NETWORK MANAGEMENT FOR PICTURE ARCHIVING AND COMMUNICATION SYSTEMS

by

Edlic Nga-Lik Yiu
B.A.Sc., Simon Fraser University, 2000

Edwood Nga-Wood Yiu
B.A.Sc., Simon Fraser University, 2000

PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF ENGINEERING

In the
School
of
Engineering Science

SIMON FRASER UNIVERSITY

Spring 2007

Approval

Name: Edlic Nga-Lik Yiu
Edwood Nga-Wood Yiu

Degree: Master of Engineering

Title of report: Network Management for Picture Archiving and Communication Systems

Examining Committee:

Chair: Dr. Daniel Lee
Associate Professor of Engineering Science

Dr. Ljiljana Trajković
Senior Supervisor
Professor of Engineering Science

Dr. Stephen Hardy
Supervisor
Professor of Engineering Science

Date Approved: _____

Abstract

Picture Archiving and Communication Systems (PACS) have been rapidly deployed to hospitals around the world over the past five years. Although these systems significantly improve the efficiency of the hospital workload, their connectivity with other diagnostic imaging devices is a challenging task. Thus, we have developed the PACS Monitor system based on the Simple Network Management Protocol (SNMP). PACS Monitor system can be used to efficiently identify and resolve connectivity issues.

This project provides a framework development of the PACS Monitor system. Although we have defined and developed only a subset of the PACS management data, the design may be scaled to support the entire management data used for PACS administration.

Acknowledgments

In preparing this project, we have been fortunate to receive valuable support from numerous professors and friends. These include Dr. Ljiljana Trajković, Dr. Stephen Hardy, Fiona Chan, and Ada Pang. We would also like to express our appreciation to McKesson Medical Imaging Group for providing the medical imaging software.

Table of Contents

List of Figures

List of Tables

Glossary

AE	Application Entity
CCITT	Consultative Committee for International Telegraph and Telephone
CMOT	Common Management Information Protocol over TCP/IP
CR	Computed Radiography
CT	Computed Tomography
DES	Data Encryption Standard
DICOM	Digital Imaging and Communications in Medicine
DIMSE	DICOM Message Service Element
DLL	Dynamic Link Library
HEMS	High-Level Entity Management System
HIS	Hospital Information System
HL7	Health Level Seven
ICMP	Internet Control Message Protocol
IAB	Internet Activities Board
IHE	Integrating the Healthcare Enterprise
IOD	Information Object Definition
ISO	International Organization for Standardization
IT	Information Technology
MAC	Message Authentication Code
MIB	Management Information Base
MPPS	Modality Performed Procedure Step
MRI	Magnetic Resonance Imaging
NEMA	National Electrical Manufacturers Association
NM	Nuclear Medicine
OID	Object Identifier
PACS	Picture Archiving and Communications System
PDU	Protocol Data Units
PDV	Presentation Data Values
PET	Positron Emission Tomography
PING	Packet Internet Groper Program
RFC	Request for Comment
RIS	Radiology Information System
SCP	Service Class Provider
SCU	Service Class User
SGMP	Simple Gateway Monitoring Protocol
SNMP	Simple Network Management Protocol
SOP	Service-Object Pair
TCP/IP	Transmission Control Protocol/Internet Protocol
UID	Unique Identifier

UL	Upper Layers
US	Ultrasound
VACM	View-Based Access Control Model
VR	Value Representation

1 Introduction

With advancements of telecommunication technologies, data networking has become an integral part of our lives. Organizations rely on networks to share information, while general public depends on them to communicate. Today, most companies are connected to the Internet via an Internet Service Provider. Large corporations may even have their own network for exchanging information. With the technological advancement, networks may be interconnected via routers and bridges from various types of local area networks, such as Ethernet and Token Ring. Over the last decade, efforts have been made to standardize networking protocols so that network equipment may be deployed in a multi-vendor environment.

Growing demands for network communication are no longer restricted to the business world. Since the mid 1990s, hospitals have begun to interconnect diagnostic imaging devices. With the advancement of network technologies, these devices are connected to a central archiving system (PACS). PACS provides storage and management services for diagnostic images and reports. This system helps improve hospitals' workflow by reducing the diagnostic time from hours to minutes. Furthermore, the central archiving capability significantly reduces the administration cost.

As the scientific research continues, many types of diagnostic imaging devices are being developed and deployed in hospitals to address various needs. Due to the increased number of diagnostic imaging devices, connectivity to PACS can no longer be manually monitored and maintained. In response to this need, we have developed the PACS Monitor system to provide statistic collection, configuration management, and fault management for the PACS administrator. With the PACS Monitor system, connectivity problems can be efficiently identified and resolved. In our design of the PACS Monitor system, we employed Simple Network Management Protocol (SNMP) for monitoring medical systems. In this project, our focus on the PACS management data is limited only to the storage service, even though PACS provides numerous service operations for diagnostic imaging devices, such as storage, query, and retrieve services. However, this design can be scaled to support management data for other services.

An overview of the hospital environment is provided in Section 2. Section 3 discusses the communication protocol used between PACS and diagnostic imaging devices. An overview of the SNMP protocol is presented in Section 4. Motivation for applying SNMP to medical systems is given in Section 5. Section 6 focuses on the design and implementation of the proposed PACS Monitor system. Finally, we review the result of our development in Section 7.

2 Hospital

2.1 History of Hospitals

Hospitals were established by the Romans prior to the 3rd century to treat sick and injured soldiers. In those days, types of treatments were limited and isolations of patients with uncured diseases from the community were frequently seen. Towards the Middle Ages, the number of hospitals built by religious groups grew. Few of those hospitals were large enough to house more than 2,000 patients. During that period, medicinal plants were the primary sources of treatments since the advancement of medicine was still very slow. Towards the beginning of the 18th century, secular authorities started to provide healthcare service, with many voluntary hospitals established to serve the sick. Countless scientific and medical breakthroughs initiated in the mid-19th century allow physicians to better understand both the human body and diseases. For example, Wilhelm Conrad Röntgen discovered x-rays in 1895, which can be used to develop film for displaying interiors of the human body. Other scientific advances in the 1900s also led to the development of numerous non-invasive diagnosis devices, such as Magnetic Resonance Imaging (MRI), Computed Tomography (CT), Computed Radiography (CR), Ultrasound (US), Positron Emission Tomography (PET), and Nuclear Medicine (NM). These significant scientific advancements allowed physicians to make accurate diagnosis and to perform clinical researches. Physicians learned specialized knowledge and developed new medical techniques to treat patients. Specialization in various areas of medicine ultimately led to various departments in hospitals. Nowadays, a general hospital consists of numerous departments, such as cardiology, neurology, oncology, and radiology. They work together to provide medical care to patients. As hospital structure became more complex, new IT technologies were introduced into medical institutions to help administration and to provide better healthcare services to the public [20].

2.2 Hospital IT Systems

Medical-trained staffs in modern hospitals use diagnosis devices and IT systems on a daily basis. These systems reside in various parts of hospital buildings. Nevertheless, they work together to provide numerous functionalities, such as billing, imaging, medication management, and diagnosis reporting. Figure 1 illustrates the interaction among these systems.

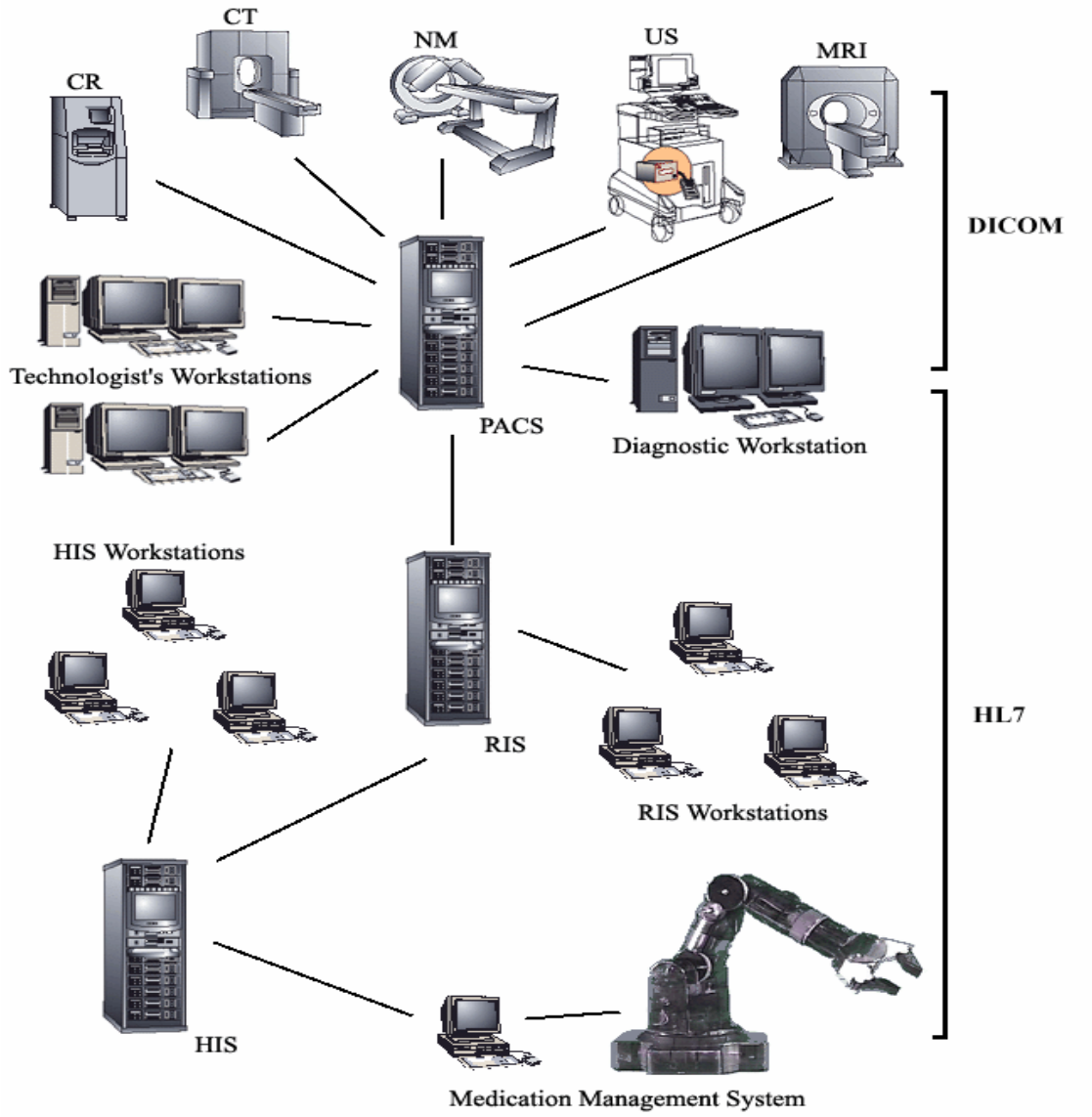

Figure 1. Interactions among hospital IT systems including modalities, picture archiving and communications system (PACS), radiology information system (RIS), hospital information system (HIS), and automation systems.

2.2.1 Hospital Information System (HIS)

HIS is used for administrating hospital and managing clinical processes. It is also called ADT because the Admission, Discharge, and Transfer of patients' information are recorded via the HIS interface. When a patient is admitted to a hospital, demographic information concerning the patient is entered into HIS. This information is sent automatically and electronically to RIS, PACS, and modalities. As a result, typing errors of patient demographic information at numerous IT systems and diagnostic devices are minimized. Another benefit of HIS is its billing capability to automatically record charges to various services including bed, lab tests conducted, medicine used, consultation fee, food, and beverages [19]. In some hospitals, HIS may also connect to numerous automation systems to improve productivity and patient safety.

2.2.2 Automation Systems

Automation technologies employed in hospitals are typically used for medication management processes and patient administration. These technologies are usually based on bar code scanning to reduce medication errors and increase patient safety. For example, some hospitals automate the entire medication management processes that include prescribing, packaging, and dispensing medication without human intervention. Caregivers may then utilize a wireless device with a bar code scanner to verify the correct dosage of medication for a patient [17]. Hospitals can guarantee to provide high quality patient care with these automation systems.

2.2.3 Radiology Information System (RIS)

RIS is used in the radiology department for tracking and managing patients, films, and supplies. When a physician requests the technologist to perform a scan on a patient, the scheduled procedures will be ordered through RIS. The procedures and the patient demographic information are immediately sent to PACS in electronic form. This information is also sent to the modality upon the request from the scanner. Through RIS, patient demographic information can be reconciled and updates are automatically

propagated to PACS and modalities. RIS enables the administration and clinical workflow in the radiology department to be performed in a more efficient manner.

2.2.4 Picture Archiving and Communications System (PACS)

PACS is used to help hospitals to capture, manage, store, and view diagnostic images. The diagnostic images are generated from modalities and are sent to PACS in electronic form after the scan. Upon receiving the diagnostic images via PACS, technologists may review and validate the images with the performed procedures. After the review and validation, radiologists may start reading the images and performing the diagnosis. This scanning-reviewing-reporting process used to take more than an hour with traditional films. PACS shortens this process to less than five minutes, thereby enabling patients to promptly receive proper treatments. Another benefit of using PACS is time saving, since radiologists no longer need to sort and handle films. Instead, they can now fully devote their times to read images and report the diagnosis using PACS [2].

2.2.5 Modalities

Modality is another name for diagnostic imaging device, which allows physicians to make accurate diagnosis by revealing interiors of the human body without anatomical procedures. Various types of modalities can easily be found in modern hospitals, such as MRI, CT, CR, US, PET, and NM. The diagnostic images generated from these modalities have distinctive characteristics. For example, CR focuses the x-rays on the patient plane while CT aims the x-rays on the cross-section plane of the patient. Furthermore, MRI images provide the best contrast between normal and damaged tissues [22] while NM images are ideal for showing the presence and size of abnormalities in the body organ [23]. With the availability of these medical imaging technologies, a physician can request different kinds of scans to obtain anatomic views of the patient in order to make the most accurate diagnosis. Fifteen years ago, radiologists would only receive x-ray films a few hours after the scan was performed. With the introduction of PACS, radiologists and physicians can immediately access these images at a workstation.

2.3 Communication Protocols and Integration Framework

Within a hospital, IT systems and diagnostic devices convey information with each other through multiple communication protocols. In the radiology department, Digital Imaging and Communications in Medicine (DICOM) and Health Level Seven (HL7) are used for communication among RIS, PACS, and modalities. Nevertheless, these protocols are not sufficient for successful integration among hospital systems built from different vendors. Consequently, Integrating the Healthcare Enterprise (IHE) started in 1998 to address the integration issue commonly seen in hospital workflow.

2.3.1 Digital Imaging and Communications in Medicine (DICOM)

DICOM is a network protocol used in hospitals. The emergence of this protocol marks the beginning of digital evolution of medical imaging industry. The DICOM standard facilitates interoperability among medical imaging devices from different vendors. This standard defines the network communication layer for message exchange, the syntax and semantics of commands and associated information, and the file storage format. This protocol is widely used by diagnostic devices and PACS for exchanging images and associated patient information.

2.3.2 Health Level Seven (HL7)

HL7 is another important communication protocol used in hospitals. This standard provides the groundwork for encoding and exchanging health care information of a patient among different hospital systems using the TCP/IP network. With this standard, patient record, order entry, and financial information can be passed to HIS, RIS, and PACS without human intervention. However, HL7 do not provide any support of imaging data.

2.3.3 Integrating the Healthcare Enterprise (IHE)

IHE is an initiative started by healthcare professionals and medical vendors to improve integrations among diagnostic devices and hospital IT systems. The DICOM and HL7 standards are already used by hospital equipment for communications. Nevertheless, the

interpretations of these standards vary somewhat from vendor to vendor. Consequently, integration of hospital equipments made by multiple vendors may pose some challenges. IHE resolves these conflicting interpretations by first identifying common integration problems in administration, clinical workflow, information access, and underlying infrastructure. IHE then selects standards to address integration needs and the implementation details of using these standards are subsequently documented in the IHE Technical Framework. When medical vendors develop their products according to this framework, integration in hospitals with these medical systems will be simpler [15].

2.4 Hospital Workflow

Hospital workflow is defined as the process and information flow of patient care within a hospital. A typical hospital workflow is presented in Figure 2, where the communication protocol used for each transaction is also identified. The demographic information of a patient is entered into HIS by the admission department as soon as the patient arrives, and the information is subsequently forwarded to RIS. Should the patient require a scan, procedures are scheduled through RIS and forwarded to PACS. Prior to the scan, the modality performs a query with RIS to obtain the worklist, which contains patient demographic and scheduled procedure information. After the scan, the modality sends the diagnostic images to PACS for storage. The modality also sends a scan-completed message, known as Modality Performed Procedure Step (MPPS), to PACS. In order for RIS to realize the status of the scheduled procedures, PACS redirects the MPPS message to RIS. The modality then sends the storage commitment message to PACS to verify that the entire set of images was properly saved in PACS. RIS also queries PACS to confirm the existence of these images. After the confirmation, RIS sends the order update message to HIS so that the imaging charge is posted to the patient's account.

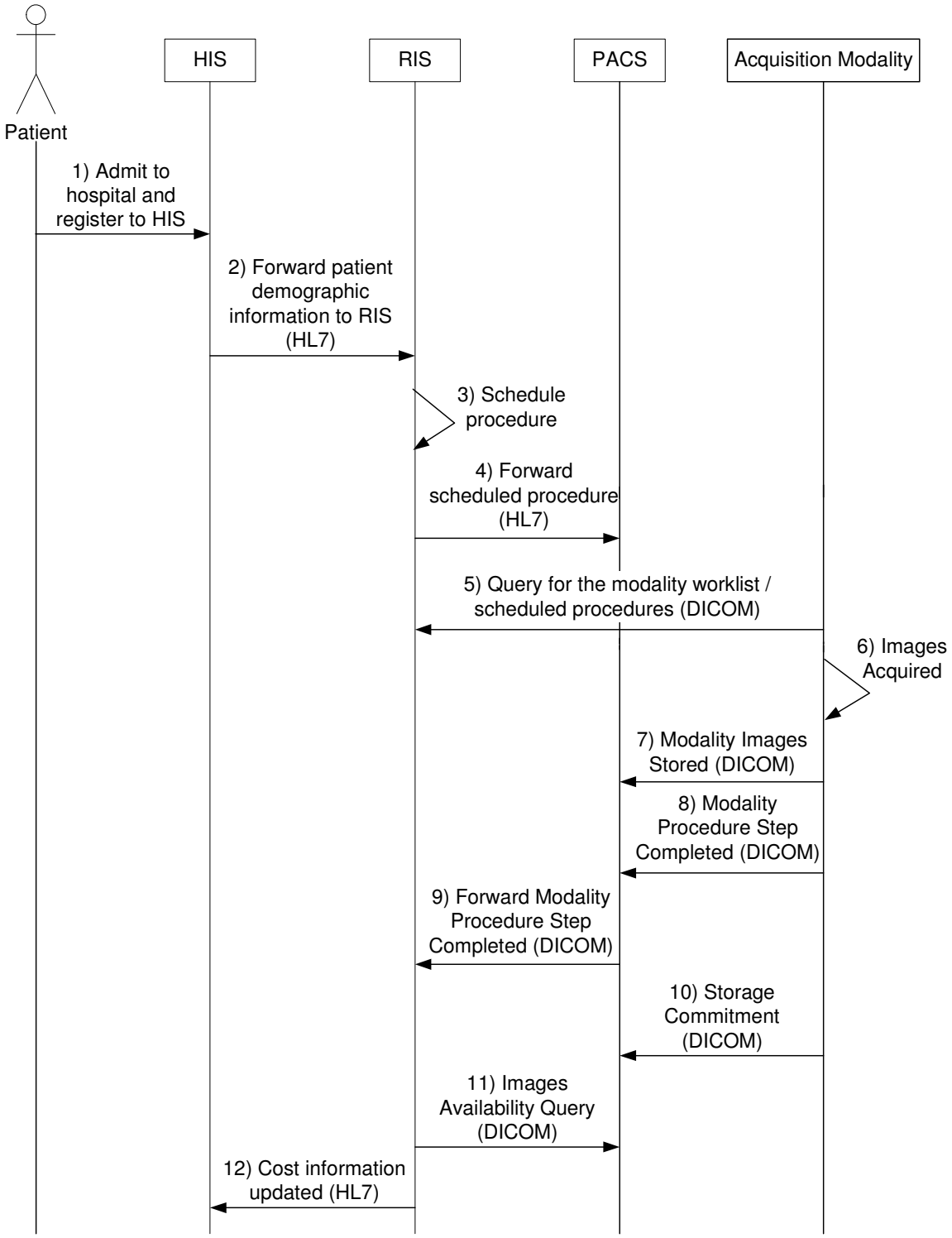

Figure 2. Illustration of a typical hospital workflow: process and information flow of patient care within a hospital.

