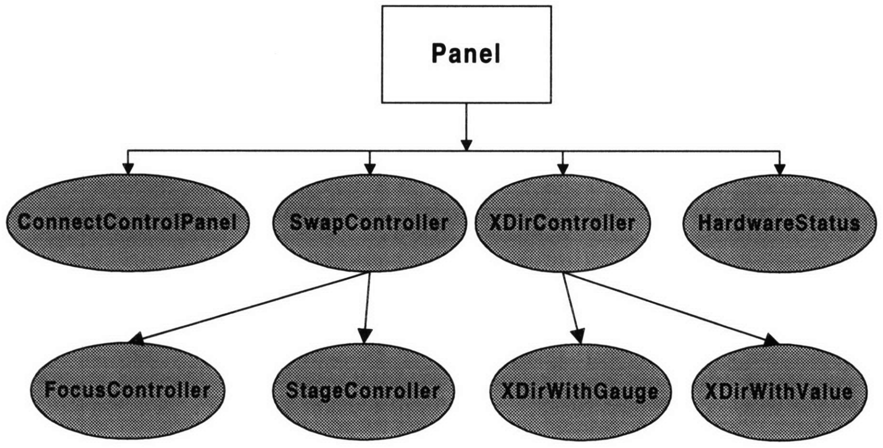

Figure 8: Client's Panel classes.

4.2.3.5 Event Handling

The events on the GUI are handled by the event listeners, shown if Figure 7. In particular, KeyListeners provide a keyboard-based control of the stage and the microscope focus to the user. Adjustment listeners allow for setting of the speed at which the stage can be moved in x, y, and z dimensions. Mouse listeners handle all mouse events on the GUI and are depicted in detail on Figure 9.

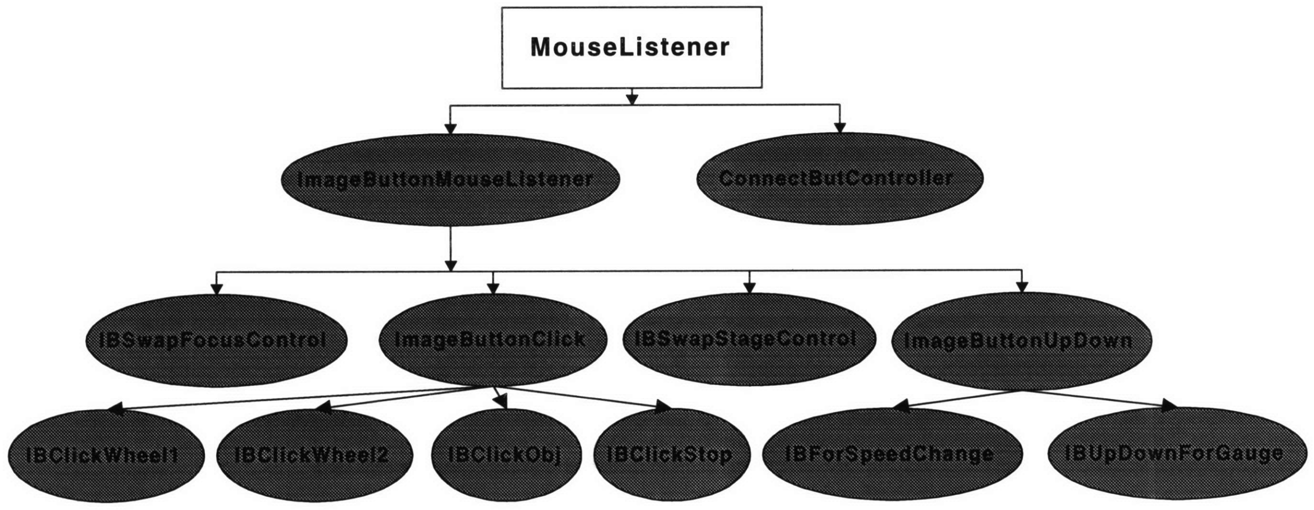

Figure 9: Client's Mouse Listeners.

4.2.3.6 Non-GUI Controls

Finally, Figure 10 shows the classes that do not pertain to the GUI explicitly but are vital to the client's functionality. Two of the classes shown deserve a brief description. The NetworkConnection class is responsible for providing communication channels to the server. It is responsible for establishing and terminating the connection to the server and provides two streams: one to read from the server, the other to write to the server. The AxioCode class specifies the constants which are used in the protocol for client server communication. The meaning of the majority of these constants is revealed in Tables 3, 4, and 5. It is essential that both the client and the server have access to the same implementation of this class.

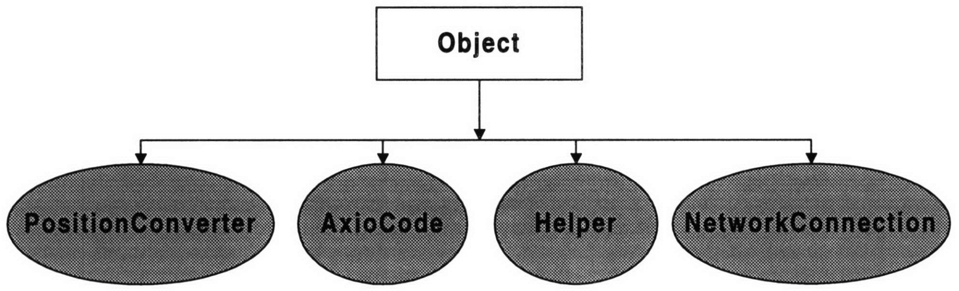

Figure 10: Client's non-GUI classes.

