Transforming Medical Imaging Applications into Collaborative PACS-based Telemedical Systems

by

Rouzbeh Maani

A Thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements of the degree of

Master of Science

Department of Computer Science
University of Manitoba
Winnipeg

Thesis advisor

Neil Arnason, Sergio Camorlinga

Author

Rouzbeh Maani

Transforming Medical Imaging Applications into Collaborative PACS-based Telemedical Systems

Abstract

One of the medical technologies distinguished for its clinical value is telemedicine. Telemedical systems provide healthcare services at any time and any where irrespective of geographical location. These systems are even more interesting when they provide collaboration among the users (e.g., teleconferencing). Nonetheless, all these systems are not practical for use in clinical workflow unless they are able to communicate with the Picture Archiving and Communications System (PACS). On the other hand, there are many medical imaging applications that are not developed as telemedical systems. A large number of them do not support collaboration and many do not communicate with the PACS and therefore cannot be directly used in clinical workflows.

This thesis presents an approach based on a three-tier architecture. The architecture and the developed components within it transform medical imaging applications into collaborative, PACS-based, telemedical systems. As a result, current developed medical imaging applications that are not telemedical, do not support collaboration, and do not communicate with PACS, can be enhanced to support collaboration among a group of physicians, be accessed remotely, and be clinically useful.

This thesis also proposes a novel idea to support PACS connectivity. The idea is to use the Digital Imaging and COmmunication in Medicine (DICOM) protocol and enhance transmission time by employing a pair of interfaces and a combination of parallelism and compression methods. This idea speeds up the transmission time of medical images especially over Wide Area Networks (WANs). Experimental results show up to 1.63 speedup over LAN and up to 16.34 speedup over WAN compared to the current method of medical data transmission.

The main advantage of the proposed architecture is that it does not impose any modification to the current medical imaging applications and does not make any assumptions about the underlying architecture or operating system.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to begin by thanking my advisors, my committee, my family, my significant other, and all the people who have supported me along the way.

I should also acknowledge and thank TRLabs, the Government of Canada (Western Economic Diversification Canada), NSERC (Natural Sciences and Engineering Research Council of Canada), the Government of Manitoba (Innovation, Energy and Mines Department), and BCC (Biomedical Commercialization Canada) for providing funding to this research project and the Department of Computer Science, the Faculty of Science, and the Faculty of Graduate Studies at the University of Manitoba for providing travel funding.

Finally, I would like to acknowledge and thank the collaborators and friends who have collaborated with me from the University of Manitoba, TRLabs, Biomedical Commercialization Canada, Industrial Technology Center (ITC)/Virtual Reality Center, Merlin and Health Sciences Center.

This thesis is dedicated to my family and all who dedicate their lives to open up new horizons for human being.

Chapter 1

Introduction

Medical imaging informatics is a subset of medical informatics dealing with all topics related to medical images. These topics include image acquisition, storage, processing, transmission, management, security, distribution, visualization and analysis [1].

Recent developments and research activities in this field have promoted medical imaging informatics to a popular discipline [2]. These developments range from development efforts in acquisition equipments to those that are motivated in the fields of pattern recognition, image processing, and computer vision.

Historically, four periods in medical imaging informatics are distinguished [3]:

1. Pre-1980 to 1984: 2D image analysis era.

2. 1985-1991: Period of emerging knowledge-based strategies and Magnetic Resonance Imaging (MRI).

3. 1992-1998: 3D image analysis era and utilization of mathematical-model-driven

approaches.

4. 1999 and beyond: Period characterized by using image-guided procedures, more realistic visualizations and advanced imaging and computing technology.

Many researchers in the past decade have tried to facilitate the diagnosis process and many new tools and features have been developed in this regard. Some of these advances have dramatically affected medical imaging systems. For instance, Picture Archiving and Communications System (PACS) [4] has improved the clinical workflows remarkably. Collaboration among groups of physicians has been noted widely and collaborative systems [5] have been developed to facilitate the communication between physicians. Finally, telemedicine[6] has become one of the essential parts of many medical systems because it provides remote and ubiquitous access to medical services.

The rest of this chapter is organized as follows: Section 1.1 explains the general components of medical imaging systems. The description will provide a better understanding for the medical imaging systems. The problems that this thesis aims to address are described in section 1.2. Finally, section 1.3 presents the thesis structure.

1.1 Medical Imaging Systems

Medical imaging systems provide three main services: image acquisition, storage and presentation. Medical imaging systems employ several components to provide services to the users. The components can be classified into three major groups [7]:

1. Modalities: This group includes software applications and hardware equipments

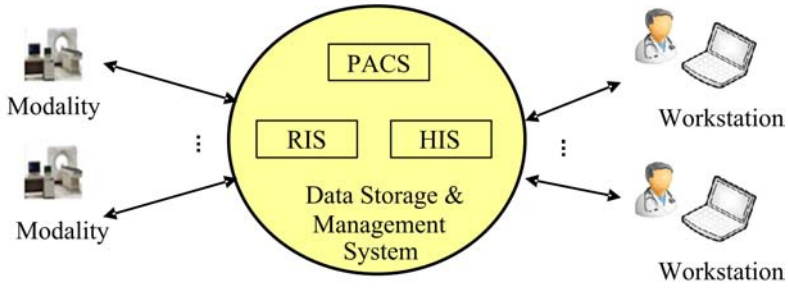


Figure 1.1: Components of medical imaging systems.

that acquire images from patients. Scanners (e.g. CT, MRI, etc.) and their related software applications are examples of this group of components.

2. Data Storage and Management Systems: This group contains all technologies and elements that aim to organize and maintain acquired images and all data associated with patients. Hospital Information System (HIS), Radiology Information System (RIS), and Picture Archiving and Communications System (PACS) are the main components of this group [8]. The first two components (i.e. HIS and RIS) deal with the patients information, while the last component (i.e. PACS) is responsible for managing images.

3. Workstations: This group aims to display and process data. It usually consists of different medical imaging visualization/manipulation tools. The group also has some utilities that enable users to read, change, or create clinical relevant patients' information.

These groups (modalities, data storage and management systems and workstations) are depicted in Figure 1.1.

All applications, tools, and utilities running on the workstations should connect to the components of data storage and management system to access data. Specifically, medical imaging applications need to communicate with the PACS which manages imaging data. This communication is carried out through the Digital Imaging and COmmunication in Medicine (DICOM) [9] protocol.

1.2 Problem Description

As mentioned above, telemedicine and collaboration are two outstanding features that many researchers and developers have provided in their medical imaging applications [5; 6; 10–14]. However, these applications are designed with remote accessibility and collaboration features and they do not offer the features to the applications that are already developed. Nonetheless, there are many developed applications that lack the features. Efforts to add these features may require some modifications which are sometimes severe or impractical. One of the main goals of this thesis is to provide a method to add these features to the medical imaging applications that are already developed but lack support for remote accessibility and/or collaboration. More importantly, the proposed method imposes no modification to medical imaging applications.

All medical imaging applications including collaborative and telemedical applications need to support the architecture shown in Figure 1.1 to be clinically useful. Nonetheless, there are some medical imaging applications like Computer Aided Diagnosis (CAD) tools that do not connect directly to the PACS to fetch appropriate data and do not support DICOM format. Recently, some researchers have noticed

this problem and presented their approach to address it [15–17]. This thesis also addresses this problem by providing an effective approach that not only connects the existing applications to the PACS, but also improves the transmission time for fetching data.

In summary, three aspects, namely, enabling telemedical services, supporting collaboration, and PACS-connectivity are the issues that are addressed in this thesis. A three-tier architecture has been used and several components have been developed within the architecture to provide these features.

1.3 Thesis Organization

The rest of the thesis is organized as follows: Chapter 2 explains related work and presents previous attempts that have tried to address the issues described in Section 1.2. Chapter 2 also gives background (e.g., three-tier architecture) for basic concepts used in the proposed approach. Chapter 3 describes the proposed method and brings in the details of the approach. Chapter 4 explains the implementation. Evaluation and experimental results are introduced and discussed in Chapter 5. Finally, the conclusion and future works are explained in Chapter 6.

Chapter 2

Related Work

This chapter discusses research endeavors that address the issues mentioned in Chapter 1; namely, remote and ubiquitous accessibility, collaboration and PACS-connectivity in medical imaging applications. Previous research studies are compared with the approach proposed in this thesis. An overview of the DICOM protocol, as one of the basic concepts used in this thesis, is given. Then, the tier-based architecture concept and the approaches that use this concept are mentioned. Finally, the use of interfaces, another essential concept used in the thesis, is explained.

2.1 Telemedicine

Telemedicine is one of the technologies noted for its valuable contribution to medical imaging informatics. Its main objective is enabling healthcare delivery at any time and any where irrespective of geographical location [6]. As a result, physicians can remotely access the patients information. Data can be downloaded for observation

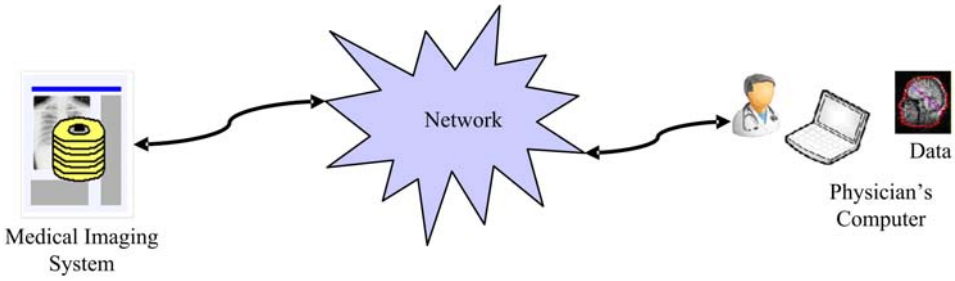


Figure 2.1: Accessing patients' information in telemedical systems.

and further processing or analysis (Figure 2.1).

Telemedical systems are widely used in many medical applications such as oto-laryngology [18], speech-language pathology [19], respiratory applications [20], and cancer care [21].

Telemedicine and its branches (e.g., Tele-consultation, Tele-monitoring, etc.) have inspired different groups to research and develop applications and services that are remotely and pervasively accessible.

One example of these efforts is UCIPE [22], which tries to supply ubiquitous medical imaging services. It provides different image processing algorithms accessible via web services.

DISMEDI [23] is a prototype system for remote medical image processing. It provides different tools and services for 3D image processing (e.g., 3D segmentation, histograms, etc.) that are remotely accessible.

TeleInViVo [24; 25] is another application with the objective of introducing pervasive and easy-to-access medical imaging services. The striking features of TeleInViVo are fast volumetric visualization algorithms and efficient network collaboration tools for remote data analysis and consultation. TeleInViVo lets users interact with a

shared data set across geographically distributed locations.

CoMed [26] is a medical application that allows specialists to share patient records and consult with each over the Internet. CoMed consists of a multimedia medical database with relevant data about laryngeal diseases and offers real-time collaboration services (e.g., teleconferencing).

TeleMed [27] attempts to provide a real-time collaborative web-based medical system. Using TeleMed, multiple physicians, possibly located remotely across a wide-area network, can consult on a patient record with media-rich graphical tools.

VIVE [28] is a grid-enabled interactive analysis tool with a simple web-based user interface for 3D medical images. It provides facilities for diagnosis, therapy, and surgical planning.

Triantafyllou et al. [29] implemented a web portal for managing patients records. Doctors can access medical data from anywhere via this portal. It also supports collaboration.

Peyton and Hu [30] developed a framework based on Service-Oriented Architecture (SOA) for remote medical consultation. The framework supports collaboration and a secure way for sharing data among doctors over the Internet.

Caceres et al. [31] developed a telemedicine web system to enhance the quality of care for HIV/AIDS patients. The system provides a collaborative tool for multidisciplinary care teams including doctors, HIV specialists, nurses, and psychologists.

Although many efforts have been directed towards supplying telemedical systems, most of the systems do not communicate with PACS, and as a result the systems cannot be directly used in clinical workflows.

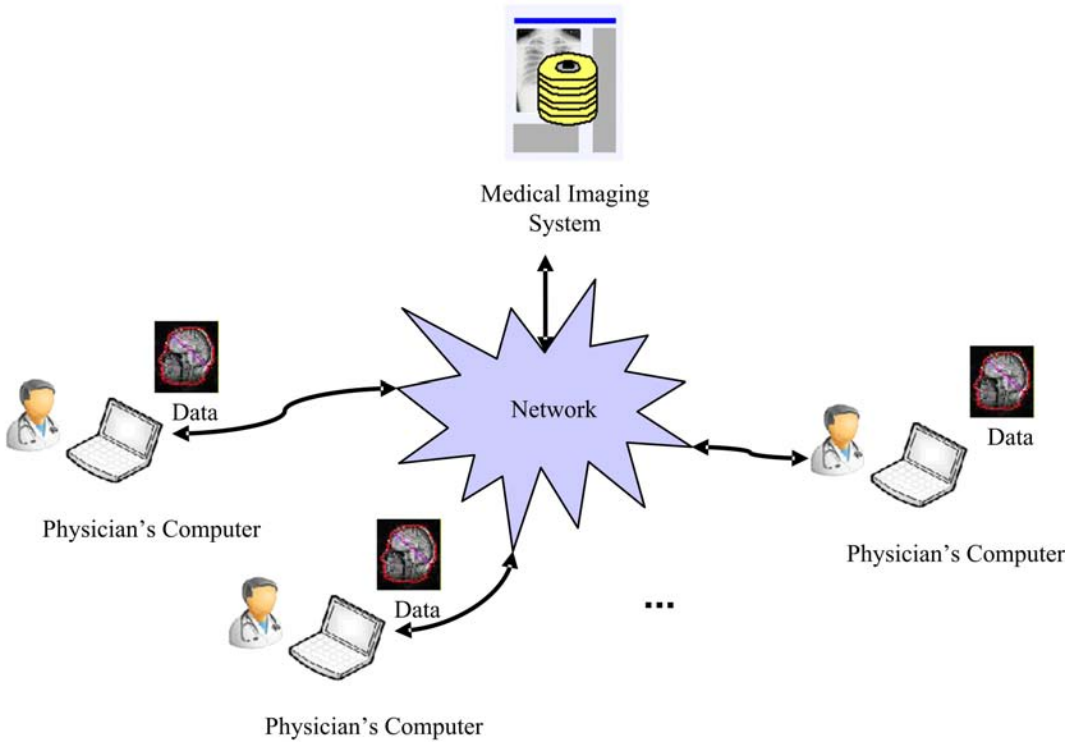


Figure 2.2: Medical collaborative systems.

2.2 Collaboration

Nowadays, collaboration among doctors has become an essential part of medical systems [5]. This feature helps a group of specialists work together. It is usually combined with remote accessibility. In other words, collaboration usually comes with telemedicine. Tele-consultation and tele-conferencing are two well-known examples of this kind. Using this feature, a group of specialists can remotely consult on a medical case (Figure 2.2).

Emerging modern technologies in telecommunication have promoted the development of new collaborative tools. Several telemedical systems support collaboration, like TeleInViVo [24; 25], CoMed [26], TeleMed [27] and the systems developed by

Triantafyllou et al. [29], Peyton and Hu [30], and Caceres et al. [31]. However, there are other collaborative systems that are not mentioned in the previous section.

Mayer and Meinzer [32] presented a collaborative system, which is based on a client/server approach. It supports a collaborative, real-time environment for medical image processing.

KAMEDIN [33] is a telemedical system which supports collaboration. It is equipped with tools for medical image processing and visualization and it provides video and audio conferencing capabilities..

SOMWeb [34] is another example of a collaborative system. SOMWeb is a semantic web-based system for supporting collaboration among distributed medical communities specifically for sharing complex oral medicine cases.

Chen [35] introduced a medical knowledge system. It supports collaboration among physicians at different locations and provides services for sharing and managing medical data.

Wossner et al. [36] developed a collaborative system for users located at different places. The system provides tools for users to collaboratively view volumetric medical data.

Computer Supported Cooperative Work (CSCW) [37] is a well-known field with the goal of studying techniques providing collaboration among several users. The users can work as a group to achieve a common task. There are several high technology solutions for supporting cooperative works. Communication technologies help us to overcome the geographical distribution of collaborators. CSCW characteristics have been supported in several medical systems.

GATiB-CSCW [38] suggests a conceptual model and an architectural proposal to satisfy the requirements of a medical CSCW system.

Wang et al. [39] developed a medical teleconsultation system. The system provides medical image processing tools and CSCW functionality.

Lu [40; 41] has presented a CSCW based remote medical diagnosis system. The system supports collaboration enhanced by videoconference, with a focus on oral healthcare.

Schulte et al. [42] presented a framework for CSCW systems to support medical research. The framework is called Wasabi and implements a service-oriented architecture.

Park [43] conducted research on web-enabled CSCW system architectures. He evaluates the feasibility of a collaborative medical system and proposes a modeling concept and architecture for web-enabled CSCW systems.

Ganguly and Ray [44] investigated the design issues of an interoperable CSCW system in a distributed healthcare environment. They conducted their research in the application of telecardiology.

Like telemedical systems, most of collaborative medical systems do not support PACS connectivity. Section 2.3 describes the studies that try to address PACS connectivity issue.

2.3 PACS-Connectivity

As we mentioned in the previous sections, many telemedical collaborative systems do not communicate with PACS and as a result cannot be directly used in clinical

workflows. Nonetheless, there are some approaches that have noticed this essential feature [7; 45; 46].

Maani et al. [7] proposed a framework that is PACS-based. The framework uses a parallel processing center based on parallel architectures to process medical data. Syngo [45] is another approach proposed by Siemens that provides collaborative remote PACS-based environment for doctors. Noor and Saman [46] developed a three-tier architecture based on the distributed Java model. Their approach also supports PACS.

Nonetheless, the abovementioned approaches do not provide any solution for medical applications that are already developed. Recently, some research works have tried to connect previously developed applications to PACS, like research works that integrate CAD tools with PACS [15–17]. However, these works do not provide collaboration and remote access to the applications.

To provide connectivity to PACS, medical systems need to communicate with PACS servers. Communication between medical imaging applications and specifically medical image transmission plays an important role in telemedical applications. The communication is based on the DICOM protocol. DICOM has several advantages such as interoperability, integrity and consistency which have made it the worldwide de facto standard for interconnecting medical imaging systems [47; 48].

One of the advanced features of DICOM is compression. Image compression is a well-known method for reducing file size and speeding up transmission and there have been many attempts to utilize compression in DICOM [49]. In spite of the fact that the protocol supports compression, DICOM images are usually stored and

transmitted in uncompressed data format, which increases the size of the bandwidth required for transmission [47].

The main reason for not using compression is that barriers exist in many medical applications [50]: A compression technique can be used between two medical applications if and only if both applications support it. If one of the applications does not support compression (which is frequently the case), both applications use the default uncompressed option. Moreover, all image compression algorithms are based on some compression parameters (e.g., compression ratio). These important parameters are not usually clear and negotiable between two applications exchanging medical images. When two applications negotiate in the first step of the communication, they usually agree on the name of the compression technique and possibly its version, but do not convey the compression settings. In other words, compression parameters are not negotiable between two DICOM applications. Therefore, many DICOM application entities do not implement compression unless the image transmission is for proprietary (i.e. manufacturer-specific) usage.

This thesis presents a novel approach [51; 52] to address the problem of improving medical data transmission speed. The proposed approach is based on the combination of parallelism and compression methods in the storage services of the DICOM protocol. The approach also uses interfaces to obviate any change in existing medical imaging applications. In the proposed approach, a pair of interfaces is used to implement the idea and provide a fast method for medical data transmission. The interfaces listen to the incoming messages from the DICOM application entities, intercept the messages, and carry out the data transmission in parallel. The next section gives an

overview of the DICOM protocol.

2.4 DICOM Protocol

DICOM is the de facto file format standard as well as communication protocol for medical images. Any two medical imaging systems need to use this protocol for communication and the proposed approach for PACS connectivity is based on this protocol. This section provides a brief overview of the DICOM protocol [50]. The terminology and major components of the DICOM protocol are presented in subsection 2.4.1. Then in subsection 2.4.2 the way that two DICOM applications communicate is demonstrated.

2.4.1 DICOM Components

The DICOM protocol is based on the Open System Interconnection (OSI) [53] network protocols architecture, which consists of different layers. Each layer has some predefined tasks and the corresponding layers communicate to each other through predefined protocols. The most common architecture uses five layers. The layers from bottom to top are [54]: Physical, Data Link, Network, Transport and Application Layers.

DICOM is an Application Layer protocol. It is compatible with TCP/IP protocol suite and therefore can be used over the Internet. In DICOM terminology, an application using the DICOM protocol is called an Application Entity (AE). Figure 2.3 illustrates the communication between two AEs using the DICOM protocol.

The DICOM protocol consists of two layers (Figure 2.4):

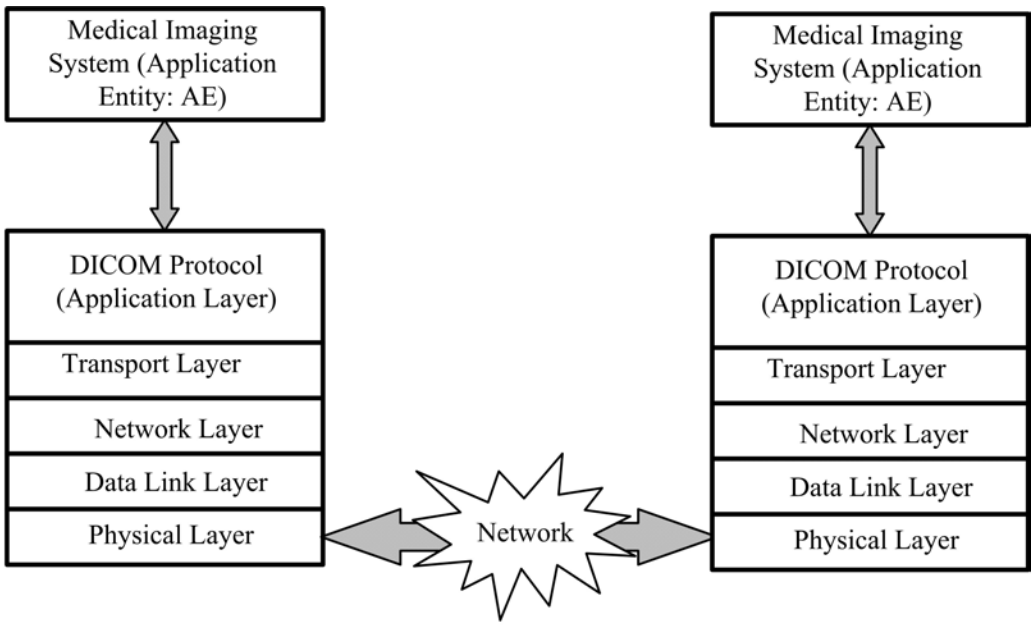


Figure 2.3: DICOM as an application layer protocol.

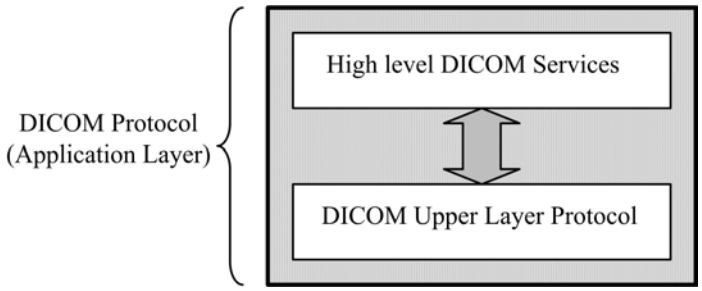


Figure 2.4: DICOM protocol layers.

1. High-level services: AEs asks for different services defined in the DICOM protocol. Services such as finding or printing a medical image are two examples of high-level services provided to AEs.

2. DICOM Upper Layer Protocol (ULP): high-level services are built on top of the ULP. It is directly connected to the transport layer (i.e., TCP) and connects the high-level services to the transport layer.

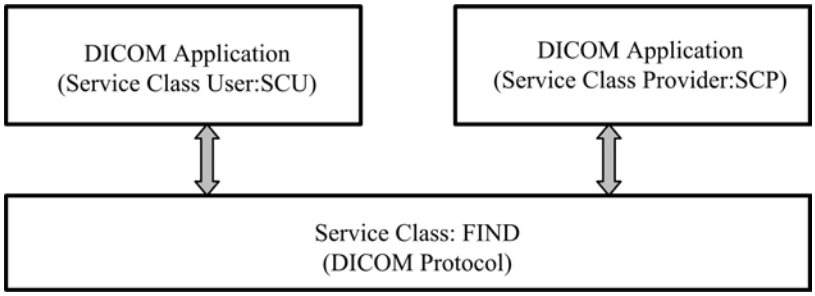


Figure 2.5: Service Class, Service Class User (SCU) and Service Class Provider (SCP).

Each AE can either request or provide one of the services defined in the DICOM protocol. The service is called Service Class. When an AE requests a service, it plays the role of a Service Class User (SCU) while the AE providing the service plays the role of a Service Class Provider (SCP). As an example, when a DICOM application wants to find an image in a DICOM server, the DICOM application is the SCU, the DICOM server is the SCP, and the DICOM FIND service is the Service Class (Figure 2.5).

Each Service Class consists of data and a function related to that data. For example, an MR image can be bound with different functions such as printing or storing; therefore, in this case the MR image can be associated in different Service Classes. Hence, in the DICOM protocol, each Service Class consists of two parts:

1. The object instance (e.g. an MR image). This object is called the Information Object Definition (IOD).

2. The function or service for the object (e.g. storage) which is called the DICOM Message Service Element (DIMSE).

Since each Service Class is a pair of an Object and a Service, it is called the Service-Object Pair or SOP (Figure 2.6).

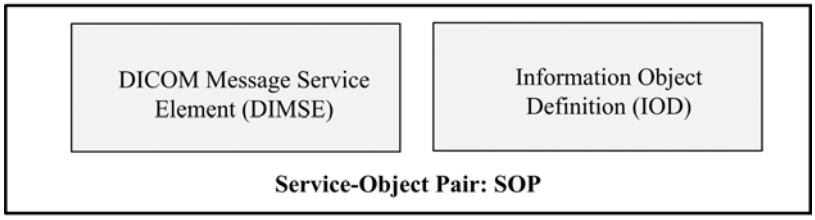


Figure 2.6: Service-Object Pair structure.

2.4.2 Communication between DICOM Applications

As mentioned before, all services of the DICOM protocol are based on the ULP. ULP follows Association rules to guarantee that two AEs communicate with each other properly. In the beginning of the communication, the two AEs exchange communication parameters and, based on the exchanged parameters, they choose the communication specifications that are supported by both. This step is called Association Establishment.

Before any agreement between two AEs, the sending AE packs all the information about itself into data packages called the Presentation Contexts (PCs) and sends the PCs to the receiving AE. A PC includes two parts:

1. Abstract Syntax: The functionalities and services that the sending AE supports. Abstract Syntax includes all SOPs that the sending AE supports.

2. Transfer Syntax: data might be encoded with different encodings or Transfer Syntaxes. There are different Transfer Syntaxes. Some of them do not support compression (e.g., Implicit Value Representation (VR) Little Endian). Some support lossy compressions (e.g., JPEG-LS Lossy Image compression) and some support lossless compressions (e.g., JPEG-LS Lossless Image compression). The

list of Transfer Syntaxes can be found in [50].

The Abstract Syntax is non-negotiable and cannot be changed. For example an MRI scanner supports MRI-related SOPs and its functionality cannot be changed to CT-related SOPs. The Transfer Syntax, however, is negotiable. All AEs are supposed to support at least the default Transfer Syntax (i.e. Implicit VR Little Endian).

When an AE receives an Association Establishment request, it checks the Abstract Syntax for each PC. If it supports that Abstract Syntax, it will choose one of the Transfer Syntaxes presented by the sender and packs the accepted PC in the Association Establishment Response message. If the receiver does not support the Abstract Syntax of a PC, it simply does not include the respective PC in the response. In this way both AEs agree on the connection parameters. Figure 2.7 illustrates the Association Establishment step [52]. Note that number of accepted PCs (i.e., k) is equal or less than the number of PCs sent in the Association Establishment Request (i.e., m).

Suggesting different Transfer Syntaxes in the Association Establishment step makes the protocol flexible. The two AEs can agree on the best setting supported by both. In the case that one of the AEs or both do not support any Transfer Syntaxes with compression capability they will switch to the default and mandatory uncompressed Transfer Syntax. Two AEs can use a Transfer Syntax with compression capability if and only if both support it.

After the Association Establishment step, the SCU can request SOPs from the SCP. Then a sequence of SOP requests and responses is exchanged between two AEs. Finally, one AE sends an Association Termination Request and the other responses

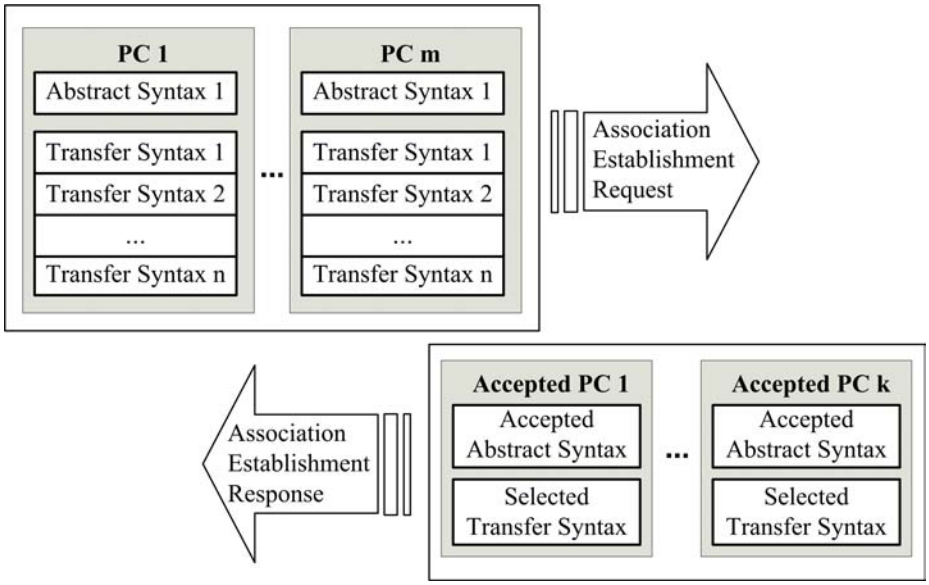


Figure 2.7: Association Establishment step [52].

by an Association Termination Response. This procedure is depicted in Figure 2.8.

The next two sections present two fundamental concepts used in the thesis; namely, three-tier architecture and interfaces.

2.5 Three-tier Architecture

A multi-tier (multi-layer) architecture consists of logically separated layers that are customized to provide different services. Many systems and applications have used a multi-layer architecture, for instance the Open Systems Interconnection Reference Model (OSI Model), the fundamental architecture for computer networks [53], and the current Internet model (a.k.a. the TCP/IP protocol suite) [54] use multilayer architectures.

A multi-tier approach offers several benefits. This architecture is more beneficial during the maintenance process and evolution stage of technologies. Adapting a

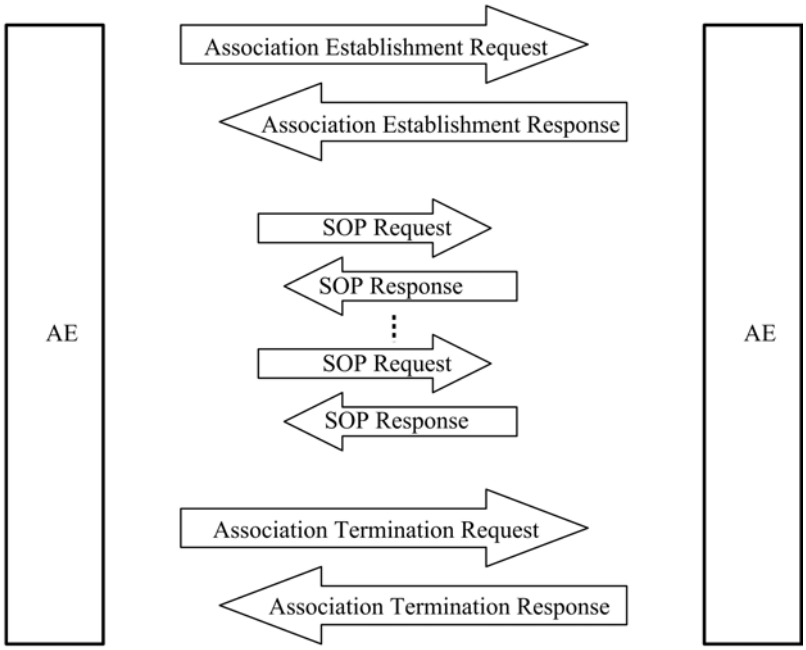


Figure 2.8: DICOM communication steps.

multi-tier system to the incoming technologies is far easier than with other system architectures. In fact, separating the responsibilities and tasks into different parts and organizing them as independent layers leads to more manageable and reliable modules (layers), which can be developed and modified without need for the modification of other layers. Enhancing scalability, performance, efficiency, security and manageability are the other important features of this concept [55–57].

The most prevalent type of multi-tier architecture has three tiers [55] and has been used in medical [45; 46] and non-medical applications [57–61]. The three tiers are the presentation (user interface), the logic (business rules) and the data layers (Figure 2.9).

The presentation tier is responsible for displaying information. It communicates only with the logic tier. For example, the presentation tier sends user input commands

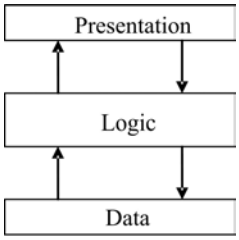
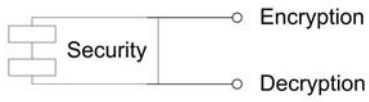


Figure 2.9: The three-tier architecture.

to the logic tier and it shows the information received from the logic tier. The logic tier communicates both with the presentation tier and the data tier. The logic tier aims to handle information exchange between the upper and lower layers. In the software implementation, the logic layer controls the applications functionality by processing the data, performing the requested commands and applying the application policies. The data layer is the place for data management, storage and retrieval. This layer keeps data independent from the logic layer. There is no direct connection between the presentation and data layers. This layer separation helps to apply different strategies, policies and measurements for accessing data. In addition to that, communications between the layers can be independently controlled.