3 Digital Imaging and Communications in Medicine

DICOM is the foundation of PACS because every transaction between a modality and PACS uses DICOM. Without DICOM, modalities do not have a common interface to communicate with PACS, and integration of PACS into a hospital's network becomes impossible. Since this project involves development to PACS, DICOM is examined in detail here.

DICOM is built on top of the TCP/IP protocol stack to facilitate communications, as shown in Figure 3. Above the TCP/IP layer, DICOM Upper Layer Protocol and DICOM Application Message Exchange can be found. Furthermore, the DICOM Upper Layer Service is defined between these two layers to act as a bridge.

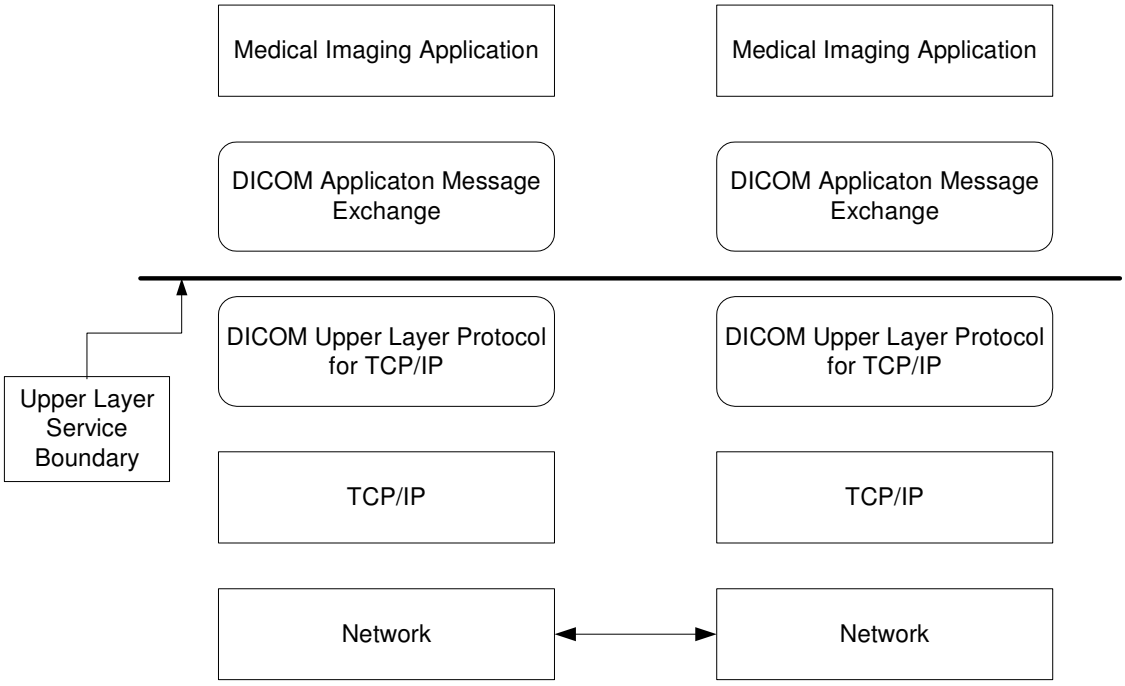

Figure 3. DICOM network protocol stack on top of the TCP/IP network [12].

3.1 DICOM Upper Layer Service

The DICOM Upper Layer (UL) Service provides generic network interfaces that can be employed by the application entities (AEs) for exchanging DICOM messages. AEs are medical imaging applications that are capable of performing DICOM communications.

These UL Services include A-ASSOCIATE, A-RELEASE, A-ABORT, A-P-ABORT, and P-DATA.

3.1.1 A-ASSOCIATE

The A-ASSOCIATE service is used to establish an association between two AEs. This service is a confirmed type, where the acceptor will reply with the A-ASSOCIATE response after receiving the A-ASSOCIATE indication from the initiator of the service, as shown in Figure 4.

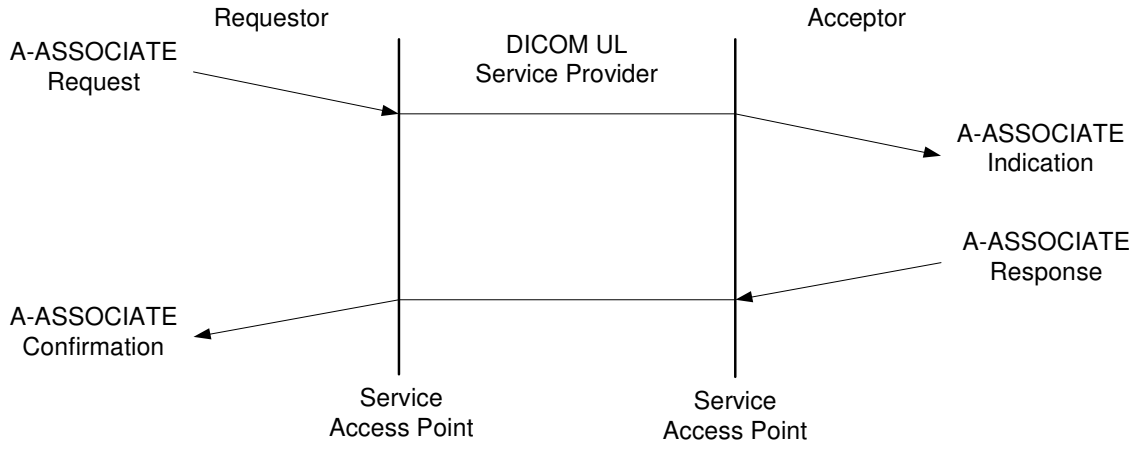

Figure 4. Association establishment of two AEs via A-ASSOCIATE service [12].

Several parameters are negotiated between AEs at association establishment. The key parameters include calling/called AE titles, calling/called presentation addresses, and presentation context list. Calling/called AE titles are employed to identify application entities of the requestor and acceptor while calling/called presentation addresses are used to specify the IP addresses of the requestor and acceptor. Presentation context list item consists of the identification, abstract syntax name, and a list of transfer syntaxes. Abstract syntax governs what Service-Object Pair (SOP) instances may be exchanged between AEs while the transfer syntax defines what compression formats may be applied on the instances. The definition of SOP contains the rules, semantics, and services for the instance. Two types of SOP class are defined in DICOM: Composite SOP Class and Normalized SOP Class. Composite SOP Class allows multiple information models embedded within the instance. Normalized SOP Class only allows one information

model. For example, a diagnostic image file (e.g., CT image) belongs to the Composite SOP Class, which usually contains patient information, study information, and the pixel data. Print queue belongs to the Normalized SOP Class, which only contains print job related information.

3.1.2 A-RELEASE

The A-RELEASE service is used by the AEs to gracefully release the DICOM association. This service is a confirmed type, as illustrated in Figure 5. Either one of the AEs may initiate this service. Parameters for this service include reason and result, with respective fixed values of “normal” and “affirmative”.

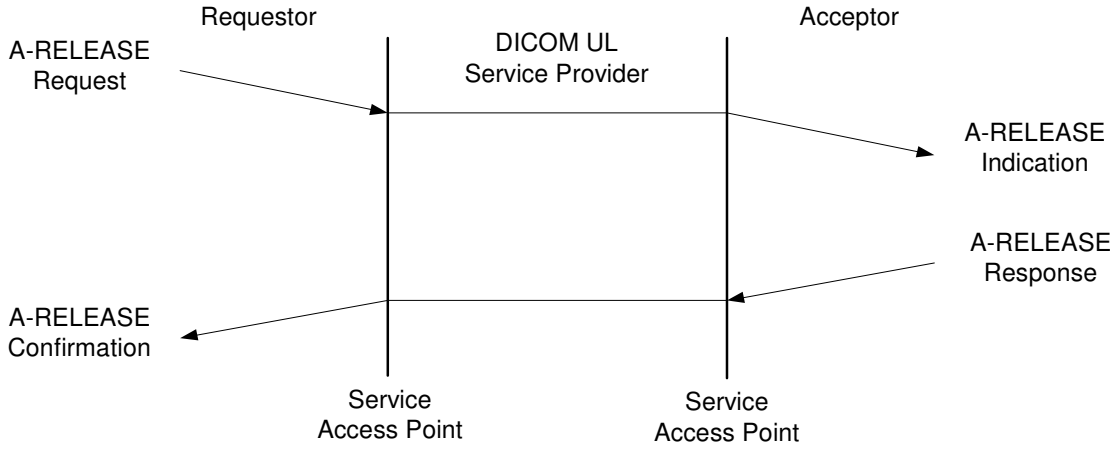

Figure 5. Association release of two AEs via A-RELEASE service [12].

3.1.3 A-ABORT and A-P-ABORT

The DICOM UL Service contains two interfaces, A-ABORT and A-P-ABORT, for aborting the association. The A-ABORT service should be used by either one of the AEs to signal abnormal termination of an association. The A-P-ABORT service should be employed by the DICOM UL service provider to terminate associations due to errors detected below the DICOM Upper Layer Service Boundary. Both services are non-confirmed type, as shown in Figure 6 and Figure 7.

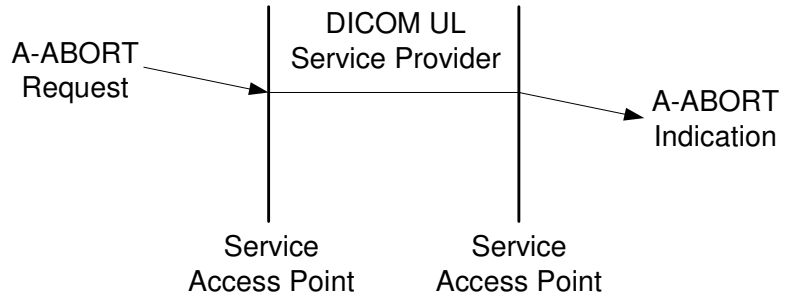

Figure 6. User association abortion of two AEs via A-ABORT service [12].

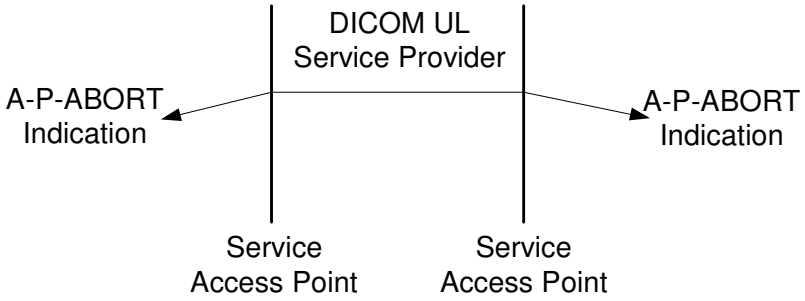

Figure 7. UL service provider association abortion via A-P-ABORT service [12].

3.1.4 P-DATA

The P-DATA service is used to exchange DICOM messages between two AEs. This service is a non-confirmed type, as illustrated in Figure 8.

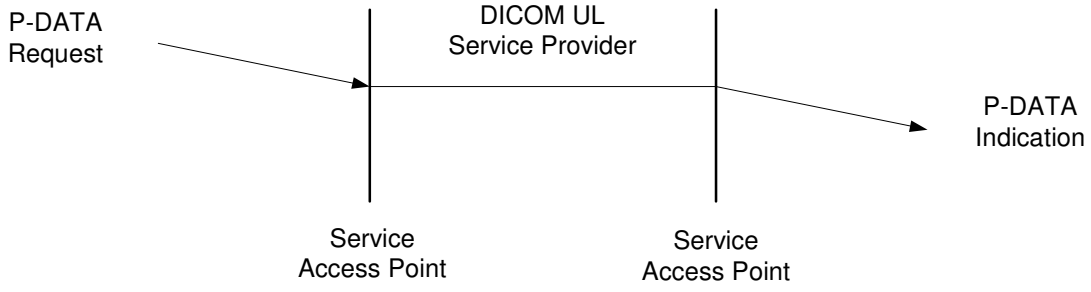

Figure 8. Data transfer between two AEs via P-DATA service [12].

Both AEs may employ this service to initiate bi-directional communications. The presentation data value list, which contains multiple presentation data values (PDV), is a key parameter in this service. Each PDV contains the presentation context ID and user

data value fields. The presentation context ID specifies the abstract syntax and transfer syntax applied on the DICOM message. This message is sent to the destination AE via the user data value field. The DICOM message is discussed in Section 3.2.

3.1.5 DICOM Upper Layer Protocol

Every successful DICOM connection consists of three stages: association establishment, data transfer, and association release. The A-ASSOCIATE service is used during the association establishment stage. The data transfer stage employs the P-DATA service. The association release stage utilizes one of the A-RELEASE, A-ABORT, and A-P-ABORT services. All of these DICOM UL services are supported by the DICOM upper layer protocol, which consists of seven Protocol Data Units (PDUs). These PDUs include A-ASSOCIATE-RQ PDU, A-ASSOCIATE-AC PDU, A-ASSOCIATE-RJ PDU, P-DATA-TF PDU, A-RELEASE-RQ PDU, A-RELEASE-RP PDU, and A-ABORT PDU. These PDUs are encoded using Big Endian byte ordering, and the structures of these PDUs are shown in Figure 9, Figure 10 and Figure 11. Note that all field sizes are measured in bytes.

PDU Type is set to 01H for the A-ASSOCIATE-RQ PDU, 02H for the A-ASSOCIATE-AC PDU, and 03H for the A-ASSOCIATE-RJ PDU. These three PDUs are used by the A-ASSOCIATE service to facilitate association establishment. PDU Type is set to 05H for A-RELEASE-RQ PDU and 06H for A-RELEASE-RP PDU. The A-RELEASE service employs these two PDUs to gracefully release the association. The A-ABORT and A-P-ABORT services utilize the A-ABORT PDU for aborting the association, and PDU Type for the A-ABORT PDU is set to 07H. P-DATA PDU is used for data transfer, and PDU Type for this PDU is set to 04H.

DICOM messages are encapsulated and exchanged between AEs using the P-DATA PDU. When a DICOM message is larger than the negotiated PDU size, the message is split into multiple Command or Data fragment. Each fragment and the message control header are placed into a PDV. A series of PDVs embedded into the PDUs are sent sequentially to preserve the fragment orders in the message. The destination AE recognizes the end of the PDV stream by examining the second least significant bit of the

message control header. The bit is set to 1 to indicate the last fragment of the DICOM message, and the bit is set to 0 to indicate other fragments. When the destination AE receives the last PDV, all fragments are assembled back to form the DICOM message.

The detailed description of each field as shown in Figure 9, Figure 10, and Figure 11 is given in "DICOM Part 8: Network Communication Support for Message Exchange" standard [12].

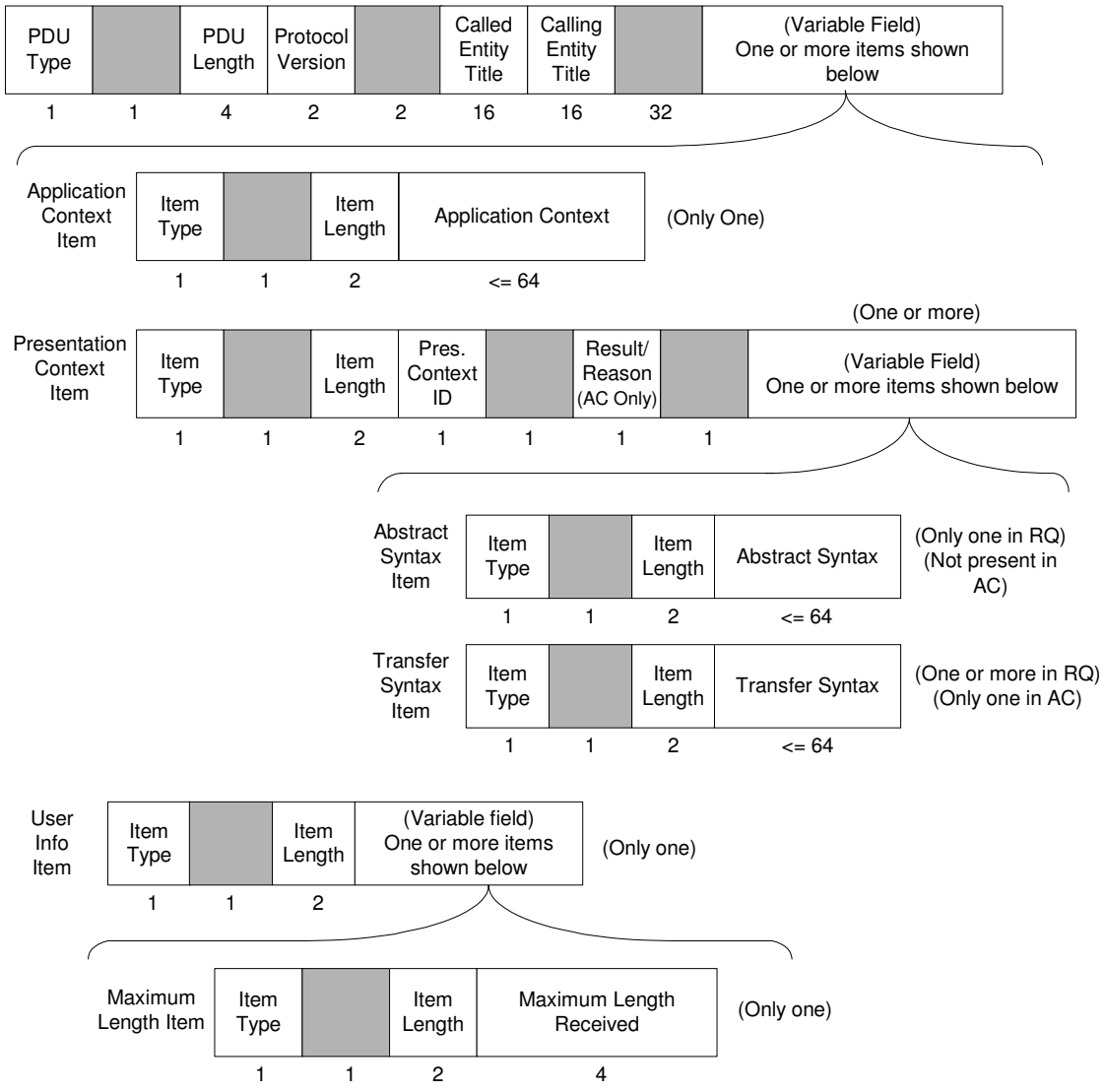

Figure 9. A-ASSOCIATE-RQ/A-ASSOCIATE-AC PDU encoding structure used during association establishment [12].

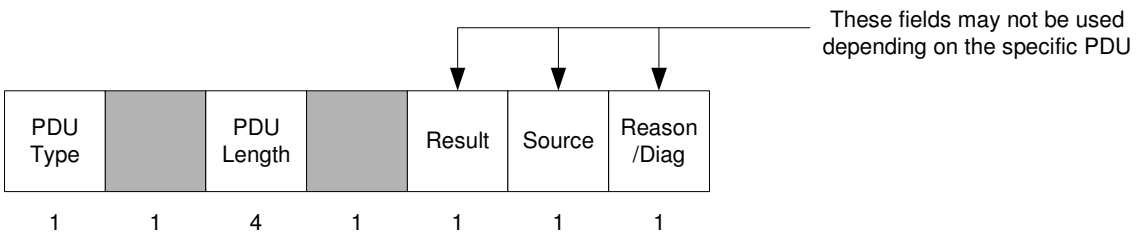

Figure 10. A-ASSOCIATE-RJ/A-RELEASE-RQ/A-RELEASE-RP/A-ABORT PDU encoding structure used during association release [12].

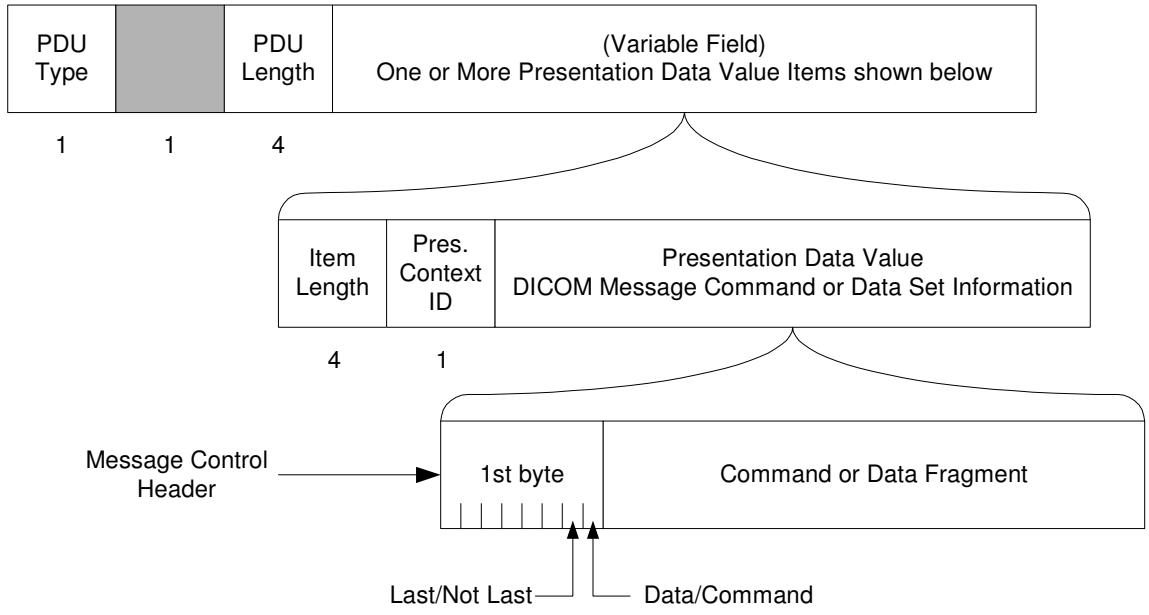

Figure 11. P-DATA-TF PDU encoding structure used during data transfer [12].