4.2.4 Server-Side Implementation

4.2.4.1 Multilevel Approach

The implementation of the server requires the integration of four pieces of hardware equipment: the computer on which the actual server's software is running, the microscope, the stage, and the stage controller. The server's core functionality is to pass the client's requests that come over the network to the stage and/or the microscope. This task can be broken down into a number of levels, where each level is responsible for a particular subtask. Figure 11 presents a view of the server from a multilevel subtask-oriented perspective.

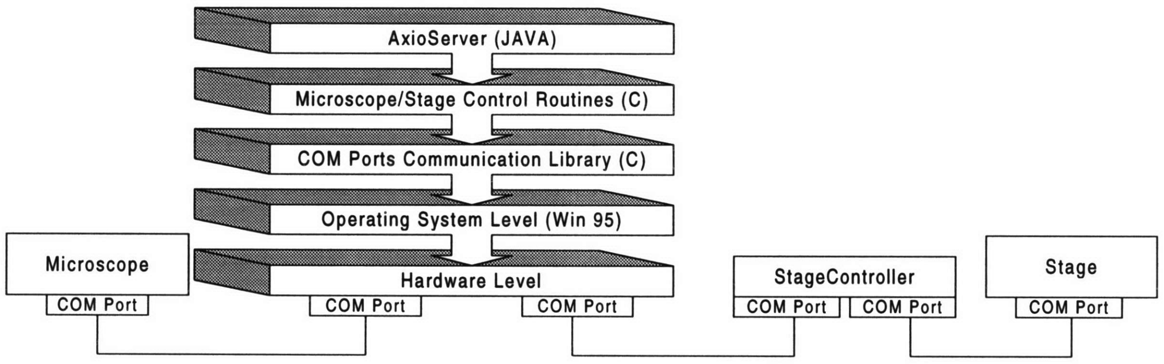

Figure 11: Multilevel design of the server.

4.2.4.2 Hardware Level

As mentioned earlier, Zeiss has provided an RS232-based interface to both the microscope and the stage. Specifically, both the microscope and the stage can be controlled by sending instructions to their respective COM ports. As shown on the figure, the microscope can be controlled directly: there is a direct COM port to COM port connection between the controlling computer and the microscope.

The control of the stage, on the other hand, is slightly more complicated: there is an RS232 connection between the computer and the stage controller, and there are additional connections between the stage controller and the stage itself. The stage controller has support for a number of different stages, including the one that is used for this project. The Zeiss RS232 interface codes allow for the control of the stage controller, which in turn mobilizes the stage.

4.2.4.3 Operating System Level

An RS232-based interface is a very simple and ubiquitous mode of low level communication: all of the established operating systems provide RS232 support. Thus, hypothetically speaking, the server could have been developed on any one of the established operating systems. For this project we chose Microsoft's Windows 95 operating system, primarily because most of the video streaming software and hardware is designed for the PC environment.

4.2.4.4 COM Port Communication

Generally speaking, Windows 95 is a rather stringent operating system as compared to UNIX [24] for example. One of the implications of Windows 95's inflexibility is that a large number of relatively simple tasks require a considerable effort. That is exactly why a COM port communication library had to be developed for this project. The 32-bit library, written in C [25], makes heavy use of Microsoft's sample code and allows the operating system (Windows 95) to read from and write to any of the computer's COM ports. Notice that once we have to resort to using Windows 95-based COM port library we limit the server side to PCs running Microsoft's Windows 95 or Windows NT. This dependency and the ways to overcome it will be discussed in a later section.

4.2.4.5 Control Routines

As seen in Figure 11, the microscope and stage control routines reside on the fourth level of the block hierarchy. These routines are written in C and compiled into a 32-bit dynamically linked library (DLL). The library uses the RS232 codes provided by Zeiss to control the stage and the microscope. This library relies on the COM port communication library, discussed in the previous subsection, to actually send RS232 codes to the microscope and stage and read the results back.

4.2.4.6 Main Application

As mentioned earlier, the actual application that waits for the connection from the client and communicates with the client once that connection is established is written in JAVA. In the figure, that functionality is accomplished at the highest level of sub-task hierarchy.

Notice that the main server application has to invoke the microscope and stage C control subroutines in order to actually change the state of the microscope and/or the stage. Fortunately, JAVA gives the capability to invoke native C methods from a JAVA application [26], which is exactly what was done in this project.

There are two main considerations associated with calling native C methods from JAVA: speed and portability. There is a minor speed penalty associated with the overhead, associated with the actual invocation of a native C method. On the other hand, as of today, JAVA is considerably slower than C. That means that once the native C method for microscope control routines is invoked, the application gains a speed benefit, since compiled C code can be executed much faster than JAVA code. Finally and most importantly, both of the above effects are relatively insignificant compared to the overhead of communicating to the hardware of the microscope, which is independent of whether native C methods are called or not. In particular, most of the high-level functions of the microscope require more than one RS232 code, and the wait time (to ensure proper execution) between sending two consecutive codes to the microscope is about 100ms. In summary, the reliance on native C methods for the microscope control will not have a significant impact on the overall speed of the application. The portability-related implications of utilizing native C methods within a JAVA application will be discussed in a later section.

4.3 Video Control

The previous section discussed issues involved in controlling the microscope and the stage. Such control will serve no purpose unless accompanied by a video transmission system, that streams video in real-time from the server to the client.

