PROTOCOL SPECIFICATION AND IMPLEMENTATION FOR ISO-BASED LOCAL AND GLOBAL PICTURE ARCHIVING AND COMMUNICATIONS SYSTEMS

by

Jiseung Nam

A Dissertation Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

DOCTOR OF PHILOSOPHY
WITH A MAJOR IN ELECTRICAL ENGINEERING

In the Graduate College

THE UNIVERSITY OF ARIZONA

1 9 9 2

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700 800/521-0600

Order Number 9225181

Protocol specification and implementation for ISO-based local and global picture archiving and communications systems

Nam, Jiseung, Ph.D.

The University of Arizona, 1992

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

PROTOCOL SPECIFICATION AND IMPLEMENTATION FOR ISO-BASED LOCAL AND GLOBAL PICTURE ARCHIVING AND COMMUNICATIONS SYSTEMS

by

Jiseung Nam

A Dissertation Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

DOCTOR OF PHILOSOPHY
WITH A MAJOR IN ELECTRICAL ENGINEERING

In the Graduate College

THE UNIVERSITY OF ARIZONA

1 9 9 2

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Final Examination Committee, we certify that we have read the dissertation prepared by Jiseung Nam entitled Protocol Specification and Implementation for ISO-Based Local and Global Picture Archiving and Communications Systems.

and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of DOCTOR OF PHILOSOPHY

Dr. Ralph Martinez	3 March 1992 Date
Dr. Larry C. Schooley	March 10, 1992 Date
Dr. Max Liu	10/13/92 Date
Dr. Olivia Sheng	March 10, 1992 Date
	Date

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copy of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

Dissertation Director	3/3/92 Date

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under the rules of the Library.

SIGNED: Jiseung Nam

ACKNOWLEDGEMENTS

I would like to express my gratitude to Professor Ralph Martinez for his invaluable guidance and support during my research period. I would also like to thank Professors Olivia Sheng, Larry C. Schooley, and Max Liu, who served on my dissertation committee, for their technical review and suggestions in this work.

I wish to express my feeling of thanksfulness to Professor Fredrick J. Hill for his encouragement during the first year in this school. Dr. Chang-won Son, Jianyi Tao, Nick Nelson, Dong-jin Lee, Ted Ozeki, Yasser AlSafadi, Sungdo Chi, Jinman Kim, and many others have been very helpful. I appreciate all their friendship and assistance.

I would like to express my appreciation to my family, Keejung and sohi, for their support and encouragement throughout my studies without which this work would not have been possible. I would also like to thanks to my parents and my wife's parents for their emotional support. This work is dedicated to the memory of my mother.

This research was supported by Toshiba Corporation. The support is gratefully acknowledged.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Picture Archiving and Communications Systems (PACS) provides an integration of digital imaging information in a hospital, which encompasses various imaging equipment, viewing workstations, database archive systems, and a high speed fiber optic network. The integration requires the standardization of communication protocols to connect devices from different vendors. The American Collage of Radiology and the National Electrical Manufacturers Association (ACR-NEMA) provides a communication standard for the interconnection of medical equipment. However, it is inadequate for networking environments because of its point-to-point nature and its inflexibility to accept new services and protocols.

In this dissertation, new communication protocols are defined based on previous experiences on PACS developments at The University of Arizona. Defined protocols are intended to facilitate the development of PACS which is capable of interfacing with other hospital information systems. Also, it is intended to allow the creation of diagnostic information data bases which can be interrogated by a variety of distributed devices. Protocol specifications are defined primarily as a combination of the International Organization for Standardization / Open Systems Interconnection (ISO/OSI) protocols and the data format portion of ACR-NEMA standard. Prototypes of defined protocols have been implemented on the SUN workstation and the VAX 11/730. Results of performance evaluation are presented to demonstrate the implementation of defined protocol.

CHAPTER 1

INTRODUCTION

Since the early 1980's, the concept of Picture Archiving and Communication Systems (PACS) has evolved to integrate digital imaging information in a hospital. PACS is a digital image information system which provides digital imaging storage, processing, and transfer. Providing a totally digital imaging information system requires the integration of various types of imaging equipment, various image viewing workstations, a large size of image databases, and a high speed network. [MAR89]

Enhancements of PACS to a conventional film image system are numerous, such as fast image transfer, easy management of images, convenient application of modern technology in image processing, better security from image loss and a cost saving in the image handling [OZE87]. But there are some obstacles, such as a large initial cost to set up the system, a potential rejection of complicated new technology from radiologists, lack of well defined standards, and the proper integration of various devices.

Following the progress of computer networks and storage device technology, PACS became realistic and important to digital radiology and medical imaging. PACS requires a high speed optical network and large image database to handle the large volume of image data, which is generated daily in a hospital [MAR90A]. Also, the standardization of communication protocols is required to integrate imaging devices from different vendors for a totally digital radiology department. The standardization includes a hardware interface, a set of software commands, a set of services provided, and a consistent set of data formats. The American College of Radiology and the National Electrical Manufacturers Association (ACR-NEMA) Working Group VI standards committee developed a point-to-point standard

for communications in digital radiology equipment in 1985 [AME89]. The ACR-NEMA standard mainly covers protocol definition for the Physical, Data Link, and Transport/Network layers. The standard does not have a network layer, and primarily defines the interface between imaging equipment and a network interface unit, or other imaging equipment. The ACR-NEMA is inadequate to use in a networked PACS environment and would be too difficult to modify [MAR90B].

During the last eight years, continuous research on PACS technology has been done at The University of Arizona and the Toshiba Corporation [MCN90B]. PACS designs have been evaluated using performance evaluation and simulation techniques. Based on previous study, papers have been written on PACS protocols at all layers of the Open Systems Interconnection (OSI) reference model [MAR91]. A new PACS communication protocol standard has been proposed to the ACR-NEMA standard committee [MAR90C]. The research described here uses the proposed ACR-NEMA protocols as a bases for development of a prototype. The implementation prototype of the proposed protocol has been done on the SUN workstation and the VAX 11/730 and is presented in this document.

1.1 Statement of The Problem

A digital image management system in a hospital consists of high speed image networks, a database archive system, imaging equipment, and viewing workstations. The image network plays a very important role in the function of PACS. The image network provides an image highway which interconnects others to transfer images. The imaging equipment includes different imaging modalities such as Magnetic Resonance Imaging (MRI), Ultrasound, Computed Tomography Scanner, Nuclear Medicine, Digital Subtraction Angiography, X-ray CT, and Image film digitizer [DAL87A]. Viewing workstations are

vary based on display resolution and depth of each pixel [HOR88]. The database archive system can be centralized or distributed.

PACS requires good communication protocol to integrate these multivendor systems. There is an ACR-NEMA protocol which has been defined since 1985, but previous study shows numerous problems in the protocol standard [MCN90B]. First of all, it does not have communication capability for a PACS networking environment, since it is defined based on point-to-point connection. Second, it cannot be used for a Global PACS environment which needs to utilize several intermediate networks, since it does not have a proper network and transport layer. Third, it does not provide room for new technology insertion, especially the technology of data link layer and physical layer which is developing fast and is essential to provide a necessary performance of PACS. Thus, the development of new protocol, which is suitable for the PACS environment, is urgent to provide the digital image management system in a hospital.

In order to realize the goals of PACS, it is appropriate to implement the defined protocol on various real systems. The implementation of defined protocol is a difficult task, since it is not a straightforward task from the specification and it should be optimized for the PACS environment.

1.2 Objective

The goal of PACS is to provide a totally digital imaging environment in a hospital. The new environment will increase the quality of image information, the efficiency of image information handling, and also the efficiency to take care of patients by physicians and radiologists. To realize PACS, the development of a suitable network protocol is essential. The main objective of this dissertation is two fields:

a. Define network protocols for PACS and Global PACS environments which will lead to product interoperability.

b. Implement a multi-stack protocol prototype which can be used as a testbed for ACR-NEMA validation.

Because of its point-to-point nature, the ACR-NEMA standard version 2.0 for digital radiology has been found to be inadequate for networked PACS environments. Therefore, the defined PACS protocol in this dissertation does not include most of the ACR-NEMA standards, except the data format portion. The specified PACS protocol follows the frame structure of the Application through Network layers of the OSI reference model. These protocol layers are the most important for Local and Global PACS environments. In fact, the defined Network layer will operate both in Local and Global PACS internetworks. These are the layers which will remain constant, independent of the digital radiology applications and communication networks. The layers are developed initially for image and textual patient information transfer. However, the flexibility of the specified PACS protocol allows for other services and protocols to be added in the future, and still be able to achieve interoperability with previous implementations. The specified protocol includes a voice service, which defines a voice traffic, integrated with image and data traffic.

The prototype implementation of the proposed ACR-NEMA protocol has been done in two separate protocol stacks: TCP/IP (DOD) and TP4/CLNP (ISO) protocol stacks. In both protocol stacks, the application, presentation, and session layers are implemented with same ISO protocol stacks.

In order to realize the goals of PACS, an accurate implementation of the defined protocol on various real systems is required. It is expected that vendors will develop Application layer services and software packages tuned to the developing needs of PACS radiology users. Also, networking technology will evolve beyond current state-of-the-art, and new

communications systems will be available at the data link and physical layers. Those networks are the Metropolitan Area Network (MAN) IEEE 802.6 standard, FDDI dual token ring network, and broadband integrated services digital network (B-ISDN) technology. Thus, the implementation of middle layers, which could be used commonly without major change, will be a valuable work for future PACS environment. The implementation of defined protocol is not a straightforward task from the specification since it does not provide programming binding. Furthermore, the interface between layers may increase the protocol overhead too much without providing a good scheme for the interface between layers. The prototype implementation of the specified PACS protocol is performed on the SUN workstation and the VAX 11/730. The underlying network is the Ethernet which is installed at The University of Arizona. The implementation is optimized to increase the PACS performance in this available environment.

1.3 Background

PACS is activated by the physician's demand to improve diagnostic capability and the available technology to support these demands. It is well known that digital image information can extract more effective diagnostic information by adding more image processing than analog image information. There is an increasing number of digital imaging modalities in hospitals today. Those digital imaging modalities are Computed Tomography (CT), Magnetic Resonance Imaging (MRI), Ultrasound diagnostic system (US), Nuclear Medicine (NM), Digital Radiology (DR), and Digital Angiography System (DSA).

The aim of PACS is a totally digital radiology department which provides a digital image archive, transmission, retrieval, and storage [DAL87]. It will integrate those digital

imaging modalities. Furthermore, it will replace a current film oriented imaging system in a hospital following the technological advance in imaging equipment, image processing, networking, and data storage device. Current development of computer and network technology can support the realization of PACS. There are several PACS scenarios, which are investigated to satisfy these goals. PACS components and scenarios are described in the following Sections.

1.3.1 PACS Components

Global PACS components include following major subsystems: a) Imaging Equipment, b) Viewing Workstations, c) Database Archive Systems, d) PACS communications Network, e) Distributed System Software, and f) Internet Gateways. A Global PACS environment consists of several Local PACS networks interconnected by a national or international backbone network, or internet. Digital images and patient information are transferred across the internet between Local PACS sites using a communications system, such as a T3-based fiber optic backbone network.

a) Imaging Equipment:

The imaging equipment performs image acquisition, generates digital images and text data from various imaging modalities, structures them in the defined network and database format, and sends them to database archive systems. Imaging equipment is developed by several vendors and the patient information and image formats are defined mostly by the ACR-NEMA Standard for Medical Imaging Systems [DAL90]. Imaging equipment generate digital images ranging in size from 512x512x8 to 4096x4096x16 bits per image. Some imaging equipment will perform on-line compression. The imaging equipment interface is a network interface unit to the local fiber optic network. By looking at the

functional operation of imaging equipment, one can see that the data traffic load on the network by each class of imaging modality is determined by several factors. These factors are the arrival rate of patients, the number of images per examination, the bit size of each image, and the number of imaging equipment in each modality.

There are various types of imaging equipments available today: X-ray, Digital Angiographic System, Digital Radiology System, Computed Radiography, Computed Tomography, Magnetic Resonance Imaging system, Ultrasound diagnostic system, and Nuclear Medicine. X-rays are the most widely used imaging modality. It generates analog images by passing X-rays into the human body as a reference energy source. Analog images can be converted to digital forms by the film digitizer. Other imaging modalities generate digital images. As an example, Digital Angiography System is an X-ray image acquisition which can amplify low-concentration intravascular iodine signals to visible levels. Images are obtained at a rate of one to two second before and after the injection of radiocontrast. Those analog images are then converted into digital format by an analog to digital converter and stored on a digital disk. Images obtained before the injection of radiocontrast are subtracted from images obtained after the injection. The image generated by this subtraction contains an iodine-containing arterial structure only.

b) Viewing Workstations

Viewing workstations provide the radiologists and referring physicians the means for displaying digitally formatted images from different modalities [ROE87]. At the viewing workstations, there are several activities done by the user. These activities are grouped in several categories: retrieving the patient's history (this includes patient's information, previous diagnosing reports on the same or related illness to the current illness, and a list of all images available for this patient), retrieving images, diagnosing the images, image

consultation, and storage of the diagnoses information. The time spent on each activity varies and is dependent on the type of illness, the diagnosing process, and the consulting radiologist. The retrieval time for a image display must be less than two seconds after the request is sent to the database archive system. Viewing workstations need high performance CPUs with large and fast memory systems to provide fast display of large images [ROB88]. Displaying images for physicians are a main function of viewing workstation. The quality of image display in the viewing workstation affects image interpretations [SEE87]. After reviewing, diagnosing and editing of the retrieved image, the radiologist might also want to consult with the patient's referring physician or another radiologist who is located at a remote site. This operation, which can be provided in viewing workstations, is called remote consultation. Each viewing workstation has a local database which can store several images. The size of the local database depends on the type of the viewing workstation and its functional operations.

c) Database Archive System:

The database archive system stores the patient's image and information in a mass storage system [SHR88]. It is responsible for retrieving images and sending them to the viewing workstations, if requested. The Database Archive System is structured in a three level hierarchical architecture [ARC87]. Level 1 maintains images that have been generated from 1 to 7 days, and are required to be delivered to the workstations within two seconds. Magnetic disks are used as storage in level 1. They feature millisecond access times and each unit has 600-900 MB disk storage. Level 2 maintains the images from 8 to 30 days. Optical juke box systems are used for storage in this level. They are write-once read-many (WORM) units. They have a slower access time, but contain more disk storage than the magnetic disks. A single juke box system can store the contents of

5-6 magnetic disks. Level 3 maintains the images beyond 1 month and up to 7 years. At this level manual optical disks are used for storage. Currently, the optical disks are manually loaded and removed by operators. The disks are categorized and stored in long term physical storage [ARC88].

Global PACS nodes include national archive centers for storage of images. The storage nodes at national archive centers are similar to the database archive systems used in a Local PACS. A distributed database system on a national basis for Global PACS has several features. The essence of a distributed database design is to distribute the logical and physical components of a single database over a communication network while keeping the distribution hidden from the users. The information on the nodes of Global PACS represents a distributed database must also be made secure in a distributed sense.

d) PACS communications Network

The main function of communications networks is to connect other subsystems for image transfers. The communication network must provide a bandwidth to transfer large amounts of images and related data in a hospital. Fiber optic networks are suitable for Local PACS networks [MCN90A]. In a fiber optic Local PACS network, network configuration is very important to get required performance. The configuration of the fiber optic network includes physical topology and media access control methods. The topologies are classified as ring, star, or bus-type. The media access control methods are classified as Time Division Multiple Access (TDMA), Carrier Sense Multiple Access with collision Detection (CSMA/CD), Token Passing, Polling and their variations [MAR87].

Global PACS can be implemented with a three level hierarchical architecture [MAR90]. The topmost level is a national, or international, backbone network, possibly based on T3 fiber lines. It interconnects regional PACS networks and single resources such as national

or international archives. Regional PACS are at the second level of the hierarchical network. These PACS will cover geographic distances such as counties, states and small countries; distances ranging from 50 km to hundreds of kilometers, or WAN long distance coverage. Metropolitan Area Networks (MANs) can meet the Global PACS regional distance requirement. Groups of Local PACS are attached to a single regional PACS. The three level hierarchical structure can handle a dynamically changing environment as well as making it easier to provide network management and policy within each level.

e) Distributed System Software

The Distributed System Software for PACS is distributed among the Imaging Equipment, Data Base Archive Systems, Viewing Workstations, and the fiber network interfaces nodes. In this case, the distributed system software is more complex since the nodes must have knowledge of the storage or retrieval location of the images and patient information. The nodes for Local PACS and Global PACS are constructed using a 7-layer OSI reference model. The Distributed System Software is embedded in each node. This software includes user services for Local and Global PACS, such as database management system, name servers, data dictionaries, and archive center information.

f) Internet Gateways

Internet gateways will eventually be required in Global PACS environments since the Local PACS will be built by different vendors. The gateways will translate Local PACS media access and protocols into the Global PACS backbone protocols, and interface the regional networks to the Global PACS. Internet gateway protocols will be used to keep

the gateway tables current on the configuration and next neighbor status in the Global PACS internet. The gateway may be used as authentication facility when establishing sessions between users on different Local PACS networks.

In addition to the PACS components, two other important issues are the management of medical images and related information and their privacy in a PACS environment. The typical managerial concerns in both cases of Local PACS and Global PACS are cost, control, and legal risks. The cost of Local PACS equipment are high and currently, most hospitals have part or none of the equipment. In the case of Global PACS, costs are even higher due to additional hardware, software purchase for networking, Local PACS database interfacing/integration, and communication cost. The control consideration applies to both Local PACS and Global PACS. It includes the usage of patient data (format, access rights consideration) and privileges to network services. Legal risks concerns primarily the security of patient data in Local PACS. Since patient data are exposed to an even wider range of users in a Global PACS environment the legal risks are even higher.

The degree of data sharing and forms of data manipulation is determined by the managerial agreement among entities involved in a Global PACS. For example, moving one patient's file from one hospital to another or (in general) data-sharing involves many difficult technical problems. The difficulty comes from the diversity of database Management Systems (DBMSs) utilized in Local PACSs in different hospitals.

In the hospital there is a wide range of services, such as preventive, diagnostic, therapeutic, maintenance and ameliorative services as well as research and medical education. All the activities done in a hospital can be divided into two categories, clinical and non-clinical. The clinical parts of services have five subparts including: inpatient, outpatient, emergency room, diagnostic conferences, and operation room. The non-clinical part includes medical

research and educational services. A portion of these parts and/or subparts are not technical, therefore they must be defined or created by the hospital administrators. For example, after the data model has been transferred to the global data model, the scale difference (conversion formulas or table of different scale) is done by administrators.

1.3.2 PACS Network Topologies

The topology of a network describes the logical and physical architecture of the PACS nodes and communications system. There are four types of well known topologies; the hierarchical tree topology, bus topology, star topology, and ring topology. Since star and ring topologies have proven useful for PACS, the detail of these two topologies are described in the following Sections.

STAR NETWORKS

In a star topology, all of the nodes in the network are connected to a single switching element as shown in Figure 1.1. Each node is connected through a network interface unit (NIU) to the central switching element.

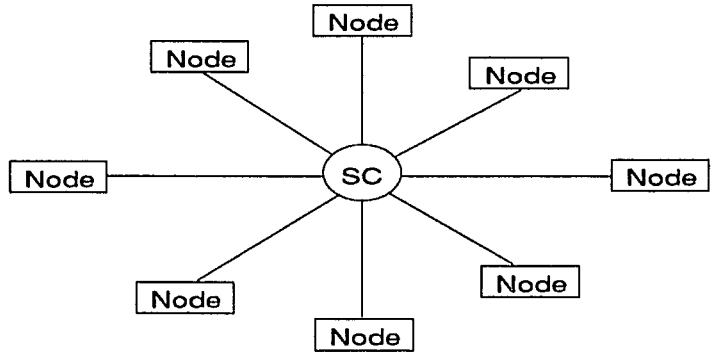

Figure 1.1. Star Network Topology.

The central switching element establishes a path between nodes. In the case of fiber optics, the central switching node is a passive star coupler, which passively broadcasts incoming packets to every node [NIS87]. The fiber optic cable is preferred as a transmission medium for PACS, because of its high bandwidth and low transmission error rate. In long-hole fiber transmission systems the bit error rate is of the order of 10^{-9} [PRO91]. A PACS network based on a star topology with fiber optic cable has several advantages. Since the star coupler is a passive device, it is protected from power loss and is thus more reliable [SAI87]. The medium access protocol is simple and the CSMA/CD protocol can be used in this topology. The disadvantage is a connection limit in the number of nodes, due to a power loss in the star coupler for packet distribution.

RING NETWORKS

In a ring topology, all of the nodes in a network are connected in a form of ring as shown in Figure 1.2. Each node is connected through a network interface unit (NIU) to the ring and is an active connection.

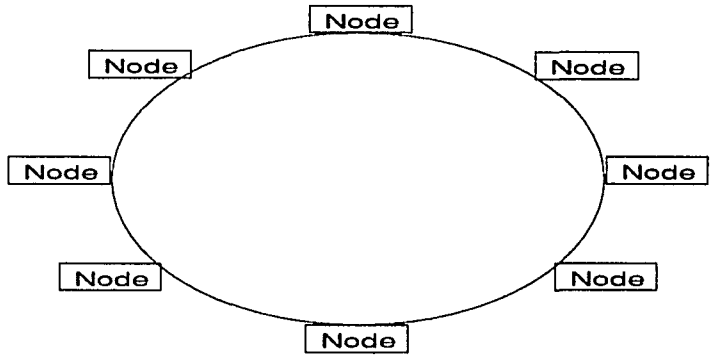

Figure 1.2. Ring Network Topology.

The data packets are transferred from node to node in a daisy-chained fashion around the ring. Each node has the ability to generate, receive, and retransmit packets. When a packet arrives in a node, the node checks the destination address of the packet. If it matches with the node's address, the node receives the data packet. Otherwise, it retransmits the packet to the adjacent node until the packet reaches the destination node. While a node is listening and retransmitting, the incoming bit stream is simply copied and regenerated to the output. Thus, the time delay in each active node is about one bit. While a node is not active, it must bypass the packet to connect adjacent nodes. There are several protocols, which have been developed for the ring topology. Depending on the protocol in a ring network, it is named to Token Ring network or Slotted Ring network. Example of a ring network is the ANSI Fiber Distributed Data Interface (FDDI) network and the IEEE 802.6 DQDB MAN Dual Ring network.

1.3.3 PACS Scenario Descriptions

Local PACS will provide a totally digital radiology department in a hospital. It will benefit radiologists and physicians with an efficient digital image archive, transmission, retrieval, and storage. It is better than the current film oriented imaging system, based on diagnostic capability, image transfer time, image loss, and cost. When patients arrive in a radiology department, images are generated from imaging equipment and stored in a local database. Radiologists review images through viewing workstations and transfer those images with patient information to a database archive system. In the database archive system, images are stored as long as necessary. Whenever images are required, those are retrieved from the database archive system along with related information.

Local PACS networks well serve a single hospital community as shown in Figure 1.3. In the Local PACS environment, the database archive system is a storage center for all the images and related data. The images, which are generated by imaging equipment or

processed at a viewing workstation, are sent and stored with related information to the database archive system. A huge amount of medical images are generated daily in a reasonable size hospital. Thus, the database archive system requires several disk storages, which can be centralized or distributed. Furthermore, the information has to be managed and shared among different users, such as physicians, radiologists, and hospital management members. Most of all, it must operate with a satisfactory fast-retrieval for users [TOM89]. Not all images can be stored in the disk storage because of their quantity, but only some of images must be stored for fast access.

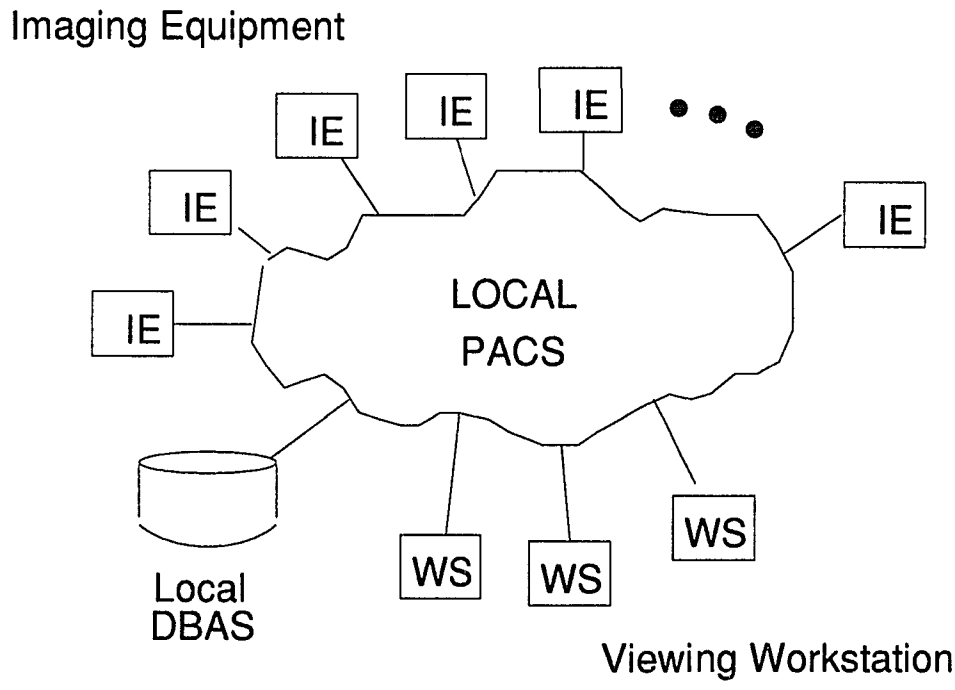

Figure 1.3. Local PACS Network.

Local PACSs need to be connected together to share their resources, which are images from the image database in other Local PACS, patient information, and other services. There will be many Local PACSs provided by many vendors. The PACS internetwork is the interconnection of Local PACSs, which may have homogeneous or heterogeneous protocols in a data link and physical layer. The internetwork of homogeneous PACS networks is relatively easy, because they are using identical communication protocols. Common addressing and routing abilities are two important and essential requirements for the internetwork of homogeneous PACS networks. But, the internetwork of heterogeneous PACS networks requires even more abilities, because of different naming, packet size, and service. All of these will be provided in the transport and network layer service.

1.3.4 Global PACS Scenarios

A Global PACS network consists of several interconnected Local PACS networks. The Local PACS are confined to hospital or organizational geographic boundaries. A Global PACS network may span a continent or several continents. The Global PACS network must have a global communications system which carries the image and data traffic between the Local PACS networks as described in Figure 1.4.

Local PACS users can store and retrieve images from geographically remote Local PACS and national archive database centers. The Global PACS communications system can consist of several communications technologies for wide area networks or backbone networks. The Regional PACS network can be implemented using several candidate technologies. Candidates include IEEE 802.6 Metropolitan area networks, FDDI token ring network, and Broadband ISDNs. The Global PACS backbone network can be implemented by Broadband ISDNs, and T1 or T3 packet switching backbone networks.

Global PACS will expand the totally digital radiology department in a local area into a geographically global area. It will provides global digital image archive, transmission,

retrieval, and storage services[]. Benefits of Global PACS are numerous. When a patient arrives in a radiology department, images are generated and stored in a local database archive system. If the patient has a serious illness and needs an emergency medical care in a remote location, the patient information and images can be retrieved from the remote database archive system through the Global PACS network. On the other hand, if the patient is sick while his doctor is on vacation or needs a care of special doctor in a remote location, then the doctor can review his image and related information through a nearby viewing workstation.

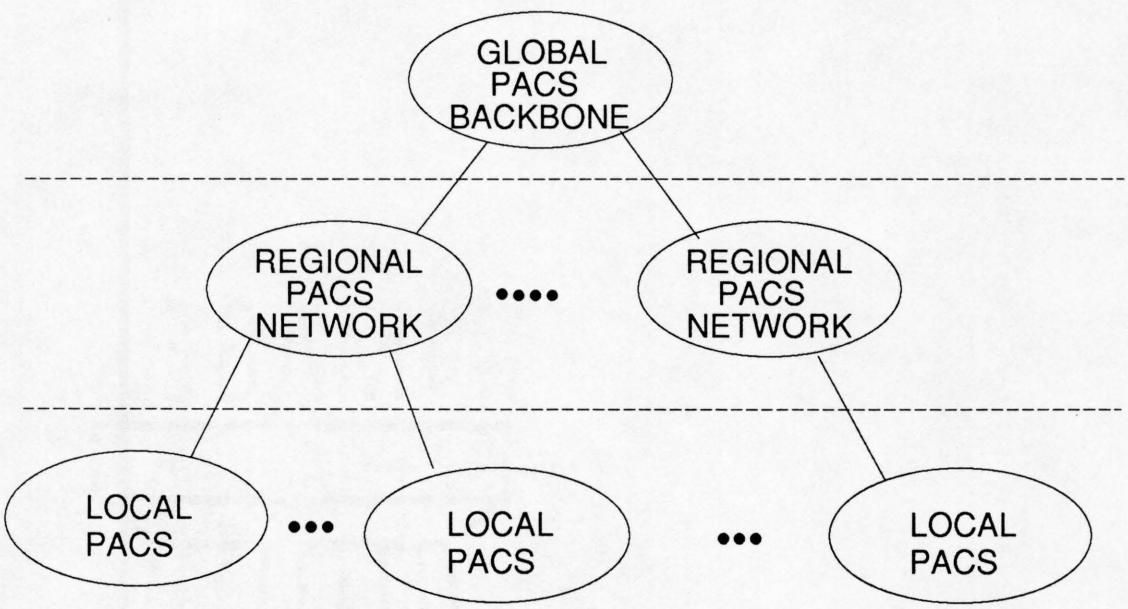

Figure 1.4. Global PACS Architectures.