3.2 DICOM Application Message Exchange

3.2.1 DICOM Message Service Element

DICOM AEs employ the services provided by the DICOM Message Service Element (DIMSE) for communication. The part of DICOM AE that uses DIMSE is called DIMSE-service-user. The DICOM AE can generate different DICOM application messages using various DIMSE services to convey application-specific information to the peer DIMSE-service-user. The DICOM messages are sent to the destination AE via the P-DATA PDU, as discussed in Section 3.1.5. The DIMSE services include C-STORE, C-

FIND, C-MOVE, C-GET, C-ECHO, N-EVENT-REPORT, N-GET, N-SET, N-ACTION, N-CREATE, and N-DELETE. The descriptions of these DIMSE services are summarized in Table 1.

Table 1. Descriptions of the DIMSE services including C-ECHO, C-STORE, C-FIND, C-MOVE, C-GET, N-EVENT-REPORT, N-GET, N-SET, N-ACTION, N-CREATE, and N-DELETE.

DIMSE Service Name	Description
C-STORE	This service allows a DIMSE-service-user to request the storage of the Composite SOP instances to the peer DIMSE-service-user.
C-FIND	Using this service, a DIMSE-service-user can request to match a series of attributes against the attributes in the set of the DICOM SOP instances managed by the peer DIMSE-service-user. The response of this service returns a list of matches containing the requested attributes and the respective values.
C-MOVE	This service allows a DIMSE-service-user to request moving the Composite SOP instances stored in the peer DIMSE-service-user to a third-party DIMSE-service-user.
C-GET	DIMSE-service-user can employ this service to retrieve the Composite SOP instances from the peer-DIMSE-service-user.
C-ECHO	Using this service, DIMSE-service-user can verify the end-to-end communication with the peer DIMSE-service-user.
N-EVENT-REPORT	This service allows the DIMSE-service-user to report an event relating to the SOP instance to the peer DIMSE-service-user.
N-GET	Using this service, DIMSE-service-user can retrieve information from the peer DIMSE-service-user.
N-SET	DIMSE-service-user can use this service to request the modification of information performed by the peer DIMSE-service-user.
N-ACTION	With this service, DIMSE-service-user can request the peer DIMSE-service-user to perform an action.
N-CREATE	DIMSE-service-user can request the peer DIMSE-service-user to create the SOP instance using this service.
N-DELETE	DIMSE-service-user can request the peer DIMSE-service-user to delete the SOP instance using this service.

All of these DIMSE operations are confirmed services. When the invoking DIMSE-service-user sends a DIMSE service request to the performing DIMSE-service-user, a DIMSE service response should be returned, as shown in Figure 12.

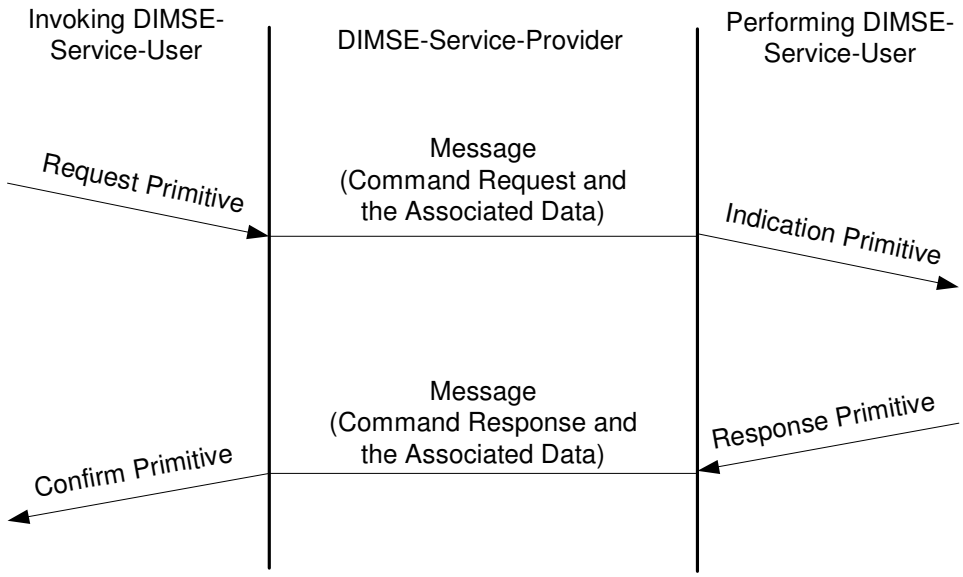

Figure 12. Request and response primitives found in all DIMSE operations [11].

3.2.2 DICOM Command Set Encoding

Each DICOM message must contain the Command Set followed by the conditional Data Set. Within the Command Set, various Command Elements may be found for different DIMSE services. In each Command Element, three fields can be identified: an explicit Tag, a Value Length, and a Value Field. Figure 13 illustrates the encoding of the DICOM Command Set. Detailed discussion regarding the encoding of the DICOM Data Set is provided in the "DICOM Part 5: Data Structures and Encoding" standard [10].

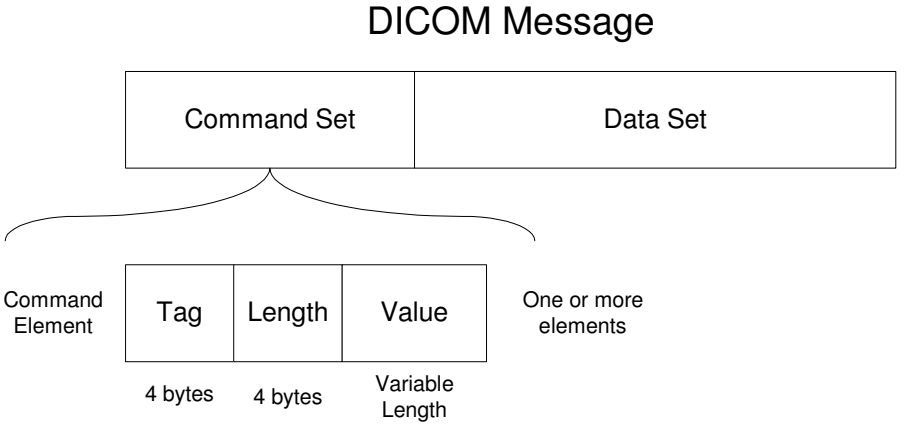

Figure 13. DICOM message structure used in the DIMSE services [11].

3.2.3 Message Fields in the DIMSE Services

Value Representation (VR) specifies the data type for the Value field in the Command Element. Some common VRs include Application Entity (AE), Unique Identifier (UI), Unsigned Long (UL), and Unsigned Short (US). The maximum byte sizes for AE is 16, and while the maximum byte size for UI is 64. The UL value length must be 4 bytes, and the US value length must be 2 bytes. The most frequently used DIMSE services are C-STORE, C-FIND, and C-MOVE. The key message fields of these DIMSE services are discussed in this Section. Detailed discussions about other message fields in the DIMSE services are given in the "DICOM Part 7: Message Exchange" standard [11].

The message fields in the C-STORE service request include Group Length, Affected SOP Class UID, Command Field, Message ID, Priority, Data Set Type, Affected SOP Instance UID, Move Originator Application Entity Title, Move Originator Message ID, and Data Set. The key message fields are summarized in Table 2.

Table 2. Key message fields in the C-STORE service request [11].

Message Field	Tag	VR	Description of Field
Command Field	(0000,0100)	US	This field should be set to 0001H for the C-STORE-RQ message.
Message ID	(0000,0110)	US	This field contains implementation-specific value. It distinguishes this message from other messages.
Priority	(0000,0700)	US	The priority shall be set to one of the following values: LOW = 0002H MEDIUM = 0000H HIGH = 0001H
Data Set Type	(0000,0800)	US	This field indicates that the Data Set field exists in the message, and the value should be set to anything other than 0101H.
Move Originator Application Entity Title	(0000,1030)	AE	Contain the DICOM AE Title of the C-MOVE requestor
Move Originator Message ID	(0000,1031)	US	Contains the Message ID (0000,0110) of the C-MOVE-RQ Message fro which this C-STORE sub-operations is being performed.
Data Set	(No tag)	-	Application-specific Data Set

The message fields in the C-STORE service response include Group Length, Affected SOP Class UID, Affected SOP Instance UID, Command Field, Message ID Being Responded To, Data Set Type, and Status. The key message fields are summarized in Table 3.

Table 3. Key message fields in the C-STORE service response [11].

Message Field	Tag	VR	Description of Field
Command Field	(0000,0100)	US	This field should be set to 8001H for the C-STORE-RSP message.
Message ID Being Responded To	(0000,0120)	US	Shall be set to the value of the Message ID (0000,0110) field used in associated C-STORE-RQ Message.
Data Set Type	(0000,0800)	US	This field indicates that the Data Set fields is not present in the message, and the value should be set to 0101H.
Status	(0000,0900)	US	This field provides the status of the C-STORE service. For example, A7xx implies that the performing DIMSE-service-user refuses the operation due to out-of-resources.

The message fields in the C-FIND service request include Group Length, Affected SOP Class UID, Command Field, Message ID, Priority, Data Set Type, and Identifier. The key message fields are shown in Table 4.

Table 4. Key message fields in the C-FIND service request [11].

Message Field	Tag	VR	Description of Field
Command Field	(0000,0100)	US	This field should be set to 0020H for the C-FIND-RQ message.
Message ID	(0000,0110)	US	This field contains implementation-specific value. It distinguishes this message from other messages.
Priority	(0000,0700)	US	The priority shall be set to one of the following values: LOW = 0002H MEDIUM = 0000H HIGH = 0001H
Data Set Type	(0000,0800)	US	This field indicates that the Data Set field exists in the message, and the value should be set to anything other than 0101H.
Identifier	(No tag)	-	A Data Set that encodes the identifier for matching.

The message fields in the C-FIND service response include Group Length, Affected SOP Class UID, Command Field, Message ID Being Responded To, Data Set Type, Status, and Identifier. The key message fields are shown in Table 5.

Table 5. Key message fields in the C-FIND service response [11].

Message Field	Tag	VR	Description of Field
Command Field	(0000,0100)	US	This field should be set to 8020H for the C-FIND-RSP message.
Message ID Being Responded To	(0000,0120)	US	Shall be set to the value of the Message ID (0000,0110) field used in associated C-FIND-RQ Message.
Data Set Type	(0000,0800)	US	This field indicates that the existence of the Data Set fields in the message. If the Data Set field is present, the value should be set anything other than 0101H. Otherwise, the value should be set to 0101H.
Status	(0000,0900)	US	This field provides the status of the C-FIND service. For example, FF00 implies that the performing DIMSE-service-user is still continuing on matching. Hence, additional C-FIND-RSP messages will arrive.
Identifier	(No tag)	-	A Data Set that encodes the identifier for each matching item.

The message fields in the C-MOVE service request include Group Length, Affected SOP Class UID, Command Field, Message ID, Priority, Data Set Type, Move Destination, and Data Set. The key message fields are summarized in Table 6.

Table 6. Key message fields in the C-MOVE service request [11].

Message Field	Tag	VR	Description of Field
Command Field	(0000,0100)	US	This field should be set to 0021H for the C-MOVE-RQ message.
Message ID	(0000,0110)	US	This field contains implementation-specific value. It distinguishes this message from other messages.
Priority	(0000,0700)	US	The priority shall be set to one of the following values: LOW = 0002H MEDIUM = 0000H HIGH = 0001H

Message Field	Tag	VR	Description of Field
Data Set Type	(0000,0800)	US	This field indicates that the Data Set field exists in the message, and the value should be set to anything other than 0101H.
Move Destination	(0000,0600)	AE	This field should be set to the DICOM AE Title of the storage destination for which the C-STORE sub-operations are being performed.
Identifier	(No tag)	-	A Data Set that encodes the identifier for matching.

The message fields in the C-MOVE service response include Group Length, Affected SOP Class UID, Command Field, Message ID Being Responded To, Data Set Type, Status, Number of Remaining Sub-operations, Number of Complete Sub-operations, Number of Failed Sub-operations, Number of Warning Sub-operations, and Identifier. The key message fields are summarized in Table 7.

Table 7. Key message fields in the C-MOVE service response [11].

Message Field	Tag	VR	Description of Field
Command Field	(0000,0100)	US	This field should be set to 8021H for the C-MOVE-RSP message.
Message ID Being Responded To	(0000,0120)	US	Shall be set to the value of the Message ID (0000,0110) field used in associated C-MOVE-RQ Message.
Data Set Type	(0000,0800)	US	This field indicates that the existence of the Data Set fields in the message. If the Data Set field is present, the value should be set anything other than 0101H. Otherwise, the value should be set to 0101H.
Status	(0000,0900)	US	This field provides the status of the C-MOVE service. For example, FF00 implies that the performing DIMSE-service-user is still continuing to move images to destination storage. Hence, additional C-MOVE-RSP messages will arrive.
Identifier	(No tag)	-	A Data Set that encodes the status information of the C-MOVE operation.

Section 3.2.4, Section 0, and Section 3.2.6 discuss the frequently used application-level services available to the DICOM AEs, and these services include storage, query, and retrieve services.

3.2.4 Storage Service Class

Storage service class is used to send Composite SOP instances from one DICOM AE to another. During the transfer, one AE takes on the Service Class User (SCU) role, which is responsible for sending. The peer AE performs as the Service Class Provider (SCP) role, which is responsible for receiving. The implementation of this storage service class employs the C-STORE DIMSE service. Figure 14 displays the message exchanges and PDU transactions between a modality and PACS for the storage service.

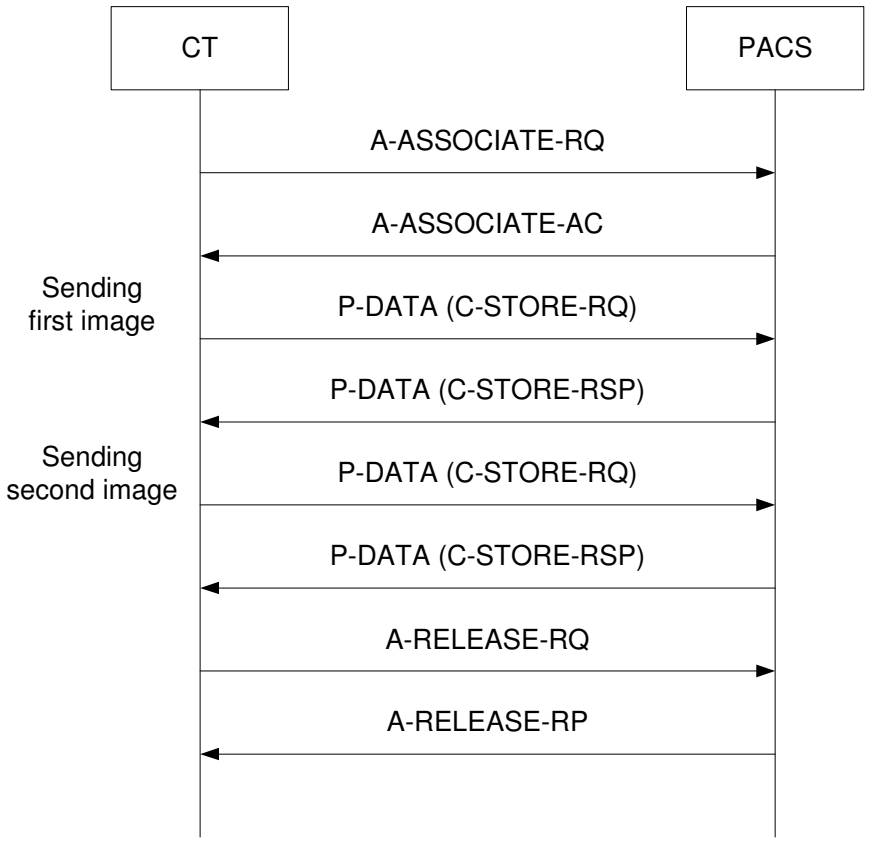

Figure 14. Storage transaction between a CT imaging device and PACS: CT sends two images to PACS.

CT performs the SCU role while PACS takes on the SCP role in the storage transaction example. The association is negotiated via the A-ASSOCIATE service before CT sends

the diagnostic images to PACS. After the association is established, each image is transferred from CT to PACS using C-STORE-RQ and C-STORE-RSP messages embedded in the P-DATA PDU. When the modality has no more images to send, the A-RELEASE service is employed to terminate the storage transaction.

3.2.5 Query Service Class

DICOM defines the query service class for one DICOM AE to perform Composite SOP instance information queries against another. The DICOM AE that sends the query requests takes on the SCU role while the peer AE that replies with the matching results performs the SCP role. The C-FIND DIMSE service is employed in the implementation of this query service class. Figure 15 shows the message exchanges and PDU transactions between a review workstation and PACS for the query service.

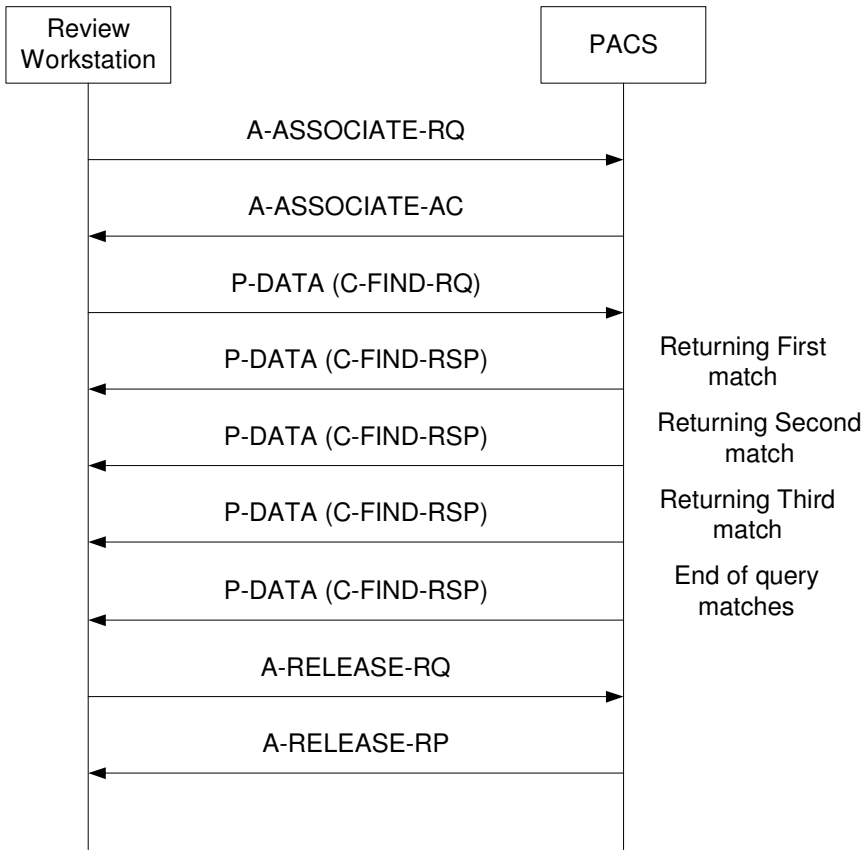

Figure 15. Query transaction between a review workstation and PACS: PACS returns three matching results to the review workstation.

The review workstation acts as the SCU while PACS operates as the SCP in the query transaction example. The A-ASSOCIATE service is employed to establish the association, and the A-RELEASE service is utilized to tear down the connection. The query request is sent from the review workstation to PACS using the C-FIND-RQ message embedded in the P-DATA PDU. After PACS executes the search, each matching result is returned back to the review workstation through one C-FIND-RSP message with the status value of FF00H. When PACS does not have any more matching result, the final C-FIND-RSP message with status value of 0000H is returned.