2.6 Interface

Object-oriented programming (OOP) is based on objects as the basic component of programs and their interactions. Each object is an instance of a class. In fact, each class is a structure that encapsulates the properties and functions of the objects. There is a very useful concept in OOP called interface that helps software developers separate the abstraction of a class from its underlying implementation. Each class




```
public interface Security {
  public String Encryption(String str);
  public String Decryption(String str);
}
```

Figure 2.10: The three-tier architecture.

can be manipulated and communicate with the other classes through its methods. The interface indicates only the methods and their specifications (i.e., the methods name, parameters types and the return type). In other words, the interface is an abstraction of a class presented to the other classes.

The separation of the external methods from internal operation and underlying implementation allows classes to be internally modified without affecting the way that the other classes interact with them. The other advantage of this concept is that multiple implementations of one interface can be provided and other classes and components may use any of the interface implementations based on their systems, limitations and specifications. Figure 2.10 illustrates an example of an interface for a security class. This class has two methods called encryption and decryption. The code is presented in Java. This interface might be implemented with different encryption/decryption algorithms.

The concept of an interface has found application not only in software engineering but also in other disciplines. For instance, the interface concept has been employed for hardware components connection (i.e. physical interface) as well as communication between human beings and computers (i.e. user interface). The concept has been widely used in a variety of applications by researchers and software developers [62–

68].

2.7 Summary

In this chapter a background of medical imaging systems was presented. Three trends of remote accessibility, collaboration, and PACS connectivity are discussed and research studies that have developed systems for supporting these features were introduced. Nonetheless, the previous works have not provided a solution for applications that are already developed. Furthermore, this chapter provided an overview of the DICOM protocol and explained the two fundamental concepts used in the thesis; namely, three-tier architecture and interface.

Chapter 3

Proposed Architecture

This chapter explains the details of the proposed approach to provide remote and ubiquitous accessibility, collaboration and PACS connectivity. As mentioned in chapter 2, a three-tier architecture has been used. This chapter also elaborates the tiers, their duties and the associated components developed for each tier.

3.1 Three Tiers

The general architecture of the solution follows the three-tier architectural model. Tiers use interfaces to communicate to each other. The three tiers and their responsibilities are:

1. Presentation Tier: Its function is data display. All software components that show information to the end users reside in this tier.

2. Processing Tier: This tier is responsible for data processing and business logic. The existing medical imaging application is put in this tier.

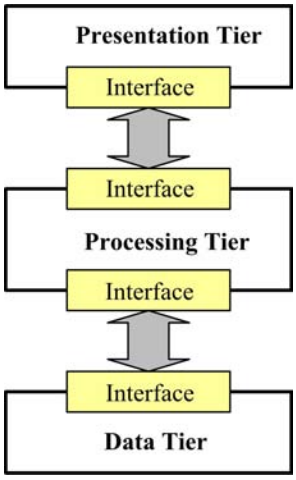


Figure 3.1: The general architecture.

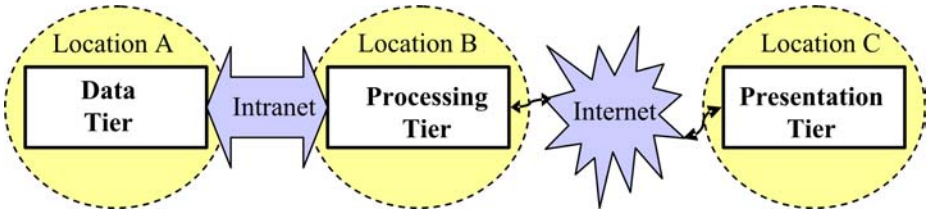


Figure 3.2: The proposed architecture tiers can be located at different places.

3. Data Tier: All data management, storage and retrieval components are located in this tier. In particular, PACS server is located in this tier.

Figure 3.1 depicts the three tiers and the interfaces.

The tiers might be placed in completely different locations and connected via different media (e.g., LAN). Figure 3.2 shows an example scenario in which the data tier is placed at location A connected to the processing tier, located at location B, via an intranet link. The presentation tier is placed at location C connected to the processing tier through the Internet.

The most important part of this architecture is the interfaces. The interfaces perform the communication between tiers. They help to abstract the services between

tiers from the real implementation and underlying technologies. Interfaces work in pairs. There are two pairs of interfaces in the proposed architecture:

1. The pair of interfaces between the presentation and processing tiers: The messages between this pair include users commands, control commands and display data.

2. The pair of interfaces between the processing and data tiers: The messages are related to query/retrieve services. DICOM [9], which is the de facto standard protocol for connecting medical imaging systems, is used for communication between this pair of interfaces.

In addition to these interfaces, several utility components are required to provide different features (e.g., collaboration). The medical imaging application that is intended to be equipped with remote accessibility, collaboration and PACS connectivity is located in the processing tier. Figure 3.3 depicts the three tiers, interfaces, utility components and the messages being exchanged between tiers.

For the pair of interfaces between the data and processing tiers, this thesis proposes a novel idea for data communication, which will be explained in section 3.2. Utility components provide the collaboration and remote accessibility as shown in Figure 3.3. These components are described in sections 3.3 and 3.4.

3.2 PACS Connectivity

Connection to the PACS is an important feature that makes medical imaging applications more clinically useful. To support this feature, the applications should

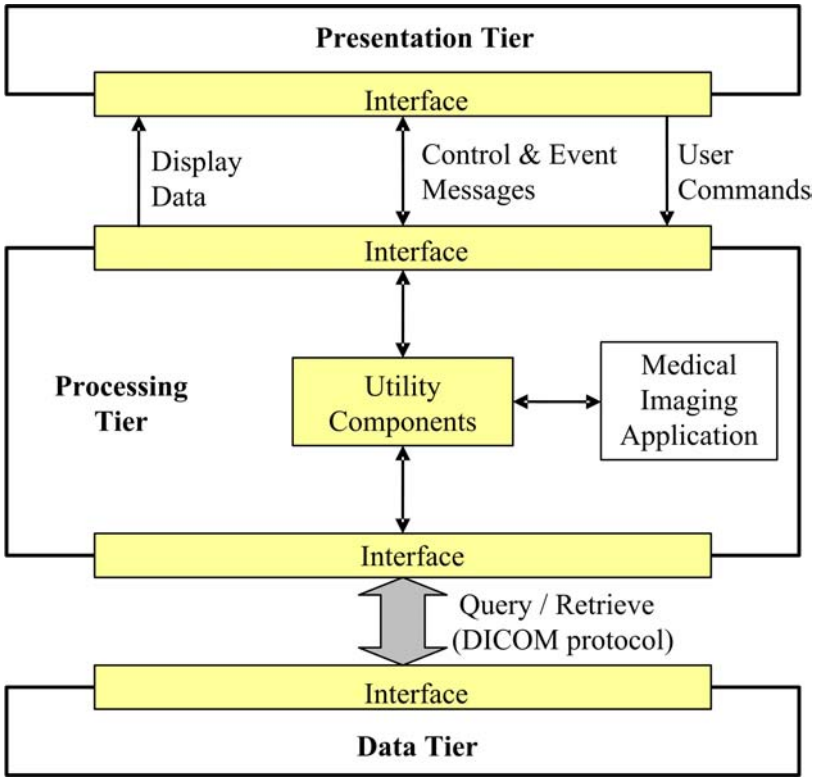


Figure 3.3: The three tier architecture and its components.

know the DICOM protocol and have network components that work with this protocol. Nonetheless, providing an effective communication feature is also important. Medical data transmission has been identified as one of the key issues that require attention to meet the data overload challenge [69].

Data overload has become a challenge for current medical imaging systems in recent years. One reason is that the modern modalities can produce large numbers of high-quality images. For instance, a routine chest CT may produce 300 to 500 images and a CT angiography runoff study may include 1500 to 2000 images [4]. By considering the number of studies performed per patient and the number of patients examined each day, we can see a large amount of data that medical imaging systems

have to deal with every day.

The common way of communication between medical imaging systems is based on a single connection. Many researchers have tried to develop compression methods to enhance medical image transmission time. However, compression methods impose some overheads and sometimes, especially over high-speed networks, these overheads increase the transmission time [50]. This proposal uses a novel idea [51; 52] based on a combination of parallelism and compression methods and the DICOM protocol. The method, however, can be used by any two medical imaging applications that exchange data without any underlying assumptions (e.g., architecture, OS) for communication.

To enhance medical data communication, parallel TCP connections are used in the pair of interfaces. Parallel TCP connections (also known as parallel TCP sockets, parallel TCP streams, parallel TCP flows, and parallel TCP data channels) is a widely used terminology for a very well-known concept [69–73] in computer networks. In parallel TCP connections, bandwidth is shared between N parallel connections. This method overlaps not only compression time and transmission time but also transmission times among several connections. The next subsection elaborates using parallel TCP connections.

3.2.1 Parallel TCP Connections

Although many research studies have developed very good compression techniques, use of compression alone is not always effective. It is useful when it is worth adding the overhead of compression and decompression to the transmission time. The time of transmission should be large enough in comparison with the compression overhead

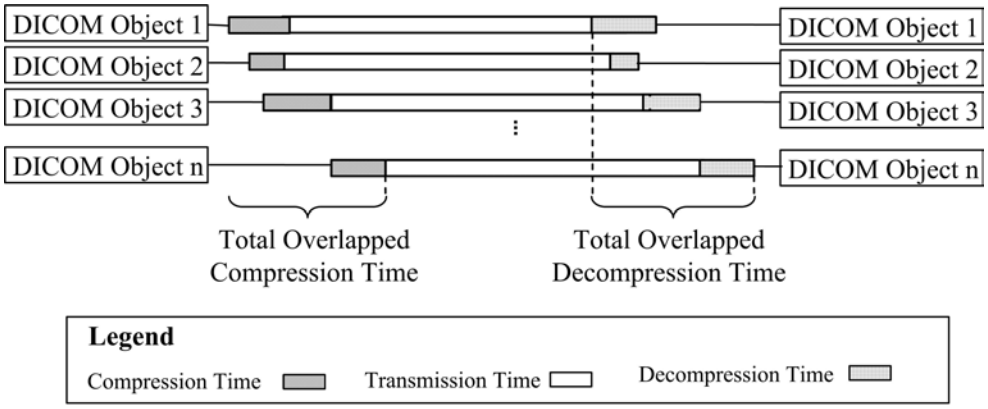


Figure 3.4: Using a combination of compression and parallelism [52].

time.

This thesis proposes parallel TCP connections. Parallelism can be used with compression and can improve its efficiency even in high-speed networks where using compression method deteriorates transmission time. This improvement is achieved by overlapping the compression overhead times with transmission time [52]. Figure 3.4 shows this concept.

The overhead times can be reduced remarkable by this technique; that is, the total overlapped compression time is significantly less than the aggregation of non-overlapped compression times. The reduction by the means of overlapping occurs for transmission times too. In fact, the total overlapped transmission time is much less than the aggregation of all non-overlapped transmission times. In chapter 5 the results of experiments for assessing this method are presented which show the superiority of this combination over the common method of medical image transmission that uses single connection.

A pair of interfaces has been used between two Application Entities (AEs) to provide the parallelism. Figure 3.5 compares the proposed method with other current

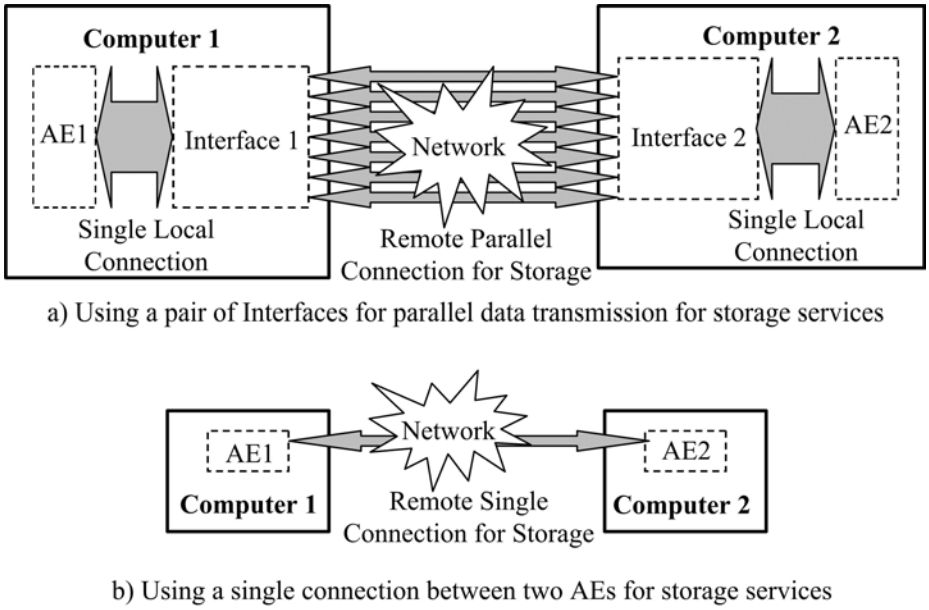


Figure 3.5: Using a pair of interfaces for parallel data transmission in comparison with using a single connection between two AEs.

methods that use a single connection.

The best place for running the interfaces is the same computer as each AE is running, so the data transmission between the AEs and interfaces can be carried out in real-time. The connection between the two interfaces is in parallel. The parallelism helps utilizing the available network bandwidth capacity and thereby it speeds up the data transmission.

By using a pair of interfaces, the need for changes in the current AEs is obviated. In other words, any two existing DICOM AEs can take advantage of using the parallel storage capability without change. Each AE sends and receives data to its corresponding interface. This communication (between the AE and the interface) is carried out by a single connection that uses the default Transfer Syntax.

Parallelism occurs only in Storage SOPs where medical images are being trans-

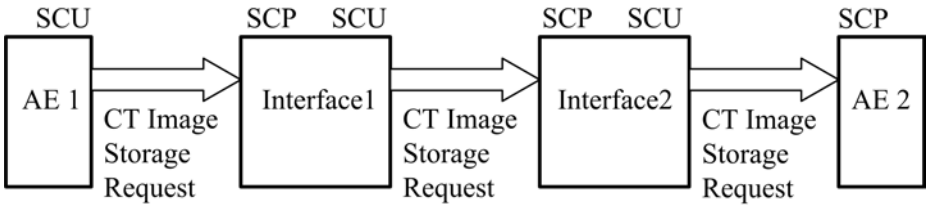


Figure 3.6: Assigning both SCP and SCU roles to the Interfaces.

ferred and in other cases only one connection is used between the interfaces. Whenever a storage SOP is requested and an image is ready to send, a new TCP connection will be created and used in the interfaces to transmit the image.

The next subsection explains the structure of the interfaces.

3.2.2 Interface Structure

Unlike a common AE that has either an SCP or SCU role at a time, the interfaces have both roles and therefore they have a different structure compared to the common AEs. Depending whether an interface is a sender or receiver, they play one of these roles. An example is illustrated in Figure 3.6. Suppose that AE1 requests a CT image storage SOP from Interface1. In this case, AE1 is the SCU and Interface1 is the SCP. Then this request is sent to Interface2. Now, Interface1 plays the SCU role while Interface2 plays the SCP role. Finally, the request is passed on to the AE2 where Interface2 is the SCU and AE2 is the SCP.

Whenever an interface receives a message, the sender type is checked. If the sender is the other interface, the message is simply forwarded to the AE. If the sender is an AE, then the service type will be checked: if the service is not a storage SOP, it will be forwarded using the single available connection; otherwise, a new storage connection

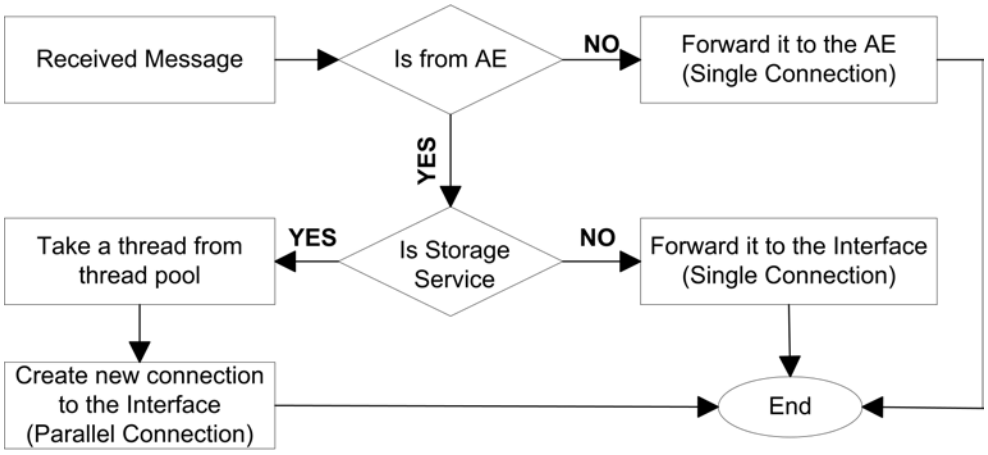


Figure 3.7: Flow diagram of message forwarding in the interfaces.

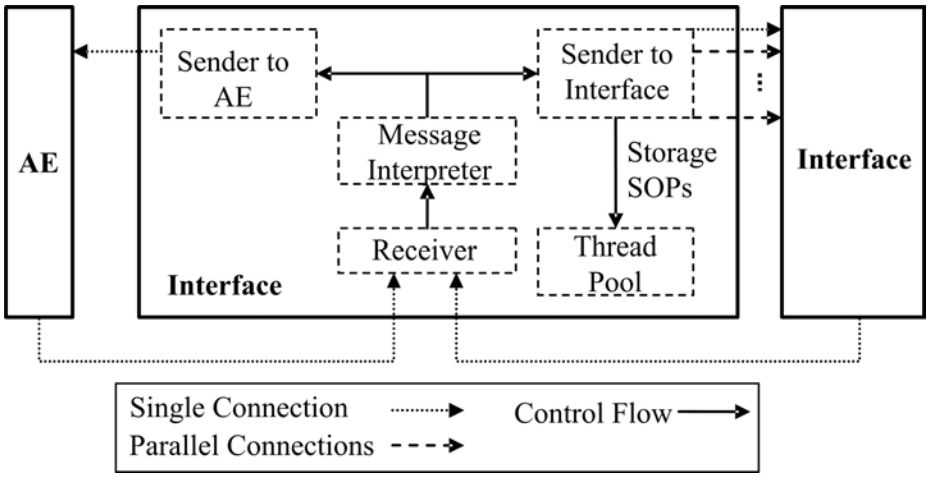


Figure 3.8: Components of the interfaces.

to the other interface will be made. A thread pool has been used for making new connections. Figure 3.7 illustrates the flow diagram of message forwarding in the interfaces. The components of the interfaces are depicted in Figure 3.8.

3.3 Remote Accessibility

All users should be able to connect to the system remotely and see the remote application and work with it. There are several utility components in the processing and presentation tiers. The utility components are responsible for providing remote accessibility for the medical imaging application laid in the processing tier. Users can access the remote application and issue their commands at the client-side Graphical User Interface (GUI). There is a pair of sender/receiver modules on the processing and presentation tiers responsible for providing remote communication between the processing tier and presentation tier. The details of these components are presented in Chapter 4.

There is no direct access to medical data in the proposed architecture. A preprocessing operation transforms DICOM data to a format supported by the medical application. This preprocessing operation prevents direct access to medical data and makes it possible for different medical applications to work with DICOM data even when they do not support the DICOM format.

This thesis uses a component that transforms images in DICOM format to VTK format. VTK [74] is a well-know format introduced by Visualization Toolkit [75], an open-source software system for 3D computer graphics, image processing and visualization. VTK has been widely used in medical image processing and visualization tools and software (Figure 3.9).

The other advantage of the preprocessing operation is security. A medical image (i.e., a DICOM object) includes not only the image itself, but also some private information such as patient's name and birth date. To avoid privacy violations, it

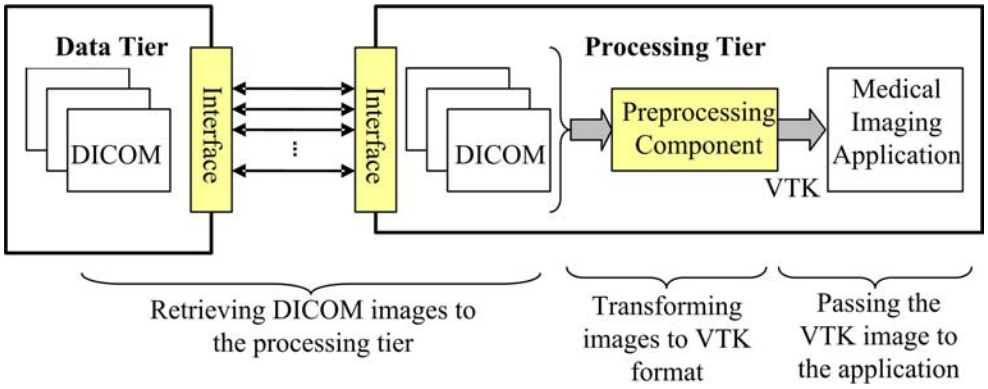


Figure 3.9: Preprocessing operation on DICOM images.

is useful to avoid sending medical images to the end users who sometimes should not have access to the private information. After the preprocessing operation, the transformed data (i.e., VTK image) is passed to the medical imaging application. Then users can access the data through the imaging application.

To provide remote accessibility to the application we need to show the application window to the users. This requirement is met by extracting the application window pixel information and streaming the pixel data to the clients; so the users can see the exact window of the application in their computers (Figure 3.10).

To support this feature, one of the utility components is responsible for extracting pixel information from the screen where the medical imaging application is running. After the completion of data processing by the processing tier (e.g., 3D visualization), the pixel information of the application window is extracted by the corresponding utility component and streamed out to the presentation tier. The feature was inspired by several software tools that extract the pixel information from the screen and send it off to multiple users simultaneously. HP-RGS [76] and Tight-VNC [77] are two good examples of such software products.

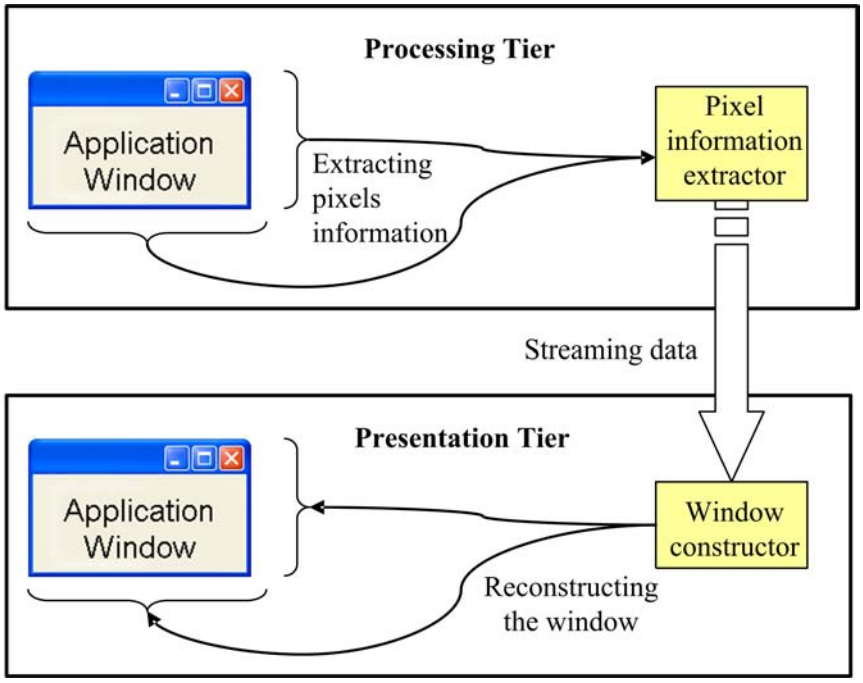


Figure 3.10: Sending pixels information to the clients.

However, unlike HP-RGS and Tight-VNC, this pixel extraction will be restricted to the medical application window and not to the whole desktop. SharedAppVnc [78], UltraVNC [79] and MAST [80] are some good free and open-source examples that provide this facility. This thesis uses MAST to extract pixel information. MAST can run properly between Linux and Windows and the quality of the screen is high. SharedAppVnc could not run properly on recent versions of Linux and UltraVNC works only on Windows platform.

The technique of pixel extraction provides more security because reconstructing raw DICOM images from the processed data is so hard. Medical imaging applications typically do some processing on data, producing results that are different from the raw data. For instance a 3D visualization application combines several images together and applies some algorithms to show a 3D visualized image. Figure 3.11 shows the

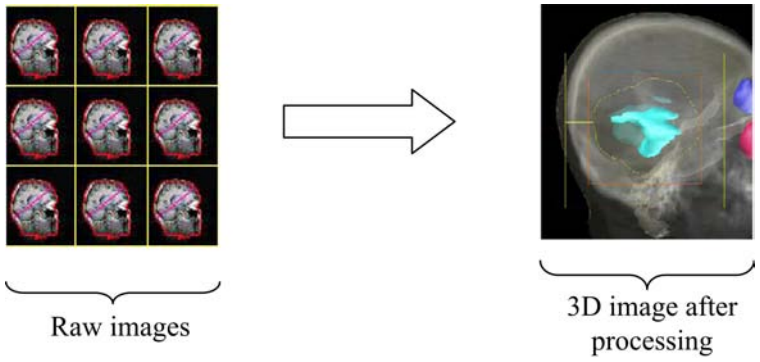


Figure 3.11: Combining raw images and producing a 3D visualized image (pictures are modifications of images in [7] and [81]).

concept (pictures are modifications of images in [7] and [81]).

Like all graphical applications, the medical imaging application maps the 3D visualized image into the pixels on the screen to be displayed for users. Users only need to have the pixel information to see the 3D image and it is very hard to reconstruct the raw images from the processed 3D visualized image (i.e., the reverse procedure shown in Figure 3.11). In addition to security, this feature also helps systems that support real-time access capability because instead of sending all individual images of a medical study, pixel information (that has a smaller volume in comparison with the total volume of individual images) is sent [7].

3.4 Collaboration

The implemented system of the thesis provides Audio and Video tools for the users to collaborate. This capability allows a group of doctors and specialists to view the processed medical images and discuss them. Audio support is provided through an open-source audio conferencing and streaming application called Robust Audio

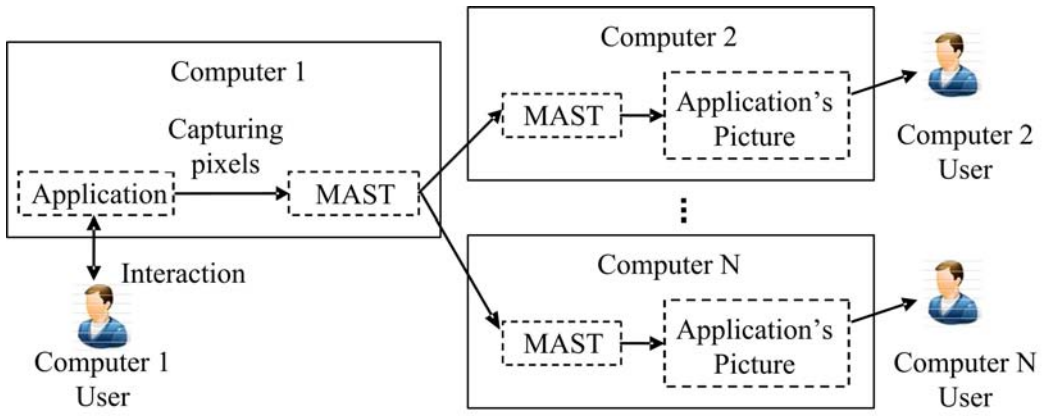


Figure 3.12: Using MAST for extracting pixel information.

Tool (RAT) [82].

For video collaboration, MAST has been used as the major component to stream the application window to the remote users. However, MAST does not support collaboration among users. It only provides the pixel extraction feature.

MAST is developed for users who want to show an application running on their local machine to other remote users. With MAST, the user who shares an application on his/her local machine is able to work with the application locally and all other users can see the result (Figure 3.12).

MAST does not support remote control and a local user controls the application while the other remote users can only see the result (Figure 3.12). However, in the proposed architecture, the application runs in the processing tier that can be a remote server. Thus, there should be some utility components to support users who want to control the application remotely.

Several utility components are developed in this thesis to support these two additional functionalities in MAST; namely, interaction and remote access for all users (i.e., not only the local user but also remote users). These components capture mouse

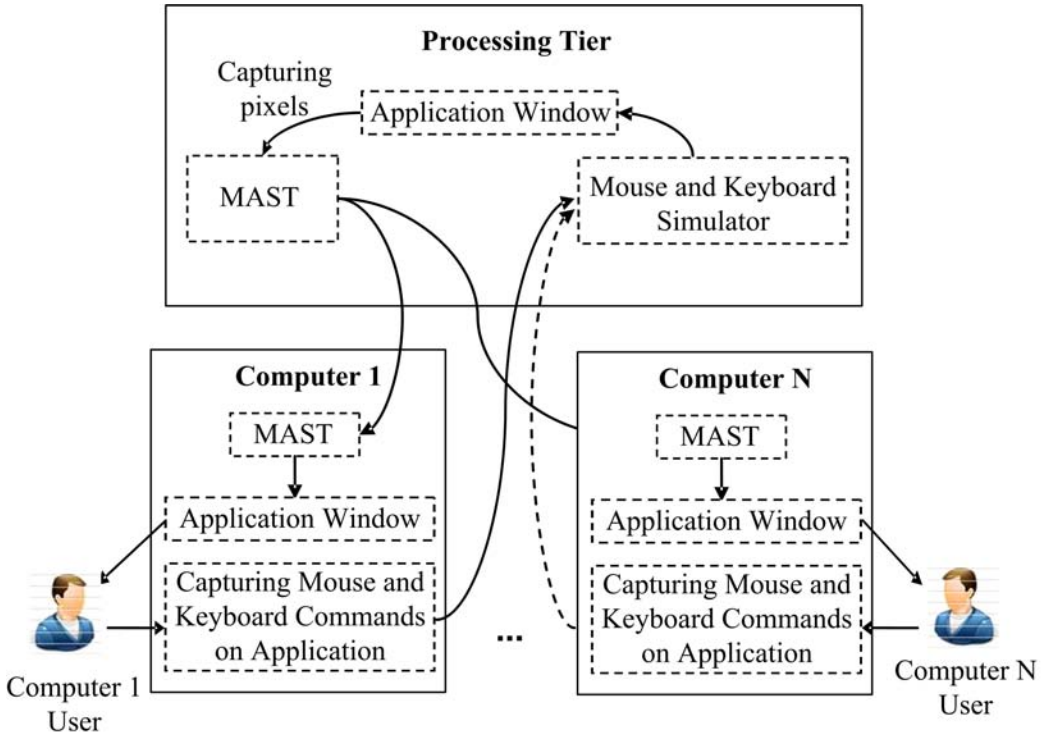


Figure 3.13: Enhancing MAST with utility components to have remote interactive access.

and keyboard events on the application window reconstructed on the remote user's machine.

Figure 3.13 shows the enhanced MAST. To prevent chaos in the application utilization, at each time only one user controls the application. The dashed arrow shows the potential control for user N.

In addition to the critical components that are mentioned in this chapter, several other utility components are developed for managing sessions, authentication, and error handling that are explained in chapter 4.

3.5 Workflow

This section provides an overview of the workflow in the developed system. The workflow explains how a group of doctors and specialists can access the system, search for patient studies and collaborate remotely with the PACS-connected application. ParaView [83] and Slicer3 [84] have been considered as the medical imaging applications. ParaView is an open-source scientific visualization tool with the 3D visualization capabilities. Slicer3 is a free, open-source software platform for medical image processing and 3D visualization.

There are three roles in the developed collaborative system:

1. Leader: There is one person who leads the sessions. This leader can search for patient studies, choose the application on the processing tier, accept or reject the other users requests for controlling the application. The leader can take application control at any time. The first person (who logs in) in each session is the leader. The leader is able to terminate the session at any time.

2. Controller: There is only one person in the group who works with the application and manipulates the image (i.e. rotation, zoom, etc.) at a time. The leader is controlling the application by default. Any user can request for application control; however, the decision will be made by the leader.

3. Viewer: All other users can view the application window and see the changes made by the controller simultaneously. They have no control over the application until they make a request for application control and the leader accepts.

Figure 3.14: the GUI for logging in.

There are three main phases in each collaborative session. The phases are as follows and explained in the next subsections:

1. Collaboration establishment
2. Examination setup
3. Image manipulation and consultation

3.5.1 Collaboration Establishment

At the first phase, users log into the system. Figure 3.14 shows the GUI window that users use for logging in.

The collaboration establishment phase includes the following steps (Figure 3.15):

1. Each user sends a login message request to the server. The login GUI includes username, password and a field that shows if the user is the session leader.
2. When the message request is received by the processing tier, the request will be forwarded to the corresponding security component (i.e., a utility component).

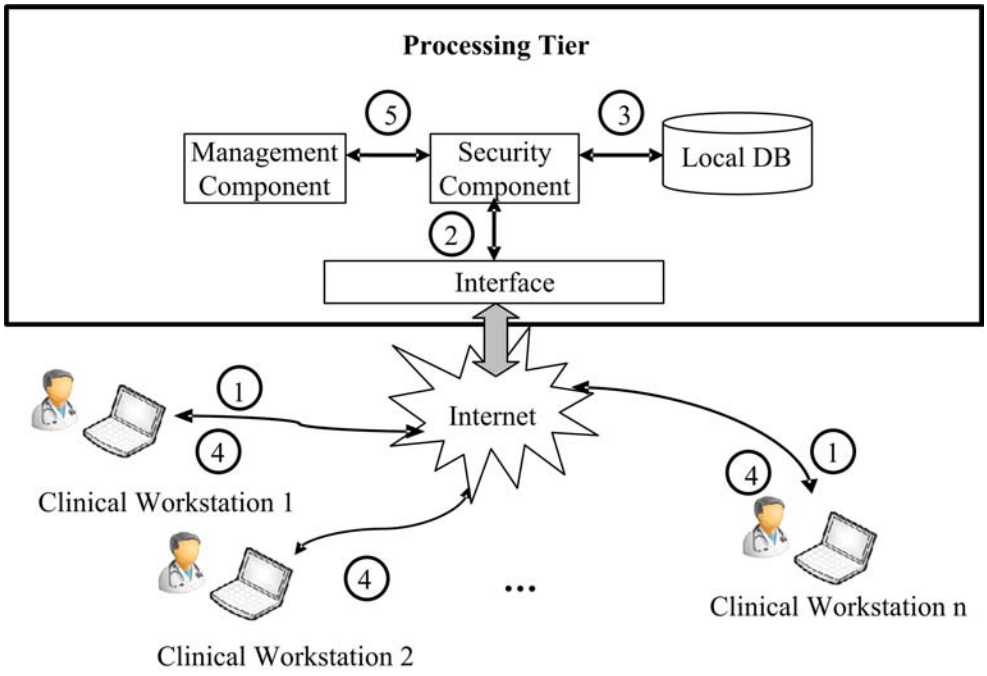


Figure 3.15: Workflow of the collaboration establishment phase.

