
Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática
2011

LUÍS ANTÓNIO BASTIÃO SILVA

SERVIÇOS DE IMAGEM MEDICA SUPORTADOS NA CLOUD

MEDICAL IMAGING SERVICES SUPPORTED ON CLOUD


Universidade de Aveiro Departamento de Electrónica, Telecomunicações e
2011 Informática

LUÍS ANTÓNIO BASTIÃO SILVA

SERVIÇOS DE IMAGEM MEDICA SUPORTADOS NA CLOUD

MEDICAL IMAGING SERVICES SUPPORTED ON CLOUD

dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Dr. Carlos Manuel Azevedo Costa, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

o júri

presidente

Prof. Dr. Joaquim Arnaldo Carvalho Martins
Professor Catedrático do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Prof. Dr. Carlos Manuel Azevedo Costa
Professor auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Prof. Dr. Rui Pedro Sanches de Castro Lopes
Professor Coordenador do Departamento de Informática e Comunicações do Instituto Politécnico de Bragança

agradecimentos

Agredeço à minha família, pelo apoio incondicional.
Agredeço aos meus amigos por estarem sempre presentes.
Agradecimento especial aos membros do grupo de bioinformática nomeadamente Prof. Carlos Costa e Prof. José Luís Oliveira.

acknowledgments

I thank to my family for the unconditional support.
I thank to my friends for being present in my life.
Special thanks to bioinformatics group, namely Prof. Carlos Costa and Prof. José Luis Oliveira.

palavras-chave

Imagem Médica, PACS; DICOM; Telemedicina; Cloud Computing.

resumo

Hoje em dia, as instituições de cuidados de saúde, utilizam a telemedicina para suportar ambientes colaborativos. Na área da imagem médica digital, a quantidade de dados tem crescido substancialmente nos últimos anos, requerendo mais infraestruturas para fornecer um serviço com a qualidade desejada. Os computadores e dispositivos com acesso à Internet estão acessíveis em qualquer altura e em qualquer lugar, criando oportunidades para partilhar e utilizar recursos online. Uma enorme quantidade de processamento computacional e armazenamento são utilizados como uma comodidade no quotidiano. Esta dissertação apresenta uma plataforma para suportar serviços de telemedicina sobre a cloud, permitindo que aplicações armazenem e comuniquem facilmente, utilizando qualquer fornecedor de cloud. Deste modo, os programadores não necessitam de se preocupar onde os recursos vão ser instalados a as suas aplicações não ficam limitadas a um único fornecedor. Foram desenvolvidas duas aplicações para tele-imagiologia com esta plataforma: repositório de imagens médicas e uma infraestrutura de comunicações entre centros hospitalares. Finalmente, a arquitetura desenvolvida é genérica e flexível permitindo facilmente a sua expansão para outras áreas aplicacionais e outros serviços de cloud.

keywords Medical Imaging, PACS; DICOM; Telemedicine; Cloud Computing.

abstract Healthcare institutions resort largely, nowadays, to telemedicine in order to support collaborative environments. In the medical imaging area, the huge amount of medical volume data has increased over the past few years, requiring high-performance infrastructures to provide services with required quality. Computing devices and Internet access are now available anywhere and at anytime, creating new opportunities to share and use online resources. A tremendous amount of ubiquitous computational power and an unprecedented number of Internet resources and services are used every day as a normal commodity. This thesis presents a telemedicine service platform over the Cloud that allows applications to store information and to communicate easier, using any Internet cloud provider. With this platform, developers do not concern where the resources will be deployed and the applications will not be restricted to a specific cloud vendor. Two tele-imagiologic applications were developed along with this platform: a medical imaging repository and an inter-institutional communications infrastructure. Lastly, the architecture developed is generic and flexible to expand to other application areas and cloud services.

Table of Contents

List of figures

List of Tables

Acronyms

Term	Description
AETitle	Application Entity Title
API	Application Programming Interface
CT	Computed tomography
DBMS	Database Management Systems
DBR	DICOM Bridge Router
DCR	DICOM Cloud Router
DICOM	Digital Image and Communication in Medicine
DIMSE	DICOM Message Service Element
GAE	Google AppEngine
GWT	Google Web Toolkit
HIS	Hospital Information System
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure-as-a-Service
IOD	Information Object Definition
IT	Information Technology
MR	Magnetic Resonance
MRI	Magnetic Resonance Imaging
PaaS	Platform-as-a-Service
PACS	Picture Archiving and Communication System
PET	Positron Emission Tomography
RDBMS	Relational Database Management System
RIS	Radiology Information System
SaaS	Software-as-a-Service
SCP	Service Class Provider
SCU	Service Class User
SDCP	Service Delivery – Cloud Platform
SDK	Software Development Toolkit
SOP	Service Object Pair
SR	Structured Report
TLV	Tag-Length-Value
UID	Unique Identifier
US	Ultrasound
VPN	Virtual Private Network
VR	Value Representation
WADO	Web Access to DICOM Persistent Objects
XA	X-ray Angiography
XML	Extensible Markup Language

1 Introduction

"Technological progress is like an axe
in the hands of a pathological criminal."
Albert Einstein

This chapter contains an introduction to this thesis, providing an overview and outlining the main goals of the project. It also provides a quick explanation of what each section stands for.

1.1 Overview

Progress of the Internet and Information Technologies (IT) has created tremendous opportunities for society to develop new products and services, communicate and share data. These new facilities allow the exchange of information anytime and anywhere, at high speed. Over the last decades many sectors have adopted computing solutions, for instance, the armed forces, governments, banks, bus companies, healthcare market and many others.

In the healthcare sector there are many areas where informatics and telematics are very important. New technologies have promoted excellence, efficiency, professionalism and safety in all those sectors. Administrative services, patient bedside monitoring, diagnostic imaging and remote diagnostics are some examples that have beneficiated from the information systems and IT in general.

The importance of medical imaging in healthcare is unquestionable. Clearly, the evolution of computers created new opportunities in medical informatics. As a consequence, synergies have been created between IT industries and healthcare researchers to enhance methods and workflows in healthcare.

Use of digital medical imaging systems has greatly increased in healthcare institutions. Today, it is one of most valuable tools supporting medical decision and treatment procedures. Storing and retrieving images within a PACS (Picture Archiving and Communication System) presents significant advantages over traditional analogical systems based on film. PACS allow an overall boost of productivity, decreased operational costs and also create an excellent opportunity for telemedicine, telework and collaborative work environments. The production of medical digital imaging has increased in the different therapeutic agents and nowadays it is very important to support medical decisions. Moreover, in order to improve diagnosis, there is a tendency to increase the resolution, slice number and some particular properties of medical imaging, and as a consequence the volume of medical studies increases too. Although, digital medical imaging brought many benefits, it also presents new challenges for storing, indexing and sharing data.

Medical images are stored in an image archive in the PACS. Besides the image archive, study information is also contained in a database, i.e., patient information, study information and reference to studies. To support all data in image archive, healthcare institutions maintain datacentres with large gigabytes of information, or even petabytes. Nonetheless, resources and

funds are limited. Therefore, with significant increases in the amount of studies completed, the outsourcing solution has been considered by some institutions.

Indeed, medical imaging has undergone noteworthy changes in medical devices, such as medical image acquisition and image viewers. So, in order to support the exchange of medical images in digital format, a standard was created: DICOM (Digital Communication in Medicine). The standards presented a protocol to share medical imaging between different vendors, since nowadays most diagnostic imaging devices support DICOM. Currently, equipment in medical institutes follows the DICOM standard to communicate, store and visualize information. PACS has a huge amount of IT infrastructure and all devices permit communication via DICOM.

In theory, DICOM standard solved all issues, but some gaps remain in real environments. These medical images contain patient's information, and there are several concerns regarding their privacy. This means that medical institutions safeguard patient's records. Therefore, this is an issue for medical inter-institutional cooperation, in a paradigm of "many-to-many" collaboration.

1.2 Goals

The emergence of Cloud computing providers creates a great opportunity to tackle the costs of purchasing hardware and software. This computing technology uses the Internet and distributed servers to preserve data and applications, and it allows the creation of IT facilities without investing in infrastructure, training personnel or licensing software. Cloud computing providers supply elastic computing power and storage. It is an economical solution following an on-demand pay-per-use scalability.

This paper aims to solve many problems in the storage and retrieval of medical images, through co-operating between multi-site medical institutions. The market is changing and there are new paradigms for deploying applications and storing information. Medical solutions should adopt these new models to improve their business processes. Following the technological evolution, outsourcing to the cloud has been adopted by several companies, and in particular the healthcare industry. For example, Google Health [1] and Microsoft Healthvault [2] provide a management panel which is easy to access for personal health information always available.

The driving idea is the outsourcing of PACS components to public Internet Cloud providers, using current cloud technologies and a computing-as-utility paradigm. With this approach we can take advantage of the elasticity and scalability of clouds, providing universal access to information, increasing data availability and avoiding hardware obsolescence. However, in medical institutions there are a great number of devices that cannot communicate with Cloud computing interfaces. This thesis aims to study and develop a PACS Archive over the Cloud technology that grants multi-institutional access. This paper proposes a self-organized PACS archive in a "PACS-as-a-service" paradigm, in the Cloud Infra-structure contributing to the "Share Medical Image, anytime, anywhere" concept.

The solution is fully compatible with DICOM and presents an architecture that is independent of the cloud provider. It supports two major DICOM services – Storage and Query/Retrieve – which allow data transfer operations to and from the repository. This platform has to be compatible with the standard, and will thus enable effortless integration for the institutions using the solution.

Also, creating a shared repository between multi-institutions allows an easier and quicker way of communicating between them. In such cases, the access control list is very important to prevent unauthorized access to medical information. There are several aspects regarding ethical and legal issues that are a key point in the medical imaging. Privacy and confidentiality is problematic

when the outsourcing method is used. Moreover, there are several countries' laws require to know where the data is stored. So real deployment of this kind of solution depends on the country where it is going to be implemented. Our architecture provides strong security features that ensure the privacy of three main entities: medical institutions, physicians and patients.

Identified features should be integrated into a platform along with other specified requirements of a new model: "PACS-as-a-Service". Hence, the engineering of an application to incorporate all the features and test its performance is the major goal of this thesis.

1.3 Outline of thesis

The thesis is organized in 8 chapters. Here is a brief description of what this document contains:

Chapter 2: explains the medical scenario that this thesis is inserted. A brief description of digital medical imaging laboratories is presented. The strategies for storage and distribution of medical imaging are also described. The DICOM standard is described a focusing on the format and service communication. Finally, we present the cloud computing technology, describing the business model and which services are available.

Chapter 3: presents the problem and explores the requirements of this project. We also present a case study of the financial availability of the solution. The approach to solve the problem is briefly described.

Chapter 4: describes a platform to access the cloud providers without any lock to suppliers. The architecture and the engineering processes are presented.

Chapter 5: presents the architecture of the PACS archive over the cloud. The implementation details are described and the workflows between the various elements of the process are presented. The important aspects regarding security are highlighted. Finally, we present some improvements that we added to increase the performance of the solution.

Chapter 6: shows an architecture to forward DICOM messages in a multicentre scenario. This relay service uses the cloud resources to forward messages. The interoperability with other medical devices is safeguarded, and this is explained in the section.

Chapter 7: presents the results and validates the solutions. Tests of performance are described and the time measures are presented and compared.

Chapter 8: gives the key point of the developed work, highlighting the main contributions and points out further work.

2 State of the Art

"When we try to pick out anything by itself, we find it is tied to everything else in the universe."
John Muir

This chapter deals with the application scenario and provides an overview of the real environment of this thesis. Current technologies are also analysed and important details highlighted.

2.1 Digital Laboratories in Medical Environment

Over the past two decades, the healthcare sector has been adopting technologies and information systems to support diagnoses, treatment and patient care. Medical imaging is no an exception because it is a segment that produces a huge amount of information and take advantage of new technologies to aid in the imaging diagnosis.

Nowadays, the use of medical imaging systems in healthcare institutions is unquestionable, even in small medical imaging centres. Thus, it is necessary to endow these medical institutions with digital storage and visualization devices to improve the physician's workflow. However, acceptance of new technologies in the healthcare sector was a challenge because clinical staff present barriers against the digital era. The conventional paper and film-based operation was part of the traditional workflow and changing to digital images was pointed out as "losing control" as doctors must ensure that procedures are under their control. Nevertheless, changes in IT caused an on-going revolution in radiology and, in general, all medicine. As a result, workflows were speeded up and costs were reduced, with a significant financial impact on healthcare institutions.

Many technologies developed for the healthcare sector have contributed to improving diagnosis and support for clinical decisions. One of the important changes was in management of digital radiology modalities, e.g. X-Rays, Computed Tomography (CT), Magnetic Resonance (MR), Ultrasounds (US) and so on. Healthcare institutions have purchased more acquisition devices, with higher resolution and larger image data size. The amount of images produced by acquisition devices has increased and the medical image repository has to scale up in order to support such a growing volume of medical studies.

2.1.1 PACS (Picture Archive and Communication System)

During the last two decades, health centres have made significant investment in IT to create and maintain medical imaging laboratories. In these labs, data storage is always a key issue. The volume of data generated, for instance, by dynamic cardiac modalities, e.g. X-Ray Angiography (XA) and Ultrasound (US), multi-slice Computer Tomography (CT), high-field Magnetic Resonance Imaging (MRI) and digital mammography is tremendous. Efficient storage and permanent availability of all produced data are huge tasks [3, 4], at least without requiring major upgrades and overhauls that significantly increase the total cost of ownership over time [4].

Moreover, institutions must deal with several problems related to IT infrastructure including solution scalability, fault tolerance, performance issues, hardware maintenance costs, system obsolescence and migration. A redundancy and disaster/catastrophe plan is also very important to protect electronic medical records. Patient data security is essential considering that hospitals hold a huge quantity of data.

PACS encompasses several hardware, communication networks and software technologies for the acquisition, distribution, storage and analysis of digital images in distributed environments [5]. The main components are: image acquisition and scanning devices, storage archive units, display workstations and databases with patient records. All those components communicate throught the network, contributing to the integrated system. PACS with Radiologic Information System (RIS) and Hospital Information System (HIS) are a filmless radiology service. Filmless radiology refers to a hospital, where at least most films have been replaced by electronic systems, which acquire, store, distribute and visualize medical digital images. A description of the most important steps in PACS follows:

- **Acquisition**: This is the process of image capture and digital codification. This method creates an image quite similar to reality, which will help physicians in diagnosis. In modalities, there are two ways to produce medical images. The first one is the conventional method, with film scanned by laser. However this method has a high risk, and still has troubles associated with the scanner. The other technique is where the digital medical image is produced directly by imaging plate. Those images can be visualized in the workstations.
- **Distribution**: This refers to the ability to move images and associated data from a location to outside the sector. In PACS, distribution is needed to transfer images from acquisition, storage, or image archive to the workstation. Moreover, the PACS distribution process can reduce costs because it avoids the loss of studies, and it becomes easier to exchange studies with other institutions, for clinical, research or academic proposes.
- **Visualization**: For most users, workstations (i.e. visualization systems) are the only element of PACS, due to their visibility. The workstation allows users to search, retrieve, visualize, manipulate and send medical images to/from the repository.

PACS is a digital image system which allows specialists, clinics, physicians and image, centres to quickly access patient's medical records. PACS tends to focus on broken down islands of dispersed data providing a consolidated platform, so information becomes more readily available to doctors, at any time of day or night. Moreover, images can be distributed in multiple departments at the same time.

There are several stages in the PACS workflow, from the patient being registered in the HIS, RIS ordering, performing an examination, image viewing, and reporting and to archiving. The workflows are different, and vary for each hospital. H.K. Huang [5] proposed 3 main solutions that fit in different kinds of workflows: stand-alone, client-server and web-based model:

- **Stand-alone:** This approach has a central repository and when the studies are stored in the central archive, they are sent automatically to the workstations for diagnosis and review, based on a store-and-forward approach. The workstations are also able to do Query/Retrieve studies from the archive server. However, the PACS Archive server automatically pre-fetches studies, i.e., sends the pertinent history to the workstations. This solution has several benefits because the modality can send images directly to workstations. Thus, there is less risk to lose the studies. It is also controversial because

images can be sent to more than one workstation, and consequently more than one radiologist can review the same exam.

- **Client-server:** In this approach, there is only one central repository, unlike the stand-alone model. The images are sent directly to the PACS Archive, and the users use the worklists to download the exams just selecting the required patient. The workstations do not have local storage, and the images are automatically discarded after the review. In this case, the reviewer has access to any exams in the PACS Archive, without using Query/Retrieve or prefetching. Nevertheless, the PACS Archive is a single-point-of-failure and the bandwidth in the local network has to be very fast. It will also have trouble in tele-imaging because the workstation needs to download the studies in real-time, and in those cases prefetching is a desirable solution.
- **Web-based**: This solution follows the web evolution. Thus, the review process is done in a web application that lives in the same datacentre as the PACS Archive and the user just needs an Internet Connection. It is a very good solution, because the workstation is not locked to any platform, and can be accessed everywhere. Nonetheless, it is problematic because the application is limited to the web browser features. This model has grown in the past 3 years and is now leading in clinical workflow.

Although these models can be suitable for the hospital's workflow, a PACS repository is required in all architectures. Thus, the PACS Archive server has many challenges to solve; one of them is how to support the enormous amount of data. All patient information has to be available, but some questions arise. For instance, do studies over 5 years old have to be accessible? There are many archive strategies, so three different approaches was adopted in the PACS Archive: (i) online, (ii) nearline and (iii) offline.

The online archive (i) meaning Direct Access Storage Device (DASD) is instantly available to PACS applications and supports the more recent studies. For instance, an average of 1200 exams live for 6 weeks in the online archive, as referred to in [6]. In a traditional PACS they use magnetic hard drives, for instance, a RAID-SATA. The time to access has to be in milliseconds and the transfer rates to get exams should be in the seconds scale (~8 to access an exam with 300Mbytes) [6]. The online archive is very costly to maintain because it is highly efficient regarding the data available, and reads/writes access very fast.

The nearline archive (ii) will store the archive that is out of the online required period. An exam stored several months previously is unlikely to be needed and the time for access will be less important. This archive has lower costs than the online archive and typically stores about 2/3 years of archive. Although it is unlikely to be accessed, clinical staff can consult it. Nearline typically uses the jukeboxes because it is cheaper than magnetic hard-drives, but it requires robotic arms to access the tapes. Generally, it can access to the information in seconds and transfer data, for instance ~1m to 300Mbytes exam.

The offline archive (iii) is a manual archive that it is not much used nowadays. This archive requires human resources to get studies in the physical archive. These different archive methods are very important to institutions because they categorize different studies to different archives depending on the probability of access. Thus, they save costs on the old archives.