3.2.6 Retrieve Service Class

DICOM defines the retrieve service class for one DICOM AE to instruct the peer AE for transferring Composite SOP instances to another AE. This retrieve service class employs both C-MOVE DIMSE and C-STORE DIMSE services. In the retrieve transaction example shown in Figure 16, the review workstation takes on both the SCU role of the C-MOVE DIMSE service and the SCP role of the C-STORE DIMSE service. PACS performs the SCP role of the C-MOVE DIMSE service and the SCU role of the C-STORE DIMSE service. Before the review workstation sends the retrieve request to PACS, the association is negotiated via the A-ASSOCIATE service. After the association is established, the review workstation may request to retrieve a list of images from PACS using the C-MOVE-RQ message. Soon after receiving the request, PACS identifies the instances, and prepares to transfer those instances to the review workstation by creating an additional association through the A-ASSOCIATE service. With this new association, PACS sends each image to the review workstation using the C-STORE-RQ and C-STORE-RSP messages embedded in the P-DATA PDU. After each image transfer, the C-MOVE-RSP message with the status value of FF00H is returned to the review workstation via the original association. When PACS does not have any more images to send, the final C-MOVE-RSP message with status value of 0000H is also returned. To finish off the retrieve transaction, the two associations are torn down using the A-RELEASE service.

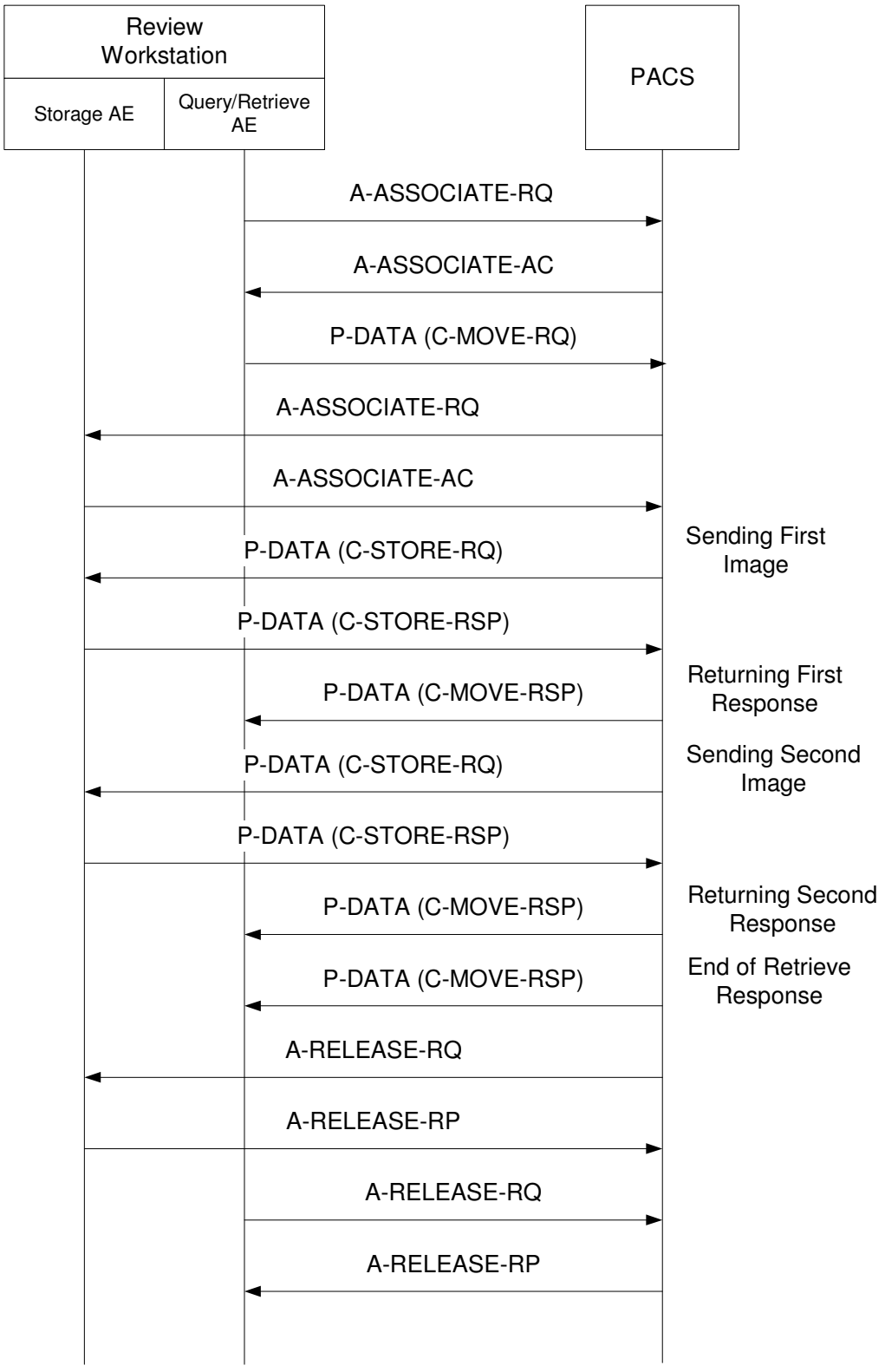

Figure 16. Retrieve transaction between a review workstation and PACS: review workstation retrieves two matching images from PACS.

3.3 DICOM Files

DICOM files are Composite SOP instances stored on the media, such as hard drive. The Composite SOP instance is an instance of the Composite Information Object Definition (IOD), where Composite IOD is made up of many related object-oriented abstract data models used to specify the information of the real-world objects. Examples of these object-oriented abstract data models are the Patient, General Study, General Series, General Image, Overlay Plane, MR Image, and CT Image module. Each Composite IOD has different usage requirements on the abstract data models, and these usage requirements include mandatory, conditional, and user option. When a DICOM file is declared to conform to the Composite IOD, all mandatory abstract data models must be present. Furthermore, the conditional abstract data models should exist if the specific condition is met. Abstract data models defined for the CT images are shown in Table 8.

Table 8. Abstract data models used in the CT image IOD [9].

Information Entity	Module	Usage
Patient	Patient	Mandatory
	Clinical Trail Subject	User Option
Study	General Study	Mandatory
	Patient Study	User Option
	Clinical Trail Study	User Option
Series	General Series	Mandatory
	Clinical Trail Series	User Option
Frame of Reference	Frame of Reference	Mandatory
Equipment	General Equipment	Mandatory
Image	General Image	Mandatory
	Image Plane	Mandatory
	Image Pixel	Mandatory
	Contrast/Bolus	Conditional: required if contrast media was used in this image
	CT Image	Mandatory
	Overlay Plane	User Option
	VOI LUT	User Option
	SOP Common	Mandatory

Various attributes are defined for conveying the instance value within each object-oriented abstract data model. These attributes may or may not be required in the Data Set depending on the attribute type:

- Type 1: this attribute must exist with valid and non-zero length data.
- Type 1C: when the specified conditions are met, this attribute must exist with valid and non-zero length data.
- Type 2: this attribute must exist, but the value can be zero length when unknown.
- Type 2C: when the specified conditions are met, this attribute must exist. However, the value can be zero length when unknown.
- Type 3: this attribute is optional, and the value can be zero length.

Frame of Reference module attributes are shown in Table 9.

Table 9. Attributes used in the frame of reference module [9].

ribute Name	Tag	Type	Attribute Description
Frame of Reference UID	(0020,0052)	1	Uniquely identifies the frame of reference for a Series.
Position Reference Indicator	(0020,1040)	2	Part of the patient's anatomy used as a reference, such as the iliac crest, orbital-medial, sternal notch, symphysis pubis, xiphoid, lower coastal margin, external auditory meatus.

Information about other IODs and module attributes are provided in the “DICOM Part 3: Information Object Definitions” standard [9].

4 Simple Network Management Protocol

Network management plays a crucial role in sustaining the health of a network. By monitoring and controlling devices within a network, problems can be efficiently resolved and in some cases completely avoided. As a network expands to account for more users and applications, information gathered via network management facilitates the plan for network growth.

In the early days of network management where networks were typically small, retrieving network information was not difficult with the use of low-level link layer protocols. As new link-layer technologies were introduced and networks grew in sizes, this traditional method of network management was not possible. To support for growing heterogeneous networks, SNMP was introduced. SNMP is a simple application-layer protocol designed for the exchange of information between network devices.

4.1 History of SNMP

Throughout the 1970s when the Transmission Control Protocol/Internet Protocol (TCP/IP) was being developed, network management was not a big concern because networks were typically small and were situated in a centralized location. The one network management tool that was effectively used was the Internet Control Message Protocol (ICMP). The echo/echo-reply capability of ICMP allowed for simple communications between network devices. One notable example of this application was the Packet Internet Groper Program (PING), which is a useful tool to monitor the operational status of network devices. However, as networks expand in sizes and become geographically diverse, the primitive capabilities offered by ICMP were not sufficient to monitor every device on a network.

The first set of standardized protocol designed specifically for network management was the Simple Gateway Monitoring Protocol (SGMP). Issued in 1987, SGMP outlined the network management framework and provided a simple means to monitor gateways. This protocol served as an interim response to address network monitoring while more

advanced protocols were being developed. In particular, three general-purpose network management concepts were being proposed:

- High-Level Entity Management System (HEMS):
  - Focused on extensive definition of network management information
- Simple Network Management Protocol (SNMP):
  - Extended SGMP to include management capabilities
- Common Management Information Protocol over TCP/IP (CMOT):
  - Concentrated on network management compliancy to the International Organization for Standardization (ISO)

Three proposals were carefully analyzed by the Internet Activities Board (IAB) and the results were presented in the Request for Comment (RFC) 1052. The HEMS proposal was withdrawn because too much effort would be required for the implementation. The IAB favored CMOT over SNMP due to its conformity with ISO. However, because CMOT specifications have not yet been finalized, SNMP was selected as the short-term solution before the long-term solution (CMOT) becomes available.

Due to the simplicity nature of the protocol, SNMP was rapidly adapted. By the time CMOT was finalized, SNMP has been widely deployed within the Internet. Hence, in May 1990, the IAB promoted SNMP as the standard for network management [1].

4.2 SNMP Features

The success of SNMP can be attributed by the feature sets offered by this simple protocol. The following list presents the benefits of SNMP:

- lightweight
- portable
- extensible
- standardized

SNMP is a simple lightweight protocol designed to minimize the performance impact on the operating device. It requires very little system and network resources, resulting in minimal implementation costs. SNMP was developed independent of the operating system and the programming language, and hence, it is portable across various platforms.

The core set of SNMP operations is not dependent on the operating devices. The protocol can be easily extended to support new devices and new management information. SNMP is a non-proprietary protocol, clearly defined and actively maintained by the IAB.

4.3 SNMP Architecture

The main components of the SNMP architecture consist of the SNMP agent and the SNMP manager. Figure 17 illustrates a typical arrangement of these components.

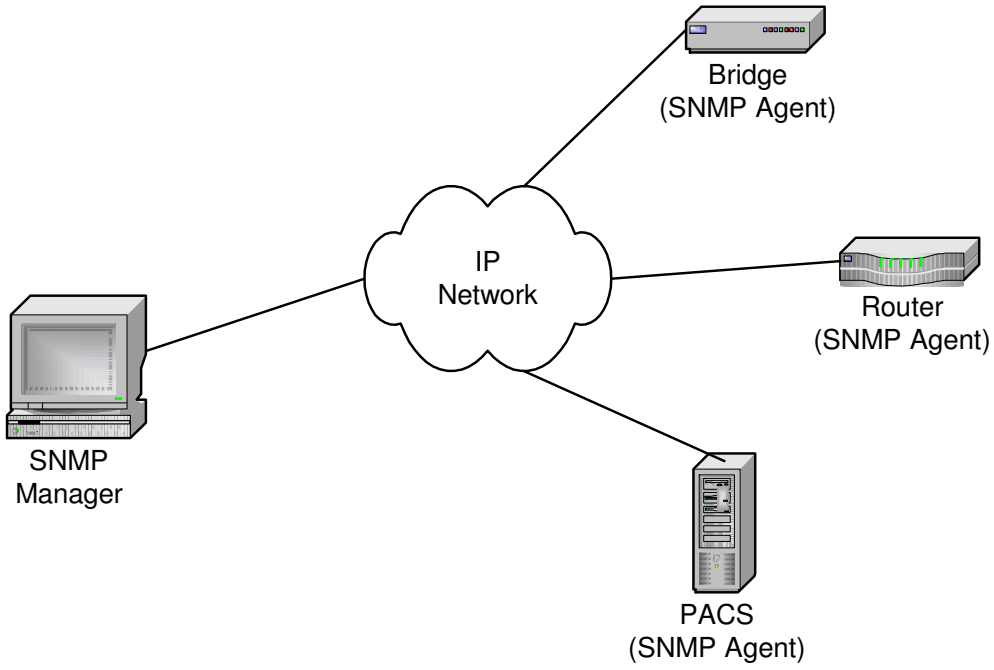

Figure 17. SNMP architecture: a SNMP manager manages several SNMP agents.

SNMP agents are programs residing in devices that require network management. For instance, network devices such as routers, bridges, and hubs are often equipped with SNMP agents. These SNMP agents offer the capability to retrieve and modify management information from network devices. Furthermore, these SNMP agents can be programmed to transmit unsolicited information when something noteworthy occurs.

The SNMP manger controls the type of information that is retrieved and modified at the SNMP agent. Typically residing in a computer, the SNMP manager provides a user

interface to control and monitor the network management process. Since communications between the SNMP manager and the SNMP agents are asynchronous, the SNMP manager can send multiple requests without receiving a response.

4.3.1 Management Information Base and Object Identifiers

To group and uniquely identify the vast amount of information that is maintained via SNMP, Object Identifier (OID) was introduced. OIDs are numerical strings that are analogous to telephone numbers where digits are hierarchically organized. Each number of an OID represents a specific branch of the management structure. For example, the first number of an OID is used to identify the standards organization. Currently, there are three main branches dealing with standards: Consultative Committee for International Telegraph and Telephone (CCITT), International Organization for Standardization (ISO) and joint ISO/CCITT. Most current network management activities occur in the ISO branch, with the OID 1.3.6.1 dedicated for the Internet community.

OIDs are used to represent two types of managed objects: scalar and tabular. Scalar objects represent single object instances whereas tabular objects combine multiple related managed objects into a single table. The last number of an OID can distinguish two different kinds of objects. A value of 0 designates a scalar object whereas a non-zero value denotes an index into a table.

During the network setup stage, OIDs are known to both the SNMP manager and the SNMP agents. When the SNMP manager wishes to query the network device for specific information, it includes the corresponding OID in the management request. Upon receiving the request, the SNMP agent interprets the OID and responds with the value of the requested parameter. For example, to query the available processor memory on a Cisco switch, the SNMP manager initiates an SNMP request containing the OID 1.3.6.1.4.1.9.9.48.1.1.1.6.1 [16]. Upon receiving the request, the Cisco switch examines its internal memory pool and generates the corresponding response.

Although OIDs are useful for accessing managed objects, the numerical nature of these identifiers makes it difficult for humans to interpret. Thus, group of OIDs are combined

to form a Management Information Base (MIB). MIB is a flat text file containing translations of OIDs to a human-readable format. Typically represented by an abstract tree with an unnamed root, each managed object is represented by the leaves of the tree. Figure 18 provides an example of a MIB tree. All SNMP parameters that are accessible in Cisco products are derived from the root of the Cisco node with an OID of 1.3.6.1.4.9.

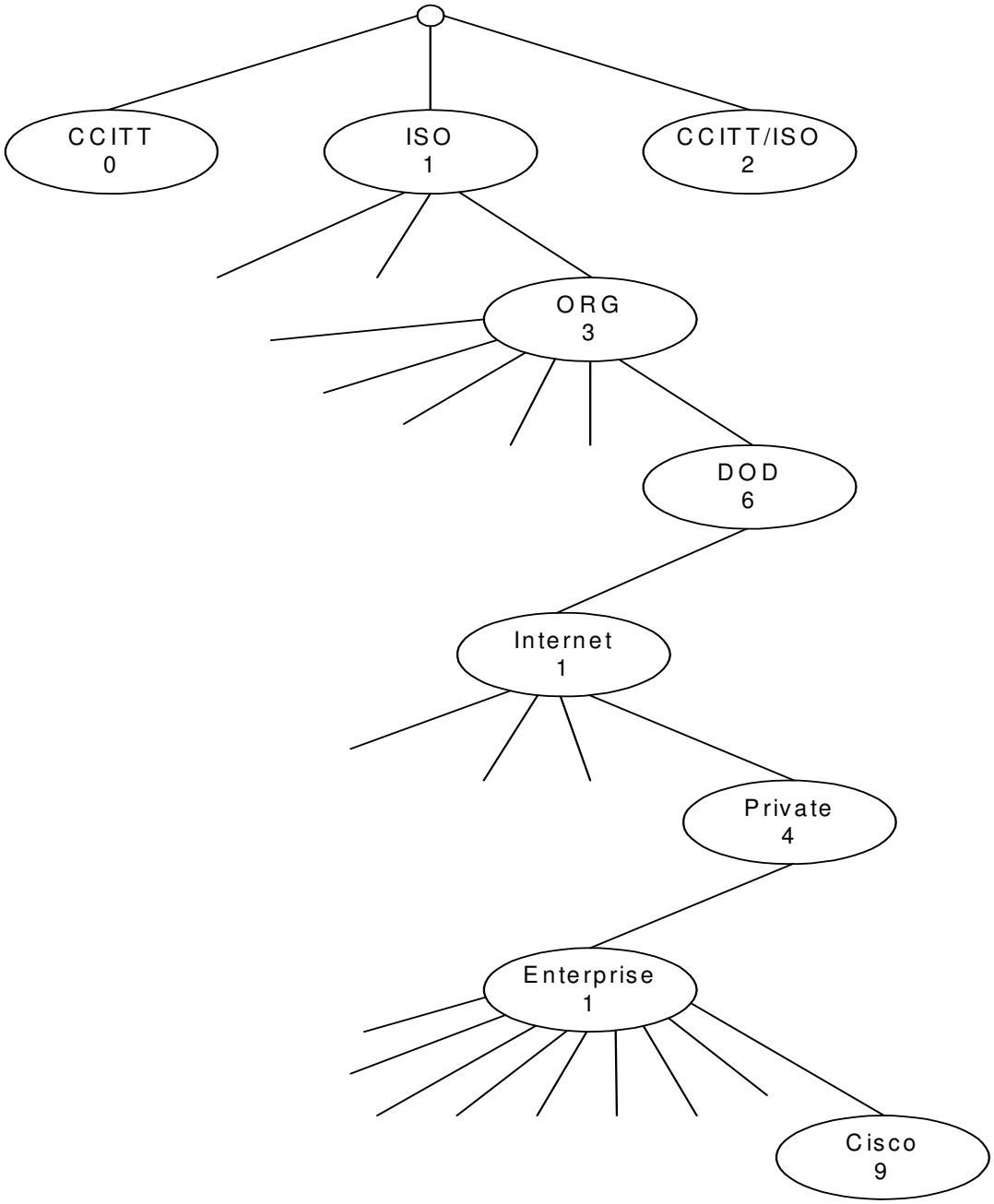

Figure 18. Example of an MIB structure for a Cisco router [8].

To access a MIB tree, it must be "installed" by both the SNMP manager and the SNMP agent. Installing a MIB involves placing the MIB in a location accessible by the SNMP module for OID translation. Due to the extra overhead involved in installing a MIB, many network applications work strictly with OIDs.

4.4 SNMP Operations

The basic SNMP operations follow a simple request/response procedure. The request is typically initiated by the SNMP manager, followed by a response from the SNMP agent. The initial version of the SNMP protocol, SNMPv1 as described in RFC 1157, introduces four basic SNMP operations: get, get-next, set, and trap. As illustrated in Section 4.6 and Section 4.7, newer versions of the protocol add new SNMP operations to enhance the network management process. In this Section, the four basic SNMP operations are presented.

4.4.1 Get Operation

The most common application of SNMP is to retrieve information from the SNMP agent. Initiated by the SNMP manager, a *get-request* is sent to the SNMP agent to retrieve the current values of the managed objects. Upon receiving the request, the SNMP agent attempts to retrieve the requested value(s) from the underlying device. If the information is successfully retrieved, a *get-response* message is replied to the SNMP manager containing the requested information. If, however, the SNMP agent is under heavy load, the request is ignored.

Processing a *get* request between the SNMP manager and the SNMP agent is illustrated in Figure 19. The sequence of events is:

1. The SNMP manager wishes to query the remaining processor memory on a router. It examines its internal SNMP database and determines the OID for this object. A SNMP *get-request* is sent to the SNMP agent with this OID.