3. The security component connects to the local DB and authenticates the user. If the user is not authenticated, a failed login message will be sent back to the user. Otherwise the system will proceed to the next step.

4. The user attributes (e.g., user ID) will be retrieved and a successful login message will be sent back to the user.

5. The management component (i.e., a utility component) registers the user and a message with an updated users list is sent to all users. The message also includes the IDs of the session leader and the application controller.

Based on the user role, different GUI windows are shown to the users after they log in successfully (i.e.: leader and common user GUI windows). Figure 3.16 shows these two windows.

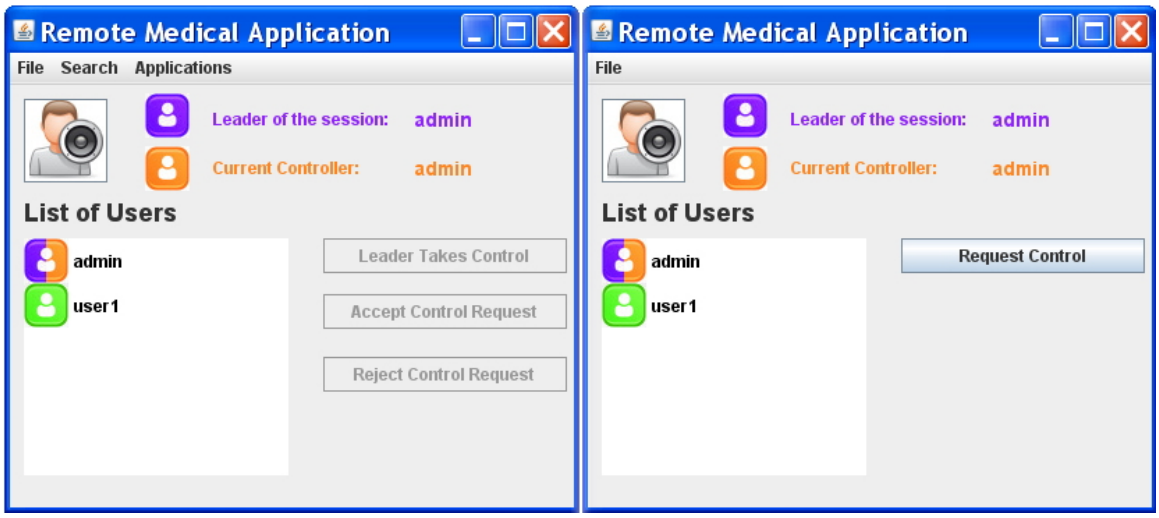


Figure 3.16: Different forms based on the role. The left form is shown for the leader of the session, while the right form is shown for common users.

Some features are provided for both windows. There is a list that shows the names of users who have logged in. The role of each user is shown by an appropriate icon. The leaders icon is violet. The controllers icon is orange and a common user is shown by green icons. The name of the leader and current application controller are shown on top of the window. There is an audio tool button on the top left side of the window. When a user clicks on the audio tool button, the RAT tool will be activated. Figure 3.17 shows the RATs GUI. It shows the list of users who have activated their audio tool. For instance, in Figure 3.17 user Admin has activated the audio tool. The audio tool enables users to talk to each other and discuss the selected patients study. More details about RAT can be found in [82].

There is Request Control button shown specifically on the common users window. Common users can request application control by pressing this button.

On the other hand, there are some features provided specifically for the leader. There are three buttons on the leader's window. When a user requests control of the

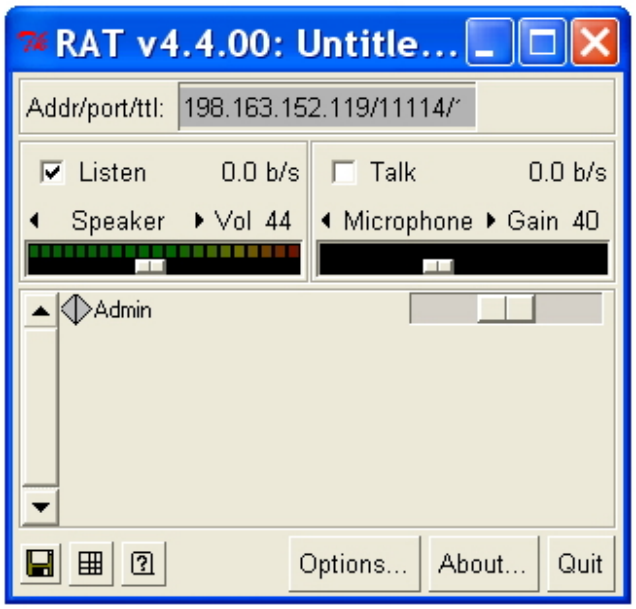


Figure 3.17: RATs GUI.

application, the Accept Control Request and Reject Control Request buttons become active. The leader can accept or reject the request. If a user other than the leader controls the application, the Leader Takes Control button becomes active. Therefore, the leader can take back the application control at any time.

The leader's window is also equipped with tools to search for patient studies and choosing an application to work on the session. These features are available on the menu. Figure 3.18 shows the menu to choose an application for running on the processing tier. ParaView is the default application. If the leader wants to change the selected application, (s)he should change it at this phase before starting the next phase.

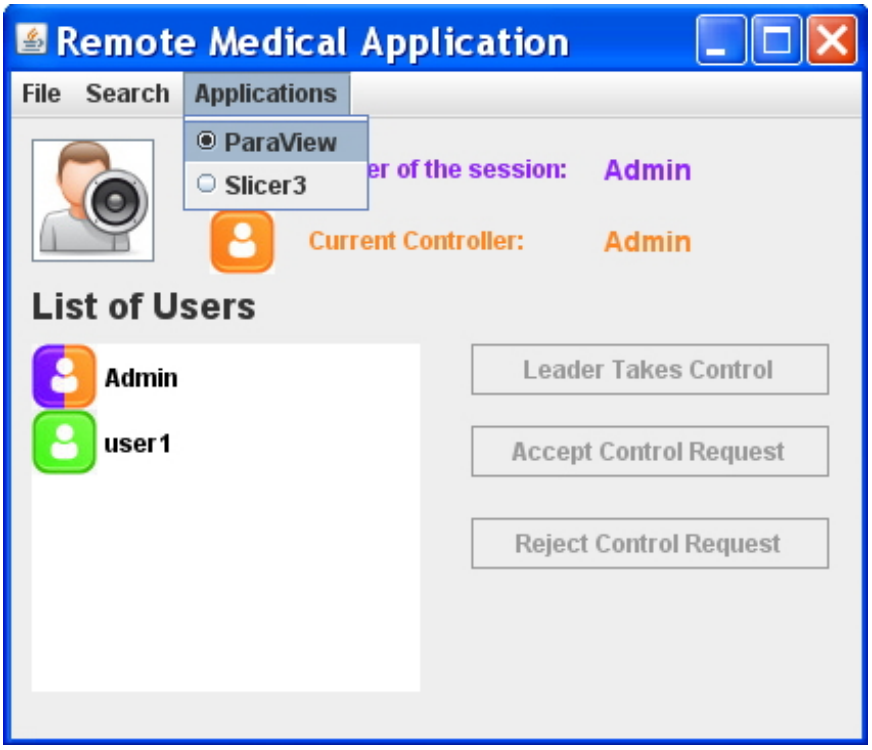


Figure 3.18: Choosing an application for running on the processing tier by the leader.

3.5.2 Examination Setup

Once the collaboration establishment phase is done, the session leader can setup an examination study. The leader is able to search patient studies and choose the one that is intended for consultation.

When the leader clicks the Search menu item, the search window will be shown (Figure 3.19). There are three fields for searching a study (i.e., patients name, study ID, study date). Any of these fields can be completed to do the search. The system generates a DICOM query corresponding to the provided information in the processing tier. The query is sent to the data tier and search results received from the data tier will be sent back to the leader.

Figure 3.20 shows a sample search result window when "patient*" is inserted into

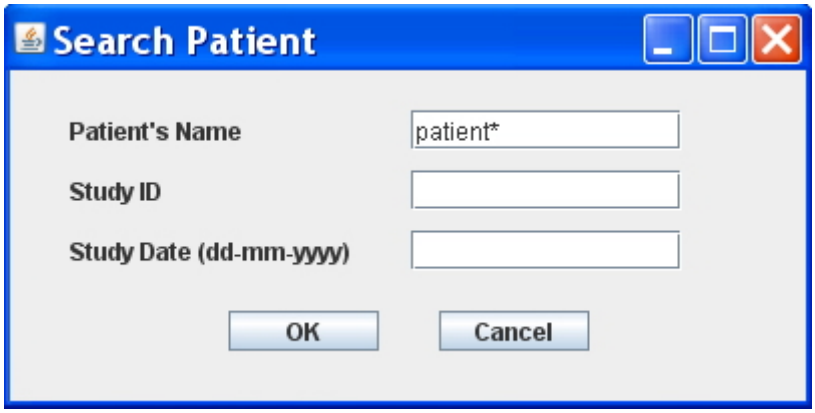


Figure 3.19: Search form.

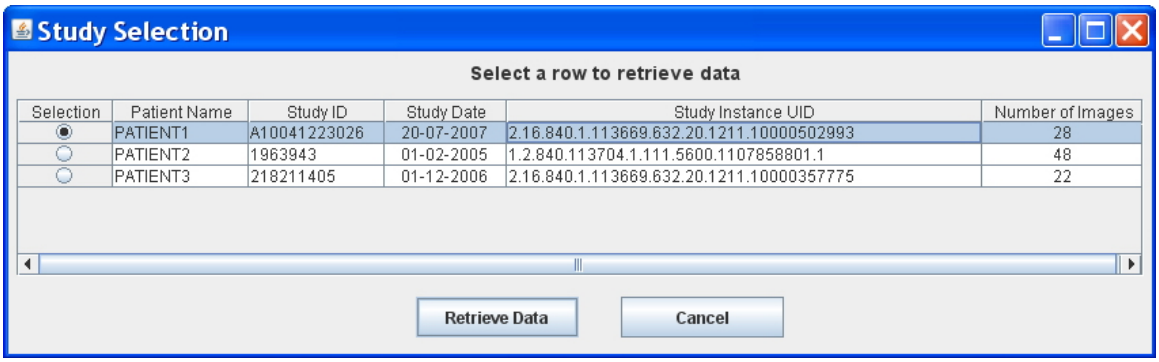


Figure 3.20: Result form shown for the leader.

the Patients Name field. The DICOM server returns all records with patients name starting with the word "patient". If we do not use an asterisk, the search will be restricted to records that have a matching patients name of "patient". Now the leader can choose one of the studies shown in the result window. The leader can cancel these results and search again by providing new search information.

When the leader presses Retrieve Data, an appropriate DICOM command (i.e., DICOM-GET) is sent to the data tier and DICOM images are retrieved into the processing tier.

After retrieving the data, the preprocessing operation is performed and the DI-

COM images are transformed into a file format that is known to the application. Then the application is executed and data will be passed to it. Now the system starts MAST. Since this part of MAST runs on the processing tier it is called the "MAST server" in this thesis.

There is a utility component that finds the handle of the application window. This handle is passed to the MAST server, so the MAST server will know which window should be streamed. At this stage, the presentation tier activates MAST for the users' clients. Since this part of MAST runs on the presentation tier and shows the remote application window, it is called the "MAST client" in this thesis. To view the remote application, users should first click the Connect button on the top left side of the MAST client GUI (left window in Figure 3.21). After connecting, the users can see one Server and all Clients who have activated their MAST clients (right window in Figure 3.21). In order to see the remote application, the users should click on the AppView item.

Figure 3.22 shows a screenshot of the remote application window on the users screen when ParaView is running as the application on the processing tier. Figure 3.23 shows the application window when Slicer3 has been chosen as the remote application. At this step the system is on the third phase; that is, image manipulation and consultation.

Note that at each step, the processing tier sends updating messages to the presentation tier to keep the users updated about the processing tiers status.

There are eleven steps in the examination setup phase (Figure 3.24):

1. The leader starts searching for a study. A search message including information

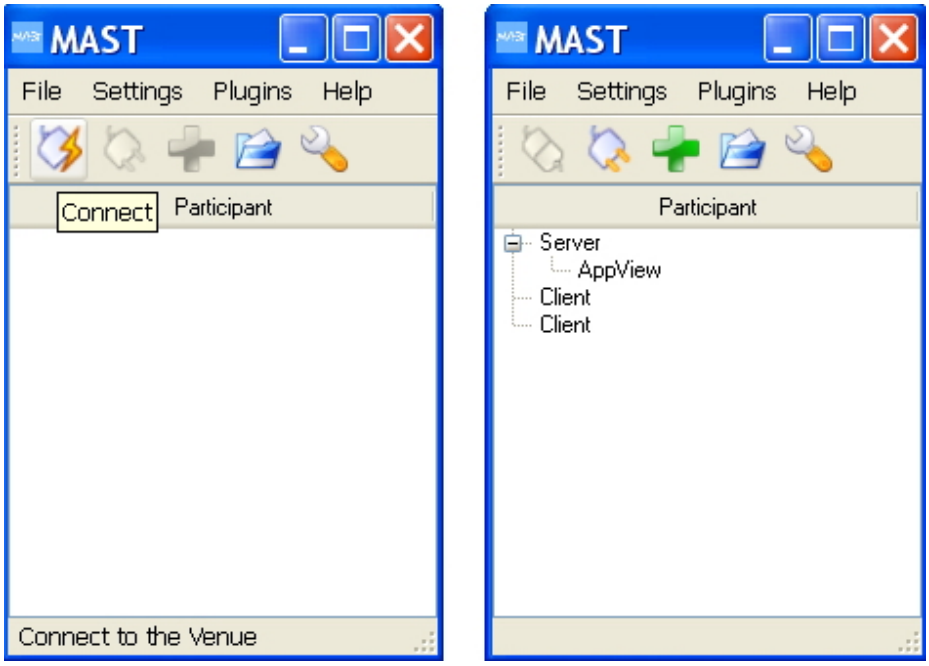


Figure 3.21: Result form shown for the leader.

about patients name, study ID and study date is sent to the processing tier.

2. When the search message is received by the processing tier, the message will be forwarded to the corresponding utility component that is responsible for making DICOM-FIND query request.

3. A DICOM-FIND message is sent to the data tier. The data tier receives the query, searches the requested data and sends back the result to the processing tier.

4. The processing tier receives the result and sends it back to the leader of the session.

5. The result is shown to the leader in the result window. If the leader cancels the results, the system will go to step 1; otherwise, a retrieve message including the

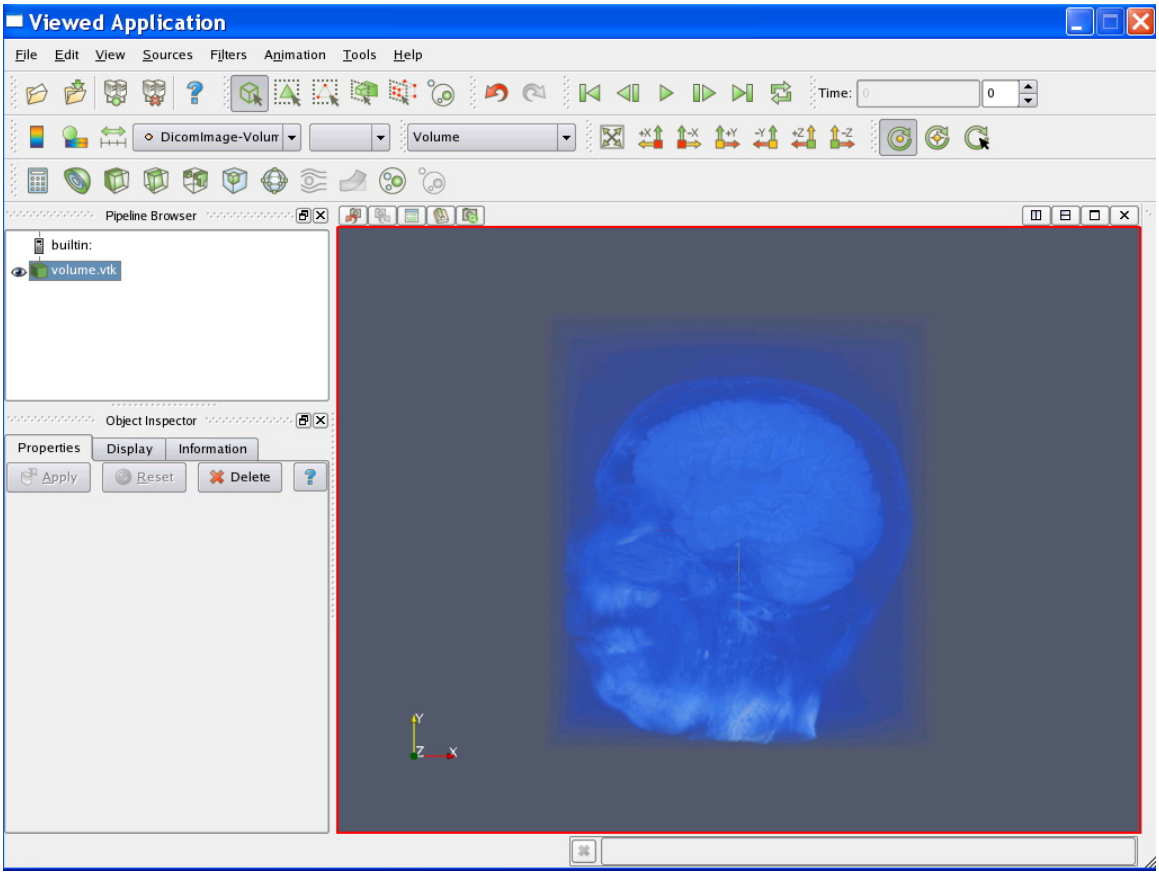


Figure 3.22: Viewing ParaView as the remote application.

chosen study number is sent by the leader to the processing tier.

6. The retrieve message is received by the processing tier and is forwarded to the corresponding utility component.

7. The utility component creates a DICOM-GET message and sends the message to the data tier. DICOM images of the study are retrieved in parallel into the processing center.

8. The DICOM images are pre-processed and transformed to an appropriate data format (e.g., VTK).

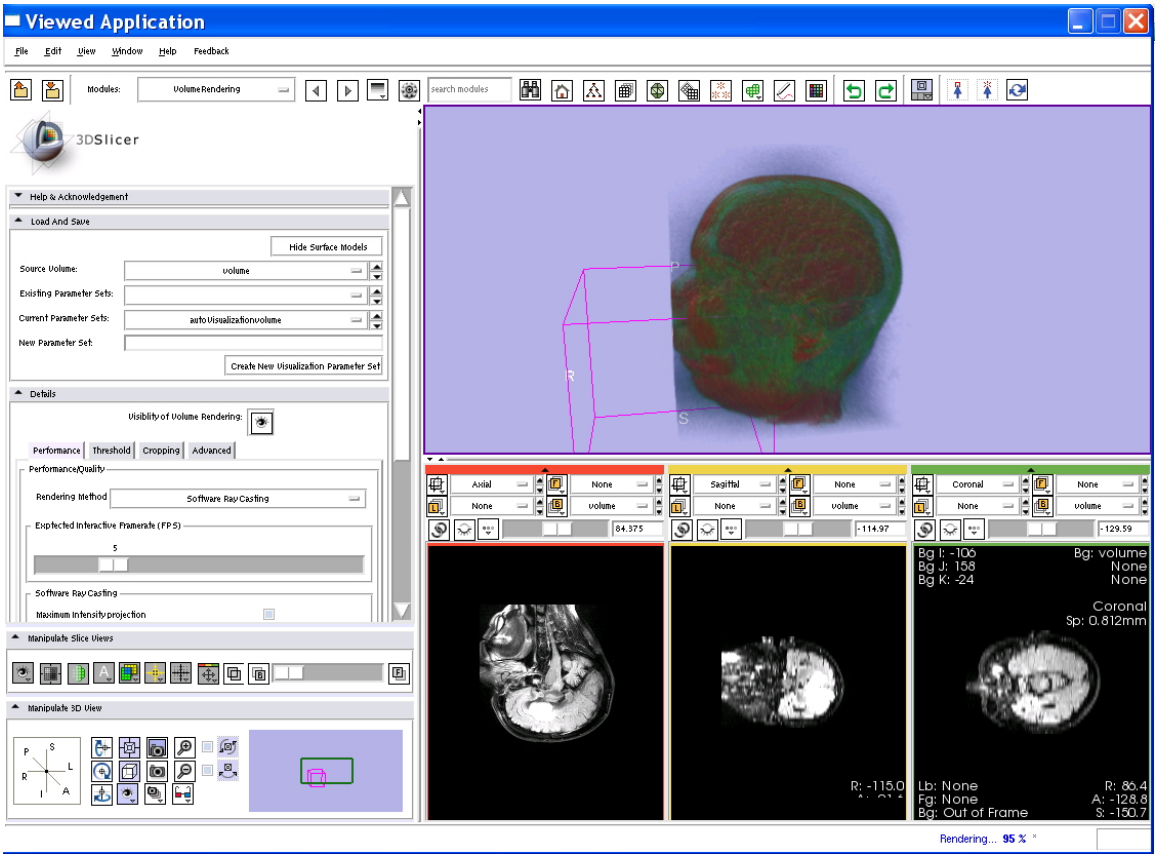


Figure 3.23: Viewing Slicer3 as the remote application.

9. The selected remote application is executed and the data is passed to it.

10. The MAST server runs and starts streaming data to the clients.

11. MAST client is activated in the presentation tier. Users connect the MAST client to the MAST server by pressing the Connect button. They view the remote application.

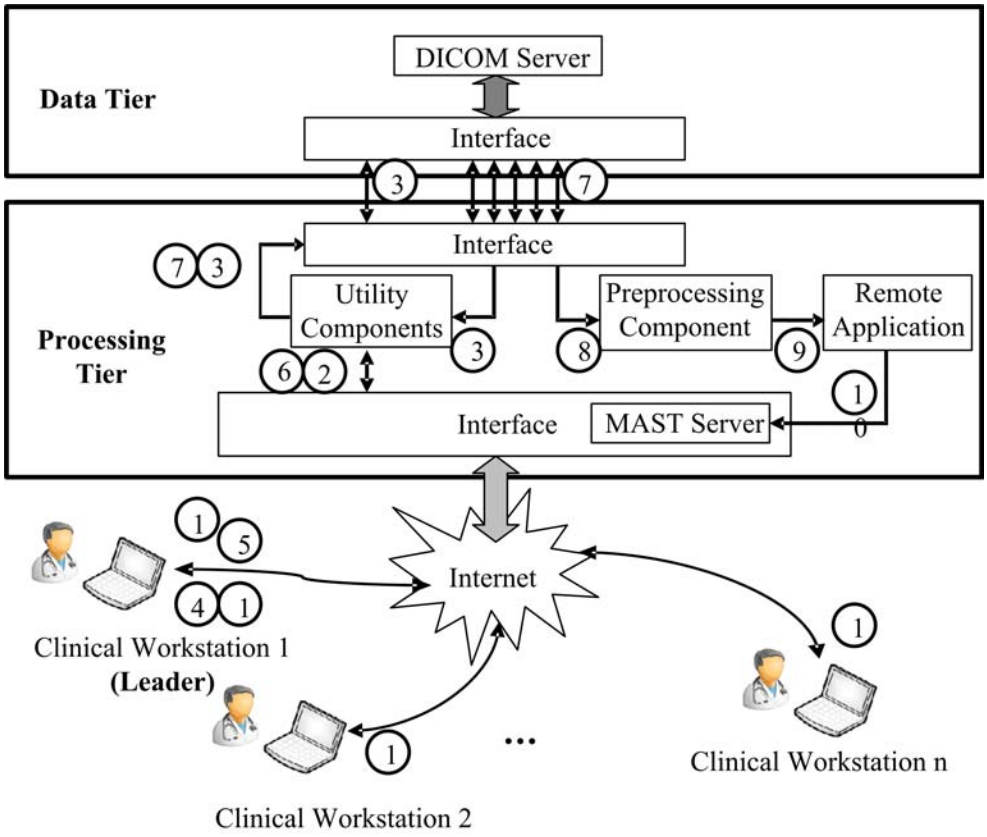


Figure 3.24: Workflow of the examination setup phase.

3.5.3 Image Manipulation and Consultation

Now that all users can view the remote application, they can start working on the study collaboratively and consult about the study. Through the audio tool they can talk together while the controller works with the remote application. Any user who needs to work with the study can request application control.

Several utility components are responsible for handling different messages coming from the presentation tier. Utility components are categorised into three groups:

1. Security components

2. Management components

3. Interaction components

The details of these components are presented in chapter 4. Generally, the messages that are sent from the presentation tier to the processing tier fall into two different categories:

1. User Management Messages: these messages include messages related to user management affairs (e.g., application control request, leader's acceptance or rejection reply, user list updates, user role updates, processing tier status updates, etc.). The processing tier management components process these messages.

2. Event Messages: These messages are relevant to the events caused by the controlling user working with the remote application. Keyboard and mouse events are of this class. The presentation tier is equipped with a module that captures mouse and keyboard events on the controller's application window. Event messages are processed by the interaction components in the processing tier.

The manipulation and consultation workflow includes the following steps (Figure 3.25):

1. The presentation tier sends a message to the processing tier. The message can be of either type mentioned above (i.e., User Management or Event).

2. When the message is received by the processing tier, the type of the message will be investigated. If the message type is user management it is forwarded to the management component; otherwise the system goes to step 5.

3. The management component processes the message and, if necessary, the management component generates a reply message.

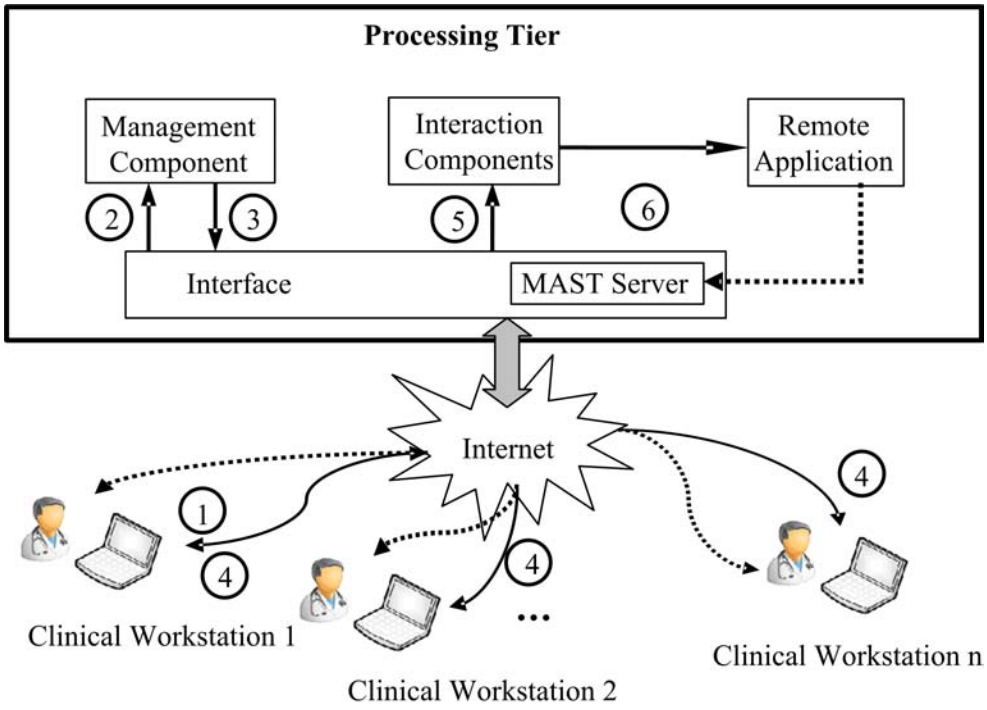


Figure 3.25: Workflow of the image manipulation and consultation phase.

4. Based on the type of the message, the reply is sent to a specific user or a group of users. For instance, when a user requests the application control, a user role update message is sent to all users.

5. The event message received by the processing tier is forwarded to the interaction components.

6. The interaction component in the processing tier raises the same event (e.g., mouse move) on the application window running on the processing tier.

Note that at all times in this phase, the MAST server streams the application window to the MAST clients so all users can view all changes (it is depicted by dotted arrows in Figure 3.25). Also sometimes one message causes a series of actions and subsequent messages among system modules. For instance when a user logs out,

not only the management components remove the user from the list, but also an updated user list is sent to all the other users. At the end of this phase, users log out or simply close their client GUI on their machines. The session is active as long as the leader is in the session. When the leader logs out, the session is closed and all other users are logged out (if they are still in the session).

3.6 Summary

This chapter explained the details of the proposed approach. The three-tier architecture, developed components at each tier, structure of the interfaces, and workflow of the system were elaborated. The steps in each phase of the workflow were described and the GUI windows of the system were introduced.

The next chapter gives the details of implementation of the components designed for the system

Chapter 4

Implementation

Several components and classes have been implemented to transform an existing medical imaging application into a collaborative, PACS-based telemedical system. This chapter explains the details of the implementation. Most of the components were developed in Java; however, some components were written in C++ to provide a higher speed. Eclipse [85] SDK has been used as a platform for developing Java code.

In addition to the components that have been implemented, some open-source, off-the-shelf components have been used. Some of these components (e.g., third party C++ classes) have been amended for usage in this architecture. Table 4.1 shows the total number of classes that are developed or amended (i.e., originally developed by a third party) in the system and their programming language.

The class diagram of the developed Java classes is presented in Appendix A.Several classes usually collaborate to provide a function for the system. These classes form a component. Based on the three-tier architecture, the components are categorized

Table 4.1: Number of implemented classes

	Java	**C++**	**Total**
Developed from scratch	77	0	77
Amended	3	6	9
Total (Developed and Amended)	80	6	86

in three groups: data tier components, processing tier components, and presentation tier components. The communications between data and processing tiers are based on the DICOM protocol. However, the communication between the processing and presentation tiers is carried out by some messages developed for this architecture.

Section 4.1 lists the different messages used in the architecture and elaborates the structure of the messages. The details of the components are explained in section 4.2. Section 4.3 explains the specifications of the testing platform, where all developed and amended components have been tested. Finally, the networking support for the system is described in section 4.4.

4.1 Messages

Various types of messages perform the communication between the processing and presentation tiers. The messages are responsible for various tasks; however, they can be generally categorized in three groups:

1. Control Messages: This group carries controlling messages between presentation and processing tiers.

2. Control Response Messages: These messages carry responses to the controlling messages.

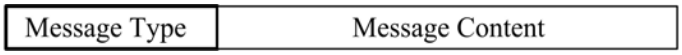


Figure 4.1: Message Structure.

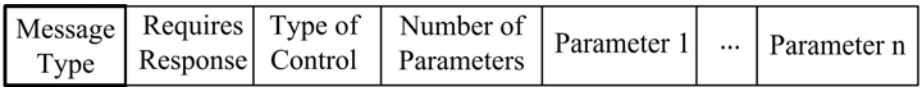


Figure 4.2: Structure of the control messages.

3. Event Messages: This group carries mouse and keyboard events from the presentation tier to the processing tier.

All messages have the same general structure and they implement the Message interface class. Figure 4.1 illustrates the general structure of messages.

All messages include two parts: Message type and content. The message type can be Control (C), Control Response (CR), or Event (E). The message contents are different and depend on the message type. Subsections 4.1.1, 4.1.2 and 4.1.3 explain C, CR, and E types respectively and list all messages of each type.

4.1.1 Control Messages

Control messages carry control commands between the presentation and processing tiers. Figure 4.2 depicts the structure of the control messages.

The first field is C which stands for Control. The second field states if the message needs a response. The third field specifies the type of the control message shown in Table 4.2. The forth field indicates the number of parameters followed by the parameters of the control message.

Table 4.2: Types of the control messages

Control Name	**Description**
LoginCtrl	Used for login information such as username and password.
LogoutCtrl	Used for logout and include the information that states who had logged out.
LogoutForcedCtrl	Used for the workflow when the processing tier forces users to logout (i.e., when the leader logs out).
TakeControlRequestCtrl	Any user can request for application control. This message carries the information of such requests.
GiveControlToRequesterCtrl	The leader may accept or reject the requests of users who want to take the application control. This message carries the information of this operation.
LeaderTakesControlCtrl	The leader may want to takes the application control. This message is used for this purpose.
CloseMessageCtrl	This message is used when one tier wants to close its connection with the other tier.
UpdateUserListCtrl	The updated list of users (as they logs in or out) is sent by this message to all connected users.
SearchPatientCtrl	When a user searches for a patient, the search fields are encapsulated in this message.
FetchStudyAndOpenApplicationCtrl	The message triggers the processing tier to fetch DICOM data, change their format into the applications format and finally open the application on the processing tier and stream the application window to the users.
UpdateProcessingCenterStatusCtrl	This message carries the updated status of the processing tier.
CloseSessionInProcessingCenterCtrl	The message closes the application and the session.
UpdateControllingStatusCtrl	The control of the application might be transferred during a session. This message updates all users of such a change.

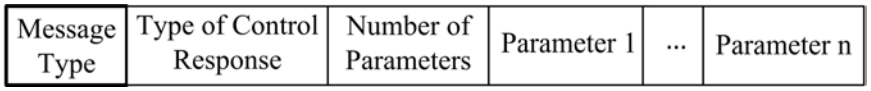


Figure 4.3: Structure of the control response messages.