We had the opportunity to closely evaluate two video-streaming systems for this project. The first system, manufactured by Xing Technology Inc. [27], is based on the MPEG-1 video encoding/decoding standard. Appendix A.4 describes the system and presents factors that prevented us from using it for the final implementation of the project. The second system, manufactured by Videoconferencing Systems Inc. (VSI) [28], is based on the H.261 videoconferencing standard and is the one chosen for this project.

Due to the delays in the delivery of the VSI's system, it was not installed at the time of writing of this document. However, VSI promised that the system is fully-operational and is well-fit for the remote-microscopy project. The core of the VSI's H.261-based system consists of an encoding board, decoding board, and software allowing to control video streaming. The encoding board, will be installed on the server side, and will take its NTSC input from the FUJIX camera attached to the microscope. That NTSC signal will be digitized and streamed across the network to the decoding board, to be installed on the client side (see Figure 2). The client will decode the received video stream in real-time and display it at 352x288 resolution.

The actual implementation of video streaming and the rationale for choosing the hardware scheme described in the previous paragraph are presented in the following Chapter.

Chapter 5 Discussion

In this section we will show how the remote microscopy system meets the expectations presented in the Background section. It must be emphasized that the discussion will focus on the evaluation of the two parts of the project that were implemented: the microscope and stage control and low-resolution real-time video-streaming control. The next Chapter discusses how the existing system can be extended to accommodate high-resolution still image acquisition.

5.1 Video Streaming System Characteristics

As mentioned earlier, there are two ways to get image data from the server to the client. One way is to acquire a high-resolution still image through a SCSI interface of the Fujix digital camera, and then deliver that image to the client. The other way is to utilize one of the existing video-transmission systems to deliver a low-resolution real-time video stream from the server to the client.

It is important to distinguish between the objectives of the two modes of image data delivery. The primary purpose of the low-resolution real-time video-transmission is to facilitate comfortable focusing of the microscope located at a remote location. The actual image that is to be closely examined by a researcher or a doctor will be delivered through the high-resolution still image acquisition module shown in Figure 2.

In this section we discuss the desirable characteristics of the video-streaming system. For this project a considerable effort was spent on finding a system suitable for our needs. The World Wide Web (WWW) resources proved to be indispensable for this task [29]. The three most important system parameters that we considered were: the latency, the resolution, and the frame rate. In what follows, we discuss the acceptable values for these parameters, and the rationale behind the chosen values.

5.1.1 Resolution

Video Streaming Standard	Maximum Resolution	Frame Rate (frames/second)
H.261	352x288	~20fps
MJPEG	640x480	~24fps
MPEG-1	352x240	~30fps
MPEG-2	704x576	~30fps

Table 6: The characteristics of prominent video streaming standards in a 10Mbps ethernet environment with an average load.

Existing video streaming systems differ greatly in their maximum resolution capabilities. Table 6 presents the maximum resolutions of the most prominent video streaming standards. When examining the table, it should be taken into account that for some of the listed standards the theoretical maximum resolution is higher than the one listed in the table. The table presents the maximum resolutions as obtained by today's practical implementations of these standards.

As seen from the table the resolutions vary greatly among the standards. However, it is important to keep in mind that no matter what algorithm is used for video compression, the vast majority of today's video transmission systems take an NTSC signal as an input. This means that, the useful resolution of these systems is limited by the resolution of their NTSC input: 352x240.

5.1.2 Frame Rate

Another characteristic that substantially influences the evaluation of any video streaming system is the frame rate. Table 6 specifies the frame rates of the most prominent video streaming standards. The listed frame rate for each standard can be accomplished with a good implementation of that standard, given an average load of a 10Mbps ethernet environment.

The telemedicine application implemented in this thesis is not very stringent regarding its frame rate requirements. The nature of the application - remote control of a microscope - does not entail any significant motion in the associated video stream. Any frame rate that allows a user to observe the change in view of a specimen once some function of the microscope was executed is acceptable. Realistically, any frame rate above 10fps will be sufficient for this application.

5.1.3 Latency

The main purpose of this prototype system is to provide a real-time remote control of an optical microscope. A certain care should be exercised when the term "real-time control" is being used, since some latency is inherited in any video transmission system. For all practical purposes the latency of less than .1 second satisfies real-time requirements since it enables a user to control the microscope in a very comfortable natural manner.

Admittedly, the vast majority of video-streaming systems today do not quite achieve the ideal .1 second latency. Some video streaming standards incur more latency than others, but a good implementation of almost any standard can achieve latencies of less than 1 second. The very best implementations can provide for latencies comparable to .1 second and satisfy the real-time requirements.

5.1.4 The Video System for the Project

For this project one of the existing H.261-based video transmission systems will be utilized. We chose an H.261-compliant system for a number of reasons. As compared to the MPEG standard, H.261 incurs a shorter latency, which is crucial for the project. H.261 theoretical frame rate (~20fps) is less than that of MPEG, but it is more than sufficient for the purposes of the project.

H.261 is also a better solution than MJPEG. First, the quality of H.261 video stream is higher than the quality of MJPEG video stream for the same bit rate. Second, although the MJPEG standard is a viable alternative today, it might not be one tomorrow, since the industry is gravitating towards MPEG for write-once read-many applications and towards H.261 for videoconferencing applications.

As mentioned earlier, the only requirement for the resolution of the video streaming system is that it is higher than the NTSC resolution of 352x240. The resolution of H.261 is 352x288 which satisfies this requirement.