2. Upon receiving the request, the SNMP agent retrieves the requested information from the router database.

3. The SNMP agent replies with a *get-response* indicating the remaining processor memory.

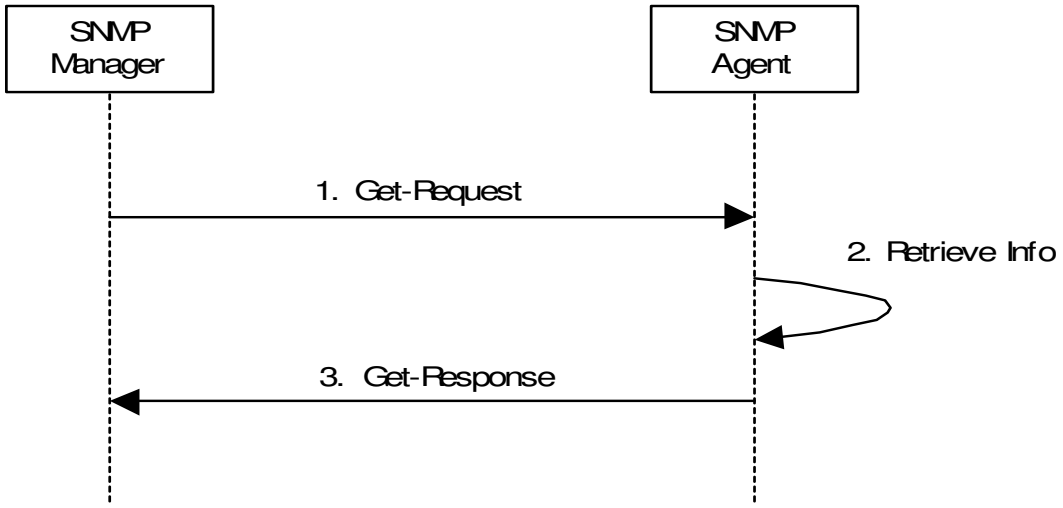

Figure 19. Example of the SNMP *get* request.

Although the *get-request* example contains only one OID, additional OIDs can be included in the request to retrieve more values. However, because the size of the *get-response* message is constrained by the agent's capability, where the maximum supported message size is expected to be 484 bytes [13], a *get-request* with multiple OIDs may not always be processed. In these situations, the SNMP agent returns an error with no data. Details of the error codes are presented in Section 4.5.

4.4.2 Get-Next Operation

The *get* operation presented in Section 4.4.1 is useful for retrieving scalar objects. The *get-next-request* operation is designed specifically for tabular objects. The SNMP manager can use this operation to transverse through the table in a sequential order. Figure 20 illustrates one application of the *get-next-request* operation. The sequence of events is:

1. The SNMP manager wishes to query the routing table containing the next hop addresses in a router. It determines the OID for the first row of the routing table and generates a *get-request* to the SNMP agent.

2. Upon receiving the request, the SNMP agent queries the routing table and retrieves the initial next hop address.

3. The SNMP agent replies with a *get-response* containing the retrieved information.

4. Instead of determining the next OID to access into the routing table, the SNMP manager initiates a *get-next-request* using the same OID provided in Step 1.

5. Since the *get-next-request* is used to retrieve the next instance that follows the current OID, the SNMP agent, upon receiving this request, retrieves the second hop address from the routing table.

6. Similar to the response for a *get* operation, the SNMP agent replies with a *get-response* containing the retrieved information. However, included in this response is also the OID for the second hop address of the routing table.

7. The SNMP manager extracts the OID returned in the *get-response* to access into the next row of the routing table.

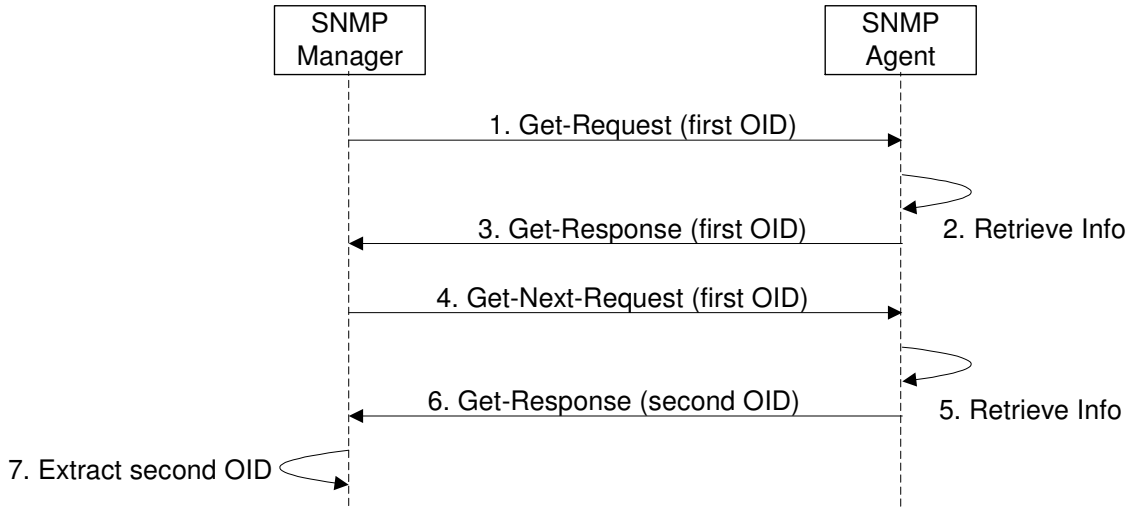

Figure 20. Example of the SNMP *get-next* request.

Multiple *get-next-request* operations can be issued by the SNMP manager until an error is returned from the SNMP agent indicating the last entry of the table has been reached. The *get-next-request* operation is useful to dynamically transverse a table when the table size is unknown in advance.

4.4.3 Set Operation

Apart from retrieving information on the managed objects, SNMP also provides an interface to modify values of the managed objects and to create a new row in a table.

Since different objects have different access control, not all managed objects are configurable. Figure 21 illustrates the usage of a *set* operation. The sequence of events is:

1. The SNMP manager wishes to modify the port speed of a router. It examines its internal SNMP database and determines the OID for this object. An SNMP *set-request* is sent to the SNMP agent with this OID.

2. Upon receiving the request, the SNMP agent first verifies if the object is configurable. If write-access is available, the new setting is applied at the router. If, however, the object is read-only, an error is returned.

3. The SNMP agent replies with a *get-response* reflecting the modified configuration.

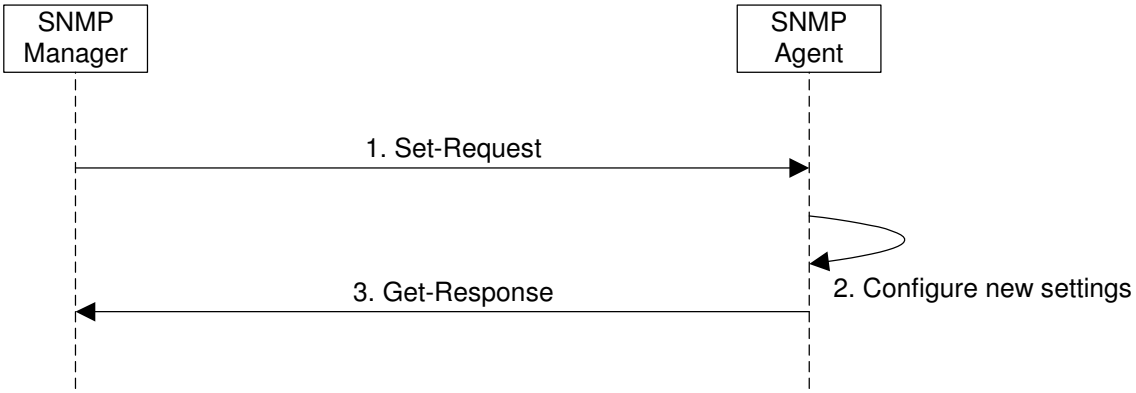

Figure 21. Example of the SNMP *set* request.

4.4.4 Trap Operation

While the SNMP operations discussed so far are initiated by the SNMP manager, the SNMP agent can be configured to initiate SNMP messages. The *trap* operation is used by the SNMP agent to asynchronously send SNMP messages to a per-defined location which is typically the SNMP manager address. The conditions that trigger *trap* messages are implementation-specific. Examples of the triggering events are:

- a network interface of the router has failed
- a specific port on the router has been impaired
- the fan on a router has failed

In contrast to other SNMP operations that employ a request/response handshaking, no acknowledgment is sent from the SNMP manager upon receiving a *trap* message. Hence, the *trap* message is especially prone to being lost when network problems arise. Figure 22 illustrates processing a *trap* operation.

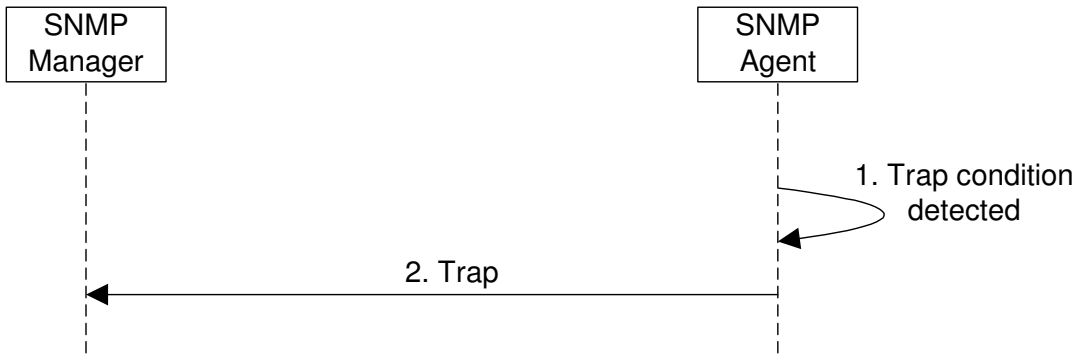

Figure 22. Example of the SNMP *trap* request.

4.5 SNMP Message Format

SNMP messages are divided into two parts. The first part of the message contains the header information which varies depending on the SNMP version. The second part is the payload component which is consistent across all SNMP versions. Figure 23 illustrates the SNMPv1 message format for the *get* and *set* operations.

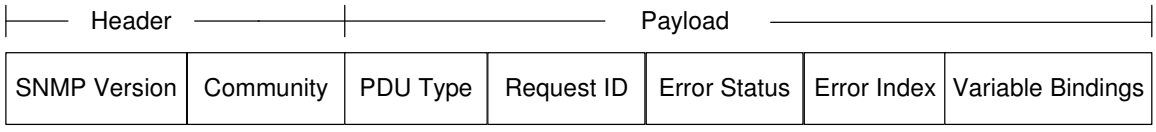

Figure 23. SNMPv1 g*et/set* message format.

The header component of an SNMPv1 message is composed of the *version* and the *community* field. The *version* field is used to ensure that both the SNMP manager and the SNMP agents are using the same SNMP version. The *community* field offers a weak form of authentication between the SNMP manager and the SNMP agents. Pre-defined during the setup stage, the correct *community* value must be provided in the SNNP request for

successful access to the managed objects. The reason why this is considered a weak form of authentication is because SNMPv1 does not support privacy. SNMP messages can be captured in the network and the *community* value can be easily extracted for unauthorized access.

The payload component of an SNMPv1 message contains the actual SNMP protocol data unit (PDU). This second part outlines the actual SNMP operation. The *PDU Type* field specifies the PDU types: *get-request*, *get-next-request*, *get-response*, and *set-request*. The *Request Identifier (ID)* field correlates the request with the response so that multiple requests and responses can be accommodated. The *Error-status* field is used in conjunction with the *Error-index* field to relay error information. Used exclusively by the SNMP agent in a *get-response* PDU, these two fields identify the status of the get/set operation. Error status interpretation in SNMPv1 is described in Table 10.

Table 10. SNMP error messages.

Error-status	**Description**
noError (0)	No problem is encountered. *Error-index* is set to 0.
tooBig (1)	Response to the request is too big to fit in a reply message. *Error-index* is 0.
noSuchName (2)	Information for the requested OID, repeated in *Error-index*, cannot be found.
badValue (3)	Used only in response to a *set-request*, this value is returned if the set operation attempted to modify an inconsistent value. *Error-index* specifies the OID with the conflicting format.
readOnly (4)	Write-access to a read-only object is denied. *Error-index* is not defined in this scenario.
genErr (5)	Generic error code to identify errors that can not be categorized into the above error categories. *Error-index* specifies the OID that triggers the error.

Finally, the *Variable-bindings* field defines the data component of the SNMP PDU. For a *get-request* and a *get-next-request* operation, this field specifies the OID(s) of the requested information. For the *set* operation, this field is interpreted as an <*OID*=*value*> pair, where *value* identifies the new value to set for the object specified in *OID*. The same <OID=value> format is used in a *get-response* operation, except that *value* identifies the value of the retrieved object.

The message format for SNMPv1 *trap* operation is presented in Figure 24. The header component of the trap message is identical to Figure 23, with the payload re-defined for trap notifications.

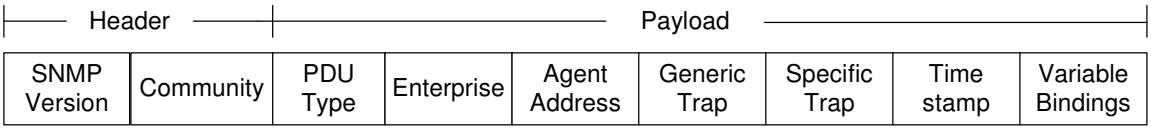

Figure 24. SNMPv1 *trap* message format.

The *enterprise* field of a *trap* message identifies the type of network entity generating the trap. Typically, this field has the same value as the System Object ID (sysObjectID), which is a unique value identifying the implementer of the SNMP agent. The *agent-address* field contains the IP address of the SNMP agent that triggers the trap. The *generic-trap* field denotes the trap category. Used in conjunction with the *specific-trap* and *variable-bindings* field, the *generic-trap* field defines seven traps as presented in Table 11.

Table 11. SNMP trap identifier.

Generic-trap	**Description**
coldStart (0)	The SNMP agent has rebooted and all the management variables accessible via the SNMP interface may have changed.
warmStart (1)	The agent has reinitialized but all the management variables accessible via the SNMP interface have been maintained.
linkDown (2)	One of the communication interfaces has been disabled. The identifier of the affected interface is provided in the *variable-bindings* field.
linkUp (3)	One of the communication interfaces has been enabled. The identifier of the affected interface is provided in the *variable-bindings* field.
authenticationFailure (4)	Received an SNMP message with an incorrect community field. An unauthorized access to the SNMP agent has been attempted.
egpNeighborLoss (5)	The SNMP agent has lost an Exterior Gateway Protocol (EGP) neighbor. The identifier of the peer is provided in the variable-bindings field.
enterpriseSpecific (6)	An enterprise-specific event has occurred. The *specific-trap* field can be used in conjunction with the *variable-bindings* field to relay enterprise-specific trap information.

4.6 SNMPv2

The rapid adaptation of SNMPv1 led to an increasing exposure of the protocol deficiencies. Some of the shortcomings include the inability to retrieve bulk data from the managed objects as well as the negligence to authentication and privacy. Thus, the second version of SNMP, namely SNMPv2, was introduced. Defined in RFC 1901 to RFC 1908, the main focus of SNMPv2 is to address the functional deficiencies of its previous version. Different variants of SNMPv2 have attempted to address the security deficiencies. However, due to the lack of consensus in the proposals, security enhancements are deferred to the next version of SNMP. In this Section, the major functional changes introduced in SNMPv2 are presented.

4.6.1 SNMPv2 Operations

SNMPv1 introduces four operations for network management. While these four operations are sufficient to provide basic configuration control, they are not designed to support large-scale networked systems. Thus, two new operations are introduced in SNMPv2: *get-bulk* and *inform*.

4.6.1.1 Get-bulk Operation

Although the *get* operation introduced in Section 4.4.1 is capable of retrieving multiple values from the managed object, the SNMP agent may not be able to return the requested values due to message size constraints at the SNMP agent. In these situations, the SNMP agent simply responds with an error of 'tooBig' in the *get-request* message and none of the requested information is returned. Consequently, the SNMP manager must separate the requested information into different *get* operations to retrieve all the desired data. This process becomes very inefficient when large amounts of data are required from different managed objects.

The *get-bulk* operation is designed specifically to address this inefficiency. Initiated by the SNMP manager, this operation informs the SNMP agent to include as much data as possible in the *get-response* message. This approach eliminates the need for repetitive *get*

or *get-next* operations from the SNMP manager. Figure 25 illustrates the usage of a *get-bulk* operation. The sequence of events is:

1. The SNMP manager wishes to query the routing table containing the next hop addresses in a router. It determines the OID for the first row of the routing table and generates a *get-bulk* request to the agent. Detailed contents of the request is given in RFC 1905.

2. Upon receiving the request, the SNMP agent queries the routing table and retrieves the complete list of next hop addresses.

3. The SNMP agent replies with a *get-response* containing the list of addresses.

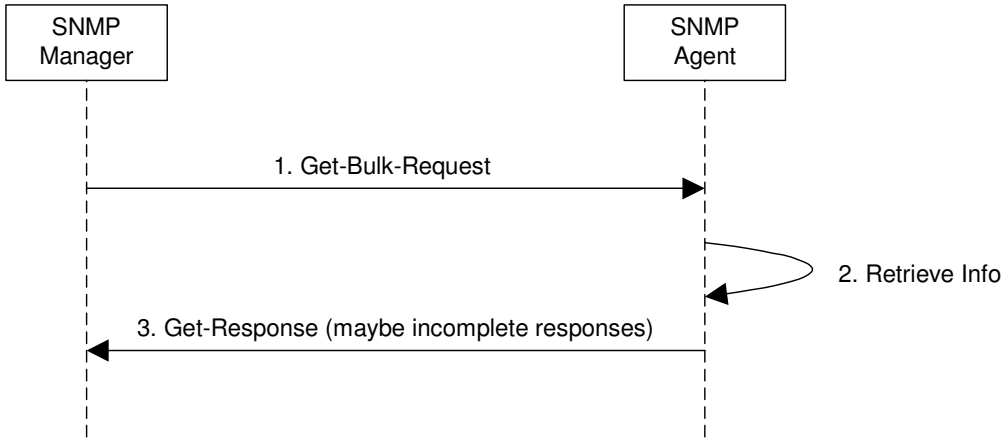

Figure 25. SNMPv2 *get-bulk* request example.

4.6.1.2 Inform Operation

In large-scale networked systems, multiple SNMP managers are typically used to maintain the complete infrastructure. Thus, SNMPv2 introduces the *inform* operation to account for the manager hierarchy. This new operation facilitates network management by allowing manager-to-manager communications. Figure 26 illustrates the usage of the *inform* operation. The sequence of events is:

1. A SNMP manager responsible for the backup network wishes to inform the root SNMP manager of network problems on one of its nodes. An *inform* message is initiated to the root SNMP manger describing the problem.

2. The root SNMP manager replies with an identical *inform* message indicating the successful receipt of the original message.

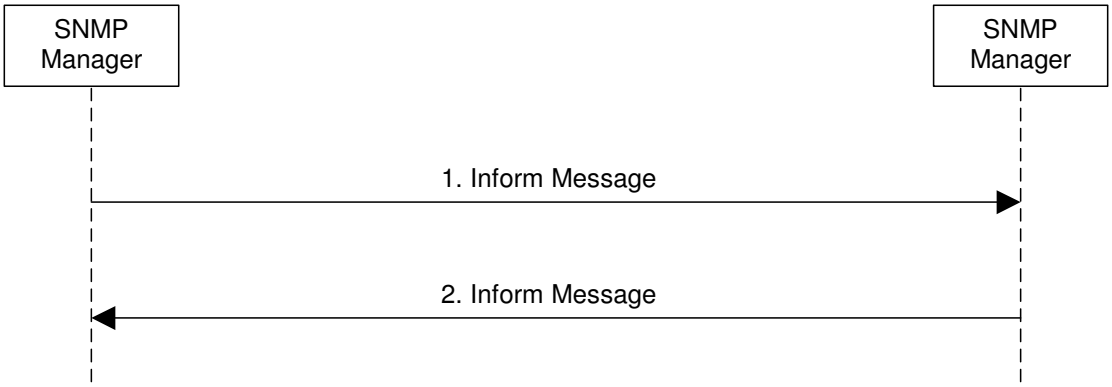

Figure 26. SNMPv2 *inform* example.

4.6.2 SNMPv2 Message Format

The basic message format for SNMPv2 is identical to that defined in SNMPv1. However, error handling has been expanded in SNMPv2 to account for additional error scenarios. New values, as described in Table 12, have been defined for the *error-status* field in SNMPv2.

Table 12. SNMPv2 error messages.

Error-status	**Description**
noAccess (6)	Attempt to set an inaccessible variable.
wrongType (7)	Wrong variable type is included in the set attempt.
wrongLength (8)	Specified value in the set attempt exceeds the allowable size.
wrongEncoding (9)	Wrong encoding is used in the set attempt.
wrongValue (10)	Attempt to set to a value that cannot be understood.
noCreation (11)	Attempt to create a nonexistent variable.
inconsistentValue (12)	Failed to set a variable under current circumstances (even though it can be changed under other circumstances).
resourceUnavailable (13)	System resources are not available to process the set operation.
commitFailed (14)	Generic error code to catch all set errors.
undoFailed (15)	Failed to undo all the sets operation prior to the point of failure.
authorizationError (16)	Unauthorized access to the agent has been attempted.
notWriteable (17)	Failed to accept a set operation even though the variable is writeable.
inconsistentName (18)	Failed to create a new variable under current circumstances (even though it could be created under other circumstances).