1.4 Conventional Approach by ACR-NEMA

In 1985, the American College of Radiology (ACR) and the National Electrical Manufacturers Association (NEMA) introduced a first standard method for transferring images and associated information between digital diagnostic imaging modalities manufactured by various vendors. This standard publication was motivated by the introduction of various digital diagnostic modalities, the increasing application of computers, and the deficiency of a standard for data exchange between multivendor devices [ACR89].

The ACR-NEMA standard defines the hardware interface, a minimum set of software commands, and a set of data formats for an interface between an imaging device and a network interface unit or another imaging device. It defines following four layers similar to the ISO-OSI model; session layer, transport/network layer, data link layer, and physical layer. It is defined based on point-to-point connection and does not provide network layer protocol which is required for internet operation. A typical PACS network configuration and the location of Version 2.0 ACR-NEMA standard interface is depicted in Figure 1.5.

This approach generates several problems. Most of all, network interface unit must convert ACR-NEMA data format to network data format to send out image data. It is an unacceptable overhead which affects PACS performance a great deal. Second, ACR-NEMA standard interface can be a bottleneck of overall performance of the network, especially when the network is a fiber optic network. The physical layer of ACR-NEMA standard is defined as a 50 pin bus and the target data transfer rate is 8 Mbytes/second.

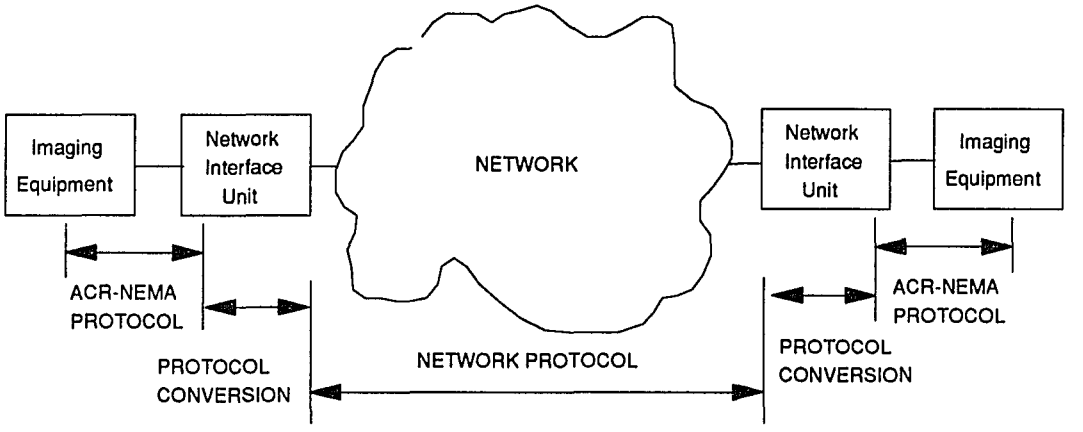

Figure 1.5. Version 2.0 ACR-NEMA Interface in a PACS Network.

1.5 New Approach for ACR-NEMA Version 3.0

The scope of the PACS protocol specification, which is proposed for ACR-NEMA Version 3.0, covers the protocol definition for the Application, Presentation, Session, Transport, and Network layers. These layers would reside in nodes on a PACS. The nodes would be imaging equipment, viewing workstations, and database archive systems, or other specialty nodes in digital radiology. The PACS networks can be Local PACS confined to a hospital community, or they can be Global PACS spread across a large geographical area, such as national or international boundaries. Figure 1.6 shows Global PACS environment interconnecting several Local PACS.

The protocol specification purposely leaves out the Data Link and Physical layers on the assumption that these are PACS network specific implementations by multiple vendors. The technology implemented in these layers may include networks with various topologies and transmission media.

As new technologies evolve for PACS networks, these may then be interfaced to the PACS Network layer using a common set of interfaces and protocols. The definition presents an initial set of Application layer protocols and user services. It is assumed that the Application layer services and functions will also evolve as digital radiology users find new methods of viewing and diagnosing medical images using PACS.

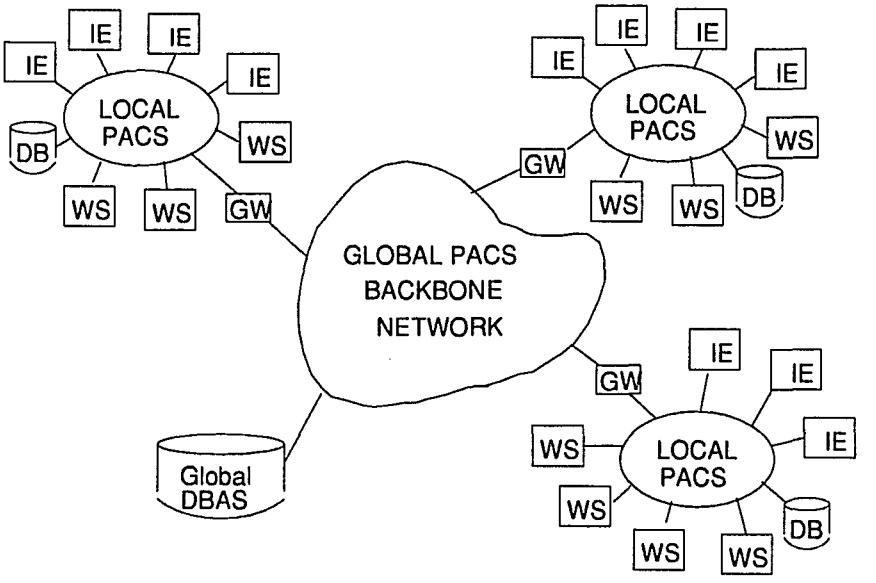

Figure 1.6. Global PACS Interconnecting Several Local PACS.

This specification of protocol contains an introduction for voice service of Local and Global PACS environments. It defines several ISO protocol terms and networking definitions required to understand the purpose and scope of the proposed PACS standard. The specified protocol may be applied on the described networking topologies for PACS and the user scenarios for Local and Global PACS environments. The details of PACS protocol for the Presentation, Session, Transport, and Network layers are presented in

Chapter 2. Figure 1.7 shows the scope of the standard according to the seven OSI layers. The PACS protocols for the Presentation, Session, Transport, and Network layers are defined based on the following characteristics and requirements:

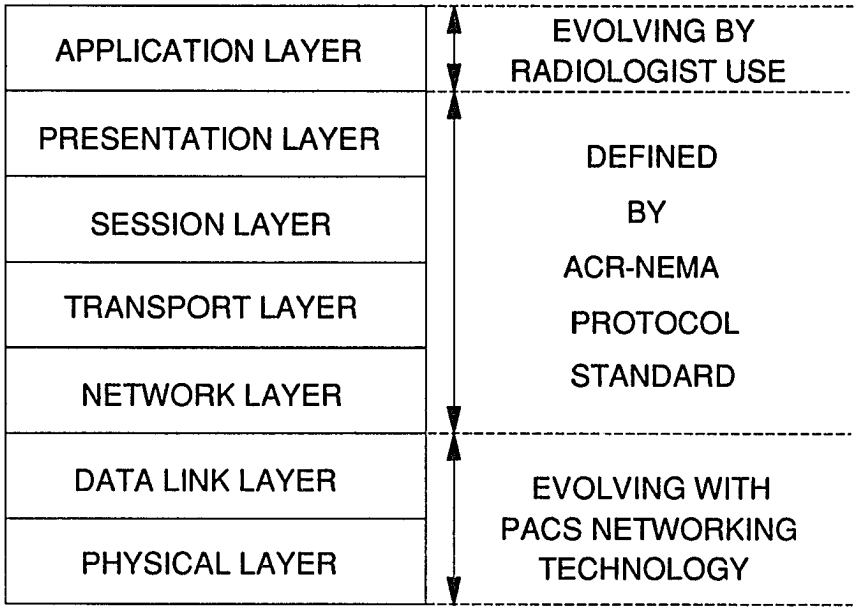

Figure 1.7. Protocol Layers Proposed for ACR-NEMA Version 3.0.

1. The size of messages can be varied from a few bytes for control, voice, or text data to several megabytes for image data.
2. Image transfer and response time for images generally requires less than 2 seconds for the first image.
3. Text and control data requires error-free protocol, but image and voice data allows some degree of error. Image data can allow bit error (less than 10^{-4} [PRO91]), but not packet loss. The packet loss of voice data is tolerable to some range.
4. The transfer of voice data is delay sensitive between inter-arrival packets.

5. The protocol must operate not only in the Local PACS environment, but also in the Global PACS environment.
6. User-to-user real time communication of image, text, voice, and overlay pointing image should be supported to serve the consultive nature of radiology.
7. Text data encryption should be allowed for security of patient information.
8. Distinct data compression should be allowed for text and image data, and the image data compression scheme is different from text data compression algorithms.
9. Since the large amount of image data will be stored in the database archive system, the database may be stored in the centralized database or the distributed database.

To satisfy these characteristics and requirements of Local PACS and Global PACS, protocol data units of each layer are defined as three different kinds: data PDU, image PDU, and voice PDU. The data PDU is designed to transfer patient information, control information, and image related information. The image PDU is designed to carry pixel data of image. The voice PDU is defined to transfer voice data. These three types of data is defined to have three separate data flows, controls, and processing in each layer. Figure 1.8 shows the three separate data flows in each layer and service access points.

The presentation layer provides a common representation of information between application entities. The representation of information includes the representation of data, the representation of image, the representation of data structure, and the representation of syntax. Using this common representation of information, the presentation entity provides a common transfer data format between peer application entities. To generate common transfer data format, it is decided to use ACR-NEMA data format for user data, not only because of its popularity in current hospital imaging network, but also because it is well defined to generate transfer data stream with little overhead.

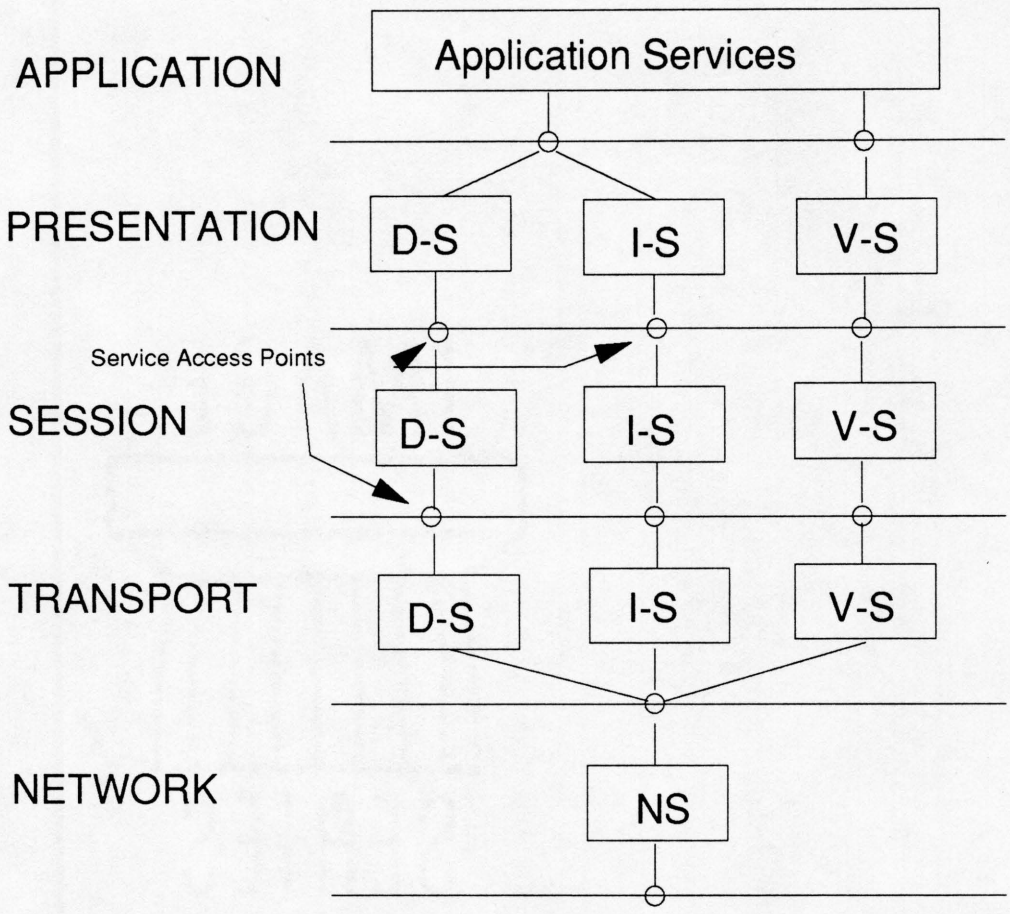

Figure 1.8. Data, Image, and Voice Services and Service Access Points.

The ISO-OSI standard has defined Abstract Syntax Notation one (ASN.1) and Basic Encoding Rules (BER) to provide the common representation of information and generate the common transfer data format. But it cannot generate ACR-NEMA data format and it requires more redundant bits to transfer user information than ACR-NEMA data format. Thus, ASN.1 and BER are enhanced to generate ACR-NEMA data format for the PACS user data. The presentation layer protocol allows that the user may send encrypted text data and compressed image data at the same time. The encryption and compression scheme

between application entities are negotiated while they are establishing session.

The voice service in application layer has two separate services. One is a control of voice transfer, the other is a delivery of voice data streams. The voice data must be delivered within 150-200 msec to preserve the integrity of a conversation [TAT89]. Thus, voice data format is not constructed using the enhanced ASN.1 and BER. The voice data is just a group of sampled voice data, typically a 40 to 400 bytes sample [CLI89], and is delivered directly to the transport layer to reduce the overhead from packetization. The control of voice transfer in the application layer uses presentation layer protocol to negotiate voice data format, which contains the size of voice data in a packet.

The session layer provides the cooperation of presentation entities to organize and synchronize their dialog and to manage their data PDU, image PDU, and voice PDU exchange. The session layer provides a service such as session connection establishment, session connection release, data PDU transfer, image PDU transfer, and control of voice transfer. When user requests to transfer data, image, and voice PDU, which requires distinguished data flow, session layer will establish three separate sessions in the transport layer. Also, the session layer provides three distinct dialog controls and synchronizations. This separation in the session service provides a powerful feature to develop conference applications and distributed database applications.

The transport layer service provides a transparent transfer of data, image, and voice PDU from sender to receiver. It provides reliable data transfer to the transport layer service user over the possibly varying reliability of the network. ISO-OSI standard defines five different classes of connection oriented transport layer protocols: TP0, TP1, TP2, TP3, and TP4. The usage of five classes of services are based on the underlying network. TP4 is selected to transfer image PDU and data PDU including control data for voice. It does a reliable data transfer between transport user entities, by recovering the lost packets,

removing the duplicated packets, and resequencing the out of order packets. TP0 is designated to transfer voice data, because voice data does not need to recover lost packets. Packets will not be duplicated in this protocol, since it does not transfer the voice packet with requesting acknowledgement. Furthermore, it must be transferred in sequence using a source routing. Thus, the other redundant protocols in TP4 are not required for voice PDU transfer. IEEE 802.6 metropolitan area networks and Fiber Distributed Data Interface network, which are candidates for PACS, support a synchronous data transfer for voice and an asynchronous data transfer for data.

The network layer grants the independence of the higher layer from underlying local area networks. Also, it provides functions which transfer data from sender to receiver over various subnetworks. The Connection-less mode Network Protocols are defined as a network layer protocol. The overall protocol specifications are defined by describing the external behavior of the system, which is inherently abstract and independent of the internal constructions. It does not contain program language binding or operating system binding to provide a flexible implementation environment.

The specified PACS protocol is fully compatible with the ISO protocol suite. The service primitives defined in the ISO documents are used to define the service interfaces between the PACS protocol layers. The PACS protocol support both the Connection-Oriented and Connection-Less services of ISO.

The specified PACS protocol is submitted to ACR-NEMA standard working group VI as a proposal for ACR-NEMA version 3.0 in 1990 [MAR90c]. Since the proposal of new standard, ACR-NEMA standard committees are working on new standard which is named as Digital Imaging and Communications in Medicine standard Version 3.0 (DICOM V3.0). The DICOM V3.0 standard is planed to be accomplished by 1992. Figure 1.9 shows current protocol structures of DICOM V3.0 standard. It has three stacks of protocol: an

ACR-NEMA version 2.0 protocol stack, a TCP protocol stack, and an ISO protocol stack. The ISO protocol stack is same as our proposal excluding the OSI Generic Upper Layer Service in the application layer. We are currently involved in defining details of the DICOM V3.0 standard as a member of ACR-NEMA working group VI. The specified PACS protocol in this dissertation is a subset of the DICOM V3.0 standard.

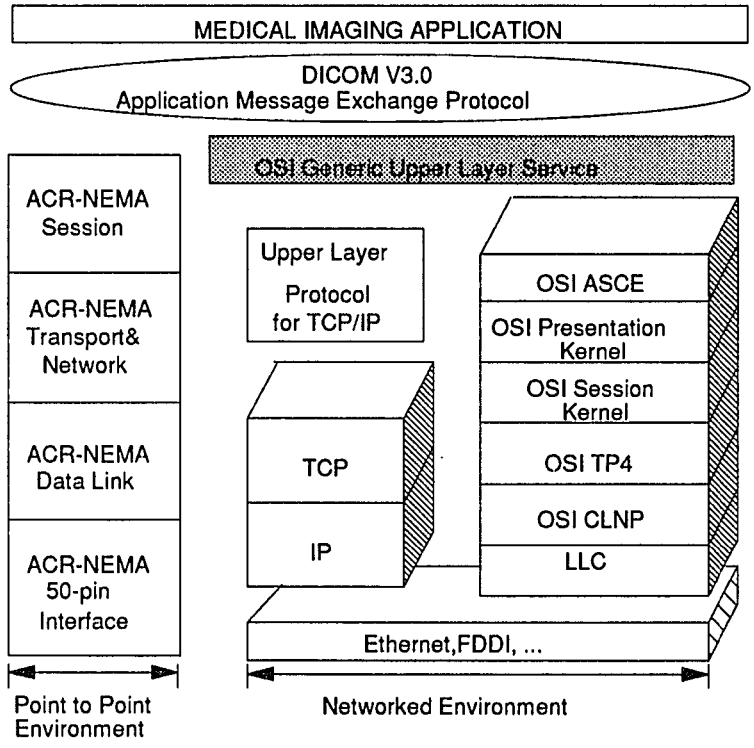

Figure 1.9. Protocol Stacks of DICOM V3.0 Standard.

1.6 Approach for PACS Prototype Implementation

The protocols discussed above are implemented as a PACS prototype in this research. The environment of implementation includes workstations, operating systems and network in the Computer Engineering Research Laboratory (CERL). SUN workstations and VAX 11/730 are used as nodes of prototype PACS. The network software in the PACS prototype is coded entirely with a C programming language. It is implemented on the top of SunOS in the Sun Microsystem and Berkeley BSD4.3-Reno operating system in the VAX. The underlying network is the Ethernet, which connects hosts at The University of Arizona.

In this research, the network programming of PACS required the use of Interprocess Communication (IPC) facilities for the interaction of two or more processes. There are different facilities to provide the interaction between processes, such as pipes, message queues, semaphores, shared memory, sockets, and Transport Layer Interface (TLI). The main goals of these facilities are to allow multiprocess programming and to provide access to communication networks. The pipe facility is a reliable, flow-controlled, byte stream that can be established between two processes on the same machine. It is used for inter-process communication between processes which are running on the same host.

The socket is the abstract object which is created within a network communication domain. It is built as an interface between several different protocols on top of the transport layer, such as UDP, TCP, XEROX NS, ISO TP4, and ISO TP0. The concept of socket was developed in Berkeley UNIX[]. The TLI in System V is the corresponding abstract object to a socket as in Berkeley UNIX. In this implementation, sockets are used to send and receive messages between two processes on different machines and platforms, mainly the socket interface is available in the 4.3 BSD-Reno Berkeley UNIX operating system and in the SunOS operating system.

The PACS protocol is implemented on the top of the TP4 socket as a required protocol stack of PSTN layers and also on the top of the TCP/IP socket for a station which does not provide ISO services. The TCP/IP side of implementation can be used as a transition protocol until migration to ISO occurs. The dual protocol stack of the PACS prototype in this research is shown in Figure 1.10.

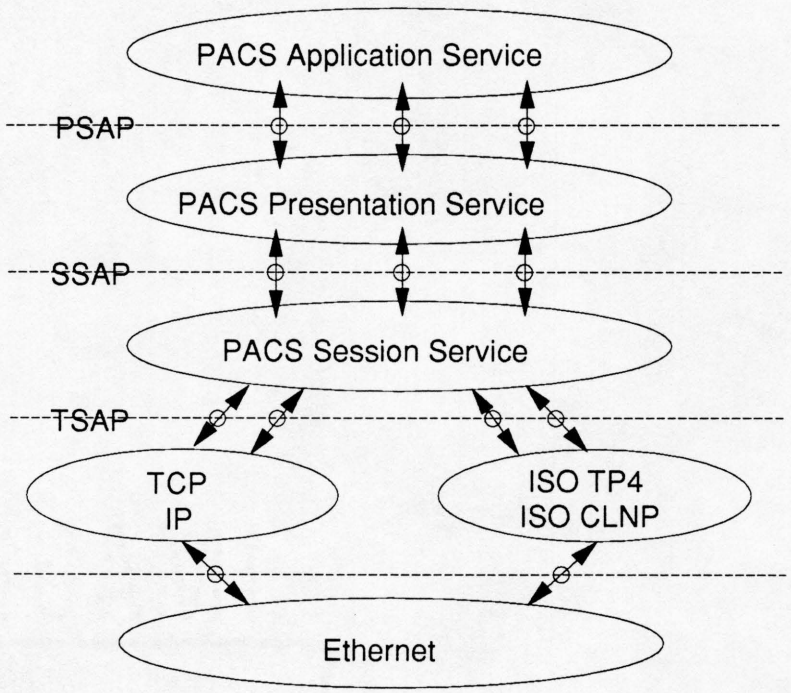

Figure 1.10. Implementation of PACS Prototype in This Research

CHAPTER 2

PACS PROTOCOL SPECIFICATION

This chapter defines the functional specifications of the protocol layers in a PACS network. The presentation and session layers are defined based on the ISO service definitions and protocol specifications. The transport layer and network layer protocols are defined as the TP4 protocol and the CLNP protocol in the ISO standard respectively.

The service primitives, which is used in this PACS protocol specification, are under the umbrella of ISO standard to increase the mutual understanding and the inter-operability between current networks. Primitives for each layer operations are classified into four types: request, indication, response, and confirm. As an example, the four primitive types of N layer are N.request, N.indication, N.response, and N.confirm.

The N.request primitive is initiated by the service user from N+1 layer to N layer to request or activate a particular service. This primitive originates at a source node layer. The N.indication primitive is used by the service provider of N layer to the service user in N+1 layer, to indicate the occurrence of a particular service. This primitive occurs at a destination node layer. The N.response primitive is used by the service user in N+1 layer to reply the service Indication from the N layer. This primitive occurs at a destination node layer. The N.confirm primitive is used by the service provider of N layer to reply to the generator of the request for a particular service that the service has been successfully performed. This primitive occurs at a source node layer. The four types of primitives are shown in Figure 2.1. Each primitive is used in conjunction with each layer and an associated service at the layer.

There are some services which have all four primitive types, such as the connection establishment service and the reset service. But, other services are defined to have only two primitives. Examples are a data transfer service, a data transfer acknowledge service, and connection release service. Figure 2.1 also shows a time sequence for the use of the four primitives. The time start at the top in this diagram.

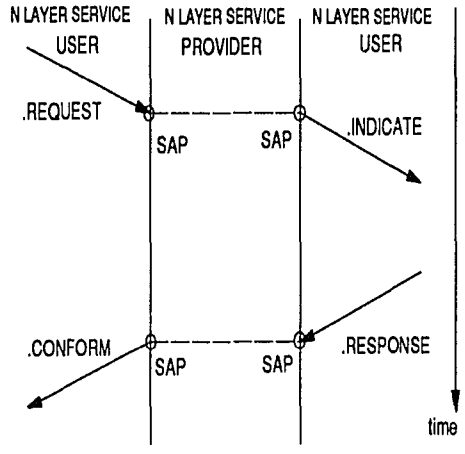

Figure 2.1. Basic Service Primitives.

A service access point (SAP) is an interface to a set of functions, which is available to the service user. A service user use a service only through a SAP at the layer immediately below [I7498]. An entity may provide services to one or more entities in the adjacent upper layer and use the services of one or more entities in the adjacent lower layer through service access points.

To satisfy the characteristics and requirements of PACS and Global PACS, protocol data units of each layer are defined as three different kinds: data PDU, image PDU, and voice PDU. Figure 2.2 shows the three separate data flows in each layer and their connections.

The data PDU is defined to transfer patient information, control information, and image related information. The image PDU is defined to carry pixel data of image. The voice PDU is defined to transfer voice data. Since each data type has different characteristics and requirements, these types of data are defined to have three separate data flows, controls, and processing in each layer.

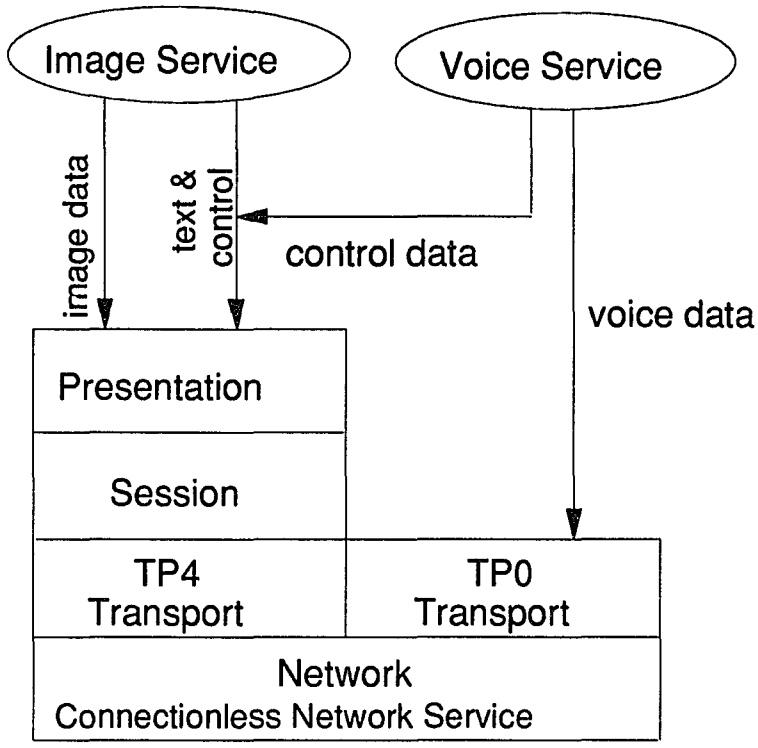

Figure 2.2. Flow and Control of Three Types of Data.

The presentation layer provides a common representation of information between application entities across a network. The session layer provides the cooperation of presentation entities to organize and synchronize their dialog and to manage their data PDU, image PDU, and voice PDU exchange. The transport layer service provides a transparent transfer of text, image, and voice data from sender to receiver. The voice service in the application layer has two separate services. One is a control data of voice transfer, the other is a delivery of voice data streams.

Since the voice data need real time transmission and do not need data format conversion,

voice data streams are delivered directly to the transport layer to reduce the overhead from packetization. Only the control data of voice transfer in the application layer use presentation layer protocol to negotiate the size of voice data in a packet and the control of voice session and dialog. The international standard ISO 8072 and 8073 defines five different classes of connection oriented transport layer protocols: TP0, TP1, TP2, TP3, and TP4. TP4 is selected to transfer data PDU, image PDU, and control data for voice as a form of data PDU. TP4 does a reliable data transfer between transport user entities, by recovering the lost packets, removing the duplicated packets, and resequencing the out of order packets. TP0 only provides segmentation and reassembly and is used to transfer voice data, because voice data does not need to recover lost packets. The network layer grants the independence of the higher layer from underlying local area networks. Also, it provides functions which transfer data from sender to receiver over various subnetworks. The Connection-less Network Protocols (CLNP) are defined as a network layer protocol.