Finally, several studies show that currently digital medical image data can be generated in practically any healthcare institution, even one with limited human or financial resources [7] . Medical imaging equipment, e.g. image data generators, became gradually more powerful and less expensive, resulting in their proliferation in small imaging centres. However, some of these centres

do not have resources to acquire and maintain a traditional PACS solution with its IT infrastructure. This led hospitals and imaging centres to consider the outsourcing of PACS IT infrastructure to a remote datacentre. It is a service model that eliminates, or at least reduces, many of problems previously identified and maximizes the investment revenue. Another key benefit is that those repositories are vendor-neutral, so aging and PACS upgrading is not an issue. Moreover, this new paradigm maximizes PACS efficiency because the solution can serve multiple sites, facilitating inter-institutional data share and common workflows [8].

2.1.2 DICOM Standard

The use of technologies and information systems in the medical imaging area started with the manufacture of equipment able to acquire, store and transfer data between medical stations. However, communication between all these devices did not follow a standard and many manufacturers developed their own communication protocols increasing the difficult in accessing from different vendor devices. In the 90s, a consortium was created formed of NEMA (National Electrical Manufactures' Association) and ACR (American College of Radiology) aiming to normalize formats and communication processes in the medical environment. This group developed a set of standardizations and guidelines, which allows communication and transfer of medical imaging data between different vendors' devices. This movement had a strong impact on the development and expansion of PACS.

At the beginning of the 90s, this consortium released a standard named DICOM (Digital Image and Communication in Medicine). In the 3.0 version, it is split in 18 parts, containing 155 work supplements about specific issues of one or various parts of the standard. DICOM is an international standard that defines the file format and directory structure needed for offline communication. In addition, the communication protocols and packet semantics are described to contribute to operation between medical imaging devices.

DICOM standard not only grants basic connectivity between imaging devices, but also supplies guidelines for workflow in an imaging department. Nowadays, it is a major contributor to the exchange of structured medical imaging data and almost all medical imaging manufacturers are following the DICOM rules.

DICOM Information Model - DIM

The real world has direct representation in DICOM standard. DICOM Information Model (DIM) [9] represents the items that match real objects and describes their relationship. These relations are quite important to organize the information systems. DICOM has the following hierarchy: Patient-Study-Series-Image (Figure 2.1). The hierarchy reproduces real life entities, i.e. a patient can have multiple examinations, and these examinations may be from different modalities. Each modality uses different protocols and may produce a different number of images.

A unique key is assigned to each level of hierarchy. This is very important to identify the Patient, Study, Series and Image. PatientID identifies the patient level. At the study level, Study Instance UID is used. Series Instance UID is the unique id at the series level. Finally, each DICOM object is instanced with a SOP (Service Object Pair) Instance UID – image level.

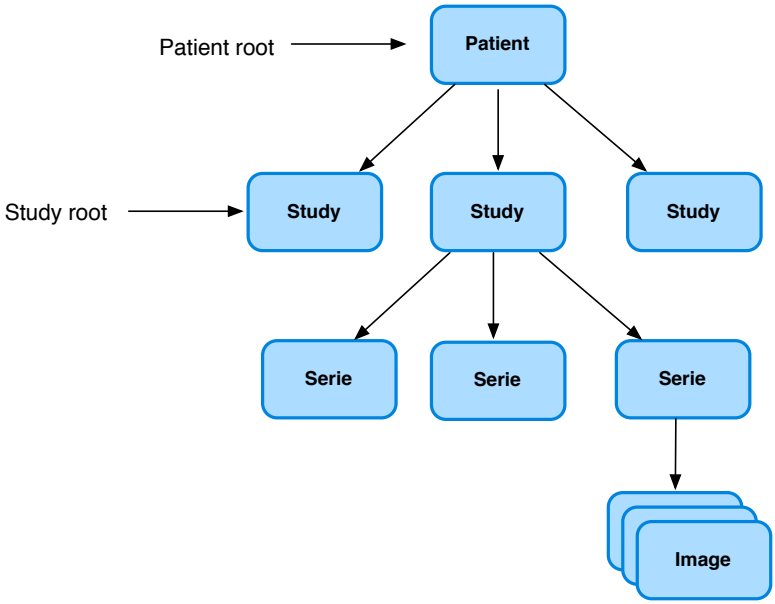


Figure 2.1: DICOM Information Hierarchy – one patient may have multiple studies; each study may include one or more series; each series has one or more images

DICOM File Format

DICOM supports different kinds of information, including different modalities of image (CT, XA, US and many others), reports and waveforms. Besides the image, the DICOM file contains the metadata header with information related to patient, clinical staff, medical institution, acquisition device, conditions of exam, clinical protocol and much other relevant clinical information.

A DICOM file contains metadata header and other kinds of content (e.g. image, waveform), as shown in Figure 2.2. The header contains a varying number of fields according to the modality of the study. Several fields are identified in the DICOM Information Model [9] which outlines fields that are mandatory in every DICOM file.

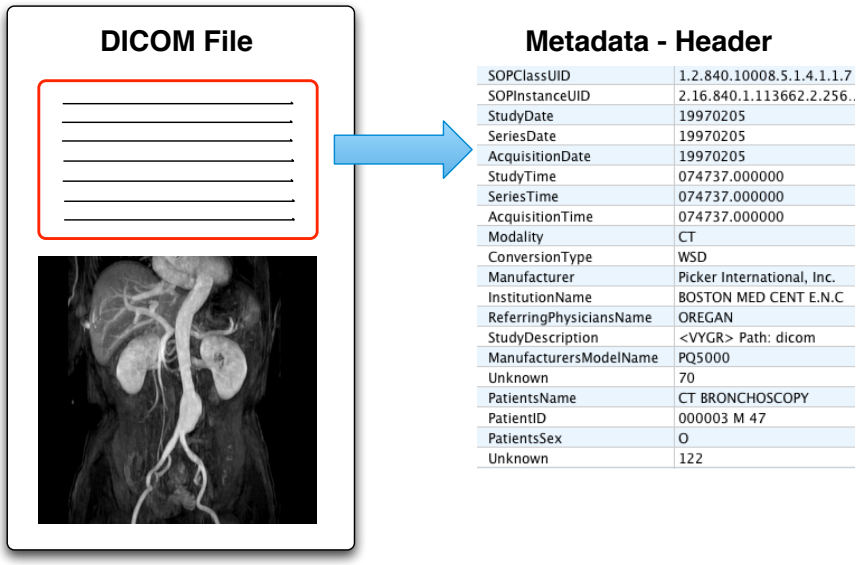


Figure 2.2: DICOM object with metadata and image. The header is zoom in (right hand side) and it illustrates a few of the attributes that the object contains

The DICOM objects are organized in a TLV format (Tag, Length, Value) as shown in Figure 2.3. The tag is a pair of two values, a 16-bits unsigned integer, representing the group and element number. For each element there are several attributes, i.e. information. Each tag is unique, and in the same file, there are no duplicated tags. A hexa-decimal number between 0x0000 and 0xFFFF represents the tags. For instance, the Study Date is in group 0x8 and subgroup 0x20. Thus, the Study Date tag is (0008,0020).

There is another field named Value Representation (VR) that covers how attributes are encoded. It contains the code that describes the type of element, e.g. PN means Person Name, DA means Date and OB means Object Binary. This label is optional because the type of element can be reached using a DICOM Dictionary, i.e. for each tag the dictionary defines what type it stands for. Length of attribute may vary for each tag, so the field length defines the size of each attribute (value field). This size uses byte scale. Finally, the value field contains the element of the tag that is used to store attribute contents (e.g. image pixel data, patient name, etc).

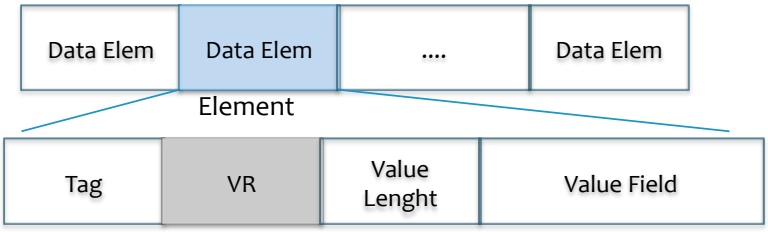


Figure 2.3: DICOM format - Tag-Length-Value

DICOM files and protocol follow the TLV sequences. Each file is considered a DICOM object, i.e. by definition a DICOM object is a set of data elements, as described in Figure 2.3. It means that new tags can be defined in DICOM objects, increasing the flexibility of file/communication configuration. Thus, its structure is dynamic and the DICOM parsers can be generic, decoding any kind of DICOM file.

To encode or decode a DICOM object it is necessary to know the Information Object Definition (IOD). IOD is very similar to the templates approach, which means that each DICOM object is carried out by specific IODs in a high level definition. The idea of the IOD is to represent the most common data types in medical digital imaging. IODs are hierarchy and one single IOD may be a sub set of other IODs. For instance, an image of modality follows its own IOD, e.g. CT IOD contains few IODS like Patient Information Object Definition, although the other modalities also use this IOD. A SOP Instance is an instantiation of an IOD, which means that it has real attributes representing the definition, i.e. IOD.

DICOM Media Storage - DICOMDIR

DICOM standard allows exams to be saved in a media device, for instance a CD, DVD and pen drive. These media storage devices are very important to exchange data between institutions. These media devices can be imported to the central archive, displayed and printed.

Although the DICOM file contains the information about the patient, study and series, access to this information in real time is not easy because all files have to be parsed. In fact, it is not a desirable solution due to lack of performance. So, the DICOM committee created an indexing file structure called DICOMDIR. This file catalogues the most important fields that are necessary to use in searching the query/retrieve process. Moreover, it uses a hierarchical tree data structure that permits fast queries and DIM elements to be displayed.

DICOM Services

DICOM protocol is quite important for interoperability between medical devices. This protocol works over TCP/IP granting a reliable connection between actors. It follows the same structure of DICOM objects and, in the network, encapsulates messages as TLV elements. There are many concepts involved in the protocol, but we will address just the service layer, i.e. exchanged messages.

Over the protocol layer, DICOM has several services that follow a client/server architecture. In the DICOM scenario there is Service Class Provider (SCP) and Service Client User (SCU). A DICOM equipment can have different roles, i.e. SCP or SCU, depending on the type of device. For instance a modality that produces images is responsible for storing the image in the PACS Archive. Thus, a modality is considered a SCU because it uses a service. On the other hand, PACS Archive is offering a service, therefore it belongs to SCP.

Each DICOM device has an Application Entity Title (AETitle) that identifies the DICOM device. DICOM is a protocol over TCP/IP, thus AETitle is an addressing mechanism that works similarly to IP and port in the lower layer of the network stack.

To communicate with a DICOM device, the first step is to purpose an exchange of information, called DICOM association. In this procedure, devices negotiate several parameters for the association, such as, what kind of information will be transferred, how it is encoded and the duration of the association. After the negotiation, the service commands are executed between SCU and SCP to perform the service goal.

Each service is associated with one or more commands (Table 2.1). The most important commands of DICOM are shown in Table 2.1: Verification, Storage, Query/Retrieve and Worklist management.

The verification service allows the SCU to check end-to-end communication, i.e. verify if the SCP is working properly through the C-ECHO command.

Table 2.1: Most common DICOM Service: it includes several DICOM services, description and which command is associated with the service

Service	Description	DICOM Command
Verification	Check the status of DICOM devices	C-ECHO
Storage	Send images to the PACS Archive (Push)	C-STORE
Query/Retrieve	Search over Databases and get the images	C-FIND C-MOVE/C-GET
Worklist management	Get work lists	C-FIND

Storage is a service that allows the SCU to store images in a PACS Archive (C-STORE command). Basically, the modality or image generator (i.e. Storage SCU) sends the images to the PACS archive (i.e. Storage SCP) - Figure 2.4. For each image, a C-STORE Request is invoked. All the contents of the DICOM objects are inside the C-STORE request message. A C-STORE response is sent from the Storage SCP after the file be received.

Query/Retrieve is a service composed of two commands. Query allows the SCU (i.e. workstation) to search for a study or patient, using the C-FIND command (Figure 2.5). The workstation can search over the image archive using several fields like, for instance, patient name, study date and modality. Figure 2.5 illustrates a query action, which looks for exams from today with names starting with A and in the response, two studies were retrieved (Ana and Antonio).

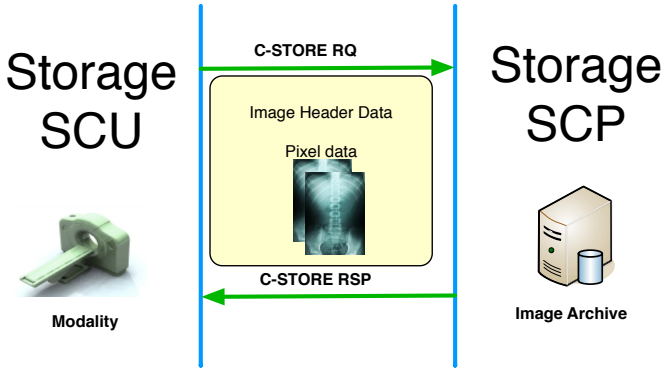


Figure 2.4: DICOM storage service

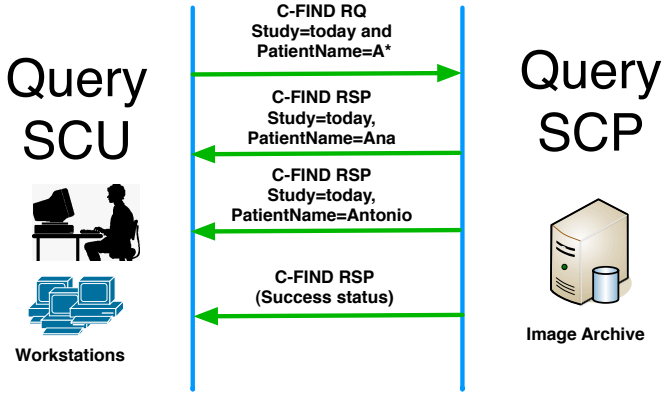


Figure 2.5: DICOM query (C-FIND)

Retrieve method allows the SCU (i.e. workstation) to get/move image from the SCP (i.e. image archive) - Figure 2.6. The retrieve operation uses the C-MOVE or C-GET command. The C-MOVE is a retrieve command that uses a C-STORE to transfer the images. The C-MOVE command does not download the images directly. Instead of transfering directly, it performs an action meaning the image archive sends the study to a specific location that typically is its own workstation.

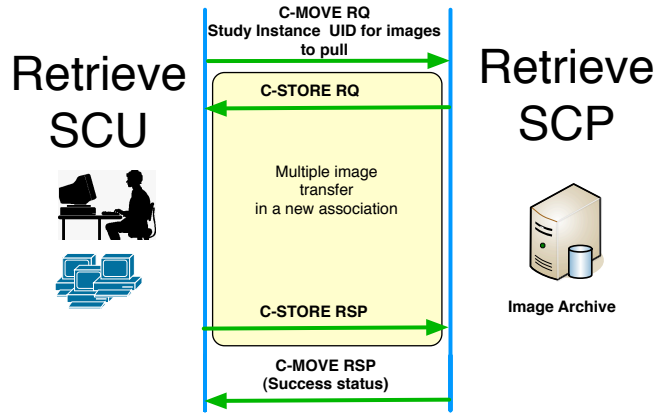


Figure 2.6: DICOM retrieve image

Finally, Worklist Management allows the SCU to get worklist, i.e. patients' studies required by the physician. It uses the C-FIND command to access this information.

Web Access to DICOM persistent Objects – WADO

Web Access to DICOM persistent Objects (WADO) is a web-based extension to DICOM protocol. It allows accessing and viewing DICOM persistent objects, i.e. images and reports. Objects are encapsulated in a HTTP/HTTPS connection, and can there be easily integrated in web applications.

Trans-institutional access benefits from this solution because it is deployed as a web service in the Internet. WADO is a service dedicated to retrieve process, similar to C-MOVE and C-GET commands. It defines the query and mime type, i.e. a content type that defines in what way the information is encoded, for instance "application/dicom" retrieve the DICOM object and "image/jpeg" convert the DICOM image to a jpeg and retrieve it. The WADO query format is according to this nomenclature:

```
http://hospital-url.com/?requestType=WADO&studyUID=
2.16.840.1.113662.2.256403420667961497020507473 7&seriesUID=
2.16.840.1.113662.2.256403420667961497020507473 7.1&objectUID=
2.16.840.1.113662.2.256403420667961497020507473 7000191069
```

This service also has some drawbacks, mainly related to the search methods. WADO does not support search over the PACS archive and it is only possible to download DICOM objects knowing the resource identifier.

2.1.3 Ethical and legal aspects

Privacy and confidentiality

The confidentiality of patients' records is a social and medico-legal issue. Thus, medical images are considered valuable information for many entities including hospitals, doctors, researchers and insurances companies[10]. Because of the importance of this data, healthcare institutions safeguard medical records. However, it is not sustainable to keep the responsibility of maintaining those within healthcare institutions because it is not part of their core business. Outsourcing is a good solution to this problem.

In this paper, outsourcing to the cloud is described i.e. exporting medical records to third parties whose core business is to provide their customers with computing and storage power. Telemedicine and remote diagnosis is also a main reason for sharing medical image. However, in order to guarantee the privacy of shared data, there was an issue: how can a hospital preserve privacy with data outsourcing? In 1993, Peter Steiner published in *The New Yorker* magazine the following sentence: "On the Internet, nobody knows you're a dog". It means that medical institutions cannot trust other companies, because someone can steal the data.

Data sharing is very important in distributed systems in telemedicine, however those systems have to guarantee data protection and privacy. The following aspects is the key point:

- **Storage:** medical images should be stored with privacy, only authorized people being able to access them.
- **Transmission:** transfer of data has to be confidential, avoiding man-in-the-middle attacks or other ways to access the information.
- **Processing:** queries and requests have to be ciphered.

Finally, the outsourcing to public providers depends on a country's laws, and even a high level of security implies dealing with bureaucracy and other political issues.

Data protection

Another topic to be considered is the permanence of patient data. Data protection laws, in several countries, require to know where data are stored. For this reason, storage of patient data in the cloud will be very difficult to use in countries like Spain, France or Italy. However, in outsource to the cloud, several providers allow obligatory data storage in a specific location, for instance Amazon AWS. Thus, the problem addressed can be minimized and even countries with these restriction laws might accept the solution.

Permanence record time

This aspect can be a costly problem for institutions. It also depends on the country where the solution will be deployed because the permanence of patient data varies according to patient age, modality and country. For instance, a one-year-old pathology study of a patient should be stored for long time. Therefore, the infrastructure costs of a healthcare information system vary according to those variables.

2.2 Cloud Computing

Computing devices and Internet access are now available anywhere and at anytime, creating new opportunities to share and use online resources. A tremendous amount of ubiquitous computational power, such as Google and Amazon, and an unprecedented number of Internet resources and services, such as email and storage, are used every day as a normal commodity. Also, Internet bandwidth is plentiful, which allows online data storage and time-efficient remote access from anywhere. So the major idea is that we can get from the Internet cloud, in a self-service model, things that we are used to relying on a central computer and services for.