Table 4.3: Types of the control response messages

Control Name	**Description**
LoginCtrlRsp	The message shows if the user is logged in successfully, if so what is the users ID. It also includes the leaders ID and application controllers ID.
LogoutCtrlRsp	Message acknowledges that the user could log out successfully.
SearchPatientCtrlRsp	The message returns the information found in the data tier about the searched study. If more than one study is found (e.g., there are two or three studies that match the searched fields), all will be carried by this message.

4.1.2 Control Response Messages

Some control messages require a reply. The reply is encapsulated in a message called control response message. Figure 4.3 shows the structure of these messages.

The first field is RC which stands for Response to Control. There are three types of control response messages shown in Table 4.3. The third field states the number of parameters in the message followed by the parameters.

4.1.3 Event Messages

The controller of the remote medical application must be able to steer the application. In order to provide such capability, the system should capture all keyboard and mouse events occurring in the application window at the controller's machine. Event messages are responsible for carrying all these events. Figure 4.4 shows the structure of the event messages.

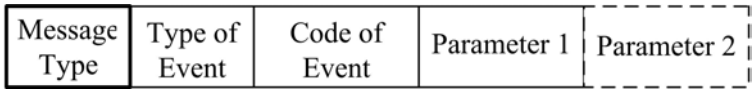


Figure 4.4: Structure of the event messages.

Table 4.4: Event codes

Code of Event	**Description**
MouseLeftButtonPress	Occurs when the user presses mouse left button.
MouseLeftButtonRelease	Occurs when the user releases mouse left button.
MouseRightButtonPress	Occurs when the user presses mouse right button.
MousRightButtonRelease	Occurs when the user releases mouse right button.
MouseMiddleButtonPress	Occurs when the user presses mouse middle button.
MouseMiddleButtonRelease	Occurs when the user releases mouse middle button.
MouseMove	Occurs when the user moves mouse.
MouseWheel	Occurs when the user scrolls mouse wheel.
KeyPress	Occurs when the user presses a keyboard key.
KeyRelease	Occurs when the user releases a keyboard key.

The first field here is E that stands for Event message. There are two types of events which are carried in the second field of the event messages:

- Mouse events
- Keyboard events

The third field is the event code that shows if the action is caused by the mouse or the keyboard. The list of event codes is shown in Table 4.4.

Depending on the event type, the message has one or two parameters. If the type of the message is Mouse event, it carries two parameters showing the mouse coordinates. If the message is Keyboard event, it includes only one parameter that shows which key is pressed or released.

4.2 Components

A brief list of the implemented components follows. In this list, the components used without any modification are labeled [No Change], those that have been amended are labeled [Amended]. The rest were developed completely. Figure 4.5 depicts these components and their location:

1. Data Tier

   (a) DICOM server [No Change]: PACS-based systems need to connect to a PACS server where medical images are stored. DCM4CHEE [86] has been used as the PACS server. DCM4CHEE is an open-source, Java-based, clinical data management system.

   (b) Interface to processing tier: The interface is connected to the PACS servers on one side and to the other corresponding interface on the other side. It uses the DICOM protocol and parallel TCP connections to exchange control messages and data.

2. Processing Tier

   (a) Interface to data tier: The interface is connected to the corresponding interface on the data tier. It uses the DICOM protocol and parallel TCP connections to exchange control messages and fetch data to the processing tier.

   (b) Security components: They include classes responsible for providing different security services such as authentication and encryption/decryption.

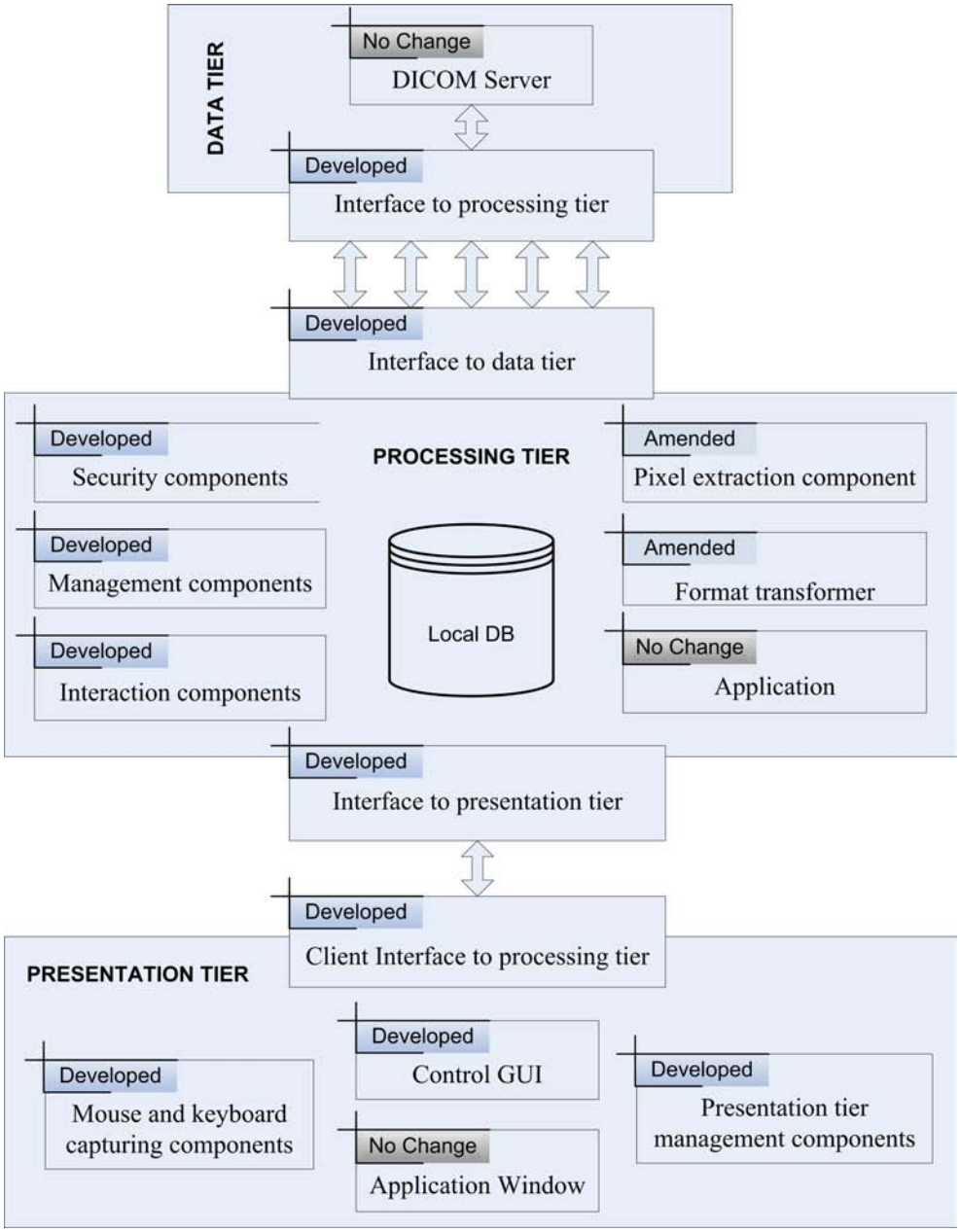


Figure 4.5: The components and their location.

(c) Management components: They include classes responsible for managing data and control flows, sessions, and exception handling.

(d) Interaction components: They include classes responsible for providing in-

teraction between end users. The classes interpret the mouse and keyboard commands from the end users.

(e) Pixel extraction component [Amended]: Include classes responsible for extracting pixel information and sending them to the clients. MAST [80] has been used for this purpose.

(f) Applications [No Change]: The application that sits in the processing tier. The application remains intact while the provided components enhance it with collaborative and remote accessibility features. The application can also work with PACS servers. ParaView [83] and Slicer3 [84] are used as the medical imaging applications in the processing tier.

(g) Format transformer [Amended]: Real medical data is in DICOM format while ParaView and Slicer3 use VTK format. As mentioned in section 3.3 this thesis uses a preprocessing component. This component transforms DICOM images to VTK format. One of the classes of gdcm [87] was amended for this purpose. Subsection 4.2.7 provides more details.

(h) Interface to presentation tier: It receives messages from the end users. The interface interprets the messages and forwards them to appropriate components. The interface also includes network classes responsible for the communication between the processing tier and presentation tier.

(i) Local Database: It stores the users information. MySQL [88] has been used as the local database.

3. Presentation Tier

(a) Client interface to processing tier: It includes network classes responsible for sending mouse and keyboard events and control messages to the processing tier and receiving control commands.

(b) Control GUI: it shows the applications status, a list of users and their roles. It also provides a graphical interface for managing sessions.

(c) Application Window [No Change]: MAST client shows the same window that is captured in the processing tier. MAST application has been used as the MAST client in the presentation tier without change.

(d) Mouse and keyboard capturing components: As described in section 3.4, MAST has been enhanced to support collaboration among users. Several components have been developed to capture the keyboard and mouse commands. The details are given in subsection 4.2.11.

(e) Presentation tier management components: they are responsible for managing sessions as well as data and control flows.

Note that components 2b to 2e are called utility components in the previous chapter (Figure 3.3). The rest of this section explains the implementation of the components that were developed from scratch or amended to work in the system.

4.2.1 Interface to Processing Tier

To enhance data communication between the PACS servers and the processing tier, a pair of interfaces has been developed. These interfaces use the DICOM protocol and parallel TCP connections to exchange data and message. Dcm4che2 [89] library has been used and the new components are built on top of this library.

4.2.2 Interface to Data Tier

The interface to data tier is the corresponding part of the other interface laid in the presentation tier. This pair of interfaces use parallel TCP connections for storage SOPs. The structure of this component is the same as its counterpart in the data tier.

4.2.3 Security Components

Security components include three classes: Authentication, CryptoLibrary and User. These classes are responsible for providing security features to the system.

Authentication class is responsible for checking if a user with specified user name and password is a valid user in the system. This is carried out through Authenticate() method in this class. The method connects to a database and does the query.

CryptoLibrary provides two methods encrypt() and decrypt(). All messages are encrypted before being sent and are decrypted after being received; as a result, the security is enhanced.

The User class provides the basic user information, including user name, a unique ID, and a flag that shows if the user is controlling the remote application and/or leading the session.

4.2.4 Management Components

These components are responsible for managing data and control flows, sessions, and handling errors and exceptions in the processing tier. One of the main classes of this category is ServerSessionManagement. All control messages received by the

processing tier are processed in this class. It includes methods for:

- Adding/removing users to/from sessions
- Reporting status of the data and processing tiers to end users
- Reporting errors to end users and the session leader
- Managing users requests for controlling the remote application
- Taking/granting control of the remote application from/to a user
- Updating the list of users
- Sending appropriate messages to the processing tier

4.2.5 Interaction Components

The interaction components in the processing tier include classes that simulate mouse and keyboard events. In addition to classes that were developed from scratch, Robot [90] and X.java [91] classes were used as interaction components. Robot class raises mouse and keyboard events on the screen. Class InputEventGenerator uses an instance of Robot and raises events listed in Table 4.4. Robot class uses Abstract Window Toolkit (AWT) [92] to distinguish the keyboard keys. AWT is Java's user-interface widget toolkit to provide GUI for Java programs.

X class is used for working with X window systems (i.e., the underlying GUI environment in Linux) by Java. This class was used to find the running application window and get the windows handle. The handle is passed to the MAST server, so the window will be streamed and all users can view it.

4.2.6 Pixel Extraction Component

MAST [80] has been used in the processing tier to extract pixel information and stream the application window to the users. However, MAST code has been amended to support systems requirements. This part of MAST is located in the processing tier and called the MAST server because it streams the application window to the users. The main amendments to the MAST code are as follows:

1. The MAST GUI is removed and it runs in the background.

2. The code gets a window handle as an input parameter and it starts streaming the application window of the given handle.

3. The original code is developed for 32-bit Linux versions. The code has been updated to run properly on 64-bit Linux versions. This amendment is done because the processing tier is designed and developed on a 64-bit Linux sever.

4.2.7 Format Transformer Component

Real medical data is in DICOM format while ParaView and Slicer3 use VTK format. In addition, the images are stored in the DICOM server slice by slice and not as a 3D volume, so they cannot be used directly for 3D visualization. As a result, there is a pre-processing step that converts the original DICOM images into VTK format. The images need to be integrated to form a single 3D image as well.

Class vtkdgcmViever.cxx in gdcm [87] has been amended for this purpose. Gdcm is an open source C++ library that processes DICOM data. The class has been amended as follows:

1. The original code was developed to show the DICOM images and to provide interaction tools for the users. These features are removed. Instead the code writes the images into one VTK file.

2. The code gets the output VTK file location path as input parameter.

The system architecture can support other applications that do not support VTK format, however an appropriate pre-processing component should be developed.

4.2.8 Interface to Presentation Tier

The presentation and processing tiers communicate with each other through the messages explained in section 4.1. A pair of interfaces in these two tiers provides this communication. The interface residing at the processing tier has a network component to send and receive messages. The interface has an interpreter that distinguishes message type, gets the message parts and forwards the message information to an appropriate component. The interface uses encryption and decryption for communication.

4.2.9 Client Interface to Processing Tier

This part is another part of the pair of interfaces that provides communication between the processing and presentation tiers. To distinguish it from the interface in the data tier that provides connection to processing tier (subsection 4.2.1) the interface is called client interface to processing tier.

Like its counterpart in the processing tier, this interface is equipped with network and security components to send and receive encrypted messages. It also includes

interpreters that distinguish messages type, gets the messages parts and forwards the information to the appropriate component for processing.

There are different components in this tier that provide different control commands (e.g., request control command, etc.). The components send appropriate messages when a user issues commands by the GUI window (e.g., clicks on the Request Control button).

4.2.10 Control GUI

Control GUI (i.e., window) provides the list of the users participating in sessions and supplies a tool for them to issue their commands. The GUI windows show different features for a leader and a common user. Figure 3.16 shows the GUI. User friendliness and simplicity were primary goals in the design of the GUI. Visual Editor [93] and Eclipse [85] have been used to design the GUI.

There are other developed GUI windows for logging in, searching and showing search results (Figures 3.14, 3.19, and 3.20 respectively).

4.2.11 Mouse and Keyboard Capturing Components

These components capture the keyboard and mouse events. The components deliver the events to the interface, which sends them to the processing tier. In addition to the classes developed from scratch, JNative [94] library was used. JNative provides access to native libraries in Windows (i.e., dll libraries) from Java. These libraries provide mouse and keyboard event information.

To capture the events appropriately, the system first checks if the user is the

remote application controller. Then it checks if the remote application window is in foreground. If both conditions are true, the system captures the events.

For mouse events, the system needs another condition. It checks if the mouse is in the window. If the mouse is in the application's window, the relative position of the mouse to the window is calculated (by having the absolute positions of the window and the mouse).

JNative library captures the keyboard events with Virtual Key (VK) coding (the defined key codes in Windows) while the events on the robot class uses AWT coding (see subsection 4.2.5). To convert the VK key codes to AWT codes a class has been developed called KeyboardCoder. Method ConvertVKToAWT() gets a VK key code and returns the corresponding AWT code of that key.

4.2.12 Management Components in Presentation Tier

These components have similar functionalities as the management components in the processing tier. They are responsible for managing control flows in the presentation tier. Management components update control status based on the messages received from the processing tier. The components also keep and update all session information, such as the users list, and the leader and controlling user IDs. When a user logs out, these components make sure that all allocated resources become free and all related programs (i.e., MAST and RAT) are closed.

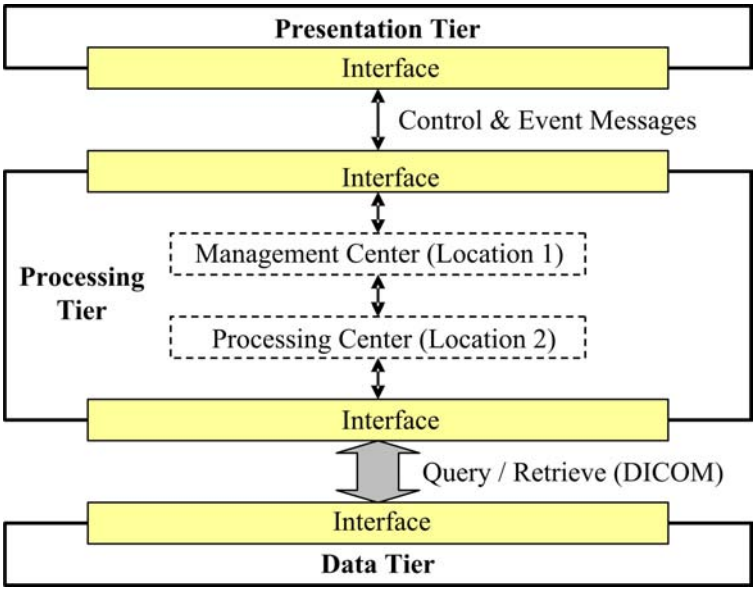


Figure 4.6: Using a processing center in the processing tier.

4.3 Testing Platform Specifications

All components have been tested to make sure that they work properly. In the testing platform, the processing tier runs in two different locations (Figure 4.6). In many cases a powerful processing center is available. This separation is to support such external processing centers.

The first location is called Management Center where the Security (subsection 4.2.3) and Management components (subsection 4.2.4) are laid on. The second location is called Processing Center where the interface to the data tier (subsection 4.2.2), the pre-processing (subsection 4.2.7) component, the remote application, the MAST server (subsection 4.2.6), and the interaction components (subsection 4.2.5) are placed. Here, a cluster of 6 HP Scalable Visualization Array (SVA) computers has been used as the processing center.

For the testing platform the following platforms have been used in each tier:

Table 4.5: Platform-dependent components

Components	**Location**	**Platform**
Interaction components	Processing tier (Processing Center)	64-bit Linux
Pixel extraction component (MAST server)	Processing tier (Processing Center).	64-bit Linux
Format transformer component	Processing tier (Processing Center)	64-bit Linux
Mouse and keyboard capturing components	Presentation Tier	Windows
MAST Client	Presentation Tier	Windows

- Presentation tier: Windows XP/Vista/Seven.
- Processing tier (Management Center): 32-bit Linux, CentOS v.5.2.
- Processing tier (Processing Center): 64-bit Linux, Red Hat Enterprise v.4.
- Data tier: 32-bit Linux, CentOS v.5.2.

Most of the components are platform independent. However, some of them were developed for a specific platform. The platform-dependent components are developed in C++. They work directly with the underlying Operating System (OS). For running the system on a different platform, these specific components should be developed. Table 4.5 lists the platform-dependent components, their location and platform.

4.4 Network Support

As described in section 3.1, the tiers may be placed at different locations and connected through different types of networks. For connecting the tiers, a LAN or

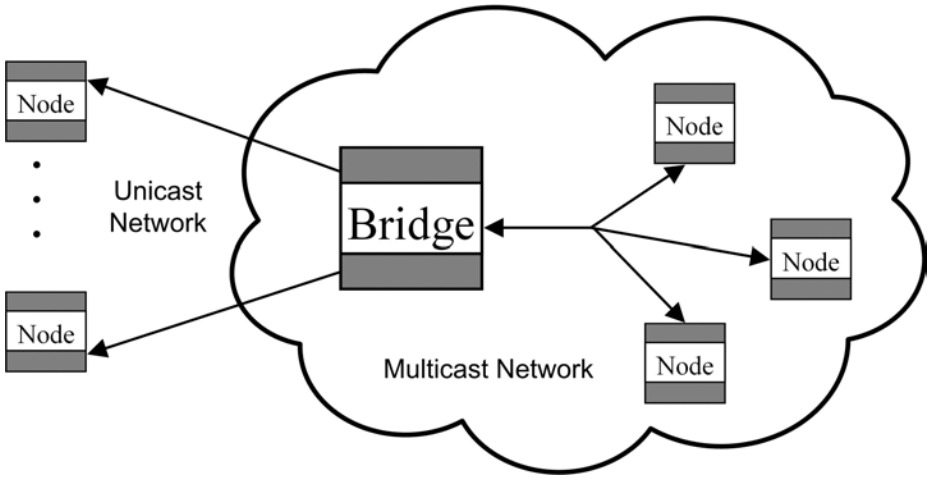


Figure 4.7: Using multicast bridge to support unicast.

WAN network could be used.

The system supports both Multicast and Unicast network technologies. Multicast provides the most efficient way to deliver messages to a group of destinations simultaneously. The basic components used in the system; namely MAST and RAT are designed to work in a multicast-enabled network.

Although multicasting is the most efficient way for connecting a group of users, it is not widely supported in the Internet. However, it is expected to have multicast support on the Internet protocol version 6 (IPv6). As a result, currently a system that uses multicasting cannot be used widely. To address this problem a multicast bridge can be used. These bridges connect unicast nodes to a multicast address (Figure 4.7).

This thesis uses Quick-Bridge [95] for this purpose. The bridge is installed in the processing tier (Management Center) in such a way that users, who use a regular Internet based on IPv4, can connect to the system remotely.

4.5 Summary

This chapter provided the details of the implementation. Java was used as the main programming language to develop the components; however, some components were developed in C++. The communication between tiers is carried out via messages. The structure of the messages are described. The developed components and their location as well as testing platform specifications and the underlying network are described.

The next chapter explains the experiments performed on the system.

Chapter 5

Evaluation

This chapter explains the experiments used for evaluating the methods proposed in this thesis. This evaluation was performed in two parts. In the first part the proposed method for PACS connectivity was assessed. In this part, the proposed method was compared with the common method of medical image transmission. A quantitative approach was used in this part of the evaluation.

The second part of the evaluation was carried out to test the system collaboration and remote accessibility capabilities. In this part, both quantitative and qualitative approaches were used.

5.1 PACS Connectivity Evaluation

As mentioned in section 3.2, this thesis proposes a novel idea [51; 52] based on combination of parallelism, compression methods, and the DICOM protocol to enhance medical data communication.

A pair of interfaces is used to obviate any change in existing medical imaging applications. These interfaces are located between the data and processing tiers (or any two DICOM applications in the general case).

To assess the efficiency of the proposed method, different factors are considered in the procedure of evaluation:

- The modality type by which the image is acquired: CT and MR images were tested in the evaluation. Four datasets were user for the experiments.
- The underlying network type: Local Area Network (LAN) and Wide Area Network (WAN) were considered for the evaluation.
- The effect of compression on the proposed method: Three lossless compression encodings were evaluated and the results were compared with the method that utilizes no compression.

The next subsections explain the experiments performed for the evaluation of PACS connectivity. Subsection 5.1.1 describes the datasets specifications. The evaluation procedure is explained in subsection 5.1.2. The next subsections describe the experiments.

5.1.1 Datasets and Compression Methods

Two CT and two MR image datasets were used for the evaluation. By testing two different datasets for each modality we can have more reliable results and therefore make more broadly based conclusions. Table 5.1 shows the datasets specifications. BREBIX [96] is a dataset which includes CT images of hypernephroma, arterial and

Table 5.1: Datasets specifications

No.	Datasets name	Modality	Number of images	Total volume of uncompressed data (MB)
1	BREBIX	CT	638	320.4
2	CARCINOMIX	CT	437	219.3
3	MR-BODY	MR	218	33.6
4	MR-BRAIN	MR	140	61.1
Total			**1433**	**634.4**

venous acquisitions. CARCINOMIX [96] includes lung carcinoma CT images. MR-BODY [97] collection includes MR images for different body parts including the brain, abdomen, heart, knee, shoulder, and spine. MR-BRAIN [98] is a brain MR images dataset.

Besides the default uncompressed encoding, three lossless encodings were used for the experiments. In DICOM terminology, the data encoding is represented by the Transfer Syntax term. The chosen compressed Transfer Syntaxes are lossless because the final images should be the same as the original images to compare the proposed method with the default method of communication (i.e. the default uncompressed Transfer Syntax without parallelism). The chosen Transfer Syntaxes properties are shown in Table 5.2.

5.1.2 Evaluation Procedure

For each dataset several experiments were conducted. The general procedure of the experiments includes the following steps:

1. In the first step, the three compression methods were compared. The comparison was performed by assessing compression ratio and speed of the methods.

Table 5.2: Transfer Syntaxes properties

No.	Transfer Syntax Name	Transfer Syntax Unique ID
0	Implicit VR Little Endian (Default)	1.2.840.10008.1.2
1	JPEG Lossless, Non-Hierarchical (Process 14)	1.2.840.10008.1.2.4.57
2	JPEG Lossless, Non-Hierarchical, First-Order Prediction (Process 14 [Selection Value 1])	1.2.840.10008.1.2.4.70
3	JPEG-LS Lossless Image Compression	1.2.840.10008.1.2.4.80

2. The dataset images were sent by one of the following methods and compared in terms of data transmission time:

   I. Default: the images were sent by one single connection and Transfer Syntax 0 was used (i.e., no compression and no parallelism).

   II. Pure Parallelism: the images were sent by parallel connections and Transfer Syntax 0 was used (i.e., using parallelism but no compression).

   III. Combination of Parallelism and Transfer Syntax 1: the images were sent by parallel connections and Transfer Syntax 1 was used as the compression method.

   IV. Combination of Parallelism and Transfer Syntax 2: the images were sent by parallel connections and Transfer Syntax 2 was used as the compression method.

   V. Combination of Parallelism and Transfer Syntax 3: the images were sent by parallel connections and Transfer Syntax 3 was used as the compression method.

In the second step above, the experiments were performed on both LAN and WAN networks and the efficiency of the methods based on the network type was studied. To evaluate the effect of parallelism, 2 to 30 parallel connections were utilized. The experiments were repeated 20 times on WAN and 70 times on LAN. Speedup of each method was reported. The speedup was calculated by dividing the average transmission time of each method (i.e., methods II, III, IV and V) by the average time of the default method (i.e., method I).

A Pentium II with Quad-Core CPU and 2 GB of RAM running Linux CentOS was used as the receiver for both LAN and WAN experiments.

The LAN experiments were carried out on a 100Mbps Ethernet network. A Pentium IV computer with Dual-Core CPU and 1 GB of RAM running Windows XP was used as the sender.

For WAN experiments, a Pentium IV computer with Dual-Core CPU, 2 GB of RAM and Windows XP were used as the sender. The sender was located in Vienna University of Technology in Austria while the receiver was located in Telecommunication Research Labs in Winnipeg, Canada.

Subsection 5.1.3 explains the comparison of the compression methods. Then Subsections 5.1.4 and 5.1.5 show the experimental results over LAN and WAN respectively. Finally in subsection 5.1.6 the results obtained are analyzed and discussed.

5.1.3 Compression Methods

This section compares the three compression methods by measuring compression ratio and speed for the four datasets. This experiment helps to study which factor

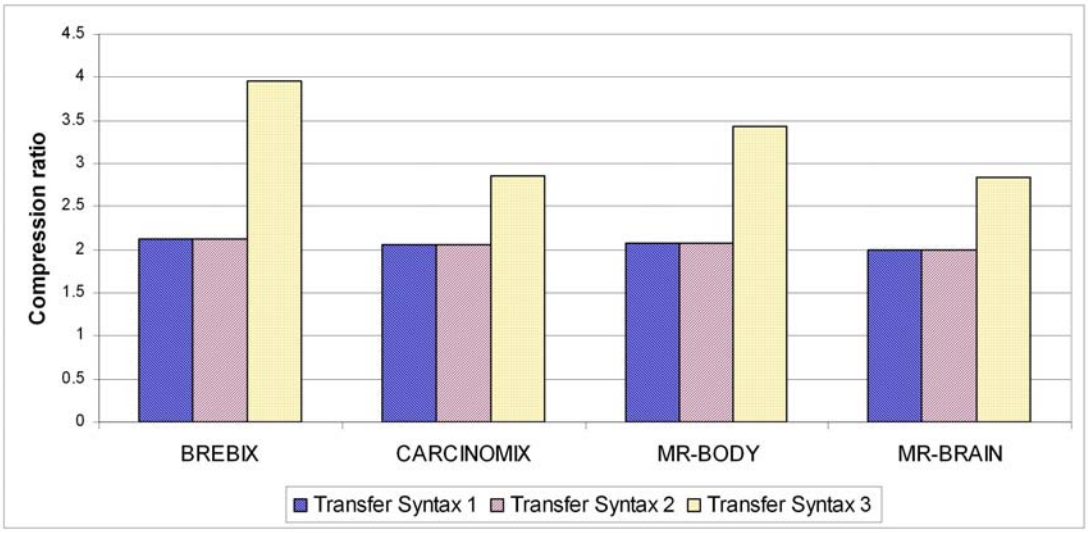


Figure 5.1: Comparison of compression ratio.

(compression ratio or compression speed) plays a more important role in the final transmission time (it is discussed in subsection 5.1.6). The computer used for this experiment was running Windows XP equipped with a Pentium 4 Dual-Core CPU and 1 GB of RAM.

Compression ratio is calculated by the ratio of volume of uncompressed data to compressed data. Figure 5.1 shows data compression ratio of the three compressing Transfer Syntaxes.

In all datasets, Transfer Syntax 3 provides the best compression ratio. Transfer Syntaxes 1 and 2 have a similar compression ratio and rank after Transfer Syntax 3.

The average compression time with 95% confidence interval for each Transfer Syntax is depicted in Figure 5.2. In all datasets, Transfer Syntaxes 1 and 2 are in the same order in terms of speed and faster than Transfer Syntax 3.

These two experiments (i.e., compression ratio and time) suggest that the fastest way of compression is using Transfer Syntaxes 1 or 2, while obtaining the best com-

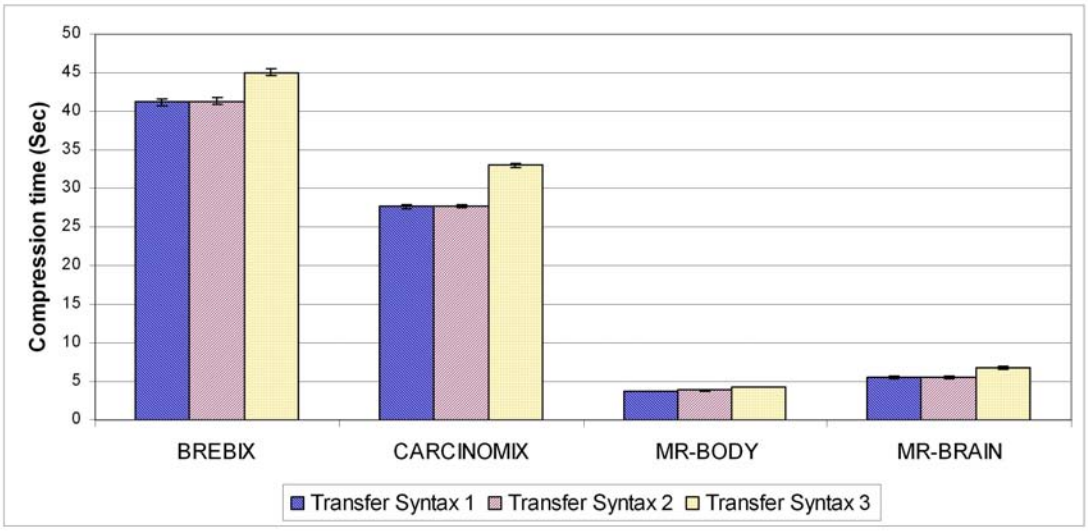


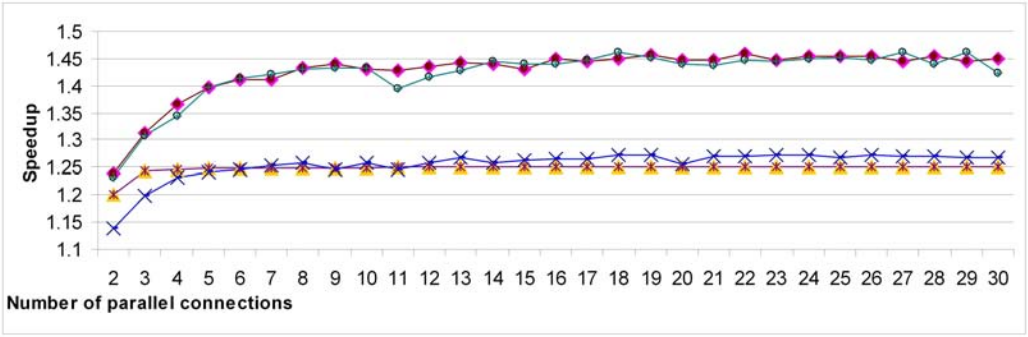
Figure 5.2: Average compression time of the Transfer Syntaxes with 95% confidence interval.

pression ratio can be achieved by using Transfer Syntax 3. Speed and compression ratio can play an important role in transmission time based on the type of the network.

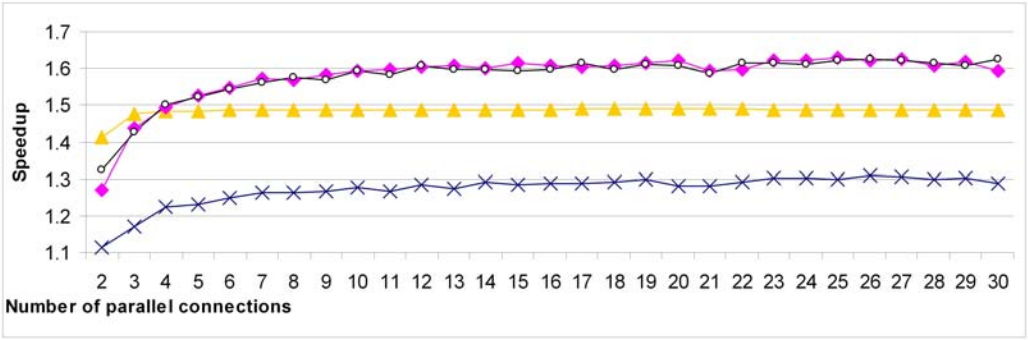
5.1.4 Experiments over LAN

In this experiment the datasets were sent on a LAN network with the five transmission methods mentioned in subsection 5.1.2. Figure 5.3 illustrates the speedup of each method based on the number of parallel connections for the four datasets. Slight improvements were obtained on LAN for CT images (about 1.45 to 1.63). The proposed method achieved a relatively high speedup of about 2.21 to 3.47 times for MR images.