2.1 Protocol Data Units in Each Layer

The Protocol Data Unit in each layer contains a header, which is the Protocol Control Information of each layer. The PDU of N+1 layer (N+1 PDU) is passed down to N layer and is bound with N_PCI to construct N_PDU. The PCI of each layer contains the fixed part and variable part as shown in Figure 2.3.

The fixed part contains frequently occurring parameters including command code of the layer. The PCI structure of each layer is defined separately following their requirements. Generally the fixed part includes a Length Indicator and Layer Protocol Identifier. The Length Indicator contains the binary number that indicates the PCI length including the fixed part and variable part. The Layer Protocol Identifier identifies the different protocol

that exist in the layer. The variable part contains less frequently used parameters. The variable part may contain one or more parameters. The size of parameter in the variable part is not fixed. Each parameter is structured with a parameter code, a parameter length indication, and a parameter value as shown in Figure 2.4.

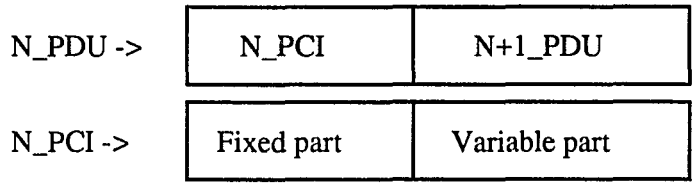

Figure 2.3. Protocol Control Information Structure

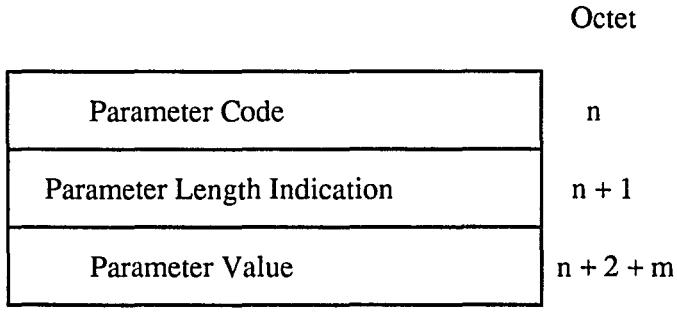

Figure 2.4. Structure of Variable Parameter.

The parameter code indicates the meaning of parameter value and the parameter length denotes the parameter length as a number of octet.

2.2 Application Layer Services

The application layer is the highest layer of the seven layer model, which provides a interface between user services and the layers of the network. Although the user services required at the application layer are expected to evolve, examples of application services are discussed here. Radiologists, examining physicians, system managers, and technicians are the system users at the application layer. They require several operations at PACS viewing workstations and imaging equipment. The functions considered in the application layer are image storage, image retrieval, database management and query, PACS name server, database access control, user access control, image diagnosis, and image and patient browse. Examples of the application layer consist of four group of services to provide these application functions, as shown in Figure 2.5.

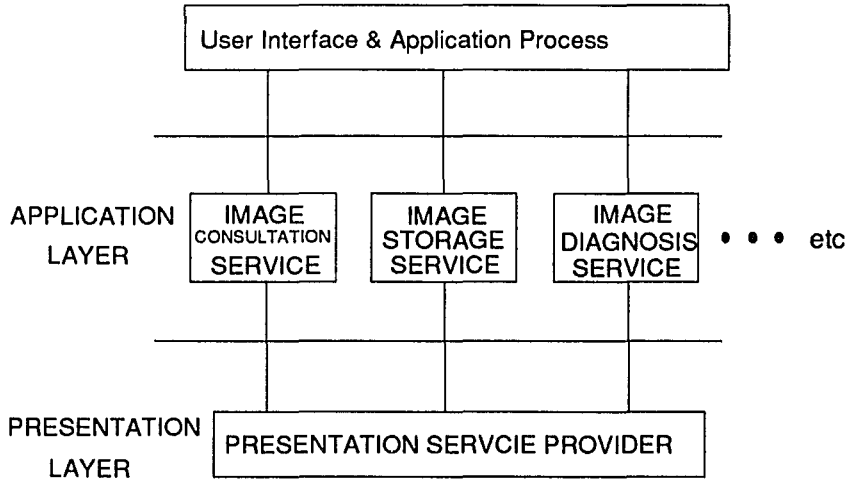

Figure 2.5. Application Layer Services

The Image Diagnosis Service provides a service related to the diagnosis of image; a patient browsing, an image retrieval function, a diagnosis of image, and a diagnosis result data storage. The Image Consultation Service provides a service of consultation; an image distribution, a voice distribution, a related image information distribution, and a patient information distribution. The Image Storage Service provides a database handling service, such as database management and query, and database access control [MAR88]. These services are utilized by a PACS user interface and application process, which varies from vendor to vendor. Since a user of the PACS generally is not a computer engineer, the user interface service will require extensive service to provide the transparency of the underlying computer network layers for those users. Four group of services are supported by the following application layer service entities.

2.2.1 Image Storage

Image storage is a function which transfers an image from an application entity of an image source node to an application entity of the database archive system. The image storage command is activated by a user through a user interface in the application entity of an image source node, and sends an image with image related information to the application entity of the database archive system.

2.2.2 Image Retrieval

Image retrieval is a function which transfers an image from an application entity of the database archive system to an application entity of a viewing workstation. The image retrieval command is activated by a user through a user interface in the application entity of a viewing workstation, and retrieves an image with image related information from an application entity of the database archive system.

2.2.3 Database Management & Query

A part of database management may be implemented as a function of the application layer. The database in PACS is expected to have a large size and can be distributed. Thus, it is required to have some of the functions related to database management and query. The basic functions required for database management and query are Find function, Move function, Change Directory function, Make Directory function, and Remove Directory function.

2.2.4 PACS Name Server

The PACS needs a name server which will be used for the common acknowledgement of textural names in different Local PACS. Each processing element (PE) in the network needs a network service access point (NSAP) address in order to open a connection or send data. The NSAP addresses are unique numbers allocated to PEs. Since it is not convenient that users remember the NSAP address as a number, the name server is provided in new ACR-NEMA standard. It translates a PE name as a corresponding address number using a look up table. The name servers are distributed across all the inter-network instead of a single central one and each name server has its own domain with hierarchy. Thus, each name server does not need to have a look up table that lists all of the other PE names and corresponding numbers. Instead, the name server keeps track of PE names and the corresponding NSAP addresses.

2.2.5 Image Diagnosis and Browse

Image diagnosis is one of the elementary purpose of PACS. When physicians or radiologists are diagnosing a patient image, they must retrieve the image along with patient information and image information. Then they will do the image diagnosis and will save

the result of diagnosis. If there is no image processing or image modification, only the result of diagnosis will be saved in the database. The image diagnosis service will provide all appropriate services for PACS user, and will increase the efficiency of communication.

The image and patient browsing is the action which searches for a specific image or patient in the local database or in the remote database archive system. When a referring physician or radiologist tries to diagnosis a film, he will search for the image and patient information. If the requested information is located in the local database, it will be much easier. But the requested information is generally located in the remote database archive system, because of huge amount of data in hospital. The image and patient browsing for the remote database archive system will require unique communication protocol. The image and patient browsing requires the method to identify a certain patient and image. Four identifications are defined for a patient and three types of images, such as patient identification, original image identification, modified image identification, and composite image identification. Each of these is a set of data elements, which are strictly following ACR-NEMA format and described briefly in Chapter 4. The patient identification is a set of patient name, patient ID, patient birthday, and other patient IDs. The original image identification is a set of station ID, study date, study ID, series, acquisition, and image number. The modified image identification is a set of modifying device ID, modified image ID, and modified image date. The composite image identification is a set of modifying device ID, modified image ID, and modified image date.

2.2.6 Voice Service

The integration of digital voice in a PACS offers several benefits, including reduced system cost through sharing of transmission resource, flexible internetworking among Local PACSs utilizing different transmission media and media access control, and

enhanced service for users providing image conference.

The characteristics and performance requirements for voice transmission are quite different from data transmission. The size of data varies from a few bytes to several mega bytes. Data transmission requires strict error control and recovery process, but real time data delivery is not of primary importance. Voice transmission requires strict real time delivery, but does not require strict error control and recovery process.

Voice transmission requirement is inherently robust from error. Thus, speech can be reconstructed at the destination user with acceptable quality, when the voice packet loss is less than 1% [CLI83]. Voice communications last for a few minutes to hours in the case of a conference, and 60% to 65% of the communication time the channel remains idle. Thus, significant channel bandwidth savings for packet voice can be achieved by transmitting packet during talkspurts and no packet generation during silence. Voice data is generated by a digitization with an interval of 125 microseconds. Voice packets are constructed by concatenating each 8 bits of sampled data in the duration of a 5 to 50 millisecond speech. The choice of packet size is also influenced by limitations on network throughput in packets/sec and packet overhead by the header size of many existing networks. The maximum recommended time delay allowed in delivery of voice packet is typically 150 to 200 millisecond. For example, 5 millisecond of 64 Kbits/sec speech is represented by 320 bits. Thus, voice data transmission in application is directly connected to TP0 protocol, and only control data transfer is connected to the ISO stack defined for data and image transmission for reliable transmission. This scheme provides the reliability of voice communication control and less overhead of voice data transmission.

2.2.7 Folder Concept

A folder is a means of associating sets of related Data Sets. All concepts and definitions, which are developing in current ACR-NEMA standard working group VI, are supported to define the underlying layers, especially in the presentation layer. Image films are stored using a holder at a film library. There are two types of folders: folder by reference and folder by value. A folder by reference is a list of Data Set Identifiers, types, locations, and length. A folder by value is a Data Set that contains the entire contents of associated data sets. Holders are containers that hold a group of image films related to a certain patient. For several reason, the films are managed as a form of folder. Film management includes folder transaction, folder inquiry, folder loan, and folder of interesting cases. Using holder not only simplifies of image handling and keeping, but also referring physicians or radiologists are frequently diagnosing film with other related images in folder. Even though it is not required for PACS to store films at the same physical location, the applying folder concept gives a user several advantages for handling image data. The advantages are not only structured image handling, but also reduced overhead for storing a patient images and related informations. The concepts of folders have been well defined and accepted in ACR-NEMA working group VI. It is in the fine tuning stage to complete the definitions.

2.3 Presentation Layer

The presentation layer provides a common representation of information between application entities [I8822]. The representation of information includes the representation of data, the representation of data structure, the representation of image, and the representation of syntax. This layer also provides PACS network security and

management, and image compression and decompression. The detailed specification of presentation service access points and presentation protocol data units are described in Section 3.4.

2.3.1 Image Format Conversion

The image format in each imaging equipment can be different; thus there is a need of a common image format which will be used between two different application entities (possibly between application entity of the imaging equipment and application entity of the database archive system). The presentation layer provides image format conversion as a common image format while the application entities are transferring images. The common image format is selected by negotiation between the presentation entities.

The negotiation of image format implies the communication to determine what conversion is needed and which side needs image format conversion. In case that both sides of the image format are equal, the image format conversion function is not needed. But if they are different, then presentation layer services take care of the mis-match of image format between application entities.

Images generated from different image modalities does not have common image format and size. The projection x-ray images are currently recorded on a range of film size from the most common 14" x 17" to 8" x 10", both for viewing and for storage [SHI89]. These film images can be converted to the digital images by means of film digitizer. The digital images created by this method are vary in density range to what is captured on film and could form an image of 2k x 2.5k x 10 bits. The images formed by CT, MRI, and US are digital from the generation. But, the lack of a standard interface across imaging system remains a major obstacle for a common method of archive and display. As an alternative

solution, images can be retrieved by the digitization of the video signal from CRT display channel. The data format of CT, MRI, and US are generally raw data with 512 x 512 x 12 bits, 256 x 256 x 12 bits, and 512 x 512 x 8 bits respectively [DES89].

2.3.2 PACS Network Security & Management

The presentation layer may support network security and management function. The need to provide security features in PACS subsystems is important to the operation and acceptance of PACS technology by the medical imaging community. Patient information is private and confidential and must be secure to insure the patient's privacy rights. Patient information is textual and includes attributes such as name, ID number, social security, birthday, marital status, medical history, medication history, and diagnosis information. The patient information accompanies the digital images acquired during a patient's several visits to the hospital or doctor's office. The digital images contain visual information pertaining to the patient diseases and health history. Patient and image information in the clear leaves a hospital administration open to corruption of the data and can lead to severe health problems and subsequent malpractice litigation. The need to protect this information is becoming more important in these days of high technology in the medical sciences which can lead to social problems.

There are several reasons for security in a Local and Global PACS environment. Data or image loss and corruption may cause misdiagnosis or may cause the patient to undergo another dose of radiation. Global PACS presents even more security threat as images are transmitted over transcontinental fiber optic communication lines. Computer and network security issues for Local and Global PACS can classified into three areas 1) user access control security, 2) database archive access control security, and 3) communications network security. These issues are briefly summarized as follows.

User access control security protects against unauthorized use of imaging equipment, workstation, database systems, and other PACS components. In order to accomplish access control, a user's identity must be authenticated. This can be done by methods such as passwords, handshaking schemes or digital signatures or by methods providing physical access control, such as key locks, ID badges or smart cards. Users of PACS facilities include radiologists, physicians, technicians, systems managers, and support personnel. The level of user access control must be identified and can be provided with the PACS environment and in the application layer of communication protocol.

Database archive system security is required to protect patient information and image data from unauthorized users in the storage systems of PACS. The Database Archive System (DBAS) must provide both data confidentiality and data integrity. A popular method of securing a database management system is to provide a security shell around the database. This shell acts as an interface between the user and the PACS database. When a remote user wants to use the system, the user's identification will be looked up in the name server table, where there is listing of the user's identification, and the corresponding host machine. Upon verification, the security shell then allows the remote host to use the system. For data integrity, there will also be a access permission level listed. In this way the security shell can verify the identification of the remote user and assign a access permission level. The security level can be taken to the schema and file level, if required. An important part of database security is protection of the data and images in a distributed system such as Global PACS. The security shell can be provided in the server side of the application layer.

Communications network security is required to secure patient information and image data as it traverses the local and global communications network. The methods to do so include physically securing the network cables, implementing protocols which detect

network intruders, and encryption of the data before transmission. Although fiber optic communications lines provides an inherent security feature, the level of communications network security must be determined. The network security function using encryption / decryption algorithm can be provided in the presentation layer jointly with key distribution management.

2.3.3 Image Compression / Decompression

The presentation layer may support image compression and decompression function. The common image compression and decompression algorithm is selected by the negotiation between the presentation entities. The negotiation of algorithm is the communication to determine what algorithm is used for compression and decompression. It may be possible to store the compressed image in the database archive system and decompress the image when it is retrieved.

Image compression techniques can substantially reduce the volume of data that must be processed in image transfer and storage [BRU88]. There are numerous image compression methods today. In general, two methods of image compression are error-free and irreversible. Error free image compressions can compress radiological images with compression ratios of 2:1 to 3:1 [COH90]. Among irreversible image compressions, the Discrete Cosine Transform technique can achieve compression ratios of 10:1 to 20:1 while retaining acceptable diagnostic quality in images reconstructed from the compressed data [KEL88].

Any specific techniques are not proposed in this protocol specification. Providing the negotiation of algorithm will support any image compression schemes which is widely used in the PACS community. There are three major image compression techniques which has been proposed to the ACR-NEMA standard committee since 1988. These are

Differential Pulse Code Modulation, Discrete Cosine Transform, and S-Transform [BLU88]. Differential Pulse Code Modulation is the representation of an image by the difference of the pixel value and a predicated value based on previous pixels. Discrete Cosine Transform is the representation of an image by the highly correlated data with fewer coefficients. S-transforms, or pyramid transforms, are linear transforms with non-sinusoidal base functions which decompose an image into an ordered set of picture matrices of successively coarser spatial resolution.

2.3.4 Abstract Syntax

The abstract representation of data types provides virtualization of data types in a network, because it can describe each data type with independence from a machine oriented structure. To allow abstract representations of data types, the abstract syntax notation one (ASN.1) is defined in ISO. But, we need to develop enhanced abstract syntax notation for PACS, other than ASN.1 in ISO. Reasons of the requirement for new notation are as following:

a. Current ASN.1 language and basic encoding rule (BER) cannot generate ACR-NEMA transfer syntax, because the transfer syntax generated by ASN.1 and BER is different from ACR-NEMA transfer syntax.

b. ISO specification allows the possibility of using other notations and encoding rules.

c. ACR-NEMA standard has not only well defined transfer syntax for PACS, but is also well known to PACS users.

d. Developing the level of abstraction of such definitions closer to semantics of PACS will allow users more flexibility.

e. PACS requires different primitive elements and construction techniques.

f. ACR-NEMA transfer syntax has less transfer data size than transfer syntax generated

by ASN.1 language and BER.

The new abstract syntax notation for PACS is defined by adding new abstract syntax notations on ASN.1. These new abstract syntax notations and corresponding encoding rules are added to generate ACR-NEMA transfer syntax, which is going to be used to generate user data stream for PACS.

2.3.4.1 ASN.1 in ISO

ASN.1 is defined to transfer structured data between communication systems in ISO [I8823]. ASN.1 is used to describe each data type without regarding the actual data structure that depends on a particular system. When the units of data to exchange were simple, it was not needed to use abstract notation, but since the units of data by application layer protocols becomes arbitrary and complex, it is required to add structure to the units of data with common acknowledgement. Especially, this provides a virtual data types that is independent from machine oriented data structure. ASN.1 includes two separate specifications, such as the ASN.1 language and the BER. The ASN.1 language is defined to describe data types and structures. The BER is defined to generate a transfer syntax (a stream of octets for transmission) from the ASN.1 notation.

ASN.1 defines the following simple primitive types:

a. Boolean : a TRUE or FALSE.

b. Integer : a cardinal number and the range is unbounded.

c. Bit string : an ordered set of zero or more bits.

d. Octet string : an ordered set of zero or more octets.

e. Null : a place holder which is used when the optional element is absent.

Constructor primitive types are defined to compose complex data structures, as following:

a. Sequence : an ordered list of data elements.

b. Sequence of : an ordered list of same data elements.

c. Choice : any one data element in a list of data elements.

d. Component of : a simple inclusion of all of the elements of the sequence.

e. Set : an unordered list of data elements.

f. Set of : an unordered list of same data elements.

g. Any : any one of all data types, simple or constructor types.

Next example shows construction of data structure using those primitives.

```
ExampleMessage ::=
    SEQUENCE {
      version
        INTEGER,
      hospitalName
        OCTET STRING,
      error-num
        INTEGER {
          noError (0),
          nameMismatch (1),
          badVersion (2)
        }
    }
```

2.3.4.2 Syntax Notation for PACS

Syntax Notation for PACS (SN.PACS) has been defined to transfer structured data between communication systems in PACS. SN.PACS is a super-set of ASN.1. It includes all primitive types in ASN.1 and contains new primitive types required for PACS. These new primitive types are defined not only to describe each data type independent from a particular system in PACS, but also to generate a transfer syntax following ACR-NEMA standard. The units of data to exchange for image transfer are well defined in a transfer syntax definition of ACR-NEMA standard, but the complexity of data requires Abstract Syntax Notation for a common acknowledgement and a flexibility of application layer protocol development. Especially, this provides a virtual data type that is independent from machine oriented data structure. A Encoding Rule for PACS (ER.PACS) is also

defined to generate transfer syntax from SN.PACS notation.

SN.PACS defines ten more constructor primitive types; Group set and nine groups. These are defined to compose PACS data structures, as follows:

a. Group set : an unordered list of standard groups.

b. Identification group : an unordered list of identification group data elements.

c. Patient group : an unordered list of patient group data elements.

d. Acquisition group : an unordered list of acquisition data elements.

e. Relationship group : an unordered list of relationship group data elements.

f. Image presentation group : an unordered list of image presentation group data elements.

g. Text group : an unordered list of text group data elements.

h. Overlay group : an unordered list of overlay group data elements.

i. Pixel group : an unordered list of pixel group data elements.

SN.PACS includes the following ISO simple primitive types for each group data element.

a. Boolean : a TRUE or FALSE.

b. Integer : a cardinal number and the range is unbounded.

c. Bit string : an ordered set of zero or more bits.

d. Octet string : an ordered set of zero or more octets.

e. Null : a place holder used to optional element is absent.

SN.PACS defines the following PACS simple primitive types for each group data element. These primitive types belong to each group and cannot be used in different groups.

The Primitive types of Identification group :

GroupLength
RecogCode
Studydate
StudyTime
Modality
Manufacturer

InstitutionID
ReferPhysian
NetworkID
StationID
Diagnosis
Comments

The Primitive types of Patient group :

GroupLength
PatientName
PatientID
PatientBirthday
PatientSex
OtherPatientIDs
OtherPatientNames
PatientMaidenName
PatientAge
PatientSize
PatientWeight
PatientAddress
InsurancePlanID
PatientMotherMaidenName
Comments

The Primitive types of Acquisition group :

GroupLength
ContrastBolusAgent
ScanningSequence
Radionuclide
CineRate
SliceThickness
KVoutputXray
CountAccumulated
RepetitionTime

The Primitive types of Relationship group, Image presentation group, Text group, Overlay group are not yet defined. Those will be defined following ACR-NEMA data format definition which is in a transition state.

The Primitive types of Pixel group :

GroupLength
PixelData

The next example shows construction of data structure using those primitives.

```
ImageDataSet ::=
    GROUP SET {
      PATIENT GROUP {
          GroupLength,
          PatientName,
          PatientID,
          PatientBirthday,
          PatientSex,
          PatientAge,
          InsurancePlanID,
          },
      ACQUISITION GROUP {
          GroupLength,
          ScanningSequence,
          Radionuclide,
          },
      PIXEL GROUP {
          GroupLength
          PixelData
          }
    }
```

2.3.5 Data Definitions

Data is information that has been prepared, often in a particular format, for a specific purpose. The common data format is defined to integrate various data formats from different imaging equipment and viewing workstations. Data consists several groups based on the type of information. The definition of groups strictly follows definitions in ACR-NEMA standard 300-1988. The commend group is not included, since it is defined in each layer as a service access point. Standard groups are Identification, Patient, Acquisition, Relationship, Image Presentation, Text, Overlay, and Pixel Data.

Standard Groups	Group Number
Identification	0008H
Patient	0010H
Acquisition	0018H
Relationship	0020H
Image Presentation	0028H
Text	4000H
Overlay	6000H-601EH
Pixel data	7FE0H

An Identification group provides unique image identification parameters to the user. A Patient group provides information related to the patient. An Acquisition group provides information related to the image acquisition equipment and imaging procedure. A Relationship group provides information related to the location of image within the patient and in relation to other associated images. An Image Presentation group provides information related to the manner in which the image can be presented or displayed in a consistent and reproducible manner. A Text group is ASCII text. An Overlay group provides information related to overlay pixel data associated with the image. A Pixel Data group provides information of Image pixel data. Each group is subdivided into data elements that contain individual segments of information. The components of the data element are contained in four fields: a group number, a data element number, a length, and

a value. The group number field and the data element number field have two bytes each. The length field is used to define the length of a value field as a number of bytes and consists of four bytes. One byte consists of eight bits.

2.3.5.1 Patient Data Elements

Patient data is the information which is related to a patient including patient name, sex, age, patient identification number, size, weight, address, and etc. Each of the patient information is defined as a patient data element, which is an element of patient group. Patient element is constructed with a group number, a data element number, a length, and a value. The group number of patient data is defined as 0010H and the element number defined in ACR-NEMA is shown in Table 2.2.

Table 2.2. Patient group elements

Group n	Element n	Name
0010	0000	Group Length
0010	0010	Patient Name
0010	0020	Patient ID
0010	0030	Patient Birthday
0010	0040	Patient Sex
0010	1000	Other Patient IDs
0010	1001	Other Patient Names
0010	1005	Patient's Maiden Name

0010	1010	Patient Age
0010	1020	Patient Size
0010	1030	Patient Weight
0010	1040	Patient Address
0010	1050	Insurance Plan ID
0010	1060	Patient's Mother's Maiden Name
0010	4000	Comments

2.3.5.2 Acquisition Data Element

The acquisition data is the information which is related to a image acquisition device and image archiving procedure. The information includes the contrast, the scanning sequence, the imaging frequency, the distance from source to patient, the exposure time, the exposure rate, and etc. Each of acquisition information is defined as an acquisition data element, which is an element of acquisition group. The acquisition element is a set of a group number, a data element number, a length, and a value. The group number of acquisition data is defined as 0020H and some of the element numbers defined in ACR-NEMA are shown in Table 2.3.

Table 2.3. Acquisition group elements

Element n	Name
0000	Group Length
0010	Contrast/Bolus Agent
0020	Scanning Sequence
0030	Radionuclide
0040	Cine Rate
0050	Slice Thickness
0060	KV output of the x-ray generator
0070	Count Accumulated
0080	Repetition Time

2.3.6 Image Data Set

Data set type of IMAGE is defined as a data set containing Pixel Data group and other related data groups. The related data groups, that must be contained in image data set, are Identifying group, Patient group, Acquisition group, Relationship group, and Image Presentation group. The Identifying group is defined for information that provides unique image identification parameters to the user. The patient group contains information related to the patient. The acquisition group is a set of information related to the image acquisition

device and imaging procedure. The relationship group contains information relating the image to the location within the patient and to other associated images. The image presentation group is a set of information related to the manner in which the image can be presented or displayed in a consistent and reproducible manner.

2.4 Session Layer

The session layer provides the cooperation of presentation entities to organize and synchronize their dialogue and to manage their data and image exchange [I8326A, I8327A]. The session layer supports the presentation layer with connection oriented mode and connection less mode. A presentation entity can access another presentation entity by establishing a session connection or several session connections simultaneously. The session layer provides a service such as session connection establishment, session connection release, data transfer, image transfer, data transfer with acknowledge, image transfer with acknowledge, expedite data transfer, and exception reporting. The session service defines not only the constraints and interactions of using synchronization points for synchronized exchange of data, but also their interactions and the precise state that the connection will be set to when issued and when resynchronization occurs. The usefulness of this to PACS applications is dependent on how the application standard chooses to use the service, but all the required services are considered and included to maximize the PACS in a hospital environment. The use of each of these services will be negotiated at connection establishment time in connection oriented mode. The detailed specification of presentation service access points and presentation protocol data units are described in Section 3.5.

2.4.1 Synchronization and Dialogue Control for Data Transfer

The session layer provides the service of organized and synchronized exchange of data between corresponding session service users [I8326B, I8327B]. Two types of dialogue, the duplex and half duplex, exist as a data transfer techniques. The half duplex mode allows the data transfer in one direction, but the duplex mode allows the data transfers in two directions at the same time. The dialogue control and synchronization scheme is different depending on the type of dialogue. Also, the requirement for the complicated PACS scenario is a well structured synchronization service. The synchronization services of the session layer are providing checkpoints to the session service user while they are transferring image/data units. The checkpoints are called as synchronization points, which are used to insert points into the data transmittion. Each synchronization point is distinguished by a serial number provided by the session service provider.

Two kinds of synchronization points are used to structure the exchange of data. These are minor synchronization points and major synchronization points. The major synchronization points are suitable for general dialogue and divide an activity into a series of dialogue units. The major synchronization points are located between the dialogue units to separate them from all communication before and after them. The minor synchronization points are suitable for one way data flow with no use of Expedited data and divide a dialog unit into a series of small units.

The synchronization primitives are provided for both major and minor synchronize services, which enables the session service users to place synchronization points in the normal data flow. Also, the resynchronization primitives are provided to support the resynchronize service, which enables the session service users to alter the synchronization point serial number.

2.4.2 Token and Token Management

The token is an attribute of a session connection which is used to manage activities by a session service user. The token is alternatively allocated to a session service user to give exclusive right to invoke a certain service. In PACS, a large number of application exchanges need only half duplex mode, which requires more simplified application protocols than duplex mode. Five tokens are provided, as shown in Table 2.4.

Table 2.4. The Session Layer Tokens

Token Type	Control
Data Token	Data transfer in half duplex mode
Image Token	Image transfer in half duplex mode
Release Token	Initiation of orderly release
Synchronize minor token	Insertion of minor sync points
Major/Activity token	Activity of major sync operation

The use of tokens is resolved by the negotiation. Three token management primitives are provided to handle tokens for activity management:

a) Give-Tokens : This primitive is used by a session service user to give one or more specific tokens to the other session service user. This primitive can only be used to give tokens which are in possession.

b) Please-Tokens : This primitive is used to request that a session service user get one or more specified tokens from the other session service user. Only tokens which are not possessed by the requesting session service user can be requested.

2.4.3 Activity Management

An application level exchange may consist of several smaller exchanges in PACS. As an example, an image storage device may have several image transfer requests. Each of these image transfers is an activity. If an image is being transferred and the sending entity finds a higher priority image retrieval request, then the entity may interrupt current activity and send a higher priority image by starting new activity by using session service. The previous activity will be reactivated and send the rest of the image after the new activity.

2.4.4 Session Segmentation

The session segmentation is the division of SSDU and the generation of several SPDU from single SSDU. The segmentation of SSDU occurs when the size of SSDU is bigger than the maximum TSDU size.