Additional modifications to SNMPv2 message format were made for trap processing. To simplify traps in SNMPv2, the trap message format outlined in Figure 24 has been removed. All traps messages use identical format as the *get* and *set* PDUs.

Access into tabular objects has also been enhanced in SNMPv2. A new column called *RowStatus* must be added to any tables that are configurable via SNMPv2. The purpose of this column is to prevent simultaneous edits by multiple managers. The value in this column reflects the accessibility of each row in a table.

Due to the additional complexity in SNMPv2, implementations for SNMPv2 are typically much larger and more complicated than its SNMPv1 counterparts. Regardless, both versions of SNMP have been commonly used. However, due to the lack of security features in both protocols, its applications are typically intended for read-only access. To provide for secured and authenticated read-write access, SNMPv3 is introduced.

4.7 SNMPv3

SNMPv3 is introduced specifically to address the insecurity inherent in the previous two versions. This new version is built upon the first two SNMP versions, so it is not a stand-alone replacement. SNMPv3 adds authentication and privacy for network management.

4.7.1 Authentication

Authentication is the process used to confirm the identity of the originating party. To incorporate authentication protocols within the SNMP infrastructure, both the SNMP manager and the SNMP agent share a secret authentication key that is typically derived using a secure hash function. Two authentication protocols have been defined for SNMPv3: the mandatory authentication protocol is HMAC-MD5-96 using the MD5 secure hash function, while the optional authentication protocol is HMAC-SHA-96 using the SHA-1 hashing algorithm. Details of these two authentication protocols are beyond the scope of this project. However, an overview of the authentication application in SNMP is provided in Figure 27.

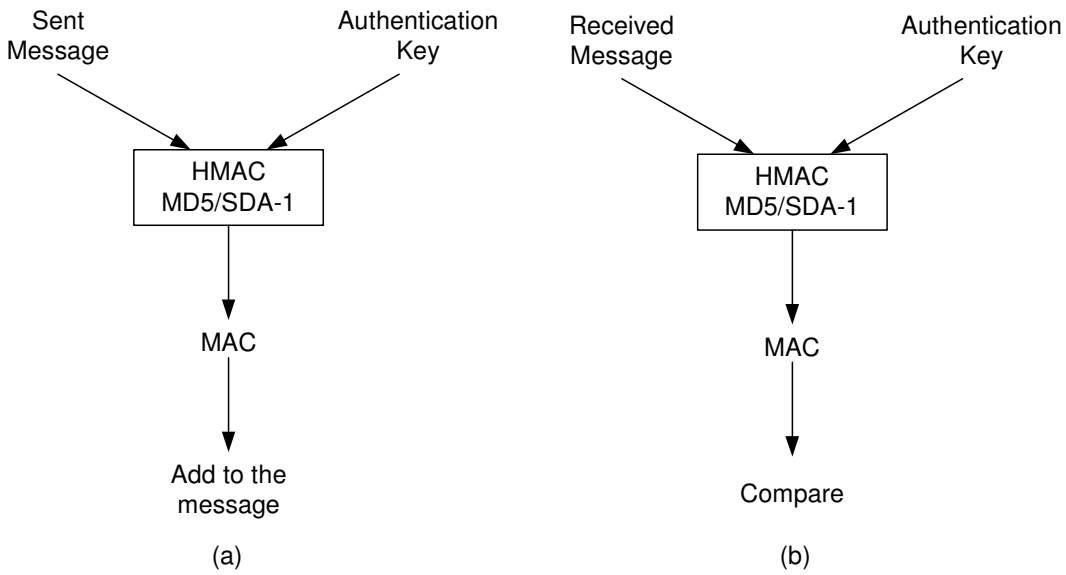

Figure 27. SNMP message authentication: (a) send authentication process (b) receiver authentication process.

Based on a shared authentication key, the originator calculates the Message Authentication Code (MAC) and adds it to the message. Upon successful authentication at the terminating party, the originator authenticity and the message integrity can be assured.

Apart from providing authentication of the originator and message integrity, SNMPv3 also incorporates a timeliness functionality to prevent message tampering. A complex synchronization mechanism is used to determine the communication time with a remote engine. If a response to a message is received outside the estimated time window, the message is ignored. Typically, this acceptable time window is set to 150 seconds.

4.7.2 Privacy

To provide privacy in SNMPv3, encryption protocols must be used. However, since restrictions on cryptographic technology vary between countries, privacy is optional in SNMPv3. If an SNMPv3 implementation chooses to employ encryption, the Data Encryption Standard (DES) encryption algorithm must be used.

Similar to authentication processing, a privacy key must be created between the SNMP manager and the SNMP agent for encryption. This key is used to encrypt only parts of the

SNMP message. Details concerning the actual encryption process are beyond the scope of this project. However, an overview of the encryption application is presented in Figure 28.

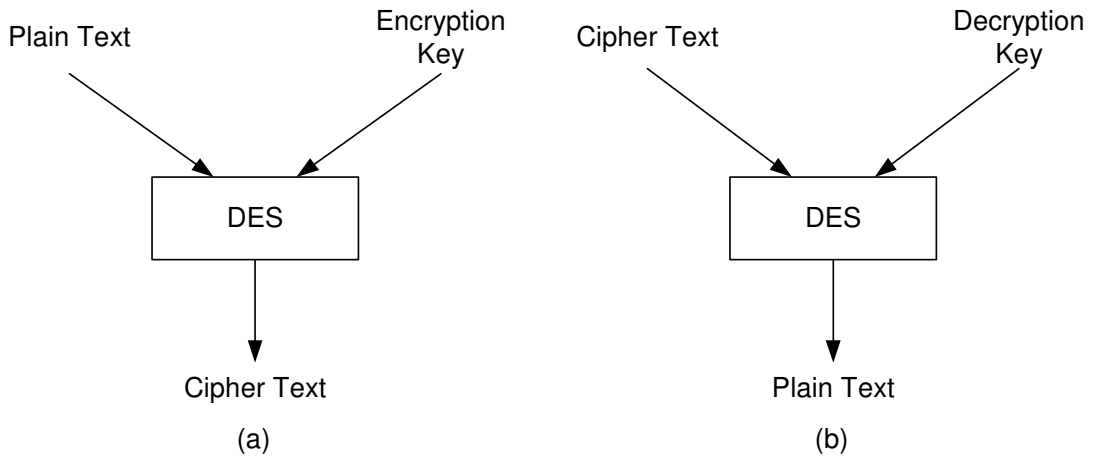

Figure 28. SNMP message encryption: (a) sender encryption process (b) receiver decryption process.

Three security levels have been introduced in SNMPv3. The lowest level provides no authentication or privacy, which is equivalent to what SNMPv1 and SNMPv2 offer. The second level uses authentication but no privacy. Finally, the highest security level is imposed when both authentication and privacy is in used.

4.7.3 Access Control

Both the authentication and the privacy protocols provide security functionalities at the manager and agent levels. Access control protocols, however, provide security functionalities at the PDU level. Access control protocols are used within SNMP to determine whether the managed objects can be modified by a remote entity.

In SNMPv3, the access control mechanism is called View-Based Access Control Model (VACM). Introduced in RFC 2575, VACM defines access rights on a per-group basis. Different SNMP managers can be assigned with different groups, with each group being used to represent a specific security level. For example, the root SNMP manager can be granted full access rights so that all managed objects can be modified. Similarly, a different SNMP manager could be assigned to a different group so that only read

permissions are allowed. By implementing access control within the SNMP infrastructure, a more secured network management process can be created.

5 Motivation for Applying SNMP to Medical Systems

Currently available PACS in the market provides very limited user interfaces. A majority of the services used in PACS are implemented as background tasks and, hence, users are not prompted upon error occurrences within PACS. Therefore, detecting and resolving errors found in PACS can be time consuming. Furthermore, since diagnostic imaging devices are typically operated by different technologists, they may not possess the necessary computing expertise to resolve issues within modalities. Thus, the time required to resolve errors found in diagnostic imaging devices can also be long. Finally, communication channels between PACS and diagnostic imaging devices may also be impaired. Since no tool is available to detect network connectivity problems between the two entities, communication impairments can be difficult to investigate.

Due to high number of errors that can occur within PACS and between PACS and diagnostic imaging devices, most hospitals hire PACS administrators to manage and maintain these systems. However, because most hospitals only employ at most two PACS administrators to manage an average of 100 medical systems, bottlenecks are created if multiple systems report problems and PACS administrators are not readily available. Consequently, services from diagnostics devices may be interrupted and the patients' diagnosis may ultimately be delayed.

The preceding deficiencies can be overcome by integrating SNMP support within every medical system and adding a centralized management tool to monitor and control the systems. In doing so, features such as statistics collection, device configurations, and error detections can be easily added to medical devices. Consequently, these features may reduce the total cost of PACS ownership [14]. The following scenarios describe in details how the introduction of SNMP support can help to avoid or efficiently resolve the inherent problems found within medical systems.

Case 1: PACS Storage Failures

Diagnostic images generated by DICOM modalities are typically stored in PACS. Once the number of images in storage reaches a pre-defined maximum storage capacity, PACS

automatically moves older images to the archive system to free up storage capacity. However, if this auto-archive capability is impaired, the number of images stored within PACS will grow until the system becomes completely full and no additional images can be stored. At this time, none of the modalities can perform scans because no space is available to store new images. Consequently, patients' treatments are delayed since radiologists and physicians rely on the technology of modalities to diagnose patients.

This problem can be addressed by adding SNMP support within PACS. When an archiving problem is detected, an SNMP trap message can be sent to the central management tool. By detecting the problem earlier on, the time spent on moving the backlogged images to the archive system can be avoided. Furthermore, by adding a notification trigger when an archive error is detected, the source of the problem can be identified more easily. This way, failure to send images from modalities to PACS will not be interrupted as an archiving problem by mistake.

Case 2: Unacknowledged PACS Updates

When sending over a large number of images such as a study comprising of over 100 images to PACS, technologists may not notice the transmission errors of a few images. If this oversight is not detected by the radiologist undergoing the diagnosis, and if the missing images actually indicate signs of abnormalities, the patient may be mis-diagnosed. Then, not only is the patient’s health in jeopardy, the hospital may also face lawsuits. If a centralized management tool is introduced to detect the missing images, SNMP trap messages can then be sent to the PACS administrator. The PACS administrator can subsequently notify the technologist and quickly resolve the mistake.

Case 3: PACS Network Impairments

During heavy network traffic, network connections between modalities and PACS may be dropped. Software applications such as Ethereal packet sniffer are used for troubleshooting network connectivity issues. However, since these tools are typically third party software applications, PACS administrators may not have the knowledge to operate them. If a centralized management tool based on SNMP is available, PACS administrators can query the network interfaces cards, routers, PACS, and modalities for

network statistics, system status, and device configurations. By doing so, network issues can be identified more quickly and problems can be resolved more efficiently.

6 Design and Implementation of PACS Monitor System

Section 5 discusses the motivation of our project. This Section provides an overview of our design and implementation. The objective of this project is to develop the PACS Monitor system that performs the PACS management via the SNMP protocol.

6.1 Hardware Platform

To demonstrate our development, two computers are connected via a router. One end is the PACS system from McKesson Medical Imaging Group. The other end is the developed PACS Monitor system. Figure 29 illustrates the hardware development platform.

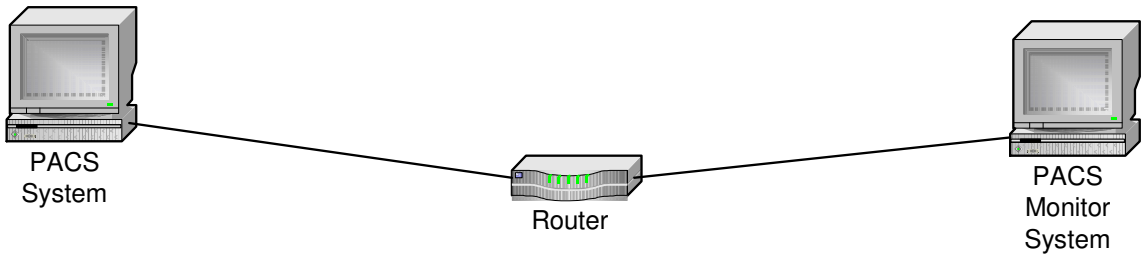

Figure 29. Hardware development platform.

6.2 Software Architecture

Our software architecture is organized into several layers of PACS and the PACS Monitor system. In PACS, these layers include the PACS services, PACS SNMP extension agent, Microsoft SNMP service and Microsoft TCP/IP stack. In the PACS Monitor system, these layers are the PACS Monitor GUI, PACS SNMP manager, Microsoft SNMP library, Microsoft SNMP service and Microsoft TCP/IP stack. Our implementation involves the upper two layers in the PACS system and the PACS Monitor system. Only SNMPv1 is implemented since we wish to have a simple design to prove this medical device management concept. Figure 30 illustrates the software architecture.

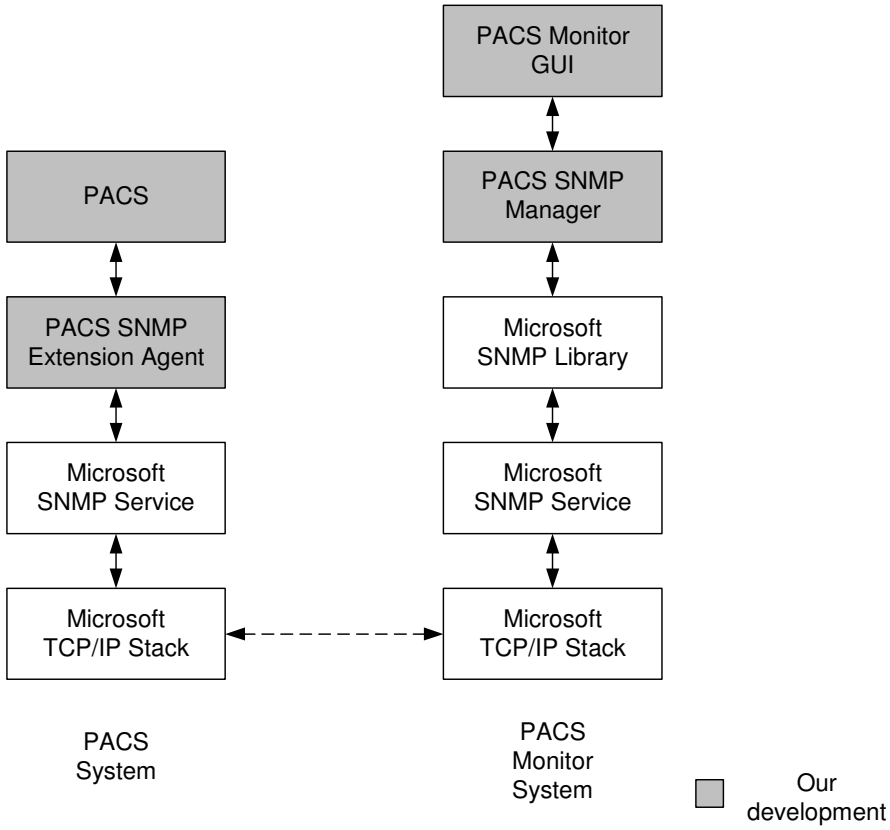

Figure 30. Software architecture diagram.

6.2.1 PACS Monitor GUI

The PACS Monitor GUI is an executable file developed using MFC in Visual C++. As shown in Figure 31, this layer provides the graphical user interface for the PACS administrator to manage PACS. The left (system) window shows all modalities connecting to PACS. Information regarding a modality is displayed on the main screen. The bottom (trap) window notifies the PACS administrator of errors. The PACS Monitor GUI does not perform any major tasks other than calling functions that are within the underlying layer.

The PACS Monitor GUI is developed using the object-oriented paradigm. The main screen is implemented using the *CMainFrame* class, which contains two *CControlBar* objects for the system window and the trap window. The object containment allows the system window and the trap window to dock on any edge of the main screen. The system

window displays the modalities using the *CTreeCtrl* class, while the trap window shows the error message via the *CListCtrl* class. Figure 32 illustrates the class containment of the PACS Monitor GUI.

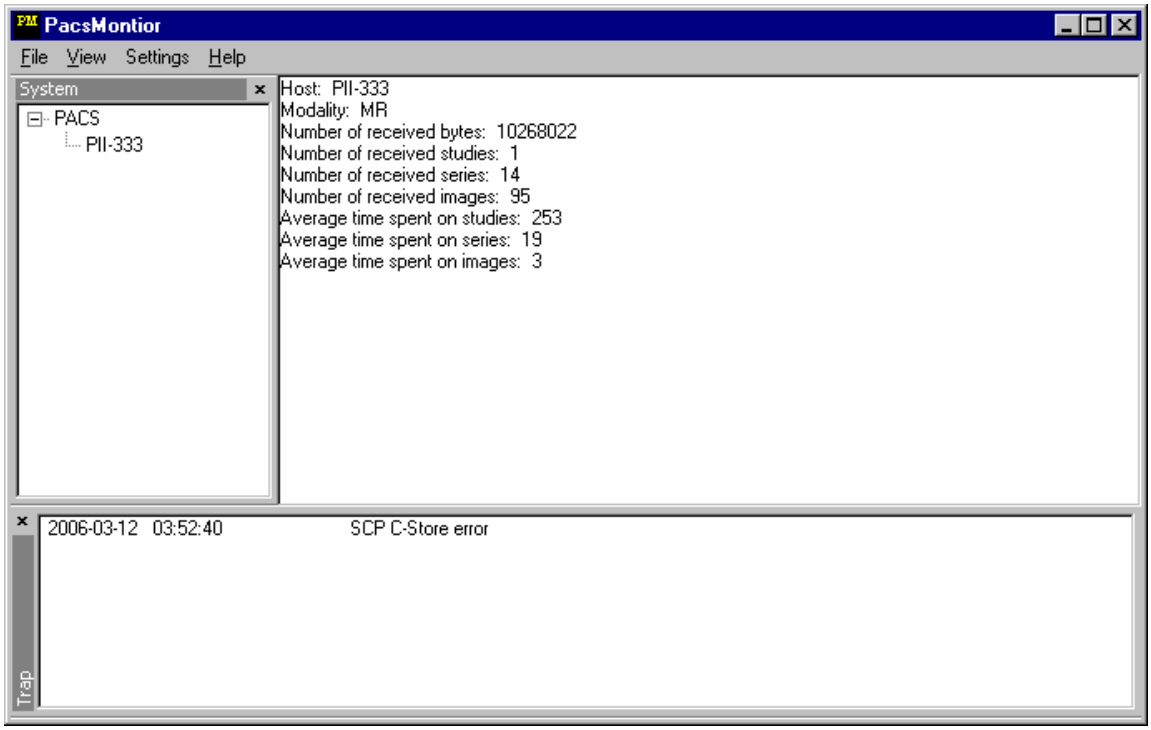

Figure 31. A snapshot of the PACS monitor GUI.

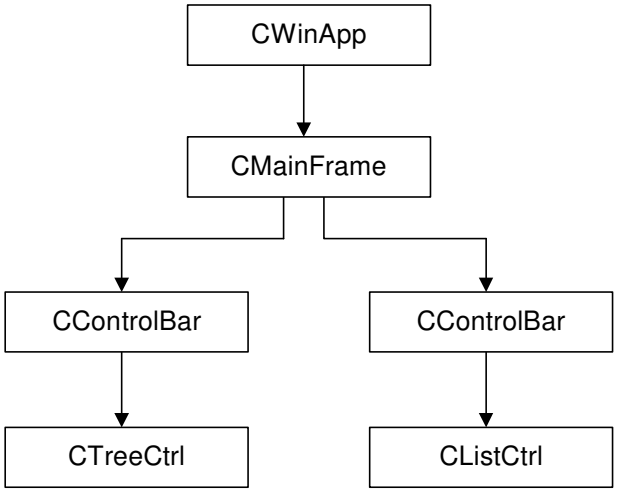

Figure 32. PACS monitor GUI class containment.

6.2.2 PACS SNMP Manager

The PACS SNMP manager is a 32-bit Dynamic Link Library (DLL) developed using Visual C++. This layer is responsible for transmitting SNMP requests from the PACS Monitor system to PACS. Furthermore, the PACS SNMP manager handles the SNMP trap upon receiving a notification.

To avoid possible race conditions, the DLL is implemented to allow the attachment of a single process. If a second attachment is initiated, an error dialog box will be generated. In other words, the PACS administrator cannot simultaneously open two PACS Monitor GUIs. This constraint is implemented inside the *DllMain* function, where the check is performed during the attachment of the PACS Monitor GUI.