The video streaming system chosen for this project is VSI's Omega MVP. Unfortunately, at the time of writing of this document the system was still being delivered; however, VSI assured that their system has the following characteristics. Omega MVP supports 352x288 resolution at 15 frames/second. The latency of this system is less than .3 seconds which makes it a good alternative to the ideal real-time latency. Overall, this system will allow for a reasonably comfortable remote control of the microscope at an acceptable resolution.

5.2 User Interface

As mentioned earlier, the target group of users for this application is expected to have limited computer experience. Taking that into account, we have developed a self-explanatory, easily-navigatable user interface depicted in Figure 5. This interface clearly shows which buttons control which microscope functions. The interface also provides buttons to establish and terminate connection to the server as well as the buttons displaying information about the program (the "About" button), and information about the microscope (the microscope image button).

In addition, this user interface provides dynamic status checking. The bottom part of the GUI shows the current status of the major components on the server side: the actual server, the stage, and the microscope. The availability of each component is indicated by a green light. Moreover, the buttons allowing the user to control the microscope can be pressed only if the connection to the server was established and the microscope is available. Analogously, the buttons controlling the stage can only be pressed if the stage is available, and the server is connected. Finally, if during the execution of the program the server dies, the error message will appear and the client will gracefully terminate the connection and update the GUI accordingly.

5.3 Platform Independence

Platform independence is a desirable feature for any system, and it is especially important for systems based on the client-server paradigm. The telemedicine project described in this thesis integrates a number of technologies from a number of vendors, and attempting to maintain platform independence was a challenging task. We have partially succeeded in this endeavor, as discussed in the following two sub-sections which evaluate the design of the server and the client from the platform independence perspective.

5.3.1 Server

The main engine of the server is written in JAVA which is a high-level platform independent programming language. However, to actually control the microscope and the stage, the server is forced to resort to invoking native C methods. It is clear that once we have a JAVA application invoke a native C method, JAVA's main advantage over other languages - platform independence - is no longer present. There are two ways of looking at this issue.

First, given that the project requires low-level RS232 communication to control the microscope and the stage, some platform dependence is imminent, since each operating system provides different means for controlling RS232 interface. Thus, we didn't introduce more dependence than was inherently present in the system.

Second, and more important, this platform dependence exists only on the server side, which is only encountered once when the server is installed and configured: integrating the microscope, the stage and the video streaming technologies. Indeed, the server side of the application exhibits significant hardware dependency due to the platform dependence of both the video streaming technology, and the necessary RS232-based control of the stage and the microscope. However, what is considerably more important for this project is that the client is platform independent.

5.3.2 Client

The client is written purely in JAVA (JDK 1.1), and its implementation is independent of how the server controls its hardware. Notice, that one of the many advantages of JAVA is that networking can be done in a very straightforward manner. For example, to establish a TCP/IP-based connection to the server, the programmer does not need to know what platform the client is running on as long as that platform provides TCP/IP support.

In general, once TCP/IP is chosen for communication JAVA allows to write networking code without taking into account what platform the application will be ported to. What limits client's portability is the real-time video decoding and display that has to be performed on the client side, when the video stream from the server is received.

However, the essence of any video transmission system is its encoding technology, which must be done in hardware to allow for real-time streaming. Decoding a video stream is a much simpler task and can be implemented purely in software, while still conforming to the real-time requirements of the system. It is due to that encoding/decoding asymmetry that many of the existing transmission systems place more platform dependent restrictions on their encoding end, while decoding is relatively platform independent.

In this telemedicine application, the encoding is done at the server side, while decoding is performed at the client. The current implementation of the video transmission system utilizes hardware for both encoding and decoding, which unfortunately restricts the client. On the other hand, this video streaming technology is based on the H.261 standard, which is becoming the default standard for videoconferencing. We foresee that increasingly better implementations of H.261 will be emerging in the near future. It is likely that the next version of the video transmission system used for this project, will perform decoding in software. Alternatively, such a software decoder may be provided by a different vendor in the future. Thus, at least in principle, the client is more flexible compared to the server.

The above illustrates that, although the client exhibits some platform dependency, migrating between client platforms can be done in a relatively straightforward manner. In particular, such migration would encompass a different version of the video stream decoding package, which is expected to be provided by the company that manufactured the video-transmission system. Most importantly, no new JAVA code would have to be written for the client.

5.4 Modularity

In developing the system, a significant effort was exerted to keep separate modules as independent as possible. Such strategy allows for easier extension and modification of the system as well as easier platform migration if it is deemed necessary. The modularity of the project can be traced on three different levels. First, the existing system can be seen as consisting of two functionally very different parts: low-resolution video control and microscope control. Second, the remote nature of the application led us to employ the client-server architecture design which is by nature very modular. Third, the server itself integrates a number of technologies that span different levels of the machine on which the server is running. In the following subsections we demonstrate how the modularity of each level was accomplished.

5.4.1 Level I: Video and Microscope

As mentioned earlier, the current implementation of the project encompasses the setup of two distinct parts: control of the microscope and stage, and the control the video streaming system. These two parts are independent: changes in the video-streaming system will have absolutely no effect on the control of the microscope and the stage. Similarly, the modifications of the microscope and the stage controls will not effect the video-streaming system. The reason for such independence resides in a fact that the project utilizes one of the commercial plug-and-play video systems, and as such these systems can not be modified even if there was a need to do so.