2.4.5 Connection-oriented Mode

The connection oriented mode of the session layer service has three distinct phases to transfer data or image between the session service user, such as session establishment, data or image transfer, and session release. While the session is established, an association is also established between session service user entities and the underlying layer. The session is established sending connection primitive by session service user. The parameters needed for data or image transfer are delivered to the underlying layer and other side entity for coordination. Once a connection has been established, the exchange of image and data

can occur between session users under the rules of established session. The session release is the last phase which releases the association established between the session users of two sides and the underlying layer. The parameters are reset and are used for coordination during data or image transfer in the underlying layer and two side entities.

2.4.5.1 Session Establishment in Session Layer

The session establishment phase is to establish an association between two session service users, and to establish an association between the session layer and the transport layer. The session establishment primitives are used to set up a session connection, to negotiate tokens and parameters, to arrange session addresses with transport addresses, to specify QOS parameters of the transport layer, and to transfer a limited amount of user data.

2.4.5.2 Data & Image Transfer in Session Layer

The data or image transfer functions in the session layer are used to transfer session text SDU or session image SDU between session service users. The session text PDU is driven from the session text SDU and the session image PDU is driven from the session image SDU. Normal data is transferred by use of the Data Transfer Session Text PDU, and normal image data is transferred by use of the Data Transfer Session Image PDU. If the extended concatenation option is selected during the session establishment and intermediate system does not support image PDU, concatenations of the Data Transfer Session Text PDU with the Data Transfer Session Image PDU are allowed. This concatenated Data Transfer Session Text PDU will be transferred to the destination through the intermediate system, even it does not support image data handling. The S-Data primitives are defined for user data transmission, and the S-Image primitives are defined for user image data transmission. The sender issues an S-Data request with user data or an S-Image request

with user image data. The size of data or image data has no limit in maximum length but more than zero octets. Each issued request is passed to the receiver as an S-Data indication or an S-Image indication.

2.4.5.3 Expedite Data and Image

The session layer provides services for Expedited data transfer and Expedited image transfer. Expedited data does not necessarily travel faster than normal data. Simply, it should be able to be conveyed even when normal data is subject to flow control and cannot be conveyed. Generally, Expedited image data transfers faster than normal image data. The S-Expedited data primitive permits maximum octets of session user data to be transferred.

2.4.5.4 Session Release in Session Layer

The session release, which releases a previously established session, can be achieved in one of three methods :

(1) orderly release : The orderly release service provides that session can be released cooperatively between the two session service users. The S-RELAESE primitives are used for orderly session release.

(2) User-abort : The session release by user-abort provides a service whereby a session service user initiates the release of a session. The S-ABORT primitives are used for user-abort session release.

(3) Provider-abort : The session release by provider-abort provides a service that a session service provider initiates the release of a session. The S-ABORT primitives are used for provider-abort session release.

2.4.6 Session Layer PDU

The Session layer PDU (SPDU) consists of a Protocol Control Information (PCI) and a data field. The PCI contains the SPDU Identifier (SI), the Length Indicator (LI), and the Parameter field. The data field contains a user data associated with the SPDU. The SI and the LI of the PCI field must be included in every SPDU, but the Parameter field of the PCI field and the data field is optional. The following Figure 2.6 shows the structure of SPDU.

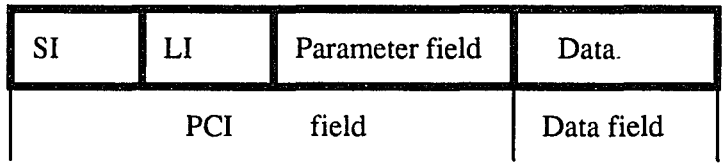

Figure 2.6. Structure of SPDU.

The parameter field may contain several parameters and parameter groups. A parameter consists of a Parameter Identifier (PI), Length Indicator (LI), and Parameter Value (PV). A parameter group consists of a Parameter Group Identifier (PGI), a Length Indicator (LI), and parameters. The structures of a Parameter and a Parameter Group are depicted in the following Figure 2.7.

Figure 2.7. Parameter and Parameter Group.

2.4.7 Connection-less Mode

The connection-less mode of the session layer service does not need to establish a connection before data or image transfer. Instead, session service makes each unit of data and unit of image to be entirely self-contained. In other words, all the information required is added in a unit of data or a unit of image. The information includes source address, destination address, control data, and etc.. This information, including data or image, is delivered to the connection-less service provider by a single service primitive.

2.5 Transport Layer

The transport layer provides transport service based on the service available from the network layer. This transport service will be used by the transport service user of the session layer. The provided service is the transparent transfer of data and image over the possibly varying reliability of the network between transport service user, and the ability to define quality of service requirements for data and image transfer. The requirements of Transport service are very simple for connection oriented network service, but are very complex for connectionless network service to provide a required level of service.

The essential properties of a reliable image transfer, reliable data transfer, connection management, quality of service, and multiplexing/demultiplexing are defined as follows. ISO-OSI standard defines five different classes of connection oriented transport layer protocols [I8072, I8073]: Transport Protocol class 0 (TP), Transport Protocol class 1 (TP1), Transport Protocol class 2 (TP2), Transport Protocol class 3 (TP3), and Transport Protocol class 4 (TP4). The usage of five classes of services is based on underlying network. TP0 is used on the network which offers a high level of reliability. Since this underlying network has allowable probability of losing data, TP0 only performs segmentation and

reassembly. TP1 is used for the network which is less reliable and needs to handle network disconnection. TP2 is also used on the reliable network. It provides not only segmentation and reassembly, but also multiplexing and demultiplexing function. TP3 just merges the functions of TP1 and TP2. TP4 is used on the unreliable networks, and is especially designed to use on the connection-less mode network service. It provides all the functions required to detect and recover errors, such as packet loss, packet duplication, and out-of-order packets arrival. TP4 is selected to transfer text and image data packets, and the control data packets.

The transport PDU has flexible packet size, and its limit is defined based on the maximum size of network layer PDU. The limit of user data is 2^{47} octets, since large size user data is transmitted with sequence number after being divided into several transport PDU. It allows no limit in the size of user data, and is one of important suitability to image data transmission.

The voice data transfer uses TP0 protocol, since it requires minimum overhead generating packet header and allows voice packet loss in some specified fraction. The voice packet loss fraction is allowed up to 1% in a general application, and can be controlled using quality of service parameter provided in the protocol.

2.5.1 Reliable Image Transfer

The transport service provides reliable image transfer function. The transport connection establishment permits image transfer or expedited image transfer in the connection oriented service. During the connection establishment, the parameter for the image transfer is set to define the quality of service required. If image data was not compressed, a bit error in

a packet image unit can be ignored, but a loss of packet image unit must be checked and the retransmission of packet image unit must be done. However, a bit error and a packet image unit loss must be checked for a compressed image data.

2.5.2 Reliable Data Transfer

The transport service provides reliable data transfer function. The transport connection establishment permits data transfer or expedited data transfer in the connection oriented service. During the connection establishment, the parameter for the image transfer is set to define the quality of service required. If image data was not compressed, a bit error in a packet image unit can be ignored, but a loss of packet image unit must be checked and the retransmission of packet image unit must be done. However, a bit error and a packet image unit loss must be checked for a compressed image data.

2.5.3 Connection Management

The connection management service of the transport layer gives flexible features to the transport service user. The management service provides the function to re-establish the failed network connection, as well as the function to give permission to the receiver side entity that is able to assign transport connections to the network connections.

2.5.4 Quality of Service

The quality of service is a number of parameters to specify the requirements for a transport service. The parameters needed are connection establishment delay, transfer delay, connection release delay, connection establishment failure rate, transfer failure rate,

connection release failure rate, connection protection level, throughput, connection priority, and residual error rate. These parameters give transport service users a way to define their requirements and are negotiated between the transport service users.

a. **Transport connection establishment delay** is used to specify the allowable maximum delay between the issuing of a transport connection request and receiving of corresponding confirmatiom.

b. **Transfer delay** is used to specify maximum delay between the issuing of transport data and receiving remote transport user of corresponding indication. The transport service provider uses this to choose the size of the data unit and to choose the network service options.

c. **Transport connection release delay** is used to specify the allowable maximum delay between the issuing of transport connection release request and receiving remote transport service user of corresponding indication.

d. **Transport connection establishment failure rate** is the ratio of connection establishment failures to total connection establishment attempts. The connection establishment failure occurs when a connection is not established within the specified maximum acceptable transport connection establishment delay.

e. **Residual error rate** is the ratio of total incorrect, lost, and duplicate transport SDU to total transport SDU transferred.

f. **Transfer failure rate** is the ratio of total transfer failures to total transfer samples. A transfer failure is a transfer sample that does not fulfill a specified minimum allowable level of throughput, transit delay, or residual error rate.

g. **Transport connection release failure rate** is the ratio of total connection release failure to total connection release request.

h. **Transport connection protection level** is used to request a protection. Level 0 is no protection, level 1 is protection from passive monitoring, level 2 is protection from modification, relay, addition or deletion, and level 4 is both of level 2 and 3.

i. **Throughput** is the smaller number of following two; first is the number of user data octets contained in the last n-1 transport service data units divided by the time between the first and last transport data indication in sequence, second is the number of user data octets contained in the last n-1 transport service data units divided by the time between the first and last transport data request in sequence.

j. **Transport connection priority** defines the relationship between transport connections based on relative importance.

2.5.5 Multiplexing/Demultiplexing

The transport service provides a multiplexing/demultiplexing function which allows more then one session connection establishment over transport connection. This service is used to optimize the bandwidth available in the transport layer and used to optimize the time to establish another same connection between session entities. The data flow from session to transport can be multiplexed, and the multiplexed data flow in the receiver side must be demultiplexed in the transport layer before delivery to session entities.

2.5.6 Transport Layer PDU

2.5.6.1 Structure of Transport Layer PDU

The Transport layer PDU (TPDU) consists of a header field and a data field. The header field contains the Length Indicator (LI) part, the fixed part, and the variable part. The data field contains a data from/to session layer. The length indicator part and the

fixed part of the header field must be included in every TPDU, but the variable part of the header field and the data field is optional. The following Figure 2.8 shows the structure of TPDU.

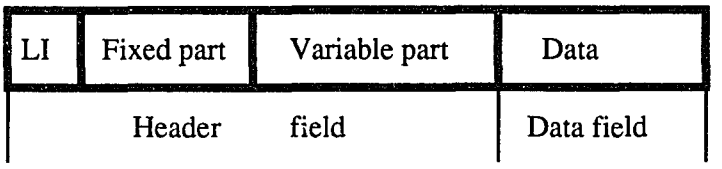

Figure 2.8. Structure of TPDU.

2.5.6.2 The Length Indicator Part of TPDU Header

The length indicator part is the first one octet of TPDU. The value of LI indicates the length of the header as a number of octets excluding the length indicator part and data field. The maximum value of LI is 254 (1111 1110), since (1111 1111) is reserved for possible future extensions.

2.5.6.3 Fixed Part of TPDU Header

The fixed part of TPDU contains typical transport parameters which is required regularly. The TDPU code is the only parameter which must be contained in the fixed part of TDPU header. The other parameters, which need to be included in a fixed part, are defined depending on each TDPU code value. The following Figure 2.9 shows the structure of the fixed part, which is used for connect request (CR), and connect confirm (CC). In the first octet of the fixed part, the first four bits are allocated for the code value. The code values are 1110 for CR, 1101 for CC, 1000 for DR, and 1100 for DC. The next four bits are used for credit (CDT) value. The octet 2 and 3 contains the destination reference (DST-REF), that is the reference identifying a transport connection at the

destination entity. The octet 4 and 5 contains the source reference (SRC-REF), that is the reference identifying a transport connection at the source entity. The octet 6 contains class (bits 8 to 5) and option (bits 4 to 1) variables. The class defines the selected transport protocol class as a binary number.

	Code,CDT	DST-REF	SRC-REF	Class,Option
octet	1	2,3	4,5	6

Figure 2.9. Fixed part of TPDU for CR and CC.

The structure that is defined for the fixed part of disconnection request (DR) is shown in Figure 2.10. The DR code value is 1000 0000. The octet 2 and 3 contains the destination reference (DST-REF), that identifies the transport connection at the remote transport entity. The octet 4 and 5 contains the source reference (SRC-REF), that identifies the transport connection at the transport entity initiating the TPDU. The octet 6 contains the reason for disconnecting transport connection.

	DR	DST-REF	SRC-REF	Reason
octet	1	2,3	4,5	6

Figure 2.10. Fixed part of TPDU for DR.

The structure that is defined for the fixed part of disconnection confirm (DC) is shown in Figure 2.11. The DC code value is 1100 0000. The octet 2 and 3 contains the destination

reference (DST-REF) that identifies the transport connection at the remote transport entity. The octet 4 and 5 contains the source reference (SRC-REF) that identifies the transport connection at the transport entity initiating the TPDU.

	DC	DST-REF	SRC-REF
octet	1	2,3	4,5

Figure 2.11. Fixed part of TPDU for DC.

2.5.6.4 Variable Part of TPDU Header

The variable parameters are coded into the variable part. The variable part may contain several variables, such as transport service point identifier parameter, TPDU size parameter, version number parameter, security parameter, checksum parameter, additional option selection parameter, acknowledge time parameter, throughput parameter, residual error rate parameter, priority parameter, transit delay parameter, and reassignment time parameter. Each of these parameters has three fields: parameter code field, parameter length field, and parameter value field as shown in Figure 2.12.

	parameter code	parameter length	parameter value
Octet	n	n+1	n+2,...,n+m+1

Figure 2.12. Option Parameter Structure in TPDU Header.

The value of parameter code distinguishes each parameter as in Figure 2.13. The parameter length field defines the length of parameter value in the number of octets.

Parameter Code	code
Calling TSAP identifier	1100 0001
Called TSAP identifier	1100 0010
TPDU size	1100 0000
Version number	1100 0100
Security	1100 0101
Additional option selection	1100 0110
Alternative protocol class	1100 0111
Acknowledge time	1000 0101
Throughput	1000 1001
Residual error rate	1000 0110
Priority	1000 0111
Transit delay	1000 1000
Reassignment time	1000 1011

Figure 2.13. Parameter Code in Variable part.

2.6 Network Layer

The network service is a service provided to transport layer functions in end systems. This layer provides independence of the higher layer from the variety of technologies that may be encountered in a path across a number of networks. Since the network service provides independence to the network service user, the network service user may transfer data, image, and voice over various subnetworks with transparency on the difference of subnetwork. But, the quality of service must still be considered by the network service user. Furthermore, the network layer provides routing and relaying functions to deliver data or image to the destination through intermediate systems [I8348A]. This end-to-end transfer between network service users is available from the network service provider. The network layer protocol provides a very flexible size of packet, and the limit of network PDU size is 65535 octets in the point of protocol definition [I8348B]. Thus, the maximum size of network layer PDU can be defined in each Local PACS based on their buffer capacity, data link and physical layer protocol, and performance requirement.

2.6.1 Routing

The routing function is used to select a path from sender station to destination station [I8473]. The routing function may reside at end systems and select the pass before it sends the data. It is called source routing. In the source routing, the source broadcasts a "discovery" frame to find a route to a given destination. These frames travel all possible paths to the destination station while each frame records the route it takes. The destination station returns all arrived frames with the recorded path to the source station. The source

then makes a choice of route to take. This routing may be used to support connection oriented service along with other useful routing algorithms. The routing algorithms can be distinguished as two kinds, such as static and adaptive routing. [I8348D]

2.6.2 Segmentation/Reassembly

The segmentation and reassembly is used to transmit the larger size of PDU or IPDU than the maximum size of the underlying network, or to connect two networks that have different allowable packet sizes. If the IPDU of the network service user is larger than the packet size of the underlying network, then the IPDU must be segmented and sent as multiple smaller units. These units are sent to the destination through intermediate networks. While the segmented unit is passed through the intermediate networks, it may be reassembled or segmented again to satisfy the packet size requirements of intermediate networks. The multiple smaller units arrived at the destination are reassembled before being delivered to the network service user entity. As an example, maximum packet size of the FDDI standard is 4500 bytes, ARPANET is 1008 bits, X.25 is 8192 bits, and ETHERNET is 1500 bytes.

2.6.3 Addressing

The addresses, which are used in a network layer primitives, are all network service access point (NSAP) addresses. The NSAP address has a flexible size in length, which can be defined up to a maximum of 20 octets (40 decimal digits) using length parameter [I8348C]. The parameters for addressing in a network layer primitive are a calling address parameter, a called address parameter, a responding address parameter, and a receipt confirmation selection parameter. The calling address is the address of the NSAP from which a network layer primitive is sent. The called address is the address of the

NSAP to which a network layer primitive is sent. The calling address parameter and the called address parameter are used in a N-connect.request and a N-connect.indication primitive. The responding address is the address of the NSAP at which a network layer primitive is received. The responding address parameter is used in a N-connect.response and a N-connect.confirm primitive. The receipt confirmation selection parameter is used to check the availability of the receipt confirmation service or to indicate the use of that service for data transfer in the network layer.

2.6.3.1 The Structure of NSAP Address

The NSAP address consists of two parts: the Initial Domain Part (IDP) and the Domain Specific Part (DSP). The IDP identifies a network addressing domain, which specifies the network addressing authority and the addressing format. The DSP contains the corresponding domain address. The PACS NSAP address structure is shown in Figure 2.14.

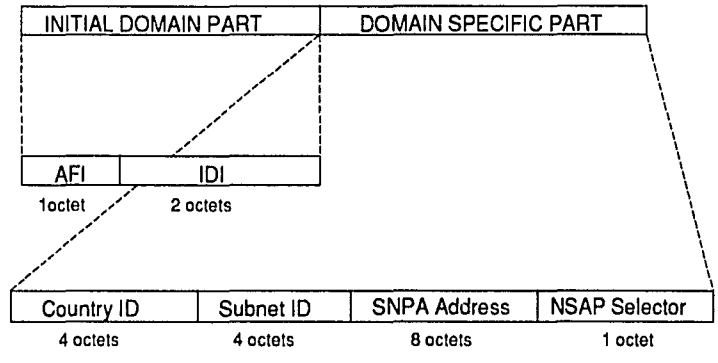

Figure 2.14. NSAP Address Structure.

The IDP is subdivided into the Authority and Format Identifier (AFI) and the Initial Domain Identifier (IDI). The AFI specifies the format of the IDI, the addressing authority,

and the abstract syntax of the DSP. In the PACS, the AFI value is defined as 47. It defines that the IDI consists of 2 bytes of International Code Designator (ICD), which will be assigned according to ISO 6523 (It needs an agreement with ISO). It also defines that the DSP address is represented in binary.

The DSP is subdivided into country ID, subnetwork ID, SubNetwork Point of Attachment (SNPA) ID, and NSAP selector. The first 4 bytes are assigned for country ID. The next 4 bytes are assigned for subnetwork ID. The following 8 bytes are assigned for SNPA ID. The last byte is used for NSAP selector.

2.6.4 Flow Control

The flow control is used to control the traffic over the intermediate network. It regulates the amount of data on an individual connection between source and destination pair to prevent overflow of the buffers dedicated to an individual connection. It is achieved by using such flow control that buffers, transmission bandwidth, processor time, and logical channel are utilized properly with satisfying performance requirements of the network service user.

2.6.5 Internet Addressing

A internet addressing scheme is required to identify the NSAP by one single address. The internet addressing scheme for PACS needs a global agreement for unique identification of allocated addresses. There are several addressing schemes, which are currently used, such as X.121 for public data network, E.163 for public switched telephone network, E.164 for ISDN, F.69 for telex, and etc. But this addressing scheme is not available for the global identification of private networks. This leads to a new addressing

scheme defined in Section 2.6.3. It follows a network layer addressing standard ISO 8348/Add.2 which can be used to allocate NSAP addresses with global uniqueness. Furthermore, it can interconnect a private network and a public network.

2.6.6 Internet Routing

The internet routing scheme is recommended as a source routing, which follows the routing scheme in ISO 8473 protocol. In the source routing, the route is determined by the PDU sender entity in the network layer, and the chosen intermediate routes are inserted in the option part of the PDU header. The network entity in the intermediate system relays the PDU using the list of intermediate routes in the PDU header.

2.6.7 Connectionless Mode Network Layer PDU

2.6.7.1 Structure of PDU

The Network layer PDU (NPDU) consists of a header field and a data field. The header field contains the fixed part, the address part, the segmentation part, and the option part. The data field contains a data from/to Transport layer. The fixed part and the address part of the header field must be included in every PDU, but the other part of the header field and the data field is optional. The following Figure 2.15 shows the structure of NPDU.

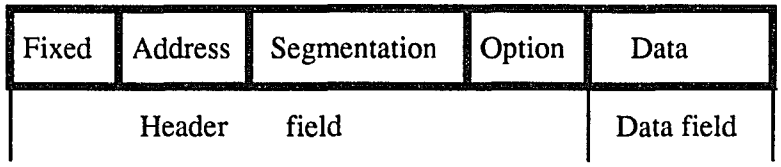

Figure 2.15. Structure of NPDU.

2.6.7.2 The Fixed Part of NPDU Header

The fixed part of NPDU contains Network Layer Protocol Identifier (NLPI), Length Indicator (LI), Version / Protocol ID Extension (VPID), Lifetime (LT), Flags and Type (FT), Segment Length (SL), and Checksum. The value of these fields, except SL and Checksum fields, consists of one octet (8 bits) per field. The SL and Checksum consists of two octets each as shown in Figure 2.16.

	NLPI	LI	VPID	LT	FT	SL	Checksum
octet	1	2	3	4	5	6,7	8,9

Figure 2.16. Fixed part of NPDU

The value of NLPI indicates a network layer protocol. As an example the value of new ACR-NEMA network layer protocol option is (1000 1111) and the value of ISO 8473 protocol option is (1000 0001). The value of LI indicates the length of the header as a number of octets. The maximum value of LI is 254 (1111 1110), since (1111 1111) is reserved for the possible future extensions. The value of VPID indicates a version or an extension of network layer protocol. The value of LI for connectionless mode is (0000 0001). The value of LT indicates the remaining lifetime of NPDU as a binary number with a time unit of 1 milliseconds. The initial value of LT is set by the entity in the originating network layer. The value decrements when the NPDU is processed by a network entity. The value of LT is decrement at least one in each entity, and it is decrement by additional one whenever the sum of the transit delay and the delay within the system processing the PDU exceeds additional 1 millisecond. The FT field

contains three flags and a type code. Three flags are Segmentation Permitted flag (SP), More Segment flag (MS), and Error Report flag (ER). The three flags of one bit and a type code of five bits are shown in following Figure 2.17.

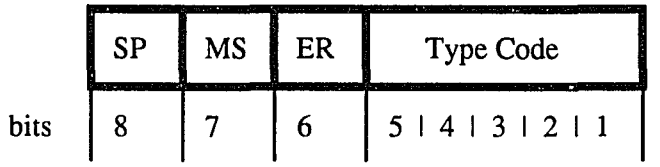

Figure 2.17. Flags and Type field Structure

The SP flag is set to 1 when the segmentation is permitted. The MS flag is set to 1 when the segmentation has taken place and the last octet of the NSDU is not contained in this PDU. The ER flag is set to 1 when it is requested to generate an error report PDU upon discard of the PDU. The type code indicates the type of the PDU, such as : Data PDU is (11100), Image PDU is (10100), and Error PDU is (00001). The value of SL field indicates the entire length of PDU segment which includes not only header but also data. The value of checksum is derived only from the header, not including data. When this value is zero, the checksum function must be ignored. But when it is not, the checksum must be processed or the PDU mast be discarded.

2.6.7.3 The Address Part of NPDU Header

The address part of NPDU consists of length field and address field. The length field indicates the length of the address field and locates in front of address field. Since the length field is one octet, the maximum length of address is 255 (1111 1111). The following Figure 2.18 shows an address part, which immediately follows the fixed part.

	DALI (n)	Destination Address n octet	SALI (m)	Source Address m octet
Octet	10	11, ..., n+10	n+11	n+11,..,n+11+m

Figure 2.18. Address part in PDU header.

The DALI is the Destination Address Length Indicator and is assumed to have a value (n). The SALI is the Source Address Length Indicator and has a value (m). The addresses for destination and source are always Network Service Access Point addresses.

2.6.7.4 The Segmentation Part of NPDU Header

The segmentation part contains information of segmented PDU, including Data Unit Identifier, Segment Offset, and Total length as in Figure 2.19. The segmentation part exists as an option, when the segmentation is permitted.

	Data Unit Identifier	Segment Offset	Total Length
Octet	n,n+1	n+2,n+3	n+4,n+5

Figure 2.19. Segmentation part in PDU header.

The Data Unit Identifier field contains an identification of the originating NPDU, which will be used for reassembly in the destination network entity. The Segment Offset contains the information of relative position of data part in the originating NPDU. The value of Total Length field indicates the total length of the originating NPDU including

user data part.

2.6.7.5 The Options Part of NPDU Header

The optional parameters are coded into the options part. The options part may have several options, such as source routing parameter, quality of service parameter, priority parameter, recording of route parameter, security parameter, and padding parameter. Each of these parameters has three fields: parameter code field, parameter length field, and parameter value field as shown in Figure 2.20.

	parameter code	parameter length	parameter value
Octet	n	n+1	n+2,...,n+m+1

Figure 2.20. Option Parameter Structure in PDU Header.

The value of the parameter code distinguishes each parameter as in Figure 2.21. The parameter length field defines the length of parameter value in the number of octets.

2.7 Data Link Layer

The data link layer provides functional and procedural means of the transparent and reliable transfer of data between data link service users. It also provides independence of the underlying physical layer to the data link service user. The standard is not aimed to develop data link layer protocol, but to recommend available protocols and to provide function requirements for a designer who is developing new protocol. The current data link layer protocols based on coaxial cable may be used for Picture Archiving and

Communication System. But since the size of pixel data requires large bandwidth, new available protocol based on fiber optic cable is suitable for real time image transfer. Examples of high performance fiber optic networks are a Fiber Distributed Data Interface (FDDI) local area network and an IEEE 802.6 Distributed Queue Dual Bus (DQDB) metropolitan area network.

Parameter Code	code
source routing parameter code	1100 1000
quality of service parameter code	1100 0011
priority parameter code	1100 1100
recording of route parameter code	1100 1011
security parameter code	1100 0101
padding parameter code	1100 xx00

Figure 2.21. Parameter Code in Options part.

The medium access control sublayer of DQDB supports asychronous and isochronous transfer service. It also support logical link control sublayer by a connectionless medium access control service in a manner consistent with IEEE 802.2. It uses slots under the control of the distributed queue for the medium access control. It has two types of slots: queued-arbitrated slot for asynchronous transfer and non-arbitrated slot for isochronous

transfer. Each slot contains an access control field which is used to control the writing of data into slot and the reading of data from slot. The distributed queue with counters in each node allows minimum access delay and uniform access of shared channel.

2.8 Physical Layer

The objective of physical layer service is transmitting row bits over transmission medium. The physical layer service also provides transparency of physical medium to the physical layer service user. The physical layer protocol depends highly on the transmission medium and the topology of network. The main functions of the physical layer is providing mechanical, electrical, functional, and procedural means to activate, maintain, and deactivate physical connections for the transmission of bit streams between data link entities.

As an example, the FDDI is a dual ring network running at 100 Mbps with up to 1000 stations connected. The maximum distance is up to 200 Km with fiber optic transmission medium. It uses light emitting diodes to emit light pulses when an electrical current is applied. It uses a photodiode to generate an electrical signal when light falls on it. The FDDI consists of two counter-rotating fiber optic rings. This allows a backup when one ring breaks. The primary ring transmits data counter-clockwise and the secondary ring transmits data clockwise. The bit error rate should be no more than 1 error in 2.5×10^{10} bits. The FDDI uses 4 out of 5 encoding instead of Manchester encoding, because it requires only 125 megabaud comparing 100 Mbps Manchester encoding requires 200 megabaud.

CHAPTER 3

PACS IMPLEMENTATION

The protocols defined in the previous chapter are implemented as a PACS prototype. The purpose of protocol implementation is to accelerate the process of the development of PACS in the proposed ACR-NEMA Version 3.0 protocol suit. The PACS prototype is implemented on SUN workstations and a VAX 11/730 system. The network software of PACS is coded entirely in the C programming language. It is implemented on the top of Sun Microsystems SunOS and ported to Berkeley BSD4.3-Reno operating system. The underlying network is the Ethernet, which connects hosts at The University of Arizona. This Chapter describes the prototype environment and the protocol software in the PACS prototype.

3.1 Overall Software Structure

This software can support two different network service below the transport service access point. One of these service is the TCP/IP of Internet, and the other is the TP4/CLNP of ISO. The implementation of higher layer protocols on the TCP/IP allows the protocol development in a robust and mature internet environment. The implemented protocols with TCP/IP environment are moved to the TP4/CLNP and operates over pure ISO lower level software.