A Cloud computing service consists of the aggregation of distributed resources into one single virtual system, aiming for virtualization, i.e., decoupling the business service from the infrastructure, and for scalability, i.e., the system capability grows as it is needed [11]. Besides, one of the great advantages of Cloud computing is its resilience. In theory, Cloud computing services are built in such a way that, if a machine fails, the system readjusts itself so the user will never know that one machine failed. Taking into account this resilience, Cloud computing is a promising technology to ensure a level of stability that a single server cannot provide. Moreover, the cost saving on local datacentre infrastructure (hardware, software, air conditioning, fire alarms, physical security, etc.) is tremendous, including continuous IT updates, licensing aspects and electricity consumption.

It is evident that the computing-as-utility business model is becoming prevalent in the electronic world and numerous institutions are adopting it. However, there are also some important weaknesses that must be considered when someone decides to migrate an existent solution (infrastructure and/or application) to a public Cloud provider [12]. One of the most important is the latency introduced with factor distance, a relevant aspect in huge volume data transfers. It is true that broadband Internet links minimize this aspect. Nevertheless, Internet latency times cannot be compared with values obtained in scenarios where servers and clients are located in the same Intranet. Another problem is the lack of interoperability between providers, i.e. services are not interchangeable across cloud providers. The current absence of interface normalization does not allow transparent migration of Cloud applications between providers. Finally, a critical concern is related to security, namely data privacy, durability and availability. The sharing and dynamic resource allocation of Cloud computing reduces user control over proprietary data and poses new security issues when compared with a traditional application server hosted behind an institutional

firewall. Cloud provider selection is important to reduce some security risks. For instance, the Amazon Simple Storage Service (S3) was planned to provide 99.99% availability of objects over a given year, excellent values when compared with intranet datacentres. However, the quality of the client Internet link service is fundamental to ensure data availability.

There has been great investment in building Cloud computing infrastructures for health purposes. For instance, Harvard Medical School built an internal computing cloud to enable collaborative research among several departments and partners. Another example, TC3Health Company, is already providing healthcare players with an integrated solution supported by Amazon Web Services, namely S3, EC2, and SQS technologies [13].

2.2.1 Cloud Computing Layers

Cloud computing rises to import business models. Firstly, applications can be delivered as services over the Internet. However there are many different ways to access to Cloud computing resources.

At the top of this layer, there is a "Software-as-a-Service" – SaaS (Figure 2.7). This model is very close to the customer/end-user side. The key idea is to deliver an application over a web platform to the end-user. These applications are typically web applications that run embedded in the browser. This layer intends to hide complexity from the user. From user's perspective, there is no need to worry about servers, network or even application deployment details. It is a commodity to use these applications like Gmail[14], Dropbox [15], Google Docs[16], and many others. According to cloud computing [17] this kind of software is named as "Software-as-a-service" – SaaS (Figure 2.7).

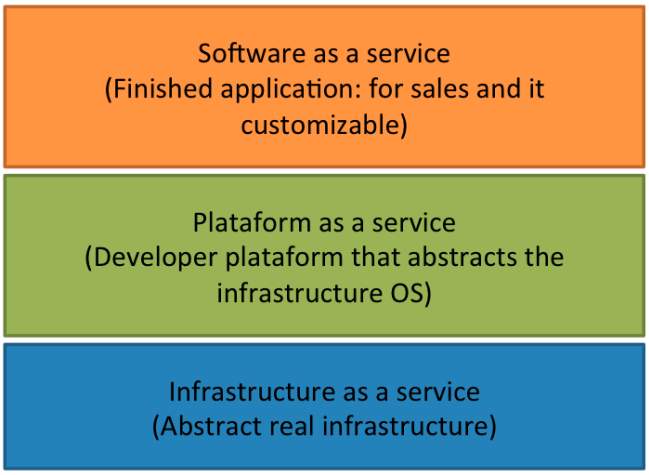


Figure 2.7 - Cloud Layer: SaaS refers to a software provided over the browser, eliminating the need to install applications; PaaS is a facility to deploy a software without the need to purchase infrastructure; IaaS refers to computer infrastructure, mainly virtualization environment as a service.

The goal is that the end-user does not need to download any external applications to use the service or application. There are many advantages in distributing software as a service, but the ability to supply an enterprise application without having to deploy software in the enterprise machinery stands out as a major one.

"Platform-as-a-service" (PaaS) is a lower level layer in the architecture, where developers may create and deploy new applications following the provider's API (Application Programming Interface). PaaS is a platform that contains developer tools to do applications, web graphical interface development, database development, and delivering to a remote hosted platform provided by Cloud provider in a powerful hardware infrastructure. PaaS supplies everything the developer

needs to create and deploy the web applications and web services on Internet based on Cloud services. It is a self-contained platform to build and deploy an application - "Software-as-a-Service". For instance, Google AppEngine and Salesforce are two examples of PaaS that allows the developer to deploy a web application easily.

Finally, but not least, there is "Infrastructure-as-a-Service" (IaaS), whcih supplies resources such as virtualized servers, network devices, and many other resources, e.g. Amazon EC2 [18] is considered a IaaS. It is very important to the Cloud providers because the PaaS/SaaS are working over the IaaS layer.

2.2.2 Deployment models

Since the beginning of computing applications have been on-premise software, i.e., hosted on servers inside the enterprise that own the services. On the other hand, with rise of cloud computing the applicability of off-premise software has increased and nowadays applications are offered as services through Cloud providers. Thus, Cloud computing has been meeting customers' requirements, and distinguishing between different deployment models was a natural progression. Three deployment models can define the cloud architecture: public cloud, private cloud and hybrid cloud.

Public Cloud

Large industry groups leverage the increase of cloud computing services and the SaaS paradigm. Clearly, it is a self-service offered by these companies, where the resources are available anytime and anywhere to customers. The resources are dynamically assigned over the Internet. For the customers, the resources are infinitive. Providers typically offer a web services or web application to access the infrastructure. Third party applications can access and allocate resources in a dynamic way, without complex engineering processes.

Private Cloud

A private cloud is an in-house deployment of Cloud computing infrastructure that leverages the virtualization within the private datacentres. Nowadays there are still servers such as database servers, application servers, government servers and many others that typically run at 5% of their capacity [19], wasting money on their maintenance and operation, for instance, their energy bill. A private cloud uses virtualization, quite similar to what VMware Workstation or Virtualbox does, thereby aggregating many physical servers in just one physical server with numerous virtual servers. Moreover, Cloud computing allows more than one instance of one machine to be run, meaning that if one instance fails, other instances can ensure the service.

This approach has several advantages for organizations, because data and process management are not out of the organization's control. Moreover, several sectors, for instance, hospitals and insurance companies have a strict security measures and legal issues, and several data have to remain inside the organization. Similarly to traditional solutions of on-premise software, it offers local bandwidth and does not require Internet access to get data. Nonetheless, it also has several disadvantages compared to a public cloud solution. In this approach the private cloud owner has to deal with server maintenance and purchase hardware resources to scale up the solution.

Hybrid Cloud

The cloud infrastructure contains multiple cloud, internal and external. A cross solution has several advantages in many dimensions. If we consider applications where performance, privacy

and privacy concerns are critical, it can be a good solution. There are several examples of where this method can be used, for instance web sites hosting in a private cloud, but the huge public data are stored in the public cloud. Furthermore, in a storage solution the public cloud can be used to make data redundant.

2.2.3 Cloud service – categories

There are several services in Cloud computing. We will address the most important services focusing on benefits and drawbacks of these solutions.

Storage-as-a-Service

Storage-as-a-Service is the ability to offer remote storage in a local virtualized way, for any operating system and application. Nowadays, Cloud providers are offering storage using the Blobstore concept, which, *per si*, is not new. In the past, these concepts were used in Database Management Systems (DBMS) in the storage and movement of large data blocks. Blobstores are associative memories, i.e. key-value storage providers, where the blob is unstructured data (value) stored in a container and the lookup is performed through a text key. A container is a namespace or domain for the blobs. A blob is always uploaded to a container. The blobstores have a list of containers where the developer can create, remove or rename them. The container holds content, which can be blobs, folders or virtual paths.

Cloud providers have released the blobstore service that allows their customers to store data in a container over the Cloud. For instance, Amazon S3, Microsoft Azure and OpenStack Swift have blobstore APIs. These services are considered SaaS because they allow developers to take advantage of the remote storage service to support their application data in a transparent way, without worry about scalability. There are many examples of SaaS use, for instance, the Dropbox application that stores customers' files in Amazon S3 or commercial web portals that store great quantities of pictures in cloud blobstores.

Database-as-a-Service

Nowadays, databases are vital in enterprises' applications. They represent a very important role in business software, and enterprises have to purchase the required hardware, deploy database products, acquire database licenses, insure network connectivity and employ professionals to maintain database systems. Database-as-a-Service (DaaS) is a new paradigm that outsources these costs to the cloud provider. Therefore, the database is hosted in a remote datacentre and can be shared between users in a transparent way.

For instance, Amazon AWS, Windows Azure [20] and Rackspace [21] offer a database as a service where customers pay for what they use. All database operations are supported by these services, for instance, creating tables, loading and accessing data in the tables. Cloud players often supply an API to access the database, and execute operations through a web service API (Figure 2.8).

Furthermore, database maintainers do not need to worry about the server's redundancy, upgrades, back-up plan and recovery from disaster. Nonetheless, some enterprises are concerned about enruring data privacy. In fact, this is one of the weaknesses of DaaS. Despite Server Level Agreements (SLA) by Cloud Providers, there is no guarantee that an intruder in the cloud company does not access the clear data. Also, store procedures and triggers might not be supported in the overwhelming majority of cloud providers supplying DaaS. Finally, performance might deteriorate because applications will access the data in remote datacentres when located in the public cloud provider.

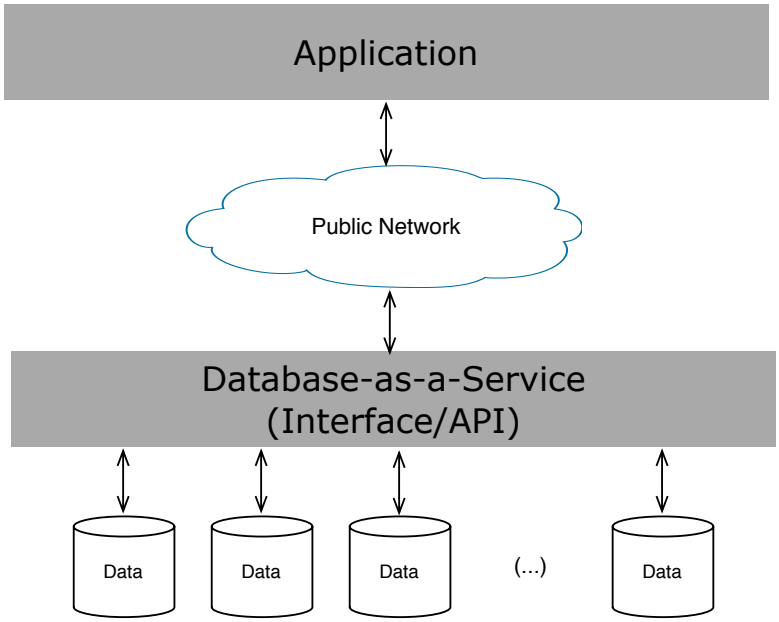


Figure 2.8 - Database-as-a-Service

Signaling-as-a-service

Nowadays real time information is very valuable and everything that happens has to be checked in less time possible. Cloud providers were influenced by this tendency and also created services to communicate in real time. "Signaling-as-a-Service" or notification services refer to the ability to communicate with another remote entity through Cloud Services.

There is some variety of signaling services, but all of them use the Publish/Subscribe pattern. Several companies work in notification services, such as Amazon SQS [22], PubNub [23] and Azure Queue [20], etc. All of them have the same concept: a web service that sends notifications to hanging users by event-driven workflow.

2.2.4 Cloud Platforms to development

Previously in this chapter, the differences between cloud computing layers were discussed. Rather than providers supplying end-user applications for well-defined purposes, cloud providers supply an API as a service that allows the developer sto implement their need functionalities. These services may include hosting as part of the service. Typically the PaaS includes: web services, service platforms and management dashboard. It is a set of software that provides tools a developer needs to build a Software-as-a-Service solution.

Google AppEngine

Google AppEngine supports two widely known programming languages, Java and Python. This platform also provides several application gadgets and extensions. For instance, it allows high interaction with Javascript through a framework named Google Web Toolkit (GWT). GWT is a platform that generates AJAX/javascript code, which supports lot of features such as asynchronous events, internationalization, etc.

Google offers the platform and the developer can host for free the application in the Google cloud services, although the free services have limited use. The cloud services provided by Google are automatic scale and load the balance of the application.

Azure Platform

Microsoft's equivalent platform is Azure, which also allows developers to run applications into Microsoft datacentres, i.e. over the cloud. They offer a SDK that permits the development of applications with their tools, and uploading them to Microsoft servers. SDK also enables the developer to run applications locally. They support C#, Java and Ruby web applications. Unfortunately, they only offer free accounts with several limitations. More recently, they have allowed the user to try out the service, although this requires credit card information.

Salesforce

Salesforce is a company providing SaaS. They also provide a PaaS named Force.com. It allows the developer to deploy the application in the Salesforce cloud hosting and integrates with the development tools of Salesforce. They use Apex programming language, which is a Java-like language. They also have a VisualForce that allows creation of application to web browsers, generating HTML and Ajax or Flex program. Although this platform has many advantages, it also has some drawbacks. For instance, Apex is a proprietary language, based on Java 5, that is an old version of Java. The developer platform lacks many features for the developer, for instance, code refactor. Moreover, the method for deploying the solution in the cloud hosting is complicated.

2.2.5 Standards and Cloud Interoperability

The Cloud market has grown over the past few years and follows IDC (International Data Corporation) researchers. In fact, there is a huge amount of interest in the IT industry to migrate services to Internet Cloud platforms. In order to response to their request, many cloud companies have been created to meet their demands. There was a significant increase from Cloud providers in offering new features for consumers. For instance, Amazon Web Services has released many services to fulfil their customers' requirements: S3, SQS, SimpleDB, etc. In turn, Google AppEngine [24], Windows Azure and many others improved their solutions with new APIs to overcome the challenges of their targets. They provide many services on the server-side, or platforms where users can deploy their applications more easily. Moreover, there are many types of free Software-as-a-Service supplied by an unprecedented number of entities, organization or companies, (e.g. Gmail, Dropbox, Live Mesh, SkyDrive, and many others). Unfortunately, access for those kinds of platforms is very restricted, and only official applications can enjoy the facilities over the cloud.

There is great competition in a large industry to provide better and more services over the Cloud. However, these services provided by different players are not mutually compatible. Undoubtedly, interoperability is required to allow applications to be ported between different cloud providers.

Recently, many groups have been formed to create standards and common interfaces with the intention of enabling interoperability between different cloud players. The Storage Network Industry Association [25] has been working on a storage data standard in the cloud [26]. This standard explores the features that vendors are offering and extracts the common domain, aiming for a high quality cloud storage interface in the future. Also, they focus on the paradigm of "pay-as-you-go" because they consider that such attributes will interest many businesses.

Cloud Computing Interoperability Forum [27] aims to standardize computing virtualization over the cloud. They are an open and vendor-neutral organization, which intends their solutions to be quick, potentiating successful adoption in industry. There are also other committees with the

same goals, for instance, Open Cloud Computing Interface [28], Open Grid Forum [29] and Open Cloud Consortium [30].

Although several organizations have come together to constitute interoperability groups, the services over the cloud are not yet compatible yet. It is not a technical issue, because several standards provide the features that the cloud providers supply. Hence, it seems more a political and economic issue, whereby providers can offer their own features and stand out from other cloud vendors.

2.3 Related work

The pace of medical image generation has increased significantly over the last decade [31-33], as a natural result of the increase in studies typically performed in medical centres. Moreover, this trend is expected to continue over the coming years [34]. According to [4], it is estimated that over one billion diagnostic imaging procedures will be performed in the United States during 2014, representing about 100 Petabytes of volume data. Considering this expectation, medical image archive systems will have to scale up leading to a gradual increase in their maintenance costs. On the other hand, there is a new general tendency to outsource computing power and storage capacity from desktop, portable PCs, and mobile devices to large datacentres [9]. This fashion does not represent a completely new concept, given that there are several well-known solutions, such as mainframes, clustering, grid computing, and the most recent Cloud computing.

The use of DICOM services within a healthcare situation is a common activity. The volume of data generated, for instance, by dynamic cardiac modalities (e.g. XA and US), multislice CT, high-field MRI and digital mammography is tremendous and this will increase in the coming years [34]. For instance, CT scans (64/128-slices), PET, MRI with high-resolution have typically more than 100MB. Besides the huge amount of information there are many concerns about the safety of data, meaning that losing studies could become a nightmare for PACS administrators, physicians, medical staff, and obviously for the patient. There is also a new movement to outsource data storage to external datacentres institution [35, 36], reducing the maintenance costs within medical institutions, of what is not their core business.

PACS outsourcing based on the Application Service Provider (ASP) model is a concept that has been maturing over the last decade. Basically, it is possible to identify two main types of ASP PACS model. One is the storage service provider model, where medical imaging images are stored off-site and the customer will obtain them on demand. This storage model is used particularly to implement backup and long-term archive servers [37-39]. The other model is a full PAC system renting, including applications, database, storage resources, etc [37]. Those components are hosted in remote servers not owned by the healthcare institution. Operations are executed on the server side and only results are displayed in the end-user computer. Typically, third-party ASP vendor provides those PACS solutions and institutions are obliged to pay for the service, a fee usually based on the number of examinations stored (or Mbytes used) per year. In some scenarios, they represent a very attractive financial solution [12]. Nevertheless, several risk factors were identified, for instance, legal issues, security aspects and institutions wary of long-term ASP vendor reliability [37].

The Grid computing paradigm has been also explored to provide federated access to distributed image repositories [36, 40, 41]. Distinct usage scenarios could be identified, including backup services and multicenter PACS. The repositories are typically distributed over independently owned and geographically distributed servers. The Grid has a middleware layer that provides tools, runtime support and access services to a federation of image data resources. Those

PACS infrastructures are usually private consortiums and normalized technological solutions are not used. Moreover, the consortium often imposes minimum requirements on the participants' hardware and software configurations [12].

There is an implementation of a medical image file system, [36] which presents a Grid computing model in co-location. The authors claim that the bottleneck occurs typically in the PACS server and propose a distributed file system in different grids, where studies are split into blocks and the blocks are spread over different locations. Several strategies have been implemented to avoid flooding PACS servers, present some predictable methods and to avoid overload. Transfer times have been reduced by using distributed file locations and intelligent heuristics to download files faster.

Also, another approach [35] presents a similar scenario based on Cloud computing. In this case the solution focuses on exchanging, storing and sharing medical images across the different hospital. Apache Hadoop is a distributed file system, which is used by several important companies to distribute their contents, e.g. "Yahoo!". The solutions can be installed on a private cloud or, on the other hand, deployed on a public cloud, even though in both situations there is a significant set-up difficulty. Besides, it is a scalability solution providing replication, though without security concerns.

Finally, "the medical image archive solution in the cloud" [31] work is a "PACS-as-a-Service" solution similar to the aim of this thesis. The present solutions grant interoperability with DICOM devices and implementation was created in Microsoft Windows Azure. However, this solution does not supply any security method, and is therefore an easier target to attack. Moreover, no results are given regarding performance and robustness of the presented architecture. All solutions discussed are dependent on technology, platform or provider.