5.4.2 Level II: The Client and the Server

The client and the server are independent of each other, in a sense that their only concern is that the opposing node conforms to the established client-server protocol. As long as the other end conforms to the protocol, its implementation is irrelevant. In other words the client treats the server as a "black box" and vice versa.

5.4.3 Level III: Modular Design of the Server

Finally, the server consists of a number of modules or building blocks discussed in a great detail in Section 3.2.1.4. These building blocks operate on different levels ranging from the hardware level to the main application level as shown in Figure 11. Notice that the building blocks are fairly independent of each other. Each level utilizes the exported interface of the level below and does not depend on the implementation of the lower level. Likewise, each block provides an implementation-independent interface to the block above it. This design eases the task of modifying, extending, or adapting the system.

For example, if a migration to a different operating system were required, the COM port communication library would have to be revised, but the interface would stay the same. The exception to this scenario would be a migration to a Windows NT operating system; in this case only recompilation of the library would be called for. Any migration would also result in the recompilation of the microscope and stage control subroutines, but their implementation would stay the same. In summary, hierarchial design of the server allows for a relatively easy migration to different operating systems. The only part of the server that would require reimplementation is the COM port communication library, which is trivial to design and deliver for any operating system.

Chapter 6 Future Directions

6.1 Improving Resolution

A major area that needs improvement is resolution. Currently, the system allows for a real-time streaming of 352x240 resolution images. Although such resolution might be appropriate for videoconferencing applications in a business environment, it is not suitable for most medical or scientific applications, where a high image quality is a major consideration. We propose two ways of improving the current situation: by remotely acquiring high-resolution images through the video camera's SCSI output; and by migrating to a higher-bandwidth network such as ATM.

6.1.1 Remote Acquisition of High-Resolution Images

The remote acquisition of high-resolution still images was supposed to have been an essential part of the original project (see Figure 2), however, for a number of reasons this part was never implemented. This section shows how the acquisition and delivery of high-resolution still images can be accomplished and discusses what factors prevented us from completing this part of the project.

6.1.1.1 Proposed Solution

Most of the researches and doctors who use microscopes spend a significant percentage of their time examining stable images as opposed to manipulating them. Thus, once a researcher or doctor manipulates the settings of a microscope to provide an image in which he is interested, he spends the next 5-10 minutes closely examining the image, without changing any of the parameters.

Given the above observation, the system's design can be altered in the following manner. The Fujix digital camera attached to the microscope contains two outputs: NTSC (352x240 resolution) and SCSI (1280x1000 resolution). The real-time video signal on the client side is 352x288. Once a user of the remotely-controlled microscope stops manipulating the image or explicitly requests a higher resolution image, the server can be forced into terminating the real-time video transmission and will instead send a single 1280x1000 image to be displayed on the client's screen.

Once the client resumes the manipulation of the microscope, the 1280x1000 image would be discarded, and the server will restart H.261 based video transmission. This scenario provides for low resolution video signal while manipulating the image, but delivers a high resolution image once no more adjustments of the microscope are required or upon the user's request.

6.1.1.2 Implementation Issues

Delivering a single high resolution image to the client requires two steps: acquiring the image from a camera and transmitting the resulting image data across the network. The process of transmitting data over the network is ubiquitous to many applications today and is fairly easy to implement. The process of acquiring an image from a high-resolution camera is managable from a technical perspectives, but involves certain administrative connotations to be discussed below.

Acquiring a high-resolution image on the server side should not be particularly complicated. In fact, most of the sophisticated modern cameras provide software, capable of acquiring images locally (usually over a SCSI port). However, most commonly, the software provided comes in a form of a standalone application as opposed to an Application Programming Interface (API).

The implementation of the remote acquisition of high resolution images, on the other hand, would require a set of libraries (API) that provide procedures to acquire an image from a camera. Unfortunately, the vast majority of camera providers are adamant about not giving out libraries for their products. The remote acquisition of high resolution images can be implemented in a relatively straightforward manner once such API is released.

6.1.1.3 Flexibility

The described scenario fits in very well with the modular design of the original system, since the remote image acquisition piece is relatively independent of the other parts of the project (video streaming and microscope control). We foresee that the new system would be quite flexible in handling the future improvements in technology, which are imminent. Specifically, as telemedicine and imaging industries mature, the market will see drastic improvements in the resolution of the digital cameras on the market. Admittedly, once better products become available at reasonable prices, a decision to upgrade to a better product is likely to occur.

This expected upgrade, however, will not have a significant effect on the implementation of the project. The control software for the microscope and the stage will remain the same. The video streaming system, which requires only an NTSC output of the video camera, will remain unaffected by the camera upgrade. Finally, the software module allowing for transmitting an acquired high resolution image across the network will also stay unchanged, since the network transfer routines for this project will be indifferent to the data being transmitted.

The only software that will require modification is the image acquisition module. Ideally, the image acquisition module (a set of libraries) will be provided by the company that manufactures the camera. Thus, the task of upgrading the system to a better camera will be reduced to recompiling a slightly modified code for the image acquisition module and relinking the server-side of the project.

6.1.2 Migration to a higher bandwidth network

The bandwidth of underlying network environment has a significant effect on the remote microscopy project. First, the frame rate of the video-transmission system relies on the available network bandwidth. In particular, the more bandwidth that is available, the higher the frame rate that can be achieved, given constant resolution. Second, the transfer speed of a single remotely-acquired high-resolution image would increase if more network bandwidth were available. This speed increase will make the control of the microscope considerably more natural since the user, will experience very little delay between requesting a high-resolution image and the arrival of that image.