Implemented protocol softwares will reside in each node to communicate with Local or Global PACS. Local PACS connects distributed Imaging Equipments, Workstations, and Local DBASs. Global PACS interconnects distributed Local PACSs, Regional DBASs, and National DBASs through a national communications network. PACS users

are spread all over the country and are put into Local PACS, or Global PACS, by the definition of a user group. Each group of users will have permission to use Local PACS or Global PACS. Figure 3.1 shows a PACS with implemented protocol softwares. There are two processes: a PACS server and a PACS client. The PACS server is running on Local DBASs, Regional DBASs, and National DBASs. The PACS client is running on Imaging Equipments and Workstations. Each process provides several user functions: Initialize function, Connect function, User Authentication function, Text file send function, Text file retrieval function, Image file send function, Image file retrieval function, List directory function, Change directory function, and Disconnect function. The Initialize function initializes the environment of PACS node as a PACS server or a PACS client when the program is executed. PACS users in each client node establish connections to a DBAS using Connect functions. After these links are established, users can use the User Authentication function to get a permission to access DBAS. Once users are authenticated other functions are available to them. PACS users use these available functions for text data transfers, image file transfers, and system command transfers until the established connection is released by the Disconnect function.

User functions are mapped to primitives provided in the applicative layer, the presentation layer, and the session layer. Figure 3.2 shows the overall architecture of mapping functions to primitives in each layer. As an example the Connect function mapped to an A-Connect request primitive, an A-Connect indication primitive, an A-Connect response primitive, and an A-Connect conform primitive in the application layer. The application primitives are mapped to a P-Connect request primitive, a P-Connect indication primitive, a P-Connect response primitive, and a P-Connect conform primitive in the presentation layer. The

presentation primitives are mapped to a S-Connect request primitive, a S-Connect indication primitive, a S-Connect response primitive, and a S-Connect conform primitive in the session layer.

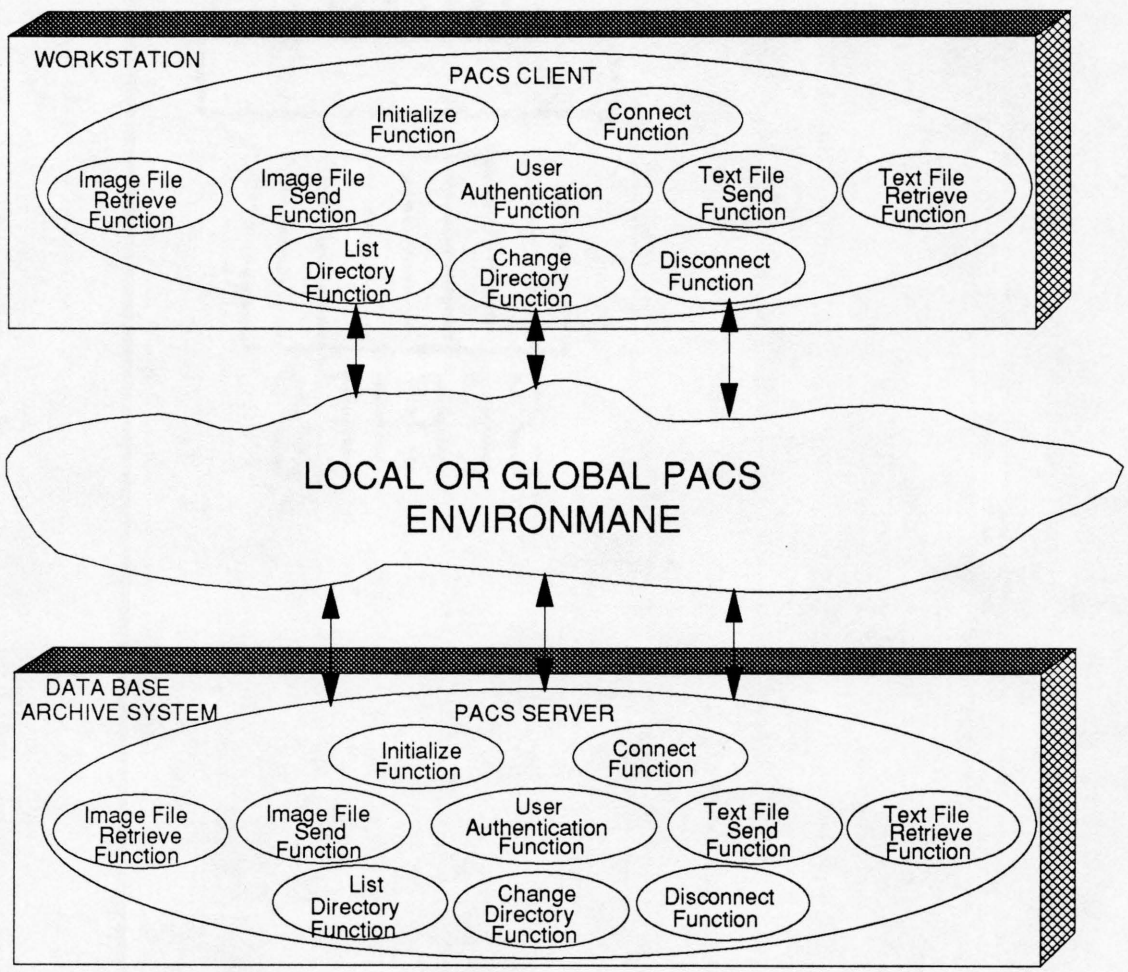

Figure 3.1. PACS Functions with Implemented Software.

The Image file send function mapped to an A-IData request primitive, an A-IData indication primitive, an A-TData request primitive, an A-TData indication primitive, an

A-GiveToken request primitive, an A-GiveToken indication primitive, an A-MinorSync request primitive, and an A-MinorSync indication primitive in the application layer. The application primitives are mapped to a P-IData request primitive, a P-IData indication primitive, a P-TData request primitive, a P-TData indication primitive, a P-GiveToken request primitive, a P-GiveToken indication primitive, a P-MinorSync request primitive, and a P-MinorSync indication primitive in the presentation layer. The presentation primitives are mapped to a S-IData request primitive, a S-IData indication primitive, a S-TData request primitive, a S-TData indication primitive, a S-GiveToken request primitive, a S-GiveToken indication primitive, a S-MinorSync request primitive, and a S-MinorSync indication primitive in the session layer. Figure 3.3 shows an example of time sequenced operations for described primitives in each layer. Provided primitives are developed as procedures in each layer with one-to-one mapping.

The implementation of protocol software consists four modules in the PACS prototype: application module, presentation module, session module, and transport interface module. The application module contains application service providers and presentation service users. The presentation module includes presentation service providers and session service users. The session module contains session service providers and transport service users. The transport interface module provides interfaces to TCP and TP4. Each of four modules has their sub-modules as shown in Figure 3.4.

The application module includes initialization, connect, user, get, put, iget, iput, ls, cd, disconnect, and test sub-modules. The initialization sub-module provides the PACS process environment to run as a client mode or a server mode. It resets variables, maintains the order of the activity of sub-modules, and also provides user interface in the client mode. Application sub-modules need to have a certain order in activation, such as connection must established by the connection sub-module as a first step of PACS operation. Then

user must be authenticated using the user sub-module before allowed to use other services. The client process must be interactive with a user to get user commands. The initialization sub-module gets user commands from a terminal and activates corresponding sub-modules. In the server mode, the initialization sub-module resets variables, forks shell, and waits for an arrival of connection indication. Variables are set based on the usage of TCP/IP or TP4/CLNP protocol in the underlying layer. Forking shell is required to process system commands in the server side.

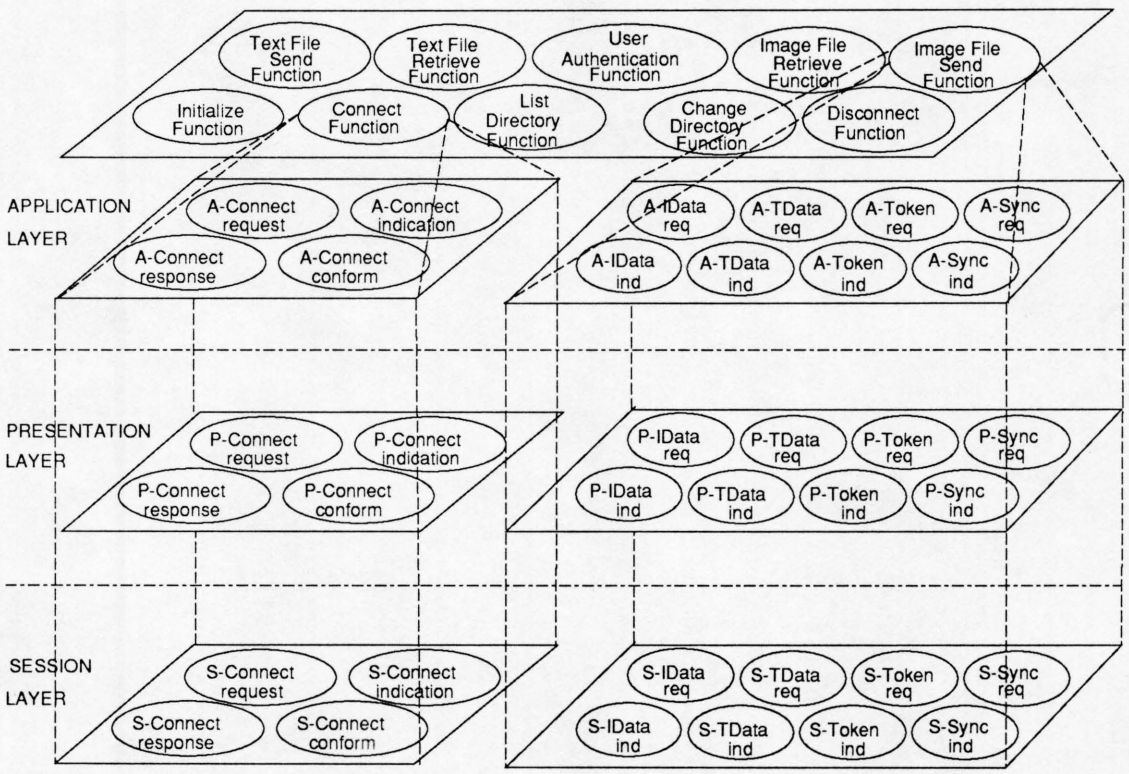

Figure 3.2. Mapping of User Functions into Protocol Primitive.

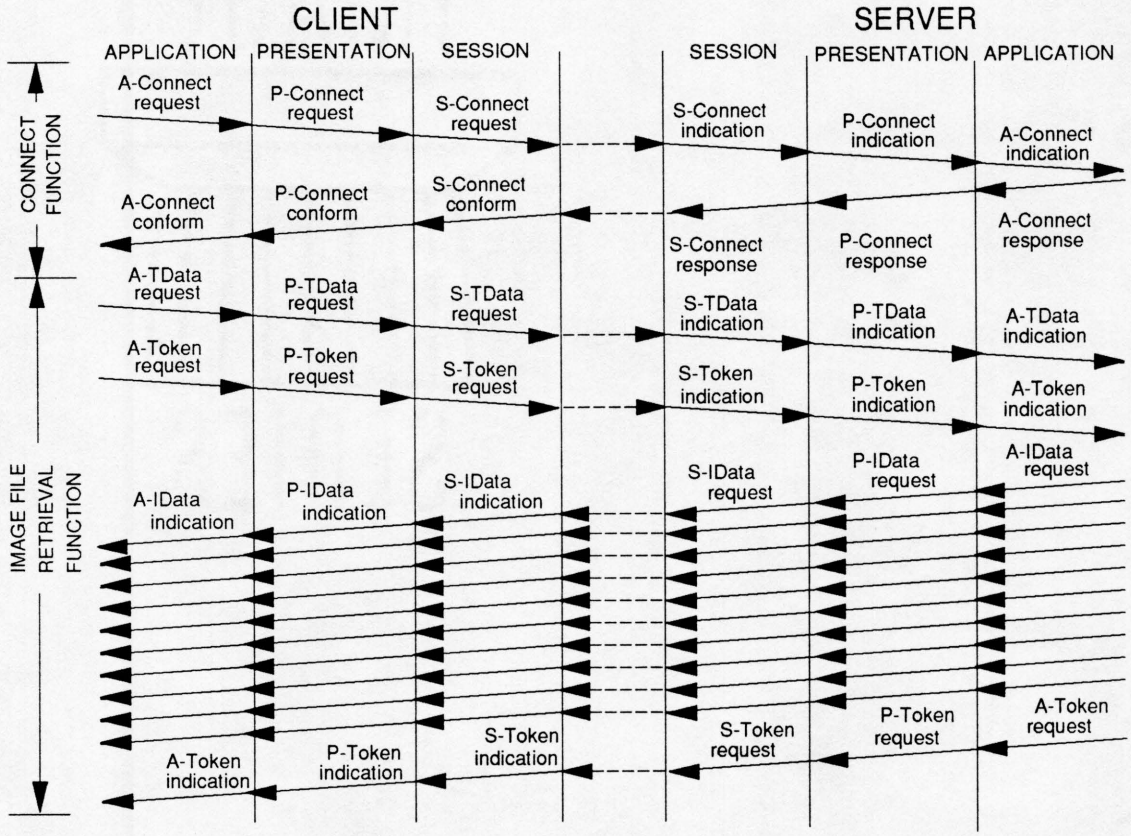

Figure 3.3. Primitive Operations in a PACS.

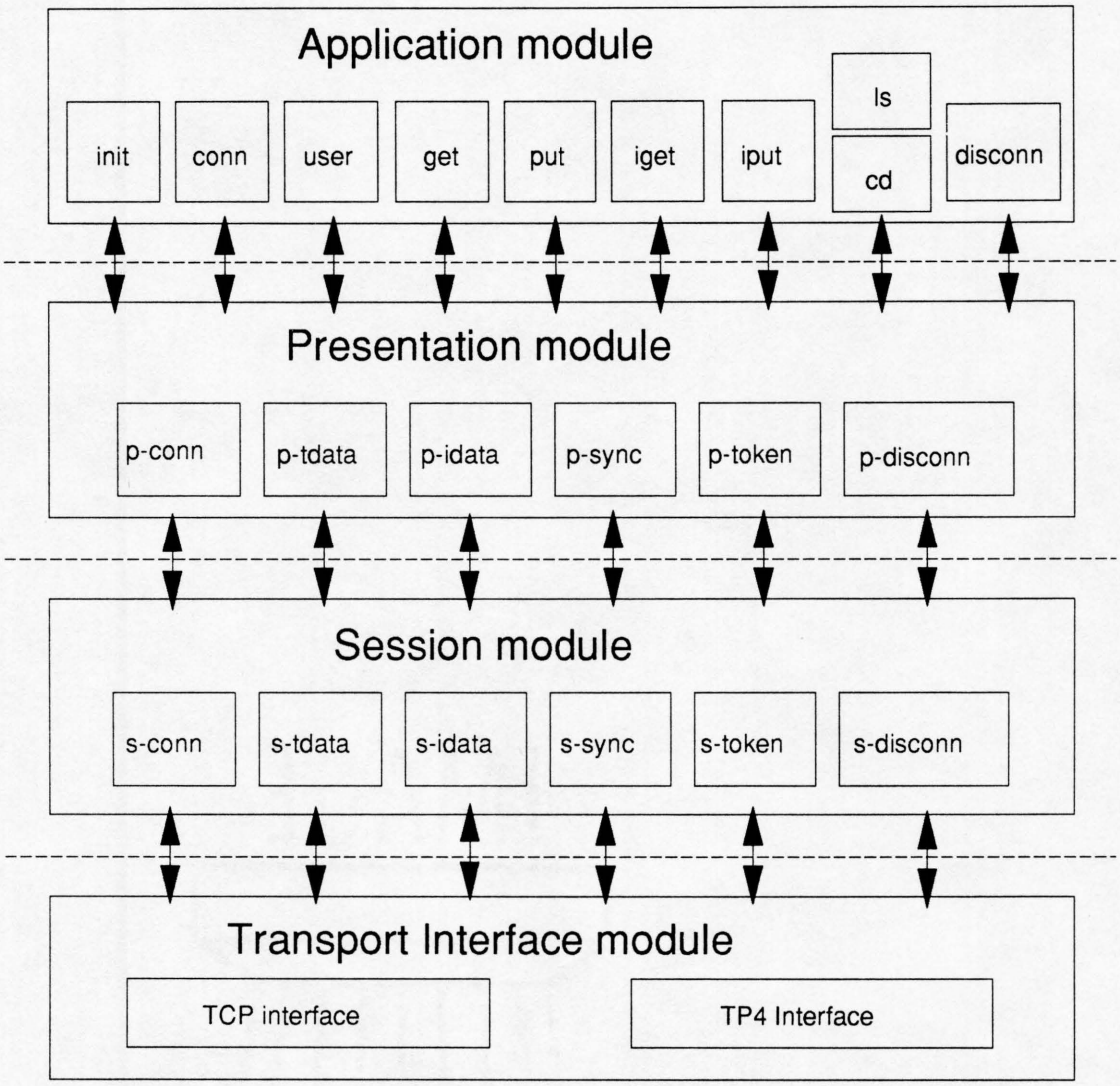

Figure 3.4. Overall Structure of PACS Implementation.

The connect sub-module provides a connection establishment between a client and a server. The disconnect sub-module provides a release of the established connection between the client and the server. These two sub-modules provides a association control

service between peer application service entities. The connect sub-module in the client side retrieves interactively a name of the remote host from a user of the PACS. The connection request and the host name are packetized and passed down to p-connect sub-module. Then, it waits for the connection response from the p-connect sub-module. The connect sub-module in the server side receives a connection indication from the p-connect sub-module in the presentation layer. The received connection indication is evaluated by the connect sub-module. If the connection is accepted in the server side, an accept packet is sent to the p-connect sub-module. Otherwise a reject packet is sent to the p-connect sub-module. When the disconnect sub-module in the client is activated interactively by a user of the PACS, the disconnection request is packetized and passed down to p-disconnect sub-module in the presentation layer. Then, it waits for the disconnection response from the p-disconnect sub-module. The disconnect sub-module in the server side receives a disconnection indication from the p-disconnect sub-module in the presentation layer. The received disconnection indication is evaluated by the disconnect sub-module. If the disconnection is accepted in the server side, a disconnection response is sent to the p-disconnect sub-module with accept variable. Otherwise the disconnection response contains a reject and a reason of the rejection variables.

The user sub-module provides an authentication service in the application layer. The get sub-module provides a text file retrieval from the text database server. The put sub-module provides a text file transfer to the text database server. The iget sub-module provides an image file retrieval from the image database server. The iput sub-module provides a image file transfer to the image database server. The user, get, and put sub-module use p-tdata sub-module in the presentation layer to transfer a command and data. The iget and iput sub-module use the p-tdata sub-module to transfer a command and the p-idata sub-module to transfer image data. The details of these five sub-modules are

described in the Chapter 3.3.2.

The presentation module includes p-connect, p-tdata, p-idata, p-disconnect, p-sync, and p-token sub-modules. The p-connect sub-module provides a connection establishment between two peer entities in the presentation layer. The p-connect sub-module includes p-connect.req, p-connect.ind, p-connect.resp, and p-connect.conf routines. The p-disconnect sub-module provides a release of the established connection between the peer entities. The p-disconnect sub-module includes p-disconnect.req, p-disconnect.ind, p-disconnect.resp, and p-disconnect.conf routines. The p-connect sub-module in the sender side is activated by the connection request from the connect sub-module in the application layer. The connection request is packetized with a presentation header and passed down to the s-connect sub-module by the p-connect.req routine. Then, it waits for the activation of a p-connect.resp routine by the s-connect sub-module. The connect sub-module in the server side receives a p-connect.ind from the s-connect sub-module in the session layer. The header of the received p-connect.ind is evaluated by the p-connect sub-module. If parameters of the header is negotiated successfully, the p-connect.ind routine activates a connection indication of the connect sub-module in the application layer. Otherwise a p-connect.resp with a reject header is sent to the s-connect sub-module. Headers of the presentation layer are defined in the Section 3.4.2. The p-disconnect sub-module provides a identical service to release the established connect between peer presentation entities.

The p-tdata sub-module provides a text data transfer between two peer entities in the presentation layer. The p-tdata sub-module includes p-tdata.req and p-tdata.ind routines. The p-idata sub-module provides a image data transfer between two peer entities in the presentation layer. The p-idata sub-module includes p-idata.req and p-idata.ind routines. A transfer syntax is generated from an application protocol data unit. A presentation protocol data unit is generated by adding header to the transfer syntax. The presentation

protocol data unit is passed to the session module in a sender. When a receiver receives this presentation protocol data unit, it generates system dependent syntax from the transfer syntax.

The p-sync and p-token sub-module provide an interface between the session module and the application module. When a presentation service user tries to access a dialog management or a synchronization service in the presentation module, these sub-modules deliver the service from the session module.

The session module provide connection oriented service, which has three distinct phases: session establishment, data transfer, and session release. In PACS, the data phase involves both of text and image data. The voice service is not implemented in this prototype PACS. The connection oriented service requires session establishment phase to transfer text or image data. The association is established between peer entities and between each peer entity and the next lower layer entity, while the session is established. This association will take care of text and image data transfer between peer entities. The association is released by session release service after the requested transfer. The sequence of state transition in connection oriented service is illustrated in Figure 3.5. The initial state in each layer is *idle*. The *idle* state enters the *wait* state, when the s-connect.req routine or the s-connect.ind routine is arrived. The *wait* state by the s-connect.req is waiting for an arrival of the s-connect.conf and the *wait* state by the s-connect.ind is waiting for an arrival of the s-connect.resp. The *wait* state is becomes *transfer ready* state by the arrival of the waiting routine with an accept parameter. Therefore, the four types of routines are used to establish association between peer entities for data transfer and image transfer. After a connection is made the primitives are used to transfer image and text data.

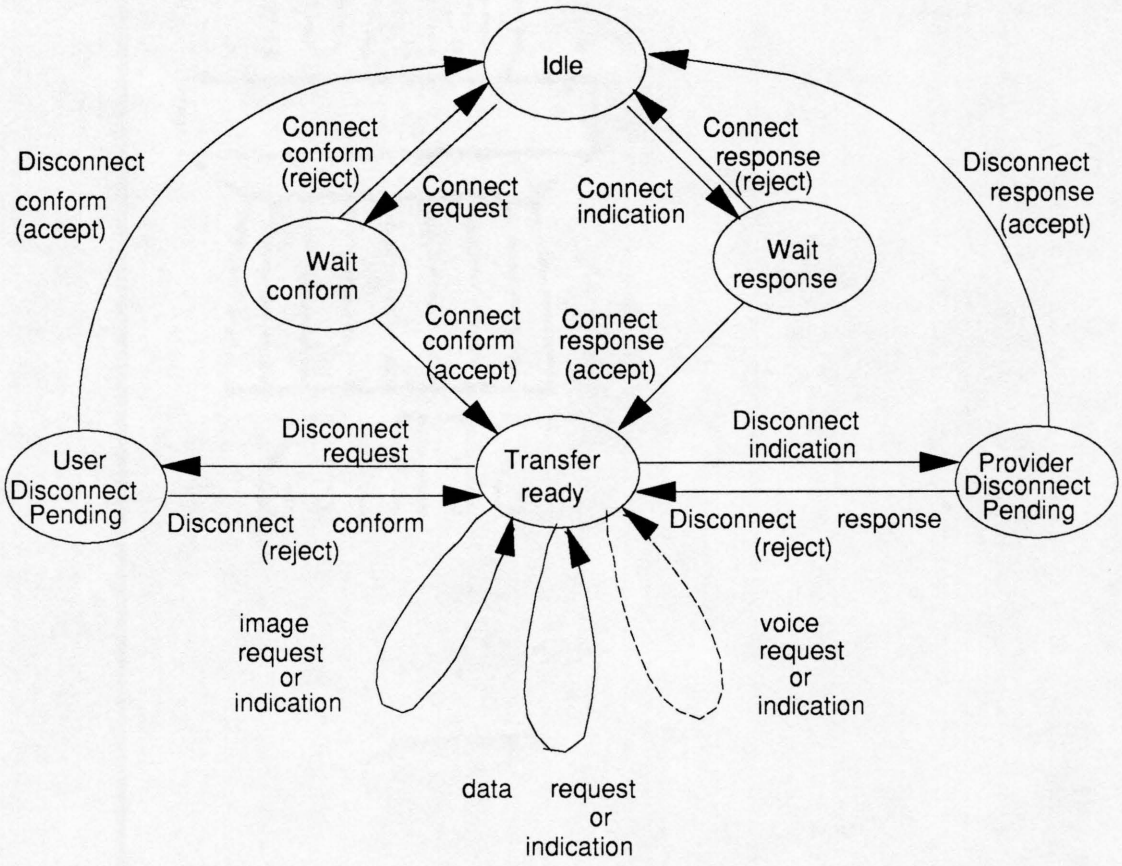

Figure 3.5. Connection Oriented Service Sequence.

The *transfer ready* state enters to the *pending* state by the arrival of the s-disconnect.req routine at the sender or the s-disconnect.ind routine at the receiver. The arrival of s-disconnect.resp routine or the s-disconnect.conf routine with accept parameter makes state transition from the *pending* state to the *idle* state. The arrival of disconnect.resp routine or the s-disconnect.conf routine with reject parameter makes state transition back to the *transfer ready* state.

The session module includes s-connect, s-tdata, s-idata, s-disconnect, s-sync, and s-token sub-modules. The s-connect sub-module provides a connection establishment between

two peer entities in the session layer. The s-connect sub-module includes s-connect.req, s-connect.ind, s-connect.resp, and s-connect.conf routines. The s-disconnect sub-module provides a release of the established connection between the peer entities. The s-disconnect sub-module includes s-disconnect.req, s-disconnect.ind, s-disconnect.resp, and s-disconnect.conf routines. The s-connect sub-module in the sender side is activated by the p-connect.req routine of the p-connect sub-module in the presentation layer. The packet from p-connect.req routine is added with a session header and passed down to the t-connect sub-module by the s-connect.req routine. Then, it waits for the activation of a s-connect.resp routine by the t-connect sub-module. The connect sub-module in the server side receives a s-connect.ind from the t-connect sub-module in the transport interface layer. The header of the received s-connect.ind is evaluated by the s-connect sub-module. If parameters of the header is negotiated successfully, the s-connect.ind routine activates a p-connect.ind routine of the p-connect sub-module in the presentation layer. Otherwise a s-connect.resp with a reject header is sent to the t-connect sub-module. Headers of the session layer are defined in the Section 3.5.2.

The s-tdata sub-module provides a text data transfer between two peer entities in the presentation layer. The s-tdata sub-module includes s-tdata.req and s-tdata.ind routines. The s-idata sub-module provides a image data transfer between two peer entities in the presentation layer. The s-idata sub-module includes s-idata.req and s-idata.ind routines. The s-tdata.req is activated by the p-tdata.req and the s-idata.req is activated by the p-idata.req. The arrived s-tdata.ind or s-idata.ind is passed up to p-tdata.ind or p-idata.ind respectively.

The s-token sub-module provides a dialog management, which is keeping track of the turn to talk in half-duplex mode. In this PACS implementation, the upper layer software is structured to expect the session users to take turns for data transmission. As an example,

the get sub-module in the application layer will send the get command packet to a server with assuming that the token is initially located in a client side. The token is send to the server using s-gtoken.req routine. It will give the right to send data to the server. The server then will send requested data and the token will go back to the client for allowing another request. The dialog management simplifies the PACS implementation, since allowing users to send other requests before the first request has been replied needlessly complicates the implementation.

The s-sync sub-module provides a synchronization service, which is used to the session entities back to the known state in the case of an error. In this PACS implementation, the upper layer software is not using the s-sync sub-module. But some of routines of the synchronization service are implemented in the s-sync sub-module for future need. It provides a service to recover from upper layer protocol errors. Communication errors are not recovered here but should be recovered in the transport layer.

The transport interface module provides an interface between session layer protocol and transport layer protocols. Transport layer protocols are TCP in Internet standard and TP4 in ISO standard.

The performance is essential for the network programming of PACS to transfer the large size of image data. The most sensitive factor of the PACS performance in the layered protocol implementation is the method of message passing. Excessive copy operations are required when each layer is built as a separate process, since one copy operation is required for writing in the sender side and another copy operation for reading in the receiver side. The protocol specification based on OSI reference model prescribes functional layering, but does not prescribe that individual layer entities must be distinct process modules in the implementation [DAV85]. Figure 3.6 shows the technique that is used to implement protocol layers.

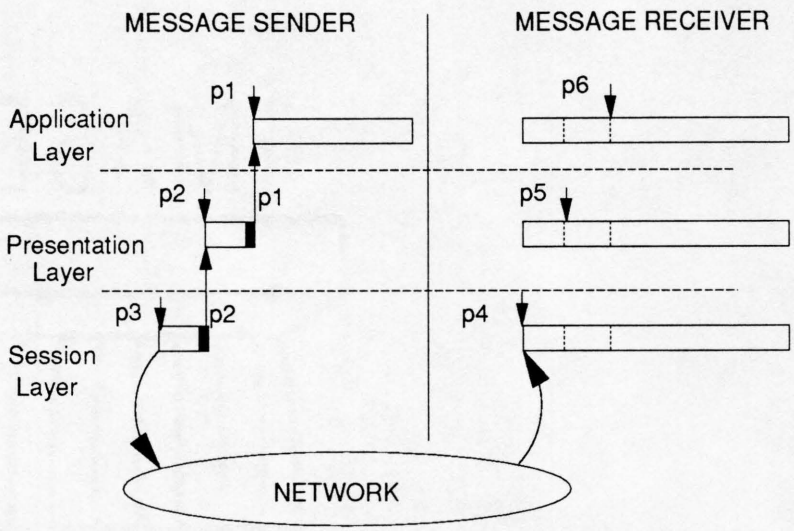

Figure 3.6. Message Passing Architecture

The receiver side of the diagram depicts a simple example of the message sending service such as request and response. The sender side depicts the message receiving service such as indication and conform. The different execution structures of two sides are derived from their inherent characteristics for memory management.