2.4 Analysis of Cloud computing applications in medical imaging

To analyse the Cloud computing in the medical imaging scenario we formed a comparative table to choose between traditional PACS approaches and Cloud computing solutions, which incurs a trade-off between many aspects detailed in Table 2.2 using a SWOT (Strengths, Weaknesses, Opportunities and Threats) analysis.

Several considerations can be expressed regarding this analysis. Medical data over the cloud has a lot of issues, mainly in the security. Moreover, medical imaging over the Cloud is more complicated to deploy in real scenarios because it outsources data to servers owned by external companies. Another weakness is that it depends on the ISP service provider. However the ISP is growing faster and providing more bandwidth at lower prices, e.g. in Portugal there is an ISP supplying 360Mb/s to home connections. Thus, it is an opportunity that the PACS Cloud solution should explore.

Traditional solutions have high costs for the medical centres, but if the PACS infrastructure is already deployed, medical institutions just have to maintain it. Nevertheless, maintenance of the datacentre is very expensive and over the years the hardware becomes obsolete. In the PACS Cloud solution, hardware upgrades are not necessary and the hardware has also the expected performance. It can also be adopted in scenarios, for instance, when PACS archive expansion is required. Despite PACS Cloud's drawbacks, it created new opportunities for new centres that want a solution without spending money on PACS archive.

In this thesis we will tackle the controversies found in the PACS Cloud solution, in order to minimize the weaknesses and threats.

Table 2.2: SWOT Analysis of Cloud Computing versus Traditional PACS solutions (in-house)

	Traditional PACS Solution (in-house)	Medical Imaging over the Cloud
Strengths	• Privacy • Confidentiality • Ownership	• Low financial risk • Low initial investment
Weakness	• Cost associated with deploying the solution/hardware and software maintenance • Technicians to maintain large datacentres • High initial investment	• High bureaucracy to deploy in real environment • Depends on third parties (ISP and Cloud Providers) • Interoperability with DICOM devices • Privacy leaks • Confidentiality • Ownership of medical records
Opportunities	• Capacity to provide high QoS through the intranet. • True interoperability regardless of DICOM platforms	• Hospital spends a lot on hardware and software maintenance • Telecommunications service more efficient • Technicians to maintain large datacentres
Threats	• Typically, it does not have physical redundancy	• Cloud providers own the medical data, and medical records can be stolen.

3 PACS-as-a-Service analysis

"Great things are not done by
impulse, but by a series of small things
brought together."
Vincent van Gogh

The exploration of new approaches to medical imaging communication was the main motivation behind this work and led to improvement of solutions in this research area. In this chapter, requirements and the proposal to address the problem are introduced.

3.1 The problem

In the previous chapter the state of the art of PACS, DICOM and several technologies to outsource computing and storage were presented. These areas are crucial to understand the problem that this thesis intends to address.

The equipment to acquire medical imaging in the various modalities has become less expensive. Consequently, the main result is the proliferation of devices for acquisition of medical imaging, even in small centres. Therefore, we have dispersed repositories in different medical centres with dissimilar modalities that leverage patient flow across several medical centres to perform all the necessary exams. The process to exchange medical exams between medical institutions is crude. Currently, healthcare institutions send medical exams via conventional mail (analogue films, CD, DVD), Virtual Private Networks (VPN) or manual email.

This thesis intends to solve the problem by creating new mechanisms to automate sharing and access to external image repositories. It will support telemedicine and tele-work, which are important in remote diagnosis nowadays. Moreover, there are small centres without the financial resources to invest in PACS archives with a backup system and all the security required. Thus, an architecture description and prototype implementation is proposed. In addition, the exploration of communication between multiple institutions, performance, security and access control is required. A PACS archive over the Cloud will promote inter-institutional access, although it also brings new challenges regarding the data security, which is another problem this thesis will address.

3.2 Conceptual Requirements

This thesis aims to create a new PACS platform with a well-defined goal: create a medical image repository, where medical institutions of the same trust domain can store, access and share data, also allowing telemedicine scenarios. We will define the functional requirements that PACS Cloud systems needs to be deployed in the real environment.

3.2.1 Interfaces with external devices

A great number of devices cannot communicate with Cloud computing interfaces. In medical imaging, communication between medical devices follows the DICOM standard. On the

contrary, Cloud data stores and database access are not DICOM compliant. Most public cloud providers supply access to their services through a web service interface. Thus, this thesis proposes to keep interoperability between DICOM equipment and PACS Cloud solution compatible both with medical practice and pre-existing medical information systems. Interoperability is an important requirement for this project because it will keep the existing equipment compatible without any break.

The following main DICOM services should be supported:

- **DICOM Storage**: is the service provided to exchange medical image data or other DICOM objects between devices and the PACS archive. It pushes images to the PACS Cloud storage resources. DICOM Storage is a service with a great impact on the solution because the medical repository will store the data through the Cloud storage service.
- **DICOM Query/Retrieve**: search the database and get the images stored in the PACS. These procedures should be compatible with Cloud database services. Moreover, retrieval should work over the Cloud storage services.

Finally, another service must be considered: DICOM Verification Service, given that it is important to check the connectivity with PACS archive.

3.2.2 Access "anytime and anywhere"

There is also another issue in healthcare information systems: high restrictions to obtain access from outside institutions. DICOM is the de facto standard to operate and transfer medical imaging between modalities and information systems inside hospitals. Due to security restrictions and fallback strategies, communication with the outside is not allowed. Opening external firewall ports involves a degree of trust, which is unacceptable to many organizations, even to enterprise networks. Thus, we have the problem of communication and access to the PACS without losing control and guaranteeing the privacy and confidentiality of the patient and clinical staff. So the system should be in compliance with the security policies already applied in the medical centres to avoid opening any security hole. Typically, medical institutions' access rules have just two main acceptable policies: HTTP/HTTPS and email protocols (e.g. IMAP, POP3, SMTP).

Thus, the new PACS archive must be quickly available even considering the ports or protocols that are available to be used. We are promoting a multi-institutional shared repository that promotes telemedicine scenarios and works in the "anytime and anywhere" philosophy. In Figure 3.1 we present a multi-institutional communication example. If we consider that Hospital A and Hospital B are in the same trust domain, they are allowed to share medical data. We have several solutions to promote this requirement. The main requirement of this thesis focuses on creating a repository shared between them. We introduced a shared PACS archive in the figure, which both institutions can access. On the other hand, they can just share their local PACS archive to grant access to the other institution. It is a supplementary requirement, which intends to allow access to DICOM devices from outside institutions in an easy way. Moreover, this new paradigm maximizes PACS efficiency because the solution can serve multiple sites, facilitating inter-institutional data share and common workflows [8].

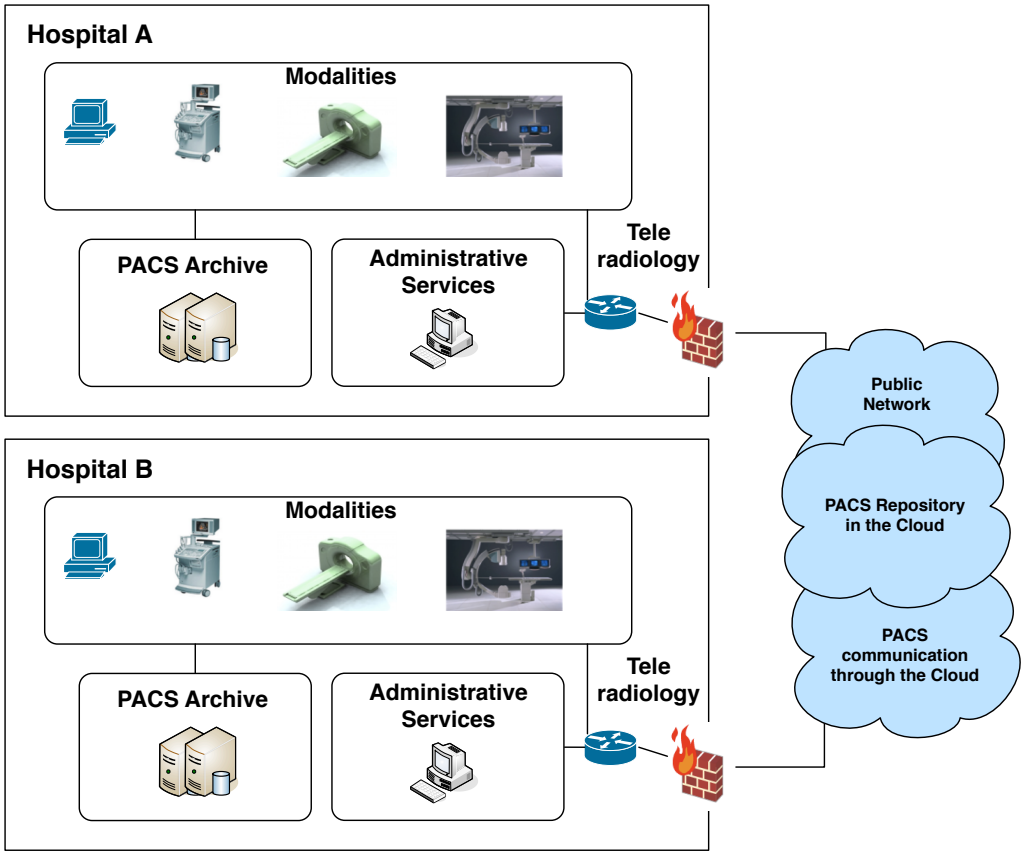


Figure 3.1: Illustrative example of multi-institutional access between two institutions.

3.2.3 Access Control

To solve the issue of sharing studies between different medical institutions we will create a new problem. Which medical data is allowed to be shared by the medical institutions?

The idea of this new PACS approach is to allow inter-institutional access data, but the interested party has to be allowed to access the information. The application must be able to manage all institutions and trusting relations between them. The PACS administrator must have a interface to customize all these settings quickly.

3.2.4 Data privacy and confidentiality

Due to the great importance of medical information, data privacy is a vital requirement and a very delicate issue, particularly when medical digital images and patient information are stored in with third parties and transmitted across the public networks. The outsourcing of medical data to external entities, i.e. cloud providers, implies both. On one hand, medical data are stored outside the institution and on the other hand, the information is transmitted over public networks. Cloud computing is dynamically scalable, grants redundancy and performance and provides a high level of security. Despite all the advantages, there are no guarantees that cloud providers will not audit information under their ownership. Thus, another challenge of this thesis is to protect the privacy of clinical staff (e.g. patient and physicians) and the protection against tampering data.

3.2.5 PACS archive configurations

A PACS archive has operational configurations, for instance, configuration of AETitle, acceptable transfer syntaxes, supported SOP Classes, etc. The shared PACS repository has to allow the PACS manager to set up these settings through an easy interface.

In addition, access control and register of medical institutions through the graphical interface is required.

3.3 Non-Functional Requirements

3.3.1 Outsourcing of data repository

The PACS archive has a huge amount of IT infrastructure and modality equipment that allows communication via DICOM. Medical images are very important records, and so the storage repository needs redundancy to be a reliable system. Medical institutions can reduce the costs of local storage maintenance through PACS outsourcing.

Moreover, outsourcing is an opportunity for small image centres that purchase modality equipment, although they do not have the financial resources to buy software and hardware to keep up a PACS repository that grants redundancy/backup system. A mandatory requirement of this thesis is to create an archive based on the outsourcing concept, when storage, database and processing are outside the medical institution. The considered approach will outsource medical imaging data to the Cloud computing, to avoid the maintenance of their own datacentre to hold the PACS archive.

3.3.2 Portability

The system should support several different Cloud computing providers. It is a key point to have this feature because the solution will be vendor-neutral, thus aging and PACS upgrading is not an issue. It will avoid vendor lock to one system and create an opportunity to migrate the PACS to another Cloud provider easily.

The applications should be ported to different systems, with other file systems, interfaces and internet accesses. So each developed module may be a web application or multi-platform application to run in different operation systems.

3.3.3 Performance and robustness

The medical sector is a critical sector where performance and robustness are very relevant. As discussed in the state of the art (chapter 2) Cloud computing has some drawbacks and there may be a critical delay in accessing the medical imaging stored over the Cloud. It is a quite serious problem that has to be addressed.

An important requirement for the system is that it works in a safe mode without Internet connection, for the on-going medical processes. Subsequently, just minimal services may be provided.

3.4 Financial analysis

The focus of this thesis is to present a PACS archive that stores the medical image over the Cloud and promotes inter-institutional access. One of the main concerns about the solution was to create a system that could be really deployed in a real scenario. However, if the solution is not financially viable, the healthcare industry will drop the idea. Thus, a financial analysis was required to prove that Cloud computing is a desirable solution for this medical imaging scenario.

To address this issue we resorted to a comparison. On one hand, we analyse the price of a traditional PACS archive and its maintenance costs. On the other, we use the Amazon AWS prices to estimate the costs of the PACS Cloud archive deployment. Moreover, we make a case study considering the average medical image produced by medical institutions.

In this case study, the calculation the average amount of exams produced by medical institutions was based on the average of exams produced between 2001 and 2008, according to the INE (Instituto Nacional de Estatistica), the Portuguese national statistics centre. Additionally, as this analysis focuses on a PACS archive over the Cloud, we are interested in the amount of data a medical institutional has to store over the Cloud. Besides, we used the average size of exams for each modality, estimated in "Storage Management: What Radiologists Need to Know" [42].

Our goal was to make a study of a PACS in the Cloud versus a traditional one. However, this is dependent on many variables and it is not possible to obtain accurate values. We did an analysis with estimated values assuming the following conditions:

- A prediction of exams produced based on 2001-2008 results from INE referred to in. The 2011-2015 prediction was used [43].
- Data volume (Mbytes per exam) does not decrease/increase each year. It was considered a constant value, and the values were mentioned in this paper [42].
- We considered that a hospital in the case study contains all modalities, although calculation of specific modalities is unimportant, based on the results that we will present.

As an illustrative example of auxiliary calculation, we considered the exams produced in 2008, to calculate the average data volume produced each year (Table 3.1). In the INE data, we have the total number of exams produced for each modality in all hospitals and the number of hospitals considered. Thus, we can calculate the average number of exams per hospital by dividing the total number of exams by the number of hospitals. Finally, we used the average size of the modality exams [42] and calculated the average amount of Gbytes produced for each modality by hospital per year. This process was iterative and we did it for each year between 2011-2015 based on prediction [43].

Table 3.1: Average amount of medical imaging produced in 2008 in a subset of hospitals considered in INE

Modality	Total number of exams produced	Number of Hospitals	Average size per exam (Mbytes)	Average exams per hospital	Produced Gbyte/year for each hospital
XA	14646	50	15	293	4.39
Ultrasound	1020012	138	9.2	7391	68.00
Radiology Exams	5146809	144	42	35742	1501.15
Mammography	86756	94	190.72	923	176.02
MR	114938	42	26	2737	71.15
CT	875561	140	262	6254	1638.55
Other	51946	50	135	1039	140.25

The analysis of the traditional PACS was performed based on the following variables' costs: server hardware, network equipment, licenses, energy, air conditioning and maintenance. The original resource is "Amazon EC2 Cost Comparison Calculator[1]", which is an Excel spreadsheet that allows comparison of solutions: on-site, co-location and Cloud computing. On the

[1]AWS Economics Center: http://aws.amazon.com/economics/

other hand, to compute the costs of PACS Cloud we considered the prices of Amazon AWS (May 2011), considering an equal amount of data. We considered costs of data storage, posting/receiving operations and data transfer. In Table 3.2 we present the intermediate calculation to obtain the price per year of the PACS Cloud. Retrieval of each study was considered 3 times.

Table 3.2: Estimated price of PACS Cloud Archive per year

Year	Amount of exams	Amount of data produced/year (Gbyte)	Accumulated data (Gbyte)	Price storage/year	Price PUT/GET	Transfer IN	Transfer Out	Final Sum
2011	10933733	3882.46	3882.46	€ 385.92	155.259	275.65	1240.45	**€ 5,553.82**
2012	11368996	4075.57	7958.03	€ 791.03	161.4397	289.37	1302.14	**€ 9,710.98**
2013	11827091	4281.22	12239.25	€ 1,086.23	167.9447	303.97	1367.85	**€ 14,079.01**
2014	12309460	4500.3	16739.55	€ 1,485.64	167.9447	319.52	1437.85	**€ 18,664.86**
2015	12817651	4733.79	21473.34	€ 1,905.76	167.9447	336.1	1512.45	**€ 23,489.83**

In Table 3.2, the costs per year of PACS Cloud Archive in an average hospital are presented. In Table 3.3 we present a comparison between a traditional solution and the PACS Cloud archive solution.

Table 3.3: Comparison between traditional PACS and PACS Cloud solution (estimated values)

On site	2011	2012	2013	2014	2015
Servers	€ 15,000.00	€ 500.00	€ 15,000.00	€ 500.00	€ 20,000.00
Network equipament	€ 5,000.00	€ -	€ -	€ -	€ -
Licences	€ 1,000.00	€ -	€ -	€ -	€ -
Energy	€ 700.00	€ 770.00	€ 847.00	€ 931.70	€ 1,024.87
Conditional Air	€ 1,200.00	€ 200.00	€ 200.00	€ 700.00	€ 200.00
Maintenance	€ 7,000.00	€ 3,000.00	€ 4,000.00	€ 5,000.00	€ 6,000.00
On site	€ 29,900.00	€ 4,470.00	€ 20,047.00	€ 7,131.70	€ 27,224.87
Accumulated On Site	€ 29,900.00	€ 34,370.00	€ 54,417.00	€ 61,548.70	€ 88,773.57
PACSCloud@Amazon	€ 5,553.82	€ 9,710.98	€ 14,079.01	€ 18,664.86	€ 23,489.83
Accumulated PACS Cloud	€ 5,553.82	€ 15,264.80	€ 29,343.81	€ 48,008.67	€ 71,498.50

The accumulated yearly price of the solutions, to analyse the investment of the solution in 5 years (2011-2015), was calculated using the following formula:

$$Cost(CurrentYear) = \sum_{i=0}^{Current\ Year} Cost(Year_i)$$

In the formula we reckoned that the initial investments are considered in year 0, which is the nomenclature of financial analysis, although in the table it was already included in the first year. Calculation of the accumulated costs allows comparison of the investment at the end of 5 years, and checks which one is more profitable (see Table 3.3 accumulated on site versus accumulated PACS Cloud).

Figure 3.2 presents a detailed comparison of the two approaches, showing the accumulated costs of the solutions over the years. PACS Cloud has several advantages in the initial years. It suits well the requirements of a small centre because it does not require initial investment. However, there is a point of operation where it is economically more rational to have datacentre storage in co-location. It is very difficult to define this tipping point because the cloud services market is rapidly changing, providing more resources at lower cost.