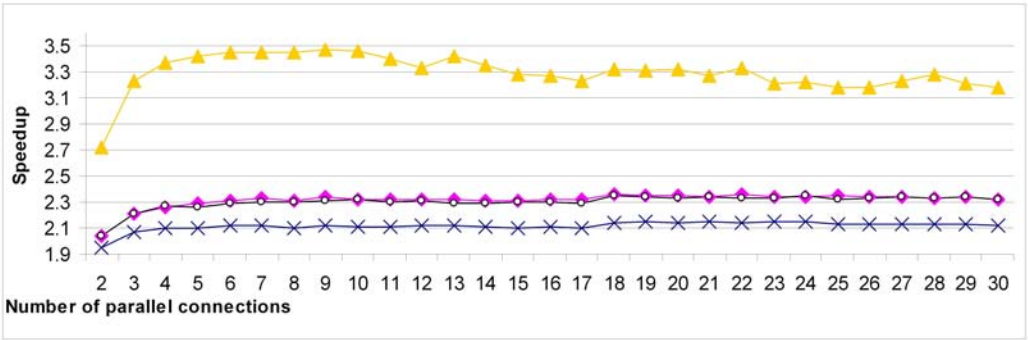
Figure 5.3(a) illustrates the speedup for the BREBIX dataset. All four parallel-based methods outperform method I. Methods III and IV have the best performance.



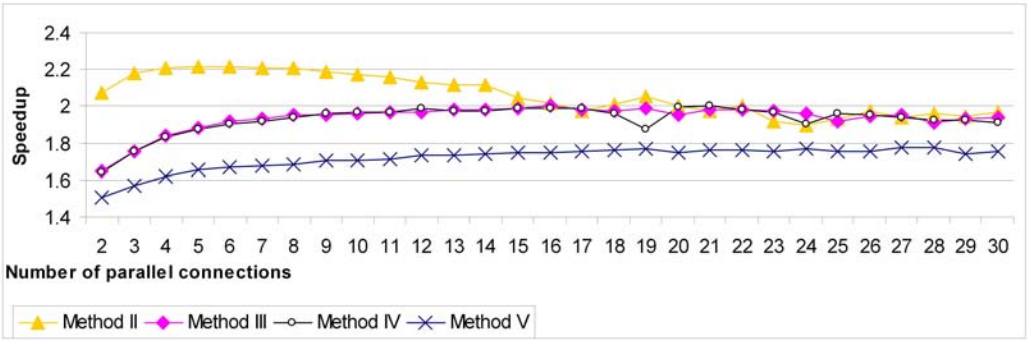
(a) Speedup for BREBIX



(b) Speedup for CARCINOMIX



(c) Speedup for MR-BODY



(d) Speedup for MR-BRAIN

Figure 5.3: Speedup of methods II, III, IV, and V over LAN for 4 datasets.

They can speed up the transmission time by about 1.40 to 1.46 by utilizing six or more parallel connections. Methods II and V improve the transmission time by about 1.23 to 1.27 by employing more than three parallel connections. All four methods reach a maximum speed up after a few levels of parallelism and after that the speedup remains constant or changes very slightly.

For the CARCINOMIX dataset (Figure 5.3(b)), all four parallel-based methods perform similar to the BREBIX dataset case. However, the speedup is higher for the CARCINOMIX dataset and method II has a better speedup than method V. Here, methods III and IV can achieve a speedup of about 1.52 to 1.62 when five or more parallel connections are utilized. Method II improves the transmission time by about 1.47 to 1.48 when it employs more than two parallel connections. Finally, method V can improve the transmission time by 1.23 to 1.30 when it uses five or more parallel connections.

For the MR-BODY dataset (Figure 5.3(c)), all four parallel-based methods improve the transmission time; however, the speedup achieved over LAN is much better in comparison with the BREBIX and CARCINOMIX datasets. Here, method II has the best speedup, improving the transmission time about 3.2 to 3.5 times by using three or more parallel connections. Methods III and IV rank second. They achieve a speedup of about 2.2 to 2.4 times by using three or more parallel connections. Finally, method V can improve the transmission time by about 2.0 to 2.1 times with three or more parallel connections.

For the MR-BRAIN dataset (Figure 5.3(d)), transmission times are improved too. Similar to the MR-BODY dataset, the methods achieve a higher speedup in

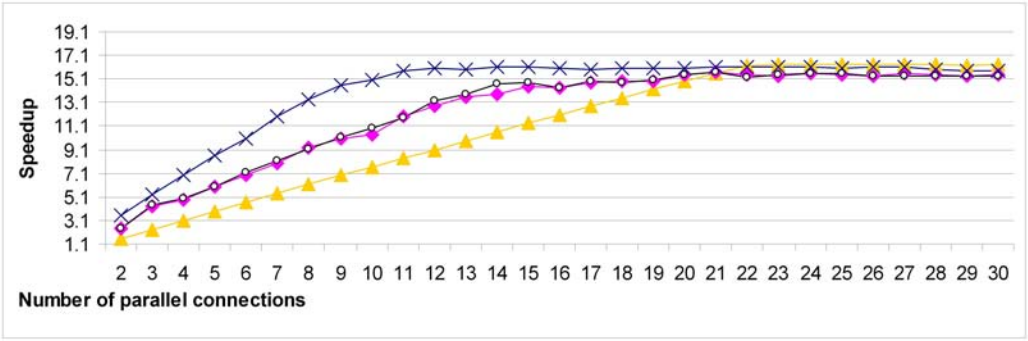
comparison with the BREBIX and CARCINOMIX datasets. Method II has the best speedup, specifically when it uses a few parallel connections. It improves the transmission time about 2.0 to 2.2 times. Method II reduces the speedup when more parallel connections are used. For instance, when method II uses 15 to 30 parallel connections, the speedup drops to about 1.9 times. Methods III and IV improve the speedup by about 1.8 to 2.0 times with five or more parallel connections. Method V has the lowest speedup by achieving a speedup of about 1.6 to 1.7 times with 5 or more parallel connections. Methods III, IV and V do not improve with more than 15 parallel connections and the transmission time even increases in some cases.

5.1.5 Experiments over WAN

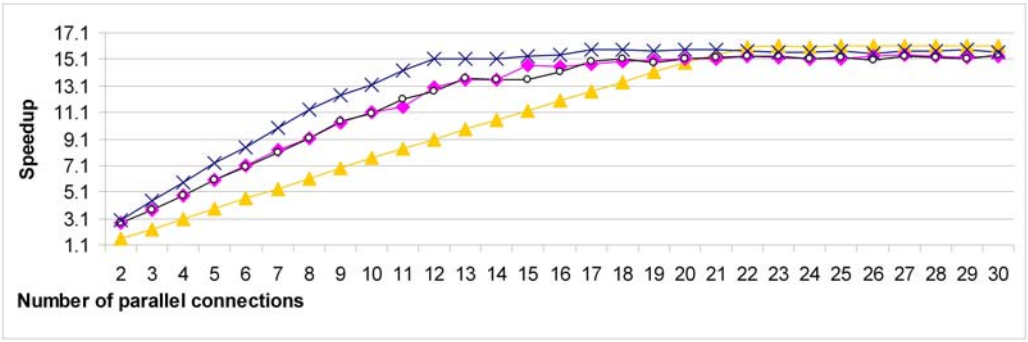
Like the previous experiment on LAN, the five methods mentioned in subsection 5.1.2 were used to transmit the medical images. Figure 5.4 illustrates the speedup of each method based on the number of parallel connections for the four datasets. Here, remarkable improvements were obtained. The speedup is about 15 to 16 times for CT images and about 5 to 13 times for MR images over WAN.

For the BREBIX dataset (Figure 5.4(a)), all four methods improve transmission time by 14 to 16 times when they use more than 20 parallel connections. By using 5 to 20 parallel connections transmission time is improved by 4 to 14 times. Method II outperforms among all methods when it uses 22 or more parallel connections. After that, method V ranks second and methods II and III are the last.

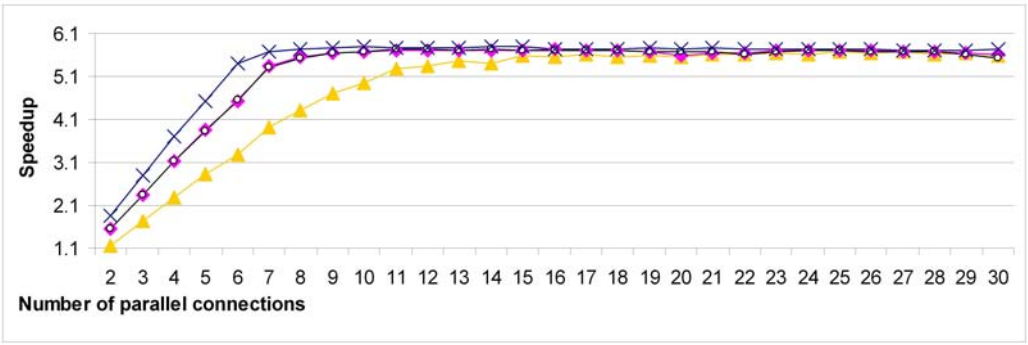
For the CARCINOMIX dataset (Figure 5.4(b)), results are similar to the BREBIX. Speedup is about by 14 to 16 times when 20-30 parallel connections are used.



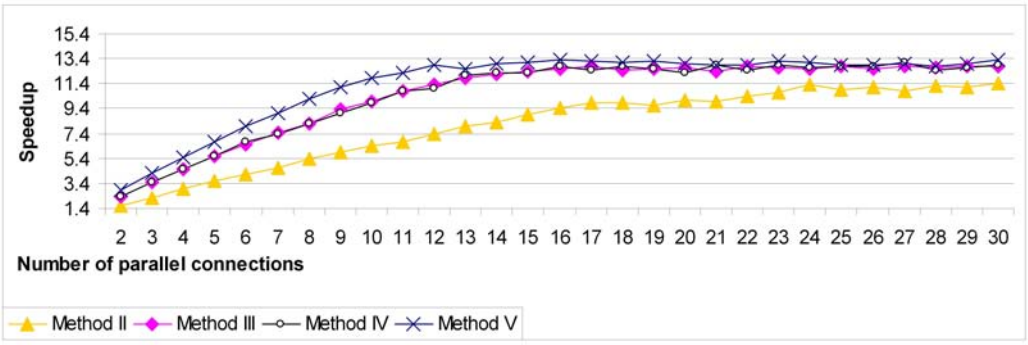
(a) Speedup for BREBIX



(b) Speedup for CARCINOMIX



(c) Speedup for MR-BODY



(d) Speedup for MR-BRAIN

Figure 5.4: Speedup of methods II, III, IV, and V over WAN for 4 datasets.

When 5 to 20 parallel connections are used, the four parallel-based methods achieve 4 to 14 times speedup. Similar to the BREBIX dataset, method II outperforms among all methods. Method V ranks the second best method and methods II and III are last.

For MR-BODY dataset (Figure 5.4(c)), all four methods improve transmission time; however, the speedup is less in comparison with the BREBIX and CARCINOMIX datasets. By exploiting 10 or more parallel connections, the methods can speedup the transmission time about 5.0 to 5.8 times when they use 10 or more parallel connections. If they use 5 to 10 parallel connections then a speedup of about 2.8 to 5.0 times is achieved.

For MR-BRAIN dataset (Figure 5.4(d)), the four parallel-based methods achieve about 10 to 13 times speedup by exploiting 20 or more parallel connections. Using 10 to 15 parallel connections, results in 6 to 10 times speedup. The methods behavior on the MR-BRAIN and MR-BODY datasets is similar; however, the speedup is higher for the MR-BRAIN dataset. Here, method V improves the transmission time slightly better than methods III and V. Like in the MR-BODY dataset, method II has the lowest speedup.

5.1.6 Discussion

In all experiments it was observed that any of the four parallel-based methods improves transmission time. The transmission time improvement is influenced by different factors and can be remarkable. One of the most important factors is the network type. The best improvements are obtained over WAN.

Table 5.3: The methods rank in terms of time overhead and compression ratio.

Encoding	**Time overhead**	**Compression Ratio**	**Method**
Non-Compression	1	3	Method II
Transfer Syntax 1	2	2	Method III
Transfer Syntax 2	2	2	Method IV
Transfer Syntax 3	3	1	Method V

The encoding type plays another important role in transmission time. If we use compression methods, the data size decreases, while overhead is added to the transmission time. Therefore there is a tradeoff for using compression and choosing a proper compression type. The compression type is characterized by two important factors: speed and compression ratio. An ideal compression method has the fastest speed and the highest compression ratio. However, these two factors usually trade off each other: fast compression methods result in lower compression ratio and vise versa.

In the first part of the experiments performed on image datasets, the compression speed and compression ratio were compared for different compression encodings. In all datasets, compression methods performed similarly in speed and ratio.

Table 5.3 ranks the three compression encodings and the non-compressed encoding in terms of minimum overhead time (speed) and maximum compression ratio. These ranks are based on the experimental results performed in subsection 5.1.3. Non-compressed encoding needs no additional operation for transmitting data, so it imposes no overhead. It is also observed in the experimental results that the difference between Transfer Syntaxes 1 and 2 is small thus they are ranked equally. The last column of the table lists the parallel-based method that was used for the encoding.

Since the network type is the most important factor for the parallel-based meth-

Table 5.4: Best method over LAN for each dataset.

Dataset	**Best method**	**No. of parallel connections to achieve the maximum speedup**	**Maximum Speedup**
BREBIX	Method IV	29	1.46
CARCINOMIX	Method III	25	1.63
MR-BODY	Method II	9	3.47
MR-BRAIN	Method II	5	2.21

ods, the performance of the five methods are discussed based on the network type. Table 5.4 shows the maximum speedup for each image dataset, the method that achieves the speedup and the number of parallel connections that the method used to obtain the maximum speedup.

The range of the speedup is different based on modality type (Table 5.4). The transmission time improvement is better for MR datasets (i.e. MR-BODY and MR-BRAIN) on the LAN. In addition the methods need a small number of parallel connections to achieve the maximum speedup for MR images. On the other hand, the speedup for CT datasets is slight on LAN and the methods need a relatively large number of parallel connections to achieve the maximum speedup.

The fastest methods that impose the lowest time overhead (rank 1 and 2 in Table 5.3) are the best methods for LAN. This occurs because of the compression overhead. Since a LAN is a high-speed network, the time overhead of data compression is comparable to the transmission time for each DICOM object; thus, the saved time by sending a smaller piece of data does not compensate for the time overhead imposed by the compression algorithm. As a result, using compression slows down the total transmission time over LAN. Consequently, the methods that impose no or very small overhead are faster over LAN.

Table 5.5: Comparison of network usage over LAN for the best parallel-based method and method I

Dataset	**Maximum Speedup**	**Network Usage by the Best Improving Method**	**Network Usage by Method I**
BREBIX	1.46	0.487	0.708
CARCINOMIX	1.63	0.470	0.592
MR-BODY	3.47	0.790	0.228
MR-BRAIN	2.21	0.829	0.374

The experiments also show that after reaching a certain levels of parallelism, the speedup cannot be improved by adding more parallel connections. This happens because the parallel connection implementation imposes overheads (e.g. Java threads for our implementation) and because of network saturation. All network traffic is bound by the available bandwidth. As a result, increasing the number of parallel connections when the transmission data is bound by the networks available capacity only adds overhead and consequently decreases the speedup.

Table 5.5 shows the network usage for the maximum speedup on each image dataset. For each dataset, the usage is given for the best experimental method and for method I, that is the common method used for medical data transmission.

The methods can only use the available bandwidth because part of the bandwidth is occupied by other applications. The use of parallel connections let the methods take advantage of the available bandwidth, thus speeding up the total transmission. However, the use of compression methods along with parallelism decreases the network usage as well because compression methods reduce the size of images.

The best parallel-based methods for BREBIX and CARCINOMIX (methods III and IV) not only speed up the transmission, but also decrease the network usage. On the other hand, the best parallel-based method for MR-BODY and MR-BRAIN

Table 5.6: Best method over WAN for each dataset.

Dataset	**Best method**	**No. of parallel connections to achieve the maximum speedup**	**Maximum Speedup**
BREBIX	Method II	24	16.34
CARCINOMIX	Method II	23	16.12
MR-BODY	Method V	15	5.78
MR-BRAIN	Method V	16	13.30

is method II. However, this method increases the network usage. The reason for the network usage increment is that method II does not use any compression method and it only uses parallelism for enhancing the transmission time.

Table 5.6 shows the maximum speedup for each dataset on WAN, the method that achieves the best speedup and the number of parallel connections that the respective method uses.

The amount of speedup is remarkable over WAN. Unlike the LAN experiments, the parallel-based methods have a better performance for CT datasets (i.e. BREBIX and CARCINOMIX). The methods need a relatively large number of parallel connections to achieve the maximum speedup.

Over WAN, two completely different methods achieve the best speedup: the method with no compression time overhead (i.e. Method II) and the method with the highest compression ratio (i.e. Method V).

Although Method V imposes the highest compression time overhead, the time that is saved by reducing the data size is remarkable over WAN. As a result, Method V provides a remarkable speedup over WAN and it is the best method for MR images. Even for CT images, it is the second best method and its speedup is comparable to Method II for the datasets used.

Method II outperforms for CT image datasets. However, this outperformance can only be achieved in high levels of parallelism (i.e. 22 parallel connections or more). We may note that this phenomenon happens when the number of DICOM objects is large (e.g., in BREBIX and CARCINOMIX datasets). When the number of DICOM objects is relatively large, in highly parallel communications, the total overlapped transmission time is large and it may even save more time than the time saved by reducing data size.

5.2 Collaboration and Remote Accessibility

This section explains the experiments performed to assess collaboration and remote accessibility of the system. In the first part, the bandwidth required by the MAST server was measured. The experiment also compared the required bandwidth of the MAST server versus TightVNC [77], one of the popular tools for sharing desktop among a group of users. TightVNC uses a pixel extraction technique for streaming the whole desktop.

In the second part of the experiment a group of users were connected to the system and worked with the remote application. In this part both quantitative and qualitative measurements were conducted. Bandwidth and the network delay for the keyboard and mouse events were measured. Here, the assessment of the required bandwidth is more realistic because it is done with a group of people who worked with the remote application. The group of users also answered a questionnaire to provide a qualitative evaluation of the system.

5.2.1 Bandwidth Evaluation

The main objective of this experiment is to evaluate the required bandwidth for streaming a remote application window by the MAST server. This measurement is done for a 3D visualized medical image rotating with different speeds. This experiment also studies if the speed of rotation changes the required bandwidth.

To do this experiment, in the first step a medical image set was 3D visualized. One of the studies of the CEREBRIX [96] dataset was used. CEREBRIX includes images of a brain tumor. Volume rendering was used for visualization. Figure 5.5 shows the visualized image. After that, Python scripting was used to rotate the 3D image. The following paragraphs explain how this rotation is defined in ParaView.

Rotation Simulation

ParaView does not provide any command for rotating objects with a specified speed. So it was simulated by moving the camera around the image. This rotation was defined in a counter-clockwise direction. The positions of objects and cameras are defined in a three dimensional coordinate system. Each camera has several specifications such as Focal Point, Camera View Up, View Angle and Clipping Planes (Figure 5.6).

In a normal view, the positive Z axis faces to the camera. In order to rotate the image in a counter-clockwise direction, the Focal Point was set to the center of the image, the Y axis was held constant and the XZ plane was rotated in a counter-clockwise direction (Figure 5.7).

The rotation of the XZ plane was simulated by changing the position of the camera.

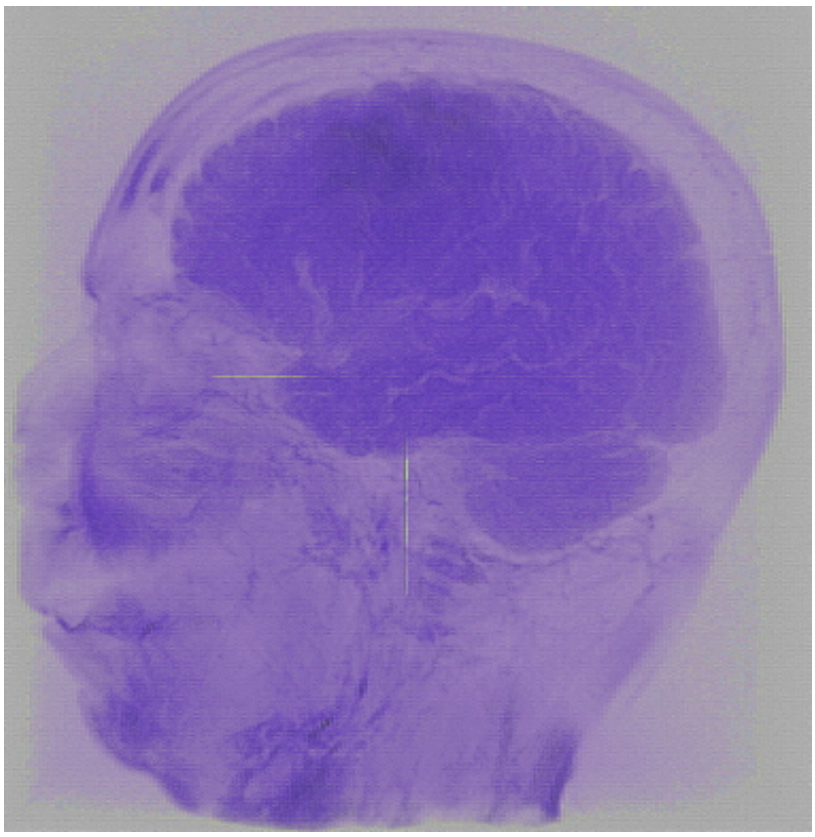


Figure 5.5: 3D visualized image.

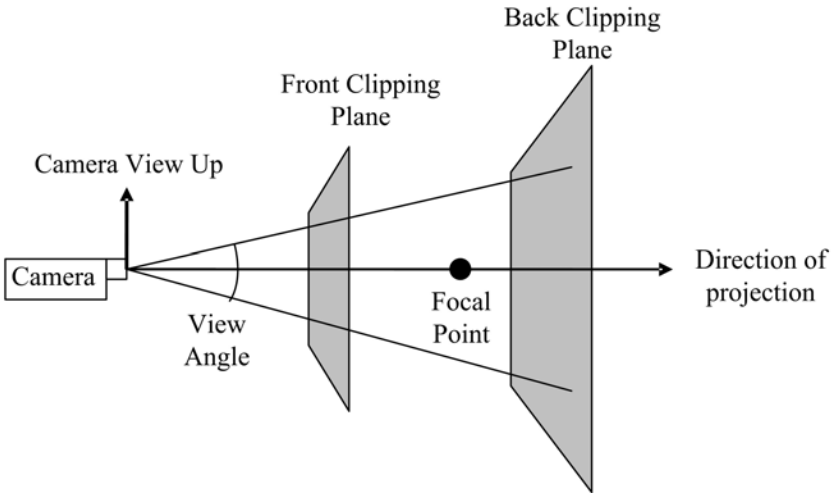


Figure 5.6: Specifications of a camera in ParaView.

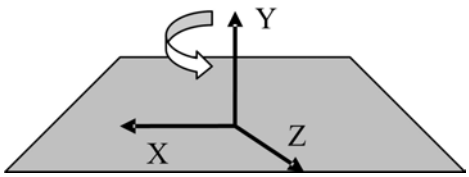


Figure 5.7: Fixed Y axis and rotating XZ plane.

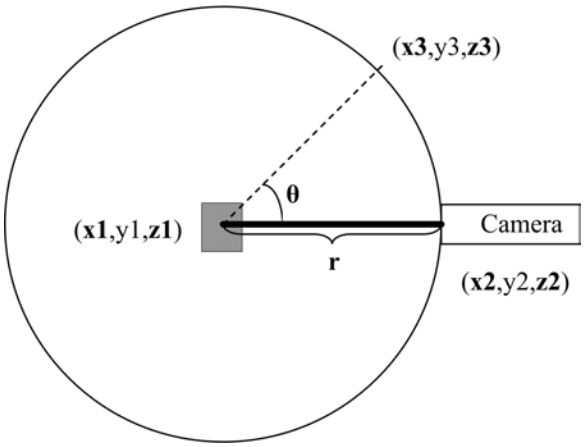


Figure 5.8: The rotation of the camera.

Figure 5.8 illustrates the rotation. Suppose that the camera is moving from (x2,y2,z2) to (x3,y3,z3). Recall that the Y axis is facing the viewer and the camera and the image are both in the same plane (i.e., XZ plane). To have the same distance from the image, the camera should move on a circle in the XZ plane centered by the image (x1,y1,z1).

The following rules hold for the situation as described:

$$y_1 = y_2 = y_3$$

$$r^2 = (x_1 - x_2)^2 + (z_1 - z_2)^2$$

For a given angle θ, the x_3 and z_3 are calculated as:

$$x_3 = x_1 + r \times \cos(\theta)$$

$$z_3 = z_1 + r \times \sin(\theta)$$

Angle θ is relative between the image and the camera. As can be seen in Figure 5.9, based on the first position of the camera and the image an initial angle is defined. For applying the above formulae the initial angle should be considered (to calculate the absolute position of the camera). The initial angle can be worked out by the following formula. However, in this calculation the area of the circle should be considered (Figure 5.9):

$$\text{Initial Angle} = \arccos((z_2 - z_1)/radus)$$

$$Area1 : (z2 - z1) > 0 \;\&\&\; (x2 - x1) < 0$$

$$Area2 : (z2 - z1) < 0 \;\&\&\; (x2 - x1) < 0$$

$$Area3 : (z2 - z1) < 0 \;\&\&\; (x2 - x1) > 0$$

$$Area4 : (z2 - z1) > 0 \;\&\&\; (x2 - x1) > 0$$

After finding the initial angle, the rotation of the image can be simulated by changing the placement of the camera each time by increasing or decreasing θ. The increment of θ causes counter-clockwise rotation, while the decrement results in having clockwise rotation.

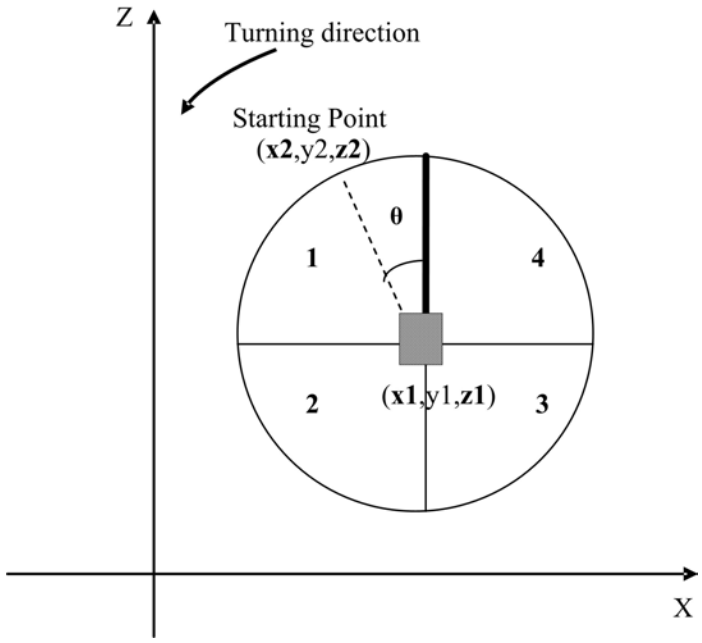


Figure 5.9: Computing the initial angle.

After this step, four animations were made at four rotational speeds: 30, 60, 90, and 120 degrees/second. The animations show the 3D image is rotating at different speeds. These animations were played and the bandwidth consumption of the MAST server and TightVNC were measured. Experimental results are shown later in the section.

To make an animation, the frame rate should be defined. For this experiment, a frame rate of 30 frames per second was defined. That means if we produce 300 frames it takes 10 second to be displayed. Table 5.7 shows the properties of the produced animations.

The first column shows the speed of rotation. The next column is time for a complete rotation which shows the longitude of the move in terms of seconds. It can be calculated by dividing the whole rotation (360 degrees) by the speed (e.g. 30).

Table 5.7: The properties of the animations.

Speed (angle/second)	**Time for a complete rotation (second)**	**No of frames for a rotation**
30	12	360
60	6	180
90	4	120
120	3	90

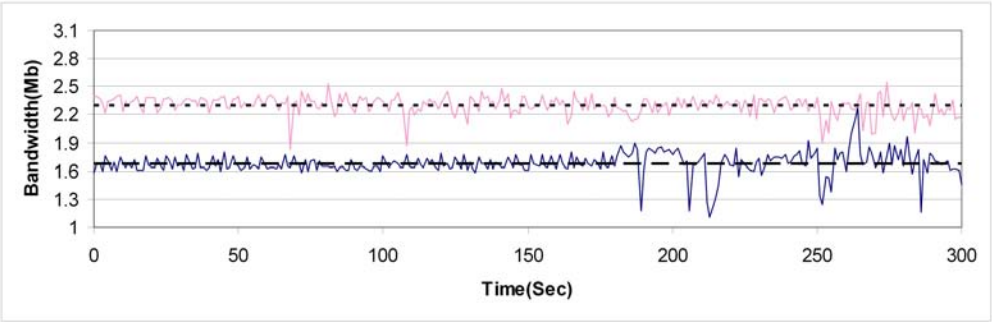
The last column in Table 5.7 shows the number of frames we should make to have a complete rotation at 30 frames/sec rate. It is computed by multiplying the the second column by 30 (frame rate). Appendix B shows the Python script that creates the rotating image at 30 degrees / second speed in ParaView. The information of the first row of Table 5.7 was used in the script.

Required Bandwidth

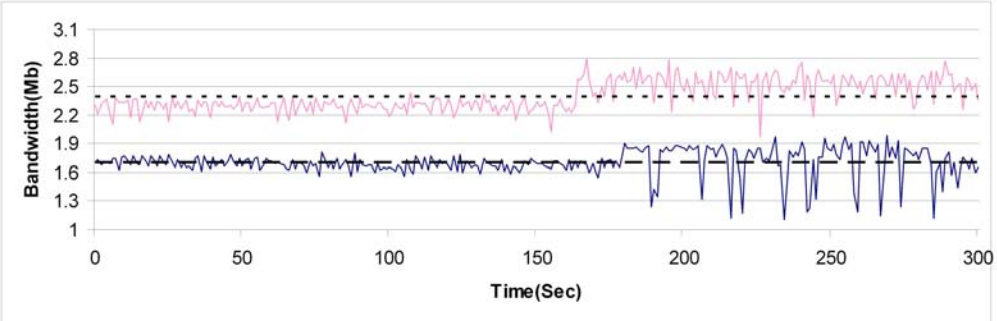
In this step, the four animations created in the previous step were played on a Pentium II computer running Linux CentOS. Then the bandwidths required by MAST and TightVNC for streaming the animations were measured. WireShark [99], a free and open-source network protocol analyzer, was used to measure the bandwidth.

Figure 5.10 compares the required bandwidth of MAST and TightVNC for streaming the four animations.

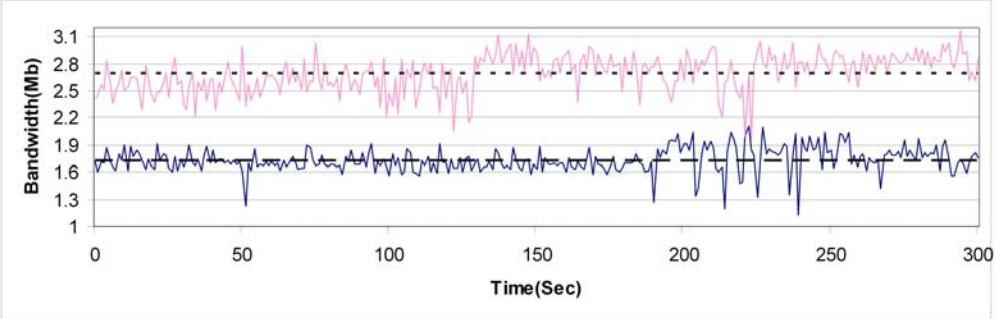
In all cases, MAST requires more bandwidth than TightVNC and therefore TightVNC is superior in terms of bandwidth consumption. The results also show that raising the rotation speed increases the required bandwidth in MAST. The required bandwidth was increased from about 2.3 Mb/sec to 2.7 Mb/sec when the speed increased from 30 deg/sec to 120 deg/sec. However, TightVNC did not utilize remarkably more band-



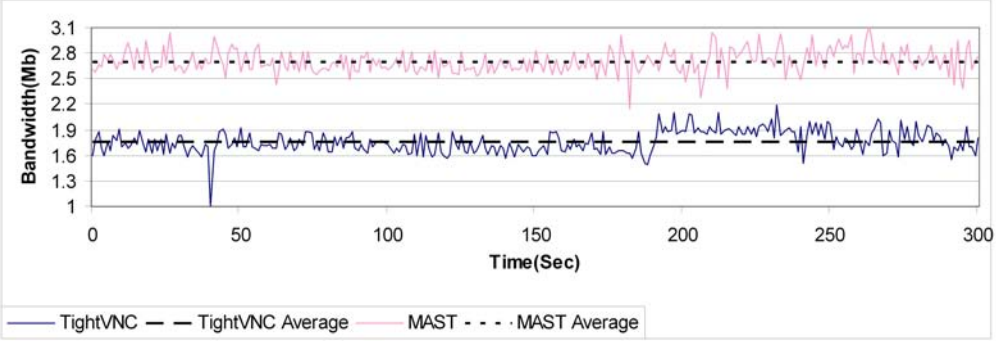
(a) Bandwidth for 30 deg/sec image speed



(b) Bandwidth for 60 deg/sec image speed



(c) Bandwidth for 90 deg/sec image speed



(d) Bandwidth for 120 deg/sec image speed

Figure 5.10: Required bandwidth of MAST and TightVNC for streaming a 3D visualized image rotating with different speeds.

width when the speed of rotation was increased. Thus, TightVNC shows a better performance in this regard as well.

One reason of the superiority of TightVNC over MAST is that MAST is based on an older technology in comparison with TightVNC. The last version of MAST was developed in 2006. Since 2006, more new technologies and compression methods have been developed and TightVNC takes advantage of them. Nonetheless, TightVNC does not support application sharing on Linux-based machines. That is the reason why TightVNC was not used for application sharing in this thesis.