In the sender side, the application layer service provider of the sender allocates memory for their service data unit, writes data in the allocated memory, puts null pointer in the tail, and passes the pointer 1 of the application service data unit to presentation layer. The presentation layer service provider receives this pointer, allocates memory for header, generates presentation header with pointer 2, puts received pointer 1 in the tail, and passes pointer 2 to session layer. The session layer service provider receives pointer 2, generates session header with pointer 3, serialize it, and sends to transport layer service provider until it reaches null pointer.

In the receiver side, the session layer receives data, reads session header, moves pointer,

and transfers the new pointer to the session layer service user. The presentation layer service provider does the same process and sends new pointer to the presentation layer service user. The application layer receives pointer and reads data following pointer.

3.1.1 UNIX Operating System

The PACS protocols are implemented under the UNIX operating systems. The first version of the UNIX is developed by Ken Thompson and Dennis Ritchie at Bell Laboratories in 1969. It was originally designed as a general-purpose, multi-user, interactive operating system for minicomputers. Support for a broad range of communication protocols has been incorporated into the system later. There are various versions of UNIX in use today. Four versions, which are AT&T UNIX System V, 4.3 Berkeley Software Distribution (4.3BSD), Sun Microsystems SunOS, and Microsoft Xenix System V, are leading UNIX in the current market.

The two UNIX platforms, 4.3BSD-Reno and SunOS Release 4.1.1, are used for the protocol implementations in the VAX 11/730 and the Sun respectively. 4.3BSD-Reno is a new version of the UNIX system for VAX family of computers. It is released from the Computer System Research Group of the University of California at Berkeley in 1990. Comparing to the previous release 4.2BSD, 4.3BSD, and 4.3BSD-tahoe, 4.3BSD-Reno provides some support for the ISO OSI protocols CLNP, TP4, and End System to Intermediate System Routing Protocol (ESIS). Kernel support for the ISO OSI protocols is enabled with the ISO option in the kernel configuration file. The ISO OSI protocols TP4 and CLNP can be accessed through a socket system call. The more detailed information on the ISO socket is described in the Section 3.6.1.

SunOS Release 4.1.1 does not provides the ISO protocol family but only provides TCP/IP protocols of the Internet protocol family. It contains an interprocess

communication environment, such as socket-based communication facilities, communications in the internet or the resource sharing protocols, and access rights of a resource sharing system on a local area network. Also, it contains almost all AT&T System V functionality and features. Socket-based communication facilities are used to access TCP/IP internet protocols. The more detailed information on the Internet socket is described in the Section 3.6.2.

3.1.2 Interprocess Communication

The network programming of PACS requires the use of Interprocess Communication facilities (IPC) for the interaction of two or more processes. There are different facilities to provide the interaction between processes, such as pipes, message queues, semaphores, shared memory, sockets, and Transport Layer Interface (TLI). The main goals of these facilities are to allow multiprocess programming and to provide access to communication networks.

The pipe facility is a reliable, flow-controlled, byte stream that can be established between two processes on the same machine. A pipe is created by the *pipe* system call and provides a one-way flow of data. A *write* system call is used to send data through a pipe and a *read* system call is used to receive data from the pipe. Pipe facilities are available from all UNIX systems described in the previous Chapter. The three types of IPC, message queue, semaphores, and shared memory, are available in System V.

A socket is the abstract object which is created as an endpoint of communication and the focal point for IPC within a communication domain. Sockets may be created in pairs, used to rendezvous with other sockets in a communication domains, used to accept connections from these sockets, and used to send and receive messages between two processes on a different machine. Sockets are available in BSD and SunOS UNIX.

TLI facility of System V provides same functions as the socket facility of BSD. TLI provides an interface to the transport layer and was modeled following the ISO Transport Service Definition. System V Release 4.0 provides the interface to TCP/IP protocols.

3.2 Implementation Considerations

The PACS protocol is implemented with considerations of modularity, portability, and performance. The nature of the interlayer interfaces, the layer management, and overall resource management in a PACS should be implemented with a proper methodology. The correct implementation of PACS systems requires a substantial investment in understanding the standards specifications and assessing the implications of different design choices [SVO89].

3.2.1 Modularity

Services defined in Application, Presentation, and Session layers are implemented as independent procedures. Each service is available only through the service access point as defined in protocol specification. The modification of current service or addition of new service will affect only those services in the bordering layer. As an example, the service available in the transport layer will affect the interface element of session layer services.

3.2.2 Portability

The implemented PACS can be easily ported into other UNIX machines. If the operating system includes the Berkeley socket, there should be no trouble porting the program by just compiling the source cord. However, the porting to the host with System V requires

a little modification on the part of transport layer interface. Message queues, semaphores, and shared memory are not used because these generate problems in porting and the pipe facility can carry out required functions.

Communication between nodes of PACS should not depend on the machine and the operating system of the machine. The developed PACS protocol is based on the UNIX machine which is a very popular operating system. Thus, it is easily ported on the machine with a UNIX operating system.

3.2.3 Performance

The network programming of PACS is limited by its performance, because of large size of image data to transfer and expectations from impatient physicians. Particularly, following layered communication protocol specification requires more attention to performance. The protocol of each layer can be implemented as a separate process with using IPC facilities or as independent functions without using IPC facilities. Although building each layer as a process with IPC facilities provides modularity, it would have required more communication overhead to access more server processes of each layer.

Protocol overhead is the quantity of processing needed to accomplish transferring images, as related to the cost and time of the computing resource. To reduce the protocol overhead, it is implemented that the data passing is small at the boundary, number of interactions are minimized at the boundary, similar functions are built in the same layer, and normal data and image data are handled separately. Copying data to and from buffers between protocol layers in general represents significant overhead [CAB87]. An alternative is to pass service data units between layers by reference, provided the respective protocol entities have access to common memory. With this implementation, different vendors can insert localized functions easily without a major modification of implementation. A new technology or

requirement can be incorporated with a minor modification of interfacing layer. Furthermore, the vendor who tries to port this protocol implementation to their machine can port it only with the required portion of implementation. As an example, some imaging equipment which only generates images and does not need to receive images can be implemented without the receiving part of protocols.

3.3 Application Layer Implementation

The application layer is the top layer of seven layer model. It takes care of user interface and provides interface to the presentation layer. The shape of the application layer is defined based on application to provide a service requested by user. The connection control service element is a common service in the application layer, which provides connection /establishment, connection release, and connection abort between application peer entities.

3.3.1 Connection Control Service Facility

The connection control service facility includes connection establishment, connection release, and connection abort for the management of a single connection. The connection establishment service facility (connect) establishes a connection between a client application service facility and a server application service facility. The connection release service facility (disconnect) provides normal release of the established connection without loss of information in transmission. The connection abort service facility (abort) provides abnormal release of established connection with possible loss of information.

When the connection establishment service facility is activated in the client, it generates a connect packet with defined variables as described in Section 3.3.3. The connect packet

is sent to the server using the underlying layer, and the client is waiting for a conform from the server. The server, which was running as a daemon, receives the connect packet, generates a child server process, and waits for another connection as shown in Figure 3.7.

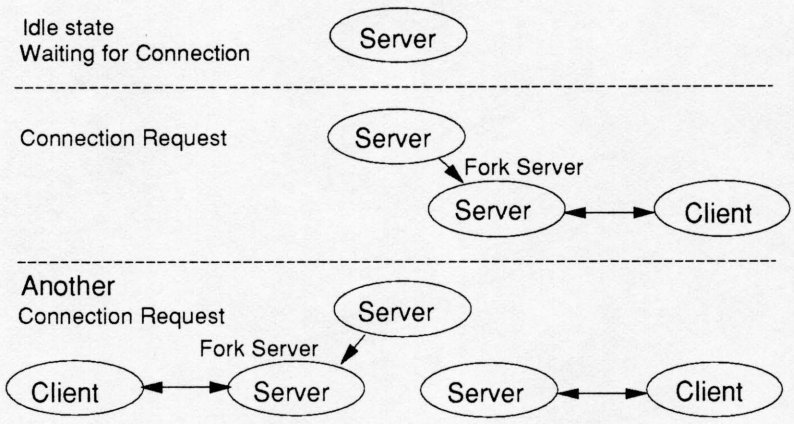

Figure 3.7. Server Activity for Connection Establishment

The child server process is independent from the parent process and kept alive until the connection is broken by the connect reject, disconnect, or connect abort. The child server process negotiates the received parameters with the connect packet and responds to the client with a connect accept packet or a connect reject packet. If the negotiation was successful, the child server responds with the connect accept packet and waits for another service request from client. But if the negotiation was unsuccessful, the child server sends the connect reject packet with a reason of rejection and exit from itself.

3.3.2 Other Application Service Facilities

The connection control service facility provides peer to peer connection for other PACS application service facility. The following application service facilities are implemented:

a) user service facility

b) get service facility

c) put service facility

d) iget service facility

e) iput service facility

f) ls service facility

g) cd service facility

h) exit service facility

The user service facility provides an user authentication service. The user service facility in the client side retrieves interactively user name and password from a user of the PACS. The user name and password is packetized and sent to connected server using presentation layer service. Then, it waits for the response of authentication from the server. The user service facility in the server side receives the user name and password from the underlying presentation layer. The received user name and password is checked from the password file. If the user is authorized to use PACS, an accept packet is sent to the client. If the user is not authorized, a reject packet is sent to the client.

The get service facility provides a text file retrieval from the text database server. The get service facility in the client side retrieves instructively local filename and remote filename from a user of the PACS. The get command and the remote filename are packetized and sent to connected server using presentation layer service. Then it opens local file and waits for the text data from the server. The get service facility in the server side receives the get command with remote filename from the underlying presentation

layer. The server opens a file with received filename, reads the file, and sends the packetized text data to client.

The put service facility provides a text file transfer to the text database server. The put service facility in the client side retrieves instructively local filename and remote filename from a user of the PACS. The put command and the remote filename are packetized and sent to connected server using presentation layer service. Then, it opens a local file, reads text data from the file, and sends the packetized text data to the server. The put service facility in the server side receives the put command with remote filename from the underlying presentation layer. The server opens a file with received filename and waits for the text data from the client. When the text data packet arrives, the server packetizes it and writes the text data to the file.

The iget service facility provides an image file retrieval from the image database server. It operates the same as the get service facility, but the handling of file and the access of underlying service are different. It uses a p-idata service instead of using a p-tdata service for the get service facility.

The iput service facility provides a image file transfer to the image database server. It operates the same as the get service facility, but the handling of the file and the access of underlying service are different. It uses a p-idata service instead of using a p-tdata service for the get service facility.

3.3.3 Application Header

The connect application protocol data unit (CN APDU) is defined as follows:

```
struct CNapduH {
        char    Mode;
        char    AContexName;
        char    CollingAPTitle[64];
        char    ColledAPTitle[64];
        char    VersionNo;
```

```
        char   IniSNum[6];
        char   TokenSet;
        char   SUserReq[2];
        char   CollingPAdd;
        char   ColledPAdd;
        char   UserData[512];
};
```

The CN APDU is used in the a-connect.request primitive and the a-connect.indication primitive. The mode parameter specifies the mode of the requested connection establishment. Values of the mode parameter are defined as 1 for PACS mode, 2 for normal mode, and 3 for X.410-1984. The default value is 1. The AContextName parameter identifies the application context symbolic name, which is proposed by the a-connect.request primitive. The default value is 1, which identifies the context style shown above. It uses the struct definition in the c language. The value 2 is reserved for ASN.1 and the value 3 is reserved for PSN, but the implementation does not provide these services. The CollingAPTitle parameter and the ColledAPTitle parameter identifies the application process name. The VersionNo parameter identifies the version of the implemented protocol. Other parameters are as defined in Section 3.4.2.

The connect response application protocol data unit (CR APDU) is defined as follows:

```
struct CRapduH {
        char   Mode;
        char   AContexName;
        char   Result;
        char   RespAPTitle[64];
        char   UserData[512];
};
```

The CR APDU is used in the a-connect.response primitive and the a-connect.conform primitive. The mode parameter specifies the mode of the accepted connection establishment. The AContextName parameter identifies the application context symbolic name, which is accepted in the server. The RespAPTitle parameter identifies the

responding application process name. The Result parameter identifies the result of connection request. The Result parameter values are 1 for accepted and 2 for rejected.

The disconnect application protocol data unit (DN APDU) and the disconnect response application protocol data unit (DR APDU) are defined as follows:

```
struct DNapduH {
        char   Reason;
        char   UserData[512];
};

struct DRapduH {
        char   Reason;
        char   UserData[512];
};
```

The DN APDU is used in the a-disconnect.request primitive and the a-connect.indication primitive. The DR APDU is used in the a-disconnect.response primitive and the a-disconnect.conform primitive. In the DN APDU, the reason parameter specifies the reason of the disconnection request. The parameter values are 1 for normal, 2 for urgent, and 3 for user defined. In the DR APDU, the reason parameter specifies the result of the disconnection request. The parameter values are 1 for normal, 2 for not finished, and 3 for user defined.

The abort application protocol data unit (AB APDU) is defined as follows:

```
struct ABapduH {
        char   AbortSource;
        char   UserData[512];
};
```

The AB APDU is used in the a-abort.request primitive and the a-abort.indication primitive. The AbortSource parameter specifies the source of the abort request. The parameter values are 1 for the service user and 2 for the service provider.

The data protocol data unit (DT APDU) is defined as follows:

```
struct DTapduH {
        char    EncItem;
};
```

The DT APDU is used in the a-data.request primitive and the a-data.indication primitive. The EncItem parameter specifies the item of the data transfer request. The parameter values are 1 for the test data and 2 for the image data.

The give token application protocol data unit (GT APDU) and the please token application protocol data unit (PT APDU) are defined as follows:

```
struct GTapduH {
        char    TokenItem;
};

struct PTapduH {
        char    TokenItem;
        char    UserData[512];
};
```

The GT APDU is used in the a-gtoken.request primitive and the a-gtoken.indication primitive. The PT APDU is used in the a-ptoken.request primitive and the a-ptoken.indication primitive. The TokenItem parameter specifies the item of the token enclosed or the token requesting. The parameter values are 1 for the text token, 2 for the image token, and 3 for all tokens. The optional value is 3.

The following PDUs are defined for the major synchronization point application protocol

data unit (MAP APDU), the major synchronization acknowledge application protocol data unit (MAA APDU), the resynchronize application protocol data unit (RE APDU), and the resynchronize acknowledge application protocol data unit (RA APDU).

```
struct MAPapduH {
        char    SyncType;
        char    SerialNo[6];
        char    UserData[512];
};

struct MAAapduH {
        char    SerialNo[6];
        char    UserData[512];
};

struct RSapduH {
        char    TokenSet;
        char    ResyncType;
        char    SerialNo[6];
        char    UserData[512];
};

struct RAapduH {
        char    TokenSet;
        char    SerialNo[6];
        char    UserData[512];
};
```

The MAP APDU is used in the a-msinc.request primitive and the a-msinc.indication primitive. The MAA APDU is used in the a-msinc.response primitive and the a-msinc.conform primitive. The RS APDU is used in the a-resync.request primitive and the a-resync.indication primitive. The RA APDU is used in the a-resink.response primitive and the a-resink.conform primitive. The SerialNo parameter specifies the serial number of the synchronization point. The SyncType parameter specifies the synchronization type. The TokenSet parameter specifies the item of the token enclosed or the token requested. The parameter values are 1 for the text token, 2 for the image token, and 3 for all tokens. The optional value is 3.

3.3.4 Forking Shell

The ls service facility and the cd service facility require an ability to execute system commands in the server. The UNIX provides a standard system command interpreter, shell, which executes system commands receiving from the terminal. The shell can be invoked through execve. When the server is activated, it generates a child process as shown in Figure 3.8.

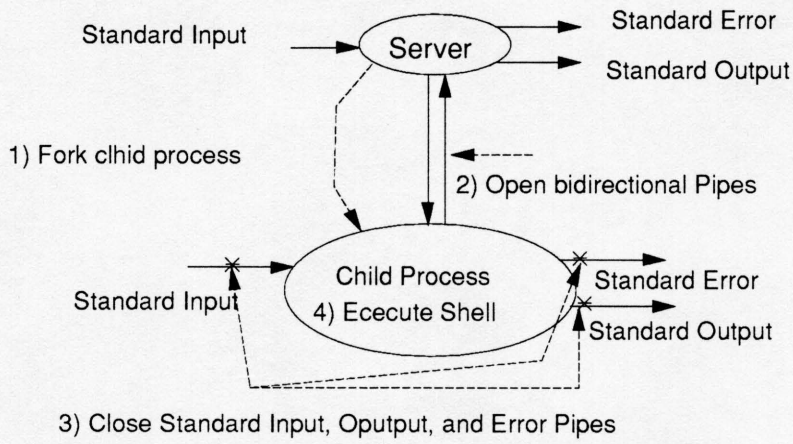

Figure 3.8. Forking Shell in the Server

The bidirectional pipe must be generated for interprocess communication between the server and its child process. Since the invoked shell uses its standard input, standard output, and standard error descriptors, the child process closes these three descriptors and uses the bidirectional pipe. The input pipe is used for the standard input and the output pipe is used for the standard output and standard error. Then the child process invokes shell through execve system call. The invoked shell receives input from the input pipe and generates output to the output pipe. Once the shell is invoked, it lives until the

connection is broken by some reason. The shell is floating around directories with an access permission of invoked user. It handles system calls arrived through the PACS network, while it is alive.

3.3.5 Interface to PSAP

An application service provider accesses a presentation service through a presentation service access point. Presentation service access points are functions which provide the requested presentation services in this implementation. Each primitive in the application service provider calls a corresponding primitive in the presentation layer. Every primitive has a one to one relationship except data transfer facility. An a-data.request calls a p-tdata.request or p-idata.request based on enclosed data type. Other corresponding primitives are as follows:

Application Layer	Presentation Layer
a-connect.request	p-connect.request
a-connect.indication	p-connect.indication
a-connect.response	p-connect.response
a-connect.conform	p-connect.conform
a-disconnect.request	p-disconnect.request
a-disconnect.indication	p-disconnect.indication
a-disconnect.response	p-disconnect.response
a-disconnect.conform	p-disconnect.conform
a-abort.request	p-abort.request
a-abort.indication	p-abort.indication
a-gtoken.request	p-gtoken.request
a-gtoken.indication	p-gtoken.indication
a-ptoken.request	p-ptoken.request
a-ptoken.request	p-ptoken.request
a-msink.request	p-msink.request
a-msink.indication	p-msink.indication
a-resink.request	p-resink.request
a-resink.indication	p-resink.indication

When an application service provider is trying to access a presentation service, it generates an application PDU and passes the application PDU through a presentation

service access point. It does not need to know the actual implementation of presentation service. It controls underlying service by using variables in application PDU. On the other hand, the presentation service provider removes a presentation control interface unit from a presentation PDU and generates an application PDU for passing data to the application service provider.

3.4 Presentation Layer Implementation

The presentation layer provides following functions to the presentation service user:

a) negotiation of transfer syntax

b) transform to/from transfer syntax

c) negotiation of image compression

d) compression and uncompression, if the negotiation was successful

e) negotiation of data encryption

f) encryption and decryption, if the negotiation was successful

These negotiations are done during the connection establishment as a variable negotiation between peer presentation layers. Variables of presentation protocol data units are described in Section 3.4.3. The main function of the presentation layer is to provide virtual information transfer which maintains the information of presentation data value during transfer. The other function is to provide interfaces between the application layer and the session layer.

3.4.1 Presentation Service Facilities

The connection control service in the presentation layer consists of the presentation connection establishment, the presentation connection release, and the presentation

connection abort. The presentation connection establishment service is provided by the p-connect facility. It is used to setup communication of two peer application entities by setting the required environment in the presentation layer. It is a conformed service which requests a session connection establishment service, negotiates and sets presentation environment, and exchanges initial user data.

The presentation connection release service is provided by the p-disconnect facility. It is used for an ordinary release of a connection between application entities. It is a conformed service which requests a session connection release service, discards negotiated presentation environment, and exchanges release user data.

The presentation connection abort service is provided by the p-abort-u facility and the p-abort-p facility. These are used for an abnormal release of the connection between application entities or the connection between presentation provider. The p-abort-u facility is activated by presentation service user and the p-abort-p facility is activated by presentation service provider. These are unconfirmed services which request a session connection release service with a reason of abort, discard negotiated presentation environment, and exchange abort user data.

The information transfer service consists the text transfer service, the image transfer service, the major synchronize service, the resynchronize service and the token service. The text transfer service is provided by the p-tdata facility. It is used for sending text data between application entities. The text data includes control information, patient information, and image related information. It converts the text data, which is received from application entity, to the transfer syntax. Then it passes the converted text data to the session text transfer service with a presentation header. The transfer syntax has the ACR-NEMA data format, which is negotiated by the presentation connection establishment.

The image transfer service is provided by the p-idata facility. It is used for sending image data between application entities. The image data includes pixel values of the image information. It converts the image data, which is received from application entity, to the transfer syntax. Then it passes the converted image data to the session image transfer service with presentation header. The transfer syntax has the ACR-NEMA data format, which is negotiated by the presentation connection establishment.

3.4.2 Presentation Header

The connect presentation protocol data unit (CN PPDU) is defined as follows:

```
struct CNppduH {
        char    CollingPAdd[64];
        char    ColledPAdd[64];
        char    PContextList[64];
        char    PContextRList[64];
        char    DefContext;
        char    QOS;
        char    PresReq;
        char    Mode;
        char    SessReq;
        char    IniSyncNo[6];
        char    IniToken;
        char    SessConnId;
        char    UserData[512];
};
```

The CN PPDU is used in the p-connect.request primitive and the p-connect.indication primitive. The CallingPAdd parameter specifies the presentation address which requests a connection establishment. The CalledPAdd parameter specifies the presentation address which is requested for a connection establishment. The PContextList parameter specifies the list of the proposed context which is used only in the p-connect.request primitive. The PContextRList parameter specifies the list of the negotiated context which is used only in the p-connect.indication primitive. The DefContext parameter specifies the default context

which is 1 for PACS context. The QOS parameter specifies the quality of service requested. Values of the QOS parameter are defined as 1 for TP4/CLNP and 2 for TCP/IP. The default value is 1. The mode parameter specifies the mode of the requested connection establishment. Values of the mode parameter are defined as 1 for PACS mode, 2 for normal mode, and 3 for X.410-1984. The default value is 1 and others are not supported yet. The PresReq parameter specifies the requirement of the presentation service requested. The SessReq parameter specifies the requirement of the session service requested. Other parameters are specified in Section 3.5.2.

The connect response presentation protocol data unit (CR PPDU) is defined as follows:

```
struct CRppduH {
        char    RespPAdd[64];
        char    PContextRList[64];
        char    DefContext;
        char    QOS;
        char    PresReq;
        char    SessReq;
        char    IniSyncNo[6];
        char    IniToken;
        char    SessConnId;
        char    Result;
        char    UserData[512];
};
```

The CR PPDU is used in the p-connect.response primitive and the p-connect.conform primitive. The RespPAdd parameter specifies the presentation address which responds to a connection request. The Result parameter specifies the result of a connection request. Other parameters are described in the CN PPDU parameter specification. Values of the Result parameter are defined as 1 for accepted, 2 for user rejection, and 3 for provider rejection.

The disconnect presentation protocol data unit (DN PPDU) and the disconnect response presentation protocol data unit (DR PPDU) are defined as follows:

```
struct DNppduH {
        char   UserData[512];
};

struct DRppduH {
        char   Result;
        char   UserData[512];
};
```

The DN PPDU is used in the p-disconnect.request primitive and the p-connect.indication primitive. The DR PPDU is used in the p-disconnect.response primitive and the p-connect.conform primitive. The Result parameter specifies the result of the disconnection request.

The abort presentation protocol data unit (AB PPDU) is defined as follows:

```
struct ABppduH {
        char   Reason;
        char   UserData[512];
};
```

The AB PPDU is used in the p-abort.request primitive and the p-abort.indication primitive. The AbortSource parameter specifies the source of the abort request. The parameter values are 1 for the service user and 2 for the service provider.

The data protocol data unit (DT PPDU) is defined as follows:

```
struct DTppduH {
        char   *UserData;
};
```

The DT PPDU is used in the p-tdata.request primitive and the p-tdata.indication primitive, the p-idata.request primitive, and the p-idata.indication primitive.

The give token presentation protocol data unit (GT PPDU) and the please token presentation protocol data unit (PT PPDU) are defined as follows:

```
struct GTppduH {
        char   Tokens;
};

struct PTppduH {
        char   Tokens;
        char   UserData[512];
};
```

The GT PPDU is used in the p-gtoken.request primitive and the p-gtoken.indication primitive. The PT PPDU is used in the p-ptoken.request primitive and the p-ptoken.indication primitive. The Tokens parameter specifies the item of the token enclosed or the token requesting. The parameter values are 1 for the text token, 2 for the image token, and 3 for all tokens. The optional value is 3.

The following PDUs are defined for the major synchronization point presentation protocol data unit (MAP PPDU), the major synchronization acknowledge presentation protocol data unit (MAA PPDU), the resynchronize presentation protocol data unit (RE PPDU), and the resynchronize acknowledge presentation protocol data unit (RA PPDU).

```
struct MAPppduH {
        char   SerialNo[6];
        char   UserData[512];
};

struct MAAppduH {
        char   UserData[512];
};

struct RSppduH {
        char   TokenSet;
        char   ResyncType;
        char   SerialNo[6];
        char   UserData[512];
};
```

```
struct RAppduH {
        char   TokenSet;      /* 26 */
        char   SerialNo[6];   /* 42 */
        char   UserData[512]; /* 193 */
};
```

The MAP PPDU is used in the p-msinc.request primitive and the p-msinc.indication primitive. The MAA PPDU is used in the p-msinc.response primitive and the p-msinc.conform primitive. The RS PPDU is used in the p-resync.request primitive and the p-resync.indication primitive. The RA PPDU is used in the p-resink.response primitive and the p-resink.conform primitive. The SerialNo parameter specifies the serial number of the synchronization point. The SyncType parameter specifies the synchronization type. The TokenSet parameter specifies the item of the token enclosed or the token requested. The parameter values are 1 for the text token, 2 for the image token, and 3 for all tokens. The optional value is 3.

3.4.3 Extended SN.PACS Database

The ISO-OSI standard has defined ASN.1 and BER to provide the common representation of information and the common generation of transfer data format. To generate ACR-NEMA data format and remove redundant bits generated by ISO transfer syntax, ASN.1 and BER are enhanced to generate ACR-NEMA data format from the PACS user data. This enhanced syntax notation is named as SN.PACS in the PACS protocol specification. It can generate ACR-NEMA transfer data format, but SN.PACS compiler is not implemented yet. Until SN.PACS compiler is built, direct data mapping method will be used. Direct data mapping method generates ACR-NEMA data format from C language style data syntax with the following rules:

a. It recognizes the *struct* definition as set.

b. It generates values of group ID and element ID for each primitive data.

c. It is applied only to user data.

d. Values of group ID and element ID are retrieved from Map database.

e. Length of each primitive data is calculated and inserted.