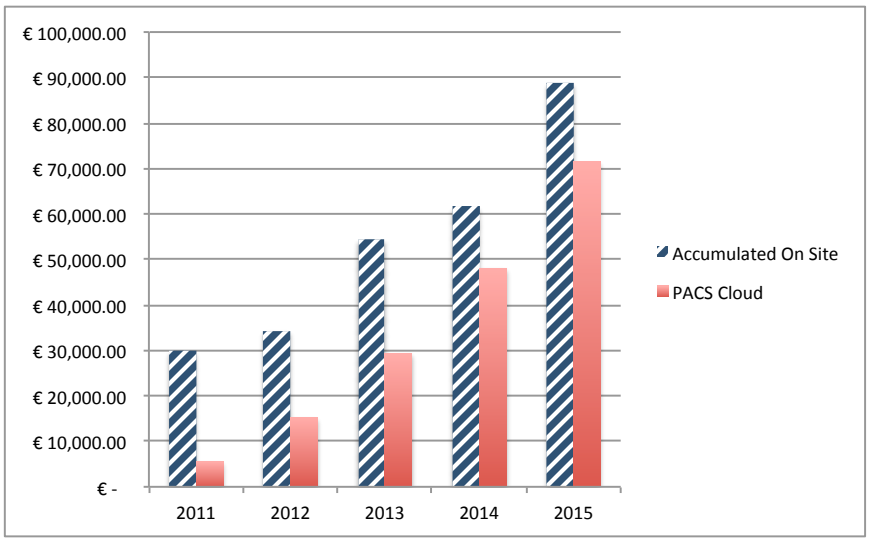


Figure 3.2: Comparison between PACS on-site versus PACS Cloud

3.5 Proposal overview

In software engineering a problem with the same requirements can be solved by several methods. To solve the problem, we decided to implement an architecture based on Cloud computing services due to their incremental scalability, agility, reliability and service orientation. These characteristics provide a solution to the problem. Nevertheless, Cloud computing by itself does not solve all PACS communication issues, or problems addressed in the requirements.

A possible approach to solve these issues was to create two complementary services:

- PACS Cloud archive: is a repository under the cloud to support medical digital images. This system will be interoperable with DICOM devices inside medical centres. Moreover, the system will allow different institutions to access the same repository as long as they belong to the same trust domain.
- DICOM Cloud Router: is a router that allows combination of two or more distinguish DICOM networks, even considering that we are in private networks that do not allow direct DICOM communication with the outside.

These two components provide a solution to the problem of inter-institutional communication. There are some common procedures between the two components, and so this approach ties them to each other. We decided to create a third component that will be the parent of these components. The basic idea is to create a platform that supports telemedicine solutions over the Cloud. It will also be very useful for expanding their services to other research areas. This platform will be explored in the next chapter. In addition, implementation of PACS Cloud solutions and integration with the main platform will be studied in the next chapters of this thesis.

4 Service Delivery Cloud Platform

"If the facts don't fit the theory, change the facts."
Albert Einstein

In this chapter, a framework to support the medical imaging repository and DICOM communications is presented. We describe the platform architecture and the methods used to implement the solution.

4.1 Description

The increasing pace of evolution in business computing services leads enterprises to outsource secondary operations that are not part of their core business. The cloud computing market has been growing over the past few years, and as a consequence, many cloud companies are now offering a rich set of features to their consumers. Nowadays, we have a great number of Cloud providers, e.g. Amazon AWS, Google AppEngine, Azure and Rackspace. Unfortunately, those cloud players have created new services, with different APIs, which imply that cloud-oriented applications might be instantiated in one single cloud provider, as described in section 2.2.5. This scenario is not desirable to the IT industry because their applications will become provider-dependent. Some efforts have been made to create standards, to grant interoperability among the several cloud providers, e.g. SNIA and CDMI as described in state of the art (section 2.2.5). The idea is to create abstractions and interfaces that should be used by all cloud providers. However, these standards are still drafts without any practical impact.

In fact, integration of enterprise applications with the cloud causes vendor lock, creating barriers to industries migrating their data and applications to the cloud. The main goal of our architecture – SDCP, Service Delivery Cloud Platform – is to create services and applications that can be delivered in any cloud infrastructure, respecting the concepts of Storage-as-a-Service, Database-as-a-Service, and "Signaling-as-a-Service". The presented architecture consists of a hybrid infrastructure that allows "Enterprise to the Cloud" and "Enterprise to the Cloud to Enterprise" applications, i.e. communication between two or more different enterprises, using multiple resources from different cloud vendors. Thus, we have created an aggregator of all cloud providers' resources that use the same interface to access/store information in any cloud provider, in a unified way. Despite of the all web cloud platforms that already exist, this platform is different from the others because it allows the on-premise applications [2] to outsource their information systems to the Cloud.

In order to solve interoperability and issues of incompatibility a platform was created with two main goals: (1) grant interoperability between cloud providers, creating an abstract layer for several cloud services; (2) deliver new services to the cloud, granting interoperability with

[2] on-premise application: software installed and run inside the enterprise

protocols that already exist. The first goal (1) provides interoperability between cloud players in a transparent way. We present a platform that allows applications to interoperate with distinct cloud providers' services using a normalized interface. The proposed approach provides a common API that minimizes actual deficit of cloud API standardization and provides secure and redundant service allocation. Basically, our application can work in as many vendors as is wished, taking advantage of the existent cloud providers. Moreover, the Service Delivery Cloud Platform allows creating of storage provider poll of multiple cloud vendors, creating a federate view of all containers. In addition, it gives the developer the possibility to grant interoperability with other protocols (2) inside private networks. For instance, it allows creating of off-premise application[3] that works inside the organizations but the storage/database resource is from the cloud.

The proposed architecture has basically two main components: the Cloud Controller and the Cloud Gateway (Figure 4.1). The Controller contains sensitive information and therefore must be deployed in a trustable provider. The Cloud Gateway is the component that makes the connection between the local applications and the cloud applications. This component allows loading of new plugins that take advantage of the cloud resources infrastructure without provider dependence. Within the SDCP architecture, we have also developed a SDK (Software Development Toolkit) that simplifies the development of SDCP-based applications. The SDCP-SDK contains the abstractions to write in any cloud provider in the diverse services. Thus, new applications must use this SDK to write in any cloud provider.

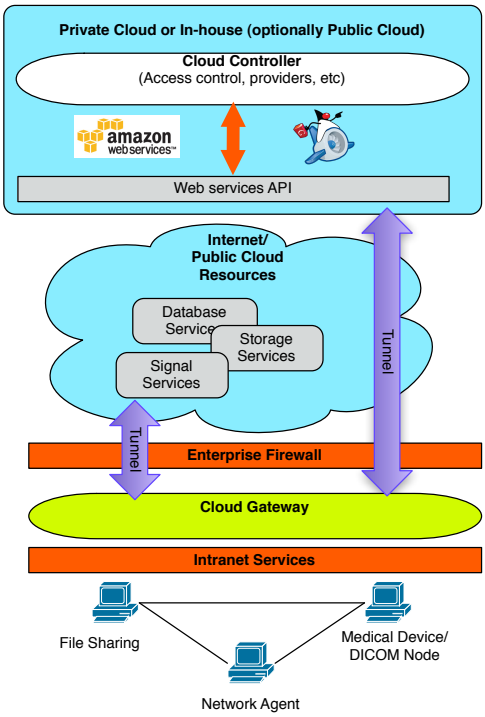


Figure 4.1 – SDCP general overview: Cloud Controller contains the references to cloud providers; Cloud Gateway communicates with Cloud Controller and uses Cloud resources.

The SDCP is able to deliver new services more easily using the cloud facilities: data store, databases, signaling and communication using cloud providers. In order to extend the platform to support different providers and services (e.g. Google, Amazon, Azure, Rakespace, etc), we have

[3] Off-promise application: it is software that runs out of the enterprise, which is frequently called "software-as-a-service"

built a specific model, the structural design of which sustains use of different modules in the same interface. We will describe each component of this architecture in the following sub-sections.

4.2 SDCP – SDK

The SDCP-SDK defines all concepts, contracts and specification of the platform, including communication between the Cloud Controller and the Cloud Gateway. The platform was developed in Java through a set of interfaces. This section describes the main entities involved and explains the developed abstractions for the diverse cloud players.

4.2.1 Entities

The fundamental entities of the infrastructure are described in Figure 4.2, including their relations. A description of each entity follows:

- **Agent**: each gateway has to login using an agent account. Basically, agents are the entities that live inside the enterprise/healthcare institution and relay the information to the cloud.
- **Domain**: is a group of agents that belong to the same enterprise/healthcare institution or the same trustable group/enterprise group. Thus, only agents of the same domain can communicate or access the data belonging to its domain.
- **Provider:** defines a cloud provider and credentials to access them. It can be a storage, database or signaling provider.
- **Service:** external services that can take advantage of Cloud Controller agents and cloud providers. This service will extend the functionality of the Cloud Controller.

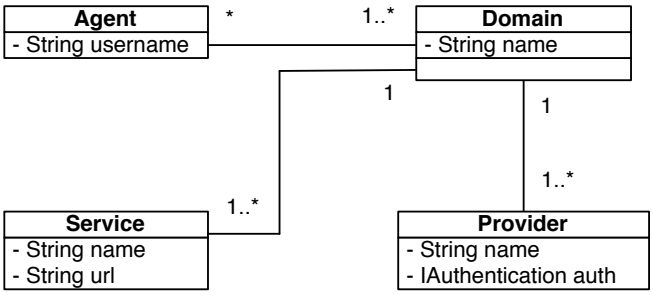


Figure 4.2 - Entities of Service Delivery Cloud Platform

These entities are actors and concepts of the SDCP. The domain is a very important concept because it characterizes the trustable model.

4.2.2 Cloud Streams

As expressed, the goal of this platform was to use any resources of the cloud without provider dependence. To implement that process, in the storage service context, we used an abstraction to write a set of bytes (i.e. blob) into the Cloud storage using the typical Java Input/Output (I/O) streams. The designed abstraction issues the provider independence but also makes it easier to extend to other cloud players. So two new I/O entities, named CloudInputStream and CloudOutputStream, were implemented (Figure 4.3). These entities are used to read/write in the storage services as a common Java stream mechanism.

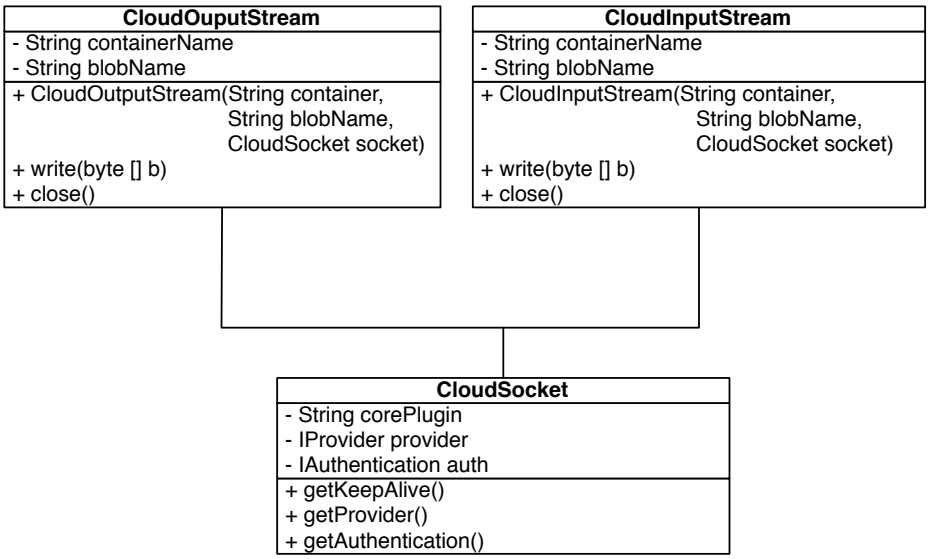


Figure 4.3 - Cloud Input/Output streams

The developed Cloud Streams extend the IO Java streams. The Cloud socket contains the identifier of the implementation that will be used to call the correct implementation for a specific cloud service. JClouds [44] is an open source framework for cloud development that already provides several cloud players, and as such we have decided to build the Cloud Streams as an instance of jclouds blobstores.

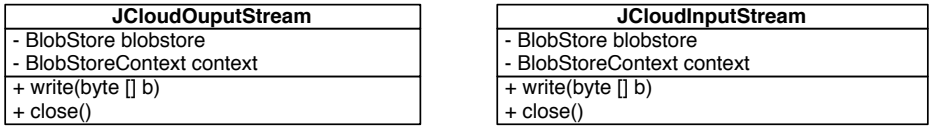


Figure 4.4 - Cloud Streams - JClouds instance

These are major components of current cloud platforms, including Google App Engine (GAE) [24], Microsoft Windows Azure [20] and Amazon Web Services (AWS) [45]. All these players provide scalable and fault tolerant blobstore data management. Thus, these blobstore services can work as a plugin of Cloud IO streams.

4.2.3 Columnar data abstraction

As in the previous storage service, we have also developed the same generic API upon cloud databases. This aims to create an abstraction to columnar data, for instance, SimpleDB, Azure Table and other cloud databases publicly available. Nowadays there is a new trend to store information in columnar data instead of the traditional relational system. These tables are very dynamic and the developer does not need to pre-create a model, because the structure is auto-fit and is mutable in production.

There are several problems regarding the scalability that has to be solved in this abstraction. For instance, Amazon Simple DB uses a horizontal scalability, i.e. different locations, as opposed to Azure Table which uses a vertical one. Each partition key represents a different node to have the information. This issue was solved through the Table id that identifies the Table name and the node label or location label. The idea is to contain features within the LDC (Lowest Domain Common). Two API were considered, but this approach might work with other tables.

The Java SDK already uses a high-level abstraction for databases, named JPA (Java Persistence API). Although it is widely used with Object Oriented databases, we decided to follow this standard for two main reasons: there is no other standard for database access and the proposed LDC fits the JPA abstraction. Thus, it was decided to use the same JPA methods and additionally include other methods that are more specific to Cloud databases, such as create/remove tables (Figure 4.5). Also, for representation of the results, we use a library named Guava (Google Collections), which provides very generic Java collections, for instance, the Table collection. Table is a triple values class <R, C, V> data structure, i.e. Row, Column and Value. This representation is perfect to retrieve the results of queries and also to insert new data into the columnar tables. Figure 4.5 shows the architecture of the columnar data abstraction. A select action is executed quite similarly to the JPA method. It uses a small set of the SQL and just conditions are considered. Complex queries with joins will not be considered in this abstraction since the columnar tables do not support such a feature.

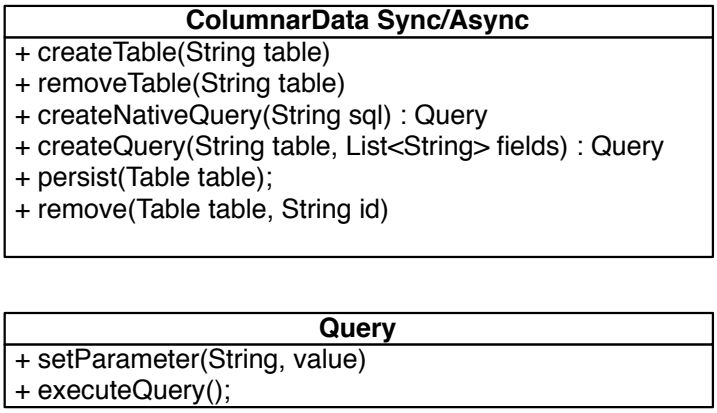


Figure 4.5 - Abstraction columnar data

An instantiation was built for the data columnar abstraction to the Amazon SimpleDB, which has a different representation of names from the JPA abstraction. For instance, each row is called Item and each item has several attributes that are not structured and can change dynamically. That is why it is called columnar data, because it can be different for each row, i.e. each record can contain different fields. Thus, in the table creation process, we do not need to define a structure as in a common database.

The SimpleDB (Figure 4.6) uses a REST provider to supply services and they retrieve XML files with the responses. So the first step was to create the XML parsers and client communication with the REST AWS interface, as described in specification [46].

The abstraction for this service follows the analogous strategy of the File Cloud Streams, and specific implementation has to be compliant with interfaces described in Figure 4.5. In the SimpleDB case, we implemented all documented functions in the abstraction. Figure 4.6 describes each method used. The AWS implementation method is private, and public methods will call the private ones, with small adaptations to cope with the same API.

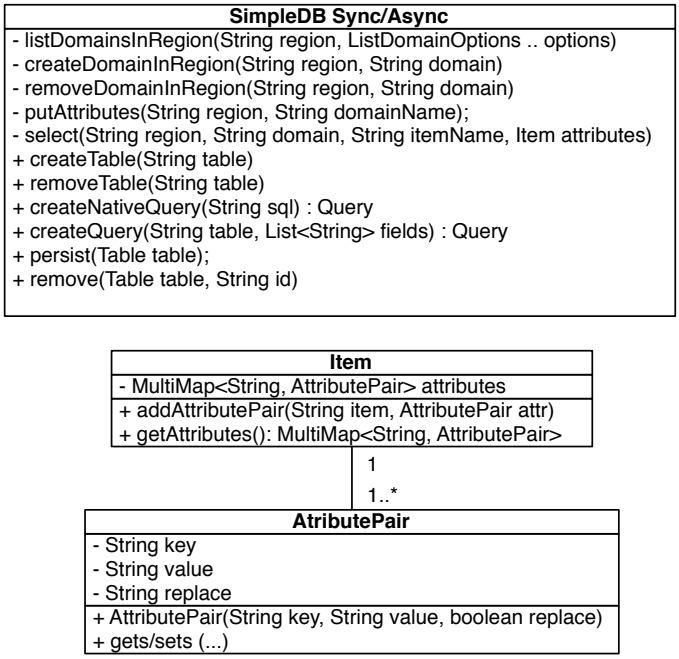


Figure 4.6 - SimpleDB instance

4.2.4 Signaling abstraction

The signal abstraction aims to dynamically create a real time system based on Publish/Subscribe mechanism. It is asynchronous, allowing the applications to communicate between each other, instantly, avoid polling strategies. However, there are not many Publish/Subscribe public services based on HTTP protocol. We will decide to use the PubNub provider [23], although other instances can be implemented using, for example, other public services such as, Channel API of Google AppEngine or other protocols like XMPP. Moreover, the polling approach can also fit the desired abstraction and, in that case, the subscriber has to poll the server until receiving a signal, and then it will call the Receiver callback. There are also other possibilities. For instance, the Ajax Push Engine (APE) [47] can be installed in a public cloud provider (e.g. Amazon EC2) and the service can be used with quite similar behaviour to PubNub. Finally, there are also other similar services based on the Publish/Subscribe pattern, for instance, the Amazon SQS and Azure Queue.

In this service abstraction, we used an Observer Pattern and in the current implementation, we created two entities: Publish and Subscribe (Figure 4.7). The channel represents the domain of each agent, and it assures that communication can only be established between agents of the same domain.