5.2.2 Collaborative Remote Meeting

In this part of the experiments, a group of people was connected to the system simultaneously and worked with the remote application (Figure 5.11). This group included 6 users, 2 at TRLabs Winnipeg, 3 at the University of Manitoba (UM) and one at the University of British Columbia (UBC). Both quantitative and qualitative measurements were performed. The network delay and bandwidth occupation were measured quantitatively. Users were also asked to fill in a qualitative questionnaire. This experiment tried to simulate a real situation in which a group of doctors consult on a study.

Network Delay

In this step, the network delay was measured. There are some network utilities such as *ping* that can be used. However, the delay of encrypting, decrypting and message processing overheads were also measured for this thesis. To have such

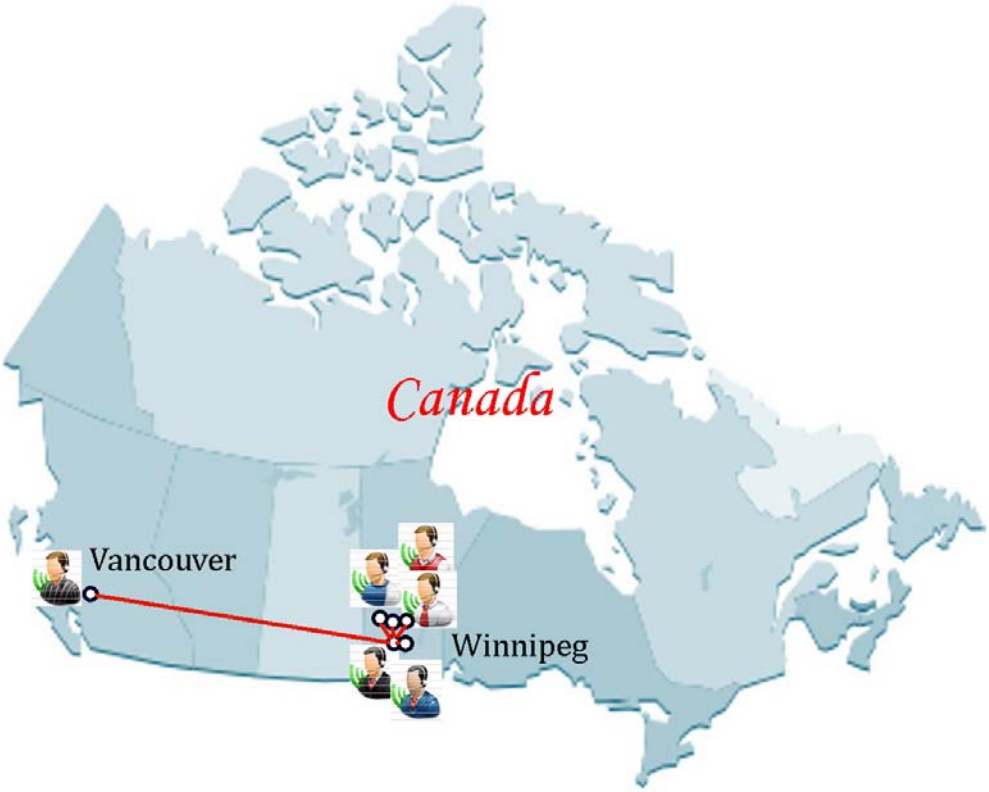


Figure 5.11: Collaborative remote meeting.

measurements, each message was tagged with a sequence number and a time-stamp. When a message was received, approximate delay of the network and other overheads were calculated by the following formulae:

$$\Delta T_{Source} = T_{Source}(i) - T_{Source}(i-1)$$

$$\Delta T_{Destination} = T_{Destination}(i) - T_{Destination}(i-1)$$

$$Delay = \Delta T_{Destination} - \Delta T_{Source}$$

Table 5.8: Approximate delays.

Location	**Average delay (ms)**	**95% confidence interval**
TRLabs	7.68	0.04
UM	9.08	0.08
UBC	28.19	0.83

Where $T_{Source}(i)$ is the time-stamp of message i in the sender and $\Delta\ T_{Source}$ is the time span between two consecutive messages. Similarly, $T_{Destination}(i)$ is the time of receiving message i and $\Delta\ T_{Destination}$ is the time span between two consecutive received messages. If there is no delay, $\Delta\ T_{Source}$ and $\Delta\ T_{Destination}$ should be equal. The difference between them is the delay imposed by the network. Table 5.8 shows the average approximate delay for the three locations.

Required Bandwidth

While the group of users were working with the remote application, the required bandwidth of the system was measured. This measurement is more realistic than what is presented in subsection 5.2.1, because it shows the required bandwidth when users utilized the system in a real scenario. Here, the visualized image was not rotated at a predefined speed (i.e. users rotated the 3D image). Figure 5.12 shows the occupied bandwidth.

The average bandwidth at TRLabs is about 165 Kb/sec, while it is about 380 Kb/sec at UM and about 400 Kb/sec at UBC. It was observed that the required bandwidth is less in TRLabs. The reason is that TRLabs is inside the multicast network, while UM and UBC are outside of the multicast group and the Quick-Bridge [95] software simulates multicasting (section 4.4), which adds additional bandwidth con-

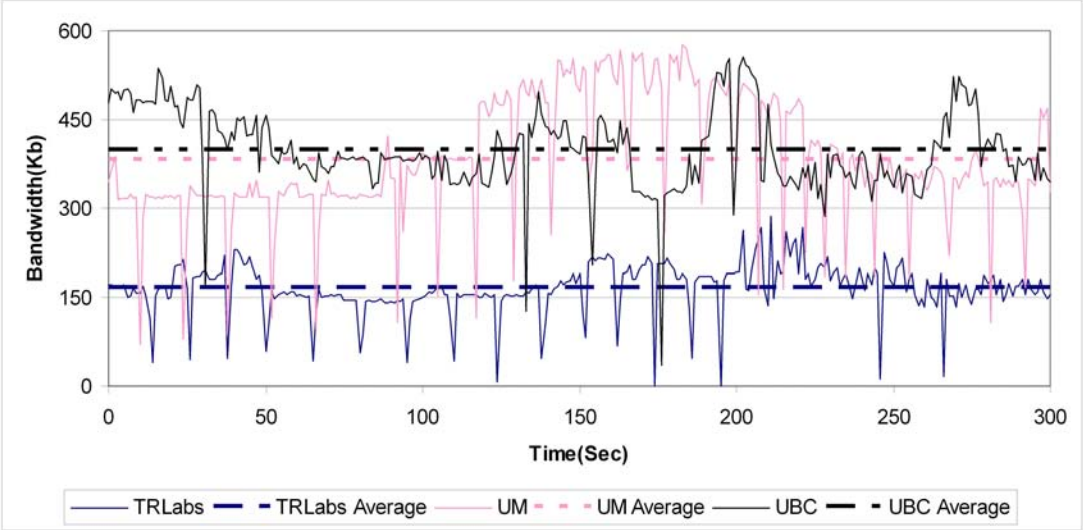


Figure 5.12: Required bandwidth in a collaborative session.

sumption.

It is also obvious that the average required bandwidth in this experiment is much less than the bandwidth requirement measured in section 5.2.1. The main reason is that in reality users do not rotate the image constantly and the speed of rotation is less in comparison with the experiment performed in section 5.2.1. Moreover, the visualization window in this experiment is smaller than the window captured in the previous experiment.

Qualitative Assessment

After working with the remote applications, the users were asked to assess the system qualitatively by answering to the following questions:

1. 3D image quality

2. Convenience of patient search

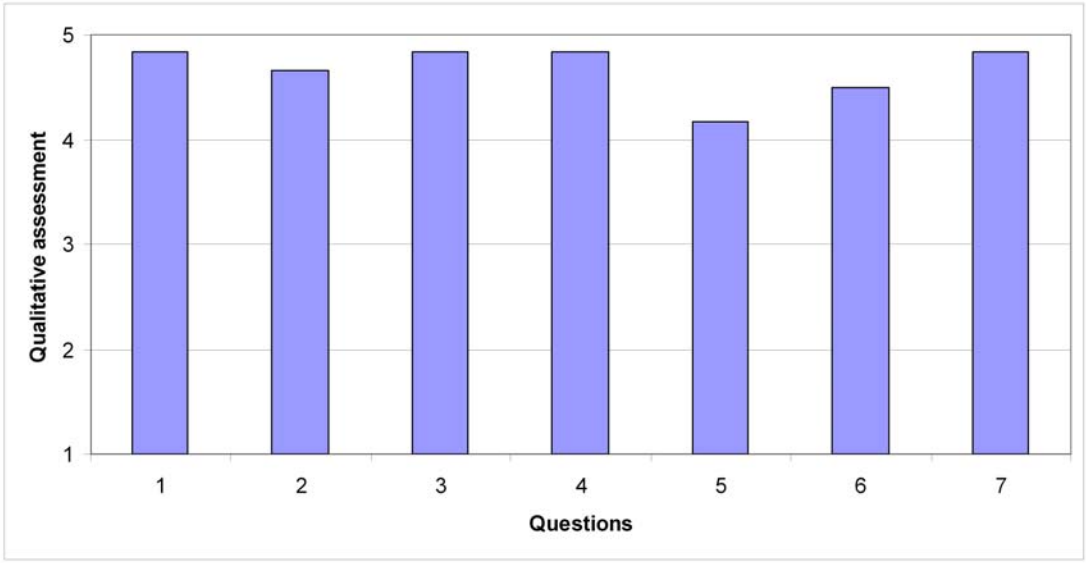


Figure 5.13: Average ratings for qualitative assessments by 6 users.

3. Exchanging the application control among users

4. GUI friendliness (ease of use)

5. Convenience of controlling the application (e.g., Focusing in/out and rotation)

6. Voice quality

7. Overall satisfaction

For each question the users chose one of the five options: Very Good, Good, Normal, Below Normal, and Bad (or equivalently ranked it from 5 to 1). Figure 5.13 shows the result of this qualitative assessment for each question. The average quality is depicted for each question.

In addition, the users were asked to estimate the approximate time for the window to be updated after each action (e.g., rotation). This approximate value shows the window update speed from the users point of view. The users reported an approximate

time value of less than a second at TRLabs and UM and between 1 to 2 seconds at UBC.

5.2.3 Discussion

This section introduced the experiments performed to assess collaboration and remote accessibility of the system. An important factor in remote applications is their required communication bandwidth. For this reason the required bandwidth was measured in two different experiments. The first experiment was performed to compare the streaming component of the system (i.e., MAST) with one of the popular remote desktop applications (i.e., TightVNC). It was observed that TightVNC performed better and required less bandwidth. It was also observed that the MAST component is sensitive to the motion speed in the streamed window.

The second experiment tried to simulate a real scenario, in which a group of doctors (i.e., users) connect to the system and collaborate on a remote medical imaging application.

It was observed that in such a real situation the require bandwidth is much less than what was measured in the first experiment. In a real application, the system requires about 400 Kb/sec bandwidth. This amount of bandwidth makes it possible for the system to be used in many places including universities, research centers, hospitals and doctors offices. Nowadays, many Internet service providers offer services that provide this or higher bandwidths.

In the second experiment two additional factors were measured; namely, network delay and quality of the system from the user's point of view. The delay included not

only the transmission time imposed by the network but also all overheads made by the system (encryption/decryption and message processing). The delay of the system was measured about 30 ms from UBC. It was less than 10 ms from UM and TRLabs. It shows that the overheads of encryption/decryption and message processing are negligible because these values are less than 10 ms.

Finally the users were asked to fill in a questionnaire. The users qualitatively assessed the system with the questionnaire. The assessment results shows that the system quality in the factors taken into account for the evaluation is very good from the user's point of view.

5.3 Summary

This chapter presented the experiments performed to evaluate the methods and components developed in this thesis. In the first step the proposed method for PACS connectivity was assessed. It was shown that the proposed parallel based methods for medical image transmission speeds up the transmission time. On the experiments over LAN, the speedup is slightly better for CT image datasets (about 1.45 to 1.63) and is relatively high for MR image datasets (about 2.21 to 3.47 times). The use of parallelism over WAN improves the transmission time drastically. The speedup is about 15 to 16 times for CT image datasets. However, a lower speedup for MR image datasets was achieved (about 5 times for one dataset and about 11 to 13 times for the others).

The second part of the evaluation was carried out to test the system collaboration and remote accessibility capabilities. Both quantitative and qualitative approaches

were used. It was shown that the system needs about 400 Kb/sec bandwidth for a real scenario. The quality of the system was assessed by the users and the results show a high quality from the users point of view.

The next chapter presents conclusions and discusses future work.

Chapter 6

Conclusion

Many medical imaging applications have been developed so far; however, many of them do not support collaboration, are not integrated into PACS and are not remotely accessible. On the other hand collaboration and remote accessibility are becoming integral parts of new medical applications and PACS connectivity is an essential feature for medical applications in order to be directly used in clinical workflows. As a result, although many medical imaging applications exist and they have provided valuable services, they lack one or more of the mentioned important features and cannot be widely and clinically used.

This thesis introduces a novel approach to transform medical imaging applications into collaborative PACS-based telemedical systems. A three-tier architecture is used in the approach. The existing medical imaging applications are located in the processing tier. Two pairs of interfaces have been used to connect the three tiers together and several utility components have been developed to provide advanced features like collaboration and remote access for the medical imaging applications.

The proposed approach uses a new method for medical image transmission that outperforms the current methods used in medical imaging. The method is based on the DICOM protocol. Current compression-based methods do not improve the transmission time in reasonably high-speed networks. Although these methods decrease the size of data resulting in transmission time savings, the imposed compression time overhead is comparable to the time saved [52]. The main idea behind the proposed method is utilizing parallel connection in the Storage SOPs between two DICOM applications. Parallelism improves transmission time by overlapping the transmission times of DICOM objects. To make the proposed method practical and avoid any change in existing DICOM applications, a pair of interfaces was used. The interface pair carries out the parallel data transmission and the DICOM applications only need to send and receive their DICOM messages to the interfaces as usual.

The proposed method was assessed over both LAN and WAN networks. Four parallel-based methods were examined: pure parallelism with no compression and combinations of parallelism and three popular compression methods. The methods were compared with the commonly used method of medical data transmission that uses neither parallelism nor compression. On the experiments over LAN, it was observed that the speedup was slight for CT image datasets (about 1.45 to 1.63) but was relatively high for MR image datasets (about 2.21 to 3.47 times). Using the proposed method over WAN improved the transmission time drastically. The speedup was about 15 to 16 times for CT image datasets. However, a lower speedup was achieved for MR image datasets (about 5 times for one dataset and about 11 to 13 times for the others).

The effects of important factors such as compression ratio, compression time, number of parallel connections, and type of modality on the proposed parallel-based method of medical data transmission were discussed. It was observed that the type of modality and the type of network play important roles. To choose the best compression method in combination with parallelism, these two factors should be taken into account. In general there is no combination that outperforms for all modalities and networks.

Remote meeting experiments were conducted to assess the collaboration and remote accessibility of the approach. In the remote meeting experiments, a group of users worked with the remote application and tried to simulate a scenario in which a group of physicians work and discuss remotely on a medical case. A questionnaire was used for qualitative assessment. From the users point of view, the quality of the system was high. In addition, bandwidth required for the system was measured. The required bandwidth in the remote meeting experiments (about 400 Kb/sec) shows that the system can be used on the Internet.

Another set of experiments compared the bandwidth of MAST (i.e., the component that streams the remote application window to clients) with TightVNC (i.e., a popular remote desktop application). MAST was inferior in terms of bandwidth. In addition, it was sensitive to the speed of image motion speed. However, it was used because it works properly across platforms and provides application sharing instead of sharing the whole desktop.

The main advantage of the proposed method is that current medical imaging applications do not need any change and can be integrated within the architecture

easily. Two applications (i.e., ParaView and Slicer3) were used as examples of this capability.

Using the proposed approach can enormously enhance the existing medical imaging applications to be practically and widely used by physicians.

6.1 Future Work

Three-tier architecture provides great flexibility to the system. For instance, components of the tiers could be developed on different platforms. This thesis used Linux for developing data and processing tiers and Windows for implementing the components of the presentation tier. One of the future works could be developing the components of the tiers on other platforms (e.g., MAC OS or Linux for the presentation tier).

There are also other important features that usually come with n-tier architectures. These characteristics provide a path for further research:

1. Scalability is an important characteristic. Systems should always be expandable to address service demand growth. Using processing centers, clusters, and processing grids in the processing tier could be a way to provide scalability. However, there are other possibilities that could be investigated.

2. Security and privacy are important issues in medical systems. Medical imaging services should not violate security and privacy principles. The approach introduced in this thesis prevents direct access to patient records and in this way it enhances security. There could be some components in the processing tier to

supply additional security features. Security is one of the characteristics that can be further investigated in this work.

3. Increasing complexity: using interfaces between tiers help us to add more complexity to one tier without affecting other parts. This can be useful when new technologies and concepts are introduced. For instance, special software could be implemented in the data tier for organizing data in a faster way. The processing tier could be equipped with new components and features for caching data.

Other important aspects of n-tier architecture are [100]: usability, adaptability, manageability, reusability, and reliability that could be investigated for the system.

One important component used in this thesis is MAST. It captures the remote application window located in the processing tier and streams it to the presentation tier. Experimental results show that MAST was inferior to TightVNC. Improving MAST could be one important area that can advance the total performance of the system. Using new compression methods is one way for achieving this goal.

The use of parallel-based methods for medical data transmission is one of the major contributions of this thesis. For the future work, more investigations are needed to explore the different factors on the method performance (e.g., available network bandwidth). Moreover, more research studies are needed to determine the optimum number of parallel connections. Using this method may also affect the other applications running on the network. These effects could be inspected as well.

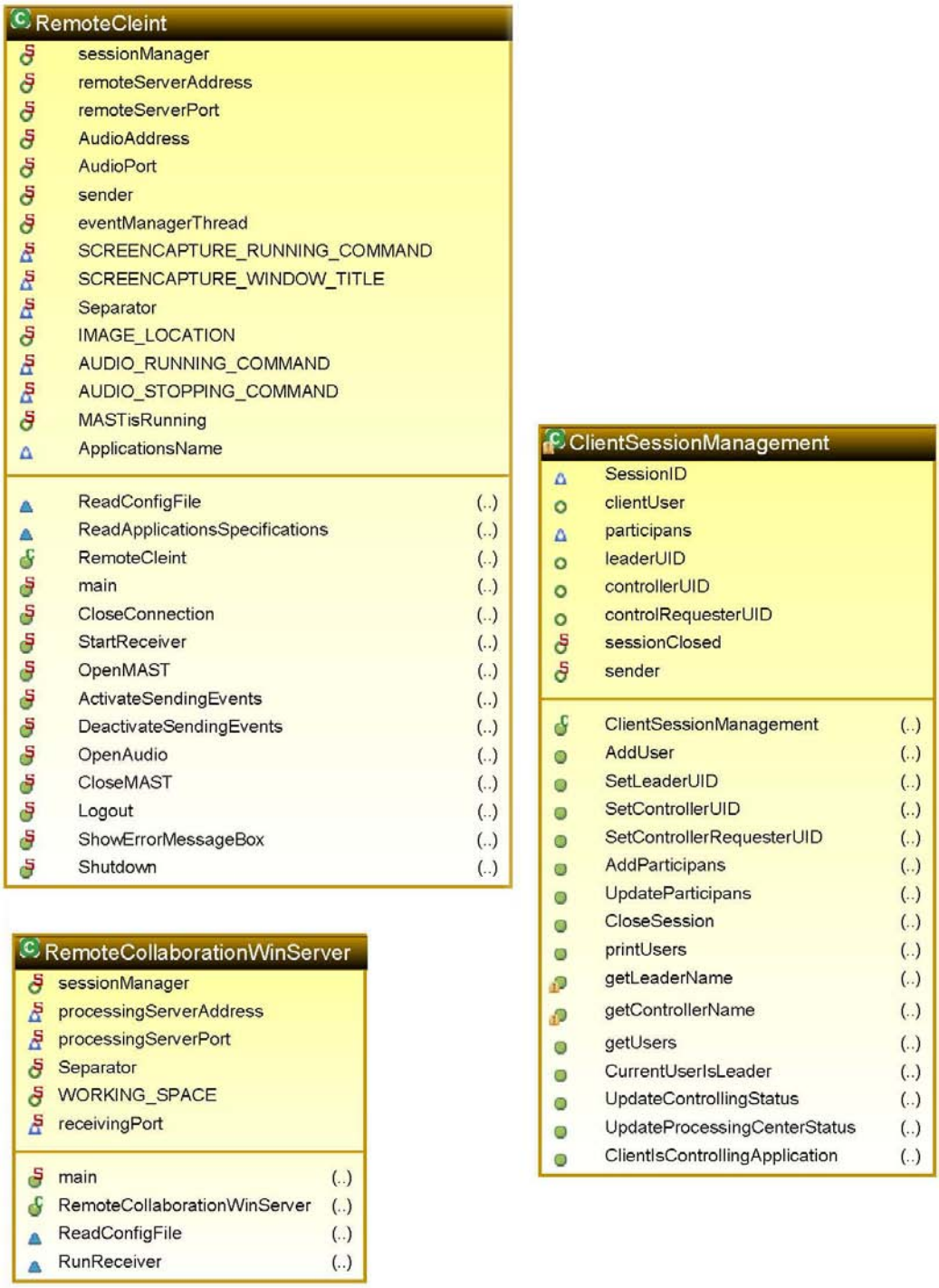
Finally, the assessment of the system was performed by ordinary users. However, it would be good to have it assessed by doctors and physicians in real applications.

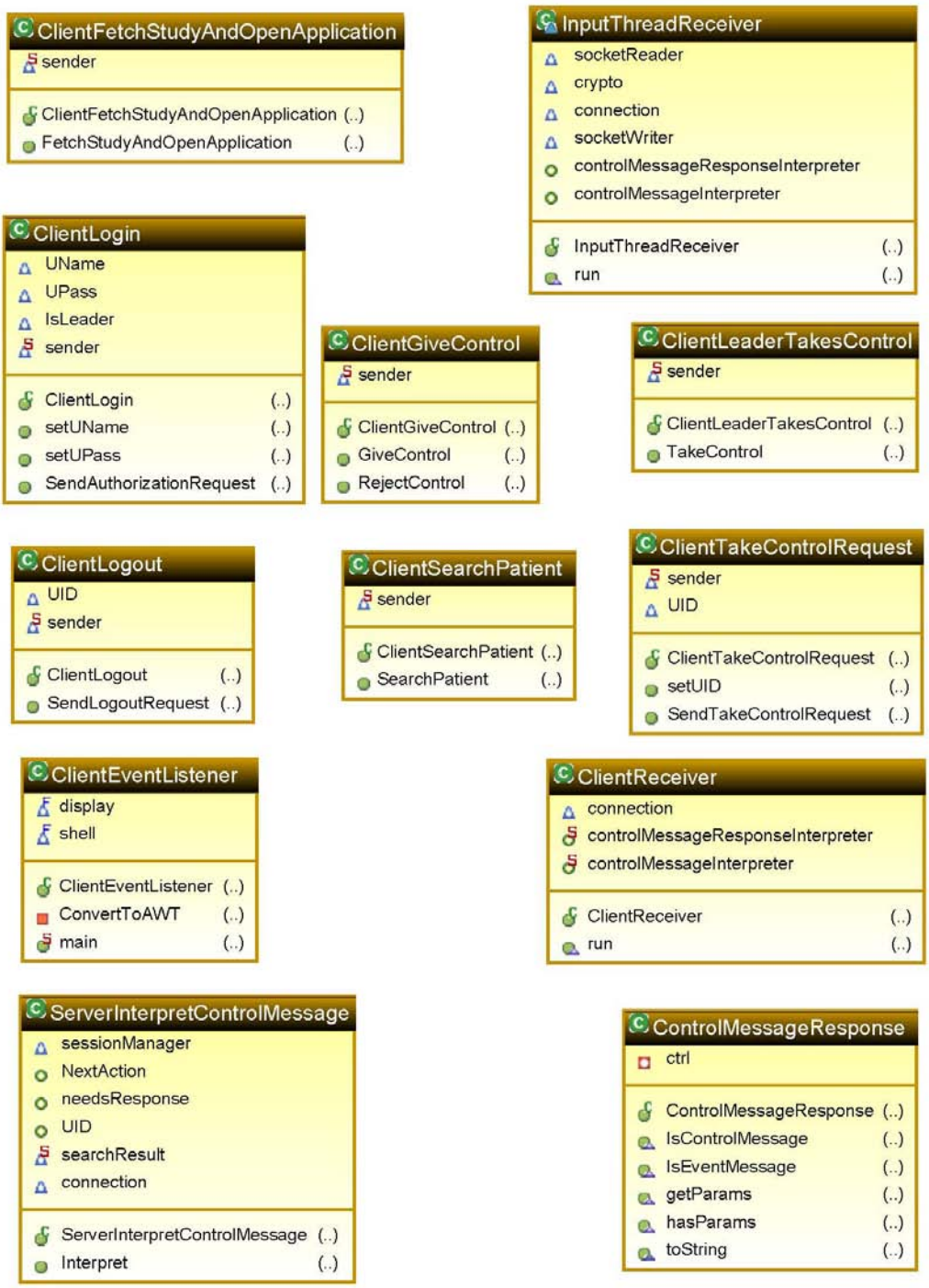
Feedback from doctors can improve the system and make it appropriate for real usage.

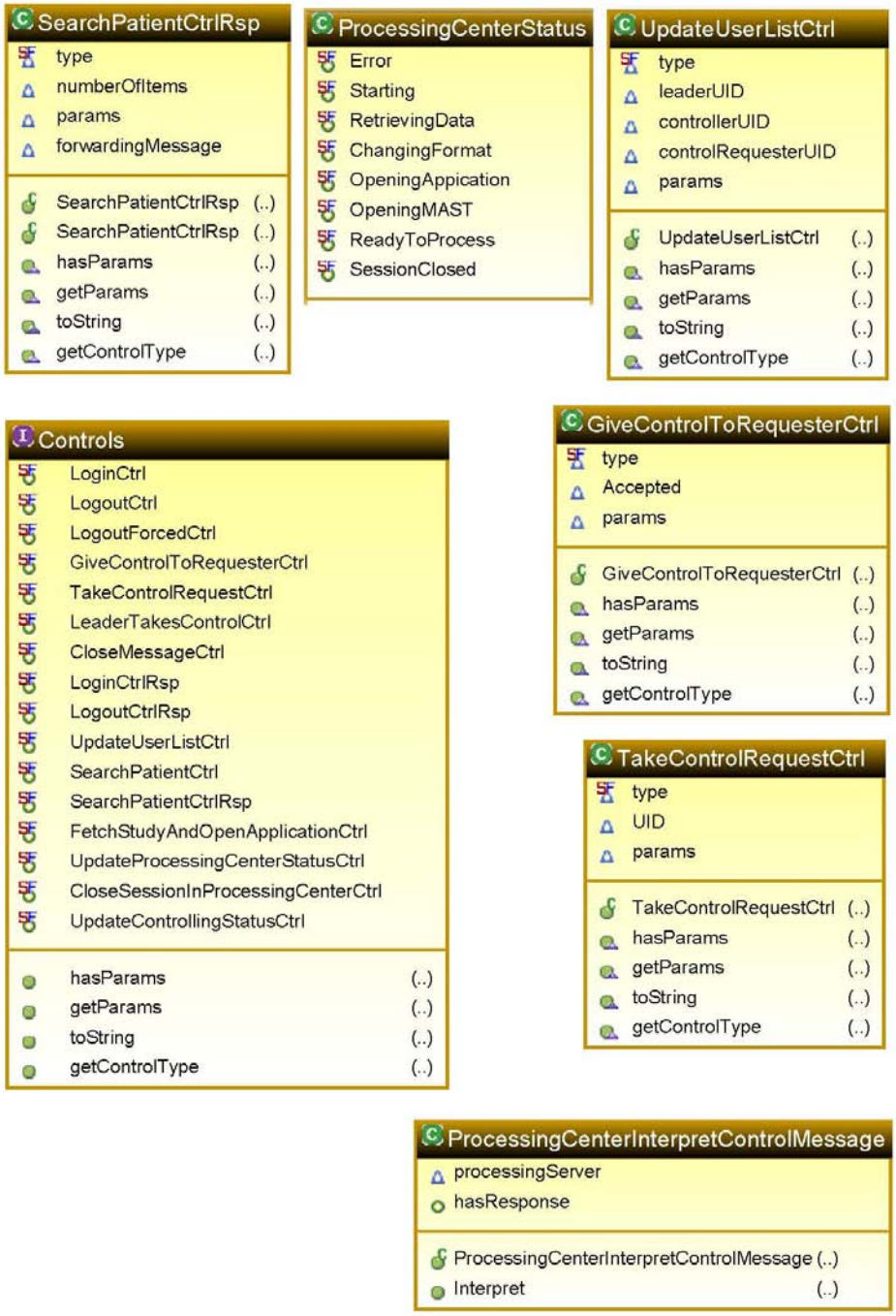
Appendix A

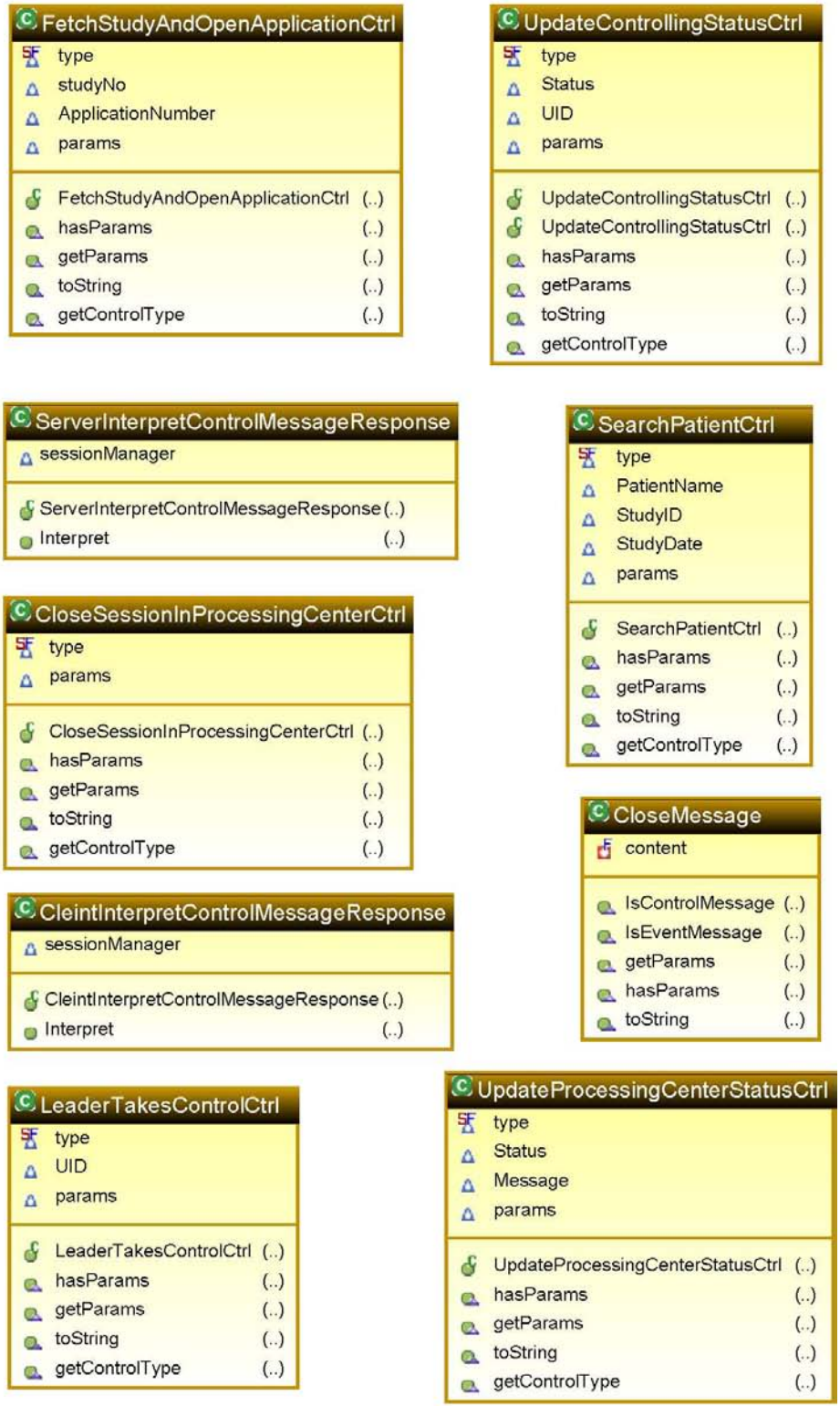
Class Diagrams

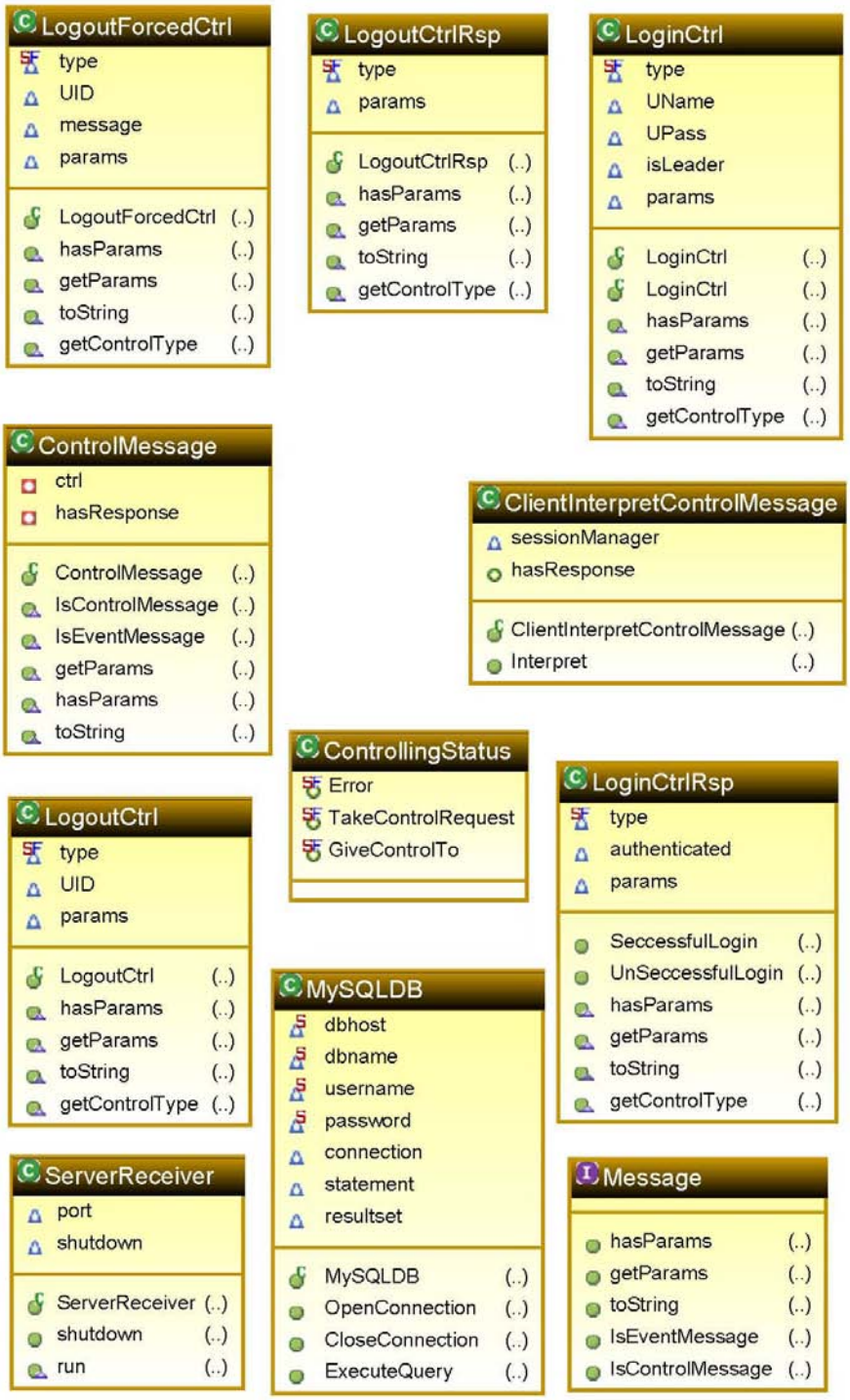
This appendix presents the class diagrams of the classes developed in Java. The relationships between classes are removed for the sake of simplicity. These classes were developped in 9 packages: client, server, processing, controls, db, events, GUI, network, and security.