The following example shows the generation of transfer syntax from defined image information. Comments on the right side illustrate user data given by application entity.

```
struct ImageDataSet {
        struct   PATIENT_GROUP    PATIENT;
        struct   ACQUISITION_GROUP ACQUISITION;
        struct   PIXEL_GROUP      PIXEL;
};
struct PATIENT_GROUP {
    char   *PatientName;   /* Jiseung Nam */
    char   *PatientID;     /* 123-45-6789 */
    char   *PatientBirthday;/* 3.15.1959  */
    char   *InsurancePlanID;/* IEEE       */
};
struct ACQUISITION_GROUP {
    char   *ScanningSequence;/* 12        */
    char   *RepetitionTime; /* 13         */
};
struct PIXEL_GROUP {
    char   *PixelData;      /* 64Kbyte pixel data */
};
```

The transfer syntax of ACR-NEMA data format is generated as follows:

G ID	E ID	LENGTH	Transfer syntax	Contents of data
0010	0000	0000 0004		
0010	0010	0000 000B	4A49 5345 554E 4720 4E41 4500	/* PatientName */
0010	0020	0000 0004	3132 3334	/* PatientID */
0010	0030	0000 0009	332E 3135 2E31 3539	/* PatientBirthday */
0010	1050	0000 0004	4945 4545	/* InsurancePlanID */
0018	0000	0000 0004	0000 0004	
0018	0020	0000 0002	3132	/* ScanningSequence */
0018	0080	0000 0002	3133	/* RepetitionTime */
7F20	0000	0000 0004		
7F20	0010	XXXX XXXX	XXXX ... XXXX	/* 64Kbyte pixel data */

3.4.4 Interface to SSAP

A presentation service provider accesses a session service through a session service access point. Session service access points are functions which provide the requested session services in this implementation. Each primitive in the presentation service provider calls a corresponding primitive in the session layer. Every primitive in the presentation layer has a corresponding primitive in the session layer as follows:

Presentation Layer	Session Layer
p-connect.request	s-connect.request
p-connect.indication	s-connect.indication
p-connect.response	s-connect.response
p-connect.conform	s-connect.conform
p-disconnect.request	s-disconnect.request
p-disconnect.indication	s-disconnect.indication
p-disconnect.response	s-disconnect.response
p-disconnect.conform	s-disconnect.conform
p-tdata.request	s-tdata.request
p-tdata.indication	s-tdata.indication
p-idata.request	s-idata.request
p-idata.indication	s-idata.indication
p-abort.request	s-abort.request
p-abort.indication	s-abort.indication
p-gtoken.request	s-gtoken.request
p-gtoken.indication	s-gtoken.indication
p-ptoken.request	s-ptoken.request
p-ptoken.request	s-ptoken.request
p-msink.request	s-msink.request
p-msink.indication	s-msink.indication
p-resink.request	s-resink.request
p-resink.indication	s-resink.indication

When a presentation service provider is trying to access a session service, it generates a presentation PDU and passes the presentation PDU through a session service access point. It does not need to know the actual implementation of session service. It controls underlying service by using variables in presentation PDU. On the other hand, the session service provider removes a session control interface unit from a session PDU and generates a presentation PDU for passing data to the presentation service provider.

3.5 Session Layer Implementation

The session layer resides on top of the transport layer, which may be TCP or TP4. The main function of the session layer is a session control function which maintains the connection of session service users during data transfer. The other function is to provide interfaces between the presentation layer and the transport layer. Negotiations of session utilities are done during the connection establishment as a variable negotiation between peer session layers. Variables of session protocol data units are described in Section 3.5.2.

3.5.1 Session Service Facilities

The connection control service in the session layer consists of the session connection establishment, the session connection release, and the session connection abort. The session connection establishment service is provided by the s-connect facility. It is used to setup communication of two peer presentation entities by setting the required environment in the session layer. It is a conformed service which requests a session connection establishment service, negotiates and set session environment, and exchanges initial user data.

The session connection release service is provided by the s-disconnect facility. It is used for a ordinary release of a connection between presentation entities. It is a conformed service which requests a session connection release service, discards the negotiated session environment, and exchanges release user data.

The session connection abort service is provided by the s-abort facility. This is used for an abnormal release of the connection between presentation entities or the connection between the session provider. The s-abort facility can be activated by the session service user or session service provider. These are unconfirmed services which request a session

connection release service with a reason to abort, discard negotiated session environment, and exchange abort user data.

The information transfer service consists of the text transfer service, the image transfer service, the major synchronize service, the resynchronize service and the token service. The text transfer service is provided by the s-tdata facility. It is used for sending text data between presentation entities. The image transfer service is provided by the s-idata facility. It is used for sending image data between presentation entities. The s-tdata and s-idata facilities serialize the user data, which was received from presentation entity including the presentation header, application header. Then the serialized user data is passed to the transport data transfer service with session header.

3.5.2 Session Header

The connect session protocol data unit (CN SPDU) is defined as follows:

```
struct CNspduH {
        char   SPDUid;          /* 13 */
        char   CollingRef[64];  /* 10 */
        char   CommonRef[64];   /* 11 */
        char   AddRefInfo[4];   /* 12 */
        char   ProtcOpt;        /* 19 */
        char   TPDUmax[4];      /* 21 */
        char   VersionNo;       /* 22 */
        char   IniSNum[6];      /* 23 */
        char   TokenSet;        /* 26 */
        char   SUserReq[2];     /* 20 */
        char   CollingSSAPid;   /* 51 */
        char   ColledSSAPid;    /* 52 */
        char   UserData[512];   /* 193 */
};
```

The CN SPDU is used in the s-connect.request primitive and the s-connect.indication primitive. The SPDUid parameter specifies the CN SPDU identifier in one octet. The

SPDUid parameter value is defined as 13. The values of CallingRef, CommonRef, and AddRefInfo parameter are defined as an option by the calling session service user. The ProtcOpt parameter specifies whether or not the requester is able to concatenate SPDUs. Values of the parameter are defined as 1 for "able to" and 0 for "not able to." The default value is 0. The concatenation of SPDU is not yet implemented. The TPDUmax parameter specifies the proposed TPDU maximum size, if the use of segmentation is requested. If the TPDUmax parameter is absent, there is no segmentation of SSDUs. The VersionNo parameter specifies the version of PACS protocol implementation which is 1 for this PACS protocol implementation. The IniSNum parameter specifies the initial serial number of synchronization point if the major synchronize functional unit was proposed. The TokenSet parameter, if present, specifies the initial position of token with one octet. The bit 1 of TokenSet parameter is set for the text token. The bit 2 is set for the image token. The bit 3 is set for both tokens. The bit 4 and 8 are reserved for future. The bit 5 is set for the initiater's side. The bit 6 is set for the responder's side. The bit 7 is set for the called session service user's choice. The SUserReq parameter specifies the requirement of the session service user proposed by the calling session service user. The bits in the SUserReq parameter value indicate the functional units proposed by the calling session service user. Kernel functional unit is always proposed without setting any bit. Half-duplex functional unit is proposed by setting bit 1. Duplex functional unit is proposed by setting bit 2. Major synchronize functional unit is proposed by setting bit 5. Resynchronize functional unit is proposed by setting bit 6. Negotiated release functional unit is proposed by setting bit 8. Other bits are reserved for future implementation. The CollingSSAPid and ColledSSAPid parameters, if present, are supplied by the calling session service user.

The accept session protocol data unit (AC SPDU) is defined as follows:

```
struct ACspduH {
        char    SPDUid;             /* 14 */
        char    ColledRef[64];      /* 9  */
        char    CommonRef[64];      /* 11 */
        char    AddRefInfo[64];     /* 12 */
        char    ProtcOpt;           /* 19 */
        char    TPDUmax[4];         /* 21 */
        char    VersionNo;          /* 22 */
        char    IniSNum[6];         /* 23 */
        char    TokenSet;           /* 26 */
        char    TokenItem;          /* 16 */
        char    SUserReq[2];        /* 20 */
        char    CollingSSAPid;      /* 51 */
        char    ColledSSAPid;       /* 52 */
        char    UserData[512];      /* 193 */
};
```

The AC SPDU is used in the s-connect.response primitive and the s-connect.conform primitive to notify the acceptance of connection request. The SPDUid parameter specifies the AC SPDU identifier in one octet. The SPDUid parameter value is defined as 14. The values of CallingRef, CommonRef, and AddRefInfo parameter are defined as an option by the called session service user. The ProtcOpt parameter specifies whether or not the responder is able to concatenate SPDUs. The TokenItem parameter, if present, specifies which tokens are requested by the called session service user. The bit 1 of TokenSet parameter is set for the text data token. The bit 2 is set for the image data token. The bit 3 is set for both token. Other bits are reserved for future. The ColledSSAPid parameter, if present, are supplied by the called session service user. Other parameters are described in the CN SPDU parameter specification.

The refuse session protocol data unit (RF SPDU) is defined as follows:

```
struct RFspduH {
        char    SPDUid;             /* 12 */
        char    ColledRef[64];      /* 9  */
        char    CommonRef[64];      /* 11 */
        char    AddRefInfo[4];      /* 12 */
        char    TDisconn;           /* 17 */
        char    SUserReq[2];        /* 20 */
```

```
        char   VersionNo;          /* 22 */
        char   Reason;             /* 50 */
};
```

The RF SPDU is used in the s-connect.response primitive and the s-connect.conform primitive to notify the refuse of connection request. The SPDUid parameter specifies the RF SPDU identifier in one octet. The SPDUid parameter value is defined as 12. The TDisconn parameter specifies whether or not the transport connection is to be kept. Values of the TDisconn parameter are defined as 0 for the transport connection is kept alive and 1 for the transport connection is closed. The Reason parameter specifies the result of a connection refuse. Other parameters are described in the AC SPDU parameter specification. Values of the Reason parameter are defined as 1 for user rejection and 2 for provider rejection.

The disconnect session protocol data unit (DN SPDU), the finished session protocol data unit (FN SPDU), and the not finished session protocol data unit (FN SPDU) are defined as follows:

```
struct DNspduH {
        char   SPDUid;             /* 10 */
        char   UserData[512];      /* 193 */
};

struct FNspduH {
        char   SPDUid;             /* 9  */
        char   TDisconn;           /* 17 */
        char   UserData[512];      /* 193 */
};

struct NFspduH {
        char   SPDUid;             /* 8  */
        char   UserData[512];      /* 193 */
};
```

The DN SPDU is used in the s-disconnect.request primitive and the s-connect.indication primitive. The NF SPDU and FN SPDU are used in the s-disconnect.response primitive and the s-connect.conform primitive. The DN SPDUid parameter value is defined as 10. The FN SPDUid parameter value is defined as 9 and notifies the acceptance of disconnection request. The NF SPDUid parameter value is defined as 8 and notifies the rejection of disconnection request. The TDisconn parameter specifies whether or not the transport connection is to be kept as in RF SPDU. Values of the TDisconn parameter are defined as 0 for the transport connection is kept alive and 1 for the transport connection is closed.

The abort session protocol data unit (AB SPDU) is defined as follows:

```
struct ABspduH {
        char    SPDUid;             /* 25 */
        char    TDisconn;           /* 17 */
        char    ReflectPV[9];       /* 49 */
        char    UserData[512];      /* 193 */
};
```

The AB SPDU is used in the s-abort.request primitive and the s-abort.indication primitive. The SPDUid parameter specifies the AB SPDU identifier in one octet. The SPDUid parameter value is defined as 25. The TDisconn parameter specifies whether or not the transport connection is to be kept as in RF SPDU. The ReflectPV parameter specifies the source of the abort request. The parameter values are 1 for the service user and 2 for the service provider.

The data protocol data unit (DT SPDU) is defined as follows:

```
struct DTspduH {
        char    SPDUid;             /* 1  */
        char    EncItem;            /* 25 */
};
```

The DT SPDU is used in the s-tdata.request primitive and the s-tdata.indication primitive, the s-idata.request primitive, and the s-idata.indication primitive. The SPDUid parameter specifies the DT SPDU identifier in one octet. The SPDUid parameter value is defined as 1. The EncItem parameter specifies the type of enclosed data. The bit 1 of EncItem parameter is set for the beginning of SSDU. The bit 2 is set for the end of SSDU. The bit 3 is set for the text data. The bit 4 is set for the image data. Other bits are reserved for future.

The give token session protocol data unit (GT SPDU) and the please token session protocol data unit (PT SPDU) are defined as follows:

```
struct GTspduH {
        char    SPDUid;             /* 1  */
        char    TokenItem;          /* 16 */
};

struct PTspduH {
        char    SPDUid;             /* 2  */
        char    TokenItem;          /* 16 */
        char    UserData[512];      /* 193 */
};
```

The GT SPDU is used in the s-gtoken.request primitive and the s-gtoken.indication primitive. The PT SPDU is used in the s-ptoken.request primitive and the s-ptoken.indication primitive. The GT SPDUid parameter value is defined as 1. The PT SPDUid parameter value is defined as 2. The TokenItem parameter specifies the item of the token enclosed or the token requesting. The parameter values are 1 for the text token, 2 for the image token, and 3 for all tokens. The optional value is 3.

The following PDUs are defined for the major synchronization point session protocol

data unit (MAP SPDU), the major synchronization acknowledge session protocol data unit (MAA SPDU), the resynchronize session protocol data unit (RE SPDU), and the resynchronize acknowledge session protocol data unit (RA SPDU).

```
struct MAPspduH {
        char   SPDUid;        /* 41 */
        char   SyncType;      /* 15 */
        char   SerialNo[6];   /* 42 */
        char   UserData[512]; /* 193 */
};

struct MAAspduH {
        char   SPDUid;        /* 42 */
        char   SerialNo[6];   /* 42 */
        char   UserData[512]; /* 193 */
};

struct RSspduH {
        char   SPDUid;        /* 53 */
        char   TokenSet;      /* 26 */
        char   ResyncType;    /* 27 */
        char   SerialNo[6];   /* 42 */
        char   UserData[512]; /* 193 */
};

struct RAspduH {
        char   SPDUid;        /* 34 */
        char   TokenSet;      /* 26 */
        char   SerialNo[6];   /* 42  */
        char   UserData[512]; /* 193 */
};
```

The MAP SPDU is used in the s-msinc.request primitive and the s-msinc.indication primitive. The MAA SPDU is used in the s-msinc.response primitive and the s-msinc.conform primitive. The RS SPDU is used in the s-resync.request primitive and the s-resync.indication primitive. The RA SPDU is used in the s-resink.response primitive and the s-resink.conform primitive. The MAP SPDUid parameter value is defined as 41. The MAA SPDUid parameter value is defined as 42. The RS SPDUid parameter value is defined as 53. The RA SPDUid parameter value is defined as 43. The SerialNo

parameter specifies the serial number of the synchronization point. The SyncType parameter specifies the synchronization type. The parameter values are 0 for the resynchronize restart, 1 for the resynchronize abandon, and 2 for the resynchronize set. The TokenSet parameter specifies the item of the token enclosed or the token requested as in the CN SPDU.

3.5.3 Session Handling for Image Transfer

An image transfer requires proper session handling utilities which are provided in the session layer. Requests of session handling are generated by the session service user and processed by the session service provider. To process those requests, the session service provider must use transport services. The access of transport service can be achieved by the socket interface. The following diagram shows the state change and the possible action using the socket interface in the client side.

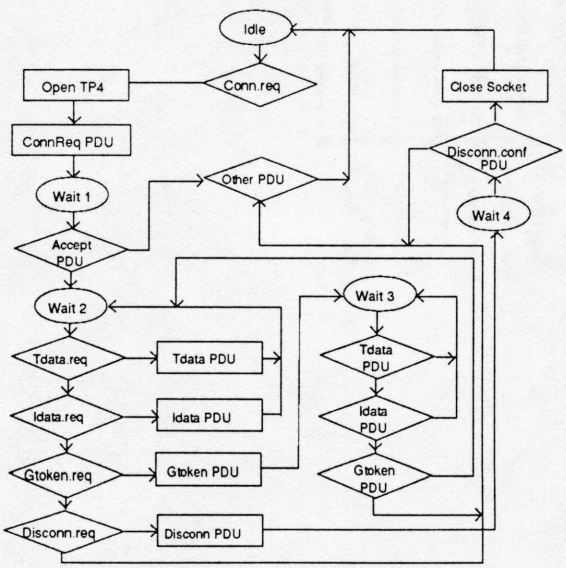

Figure 3.9. Session Handling in the Client Side

When the session service provider is activated in the client side, it is placed in the idle state and waits for an activation of s-connect.request primitive by the session service user. Upon receiving the s-connect.request, it opens a TP4 socket or a TCP socket based on the request variable. The socket is opened based on the network address and the port number from the calling session address parameter. When the socket is opened, the session service provider attaches itself to the socket. Then it sends a connection request PDU through the socket and the idle state is changed to the wait 1 state. If the connection request was successful and the accept PDU was received from the socket, the wait 1 state is changed to the wait 2 state. If the reject PDU was received, the state is changed to the idle state. If any other PDU was received, the provider generates a provider initiated abort PDU and goes to the idle state.

The session service provider is ready to send data in the wait 2 state, since a token is initialized to be located in the client side. Until it receives s-disconnect.request or s-token.request from the session service user, it sends text data PDUs or image data PDUs based on request from the user. If the s-disconnect.request has arrived, the provider sends a disconnect PDU through the socket and goes to the wait 4 state. If the s-token.request is arrived, the provider sends give token PDU through the socket and goes to the wait 3 state. The session service provider is ready to receive data in the wait 3 state. Until it receives a give token PDU from the session service user, it receives text data PDUs or image data PDUs from the socket. If it receives the give token PDU, it goes back to the wait 2 state. Upon receiving a disconnect PDU, the session service provider closes the socket and it goes to the idle state.

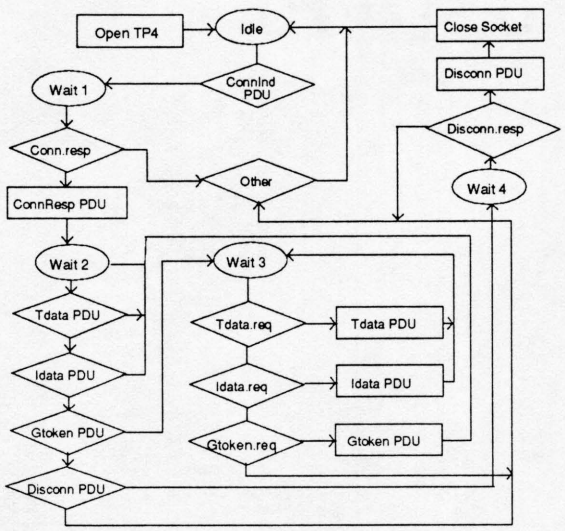

Figure 3.10. Session Handling in the Server Side

When a session service provider in a server side is activated, it opens a passive TP4 socket or a passive TCP socket. The socket is opened based on the network address of a local host, the selected port number of the local host, and an underspecified network address of a remote host. When the socket is opened, the session service provider attaches itself to the socket. Then it is placed in the idle state and wait for an arrival of a connection request PDU through the opened socket. Upon receiving the connection request PDU, it generates a p-connect.indication to a session service user and the idle state is changed to the wait 1 state. In the wait 1 state, the service provider waits for a s-connect.response. If the s-connect.reponse was received and the negotiation of the connection request PDU was successful, the wait 1 state is changed to the wait 2 state with sending a connection accept PDU. If the negotiation was not successful, the reject PDU is set and the wait 1 state goes back to the idle state. If any other PDU was received in the idle state, the provider generates a provider initiated abort PDU and stays in the idle state.

The session service provider is ready to receive data in the wait 2 state, since a token is initialized to be located in the client side. Until it receives a disconnect PDU or a token request PDU from the connect socket, it receives text data PDUs or image data PDUs based on arrival from the socket. If the disconnect PDU has arrived, the provider sends a p-disconnect.indication to the service user and waits for a s-disconnect.response in the wait 4 state. If the token request PDU is arrived, the provider generates p-token.indication to the service user and goes to the wait 3 state. The session service provider is ready to send data in the wait 3 state. Until it receives a s-token.request from the session service user, it sends text data PDUs or image data PDUs from the service user to the socket. If it receives the s-token.request, it goes back to the wait 2 state. Upon receiving the s-disconnect.response, the session service provider sends a disconnect response PDU through the socket, closes the socket, and goes to the idle state.

3.5.4 Interface to TP4/CLNP

The service user of transport layer protocols in the session layer may access a TP4/CLNP service through a socket system call. The socket, which requests TP4 connection, is called an active TP4 socket. When a TP4 socket is created by the socket system call, it is created as the active TP4 socket. The connect system call is used to request connection in the active TP4 socket. The connect system call actually establishes a connection between a client in the local system and a server in the remote system. The connect system call can be used with an address. The address assigned to the socket is the TP4/CLNP address, which is associated with the network interface through which packets are being transmitted and received. After successful connection establishment, TP4 sockets are normally used with the sendto calls to send packet and recvfrom calls to receive packet.

The socket, which responds to connection request, is called a passive TP4 socket. Since

a TP4 socket is created as an active TP4 socket by default, the listen system call must be used after binding the socket with the bind system call to create a passive TP4 socket. The bind system call assigns a name to the socket. At this time, servers register their defined address to the system before receiving any data. Passive TP4 sockets underspecify their location to match incoming connection requests from multiple networks. This technique allows a single server to provide service to multiple clients. Then the passive socket uses the accept call to accept incoming connections. Once a connection has been established, the TP4 socket's address is fixed with two addresses of client and server.

3.5.5 Interface to TCP/IP

The service user of transport layer protocols in the session layer may access a TCP/IP service through a socket system call. The socket, which requests connection, is called an active TCP socket. When a TCP socket is created by the socket system call, it is created as the active TCP socket. The connect system call is used to request connection in the active TCP socket. The connect system call actually establishes a connection between a client in the local system and a server in the remote system. The connect system call can be used with an address. The address assigned to the socket is the TCP/IP address, which is associated with the network interface unit. After successful connection establishment, TCP sockets are normally used with the write calls to send packet and read calls to receive packet.

The socket, which responds to connection request, is called a passive TCP socket. Since a TCP socket is created as an active TCP socket by default, the listen system call must be used after binding the socket with the bind system call to create a passive TCP socket.

Passive TCP sockets underspecify their location to match incoming connection requests from multiple networks. This technique allows a single server to provide service to multiple clients. Then the passive TCP socket uses the accept call to accept incoming connections.

3.6 Transport and Network Layer

3.6.1 ISO (TP4/CLNP) Socket

The TP4/CLNP implementation for the 4.3BSD-reno unix was developed at The University of Wisconsin-Madison. It was modified to include in the Berkeley Software Distribution and to add the socket interface by The University of California, Berkeley. TP4 protocol provides the reliable end-to-end data transfer, the sequencing of data packets, and the control of data flow. Also, it allows two-way transmission of data packets with an alternate stop-and-wait. CLNP is the connectionless-mode network protocol which provides connectionless-mode network service. TP4/CLNP protocol service is accessed through socket interface with the AF_ISO option and the SOCK_SEQPACKET abstraction as follows.

```
s = socket( AF_ISO, SOCK_SEQPACKET, 0);
```

A TP4/CLNP socket operation fails when trying to establish a connection on a socket which has already established a connection, when trying to send a datagram without destination address, when the system runs out of memory, when trying to create a socket with a network address which is not supported, when trying to send a datagram to the destination address which does not exist, and when trying to use unsupported options. If the TP4 entity encounters asynchronous events that will cause a transport connection to be closed, the TP4 entity issues a signal, indicating that disconnection has occurred.

The TP4/CLNP protocol uses the ISO address format, which is defined on ISO 8348/AD2, "Addendum to the Network Service Definition Covering Network Layer Addressing." The TP4/CLNP protocol uses a standard ISO address format, including a network service access point, and a transport service entity selector. TP4 may include the internet Internet address format in ISO address format. The TP4/CLNP socket has the following address structure:

```
struct iso_addr {
  u_char isoa_len;                    /* length, not including this byte */
  char isoa_genaddr[20];              /* ISO network address */
};

struct sockaddr_iso {
      u_char   siso_len;              /* size of sockaddr structure */
      u_char   siso_family;           /* addressing domain of socket */
      u_char   siso_plen;             /* length of presentation */
      u_char   siso_slen;             /* length of session selector */
      u_char   siso_tlen;             /* length of transport selector */
      struct  iso_addr siso_addr;     /* network address pointer */
      u_char   siso_pad[6];           /* reserved for gosip */
};
```

The siso_len field identifies the length of the entire address structure in bytes. The siso_family field defines the domain of socket. The siso_plen field indicates the length of the presentation selector. The siso_slen field indicates the length of the session selector. The siso_tlen field indicates the length of the transport selector. The siso_addr field defines

the network part of the address. ISO network addresses consists of 20 bytes in length. ISO network addresses can take the format defined in Section 2.6.3.1.

The protocol implementation of TP4/CLNP provides several flags to control negotiable options in the protocol. Options are sending connection request with/without user data, sending disconnection request with/without user data, sending user data with/without segmentation, and sending user data with/without the CLNP checksum. The setsockopt system call is used to set options. Once an option has been set, it will be valid until a different option is set. A TP4 header and a CLNP header are added automatically to outgoing packets. Incoming packets are received with those added headers.

3.6.2 Internet (TCP/IP) Socket

The TCP/IP implementation for the BSD unix was developed by Bolt, Beranek, and Newman in 1981. Later, The University of California, Berkeley added the socket interface. TCP protocol is a byte-stream protocol which provides the same function as TP4. However, the header part of TCP is not flexible as TP4. It allows the reliable end-to-end data transfer, the sequencing of data packets, the control of data flow, and two-way transmission of data packets. IP is the connectionless-mode network protocol which provides connectionless-mode internetworking service. TCP/IP protocol service is accessed through socket interface with the AF_INET option and the SOCK_STREAM abstraction as follows.

```
s = socket( AF_INET, SOCK_STREAM, 0);
```

An TCP/IP transport address is similar to an ISO address in that it contains a network-address portion and a transport selector portion. The transport selector, which is

called as a port in internet domain, is used to multiplex transport services among transport service users. 32 bits are reserved for the network-address and 16 bits are reserved for the port. Generally, the last 16 bits of the network address are used for the local area network address and for the host address. The TCP/IP socket has the following address structure:

```
struct in_addr {
 u_long s_addr; /* 32 bit network address */
};

struct sockaddr_in {
    short   sin_family;     /* addressing domain of socket */
    u_short  sin_port;      /* 16 bit port number */
    struct   in_addr sin_addr; /* network address pointer */
    u_char   sin_zero[8];    /* reserved for future */
};
```

The sin_family field identifies the domain of socket. The sin_port field identifies the port number of the transport layer. The sin_addr field indicates a pointer which contains the location of a 32 bit network address. The protocol implementation of TCP/IP provides negotiable options which can be set using setsockopt. Once an option has been set, it will be valid until a different option is set. A TCP header and a IP header are added automatically to outgoing packets. Incoming packets are received with those added headers. A TCP/IP socket operation fails when trying to establish a connection on a socket which has already established a connection, when trying to send a datagram without destination address, when

the system runs out of memory, when trying to create a socket with a port which has already been allocated, when trying to create a socket with a network address which does not exist, and when trying to use unsupported options.

3.6.3 BSD Kernel Rebuilding for ISO Socket

Installation of 4.3BSD-reno does not provides ISO socket until the kernel of 4.3BSD-reno is rebuilt with including ISO option. The system rebuilding process, which builds a bootable system image, takes the following six steps: creating a system configuration file, making a directory for the system to be constructed in, running the system configuration file, constructing the source code interdependency rules, compiling the source codes, and switching the kernel.

To rebuild the system with an ISO option, the system configuration file must include the system building information which are machine type, CPU type, system identification, timezone, maximum number of users, location of root file system, and available hardware. Running the system configuration file generates the files which are required to compile including ISO TP4/CLNP and to load the system image. Constructing the interdependency rules will insure that any changes to a piece of system source code will result in the recompilation of associated modules. After compiling and loading source codes, the created kernel is moved to root directory and named as vmunix. Then, the ISO socket is available with a proper network setup.

CHAPTER 4

PERFORMANCE TESTING

The environment of PACS is differs somewhat from the other computer network environments. Basically, the purpose of PACS is to provide better diagnostic care to the patients and to provide better handling of image by interconnecting the viewing workstations, imaging equipment, and database archives systems. The performance of PACS networks has a great effect on the acceptability of the PACS by the medical community. Not only is the quantity of data for image transfer very large, but also the quality of data for image transfer is extremely sensitive in the PACS environment. The parameters of performance on PACS are availability, response time, throughput, reliability, channel utilization, and security. In this chapter, we present the results of performance testing of the TCP/IP and ISO protocol implementations described in Chapter 3.

The Sparc SUN workstation with SEGATE 1.2 Gbyte disk is used for the protocol development and testing on the top of TCP/IP. The processing speed of Sparc SUN workstation is 12.5 Mega Instructions Per Second. The VAX 11/730 with DEC R80 120 Mbyte disk is used for the protocol development and testing on the top of TP4/CLNP. The processing speed of VAX 11/730 is about 0.3 Mega Instructions Per Second. Actually, the PACS protocols are developed on the SUN station and ported into VAX for the development of full stack of protocol.

4.1 Response Time over TCP/IP

Response times in Figure 4.2 are time delays of file transfer which are tested between two SUN workstations. This end-to-end scenario includes two separate tests. The time delay of disk-to-disk file transfer and the time delay of memory-to-memory file transfer are tested.

The disk-to-disk response time is tested as shown in Figure 4.1.

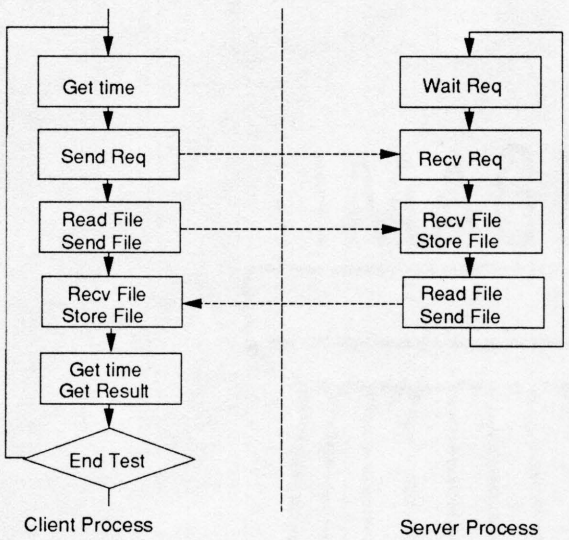

Figure 4.1. Response Time for Disk-to-disk File Transfer.