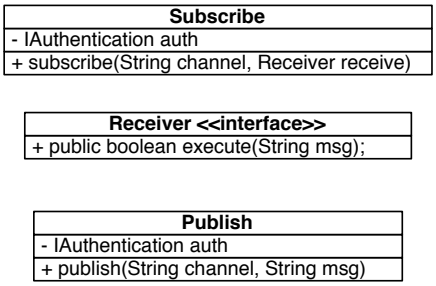


Figure 4.7 - Publish/Subscribe abstraction

We implemented the PubNub cloud provider following the abstraction proposed. So the abstraction classes will call the implementation of PubNub using the adapter pattern, similarly to the other previously presented abstractions. There are differences between the implementations, because they use different objects to communicate in the Receiver, i.e. JSON. We adapted its interface, allowing a compatible interface with the defined abstractions.

4.3 Cloud Controller

The Cloud Controller is a major component of our architecture and is responsible for functionalities such as: aggregate providers' credentials; control of access to cloud resources; managing authentication processes with gateways; and addition of new services.

This controller provides an API that can be used by third party applications to access their services. The Controller communicates through the HTTP standard protocol, using RESTful [48] specification that will facilitate access to services by other entities. The Cloud Controller contains cloud providers for different services, such as: blobstore, database and signaling (Figure 4.8). Moreover, it also supports addition of external services used by third party applications, extending in this way the Controller functionality. This platform was instanced with several end-user services associated with Medical Imaging (PACS Data Privacy, WADO, etc.), which will be addressed further on this document. In these scenarios, there is sensitive information and in this case, the developer can create a new service to Cloud Controller. This new service should be instanced in a private cloud to keep the important data safe. For instance, the Cloud Gateway can cipher the data before it sends it to the cloud, and store the keys in these private services.

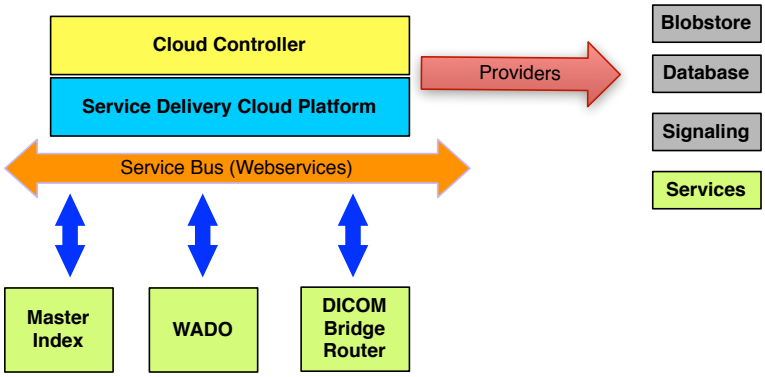


Figure 4.8 - Cloud Controller - Architecture

4.3.1 RESTful API

The interface to external applications is issued by a RESTful web service that provides several interfaces (Table 4.1), starting with an authentication mechanism. The access control is based on username and password and, if the login is valid, the web service returns a token that will be used to validate subsequent operations. Functions were created to retrieve a list of cloud providers and service information.

The Table 4.1 lists the most important methods to interact with external applications and services. The login is the routine that is called from Cloud Gateway to login in the application. It retrieves a token that will allow the agent to make operations over the Cloud Controller and allowed services.

Table 4.1 - RESTful services - Cloud Controller

Method	HTTP Operation	Short Description
/rest/login • user • pass	POST	Login with the Cloud Controller. The client will receive a token to validate operations.
/rest/provider/getList • authToken • type	GET	Get a list of cloud providers. The type can be blobstore, database, or signal.
/rest/services/get • authToken	GET	Get services that the user has access to. It includes endpoints of the service
/rest/services/validate • authToken • user	GET	Validate the token to check if it is a valid login

4.3.2 Dashboard panel

In addition to the web services API, the Cloud Controller also provides a web portal interface (Figure 4.9), where administrators can add or remove new cloud providers (storage, database, services, etc) and also check the operation's logs. This portal was implemented through GWT technologies. Also, they can create new domains, add/remove/ban agents and create new services.

This dashboard also allows the user to setup a threshold of requests or amount of data transitioned with a specific cloud provider. This is possible because all the logs of gateways cloud interactions are sent to the Cloud Controller.

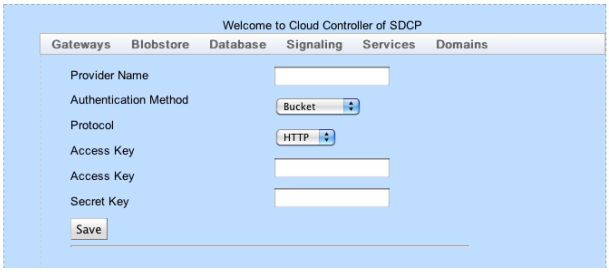


Figure 4.9 - Cloud Controller Dashboard

4.4 Cloud Gateway

The Cloud Gateway is a very important component of the architecture. Basically it is a desktop application that loads new services dynamically. It is responsible for authentication with the Cloud Controller. The plugins are added by the end-user and the application automatically loads them in the start-up. Cloud Gateway runs as a daemon in servers or desktop computers and it has a GUI that allows us to load new plugins or see operation logs.

The architecture of Cloud Gateway (Figure 4.10) also uses the SDCP-SDK. It has access to the plugin core mechanism to load new plugins. Moreover, the interfaces used in API plugins will be instantiated automatically using the Inversion of Control pattern, which will be explained further in section 4.5. The plugins to the Cloud Gateway are services programmed in Java that make direct use of SDCP-SDK. However, we offer the possibility for external applications to send information to the cloud through a web service interface. This raises a question: what is the advantage of using the web service API? Third-party application will be allowed to store, access and use databases

from multiple public clouds. Thus, third-party applications do not need to be coupled as a Cloud Gateway Java plugin.

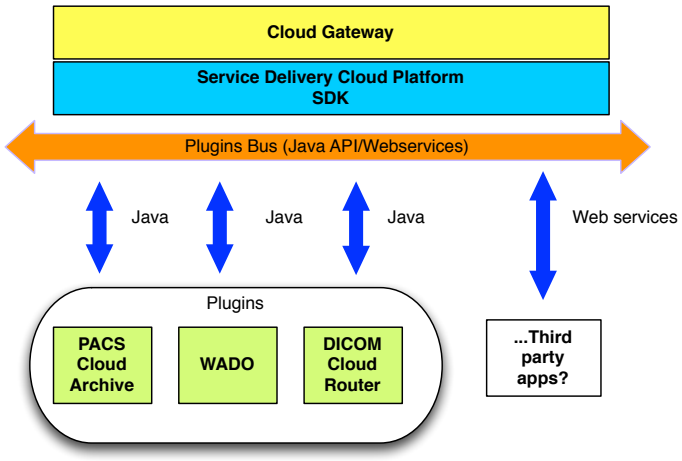


Figure 4.10 - Cloud Gateway architecture

The Cloud Gateway requires an authentication process to provide access to cloud resources. The gateway authentication (Figure 4.11) is used through the RESTful web service that accesses the Cloud Controller, previously described. When the gateway application starts, it requires a username and password for the end-user. Next, Cloud Gateway executes authentication and saves the token that will be sent to validate future operations.

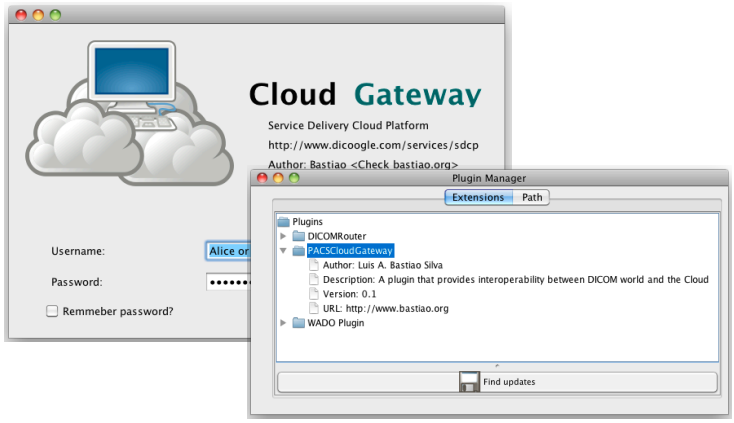


Figure 4.11 - Cloud Gateway interface

4.5 Pluggable and injection

Modular design architecture consists of dividing the system into smaller units - usually independent - that can serve different purposes inside the system. Such architecture offers several advantages, particularly in a complex system like separation of concerns. Each component is given some responsibility and flexibility of augmentation, allowing easy expansion of the system by plugging new modules without changing the main system.

However, if we want to load a web plugin module, it can be desirable to make it available also in the Cloud Controller. However, different technologies are used: JSE and J2EE-based (AppEngine). The low level layer, i.e. jar loader and class loader are common and reside in the SDCP-SDK.

A question might be raised, why implement a new plugin system? There are so many plugin systems in java, OCGi, Spring, JSP, JSPF, etc. We are aware of the many advantages of

these systems, because they support a huge amount of features. Nonetheless, these systems are highly complex and contain a huge group of dependences. Moreover, they do not fit J2EE-like properly. For this reason, it was decided to use a loader layer of a plugin framework that already exists and write a lightweight system over the JSPF [49] load layer.

The plugin system developed has two different groups: Service Plugin and Provider Plugin. There is also a sub-set of plugins, the RESTlet plugins which will be able to run in the Cloud Controller and Cloud Gateway. By definition, it will provide a RESTful web service interface. Each plugin is defined in a XML structure (Figure 4.12) that includes generic plugin information and what kind of plugin it is, i.e. service or provider.

```
<SDCPlugin>
  <url> www.dicoogle.com</url>
  <conflicts></conflicts>
  <provides>PACS Cloud Archive Gateway</provides>
  <name>PACSCloudGateway</name>
  <version>0.1</version>
  <type>service</type>
  <description>A plugin that provides interoperability between
DICOM world and the Cloud</description>
  <author>Luís Bastião </author>
  <license>GPL</license>
  <update>http://www.dicoogle.com/services/sdcp</update>
</SDCPlugin>
```

Figure 4.12: XML plugin sample

We have two types of plugins:

- Passive Plugins: Just use cloud resources, but there is no direct communication between the agents.
- Active Plugins: Use cloud resources and also available signaling services to communicate between other agents in the same domain.

4.5.1 JSE Injection

The plugin JSE injection takes place in the Cloud Gateway. So focusing on the Gateway, the first step when the Gateway starts up will be instantiation of authentication instances, provider manager interfaces, etc. Thus, when a plugin is loaded, the interfaces of the SDCP-SDK are being used.

The Gateway GUI allows us to load SDCP extensions (Figure 4.11). The user adds the directory of the plugin (Step 1 and 2 in Figure 4.13), which should contain an XML file (sdcp.xml) and the dependence libraries in a sub-directory. The class loader will load all the dependences (Step 3), and then the plugin itself, i.e. the jar in the root directory. The plugin is recorded (Step 4) and after that, the Gateway will support the services or providers that are inside the plugin.

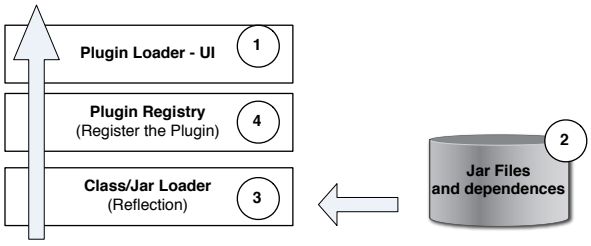


Figure 4.13 - JSE Plugin Injection steps

In order for the plugin implementation to work properly, in the start-up the injection of SDCP-SDK interfaces is executed. The plugin has to use @InjectPlugin annotation, which is native to JSPF; thereby the interface will be automatically instantiated in runtime. We inject 3 interfaces at start-up: IAgentManager, IProviderManager and IServiceManager. These three java interfaces are injected with the REST class implementation which uses @PluginImplementation annotation. Thus, the REST communication with Cloud Controller is specified in the implementation classes in the Cloud Gateway, although the plugin just refers to the interface.

4.5.2 Application Server/J2EE Injection

The plugin injection into the application server is quite different from a desktop application. At the first stage, all the plugins have to be registered in the WEBINF/sdcPlugins.xml.

The Figure 4.14 presents the plugin injection workflow. As soon as the application server is instantiated, the registered servlets will start automatically. It loads the xml (Step 1) and checks the plugins that are available to load. The Plugin Loader will load the plugins located under WEBINF directory and inject the instantiations. An interception of /rest/plugins/* URLs was created to fetch and register into the RESTlet router (in the step 2) . Thus, the RestletServlet has an Init method (step 5) that will be executed in the application server start-up action. It loads the respective RESTful resources and the correspondent plugin that will take account of the resource (step 3 and step 5).

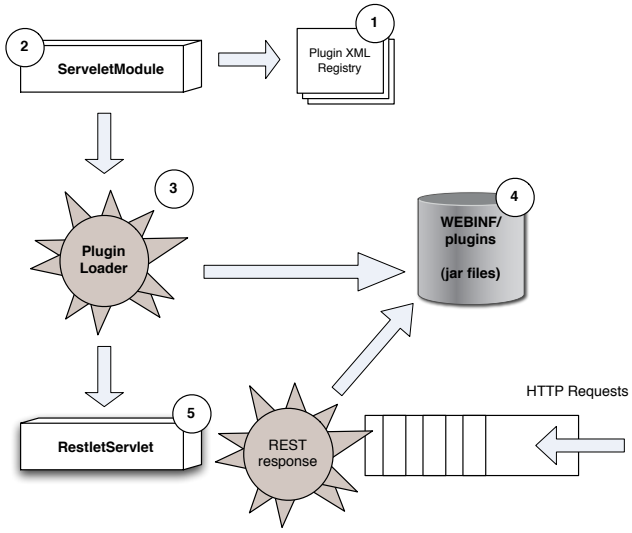


Figure 4.14 - J2EE injection

The plugins annotated with @RESTletPlugin("resourceName") will be attached in the RESTlet router, with the respectively annotated resource name (step 5). Finally, for each HTTP request received, the RESTlet router will forward to the correct resource previously loaded.

4.6 Service API

Service Plugin

A Service is the plugin that runs a service such as daemon, and it is often used to make interconnection with existent protocols to access cloud resources. File system storage is a good example of this. Thus, the Java Plugin has to use three annotations: @Init, @Start, @Stop. The init will setup initial configurations. Regarding the start/stop annotations, it is used to start and stop services.

These annotations are very important for the plugins of the Cloud Gateway because that class should contain all of them, in order to setup the initial settings of the plugin. For instance, to load configurations and to start/stop the service that the plugin is going to bind/listen to.

Providers' Plugins

An extensible architecture was created to support more than one Cloud Player. For instance, for the database service were created all abstractions and also the instances, e.g. SimpleDB. However, developers can also add support for other players. In such cases, they have to implement the specific interfaces and the classes should be annotated with one of: @Blobstore, @Signal or @Database. These plugins is registered in the correct abstraction, in order to locate the correct class for the defined service.

Moreover, if the data has to be ciphered, the developer should annotate the writing method with @Cipher annotation, and it will apply cipher functions before uploading it to the cloud.

Finally, these plugins will be used by the abstractions, and so if using Cloud Streams, Data Columnar or Signaling, the loaded provider plugins will be available.

RESTful Plugin

The RESTful plugin is quite similar to a service plugin, because it also supplies a service. However, only one method has to be provided, which is how to get resources automatically attached in the RESTlet server. Additionally, the resource classes can be annotated with RESTletPlugin("resourceName"), as described previously.

Web services API

The web services API is provided to access cloud storage and database in a transparent way. Therefore, end users are not concerned with the different APIs and the application will be interoperable. The implementation developed uses the abstraction provided in the previous sections, through the RESTful interface.

4.7 Application in real use cases/scenarios

In the literature we found several use case scenarios that will benefit from the proposed solution. We describe each one in more detail in the next sub-items.

Storage file systems

With this platform we can more easily write a file system located in different cloud providers. Systems quite similar to Dropbox, which sends the files to the Cloud and synchronizes between users. For instance, Dropbox uses just Amazon S3, but with this platform the solution could be instanced in other cloud players very easily.

In addition, this approach allows creation of an integrated file storage that enables visualization of a federated view of files and sharing of files between multiple cloud providers, e.g. share a file from a Dropbox user to a SkyDrive user.

Accounting and Invoicing Systems

All industries are obliged to use invoice systems that must always be online. There is a new trend to deploy them to the cloud. This is a very rich scenario for use of the proposed platform, because offline systems can outsource databases and the storage system from the cloud and take advantage of the redundancy between different cloud providers. Moreover, those systems can be developed to respect an abstract services interface, without concerns about distinct cloud providers' specifications. So industry administrators can freely select cloud provider and configure their

invoice system installation to work with it, an evident commercial advantage of the invoice system companies.

Medical Imaging Scenario

The Medical Imaging scenario was already discussed in the previous chapter of this thesis. The SDCP was built to support this development. Medical institutions have to store a large number of medical studies/images, i.e. DICOM object files. Thus, they need to have large datacentres inside the hospital, a major source of problems for systems administrators. The Cloud Computing model fits this scenario. Moreover, healthcare is a critical service and the information must always be available. Therefore, the approach described in this chapter is crucial to create solutions that can be instanced in more than one cloud provider.

4.8 Solution benefits

The proposal presented was designed to support several use cases with a practical impact in many industries. The various modules developed provide interoperability with several cloud providers and present a rich set of characteristics over them:

On-premise applications using Cloud

Traditional applications use resources such as databases and storage located in the enterprise datacentre. There is a new trend to move this information to the cloud infrastructure. Thus, this architecture supports a workflow to simplify data storage in the cloud, without many changes in the enterprise infrastructure. Following this approach, several hardware and software licenses are no longer needed. This will reduce costs for the enterprise, and will also be more energy efficient.

Support multiple cloud vendors

This platform is independent of the cloud vendor. Data can be easily sent and accessed in multiple cloud players at the same time. The use of a common interface should be adopted by services and it will be a contribution to Cloud resource standardization.

Low-coupling

The proposed architecture takes advantage of the capabilities and features of the cloud providers to enable service delivery. The non-existence of dependencies between services and applications is a key point for the health and success of this architecture. It follows the principle of Service Oriented Architecture (SOA), which states that services should work without any knowledge of the service location or any other form of address [50]. This architecture will permit access to the resource automatically and dynamically.

Automatic Encryption/Decryption

At present, most cloud solutions do not offer an option to encrypt data when it is uploaded to the cloud. Some companies are already offering this service but we do not know if the cloud cannot be jeopardized, giving access to data. This platform has an encryption/decryption layer on the client side, i.e. the cipher and decipher operations are executed on-the-fly in the enterprise side.

Applications can work any time and anywhere

The presented platform only uses HTTP/HTTPS services. In fact, it was our main concern to create a platform that just requires a common Internet protocol to communicate with the cloud, i.e. “well-behaved” for firewalls.

Cost Management

This framework can lock access to cloud resources when a certain amount of requests/data is reached. The system can also notify the application manager, if it exceeds a defined threshold. It is a useful functionality because some Cloud Providers, for instance, the Amazon AWS, do not allow use of the cloud to be blocked, i.e. stop the use of the resources.