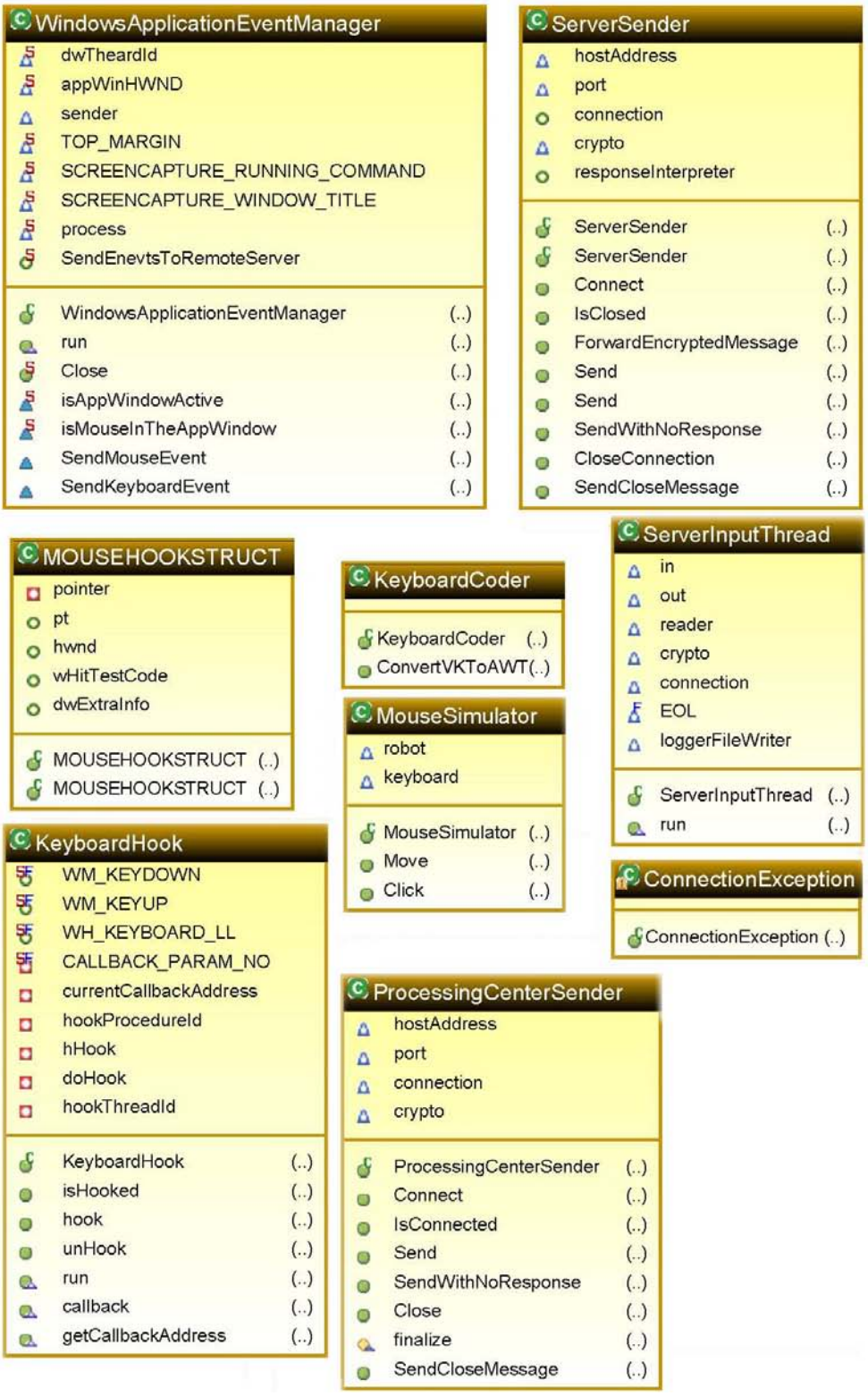

RemoteCleint
sessionManager
remoteServerAddress
remoteServerPort
AudioAddress
AudioPort
sender
eventManagerThread
SCREENCAPTURE_RUNNING_COMMAND
SCREENCAPTURE_WINDOW_TITLE
Separator
IMAGE_LOCATION
AUDIO_RUNNING_COMMAND
AUDIO_STOPPING_COMMAND
MASTisRunning
ApplicationsName
ReadConfigFile (..)
ReadApplicationsSpecifications (..)
RemoteCleint (..)
main (..)
CloseConnection (..)
StartReceiver (..)
OpenMAST (..)
ActivateSendingEvents (..)
DeactivateSendingEvents (..)
OpenAudio (..)
CloseMAST (..)
Logout (..)
ShowErrorMessageBox (..)
Shutdown (..)
ClientSessionManagement
SessionID
clientUser
participans
leaderUID
controllerUID
controlRequesterUID
sessionClosed
sender
ClientSessionManagement (..)
AddUser (..)
SetLeaderUID (..)
SetControllerUID (..)
SetControllerRequesterUID (..)
AddParticipans (..)
UpdateParticipans (..)
CloseSession (..)
printUsers (..)
getLeaderName (..)
getControllerName (..)
getUsers (..)
CurrentUserIsLeader (..)
UpdateControllingStatus (..)
UpdateProcessingCenterStatus (..)
ClientIsControllingApplication (..)
RemoteCollaborationWinServer
sessionManager
processingServerAddress
processingServerPort
Separator
WORKING_SPACE
receivingPort
main (..)
RemoteCollaborationWinServer (..)
ReadConfigFile (..)
RunReceiver (..)

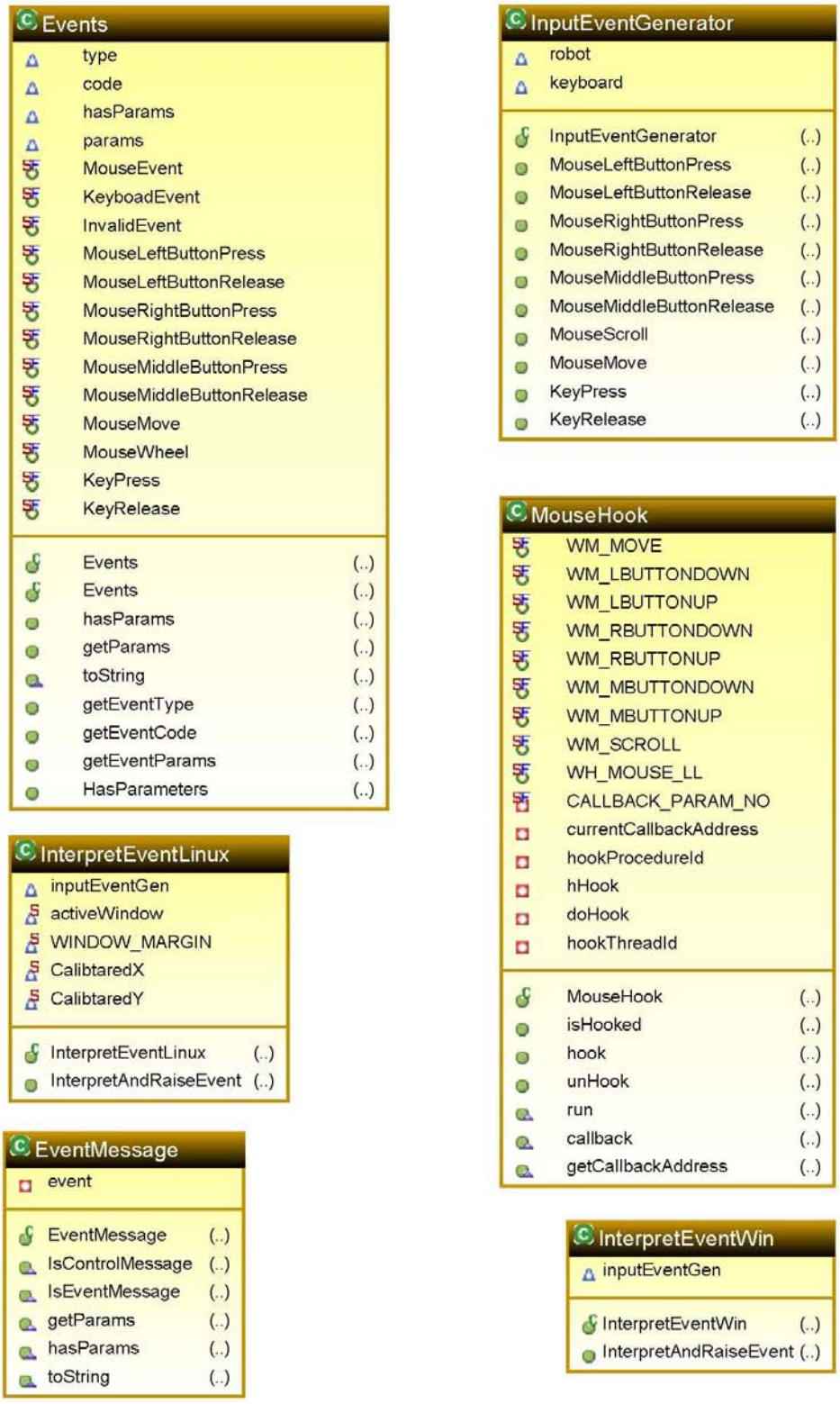

ClientFetchStudyAndOpenApplication
sender
ClientFetchStudyAndOpenApplication (..)
FetchStudyAndOpenApplication (..)
InputThreadReceiver
socketReader
crypto
connection
socketWriter
controlMessageResponseInterpreter
controlMessageInterpreter
InputThreadReceiver (..)
run (..)
ClientLogin
UName
UPass
IsLeader
sender
ClientLogin (..)
setUName (..)
setUPass (..)
SendAuthorizationRequest (..)
ClientGiveControl
sender
ClientGiveControl (..)
GiveControl (..)
RejectControl (..)
ClientLeaderTakesControl
sender
ClientLeaderTakesControl (..)
TakeControl (..)
ClientLogout
UID
sender
ClientLogout (..)
SendLogoutRequest (..)
ClientSearchPatient
sender
ClientSearchPatient (..)
SearchPatient (..)
ClientTakeControlRequest
sender
UID
ClientTakeControlRequest (..)
setUID (..)
SendTakeControlRequest (..)
ClientEventListener
display
shell
ClientEventListener (..)
ConvertToAWT (..)
main (..)
ClientReceiver
connection
controlMessageResponseInterpreter
controlMessageInterpreter
ClientReceiver (..)
run (..)
ServerInterpretControlMessage
sessionManager
NextAction
needsResponse
UID
searchResult
connection
ServerInterpretControlMessage (..)
Interpret (..)
ControlMessageResponse
ctrl
ControlMessageResponse (..)
IsControlMessage (..)
IsEventMessage (..)
getParams (..)
hasParams (..)
toString (..)

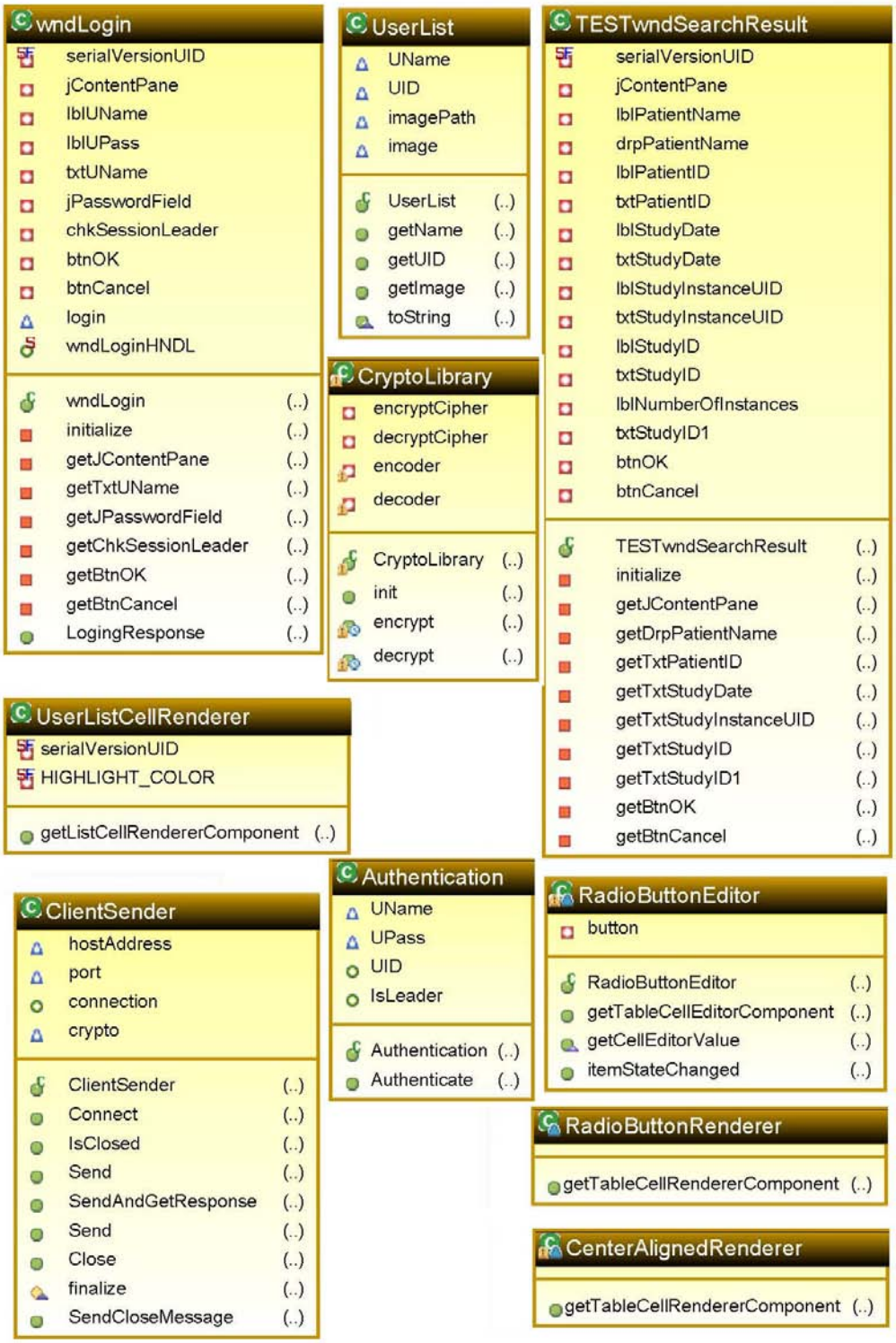

SearchPatientCtrlRsp
type
numberOfItems
params
forwardingMessage
SearchPatientCtrlRsp (..)
SearchPatientCtrlRsp (..)
hasParams (..)
getParams (..)
toString (..)
getControlType (..)
ProcessingCenterStatus
Error
Starting
RetrievingData
ChangingFormat
OpeningAppication
OpeningMAST
ReadyToProcess
SessionClosed
UpdateUserListCtrl
type
leaderUID
controllerUID
controlRequesterUID
params
UpdateUserListCtrl (..)
hasParams (..)
getParams (..)
toString (..)
getControlType (..)
Controls
LoginCtrl
LogoutCtrl
LogoutForcedCtrl
GiveControlToRequesterCtrl
TakeControlRequestCtrl
LeaderTakesControlCtrl
CloseMessageCtrl
LoginCtrlRsp
LogoutCtrlRsp
UpdateUserListCtrl
SearchPatientCtrl
SearchPatientCtrlRsp
FetchStudyAndOpenApplicationCtrl
UpdateProcessingCenterStatusCtrl
CloseSessionInProcessingCenterCtrl
UpdateControllingStatusCtrl
hasParams (..)
getParams (..)
toString (..)
getControlType (..)
GiveControlToRequesterCtrl
type
Accepted
params
GiveControlToRequesterCtrl (..)
hasParams (..)
getParams (..)
toString (..)
getControlType (..)
TakeControlRequestCtrl
type
UID
params
TakeControlRequestCtrl (..)
hasParams (..)
getParams (..)
toString (..)
getControlType (..)
ProcessingCenterInterpretControlMessage
processingServer
hasResponse
ProcessingCenterInterpretControlMessage (..)
Interpret (..)

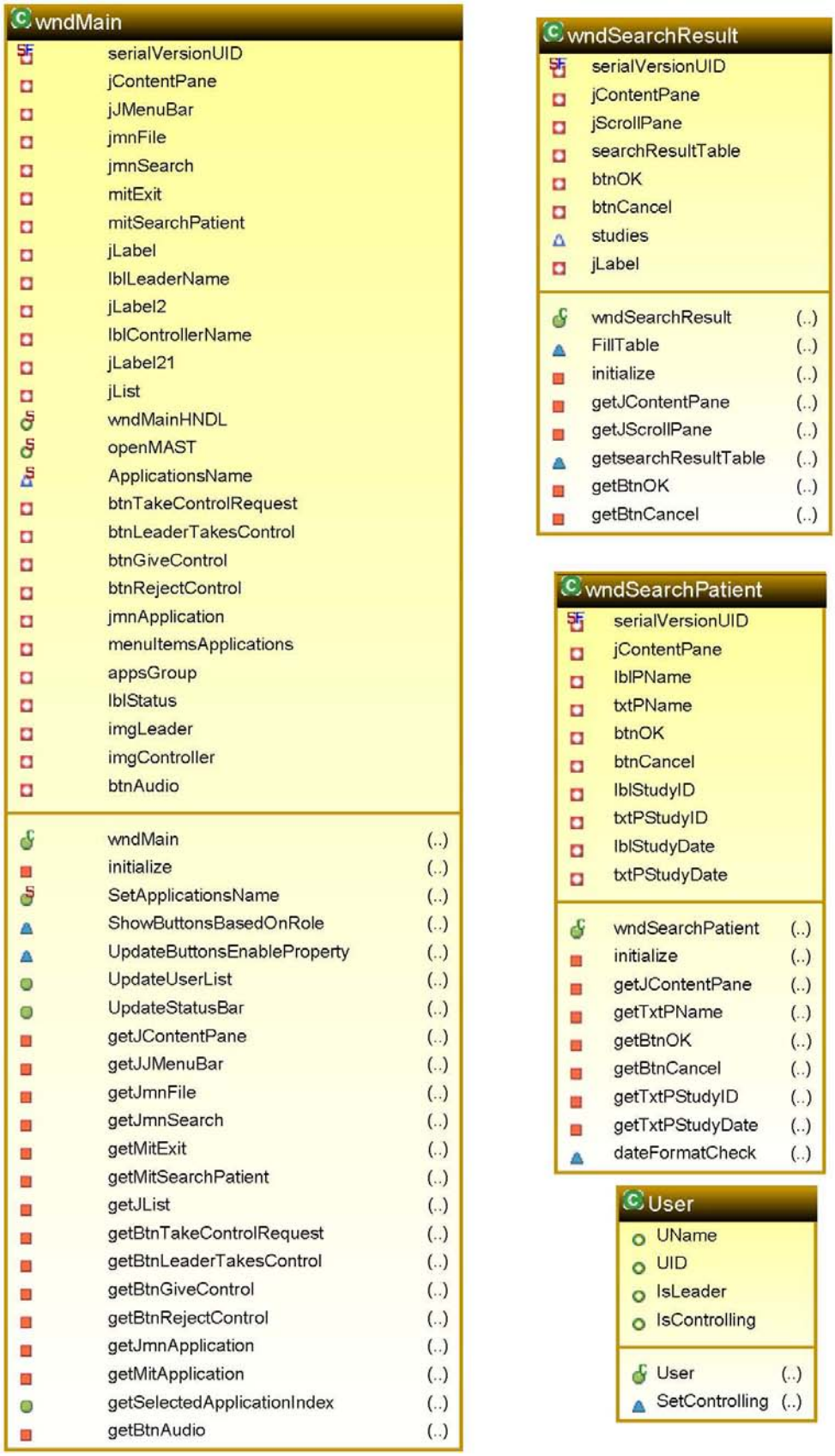

FetchStudyAndOpenApplicationCtrl
type
studyNo
ApplicationNumber
params
FetchStudyAndOpenApplicationCtrl (..)
hasParams (..)
getParams (..)
toString (..)
getControlType (..)
UpdateControllingStatusCtrl
type
Status
UID
params
UpdateControllingStatusCtrl (..)
UpdateControllingStatusCtrl (..)
hasParams (..)
getParams (..)
toString (..)
getControlType (..)
ServerInterpretControlMessageResponse
sessionManager
ServerInterpretControlMessageResponse (..)
Interpret (..)
SearchPatientCtrl
type
PatientName
StudyID
StudyDate
params
SearchPatientCtrl (..)
hasParams (..)
getParams (..)
toString (..)
getControlType (..)
CloseSessionInProcessingCenterCtrl
type
params
CloseSessionInProcessingCenterCtrl (..)
hasParams (..)
getParams (..)
toString (..)
getControlType (..)
CloseMessage
content
IsControlMessage (..)
IsEventMessage (..)
getParams (..)
hasParams (..)
toString (..)
CleintInterpretControlMessageResponse
sessionManager
CleintInterpretControlMessageResponse (..)
Interpret (..)
LeaderTakesControlCtrl
type
UID
params
LeaderTakesControlCtrl (..)
hasParams (..)
getParams (..)
toString (..)
getControlType (..)
UpdateProcessingCenterStatusCtrl
type
Status
Message
params
UpdateProcessingCenterStatusCtrl (..)
hasParams (..)
getParams (..)
toString (..)
getControlType (..)

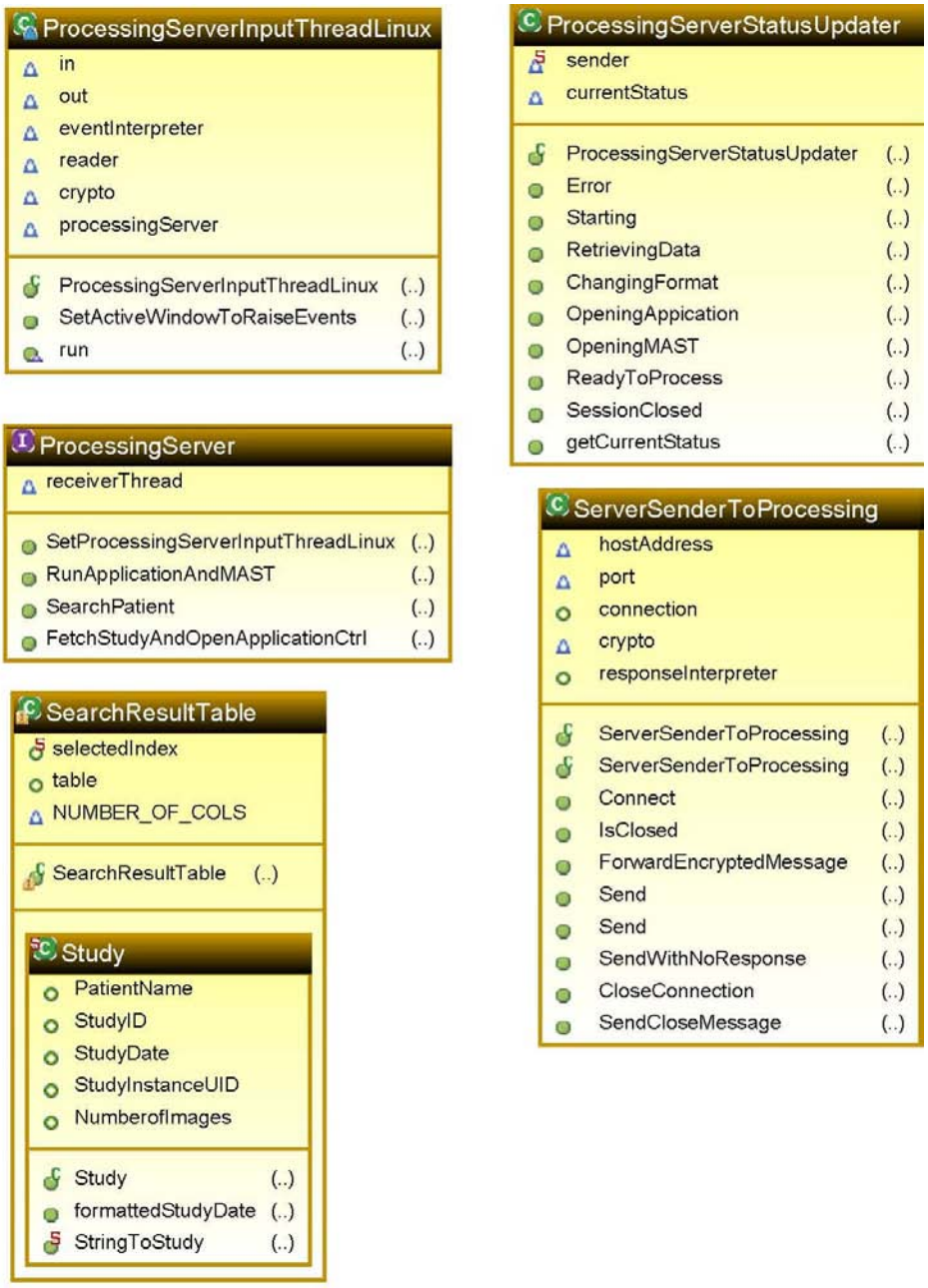

LogoutForcedCtrl
type
UID
message
params
LogoutForcedCtrl (..)
hasParams (..)
getParams (..)
toString (..)
getControlType (..)
LogoutCtrlRsp
type
params
LogoutCtrlRsp (..)
hasParams (..)
getParams (..)
toString (..)
getControlType (..)
LoginCtrl
type
UName
UPass
isLeader
params
LoginCtrl (..)
LoginCtrl (..)
hasParams (..)
getParams (..)
toString (..)
getControlType (..)
ControlMessage
ctrl
hasResponse
ControlMessage (..)
IsControlMessage (..)
IsEventMessage (..)
getParams (..)
hasParams (..)
toString (..)
ClientInterpretControlMessage
sessionManager
hasResponse
ClientInterpretControlMessage (..)
Interpret (..)
ControllingStatus
Error
TakeControlRequest
GiveControlTo
LogoutCtrl
type
UID
params
LogoutCtrl (..)
hasParams (..)
getParams (..)
toString (..)
getControlType (..)
LoginCtrlRsp
type
authenticated
params
SeccessfulLogin (..)
UnSeccessfulLogin (..)
hasParams (..)
getParams (..)
toString (..)
getControlType (..)
MySQLDB
dbhost
dbname
username
password
connection
statement
resultset
MySQLDB (..)
OpenConnection (..)
CloseConnection (..)
ExecuteQuery (..)
ServerReceiver
port
shutdown
ServerReceiver (..)
shutdown (..)
run (..)
Message
hasParams (..)
getParams (..)
toString (..)
IsEventMessage (..)
IsControlMessage (..)

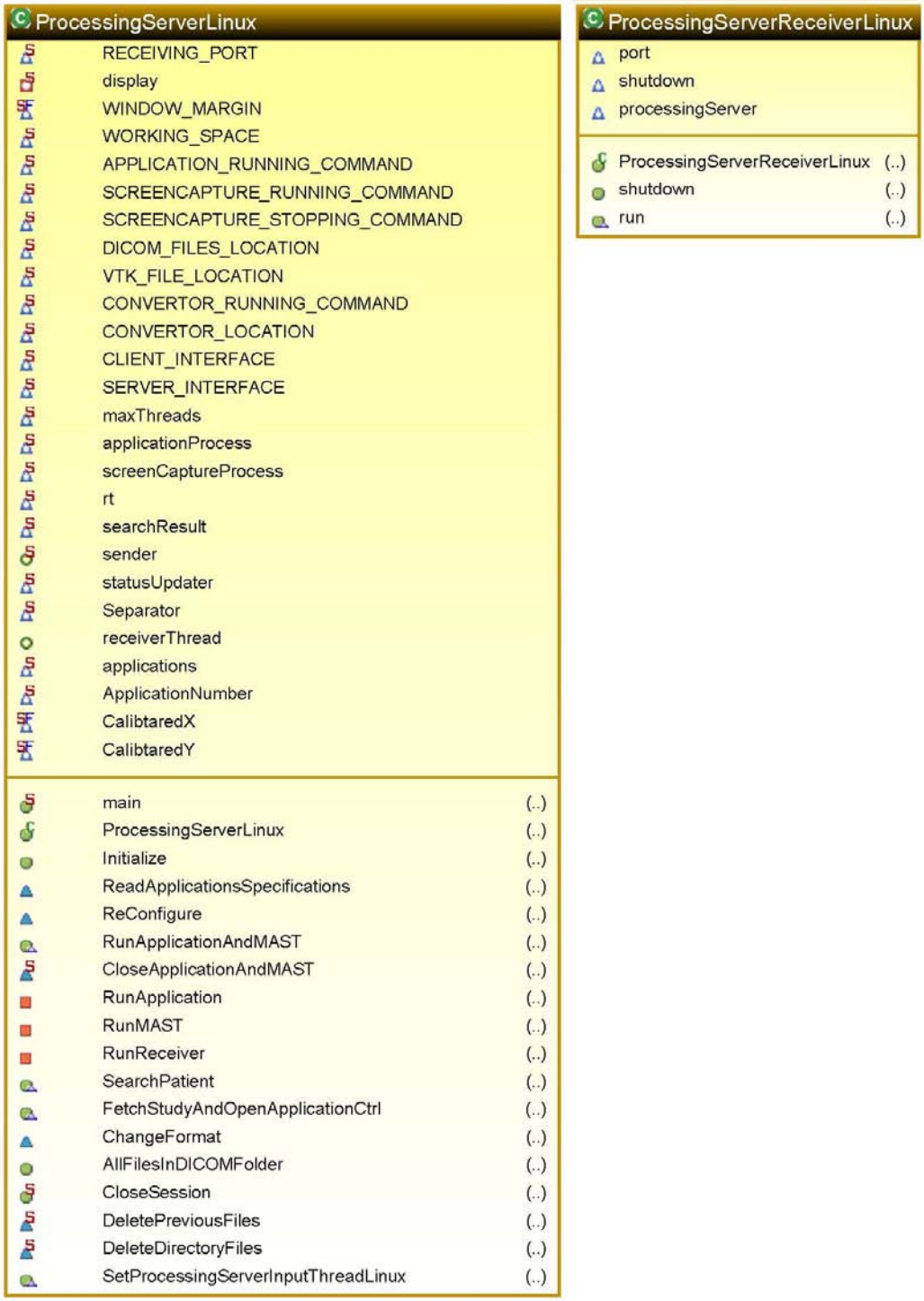

WindowsApplicationEventManager
dwTheardId
appWinHWND
sender
TOP_MARGIN
SCREENCAPTURE_RUNNING_COMMAND
SCREENCAPTURE_WINDOW_TITLE
process
SendEnevtsToRemoteServer
WindowsApplicationEventManager (..)
run (..)
Close (..)
isAppWindowActive (..)
isMouseInTheAppWindow (..)
SendMouseEvent (..)
SendKeyboardEvent (..)
ServerSender
hostAddress
port
connection
crypto
responseInterpreter
ServerSender (..)
ServerSender (..)
Connect (..)
IsClosed (..)
ForwardEncryptedMessage (..)
Send (..)
Send (..)
SendWithNoResponse (..)
CloseConnection (..)
SendCloseMessage (..)
MOUSEHOOKSTRUCT
pointer
pt
hwnd
wHitTestCode
dwExtraInfo
MOUSEHOOKSTRUCT (..)
MOUSEHOOKSTRUCT (..)
KeyboardCoder
KeyboardCoder (..)
ConvertVKToAWT(..)
MouseSimulator
robot
keyboard
MouseSimulator (..)
Move (..)
Click (..)
ServerInputThread
in
out
reader
crypto
connection
EOL
loggerFileWriter
ServerInputThread (..)
run (..)
ConnectionException
ConnectionException (..)
KeyboardHook
WM_KEYDOWN
WM_KEYUP
WH_KEYBOARD_LL
CALLBACK_PARAM_NO
currentCallbackAddress
hookProcedureId
hHook
doHook
hookThreadId
KeyboardHook (..)
isHooked (..)
hook (..)
unHook (..)
run (..)
callback (..)
getCallbackAddress (..)
ProcessingCenterSender
hostAddress
port
connection
crypto
ProcessingCenterSender (..)
Connect (..)
IsConnected (..)
Send (..)
SendWithNoResponse (..)
Close (..)
finalize (..)
SendCloseMessage (..)