Resolution	Color	Frame Rate
1600 x 1200	24 bpp	3 fps
1280 x 960	24 bpp	4 fps
1600 x 1200	8 bpp	9 fps
1280 x 960	8 bpp	14 fps

Table 7: The performance in the 155Mbps ATM environment, taking into account the ATM overhead; no compression.

The current version of the project is implemented in a 10Mbps switched ethernet environment, using VSI's H.261-based video transmission system. The remote image diagnosis prototype described in this proposal would benefit if we implemented this prototype on a 155Mbps ATM instead of a 10Mbps switched ethernet. Table 7 illustrates the kind of performance that can be achieved given the ATM bandwidth. Notice, that the table shows the frame rates, at which the uncompressed images can be transmitted. Naturally, these frame rates would improve significantly if some compression scheme were utilized. For example, if the compression algorithm allowed for a 5:1 compression ratio, the corresponding frame rate would increase by a factor of 5.

Migration to an ATM network can be accomplished in a relatively seamless manner. In particular, most of today's ATM implementations include support for Local Area Network (LAN) Emulation [30]. As its name implies LAN Emulation software allows legacy protocols (such as TCP/IP) to run on top of connection-oriented ATM. The remotely-controlled microscope discussed in this thesis is a TCP/IP based application and, ideally, can run on any ATM network supporting LAN Emulation. Realistically, the manufacturer (VSI) of the video-streaming system used for the project, will have to alter the system's hardware to make it ATM-compliant. Moreover, the data transfer module of the remote high-resolution still image acquisition part of the project will have to be modified in order to take full advantage of ATM's bandwidth.

When examining Table 7 it should also be taken into account that the theoretical ATM bandwidth of 155Mbps is not attainable in practice. A number of research projects were dedicated to estimating the practical bandwidth that can be delivered by an ATM network [31, 32, 33]. The results were not particularly encouraging: the percentage of the theoretical ATM bandwidth that can be obtained ranges between 30% and 80%. Importantly, the lower rates are associated with running applications written for legacy protocols (e.g. TCP/IP) on top of ATM. Unfortunately, the majority of ATM users today run exactly these kinds of applications (i.e. TCP/IP based) and are experiencing only a fraction (e.g. 1/3) of the theoretical ATM bandwidth.

6.2 Conclusion

Our experience with this project shows that the remote-microscopy system designed and implemented for this project can prove to be useful in the near future. From a technical perspective this work demonstrates how different cutting-edge technologies can be combined to produce a robust telemedicine system. The main idea behind the system is that the control of a remotely-located medical apparatus accompanied by real-time video support can be extended to general microscopy, X-ray analysis, and mammograpy. This thesis shows that today's technology is capable of accommodating the requirements of real-world advanced telemedicine applications and we can expect to see an explosive growth of telemedicine in the near future.

Appendix A.1

Axioplan 2: Microscope control

Control	*Description*
On/Off switch	For optical status checking, the switch lights up in green when instrument is switched on.
Light intensity control	Knob to adjust the light intensity of connected 100 HAL microscope illuminators.
On/Off display of color temperature key	Lights up when the color temperature of 3200 K has been switched on.
Color temperature key	Knob to set the 3200 K color temperature for photomicrography using color films (artificial light).
Filter wheels for transmitted light	Two rotatable filter wheels with 4 positions each are equipped with different filters.
Luminous-field diaphragm	Wheel for the continuous setting of the aperture of the luminous-field diaphragm (transmitted light).
Light exit of transmitted light equipment	Light filters with dia. 32 mm can be placed on the plane surface of the centering ring supplied.
Manual focusing drive	The universal microscope Axioplan 2 is focused via coaxial drives on both sides of the stand.
Stage carrier	The stage carrier is used to mount the stage and the condenser carrier to which the condenser is attached.
Objective nosepiece	The objective nosepiece is used for mounting the objectives and changing them quickly.
Reflector turret	The reflector turret consists of a filter wheel with 5 click stops to which the required reflector modules are attached.
Rapid stage lowering CHANGE/VIEW	Knobs on both sides of the stand for rapid stage lowering and moving it up again to the previous position

Appendix A.2

Axioplan 2 Objectives

Axioplan 2 came with Plan-Neofluar objectives for transmitted-light and fluorescence. These objectives were installed through the objective nosepiece of Axioplan 2. These universal objectives feature equally high quality in transmitted-light and epifluorescence. Their specialty: the high transmission for fluorescence excitations in the near UV range. Due to their large, completely flat fields of view they provide excellent results in photomicrography. The high numerical aperture of all models permits very high light yield in fluorescence.

Brightfield model: magnification/ numerical aperture	Working distance in mm	Model for phase contrast available:	Strain-free model for polarization available
1.25x/0.035	3.5	-	-
2.5x/0.075	9.3	-	yes
5x/0.15	13.6	Ph1	yes
10x/0.30	5.6	Ph1	yes
20x/0.50	1.3	Ph2	yes
40x/0.75	0.33	Ph2	yes

Appendix A.3

Axioplan 2: Technical Data

1. Ambient conditions

Condition	*Description*
Room temperature	-10...+35 ^{0}C
Humidity	max. 75% at +35 ^{0}C
Storage temperature	0...+35 ^{0}C, humidity 10...30%
Weight	configuration dependent

2. Safety

Feature	*Description*
Equipment class	I
Protective degree	IP 20
Radio disturbance characteristics	conforming to EN 55011 (Class B)
Electromagnetic immunity	conforming to EN 50082-2