Minimizing the risk of failure

Using cloud computing, the risk of incidents is shifted because cloud providers have the data in multiple locations. However, there is another risk that developers have to consider: What happens if cloud-computing providers stop supplying their services? Such a situation will certainly harm cloud clients. The proposed approach will greatly minimize these risks, because the data can be redundantly stored in multiple cloud providers, without impact on SDCP API client applications. Moreover, the architecture allows to automatically forward the resource to another provider, if a cloud provider fails.

4.9 Conclusion

In this chapter we presented the Service Delivery Cloud Platform, a cloud middleware infrastructure that provides a set of services using resources from multiple cloud providers. The presented architecture allows decoupling the specificities of each cloud provider API into a unique abstraction. Throughout this chapter we have extensively detailed the system architecture, discussed its main advantages and provided a set of benefits obtained from its use in a critical scenario – PACS.

5 PACS Cloud Archive

"Science progresses best when observations force us to alter our preconceptions"
Vera Rubin

We defined the requirements of the project previously (chapter 3). Also, we explained a generic platform for delivering services over multi-vendor cloud resources that was implemented. In this chapter we will introduce the PACS Cloud archive, discussing the architecture, application design and how the component of this application service fits in the Service Delivery Cloud Platform (SDCP).

5.1 Architecture

A common PACS archive has two major components: the DICOM object repository and the database system. The object repository typically demands an infrastructure with huge storage capacity to support all studies. The second module is normally a relational database management system (RDBMS) that supports the *DICOM Information Model (*DIM) [51], containing mandatory metadata related to patients, studies, series and images. When a medical image archive receives studies from imaging modality equipment, it is necessary to store the images in the file system repository and to update the database with data elements extracted from the received study headers.

Our approach is based on outsourcing these two components to the cloud, namely using the new concepts of blobstore and database accessible through web services. This solution is aimed to work in any public cloud player, i.e. support more than one provider. At present, cloud providers are offering separated storage and database services, but the access interface does not follow any standard. We proposed a solution that could work with several providers, even at the same time, in previously chapter 4. Fundamentally, instead of coupling a solution to just one provider, an access abstraction layer to blobstore and database was defined. Moreover, additional Cloud provider support is simplified due to a plugin-based system presented in Service Delivery Cloud Platform (chapter 4). To support abstraction with the cloud blobstores we used the Cloud IO (Input/Output) stream mechanism of SDCP. It allows writing in the cloud blobstore as a data stream.

5.1.1 Components

The proposed PACS Cloud architecture [52] was devised to eliminate or reduce typical cloud weaknesses referred to above. Particular emphasis was given to the separation of private data element manipulation from demanding computational operations. The developed solution is based on 3 main components: PACS Cloud Gateway [53], Master Index, and Cloud Slaves (Figure 5.1).

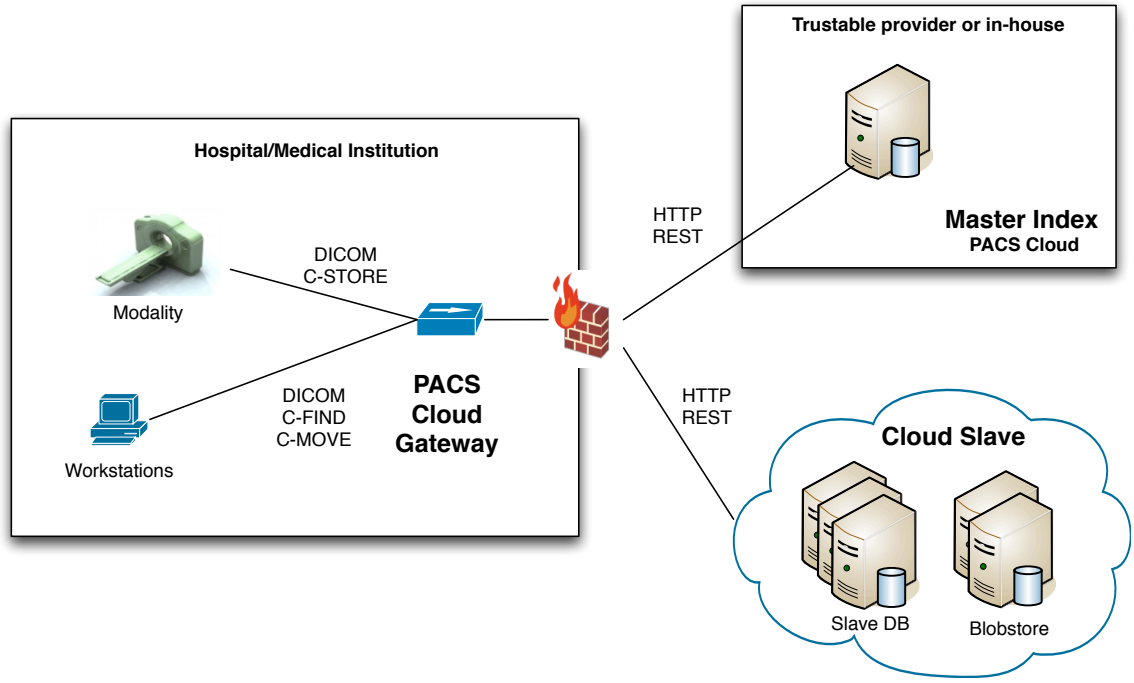


Figure 5.1: Architecture of the PACS Cloud – it includes one Master Index, one PACS Cloud Gateway in each institution and one or more Cloud Slaves.

Master Index

The Master Index is the PACS Cloud core entity because it contains the confidential data. All DIM identifiable information related to patient and study levels is stored in the Master Index database, e.g. patient name, referring physician, etc. Master Index is a fundamental component to ensure the solution's confidentiality and privacy. It also provides authentication services for institutions' gateways able to validate logins of each entity PACS Cloud Gateway and their respective domain/trustable group. It uses the accounts of SDCP, which also list of Cloud providers (repository and database) that are stored over the Cloud Controller.

Master Index implementation was developed as a plugin of Cloud Controller (presented in section 4.3). Thus, it benefits of the domains and users that Cloud Controller already supports. Due to the privacy parameters of this scenario, the Master Index should be deployed inside the institution or in trustable provider, but we will discuss privacy details further in this chapter.

Cloud Slaves

Cloud Slaves are cloud providers performing two different services: blobstore and database. The Cloud Slaves supply, on the one hand, storage of blinded data (encrypted DICOM objects) and, on the other hand, a database containing anonymous attributes extracted from DICOM studies. Cloud Slaves, composed of two independent modules (repository and database), are responsible for the most demanding storage and computational task in the solution. The database provides search services over DIM fields, but the private metadata, e.g. patient name, is not visible, and it has to be requested from the Master Index. The developed architecture allows having one or more slaves hosted in different Cloud providers, i.e. it is possible, for instance, to have a slave database and two slave repositories in three distinct Cloud providers.

PACS Cloud Gateway

The PACS Cloud Gateway provides the interface between the DICOM universe and Cloud systems. Internet Cloud providers only support Web Services API (Application Programming Interface), which is a barrier to PACS server implementation through Cloud computing. In order to solve this problem, we developed this DICOM-to-Web-Services gateway. It aims to translate DICOM commands into Web Service (WS) requests granting interoperability between DICOM devices and cloud interfaces. The Gateway provides two DICOM services: Storage (DICOM C-STORE) and Query/Retrieve (DICOM C-FIND and C-MOVE). This component is located in each institution and needs to be registered in the Master Index to access the PACS Cloud services. When these PACS Cloud Gateways is loaded in Cloud Gateway, they will authenticate with the Master Index, launch DICOM services and then, in a steady state, wait for DICOM requests from intranet PACS nodes. More details about the architecture of this module see appendix 10.1.1.

5.1.2 Frameworks and Technologies

The PACS Cloud interfaces were specified in a developed PACS Cloud Framework. As a consequence the PACS Cloud Gateway interfaces were bounded in the framework (Figure 5.2). The Framework outlines the DICOM messages in XML and API to Master Index. All components of PACS Cloud were implemented in Java, but it can be implemented in another language, since assert the PACS Cloud Framework specifications.

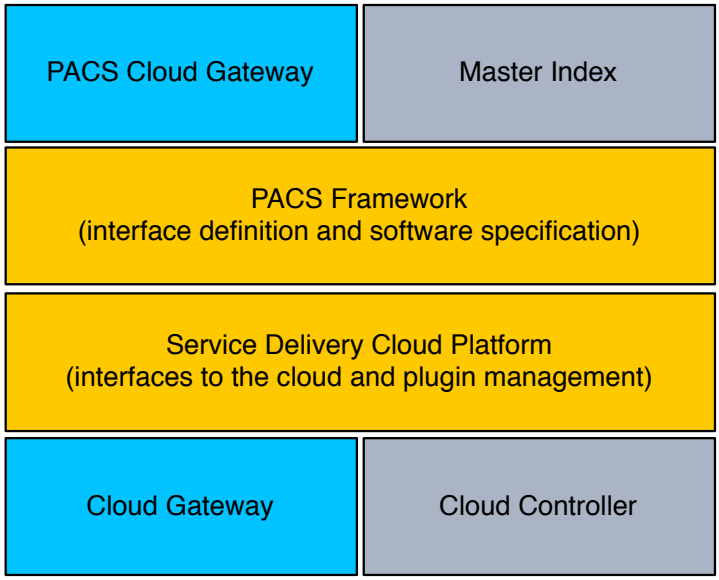


Figure 5.2: Shared framework between PACS Cloud Gateway and PACS Cloud Master Index

The various components of this PACS Cloud archive were developed in Java, namely the Master Index, the access to slave database, the access to slave repository and the PACS Cloud Gateway (plugin to SDCP). The PACS Cloud Gateway provides two distinct interfaces since it is responsible for establishing the connection between DICOM devices and Cloud providers. The PACS Cloud Gateway is a plugin of Cloud Gateway and Master Index a plugin of Cloud Controller. Both plugins are also designed over SDCP that has the interface to the cloud and plugin manager.

The implementation of DICOM functionalities is supported by the DCM4CHE library [54], a Software Development Toolkit (SDK) that is used to extract DICOM data elements from persistent objects and to implement the Storage SCP and Query/Retrieve SCP services. The PACS

Cloud Gateway receives DICOM images and forwards these images to the slave blobstore in a specified cloud provider. Besides the DICOM-to-Cloud interface, the PACS Cloud Gateway also extracts and stores DICOM attributes in several databases: Master Index and slave databases. The communication with Master Index relies on RESTful web services, through the RESTlet framework [55]. The Master Index stores data in a JPA (Java Persistence API) object-oriented database. It also provides a web interface developed with GWT (Google Web Toolkit) framework, which is used to manage the PACS Cloud archive.

5.2 Workflows and dataflows

The architecture was designed to support cloud repositories shared by different institutions, for instance, a regional PACS or a network of small imaging centers. It is possible to support that *modus operandi* because several PACS Cloud Gateways can be registered in the same Master Index. Moreover, the solution permits C-MOVE operations between different institutions, i.e. Hospital A workstation can send a C-MOVE command to PACS Cloud Gateway A with a destination node address located in Hospital B (Figure 5.3). For instance, a modality in Hospital A stores 2 studies in the PACS Cloud archive. Later, a workstation in Hospital A needs to move one study, stored in PACS Cloud, to Hospital B and sends a C-MOVE to PACS Cloud Gateway A. Because all Gateways share AE title tables, PACS Cloud Gateway B knows it is its responsibility to process that request. So PACS Cloud Gateway B requests image data from PACS Cloud and sends it (via C-STORE command) to LAN B host (Figure 5.3).

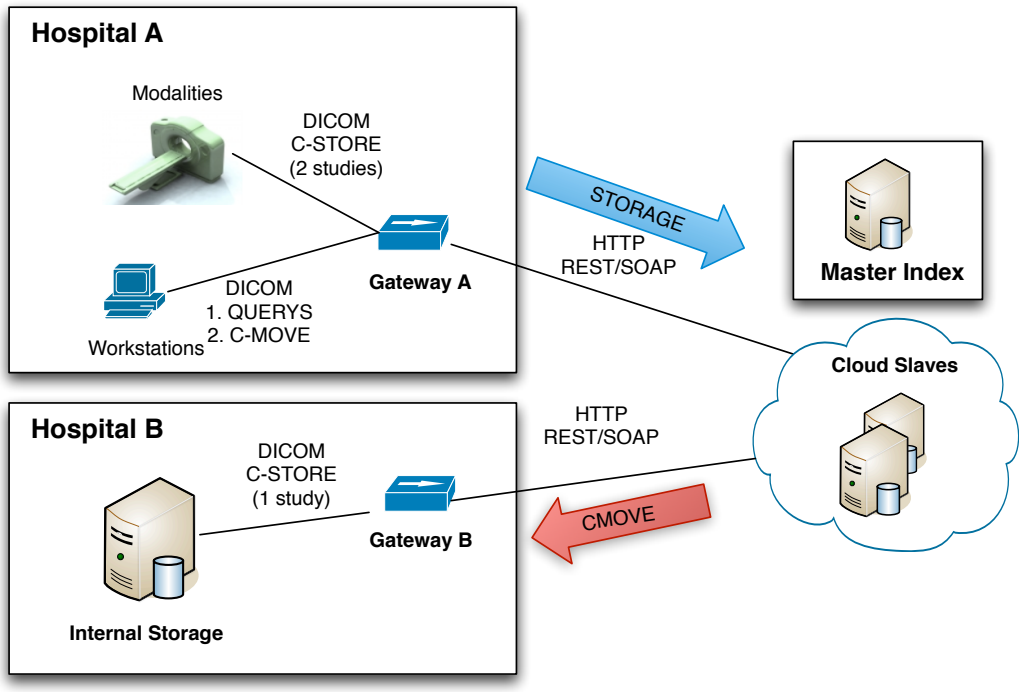


Figure 5.3: An illustrative example of the usage of the PACS Cloud across two medical centers.

To manage the PACS Cloud a web interface was developed that interacts with the Master Index. With this tool it is possible to register Gateways, permit or deny DICOM SOPs Classes (Service Object Pairs), and configure DICOM nodes and respective permissions (IP, Port and Application Entity Title).

In next subsections workflow of Storage and Query/Retrieve will be briefly described. For more details about the workflow see appendix 10.1.2.

5.2.1 Storage Process

The PACS Cloud Gateway is the entity that is responsible, in each DICOM domain, for establishing the transparent interface with the PACS Cloud archive. The storage workflow is as follows, as described in Figure 5.4. When a DICOM C-STORE command [56] arrives at the PACS Cloud Gateway, the received objects (images, SR, waveforms, etc) are temporarily stored in this element. Each object is split in chunks, i.e. data blocks with normalized size and encrypted with AES (Advanced Encryption Standard) algorithm (Figure 5.4). The encrypted chunks are then sent to the slave repository (blobstore). The metadata necessary to build DIM is extracted from the persistent objects (which use TLV structure), the sensitive fields are ciphered through a hash function (SHA-1) and the result is sent to the slave database, according to data privacy of DICOM Security Profiles [57]. The patient name, the list of pairs [study, instance UID] and the encryption session key are securely transmitted and stored in the Master Index. The DIM database also stores information (e.g. references) about object instance location (Figure 5.4).

The identification, access credentials and location of Cloud Slave providers, database and repository entities, are obtained from the Cloud Controller entity. It is transparent to the PACS Cloud Gateway plugin because it uses the interfaces defined in the SDCP.

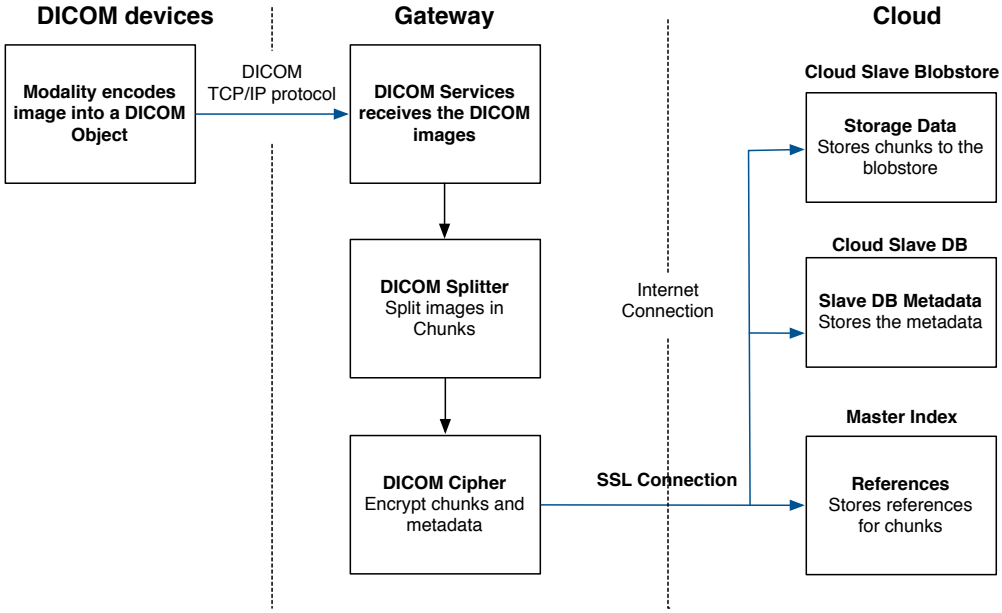


Figure 5.4: PACS Cloud Storage Workflow – each DICOM device sends the study directly to the PACS Cloud Gateway that is responsible for storing it on the Cloud Slave.

5.2.2 Query and Retrieve

The Query/Retrieve process is subdivided in two independent DICOM commands: C-FIND and C-MOVE, where the C-FIND is associated with queries and C-MOVE with image retrieving [56]. The C-FIND request contains query items created using logical expressions over DIM fields. Wildcards can also be combined in these expressions, for instance, in the patient's name. According to the scope of the query, it can be executed directly in the slave database without consulting the Master Index. If the patient's name is referred to in the query items or if they contain a wildcard, there will be a first query to the Master Index (Figure 5.5). This is necessary to identify which patient names match the query before inquiring the DIM slave database about those names

(their hash). Subsequently, the PACS Cloud Gateway will collect all data and generate a C-FIND response with the matched results.