The client operates as follows to get the response time for the disk-to-disk file transfer.

1. Get an initial time (t1) of a file transfer.
2. Send a test request message to a server.
3. Read a file from disk and send to the server.

4. Receive file from the server and write to disk.
5. Get a final time (t2) of the file transfer.
6. Get response time (t3 = t2 - t1) and store in a result file.
7. Go to step 1.

The server operates as follows to get the response time for the disk-to-disk file transfer.

1. Wait for a test request message from a client.
2. Receive a test request from the client.
3. Receive a file from the client and store to disk.
4. Read a file from disk and send to client.
5. Go to step 1.

The response time of memory-to-memory file transfer is tested in the same way without reading file from disk and writing file to disk.

File Size (Byte)	Disk -to-disk			Memory -to-memory		
	Avgerage (second)	Minimum (second)	Standard Deviation	Avgerage (second)	Minimum (second)	Standard Deviation
8192	0.199	0.149	0.096	0.146	0.119	0.062
32768	0.369	0.299	0.099	0.180	0.130	0.076
65536	0.789	0.719	0.084	0.330	0.260	0.109
131072	1.468	1.339	0.162	0.540	0.470	0.079
262144	4.739	2.789	0.408	1.370	1.120	0.592

Figure 4.2. Response Time of File Transfer (SUN, TCP/IP).

The test is repeated by 100 times at midnight to reduce the number of users on the network and the testing host. The average response time, the minimum response time, and the standard deviation are derived from the test. The average is the most common measure of central tendency of the data. If the observations in a testing of response time with size n are $x_1, x_2, \ldots, x_n$, then the average response time is

$$\bar{x} = \frac{x_1 + x_2 + \ldots + x_n}{n} = \sum_{i=1}^{n} \frac{x_i}{n}$$

The measure of dispersion, called the standard deviation, is defined as

$$s = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n}}$$

Where $x_1, x_2, \ldots, x_n$ is a response time of n observations, $\bar{x}$ is an average response time of n observations, and n is 100. The result does not show best response time of the protocol performance, because the network was connected to numerous hosts in the University and several daemons were running in the testing host. But it shows acceptable results of the performance for the PACS system in more realistic situation. The response time of the Disk-to-Disk file transfer shows the excessive time delay compared to the memory-to-memory file transfer.

The maximum theoretical throughput on a 10 Mbit/second Ethernet using TCP/IP is 1,177,606 bytes/second [STE90]. This assumes the 24 byte times of interpacket gab, the 22 bytes Ethernet header, the 20 bytes IP header, the 20 bytes TCP header, the 1464 bytes user data, and the 4 bytes Ethernet tailer. It also assumes only a single host is transmitting packets on the Ethernet. The maximum throughput derived from the result in Figure 4.2 is 278,876 bytes/second. It is the test result of the memory to memory file transfer with a 131,072 bytes file. The tested throughputs are less then 25% of the theoretical value.

The performance loss is caused by the Ethernet interface hardware and software, the IP layer, the TCP layer, the session layer, the presentation layer, and the application layer. A paper shows performance measurements on an actural Ethernet. It demonstrates the 15% loss from the maximum throughput only for the Ethernet interface hardware and software [BOG88]. Another paper on network performance shows measurements of a maximum TCP throughput under 4.2BSD in an Ethernet environment. When the test was done in VAX 11/750, the result showed less than 10% of the maximum theoretical throughput [CAB88].

4.2 Time Delay in Presentation and Session Layer

The time delay in the presentation layer and the session layer is tested separately from the previous test, because the simultaneous test possibly generates the more overhead. There are utilities which gives the information about each called routines in UNIX. These utilities are called prof and gprof which produces an execution profile of a program. The execution profile contains the percentage of time spent executing between that function and others, the number of times that function was called, and the number of milliseconds per call.

The profile data is taken from the profile file. The profile file is produced automatically by programs compiled with the 'cc -p' option or 'cc -pg' option in UNIX. This option links library functions which are compiled for profiling and also generates the execution profile of those library functions. There is a possible overhead to testing performance using these utilities, but it is assumed that the amount of overhead can be ignored in this testing. The following test result shows the time spent in the presentation layer and session layer to transfer 2 kbytes data.

Table 4.2. Profile output of time delays in protocol layers.

% total	cumulative seconds	self seconds	calls	self ms/call	ms/call	name
19.8	6.08	2.36	31	76.13	76.13	_TTDataReq
15.6	7.94	1.86	20490	0.09	0.09	_fgetc
13.9	9.60	1.66	10	166.00	511.56	_AFTDataReq
12.7	11.11	1.51	10	151.00	171.74	_AFTDataInd
0.8	11.52	0.09	300	0.30	0.33	_malloc
0.3	11.69	0.04	64	0.63	0.63	_write
0.3	11.72	0.03	44	0.68	0.68	_strcmp
0.2	11.74	0.02	42	0.48	1.02	_flsbuf
0.2	11.76	0.02	22	0.91	3.27	_STDataInd
0.2	11.78	0.02	22	0.91	1.31	_TTDataInd
0.1	11.79	0.01	145	0.07	0.07	_bcopy
0.1	11.80	0.01	138	0.07	0.07	_sigaction
0.1	11.81	0.01	136	0.07	0.07	_sigprocmask
0.1	11.82	0.01	65	0.15	0.15	_sigsuspend
0.1	11.83	0.01	32	0.31	0.88	_filbuf
0.1	11.84	0.01	32	0.31	0.31	_read
0.1	11.85	0.01	31	0.32	77.92	_PTDataReq
0.1	11.86	0.01	22	0.45	4.39	_PTDataInd
0.1	11.87	0.01	22	0.45	0.45	_findiop
0.1	11.88	0.01	22	0.45	0.91	_fopen
0.1	11.89	0.01	13	0.77	0.77	_sbrk
0.1	11.90	0.01	12	0.83	9.38	_fprintf
0.1	11.91	0.01	10	1.00	1.79	_gets
0.0	11.93	0.00	298	0.00	0.00	_free
0.0	11.93	0.00	195	0.00	0.00	_setitimer
0.0	11.93	0.00	130	0.00	0.07	_sigvec
0.0	11.93	0.00	85	0.00	0.00	_gettimeofday
0.0	11.93	0.00	69	0.00	0.07	_sigsetmask
0.0	11.93	0.00	67	0.00	0.07	_sigblock
0.0	11.93	0.00	65	0.00	0.15	_sigpause
0.0	11.93	0.00	65	0.00	0.45	_sleep
0.0	11.93	0.00	56	0.00	0.38	_fflush
0.0	11.93	0.00	38	0.00	8.54	_printf
0.0	11.93	0.00	32	0.00	0.00	_sendmsg
0.0	11.93	0.00	31	0.00	76.93	_STDataReq
0.0	11.93	0.00	24	0.00	0.00	_fstat
0.0	11.93	0.00	23	0.00	0.00	_recvmsg
0.0	11.93	0.00	22	0.00	0.00	_close
0.0	11.93	0.00	22	0.00	0.38	_fclose
0.0	11.93	0.00	22	0.00	0.00	_open
0.0	11.93	0.00	21	0.00	0.00	_bcmp
0.0	11.93	0.00	17	0.00	0.00	_strlen

0.0	11.93	0.00	11	0.00	78.26	_ACTDataReq
0.0	11.93	0.00	11	0.00	0.77	_morecore
0.0	11.93	0.00	8	0.00	0.07	_signal

The first column shows the percentage of the total running time of the program used by this function. The second cumulative column shows a running sum of the number of seconds accounted for this function and those listed above it. The third column shows the number of seconds accounted for this function alone. The forth column shows the number of times this function was invoked. The fifth column shows the average number of milliseconds spent in this function per call. The sixth column shows the average number of milliseconds spent in this function and its descendents per call. The last column is the name of the function.

4.3 Comparison between TCP/IP and TP4/CLNP

Response times over TP4/CLNP are tested between two processors in the same VAX 11/730. Since the client process and the server process are located in the same host, the packets from the client process are looped back without going to the Ethernet cable. The BSD4.3-reno operating system is installed in the VAX 11/730. When the network is configured, the CLNP network address is bound to the address of the Ethernet interface. In this configuration, the packet from the sending process passes down to the CLNP network layer and the Ethernet driver software actually does see this packet. But the Ethernet driver recognizes its own Ethernet address and send the packet back to the network layer without bothering to transmit it. As a matter of fact, the Ethernet driver just takes the packet and hands it over to the loop-back driver, which loops all packets going out back to itself and passes the packet back up. The time delay of disk-to-disk file transfer is tested in this environment with a same way in Section 4.1.

Response times over TCP/IP are tested between two processors in the same VAX 11/730 to compare with results over TP4/CLNP. It is same as the loopback test on top of TP4/CLNP that the packets from the client process are looped back without going to the Ethernet cable. When the network is configured, the IP network address is bound to the address of the Ethernet interface. Since the client process and the server process are located in the same host, the packet from the sending process passes down to the IP network layer, reaches the Ethernet driver, and bounds the packet back to the network layer without transmitting it to the Ethernet cable. The time delay of disk-to-disk file transfer is tested in this environment and compared with the previous result with TP4/CLNP.

The disk-to-disk response time is tested as same as Section 4.1. The Figure 4.3 shows the result.

File Size (Byte)	TP4/CLNP Average (second)	TP4/CLNP Minimum (second)	TCP/IP Average (second)	TCP/IP Minimum (second)	Average Resp. time Compare (%)
2048	1.818	1.767	1.378	0.977	24%
8192	5.532	5.471	4.671	4.379	15%
16384	10.535	10.390	8.729	8.435	17%
32768	20.687	20.116	18.080	17.197	12%
65536	40.832	39.942	37.246	35.567	8%

Figure 4.3. Response Time of File Transfer (VAX 11/730, Disk-to-disk Case).

Derived results show that the TP4/CLNP protocol has the longer response time than the TCP/IP. The difference of the average response time between TCP/IP and TP4/CLNP is 0.44 second for 2048 bytes file transfer. The TCP/IP protocol provides about 24% better throughput than TP4/CLNP protocol in the 2048 bytes file transfer. The difference of the average response time between TCP/IP and TP4/CLNP is 3.586 seconds for 65536 bytes file transfer. The TCP/IP protocol provides about 8% better throughput than TP4/CLNP protocol in the 2048 bytes file transfer. The delay can occur with several reason in TP4/CLNP. The header part has more bytes in TP4/CLNP than in TCP/IP, especially the address part in the header of CLNP needs 20 bytes each for the source and the destination. The header of IP has 8 bytes each for the source and the destination address. It requires more processing overhead. Also, the tested TP4/CLNP protocol is the first version of the implementation, comparing that the TCP/IP is evolved during last decade. Reliable data transfer can be achieved with very little protocol processing if the network is perfect without bit errors, large and varying packet delays in the networks, packet loss due to congestion, out-of-order delivery of packets, and overflow of buffers at various nodes in the network [NET90]. Since ISO provides five classes of transport protocols, other class of transport protocol can be used to reduce processing overhead when the network becomes perfect. In reality, it is not possible in near future. Therefore, higher speed can be achieved by implementing some of protocol processing in hardware [KRI87].

CHAPTER 5

SUMMARY AND CONCLUSIONS

The specification and implementation of PACS protocol is intended to facilitate the development of PACS, which is capable of interfacing with a variety of distributed imaging devices. Also, it is intended to provide a test base for performance evaluation and to prototype the creation of diagnostic information databases.

This dissertation presents the specification of PACS protocol specification and the implementation effort of defined PACS protocol. The protocol specification is defined and implemented to reduce the protocol overhead with the following considerations: The description of service is small at the boundary, the number of interactions are minimized at the boundary, the similar functions are put into the same layer, the normal data and the image data are handled separately, the different vendor can insert localized functions easily, and the new technology or requirement can be incorporated without major protocol change. Also, the compatibility between different vendor networks can be achieved by using the protocol defined for presentation layer, session layer, transport layer, and network layer. The specified PACS protocol in this dissertation is a subset of the DICOM V3.0 standards which are developing in the ACR_NEMA working group VI.

The data link layer and the physical layer is open to different networks, which is currently existing and evolving. It provides required features for PACS and removes the obstacle of current ACR-NEMA standards. The defined four layer protocols follow the ISO-OSI standard framework and have inserted the feature of PACS requirement in each layer. Thus, it is a kind of sub-protocol for PACS under the OSI-ISO standard, which is a result of international teams of experts and efforts with a common agreement. The defined new

ACR-NEMA standards has sufficient flexibility to accommodate advances in technology and expansion in future user demands.

The defined protocol is implemented in the SUN workstations and VAX 11/730. The SUN workstations with SunOS 4.1.1 operating systems are used to implement the PACS protocol on top of the TCP/IP. The VAX 11/730 with BSD4.3-reno operating system is used to implement the PACS protocol on top of the TP4/CLNP. The underlying network is the Ethernet and the network software of PACS is coded with the C programming language. The PACS protocol is implemented with considerations of modularity, portability, transparency, and efficiency. The protocol overhead is minimized by that the data passing is reduced at the boundary, number of interactions are minimized at the boundary, and normal data and image data are handled separately. Since the porting of implemented protocol from the SUN workstation to the VAX 11/730 was relatively simple, the porting to the other machine could be accomplished without a major modification of implementation.

The results of a performance test demonstrates the performance of the implemented PACS system. The aim of PACS network is to support the image and data transmission in the hospital environment. The test result shows that the PACS network needs a high transmission rate to handle large amount of data. Also, the disk access speed in each node will seriously decrease the performance of PACS.

5.1 Constraints of Current Implementation

The current implementation of defined PACS protocol does have several constraints to provide a PACS service in a real hospital environment. Most of all, the user interface of the implementation is difficult for physicians who are not familiar with computers. The current implementation of application user interface requires not only typing user command

through keyboard but also understanding of network operation procedures.

The database archive system and the related application layer service for database query is not included in this protocol specification and implementation. Also the protocol implementation does not include voice service and extended abstract syntax notation compiler even though the protocol specification includes voice service and extended abstract syntax notation for PACS. The database archive system is the one of most important functional unit in the PACS. A conventional PACS database system was designed as a single centralized database system with a three layer hierarchy of functional components according to the tradeoffs in costs, capacities and performance levels of storage devices. It will become a communication bottleneck because of a large volume of data with centralized traffic. Thus the research on distributed database archive system is on going in The University of Arizona. Following this research, the protocol specification of the application layer service for the database archive system will be defined and implemented.

The underlying network of the protocol implementation is the Ethernet which only provides the maximum transmission speed of 10 Mbps. It is not suitable for the real PACS. The underlying network must support minimum 100 Mbps transmission speed to provides a required performance. The performance of PACS is also constrained by the buffer size and CPU speed. Large buffer size and high speed CPU are required for PACS implementation. Also the synchronous data transfer service in the underlying network is necessary for the implementation of the voice service.

The defined PACS protocol is implemented on top of the SunOS and 4.3BSD-Reno operating systems. It needs to be ported to the other operating systems including the System V operating system. The System V has a transport layer interface that provides a user interface to the implementation of various networking protocol including TCP/IP

protocol. Thus the current implementation using socket interface must be converted to the transport layer interface for the system V operating system. Also there is a single host for the implementation of the protocol stack with a TP4/CLNP layer. The TP4/CLNP layer is available in the 4.3BSD-Reno operating system. This needs to be expand to other hosts and other operating systems.

5.2 Future Work

The PACS must provide better service than current hospital image handling methods to the user, who is the physician or radiologist. Those users are not going to accept the PACS, unless it increases their performance and ability to review images. The response time of an image transfer is critical to reduce the time delay between image readings. It is necessary to satisfy physicians that the response time of an image transfer must be less than 2 seconds. The Ethernet does not provide such a bandwidth, since the transmission medium is a coaxial cable. Fiber optic networks can support such a transmission speed with more than 100 Mbps. Thus, future steps of implementations are porting developed software on the top of fiber optic networks.

The PACS should provide an user interface which is easy to learn and use. It is not as simple task as it seems, since physicians are not familiar with computers and the operation of the PACS requires many complicated functions. The user interface, which needs only simple key strokes or even further does not require a keyboard, is desirable for PACS users. This future work will hide the complex functions from users and make the PACS more attractive to users.

The performance has great a effect on the feature of the PACS. Not only is the quantity of data for image transfer expected to be very large, but the quality of data for image transfer

is extremely sensitive in the PACS environment. Especially, the Global PACS performance needs to be studied in great detail before the perspective interconnection of Local PACSs. The Global PACS performance not only depends on the Local PACS performance, but also on the integration method between heterogeneous Local PACS. Thus the performance studies on the transport layer protocol and network layer protocol are valuable future work for the Global PACS environment.

The implemented protocol need to be modified as a prototype with an exact implementation of the Version 3.0 DICOM standard. Following standard is important to provide communication of digital image information, regardless of source format or device manufacturer. Also, this will allow the creation of diagnostic information data base that can be interrogated by a wide variety of devices distributed geographically. Furthermore, the prototype implementation of the standard is very important for the success of DICOM standard. It will provide an ability to test and verify the defined standard. Experiment with prototype will not only promote further development of standard but also wide acceptance of standard in the medical society.

REFERENCES

[AME89] American College of Radiology, National Electrical Manufacturers Association, "ACR-NEMA Digital Imaging and Communications Standard: ACR-NEMA Standards Publication No. 300-1988," Washingon, D.C. ACR-NEMA, 1989.

[ARC87] Archwamety, C., "Design and Simulation of a totally Digital Image System for Medical Image Applications," Ph. D. Dissertation, University of Arizona, Dec. 1987.

[ARC88] Archwamety, C., "Image Migration in a Three Level Data Base Archive System," Proceeding of the SPIE, Medical Imaging II, Vol. 914, Newport Beach, CA, Feb. 1988.

[BLU88] Blume, H., Fuscoe, C., Hill, D., et al., "Extension of the ACR-NEMA digital interface communications standard to compression techniques: status report," Proceeding of the SPIE, Medical Imaging II, Vol. 914, pp. 786-791, 1988.

[BOG88] Boggs, D.R., Mogul, J.C., and Kent, C.A., "Measured Capacity of an Ethernet: Myths and Reality," ACM SIGCOM, pp. 222-234, 1988.

[BRU88] Bruce K.T.Ho, Kelby K.Chan, et al., "High Speed Image Compression System, Prototype and Final Configuration," Proceeding of the SPIE, Medical Imaging II, Vol. 914, pp. 786-791, 1988.

[CAB87] Cabrera, L.P., "Improving network subsystem performance in a distributed environment. A Berkeley Unix case study," Res. Rep. RJ5719, IBM Res. Division, Almaden Research Center, June 1987.

[CAB88] Cabrera, L., Hunter, E., Karels, M.J., and Mosher, D.A., "User-Process Communication Performance in Networks of Computers," IEEE transactions on software engineering, Vol. 14, No. 1, pp. 38-53, Jan. 1988.

[CLI83] Clifford J. Weinstein and James W. Forgie, "Experience with Speech Communication in Packet Networks," IEEE Journal on Selected Areas in Communications, Vol. SAC-1, NO. 6, pp 863-980, Dec. 1983.

[COH90] Cohn, M., Trefler, M., and Young, T.S., "Compression of digital chest x-rays," Proceeding of the SPIE, Medical Imaging IV: Image Capture and Display, Vol. 1232, pp. 396-400, 1990.

[DAL87] Dallas, W. J., et al., "A prototype Totally Digital Radiology Department: Conception and Initiation," Proceeding of the SPIE, PACS V, Vol. 767, Feb. 1987.

[DAL87A] Dallas, W.J., et al., "Radiology: Digital Design," Biomedical Imaging and Communications, July 1987.

[DAV85] Davidson, I., "Testing conformance to OSI satndards," Computer Communication, Vol. 8, No. 4, pp. 170-179, Aug. 1985.

[DES89] DeSoto, L.A., Choi, H.S., Haynor, D.R., et al., "Multiplanar Imaging System For Stereotaxic Neurosurgery," Proceeding of the SPIE, Medical Imaging III: Image Capture and Display, Vol. 1091, pp. 31-41, 1989.

[HOR88] Horii, S.C., Horii, H.N., and Kowalski, P., "An Electric Look at Viewing Station Design," Proceeding of the SPIE, Medical Imaging II, Vol. 914, Feb. 1988.

[I7498] ISO 7498, "Information processing systems - Open Systems Interconnection - Basic Reference Model," International Organization for Standardization, 1984.

[I8072] ISO 8072, "Information processing systems - Open Systems Interconnection - Transport Service Definition: 1st ed," International Organization for Standardization, June 1986.

[I8073] ISO 8073, "Information processing systems - Open Systems Interconnection - Connection Oriented Transport Protocol Specification," International Organization for Standardization, July 1986.

[I8326A] ISO 8326, "Information processing systems - Open Systems Interconnection - Basic connection oriented session service definition," International Organization for Standardization, 1984.

[I8326B] ISO 8326 DAD1, "Information processing systems - Open Systems Interconnection - Basic connection oriented session service definition/ Addendum 1: Session symmetric synchronization for the session service," International Organization for Standardization, 1987.

[I8327A] ISO 8327, "Information processing systems - Open Systems Interconnection - Basic connection oriented session protocol specification," International Organization for Standardization, 1984.

[I8327B] ISO 8327 DAD1, "Basic connection oriented session protocol specification/ Addendum 1 : Session symmetric synchronization for the session protocol," International Organization for Standardization, 1987.

[I8348A] ISO 8348, "Information processing systems - Open Systems Interconnection - Network service definition," International Organization for Standardization, 1984.

[I8348B] ISO 8348/Add 1, "Information processing systems - Open Systems Interconnection - Network service definition/ Addendum 1: Connectionless-mode transmission," International Organization for Standardization, 1987.

[I8348C] ISO 8348/Add 2, "Information processing systems - Network service definition/ Addendum 2: Network layer addressing," International Organization for Standardization, 1987.

[I8348D] ISO 8348/DAD 3, "Information processing systems - Data communications - Network service definition/ Addendum 3 : Additional features of the network service," International Organization for Standardization, 1987.

[I8473] ISO/DP 8473, "Information processing systems - Data Communications Protocol for Providing the Connectionless-Mode Network Service," International Organization for Standardization, 1984.

[I8822] ISO 8822, "Information processing systems - Open Systems Interconnection - Connection Oriented Presentation Service Definition," International Organization for Standardization, 1988.

[I8823] ISO 8823, "Information processing systems - Open Systems Interconnection - Connection Oriented Presentation Protocol Specification," International Organization for Standardization, 1988.

[KEL88] Kelby K. Chan, "Implementation of Fast Cosine Transforms with Digital Signal Processors for Image Compression," Proceeding of the SPIE, Medical Imaging II, Vol. 914, pp. 782-785, 1988.

[KRI87] Krishnakumar, A.S. et al., "Translation of formal protocol specifications into VLSI design," in Protocol Spec., Testing, and Verif., VII, Elsevier Publishers, North-Holland, pp. 375-390, May 1987.

[MAR87] Martinez, R., Archiwamety, C., and Nemat, M., "Simulation and Design of ACR-NEMA Standard For Fiber Optics Network Image Transfer," Proceeding of the SPIE, Medical Imaging, Vol. 767, Feb. 1987.

[MAR88] Martinez, R. and Nemat, M., "Image Data Base Archive System Design using Parallel Architectures and Expert Systems," Proceeding of the SPIE, Medical Imaging II, Vol. 914, Newport Beach, CA, Feb. 1988.

[MAR89] Martinez, R. and Nematbakhsh M., "Design and Performance Evaluation of a High Speed Fiber Optic Integrated Computer Network for Picture Archiving and Communications System," Proceeding of the SPIE, Medical Imaging, Newport Beach, CA, Feb. 1989.

[MAR90A] Martinez, R., Sanders, W., Alsafadi Y., and Nam J., "Performance Evaluation of a Picture Archiving and Cpmmunications Systems using Stochastic Activity Networks," Proceeding of the SPIE, Medical Imaging IV, Newport Beach, CA, Feb. 1990.

[MAR90B] Martinez, R., Dallas, W.J., and Komatsu, K., "Evaluation and Critique of the ACR-NEMA standard for Picture Archiving and Communications Systems," Proceeding of the SPIE, Medical Imaging IV, Newport Beach, Feb. 1990.

[MAR90C] Martinez, R., Nam, J., Ozeki, T., "White Paper; New Proposed Standard for Local and Global PACS Environment Using ISO Protocol," ACR-NEMA Standard Working Group VI,2101 L St., Suite 300, Washington, DC 20037, 70 pages, Nov. 1990.

[MAR91] Martinez, R., Nam, J., Dallas, W., et al., "Picture Archiving and Communications Systems Protocol Based on ISO-OSI Standard," Proceeding of the SPIE conf., Medical Imaging IV, Newport Beach, Feb. 1991.

[MAR92] Martinez, R., Nam, J., "Prototype Development and Implementation of Picutre Archiving and Communications Systems Based on ISO-OSI Standard," Accepted in the SPIE conf., Medical Imaging IV, Newport Beach, Feb. 1992.

[MCN90A] Mcneill, K., Martinez, R., and Maloney, K., "Design of Network Interface Unit for a Picture Archiving and Communications Systems," IEEE Ninth Annual International Phoenix Conference on Computers and Communications, Mar. 1990.

[MCN90B] McNeill, K., Osada, M., and Martinez, R., "Evaluation of the ACR-NEMA Standard for Communications in Digital Radiology," IEEE Transactions on Medical Imaging, Vol. NO. 9. NO. 3., September 1990.

[NAM91] Nam, J., Martinez, R., "Picutre Archiving and Communications Systems Development and Performance Results," Presented in the 91' Korean Automatic Control Conference, KOEX, Korea, Oct. 1991.

[NET90] Netravali, A.N., Roome, W.D., and Sabnani, K., "Design and Implementation of a High-Speed Transport Protocol," IEEE transactions on communications, Vol. 38, No. 11, pp. 2010-2024, Dec. 1990.

[NIS87] Nishihara, E., Tawara, K., Komatsu, K., Okikawa, T., and Kosos, K., "High Speed Image Transfer for PACS," Proceeding of the SPIE, PACS V, vol. 767, Feb. 1987.

[OZE87] Ozeki, T., Suzuki, M., Umino, H., Martinez, R., Archwamety, C., and Nemat, M., "Investigation and Analysis of the Requirements for a Picture Archiving and Communication System (PACS) at the University or Arizona Health Sciences Center," SPIE Medical Imaging 1987, Vol. 767, pp. 710-716, 1987.

[PRO91] Pronious, N.B., and Yovanof G.S., "Effect of transmission errors on medical images," Proceeding of the SPIE, Medical Imaging, Vol. 1446, Medical Imaging V: PACS Design and Evaluation, pp. 108-128, 1991.

[ROE87] Roehrig, H., et al., "Physical Evaluation of CRT Displays," Proceeding of the SPIE, PACS V, Vol. 767, Feb. 1987.

[ROB88] Roberts, B., Kakegawa, M., Nishikawa, M., and Oikawa D., "Toshiba TDF-500 High Resolution Viewing and Analysis System," Proceeding of the SPIE, Medical Imaging II, Vol. 914, Feb. 1988.

[SAI87] Saito, K., "Present and Future of Diagnostic Imaging Systems," Medical Review No. 20, pp. 47-57, 1987.

[SEE87] Seeley, G.W., et al., "An Overview of Picture Archiving and Communications Systems (PACS) Related Psychophysical Research in the University of Arizona Radiology Department," Proceeding of the SPIE, PACS V, Vol. 767, Feb. 1987.

[SHI89] Shin-Chung Lo, S. Horii, S.K. Moon, et al., "Development of Pictorial Workstation for Rapid Image Presentation," Proceeding of the SPIE, Medical Imaging III: Image Capture and Display, Vol. 1091, pp. 234-239, 1989.

[SHR88] Sherman, A.B., "Storage Model for Picture Archiving and Communication Systems," Proceeding of the SPIE, Medical Imaging II, Vol. 914, Newport Beach, CA, Feb. 1988.

[STV90] Stevens, W. Richard, "UNIX Network Programming," Prentice-Hall, Inc. Elglewood Cliffs, New Jersey, pp. 687-690, 1990.

[SVO89] Svobodova, L., "Implementing OSI Systems," IEEE transactions on selected areas in communications, Vol. 7, No. 7, pp. 1115-1130, Sep. 1989.

[TAT87] Tatsuya Suda and Tracy T. Bradley, "Packettized Voice/Data Integrated Transmission on a Token Passing Ring Local Area Network," IEEE Infocom 87 Proceeding, Vol. 2, pp 836-845, 1987.

[TOM89] Tomabaugh, J.W., Dillon, R.F., and Coristine, M., "Goal Setting and User Testing to Ensure a PACS Interface Satisfactory to Radiologists," Proceeding of the SPIE, Medical Imaging III, Vol. 1093, Feb. 1989.