3. Electrical supply

Power characteristic	*Description*
Power connection	220...240 V, 50/60Hz +/- 10% convertible to 100...120 V, 50/60Hz +/- 10%
Power consumption	approx. 225VA

4. Data for connection of 12V/100W microscope lamp

Feature	*Description*
Lamp voltage	12 V
Power	100 W
Color temperature at 11.5 V	3200 K
Luminous flux	3100 Im
Mean service life	50 h
Luminous surface	3.1 x 3.1 mm^2

Appendix A.4

Xing Technology's MPEG-1-based Video-Streaming System

As mentioned in Chapter 4, Xing's MPEG-1-based system was originally considered for the video-streaming module of the project. The system was fully installed, tested, and evaluated. Figure 12 shows the configuration of the remote-microscopy project during the testing period.

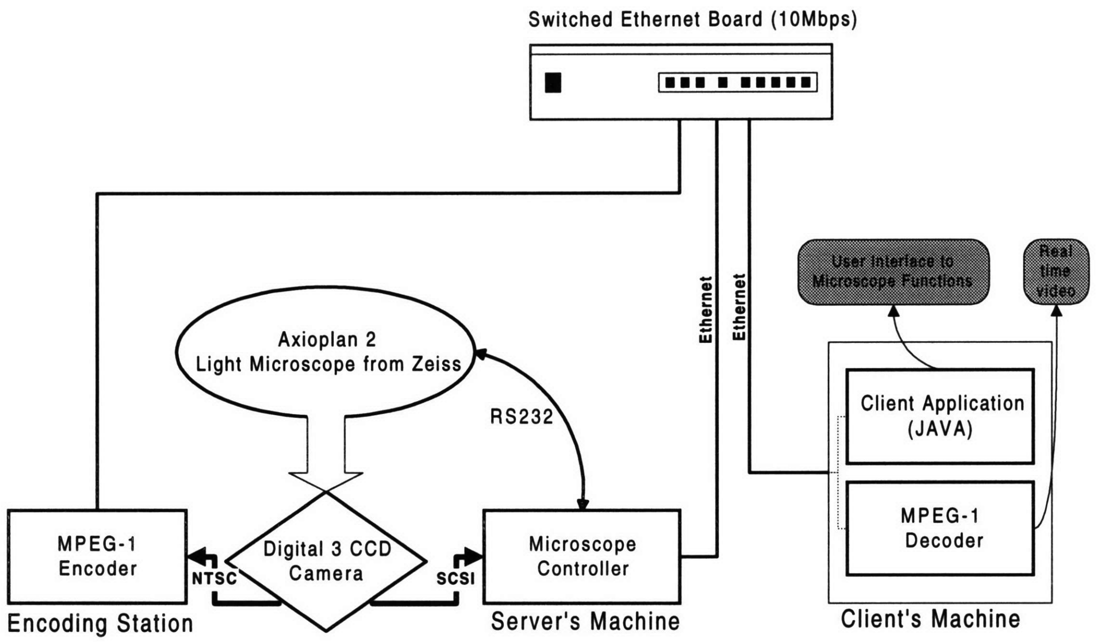

Figure 12: Testing configuration for Xing's video-transmission system.

The video-streaming system consists of a stand-alone MPEG-1 encoding station and decoding software. The encoding station needs a separate ethernet connection and operates independently from the microscope controller. As illustrated in the figure, this station takes NTSC input produced by the Fujix digital camera, encodes its analog signal into digital format, and streams the resulting video stream to the client. On the client side, the video signal is decoded by Xing's software and displayed on client's computer screen.

It is important to realize that in this project the primary aim of the low-resolution video-transmission system is to provide a natural control of the microscope from a remote locaciton. Thus, as the user controls the microscope, she should be able to see the changes in the image in real-time.

Having taken this consideration into account, consider the following table presenting the performance characteristics of Xing's video-streaming technology. The values of the resolution, frame-rate, and required bandwidth parameters are fully acceptable for this part of the project. However, this system incurs a 3 to 4 second latency. In particular, a latency of ~3 seconds is associated with movement of the stage along the Z-axis (i.e. focusing), and a latency of ~4 seconds is associated with movement of the stage in the X-Y plane.

Parameter	Value
Resolution	352x240
Frame-Rate	~30fps
Required Network Bandwidth	~1.5 Mbps
Latency	3-4 seconds

Unfortunately, the remote-control of the microscope becomes an extremely unnatural task, when there is a 3 to 4 second delay between the time at which the state of the microscope is changed and the time that the change is reflected on the user's computer screen.

The reasons for such latency were discussed on several occasions with Xing's technical personnel. In brief, the video-buffering algorithm used in Xing's implementation of the MPEG-1 standard requires ~2 second delay on the encoder's end and a similar delay on the decoder's end resulting in the overall 3-4 second latency. Although Xing promised to look into this latency issue, it was our understanding that changes to Xing's implementation of the MPEG standard were not likely to occur in the near future.

In summary, although the remote-microscopy system can be integrated with Xing's MPEG-1 technology, the usability of the resulting overall system is limited due to the unbearable latency of the video-stream. This is the main reason why we decided against the use of Xing's system in this project, until the latency of Xing's transmission system is drastically reduced.

References

[1] Telemedicine Research Center. *http://tie.telemed.org/TIEmap.html.* 1997.