Upon the attachment, the PACS Monitor GUI can configure PACS and retrieve statistics via the main control loop in the PACS SNMP manager. The PACS SNMP manager hides SNMP details (such as OID) from the PACS Monitor GUI, simplifying the design of the graphical user interface. Figure 33 shows the flowchart of the main control loop in the PACS SNMP manager.

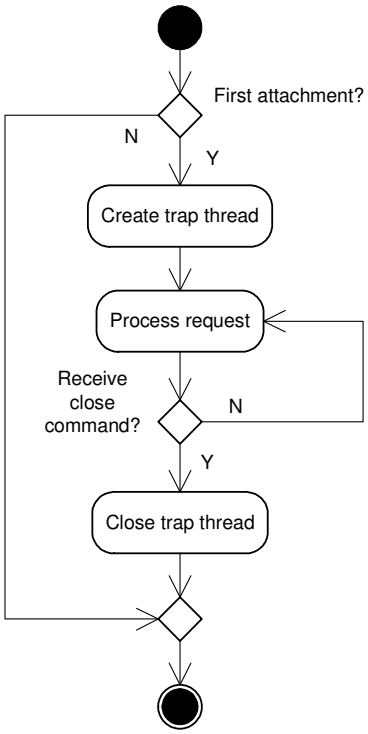

Figure 33. The main control loop in the PACS SNMP manager.

During the startup process, the DLL also spawns a thread to handle SNMP traps. A new thread is necessary due to the asynchronous nature of SNMP traps. Once a SNMP trap is received, the thread notifies the PACS Monitor GUI via the registered callback function. However, the thread is terminated upon receiving a close event. Figure 34 illustrates the flowchart of the trap thread in the PACS SNMP manager.

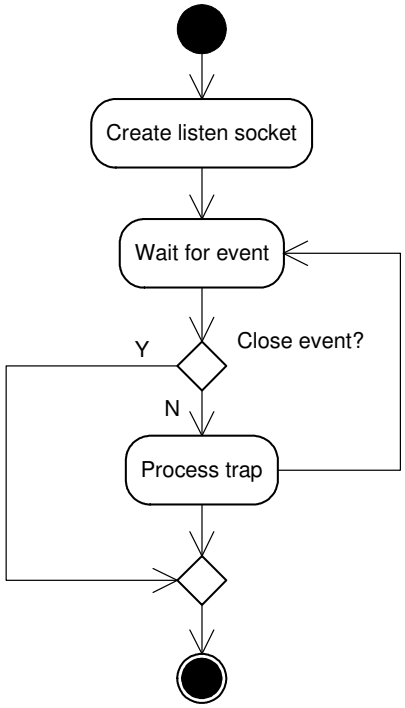

Figure 34. The trap thread flowchart in the PACS SNMP manager.

6.2.3 SNMP Extension Agent for PACS

SNMP Extension Agent for PACS, named "ADSCP_SNMP", is a DLL that works in conjunction with the SNMP Windows service. Four public functions as required by the SNMP Windows service were developed: SnmpExtensionInit(), SnmpExtensionClose(), SnmpExtensionTrap(), and SnmpExtensionQuery(). When the SNMP Windows service starts up, SnmpExtensionInit() is called to initialize the resources in ADSCP_SNMP. Similarly, SnmpExtensionClose() is invoked to release resources when the SNMP Windows service terminates. SnmpExtensionTrap() is called by the SNMP Windows service to generate a trap message to the PACS SNMP Manager upon error detection in PACS. In order for the PACS Monitor GUI to set configurations or collect statistics from

PACS, the SNMP Windows service calls SnmpExtensionQuery(). This function encapsulates the MIB tree used in the PACS system. When SnmpExtensionQuery() is called, the OID is resolved to the managed object via the MIB tree and the corresponding operation is performed on the managed object. Table 13 provides all the OIDs and their descriptions defined in this project.

Table 13. OIDs used in the storage service of the PACS system.

OID	Type	Access	Description
1.2.840.113711.593.1.1	INTEGER	Read-only	Interval between successive updates of the PACS statistic table
1.2.840.113711.593.1.100.x	INTEGER	Read-only	Row index of the PACS statistic table
1.2.840.113711.593.1.101.x	OCTET STRING	Read-only	Host name for row "x"
1.2.840.113711.593.1.102.x	OCTET STRING	Read-only	Modality for row x
1.2.840.113711.593.1.103.x	INTEGER	Read-only	Number of studies received in the past hour for row x
1.2.840.113711.593.1.104.x	INTEGER	Read-only	Number of series received in the past hour for row x
1.2.840.113711.593.1.105.x	INTEGER	Read-only	Number of images received in the past hour for row x
1.2.840.113711.593.1.106.x	INTEGER	Read-only	Number of bytes received in the past hour for row x
1.2.840.113711.593.1.107.x	TimeTicks	Read-only	Average time spent for receiving studies in the past hour for row x
1.2.840.113711.593.1.108.x	TimeTicks	Read-only	Average time spent for receiving a series in the past hour for row x
1.2.840.113711.593.1.109.x	TimeTicks	Read-only	Average time spent for receiving images in the past hour for row x
1.2.840.113711.593.2.101.1	OCTET STRING	Read-write	AE Title of this storage service
1.2.840.113711.593.2.103.1	INTEGER	Read-write	Port number of this storage service

OID	Type	Access	Description
1.2.840.113711.593.2.104.1	INTEGER	Read-write	PDU size of this storage service
1.2.840.113711.593.2.106.1	INTEGER	Read-write	Timeout value of this storage service
1.2.840.113711.593.2.107.1	OCTET STRING	Read-write	Implementation Class of this storage service
1.2.840.113711.593.2.108.1	OCTET STRING	Read-write	Implementation Version of this storage service
1.2.840.113711.593.2.109.1	N/A	N/A	
1.2.840.113711.593.2.109.1.1.y	INTEGER	Read-write	Row index of the supported SOP class of this storage service
1.2.840.113711.593.2.109.1.2.y	OCTET STRING	Read-write	Supported SOP class of this storage service for row y
1.2.840.113711.593.2.110.1	N/A	N/A	
1.2.840.113711.593.2.110.1.1.y	INTEGER	Read-write	Row index of the supported Transfer Syntax of this storage service
1.2.840.113711.593.2.110.1.2.y	OCTET STRING	Read-write	Supported Transfer Syntax of this storage service for row y

The statistic information for the same imaging device, such as the host name, modality, and number of studies received in the past hour, uses the same value for the last index of the OID. Given these OID definitions, the MIB tree can be easily expanded to account for additional imaging devices in the future.

6.2.4 PACS

The source codes for PACS used in this project are courtesy of McKesson Medical Imaging Group. In our project, we have extended PACS from supporting the storage service to providing statistic collection and error reporting. Furthermore, the introduction of the PACS Monitor GUI allows for the modification of AE settings to be saved in the configuration file. In doing so, the new settings will be readily available upon the next association request. Figure 35 illustrates the software design of the PACS extension.

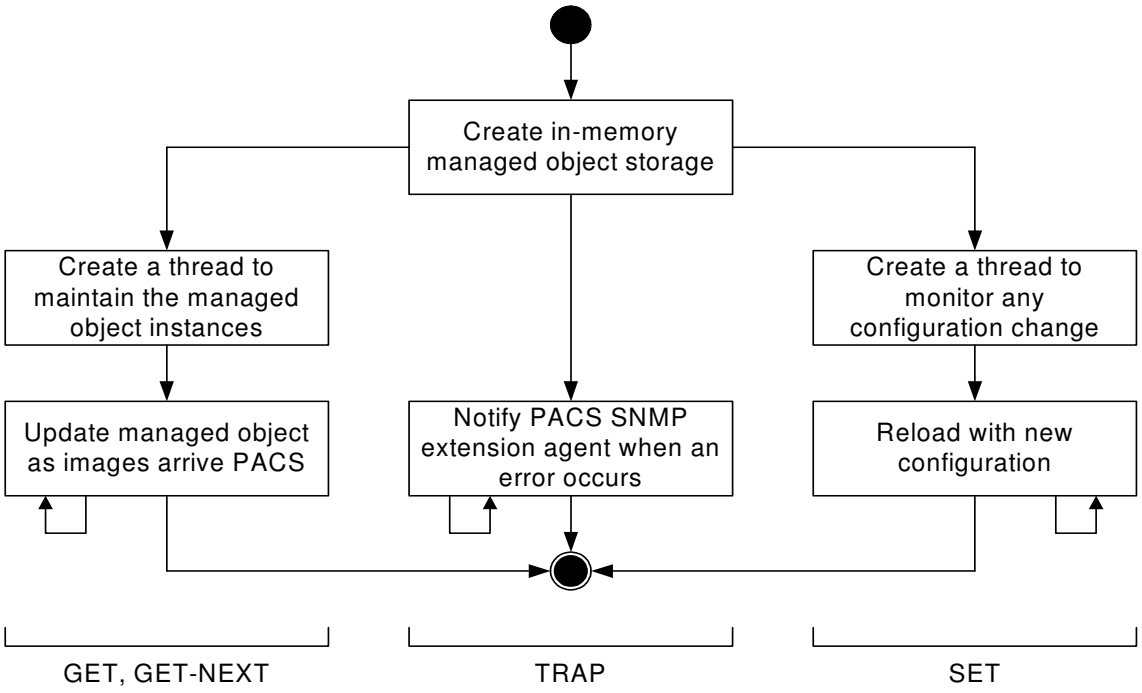

Figure 35. The PACS extension software design.

7 PACS SNMP Manger: System Testing and Validation

To confirm the functionality of the PACS Monitor system, three tests have been performed. We present here a summary of the verification and validation processes.

7.1 Module Test

The initial phase of our verification process is to compile all modules in debug mode. By inserting breakpoints at appropriate sections of the source code, the operation of each functional block can be validated. Furthermore, by inserting breakpoints at the interfaces between functional blocks, the interaction between different modules can be confirmed.

7.2 Message Test

The next phase of our verification process is to confirm SNMP transactions between the PACS SNMP manager and the PACS SNMP extension agent. By installing Ethereal packet sniffers at both entities, SNMP messages sent to the network can be captured and analyzed for packet contents and protocol compliancy. The setup presented in Figure 36 is configured to validate the SNMP transactions.

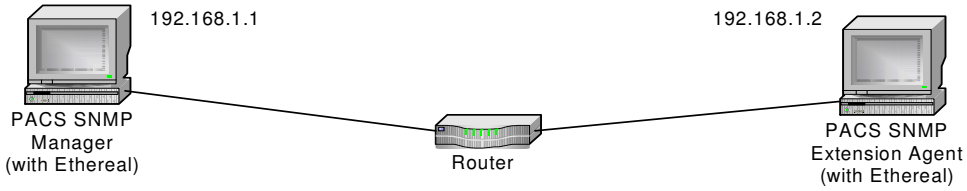

Figure 36. SNMP transaction test setup: a PACS SNMP manager connects to a PACS SNMP extension agent.

During the test process, all SNMPv1 commands have been used and the result has been verified. A set of action sequences is:

1. [Frame 1 – 2] *SNMP Get*: get the Application Entity Configuration of PACS.
2. [Frame 3 – 4] *SNMP Set*: set the Application Entity Configuration of PACS.
3. [Frame 5 – 10] *SNMP Get-Next*: get hostnames of medical imaging devices.
4. [Frame 11] *SNMP Trap*: notify of an error occurrence during the storage transaction between a medical imaging device and PACS.

The raw network captures of the corresponding frames are given in Appendix A.

7.3 System Test

The final phase of our verification process is to confirm the overall functionality of the PACS Monitor system. Using the test program from McKesson Medical Imaging Group, a generic computer can be used to simulate a modality. Figure 37 illustrates the system test environment, where multiple modality simulators are connected to PACS.

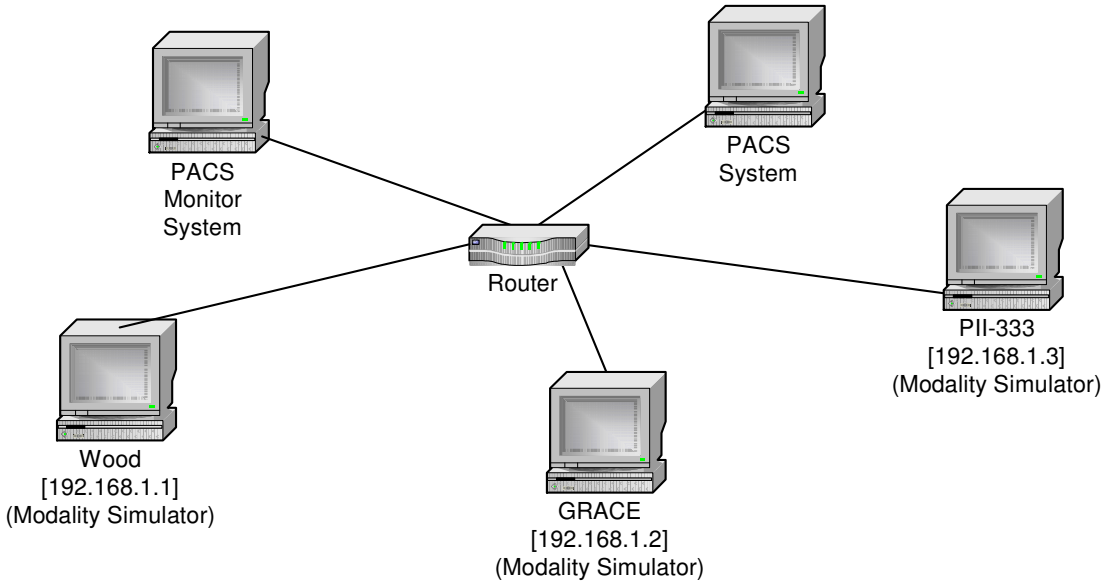

Figure 37. System test setup: PACS monitor system manages the connectivity between PACS system and three modality simulators.

During the system test process, the modality simulators generate storage service requests to PACS. The PACS Monitor system connects to PACS for statistic collection. A set of action sequences is:

1. *Initialization*: PACS starts.
2. *Query for hostnames*: PACS Monitor system retrieves hostnames.
3. *Query for host's statistics*: PACS Monitor system retrieves the statistics for the host with host index = 1.
4. *Error Notification*: PACS sends error notification to the PACS Monitor system.

The corresponding PACS logging is given in Appendix B.

8 Conclusion

The objective of our project is to develop the PACS Monitor system that provides statistics collection, configuration management and fault management for the PACS administrator. This paper provides an overview of a hospital environment, the DICOM communication, and the SNMP management protocols. Furthermore, we explored the utilization of the SNMP as the communication protocol between PACS and the PACS Monitor system. Finally, we present our design and implementation of the PACS Monitor system.

We have successfully implemented the project where the management data of the storage service request is defined and developed. The PACS Monitor system can retrieve statistics from registered modalities, configure AEs, and receive error notifications from PACS. Through a series of testing, we have verified the implementation of the proposed PACS Monitor system. Future enhancements of this project may include definitions of the remaining PACS services and provision of the SNMPv3 support.

Appendices

Appendix A Message Test Network Capture

```
-------------------------------------------------------------------------
Frame No.   Time        Source          Destination    Protocol
1           0.000137    192.168.1.1     192.168.1.2    SNMP
-------------------------------------------------------------------------
Frame 1 (170 bytes on wire, 170 bytes captured)
Ethernet II, Src: 00:a0:cc:e8:45:0d, Dst: 00:a0:cc:e7:95:3d
Internet Protocol, Src Addr: 192.168.1.1, Dst Addr: 192.168.1.2
User Datagram Protocol, Src Port: 1166, Dst Port: snmp (161)
      Version: 1 (0)
      Community: public
      PDU type: GET (0)
      Request Id: 0x00000000
      Error Status: NO ERROR (0)
      Error Index: 0
      Object identifier 1: 1.2.840.113711.593.2.101.1
      Value: NULL
      Object identifier 2: 1.2.840.113711.593.2.103.1
      Value: NULL
      Object identifier 3: 1.2.840.113711.593.2.104.1
      Value: NULL
      Object identifier 4: 1.2.840.113711.593.2.106.1
      Value: NULL
      Object identifier 5: 1.2.840.113711.593.2.107.1
      Value: NULL
      Object identifier 6: 1.2.840.113711.593.2.108.1
      Value: NULL
```

```
-------------------------------------------------------------------------
Frame No.   Time        Source         Destination    Protocol
2           0.002178    192.168.1.2    192.168.1.1    SNMP
-------------------------------------------------------------------------
Frame 2 (191 bytes on wire, 191 bytes captured)
Ethernet II, Src: 00:a0:cc:e7:95:3d, Dst: 00:a0:cc:e8:45:0d
Internet Protocol, Src Addr: 192.168.1.2, Dst Addr: 192.168.1.1
User Datagram Protocol, Src Port: snmp (161), Dst Port: 1166
      Version: 1 (0)
      Community: public
      PDU type: RESPONSE (2)
      Request Id: 0x00000000
      Error Status: NO ERROR (0)
      Error Index: 0
      Object identifier 1: 1.2.840.113711.593.2.101.1
      Value: STRING: "ALI"
      Object identifier 2: 1.2.840.113711.593.2.103.1
      Value: INTEGER: 104
      Object identifier 3: 1.2.840.113711.593.2.104.1
      Value: INTEGER: 16384
      Object identifier 4: 1.2.840.113711.593.2.106.1
      Value: INTEGER: 300
      Object identifier 5: 1.2.840.113711.593.2.107.1
      Value: STRING: "Class1"
      Object identifier 6: 1.2.840.113711.593.2.108.1
      Value: STRING: "1"

-------------------------------------------------------------------------
Frame No.   Time        Source         Destination    Protocol
3           0.017355    192.168.1.1    192.168.1.2    SNMP
-------------------------------------------------------------------------
Frame 3 (191 bytes on wire, 191 bytes captured)
Ethernet II, Src: 00:a0:cc:e8:45:0d, Dst: 00:a0:cc:e7:95:3d
Internet Protocol, Src Addr: 192.168.1.1, Dst Addr: 192.168.1.2
User Datagram Protocol, Src Port: 1166, Dst Port: snmp (161)
      Version: 1 (0)
      Community: public
      PDU type: SET (3)
      Request Id: 0x00000001
      Error Status: NO ERROR (0)
      Error Index: 0
      Object identifier 1: 1.2.840.113711.593.2.101.1
      Value: STRING: "McKesson"
      Object identifier 2: 1.2.840.113711.593.2.103.1
      Value: INTEGER: 104
      Object identifier 3: 1.2.840.113711.593.2.104.1
      Value: INTEGER: 16384
      Object identifier 4: 1.2.840.113711.593.2.106.1
      Value: INTEGER: 300
      Object identifier 5: 1.2.840.113711.593.2.107.1
      Value: STRING: "Test"
      Object identifier 6: 1.2.840.113711.593.2.108.1
      Value: STRING: "1"
```

```
-------------------------------------------------------------------------
Frame No.   Time        Source         Destination    Protocol
4           0.019765    192.168.1.2    192.168.1.1    SNMP
-------------------------------------------------------------------------
Frame 4 (191 bytes on wire, 191 bytes captured)
Ethernet II, Src: 00:a0:cc:e7:95:3d, Dst: 00:a0:cc:e8:45:0d
Internet Protocol, Src Addr: 192.168.1.2, Dst Addr: 192.168.1.1
User Datagram Protocol, Src Port: snmp (161), Dst Port: 1166
      Version: 1 (0)
      Community: public
      PDU type: RESPONSE (2)
      Request Id: 0x00000001
      Error Status: NO ERROR (0)
      Error Index: 0
      Object identifier 1: 1.2.840.113711.593.2.101.1
      Value: STRING: "McKesson"
      Object identifier 2: 1.2.840.113711.593.2.103.1
      Value: INTEGER: 104
      Object identifier 3: 1.2.840.113711.593.2.104.1
      Value: INTEGER: 16384
      Object identifier 4: 1.2.840.113711.593.2.106.1
      Value: INTEGER: 300
      Object identifier 5: 1.2.840.113711.593.2.107.1
      Value: STRING: "Test"
      Object identifier 6: 1.2.840.113711.593.2.108.1
      Value: STRING: "1"

-------------------------------------------------------------------------
Frame No.   Time        Source         Destination    Protocol
5           0.021890    192.168.1.1    192.168.1.2    SNMP
-------------------------------------------------------------------------
Frame 5 (84 bytes on wire, 84 bytes captured)
Ethernet II, Src: 00:a0:cc:e8:45:0d, Dst: 00:a0:cc:e7:95:3d
Internet Protocol, Src Addr: 192.168.1.1, Dst Addr: 192.168.1.2
User Datagram Protocol, Src Port: 1166, Dst Port: snmp (161)
      Version: 1 (0)
      Community: public
      PDU type: GET-NEXT (1)
      Request Id: 0x00000002
      Error Status: NO ERROR (0)
      Error Index: 0
      Object identifier 1: 1.2.840.113711.593.1.101
      Value: NULL
```

```
-------------------------------------------------------------------------
Frame No.   Time        Source         Destination    Protocol
6           0.023135    192.168.1.2    192.168.1.1    SNMP
-------------------------------------------------------------------------
Frame 6 (92 bytes on wire, 92 bytes captured)
Ethernet II, Src: 00:a0:cc:e7:95:3d, Dst: 00:a0:cc:e8:45:0d
Internet Protocol, Src Addr: 192.168.1.2, Dst Addr: 192.168.1.1
User Datagram Protocol, Src Port: snmp (161), Dst Port: 1166
      Version: 1 (0)
      Community: public
      PDU type: RESPONSE (2)
      Request Id: 0x00000002
      Error Status: NO ERROR (0)
      Error Index: 0
      Object identifier 1: 1.2.840.113711.593.1.101.1
      Value: STRING: "PII-333"

-------------------------------------------------------------------------
Frame No.   Time        Source         Destination    Protocol
7           0.023421    192.168.1.1    192.168.1.2    SNMP
-------------------------------------------------------------------------
Frame 7 (85 bytes on wire, 85 bytes captured)
Ethernet II, Src: 00:a0:cc:e8:45:0d, Dst: 00:a0:cc:e7:95:3d
Internet Protocol, Src Addr: 192.168.1.1, Dst Addr: 192.168.1.2
User Datagram Protocol, Src Port: 1166, Dst Port: snmp (161)
      Version: 1 (0)
      Community: public
      PDU type: GET-NEXT (1)
      Request Id: 0x00000003
      Error Status: NO ERROR (0)
      Error Index: 0
      Object identifier 1: 1.2.840.113711.593.1.101.1
      Value: NULL

-------------------------------------------------------------------------
Frame No.   Time        Source         Destination    Protocol
8           0.036697    192.168.1.2    192.168.1.1    SNMP
-------------------------------------------------------------------------
Frame 8 (89 bytes on wire, 89 bytes captured)
Ethernet II, Src: 00:a0:cc:e7:95:3d, Dst: 00:a0:cc:e8:45:0d
Internet Protocol, Src Addr: 192.168.1.2, Dst Addr: 192.168.1.1
User Datagram Protocol, Src Port: snmp (161), Dst Port: 1166
      Version: 1 (0)
      Community: public
      PDU type: RESPONSE (2)
      Request Id: 0x00000003
      Error Status: NO ERROR (0)
      Error Index: 0
      Object identifier 1: 1.2.840.113711.593.1.101.2
      Value: STRING: "WOOD"
```

```
-------------------------------------------------------------------------
Frame No.   Time        Source         Destination    Protocol
9           0.036985    192.168.1.1    192.168.1.2    SNMP
-------------------------------------------------------------------------
Frame 9 (85 bytes on wire, 85 bytes captured)
Ethernet II, Src: 00:a0:cc:e8:45:0d, Dst: 00:a0:cc:e7:95:3d
Internet Protocol, Src Addr: 192.168.1.1, Dst Addr: 192.168.1.2
User Datagram Protocol, Src Port: 1166, Dst Port: snmp (161)
      Version: 1 (0)
      Community: public
      PDU type: GET-NEXT (1)
      Request Id: 0x00000004
      Error Status: NO ERROR (0)
      Error Index: 0
      Object identifier 1: 1.2.840.113711.593.1.101.2
      Value: NULL

-------------------------------------------------------------------------
Frame No.   Time        Source         Destination    Protocol
10          0.038175    192.168.1.2    192.168.1.1    SNMP
-------------------------------------------------------------------------
Frame 10 (87 bytes on wire, 87 bytes captured)
Ethernet II, Src: 00:a0:cc:e7:95:3d, Dst: 00:a0:cc:e8:45:0d
Internet Protocol, Src Addr: 192.168.1.2, Dst Addr: 192.168.1.1
User Datagram Protocol, Src Port: snmp (161), Dst Port: 1166
      Version: 1 (0)
      Community: public
      PDU type: RESPONSE (2)
      Request Id: 0x00000004
      Error Status: NO ERROR (0)
      Error Index: 0
      Object identifier 1: 1.2.840.113711.593.1.102.1
      Value: STRING: "MR"

-------------------------------------------------------------------------
Frame No.   Time        Source         Destination    Protocol
11          13.900705   192.168.1.2    192.168.1.1    SNMP
-------------------------------------------------------------------------
Frame 11 (87 bytes on wire, 87 bytes captured)
Ethernet II, Src: 00:a0:cc:e7:95:3d, Dst: 00:a0:cc:e8:45:0d
Internet Protocol, Src Addr: 192.168.1.2, Dst Addr: 192.168.1.1
User Datagram Protocol, Src Port: 1026, Dst Port: snmp-trap (162)
      Version: 1 (0)
      Community: public
      PDU type: TRAP-V1 (4)
      Enterprise: 1.2.840.113711.593.3
      Agent address: 192.168.1.2 (192.168.1.2)
      Trap type: ENTERPRISE SPECIFIC (6)
      Specific trap type: 4
      Timestamp: 7359683
```