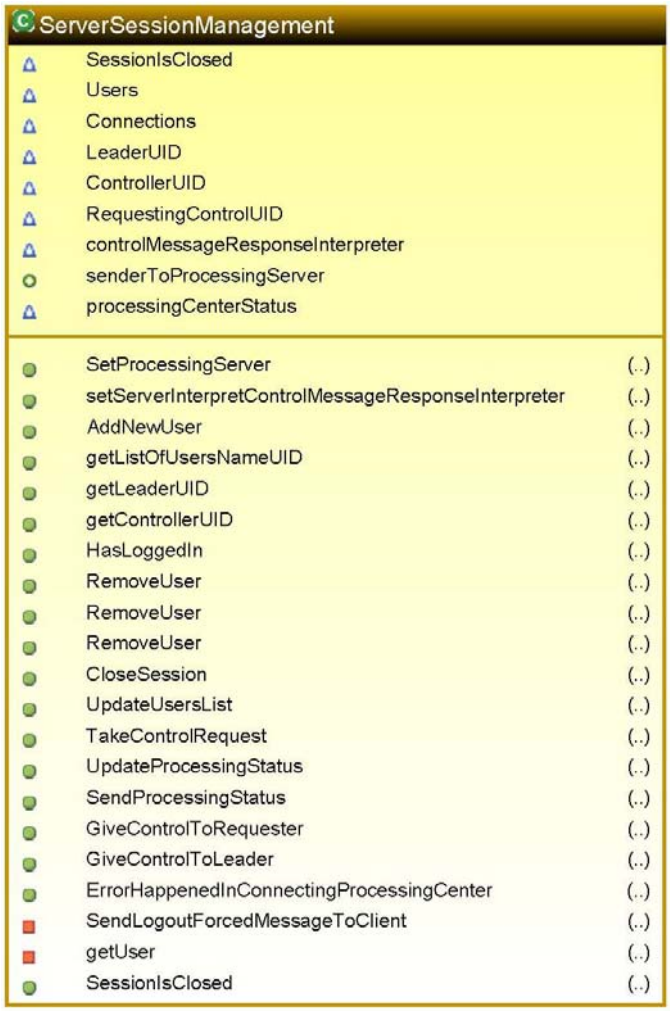

Events
type
code
hasParams
params
MouseEvent
KeyboadEvent
InvalidEvent
MouseLeftButtonPress
MouseLeftButtonRelease
MouseRightButtonPress
MouseRightButtonRelease
MouseMiddleButtonPress
MouseMiddleButtonRelease
MouseMove
MouseWheel
KeyPress
KeyRelease
Events (..)
Events (..)
hasParams (..)
getParams (..)
toString (..)
getEventType (..)
getEventCode (..)
getEventParams (..)
HasParameters (..)
InputEventGenerator
robot
keyboard
InputEventGenerator (..)
MouseLeftButtonPress (..)
MouseLeftButtonRelease (..)
MouseRightButtonPress (..)
MouseRightButtonRelease (..)
MouseMiddleButtonPress (..)
MouseMiddleButtonRelease (..)
MouseScroll (..)
MouseMove (..)
KeyPress (..)
KeyRelease (..)
MouseHook
WM_MOVE
WM_LBUTTONDOWN
WM_LBUTTONUP
WM_RBUTTONDOWN
WM_RBUTTONUP
WM_MBUTTONDOWN
WM_MBUTTONUP
WM_SCROLL
WH_MOUSE_LL
CALLBACK_PARAM_NO
currentCallbackAddress
hookProcedureId
hHook
doHook
hookThreadId
MouseHook (..)
isHooked (..)
hook (..)
unHook (..)
run (..)
callback (..)
getCallbackAddress (..)
InterpretEventLinux
inputEventGen
activeWindow
WINDOW_MARGIN
CalibtaredX
CalibtaredY
InterpretEventLinux (..)
InterpretAndRaiseEvent (..)
EventMessage
event
EventMessage (..)
IsControlMessage (..)
IsEventMessage (..)
getParams (..)
hasParams (..)
toString (..)
InterpretEventWin
inputEventGen
InterpretEventWin (..)
InterpretAndRaiseEvent (..)


wndLogin
serialVersionUID
jContentPane
lblUName
lblUPass
txtUName
jPasswordField
chkSessionLeader
btnOK
btnCancel
login
wndLoginHNDL
wndLogin (..)
initialize (..)
getJContentPane (..)
getTxtUName (..)
getJPasswordField (..)
getChkSessionLeader (..)
getBtnOK (..)
getBtnCancel (..)
LogingResponse (..)
UserList
UName
UID
imagePath
image
UserList (..)
getName (..)
getUID (..)
getImage (..)
toString (..)
CryptoLibrary
encryptCipher
decryptCipher
encoder
decoder
CryptoLibrary (..)
init (..)
encrypt (..)
decrypt (..)
TESTwndSearchResult
serialVersionUID
jContentPane
lblPatientName
drpPatientName
lblPatientID
txtPatientID
lblStudyDate
txtStudyDate
lblStudyInstanceUID
txtStudyInstanceUID
lblStudyID
txtStudyID
lblNumberOfInstances
txtStudyID1
btnOK
btnCancel
TESTwndSearchResult (..)
initialize (..)
getJContentPane (..)
getDrpPatientName (..)
getTxtPatientID (..)
getTxtStudyDate (..)
getTxtStudyInstanceUID (..)
getTxtStudyID (..)
getTxtStudyID1 (..)
getBtnOK (..)
getBtnCancel (..)
UserListCellRenderer
serialVersionUID
HIGHLIGHT_COLOR
getListCellRendererComponent (..)
Authentication
UName
UPass
UID
IsLeader
Authentication (..)
Authenticate (..)
RadioButtonEditor
button
RadioButtonEditor (..)
getTableCellEditorComponent (..)
getCellEditorValue (..)
itemStateChanged (..)
ClientSender
hostAddress
port
connection
crypto
ClientSender (..)
Connect (..)
IsClosed (..)
Send (..)
SendAndGetResponse (..)
Send (..)
Close (..)
finalize (..)
SendCloseMessage (..)
RadioButtonRenderer
getTableCellRendererComponent (..)
CenterAlignedRenderer
getTableCellRendererComponent (..)


wndMain
serialVersionUID
jContentPane
jJMenuBar
jmnFile
jmnSearch
mitExit
mitSearchPatient
jLabel
lblLeaderName
jLabel2
lblControllerName
jLabel21
jList
wndMainHNDL
openMAST
ApplicationsName
btnTakeControlRequest
btnLeaderTakesControl
btnGiveControl
btnRejectControl
jmnApplication
menuItemsApplications
appsGroup
lblStatus
imgLeader
imgController
btnAudio
wndMain (..)
initialize (..)
SetApplicationsName (..)
ShowButtonsBasedOnRole (..)
UpdateButtonsEnableProperty (..)
UpdateUserList (..)
UpdateStatusBar (..)
getJContentPane (..)
getJJMenuBar (..)
getJmnFile (..)
getJmnSearch (..)
getMitExit (..)
getMitSearchPatient (..)
getJList (..)
getBtnTakeControlRequest (..)
getBtnLeaderTakesControl (..)
getBtnGiveControl (..)
getBtnRejectControl (..)
getJmnApplication (..)
getMitApplication (..)
getSelectedApplicationIndex (..)
getBtnAudio (..)
wndSearchResult
serialVersionUID
jContentPane
jScrollPane
searchResultTable
btnOK
btnCancel
studies
jLabel
wndSearchResult (..)
FillTable (..)
initialize (..)
getJContentPane (..)
getJScrollPane (..)
getsearchResultTable (..)
getBtnOK (..)
getBtnCancel (..)
wndSearchPatient
serialVersionUID
jContentPane
lblPName
txtPName
btnOK
btnCancel
lblStudyID
txtPStudyID
lblStudyDate
txtPStudyDate
wndSearchPatient (..)
initialize (..)
getJContentPane (..)
getTxtPName (..)
getBtnOK (..)
getBtnCancel (..)
getTxtPStudyID (..)
getTxtPStudyDate (..)
dateFormatCheck (..)
User
UName
UID
IsLeader
IsControlling
User (..)
SetControlling (..)


ProcessingServerInputThreadLinux
in
out
eventInterpreter
reader
crypto
processingServer
ProcessingServerInputThreadLinux (..)
SetActiveWindowToRaiseEvents (..)
run (..)
ProcessingServerStatusUpdater
sender
currentStatus
ProcessingServerStatusUpdater (..)
Error (..)
Starting (..)
RetrievingData (..)
ChangingFormat (..)
OpeningAppication (..)
OpeningMAST (..)
ReadyToProcess (..)
SessionClosed (..)
getCurrentStatus (..)
ProcessingServer
receiverThread
SetProcessingServerInputThreadLinux (..)
RunApplicationAndMAST (..)
SearchPatient (..)
FetchStudyAndOpenApplicationCtrl (..)
ServerSenderToProcessing
hostAddress
port
connection
crypto
responseInterpreter
ServerSenderToProcessing (..)
ServerSenderToProcessing (..)
Connect (..)
IsClosed (..)
ForwardEncryptedMessage (..)
Send (..)
Send (..)
SendWithNoResponse (..)
CloseConnection (..)
SendCloseMessage (..)
SearchResultTable
selectedIndex
table
NUMBER_OF_COLS
SearchResultTable (..)
Study
PatientName
StudyID
StudyDate
StudyInstanceUID
NumberofImages
Study (..)
formattedStudyDate (..)
StringToStudy (..)


ProcessingServerLinux
RECEIVING_PORT
display
WINDOW_MARGIN
WORKING_SPACE
APPLICATION_RUNNING_COMMAND
SCREENCAPTURE_RUNNING_COMMAND
SCREENCAPTURE_STOPPING_COMMAND
DICOM_FILES_LOCATION
VTK_FILE_LOCATION
CONVERTOR_RUNNING_COMMAND
CONVERTOR_LOCATION
CLIENT_INTERFACE
SERVER_INTERFACE
maxThreads
applicationProcess
screenCaptureProcess
rt
searchResult
sender
statusUpdater
Separator
receiverThread
applications
ApplicationNumber
CalibtaredX
CalibtaredY
main (..)
ProcessingServerLinux (..)
Initialize (..)
ReadApplicationsSpecifications (..)
ReConfigure (..)
RunApplicationAndMAST (..)
CloseApplicationAndMAST (..)
RunApplication (..)
RunMAST (..)
RunReceiver (..)
SearchPatient (..)
FetchStudyAndOpenApplicationCtrl (..)
ChangeFormat (..)
AllFilesInDICOMFolder (..)
CloseSession (..)
DeletePreviousFiles (..)
DeleteDirectoryFiles (..)
SetProcessingServerInputThreadLinux (..)
ProcessingServerReceiverLinux
port
shutdown
processingServer
ProcessingServerReceiverLinux (..)
shutdown (..)
run (..)


ServerSessionManagement
SessionIsClosed
Users
Connections
LeaderUID
ControllerUID
RequestingControlUID
controlMessageResponseInterpreter
senderToProcessingServer
processingCenterStatus
SetProcessingServer (..)
setServerInterpretControlMessageResponseInterpreter (..)
AddNewUser (..)
getListOfUsersNameUID (..)
getLeaderUID (..)
getControllerUID (..)
HasLoggedIn (..)
RemoveUser (..)
RemoveUser (..)
RemoveUser (..)
CloseSession (..)
UpdateUsersList (..)
TakeControlRequest (..)
UpdateProcessingStatus (..)
SendProcessingStatus (..)
GiveControlToRequester (..)
GiveControlToLeader (..)
ErrorHappenedInConnectingProcessingCenter (..)
SendLogoutForcedMessageToClient (..)
getUser (..)
SessionIsClosed (..)

Appendix B

Script for Making Animations

This appendix shows the python script that creates the rotating image with 30 degrees/second speed. This script creates a 12-second animation. To make a longer animation:

1. The variable scene.NumberOfFrames should be changed. Since 30 frames/second has been used in the animation, the variable should be set to DurationOfAnimation*30. For instance for a 180 second animation it should be set to 5400.

2. The variable KeyframesNo should be calculated by the following formula: KeyframesNo=DurationOfAnimation/Time for a complete rotation*360

The script should run on ParaView for the 3D visualized image.

```
from paraview.servermanager import *
import math
# Initialization to set up some visualization
if not ActiveConnection:
```

```
        Connect()
view = GetRenderView()
view.ResetCamera()
# Create an animation scene
scene = animation.AnimationScene()
# Add the view to the scene
scene.ViewModules = [view]
# Set the number of frame in one loop.
scene.NumberOfFrames = 360
# Create an animation cue for the camera.
cue = animation.CameraAnimationCue()
cue.AnimatedProxy = view
# Add the cue to the scene.
scene.Cues = [cue]
# Now create keyframes.
camera = view.GetActiveCamera()
KeyframesNo = 360
[x1,y1,z1]=view.CameraFocalPoint.GetData()
[x2,y2,z2]=view.CameraPosition.GetData()
radus=math.sqrt((x1-x2)**2+(z1-z2)**2)
for i in range(0, KeyframesNo):
        # Changing the position of the camera
        rotatingX = x1-radus*math.sin(i*(math.pi/180))
```

```
        rotatingZ = z1+radus*math.cos(i*(math.pi/180))
        view.CameraPosition = [rotatingX, y1,rotatingZ]
        view.ResetCamera()
        # End of Changing the position of the camera
        keyframe = animation.CameraKeyFrame()
        # set the value of the key frame to the current camera location.
        keyframe.KeyTime = i*1.0/KeyframesNo
        keyframe.Position = camera.GetPosition()
        keyframe.FocalPoint = camera.GetFocalPoint()
        keyframe.ViewUp = camera.GetViewUp()
        keyframe.ViewAngle = camera.GetViewAngle()
        cue.KeyFrames.append(keyframe)
# Setting the properties of the animation
filename = "c:\rotation30.avi"
writer = vtkSMAnimationSceneImageWriter();
writer.SetFileName(filename);
writer.SetFrameRate(30);
writer.SetMagnification(3)
writer.SetAnimationScene(scene.SMProxy);
#Now save the animation.
if not writer.Save():
        raise exceptions.RuntimeError, "Saving of animation failed!"
```

Bibliography

[1] H. K. Huang. Medical imaging informatics research and development trendsan editorial. *Computerized Medical Imaging and Graphics*, 29(2-3):91–93, 2005.

[2] H. K. Huang and H. K. Huang. *PACS and imaging informatics: basic principles and applications*. Wiley-Liss, Hoboken, New Jersey, 2004.

[3] J. C. Duncan and N. Ayache. Medical image analysis: Progress over two decades and the challenges ahead. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):85–106, 2000.

[4] K. J. Dreyer. *PACS : a guide to the digital revolution*. Springer, New York, 2006.

[5] I. Maglogiannis, C. Delakouridis, and L. Kazatzopoulos. Enabling collaborative medical diagnosis over the internet via peer-to-peer distribution of electronic health records. *Journal of Medical Systems*, 30(2):107–116, 2006.

[6] G. Graschew, T. A. Roelofs, and P. M. Rakowsky Schlag. Digital medicine in the virtual hospital of the future. *International Journal of Computer Assisted Radiology and Surgery*, 1(1):119–120, 2006.

[7] R. Maani, S. Camorlinga, and R. Eskicioglu. A remote real-time PACS-based platform for medical imaging telemedicine. In *Medical Imaging 2009: Advanced PACS-based Imaging Informatics and Therapeutic Applications*, volume 7264, page 72640Q. SPIE, February 26 2009.

[8] S. S. Boochever. HIS/RIS/PACS integration: getting to the gold standard. *Radiology Management*, 26(3):16–24, 2004.

[9] National Electrical Manufacturers Association. DICOM official website. URL: `http://medical.nema.org/`, Accessed in May, 2010.

[10] K. M. McConnochie, N. E. Wood, N. E. Herendeen, P. K. Ng, K. Noyes, H. Wang, and K. J. Roghmann. Acute illness care patterns change with use of telemedicine. *Pediatrics*, 123(6):e989–e995, 2009.

[11] C. A. Sable, S. D. Cummings, G. D. Pearson, L. M. Schratz, R. C. Cross, E. S. Quivers, H. Rudra, and G. R. Martin. Impact of telemedicine on the practice of pediatric cardiology in community hospitals. *Pediatrics*, 109(1):E3, 2002.

[12] D. Tahmoush and Ha. Samet. A web collaboration system for content-based image retrieval of medical images. In *Medical Imaging 2007: Image Processing*, March 8 2007.

[13] S. Park, W. Kim, and I. Ihm. Mobile collaborative medical display system. *Computer Methods and Programs in Biomedicine*, 89(3):248–260, 2008.

[14] G. F. Welch, D. H. Sonnenwald, H. Fuchs, B. Cairns, K. Mayer-Patel, H. M. Soderholm, R. Yang, A. State, H. Towles, A. Ilie, M. K. Ampalam, S. Krishnan,

V. Noel, M. Noland, and J. E. Manning. 3d medical collaboration technology to enhance emergency healthcare. *Journal of Biomedical Discovery and Collaboration*, 4(4):4, 2009.

[15] Z. Zhou, A. Le, B. Liu, and H. K. Huang. PACS-CAD toolkit for integrating an independent cad workstation to diagnostic workflow. In *Medical Imaging 2007: PACS and Imaging Informatics*, volume 6516, page 651609. SPIE, March 8 2007.

[16] Z. Zhou, B. J. Liu, and A. H. Le. CADPACS integration tool kit based on DICOM secondary capture, structured report and ihe workflow profiles. *Computerized Medical Imaging and Graphics*, 31(4-5):346–352, 2007.

[17] A. Le, L. Mai, B. Liu, and H. K. Huang. The workflow and procedures for automatic integration of a computer-aided diagnosis workstation with a clinical PACS with real world examples. In *Medical Imaging 2008: PACS and Imaging Informatics*, volume 6919, page 69190U. SPIE, March 6 2008.

[18] R. Ullah, D. Gilliland, and D. Adams. Otolaryngology consultations by real-time telemedicine. *The Ulster Medical Journal*, 71(1):26–29, 2002.

[19] A. L. Reynolds, J. L. Vick, and N. J. Haak. Telehealth applications in speech-language pathology: a modified narrative review. *Journal of Telemedicine and Telecare*, 15(6):310–316, 2009.

[20] C. B. Cooper. Respiratory applications of telemedicine. *Thorax*, 64(3):189–191, 2009.

[21] R. Hazin and I. Qaddoumi. Teleoncology: current and future applications for improving cancer care globally. *The Lancet Oncology*, 11(2):204–210, 2010.

[22] A. Sun, H. Jin, R. Zheng, R. He, Q. Zhang, W. Guo, and S. Wu. *UCIPE: Ubiquitous Context-Based Image Processing Engine for Medical Image Grid*, pages 888–897. Ubiquitous Intelligence and Computing. 2007.

[23] C. de Alfonso, I. Blanquer, and V. Hernandez. Providing with high performance 3d medical image processing on a distributed environment. In *First European HEALTHGRID Conference*, pages 72–79, January 1617 2003.

[24] J. Coleman, A. Goettsch, A. Savchenko, H. Kollmann, K. Wang, E. Klement, and P. Bono. Teleinvivo: Towards collaborative volume visualization environments. *Computers and Graphics*, 20(6):801–811, 1996.

[25] J. Coleman, A. Savchenko, A. Goettsch, K. Wang, P. Bono, R. Littlefield, and C. Macedonia. Teleinvivo: a collaborative volume visualization application. *Studies in Health Technology and Informatics*, 39:115–124, 1997.

[26] M. Y. Sung, M. S. Kim, E. J. Kim, J. H. Yoo, and M. W. Sung. Comed: a real-time collaborative medicine system. *International Journal of Medical Informatics*, 57(2-3):117–126, 2000.

[27] D. G. Kilman and D. W. Forslund. An international collaboratory based on virtual patient records. *Communications of the ACM*, 40(8):110–117, 1997.

[28] B. Marovic and Z. Jovanovic. Web-based grid-enabled interaction with 3d medical data. *Future Generation Computer Systems*, 22(4):385–392, 2006.

[29] G. Triantafyllou, G. Koutelakis, C. Boukouvalas, G. Mandellos, M. Koukias, and D. Lymperopoulos. A web based telemedicine portal for centralized access to patient health records. In *5th WSEAS International Conference on Multimedia, Internet and Video Technologies*, August 17-19 2005.

[30] L. Peyton and J. Hu. A service-oriented architecture for managing privacy compliance in collaborative environments. *International Journal of Business Process Integration and Management*, 2(9):292–301, 2007.

[31] C. Caceres, E. J. Gomez, F. Garcia, J. M. Gatell, and F. del Pozo. An integral care telemedicine system for hiv/aids patients. *International Journal of Medical Informatics*, 75(9):638–642, 2006.

[32] A. Mayer and H. P. Meinzer. High performance medical image processing in client/server-environments. *Computer Methods and Programs in Biomedicine*, 58(3):207–217, 1999.

[33] H. Handels, Ch Busch, J. Encarnao, Ch Hahn, V. Kuhn, J. Miehe, S. I. Poppl, E. Rinast, Ch Romanith, F. Seibert, and A. Will. Kamedin: a telemedicine system for computer supported cooperative work and remote image analysis in radiology. *Computer Methods and Programs in Biomedicine*, 52(3):175–183, 1997.

[34] G. Falkman, M. Gustafsson, M. Jontell, and O. Torgersson. Somweb: a semantic web-based system for supporting collaboration of distributed medical communities of practice. *Journal of Medical Internet Research*, 10(3):e25, 2008.

[35] Y.J. Chen. A medical knowledge service system for cross-organizational healthcare collaboration. *International Journal of Cooperative Information Systems*, 18(1):195–224, 2009.

[36] U. Wossner, J. P. Schulze, S. P. Walz, and U. Lang. Evaluation of a collaborative volume rendering application in a distributed virtual environment. In *EGVE '02: Proceedings of the Workshop on Virtual Environments 2002*, pages 113–ff. Eurographics Association, May 30-31 2002.

[37] G. M. Olson and J. S. Olson. *Groupware and Computer-Supported Cooperative Work*, pages 583–595. Human-Computer Interaction: Design Issues, Solutions, and Applications. L. Erlbaum Associates Inc, Hillsdale, NJ, USA, 2003.

[38] K. Stark, J. Schulte, T. Hampel, E. Schikuta, K. Zatloukal, and J. Eder. Gatib-cscw, medical research supported by a service-oriented collaborative system. In *Advanced Information Systems Engineering*, pages 148–162. Springer Berlin Heidelberg, June 16-20 2008.

[39] C. H. Wang, C. C. Lee, H. C. Jiau, T. L. Yang, K. F. Ssu, and P. C. Chung. Teleconsultation enhanced via session retrieval and session scheduling capabilities. *Journal of Information Science and Engineering*, 25(4):1055–1066, 2009.

[40] X. Lu. Design and implementation of cooperative distributed dental medical information system. In *Proceedings of the 9th International Conference on Computer Supported Cooperative Work in Design*, May 24-26 2005.

[41] X. Lu. System design and development for a cscw based remote oral medical

diagnosis system. In *Machine Learning and Cybernetics 2005*, volume 6, pages 3698–3703, August 18-21 2005.

[42] J. Schulte, T. Hampel, K. Stark, J. Eder, and E. Schikuta. Towards the next generation of service-oriented flexible collaborative systems - a basic framework applied to medical research. In *Proceedings of the Tenth International Conference on Enterprise Information Systems*, pages 232–239, June 12-16 2008.

[43] H. G. Park. Collaborative medicine systems - modeling concept and architecture. pages 575–583, Berlin, Heidelberg, 2008. Springer-Verlag.

[44] P. Ganguly and P. Ray. Software interoperability of telemedicine systems: a cscw perspective. In *Seventh International Conference on Parallel and Distributed Systems*, pages 349–356, July 4 - 7 2000.

[45] Siemens Healthcare. Website of the siemens syngo suite. URL: `http://www.medical.siemens.com/syngo`, Accessed in December, 2009.

[46] A. S. Noor and M. Y. Saman. Distributed java based medical imaging informatics model. *Communications of the International Business Information Management Association*, 10:133–139, 2009.

[47] B. Ramakrishnan and N. Sriraam. Internet transmission of DICOM images with effective low bandwidth utilization. *Digital Signal Processing*, 16(6):825–831, 2006.

[48] M. Vossberg, T. Tolxdorff, and D. Krefting. DICOM image communication in

globus-based medical grids. *IEEE Transactions on Information Technology in Biomedicine*, 12(2):145–153, 2008.

[49] R. Logeswaran. *Compression of Medical Images for Teleradiology*, pages 21–31. Teleradiology. Springer Berlin Heidelberg, 2008.

[50] O. S. Pianykh. *Digital Imaging and Communications in Medicine: A Practical Introduction and Survival Guide*. Springer Publishing Company, Incorporated, Leipzig, Germany, 2008.

[51] R. Maani, S. Camorlinga, and A. N. Arnason. Parallel medical imaging transmission, 2010. Proceedings of the Society for Imaging Informatics in Medicine - SIIM 2010 Annual Meeting.

[52] R. Maani, S. Camorlinga, A. N. Arnason, and R. Eskicioglu. A practical fast method for medical imaging transmission based on the DICOM protocol. In *Medical Imaging 2010: Advanced PACS-based Imaging Informatics and Therapeutic Applications*, page 76280M. SPIE, February 17 2010.

[53] H. Zimmermann. OSI reference model-the ISO model of architecture for open systems interconnection. *IEEE Transactions on Communications*, 28(4):425–432, 1980.

[54] B. A. Forouzan and S. C. Fegan. *Data communications and networking*. McGraw-Hill Professional, New York, NY, 2003.

[55] W. W. Eckerson. Three tier client/server architecture: Achieving scalability,

performance, and efficiency in client server applications. *Open Information Systems*, 10(1):1–12, 1995.

[56] A. O. Ramirez. Three-tier architecture. *Linux Journal*, 2000(75es):7, 2000.

[57] G. R. Voth, C. Kindel, and J. Fujioka. Distributed application development for three-tier architectures: Microsoft on windows DNA. *IEEE Internet Computing*, 2(2):41–45, 1998.

[58] S. Helal, J. Hammer, J. Zhang, and A. Khushraj. A three-tier architecture for ubiquitous data access. In *ACS/IEEE International Conference on Computer Systems and Applications*, pages 177–180, June 25- 29 2001.

[59] S. Frolund and R. Guerraoui. e-transactions: end-to-end reliability for three-tier architectures. *IEEE Transactions on Software Engineering*, 28(4):378–395, 2002.

[60] D. K. W. Chiu, S. C. Cheung, E. Kafeza, and H. F. Leung. A three-tier view methodology for adapting m-services. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 33(6):725–741, 2003.

[61] Dickson Chiu, S. Cheung, and Ho fung Leung. *A Three-Tier View-Based Methodology for Adapting Human-Agent Collaboration Systems*, volume 2681 of *Advanced Information Systems Engineering*, pages 1028–1028. Springer Berlin / Heidelberg, 2003.

[62] R. M. Adler. Emerging standards for component software. *Computer*, 28(3): 68–77, 1995.

[63] D. D. Cowan and C. J. P. Lucena. Abstract data views: An interface specification concept to enhance design for reuse. *IEEE Transactions on Software Engineering*, 21(3):229–243, 1995.

[64] J. M. Daveau, G. F. Marchioro, T. B. Ismail, and A. A. Jerraya. Protocol selection and interface generation for hw-sw codesign. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 5(1):136–144, 1997.

[65] R. M. Argent and R. B. Grayson. Design of information systems for environmental managers: an example using interface prototyping. *Environmental Modelling and Software*, 16(5):433–438, 2001.

[66] G. Rossi, D. Schwabe, and R. Guimaraes. Designing personalized web applications. In *Proceedings of the 10th international conference on World Wide Web*, pages 275–284, New York, NY, USA, May 1-5 2001. ACM.

[67] D. J. Ward, A. F. Blackwell, and D. J. C. MacKay. Dasher: A gesture-driven data entry interface for mobile computing. *Human-Computer Interaction*, 17 (2):199, 2002.

[68] C. C. Chiang. The use of adapters to support interoperability of components for reusability. *Information and Software Technology*, 45(3):149–156, 2003.

[69] K. P. Andriole, M. A. Barish, and R. Khorasani. Advanced image processing in the clinical arena: issues to consider. *Journal of the American College of Radiology : JACR*, 3(4):296–298, 2006.

[70] T. J. Hacker, B. D. Athey, and B. Noble. The end-to-end performance effects of

parallel tcp sockets on a lossy Wide Area Network. In *IPDPS '02: Proceedings of the 16th International Parallel and Distributed Processing Symposium*, page 314, Washington, DC, USA, April 15-19 2002. IEEE Computer Society.

[71] D. Lu, Y. Qiao, P. A. Dinda, and F. E. Bustamante. Modeling and taming parallel tcp on the wide area network. In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Papers*, page 68.b, Washington, DC, USA, April 3-8 2005. IEEE Computer Society.

[72] E. Altman, D. Barman, B. Tuffin, and M. Vojnovic. Parallel tcp sockets: Simple model, throughput and validation. In *the 25th IEEE International Conference on Computer Communications*, April 23-29 2006.

[73] T. J. Hacker, B. D. Noble, and B. D. Athey. Improving throughput and maintaining fairness using parallel tcp. In *the 23th IEEE International Conference on Computer Communications*, March 7-11 2004.

[74] W. J. Schroeder, K. M. Martin, and L. S. Avila. *Vtk User's Guide*. Prentice-Hall, Englewood Cliffs, New Jersey, 2000.

[75] Kitware Incorporation. Visualization toolkit official website. URL: `http://www.vtk.org/`, Accessed in May, 2010.

[76] Hewlett Packards Remote Graphics Software (HP-RGS). HP-RGS official website. URL: `http://www.hp.com/workstations/software/remote/remote_graphics.html`, Accessed in December, 2009.

[77] TightVNC Software. TightVNC official website. URL: `http://www.tightvnc.com/`, Accessed in December, 2009.

[78] Kai Li's Research Group. Official website of SharedAppVnc. URL: `http://shared-app-vnc.sourceforge.net/`, Accessed in March, 2010.

[79] UltraVNC Software. UltraVNC official website. URL: `http://www.uvnc.com/`, Accessed in March, 2010.

[80] Advanced Computing and Emerging Technologies. Official website of MAST. URL: `http://www.acet.reading.ac.uk/projects/mast/index.php`, Accessed in March, 2010.

[81] R. Maani. 3D clinical visualization for enterprise hospitals. Technical report, TRLabs Annual Report, 2008.

[82] Robust Audio Tool Software. Robust Audio Tool official website. URL: `http://www-mice.cs.ucl.ac.uk/multimedia/software/rat/index.html`, Accessed in May, 2010.

[83] Kitware Incorporation. ParaView, open source scientific visualization application. URL: `http://www.paraview.org/`, Accessed in December, 2009.

[84] Slicer Community. 3D Slicer official website. URL: `http://www.slicer.org/`, Accessed in April, 2010.

[85] Eclipse Project Group. Eclipse official website. URL: `http://www.eclipse.org/`, Accessed in May, 2010.

[86] DCM4CHEE Project Group. DCM4CHEE: Open source clinical data manager system. URL: `http://www.dcm4che.org/confluence/display/ee2/Home`, Accessed in May, 2010.

[87] Creatis-Lrmn Lab. Grassroots DICOM official website. URL: `http://www.creatis.insa-lyon.fr/software/public/Gdcm/`, Accessed in May, 2010.

[88] MySQL Software Group. MySQL database official website. URL: `http://www.mysql.com/`, Accessed in May, 2010.

[89] DCM4CHE Project Group. DCM4CHE2 DICOM toolkit website. URL: `http://www.dcm4che.org/confluence/display/d2/dcm4che2+DICOM+Toolkit`, Accessed in May, 2010.

[90] Sun Developer Resources for Java. Robot Java class. URL: `http://java.sun.com/j2se/1.4.2/docs/api/java/awt/Robot.html`, Accessed in May, 2010.

[91] S. Endrullis. X window system class. URL: `https://jna.dev.java.net/source/browse/jna/trunk/jnalib/contrib/src/x11/src/jnacontrib/x11/api/`, Accessed in May, 2010.

[92] Sun Developer Resources for Java. Abstract Window Toolkit. URL: `http://java.sun.com/products/jdk/awt/`, Accessed in May, 2010.

[93] Eclipse Visual Editor Project Group. Visual Editor official website. URL: `http://www.eclipse.org/vep/`, Accessed in May, 2010.

[94] JNative Project Group. JNative website. URL: `http://jnative.free.fr/`, Accessed in May, 2010.

[95] M. Daw and J. T. von Hoffman. Guide to network bridging on the access grid. URL: `http://www.accessgrid.org/agdp/guide/network-bridging/1.0/html/book1.html`, Accessed in May, 2010.

[96] OsiriX Imaging Software. DICOM sample image sets. URL: `http://pubimage.hcuge.ch:8080/`, Accessed in January, 2010.

[97] National Electrical Manufacturers Association (NEMA). NEMA DICOM CD 1997. URL: `ftp://dicom.offis.de/pub/dicom/images/`, Accessed in January, 2010.

[98] eFile Community. DICOM sample images. URL: `http://www.merge.com/efilmcommunity/sample.htm`, Accessed in January, 2010.

[99] Wireshark Software. Wireshark official website. URL: `http://www.wireshark.org/`, Accessed in January, 2010.

[100] P. D. Manuel and J. AlGhamdi. A data-centric design for n-tier architecture. *Information Sciences*, 150(3-4):195–206, 2003.