[2] D.Kim, J.E. Cabral Jr., Y. Kim, "Networking Requirements and the Role of Multimedia Systems in Telemedicine", SPIE Proceedings, Medical Imaging, Vol. 2436, 1995.

[3] B.K.T. Ho, R.K. Taira, R.J. Steckel, H. Kangarloo, "Technical Considerations in Planning a Distributed Teleradiology System", Telemedicine Journal, Vol. 1, No.1, 1995.

[4] R.L. Bashshur, "On the Definition and Evaluation of Telemedicine", Telemedicine Journal, Vol. 1, No. 1, 1995.

[5] M. Gardy, "Telemedicine and Economic Realities", Telemedicine Journal, Vol. 2, No. 2, 1996.

[6] J.H. Mayer, "ATM Networks Transfer Medical Images at High Speeds", Diagnostic Imaging, Nov. 1996.

[7] G.A. DeAngelis, B. Dempsey, S. Berr, L. Fajardo, J. Sublett, B. Hillman, A. Weaver, K.Berbaum, S.J. Dwyer III, "Digitized Real Time Ultrasound: Signal Compression Experiment", Oct., 1996, Presented at Radiological Society of North America (RSNA) Conference 1996.

[8] U. Engelman, A. Schroter, U. Baur, A. Schroeder, O. Werner, K. Wolsiffer, H.-J. Baur, B. Goransson, E. Boralv, H.-P. Meinzer, "Teleradiology System MEDICUS", Lemke (Ed). CAR '96: Computer Assisted Radiology, 10th International Symposium and Exhibition, Paris, 26-29 June 1996. Paris. Amsterdam: Elsevier (1996) 537-542.

[9] D.R. Dakins, "Rural network extends reach of teleradiology", Diagnostic Imaging, Nov. 1995.

[10] Y. Kim, J.E. Cabral Jr., D.M. Parsons, G.L. Lipski, R.H. Kirchdoerfer, A. Sado, G.N. Bender, F. Goeringer, "Seahawk: A Telemedicine Project in the Pacific Northwest, SPIE Proceedings, Medical Imaging, Vol. 2435, 1995.

[11] Lucent Technologies. CCD History. *http://www.lucent.com/ideas2/discoveries/telescope/docs/ccd1.html* 1996.

[12] U. Black, "ATM: Foundation for Broadband Networks", Prentice Hall PTR, NJ, 1995.

[13] J.D. Murray, W.VanRyper, “Encyclopedia of Graphics File Formats”, O’Reilly & Associates, CA, 1994.

[14] A. Hung, PVRG - JPEG Codec 1.1
http://icib.igd.fhg.de/icib/it/iso/is_10918-1/pvrg-descript/doc.html 1993.

[15] Z. Jin, Development of a Transcoder from MPEG-1 to H.261 Bitstreams.
http://icsl.ee.washington.edu/~zjin/thesis/index.html

[16] Lantronix. Fast Ethernet Tutorial.
http://www.lantronix.com/htmfiles/whitepapers/lfrwp.htm 1996.

[17] Zeiss, Inc., “Axioplan 2: The Profile”, Zeiss, Germany, 1996.

[18] Zeiss, Inc., ”Axioplan 2 and Axiophot 2. Universal Microscopes. Operating Manual”, Zeiss, Germany, 1996.

[19] Zeiss, Inc., ”Stand F: Programming Instructions. Version 3.0”, Zeiss, Germany, 1993.

[20] Zeiss, Inc., ”MCU 26: X, Y, Z Axes Motor Control. Operating manual”, Zeiss, Germany, 1993

[21] Fujix, Inc., ”High Resolution Digital Camera FUJIX HC-2000. Owner’s Manual”, Fujix, Japan, 1995.

[22] Sun Microsystems, Inc. JavaSoft Home Page. *http://java.sun.com/.* 1997.

[23] D.M. Geary, A.L. McClellan, “Graphic Java. Mastering the AWT”, The SunSoft Press, CA, 1997.

[24] M.J. Bach, “The Design of the Unix Operating System”, Prentice Hall, NJ, 1986.

[25] A. Kelley, I. Pohl, “A Book on C”, The Benjamin/Cummings Publishing, Inc., CA, 1995.

[26] M. Campione, K. Walrath. The Java Tutorial Home Page.
http://ecehub.cuep.umkc.edu/docs/progGuide/index.html 1997.

[27] Xing Technology Corporation homepage. *http://streams.xingtech.com*

[28] Videoconferencing Systems Inc., homepage. *http://www.vsin.com*

[29] K.Hewitt, DT-5: Desktop Videoconferencing Product Survey.
http://www3.ncsu.edu/dox/video/survey.html. 1996

[30] 3COM Networking Solutions Center homepage. ATM LAN Emulation. *http://www.3com.com/nsc/500617.html* 1997.

[31] Y. Fouquet, R. Schneeman, D. Cypher, A. Mink, ATM Performance Measurement: Throughput, Bottlenecks and Technology Barriers. *http://isdn.ncsl.nist.gov/misc/hsnt/journals/atmperf/atmperf.html* 1995.

[32] Electronics and Computing Technologies Division homepage. ATM Network Interface Card Evaluation. *http://www.anl.gov/ECT/network/racarlson/evaluation.html* 1997.

[33] Mengjou, L., Hsiehn J., Du, H., C., Thomas, J., P., MacDonald., J., A., Distributed Network Computing over Local ATM Networks, Computer Science Department University of Minnesota, Computer Science Department Technical Report, TR-94-17.