Appendix B System Test Log Capture

```
|-----------------------------------------------------------------------
| Trace log opened at 07/17/06 22:04:45.
|-----------------------------------------------------------------------

|-----------------------------------------------------------------------
| [1] Initialization
|-----------------------------------------------------------------------
Trace thread <0xf1> has been named "C:\adscp_snmp.error".
[tracer.cpp:028]    Trace log max size set to 300000 characters.
[tracer.cpp:034]    Trace thread "C:\adscp_snmp.error" has been renamed
                    "C:\adscp_snmp.error".
[tracer.cpp:139]    Trace category "error" has been enabled.
[tracer.cpp:148]    Trace category "warning" has been enabled.
[tracer.cpp:157]    Trace category "Service" has been enabled.
[tracer.cpp:166]    Trace category "Entry" has been enabled.
[tracer.cpp:175]    Trace category "Verbose" has been enabled.
[adscp_snmp.c:137]  Entering SnmpExtensionInit()...
[adscp_snmp.c:762]  Successfully allocated the shared memory object.
[adscp_snmp.c:765]  Successfully mapped the shared memory data.
[adscp_snmp.c:249]  Exiting SnmpExtensionInit()...
[adscp_snmp.c:439]  Trace time of the following message is 07/17/06
                    22:14:01.

|-----------------------------------------------------------------------
| [2] Query for hostnames
|-----------------------------------------------------------------------
[adscp_snmp.c:439]  In SnmpExtensionQuery() with GetNext.
[adscp_mib.c:918]   No existing entry in ProcessReadOnly(), so search
                    now for index 0.
[adscp_mib.c:275]   Smallest Index=1 and Largest Index=3 in the Read-
                    Only table in GetReadOnlySnmpData().
[adscp_mib.c:281]   Requested index is NOT within index range of the
                    Read-Only table in GetReadOnlySnmpData().
[adscp_mib.c:933]   New small entry index in ProcessReadOnly() is 3.
[adscp_mib.c:944]   New large entry index in ProcessReadOnly() is 1.
[adscp_mib.c:1092]  Correct the OID for the GET-NEXT request from
                    1.2.840.113711.593.1.101 to
                    1.2.840.113711.593.1.101.1.
[adscp_mib.c:2219]  In CopyReadOnlyData(),
                    OID=1.2.840.113711.593.1.101.1 returns WOOD.
[adscp_snmp.c:504]  Exiting SnmpExtensionQuery()...
[adscp_snmp.c:439]  In SnmpExtensionQuery() with GetNext.
[adscp_mib.c:918]   No existing entry in ProcessReadOnly(), so search
                    now for index 1.
[adscp_mib.c:275]   Smallest Index=1 and Largest Index=3 in the Read
                    Only table in GetReadOnlySnmpData().
[adscp_mib.c:322]   Closest to the requested index in the Read-Only
                    table in GetReadOnlySnmpData() is 1 and 2.
[adscp_mib.c:933]   New small entry index in ProcessReadOnly() is 1.
[adscp_mib.c:944]   New large entry index in ProcessReadOnly() is 2.
[adscp_mib.c:1092]  Correct the OID for the GET-NEXT request from
                    1.2.840.113711.593.1.101.1 to
                    1.2.840.113711.593.1.101.2.
```

```
[adscp_mib.c:2219] In CopyReadOnlyData(),
                   OID=1.2.840.113711.593.1.101.2 returns GRACE.
[adscp_snmp.c:504] Exiting SnmpExtensionQuery()...
[adscp_snmp.c:439] In SnmpExtensionQuery() with GetNext.
[adscp_mib.c:918]  No existing entry in ProcessReadOnly(), so search
                   now for index 2.
[adscp_mib.c:275]  Smallest Index=1 and Largest Index=3 in the Read-
                   Only table in GetReadOnlySnmpData().
[adscp_mib.c:322]  Closest to the requested index in the Read-Only
                   table in GetReadOnlySnmpData() is 2 and 3.
[adscp_mib.c:933]  New small entry index in ProcessReadOnly() is 2.
[adscp_mib.c:944]  New large entry index in ProcessReadOnly() is 3.
[adscp_mib.c:1092] Correct the OID for the GET-NEXT request from
                   1.2.840.113711.593.1.101.2 to
                   1.2.840.113711.593.1.101.3.
[adscp_mib.c:2219] In CopyReadOnlyData(),
                   OID=1.2.840.113711.593.1.101.3 returns PII-333.
[adscp_snmp.c:504] Exiting SnmpExtensionQuery()...
[adscp_snmp.c:439] In SnmpExtensionQuery() with GetNext.
[adscp_mib.c:918]  No existing entry in ProcessReadOnly(), so search
                   now for index 3.
[adscp_mib.c:275]  Smallest Index=1 and Largest Index=3 in the Read-
                   Only table in GetReadOnlySnmpData().
[adscp_mib.c:281]  Requested index is NOT within index range of the
                   Read-Only table in GetReadOnlySnmpData().
[adscp_mib.c:933]  New small entry index in ProcessReadOnly() is 3.
[adscp_mib.c:944]  New large entry index in ProcessReadOnly() is 1.
[adscp_mib.c:1092] Correct the OID for the GET-NEXT request from
                   1.2.840.113711.593.1.101.3 to
                   1.2.840.113711.593.1.102.1.
[adscp_mib.c:2219] In CopyReadOnlyData(),
                   OID=1.2.840.113711.593.1.102.1 returns CR,CT,MR,US.
[adscp_snmp.c:504] Exiting SnmpExtensionQuery()...

|-------------------------------------------------------------------
| [3] Query for host's statistics
|-------------------------------------------------------------------
[adscp_snmp.c:431] In SnmpExtensionQuery() with Get.
[adscp_mib.c:764]  GET the read-only interval in ProcessReadOnly() with
                   the requested OID (1.2.840.113711.593.1.1).
[adscp_snmp.c:504] Exiting SnmpExtensionQuery()...
[adscp_snmp.c:431] In SnmpExtensionQuery() with Get.
[adscp_mib.c:793]  No existing data or no matching in
                   ProcessReadOnly(), so get data again.
[adscp_mib.c:275]  Smallest Index=1 and Largest Index=3 in the Read-
                   Only table in GetReadOnlySnmpData().
[adscp_mib.c:322]  Closest to the requested index in the Read-Only
                   table in GetReadOnlySnmpData() is 1 and 2.
[adscp_mib.c:815]  Successfully obtain an entry (Index=1) from the
                   read-only table in ProcessReadOnly().
[adscp_mib.c:2219] In CopyReadOnlyData(),
                   OID=1.2.840.113711.593.1.102.1 returns CR,CT,MR,US.
[adscp_mib.c:815]  Successfully obtain an entry (Index=1) from the
                   read-only table in ProcessReadOnly().
[adscp_mib.c:2206] In CopyReadOnlyData(),
                   OID=1.2.840.113711.593.1.103.1 returns 4.
```

```
[adscp_mib.c:815]  Successfully obtain an entry (Index=1) from the
                   read-only table in ProcessReadOnly().
[adscp_mib.c:2206] In CopyReadOnlyData(),
                   OID=1.2.840.113711.593.1.104.1 returns 165.
[adscp_mib.c:815]  Successfully obtain an entry (Index=1) from the
                   read-only table in ProcessReadOnly().
[adscp_mib.c:2206] In CopyReadOnlyData(),
                   OID=1.2.840.113711.593.1.105.1 returns 447.
[adscp_mib.c:815]  Successfully obtain an entry (Index=1) from the
                   read-only table in ProcessReadOnly().
[adscp_mib.c:2206] In CopyReadOnlyData(),
                   OID=1.2.840.113711.593.1.106.1 returns 76143424.
[adscp_mib.c:815]  Successfully obtain an entry (Index=1) from the
                   read-only table in ProcessReadOnly().
[adscp_mib.c:2211] In CopyReadOnlyData(),
                   OID=1.2.840.113711.593.1.107.1 returns 421.
[adscp_mib.c:815]  Successfully obtain an entry (Index=1) from the
                   read-only table in ProcessReadOnly().
[adscp_mib.c:2211] In CopyReadOnlyData(),
                   OID=1.2.840.113711.593.1.108.1 returns 11.
[adscp_mib.c:815]  Successfully obtain an entry (Index=1) from the
                   read-only table in ProcessReadOnly().
[adscp_mib.c:2211] In CopyReadOnlyData(),
                   OID=1.2.840.113711.593.1.109.1 returns 4.
[adscp_snmp.c:504] Exiting SnmpExtensionQuery()...

|-------------------------------------------------------------------
| [4] Error Notification
|-------------------------------------------------------------------
[adscp_snmp.c:348] SnmpExtensionTrap() detects the trap value 4.
[adscp_snmp.c:357] SnmpExtensionTrap() found no more trap.

|-------------------------------------------------------------------
| Trace log closed at 07/17/06 22:17:49.
|-------------------------------------------------------------------
```

Appendix C Project Contribution

This project is a joint development effort of Edlic Yiu and Edwood Yiu. The individual contributions to this project are outlined in Figure 38.

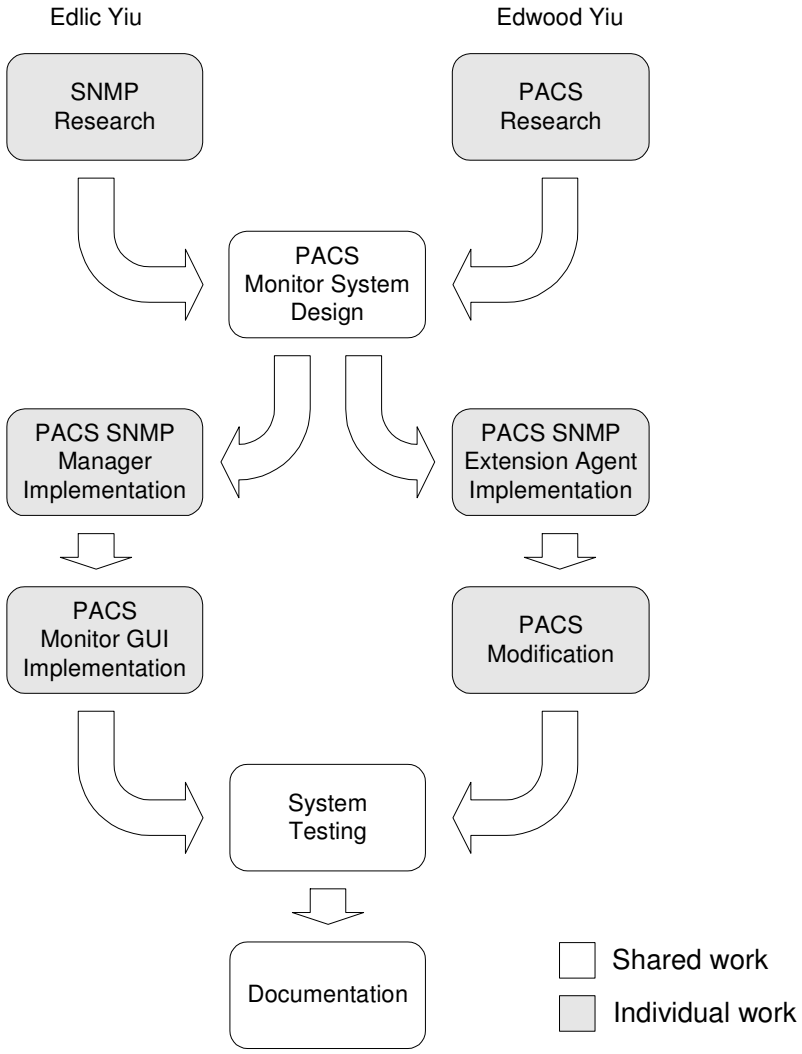

Figure 38. Individual contribution to this project.

In this project, Edlic Yiu focused on the SNMP research. During the development phase, he was responsible for the design and implementation of the PACS monitor system that consists of the PACS Monitor GUI and the PACS SNMP manager.

Edwood Yiu added SNMP support to the PACS system. He modified the PACS system to maintain our newly defined management data. He also implemented the PACS SNMP extension agent for the communication between the Windows operating system and the PACS system.

Reference List

[1] AdventNet [Online]. Available: http://www.adventnet.com/products/snmputilities/help/quick_tour/ snmp_and_mib/snmpmib_snmpoverview.html.

[2] F. H. B. Binkhuysen, "Impact of PACS on radiologists' daily work in western countries," *IEEE J. Select. Areas Commun*., vol. 10, no. 7, pp. 1158 - 1160, Sept. 1992.

[3] J. D. Case, M. Fedor, M. L. Schoffstall, and J. Davin, "Simple network management protocol (SNMP)," *IETF RFC 1157*, May 1993.

[4] J. D. Case, K. McCloghrie, M. Rose, and S. Waldbusser, "Introduction to community-based SNMPv2," *IETF RFC 1901*, Jan. 1996.

[5] J. D. Case, K. McCloghrie, M. Rose, and S. Waldbusser , "Protocol operations for version 2 of the simple network management protocol (SNMPv2)," *IETF RFC 1905*, Jan. 1996.

[6] J. D. Case, K. McCloghrie, M. Rose, and S. Waldbusser , "Coexistence between version 1 and version 2 of the Internet-standard network management framework," *IETF RFC 1908*, Jan. 1996.

[7] V. Cerf, "IAB recommendations for the development of Internet network management standards," *IETF RFC 1052*, Apr. 1988.

[8] Cisco System [Online]. Available: http://www.sec.carleton.ca/netmanage/snmp/cisco-intro.html.

[9] DICOM Standard Committee, "DICOM Part 3: Information Object Definitions," NEMA/The DICOM Standard, 2006.

[10] DICOM Standard Committee, "DICOM Part 5: Data Structures and Encoding," NEMA/The DICOM Standard, 2006.

[11] DICOM Standard Committee, "DICOM Part 7: Message Exchange," NEMA/The DICOM Standard, 2006.

[12] DICOM Standard Committee, "DICOM Part 8: Network Communication Support for Message Exchange," NEMA/The DICOM Standard, 2006.

[13] Electronic and Telecommunication Institute [Online]. Available: http://www.et.put.poznan.pl/snmp/main/mainmenu.html.

[14] J. Feng, R. Han, C. Wang, M. Wang, D. Wu, G. Zhang, J. Zhang, X. Zhang, and J. Zhuang, "Managed PACS operation with an automatic monitoring tool," *in Proc. SPIE*, vol. 4685, pp. 326 - 332, May 2002.

[15] IHE [Online]. Available: http://www.ihe.net/About/ihe_faq.cfm.

[16] IP Monitor Support Portal [Online]. Available: http://support.ipmonitor.com/ tutorials/ 684f20eeb613444dae5518d6bdf6c766.aspx.

[17] K. McCloghrie, R. Presuhn, and B. Wijnen, "View-based access control model (VACM) for the simple network management protocol (SNMP)," *IETF RFC 2575*, Apr. 1999.

[18] McKesson [Online]. Available: http://www.mckessonautomation.com/wt/auto/ nurse_index.

[19] Medinous [Online]. Available: http://www.medinous.com/hisindex.htm.

[20] R. McHenry, "Diagnostic Imaging," *Encyclopedia Britannica*, Chicago, USA, 1992, vol. 4, pp. 62 – 63.

[21] R. McHenry, "Medicine," *Encyclopedia Britannica*, Chicago, USA, 1992, vol. 24, pp. 774 – 828.

[22] R. McHenry, "Nuclear Magnetic Resonance," *Encyclopedia Britannica*, Chicago, USA, 1992, vol. 8, p. 819.

[23] R. McHenry, "Nuclear Medicine," *Encyclopedia Britannica*, Chicago, USA, 1992, vol. 8, p. 819.

[24] W. Stallings, *SNMP, SNMPv2, and RMON: Practical Network Management*. New York: Addison Wesley, 1996.

[25] D. Zeltserman, *A Practical Guide to SNMPv3 and Network Management*. New Jersey: Prentice-Hall, 1999.