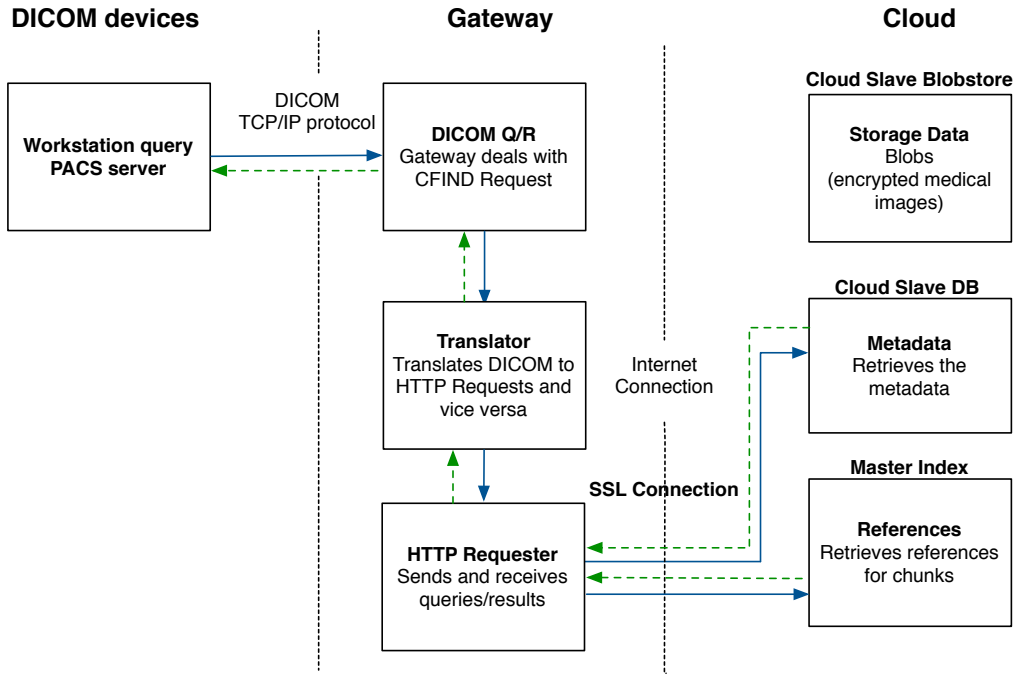


Figure 5.5: PACS Cloud Query workflow - the workstation sends a query to the PACS Cloud Gateway that inquires the Cloud database and the Master Index. The collected results are then combined and returned to the workstation.

In the retrieval process (Figure 5.6), when a C-MOVE request is received by a PACS Cloud Gateway, the DICOM destination node address (Application Entity Title) and the query level are extracted from the command data [56]. Based on a specified query level, a specific query is executed in the Cloud Slave database. For instance, the "*study instance uid"* field is used to retrieve all objects (images, SR, etc) from a study, the "*series instance uid"* to retrieve all objects from a series and the "*sop instance uid"* to retrieve a specific DICOM object (e.g. images). The database query's result is a list of "*sop instance uids"* and respective repository location, e.g. the blobstore provider where each object lives. The credential to access the respective slave blobstore is retrieved from the Cloud Controller and the download process starts, using multithreading and multiple objects. On the other side, the PACS Cloud Gateway starts a DICOM C-STORE process with negotiation with the local storage repository. Each object received from the blobstore is decrypted with its study session key and a DICOM C-STORE is issued. After all objects are transferred to the local repository, the cloud association is closed by the PACS Cloud Gateway and a C-MOVE response is sent to the C-MOVE requester entity.

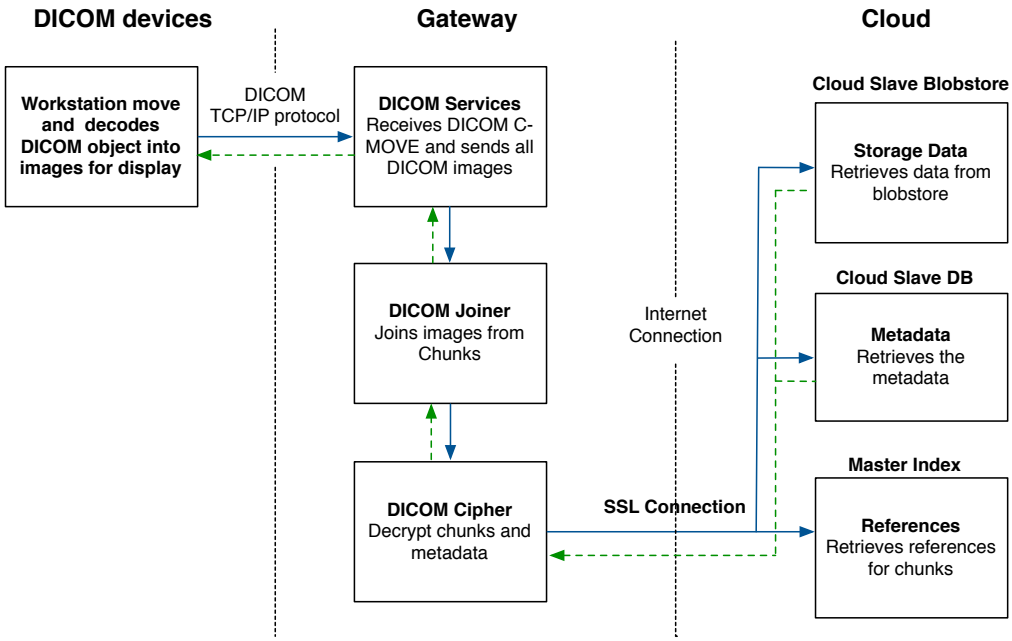


Figure 5.6: PACS Cloud retrieve workflow – the PACS Cloud Gateway downloads images from the cloud and sends them to workstations or local storage.

5.3 Improvements

In the previous items, we presented a PACS Cloud archive that, by itself, is a valuable software solution because it allows outsourcing the resources of a local PACS to the cloud without any compatible break in the medical workflow. However, it is expected that a PACS archive has a good performance, namely the retrieve times must be faster. In this section we will explore the optimization of PACS Cloud archive processes, in order to make a solution feasible to deploy in real environment.

A critical question may be raised: how do you have a better retrieve times if the data is outsourced in the cloud? The answer for this question is not trivial and we suggested two methods to tackle the problem. The first one, focuses on decrease the amount of data transferred from/to the Cloud. The second approach consists in the introduction of a cache mechanism, i.e. keep the studies with high probability to be requested in local storage. Moreover, these two approaches can work together, in order to get a better performance in the PACS Cloud archive.

5.3.1 Compress DICOM objects

To reduce the amount of data transferred from/to the cloud, firstly we need to reduce the amount of information. There are several compression techniques in DICOM like, for instance, convert to another transfer syntax. Nevertheless, these techniques might have significant loss comparing with original image. It is not desirable in the medical studies. Thus, we adopted simple file compression/uncompressing, following the well-know zip algorithm.

The zip process cannot be applied to every DICOM object. Firstly, during the storage process the PACS Cloud Gateway has to check which transfer syntax that the study are encoded. By default, we exclude just one transfer syntax (1.2.840.10008.1.2.4.50-1.2.840.10008.1.2.4.64), which corresponded on lossy jpeg. In this transfer syntax the compression does not have any positive effect and, in many cases, it can increase the size of the DICOM image. We also added an option in the parameterization mechanism, which allows the user to exclude a list of transfer syntax from the zip process.

To implement this process in the PACS Cloud archive, it was introduced another module in the pipeline of PACS Cloud Gateway. This module applies zip algorithm to the DICOM objects, using Zip Java streams[4] that permits us to zip or unzip a file or a set of files. The usage of compression process is optional, meaning that a PACS Cloud Gateway can deliver their repository in ZIP format or not. This process will be integrated in two distinguish processes of PACS Cloud archive: storage and retrieve.

The storage is a crucial part of that procedure, the compression starts in the store of DICOM objects. The retrieve in zip format is possible if the DICOM object was stored in zip format previously. So, the PACS Cloud Gateway receives a DICOM object and applies the zip algorithm (Figure 5.7 - left hand side). Then, the pipeline flows the normal process, as described previously in the PACS Cloud archive workflows (section 5.2), but with a slight difference, the database contains a compression flag for each object stored. On the other side, the retrieve process is quite similar, although the pipeline is performed in the reverse way. The file is decrypted and it joins the block (Figure 5.7 - right hand side). After that process, the unzip algorithm is applied and the images are sent to workstation using the traditional DICOM process.

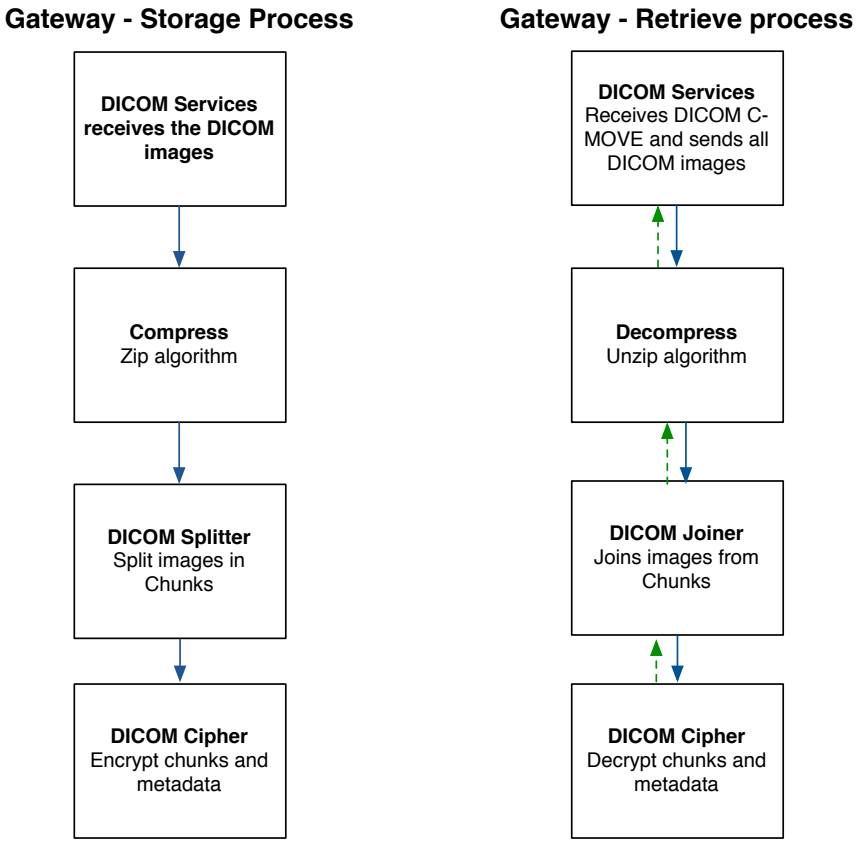


Figure 5.7: PACS Cloud Gateway: storage and retrieve workflow with compression

5.3.2 Cache

Due to latency associated to service access and communications with public cloud providers, the retrieve process can be slower. This process is extremely important in the overall quality of solution because it exists a real-time interaction with end-users, i.e. the professional is in front the computer waiting by images. In order to reduce latency in data transmission and to

[4] http://download.oracle.com/javase/1.4.2/docs/api/java/util/zip/package-summary.html

decrease the computational costs of repeatedly performing decompress and decrypt operations, a caching mechanism was implemented in PACS Cloud Gateway. This mechanism is a local storage area that temporarily stores studies with a high probability to be request in the future operations.

Moreover, introducing a cache mechanism into PACS Cloud Gateway brings reliance, availability and trust to solution because gateway will be able to answer a significant number of DICOM requests, even when the ISP (Internet Service Provider) or cloud slaves are offline. As expressed, the PACS Cloud Gateway deals with storage and database management. Thus, to provide all identified DICOM services this entity has to cache both information resources. It has been implemented 3 caches modes: storage archive, database and storage requests.

A cache in the PACS Cloud Gateway operates like a traditional PACS, working with local archive. However, only a subset of the images is stored in the cache in a defined time period (for instance, 90 days).

Cloud database access times are not relevant, when compared with the storage retrieve time. However, to create a robustness solution, with full offline mode capability, a database with the metadata is stored locally in each gateway. The DICOM standard recommends the creating of a small index database called DICOMDIR for each directory that contains DICOM files. The DICOMDIR is an archive that stores all DIM fields and point to the images directories (as introduced previously in the section 2.1.2). The main idea is to reduce the access time, avoiding open all DICOM files in directories. Every DICOM file saved in local repository is also added to the DICOMDIR. The use of DICOMDIR is a good solution because has a hierarchical DIM model, enabling an easier way to create DICOM C-FIND responses in Query/Retrieve processes.

The file archive is the main motivation to create a cache implementation because the storage and retrieve times are slower comparing with database. On one hand, it decreases the times in the retrieve process performance and, on the other hand, the solution can deal with ISP or cloud slaves failures because the cache can provide local answer to requested DICOM objects. In Figure 5.8 we showed the optimized retrieve process of the PACS Cloud archive workflow. If the resource is presented in the cache, the Master Index is not consulted and the DICOM object is not transferred from the cloud slaves to the PACS Cloud Gateway, thereby reducing latency. Depending on the cache temporal capacity and previous requests, a study may be present on cache, but not with all series or images. In such cases, just the missing items are requested to the cloud slave blobstores and the Master Index is consulted to get the key to decrypt those items.

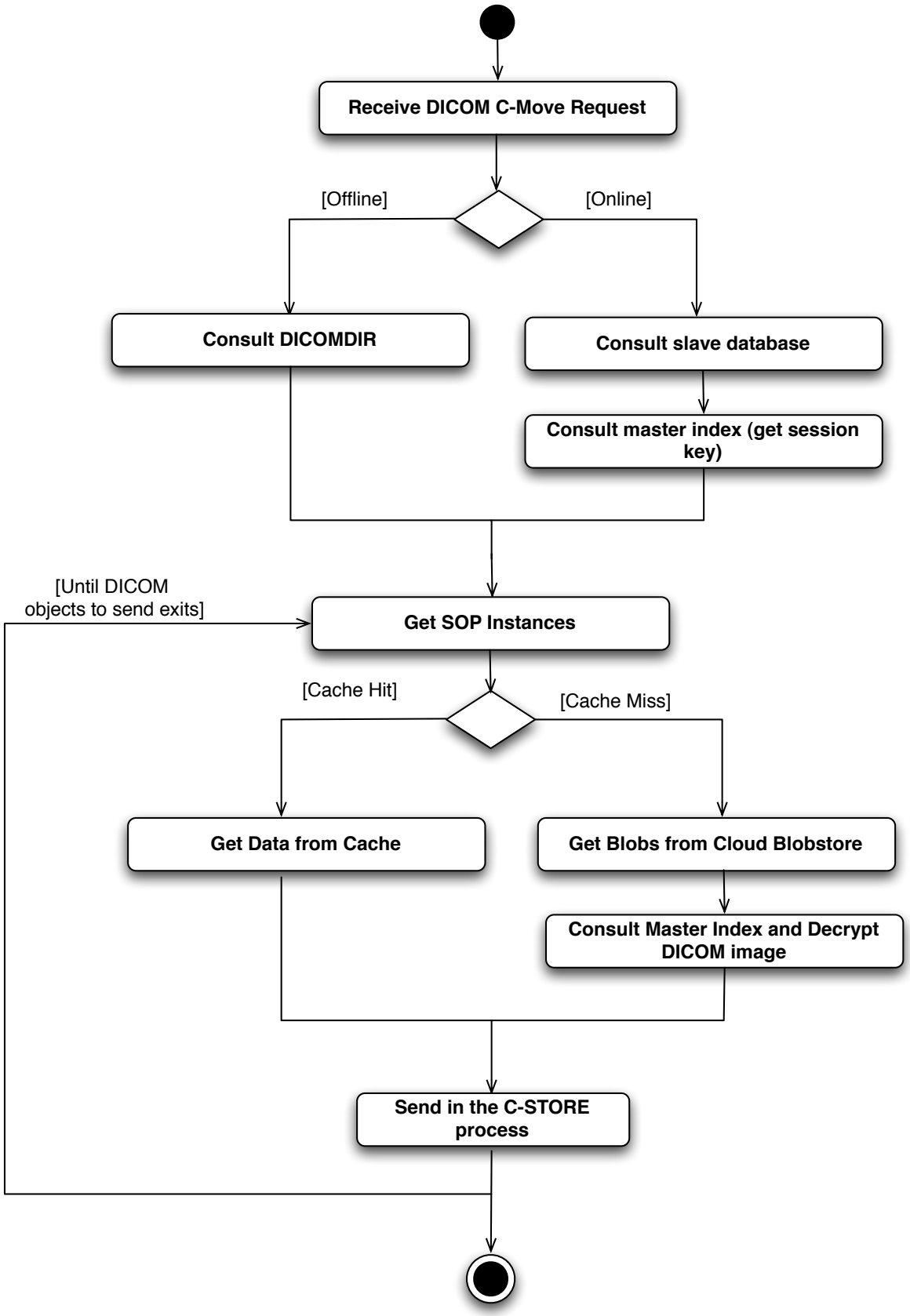


Figure 5.8 - Activity diagram - Cache mechanism in retrieve process

The PACS Cloud Gateway can deal with C-STORE requests even when it works in offline mode, i.e. there is no Internet connection or cloud provider is unavailable. Firstly, the requests are stored in local cache, i.e. the DICOM images are stored in local file system and the metadata is also stored. Secondly, the requests are scheduled for a queue and, when the cloud service is available, the PACS Cloud Gateway enters in online mode and the upload processes starts polling the requests from the queue.

Finally, there is an important issue to keep the cache process updated. It is necessary to execute a routine to clean the old files that are out of the cache scope. The cache can follow two different approaches: limit the amount of studies by date and limit the cache by amount of data. We have an independent thread to calculate the amount of data in the cache and also to check if the files are outdate. These files are removed from cache to avoid cache resizing.

The cache mechanism can be setup by the PACS manager to a defined period depending on the medical institution dimension. Moreover, being implemented as a standalone component, the caching policy is orthogonal and transparent to client side, i.e. end-users applications. In this scenario, the cache is a very useful functionality.

5.4 WADO

The Web Access to DICOM Persistent Objects (WADO) service is a normalizing web extension to DICOM protocol. WADO is provided through the HTTP protocol and the user performs requests based on UIDs of DICOM objects. This extension is a retrieve DICOM method, similar to C-Move that was already explored in previous section (5.2.2).

Following the market guidelines, it was decided to develop a standard WADO interface to PACS Cloud Archive, thus other web applications in medical informatics can benefit from it. The WADO was implemented as a SDCP plugin, namely a web module. We fulfilled the web services based on RESTful concept.

WADO supports several ways to retrieve DICOM images from the service. For instance, the application can access to the all data of DICOM object, for instance, metadata and pixel data. Or on the other side, a web medical viewer can want to access to the images in a web standard format, for instance, JPEG. Besides, we started by defining the mime types to support. Two of them were considered: “application/dicom” (i.e. integral DICOM object) and “image/jpeg” (i.e. pixel data converted to JPEG).

Firstly, for each HTTP request, the restlet core system forwards the request to the WADO resource. The request is based on the objectUID, seriesUID and studyIUD, as shown in Figure 5.9. Secondly, it is necessary to check the mime type. If it is supported, the process will proceed to the next step, otherwise it returns “Not acceptable” error. Afterwards, the WADO plugin will request the DICOM object to the cloud slaves. It resorts to the methods provided by SDCP (Cloud IO streams) in order to get the data from cloud blobstore. Also, it will request to the Master Index the correct key to decipher the content of the DICOM object based on the study instance uid, which is a mandatory field in the parameters. After receiving both responses, the data is decrypted with the key requested to the Master Index and it proceeds to the retrieve of DICOM object. Thus, depending on the mime type that it requests, it will retrieve the content. If the mime-type corresponds to “application/dicom”, the WADO resource writes these bytes to the response stream. Otherwise, it is necessary to convert the DICOM object to JPEG. This conversion was inspired on the code of Dicoogle, which uses JPEGImageEncoder from Java and imageio from dcm4che2. This feature is disabled when the plugin is deployed on AppEngine due to platform limitations.

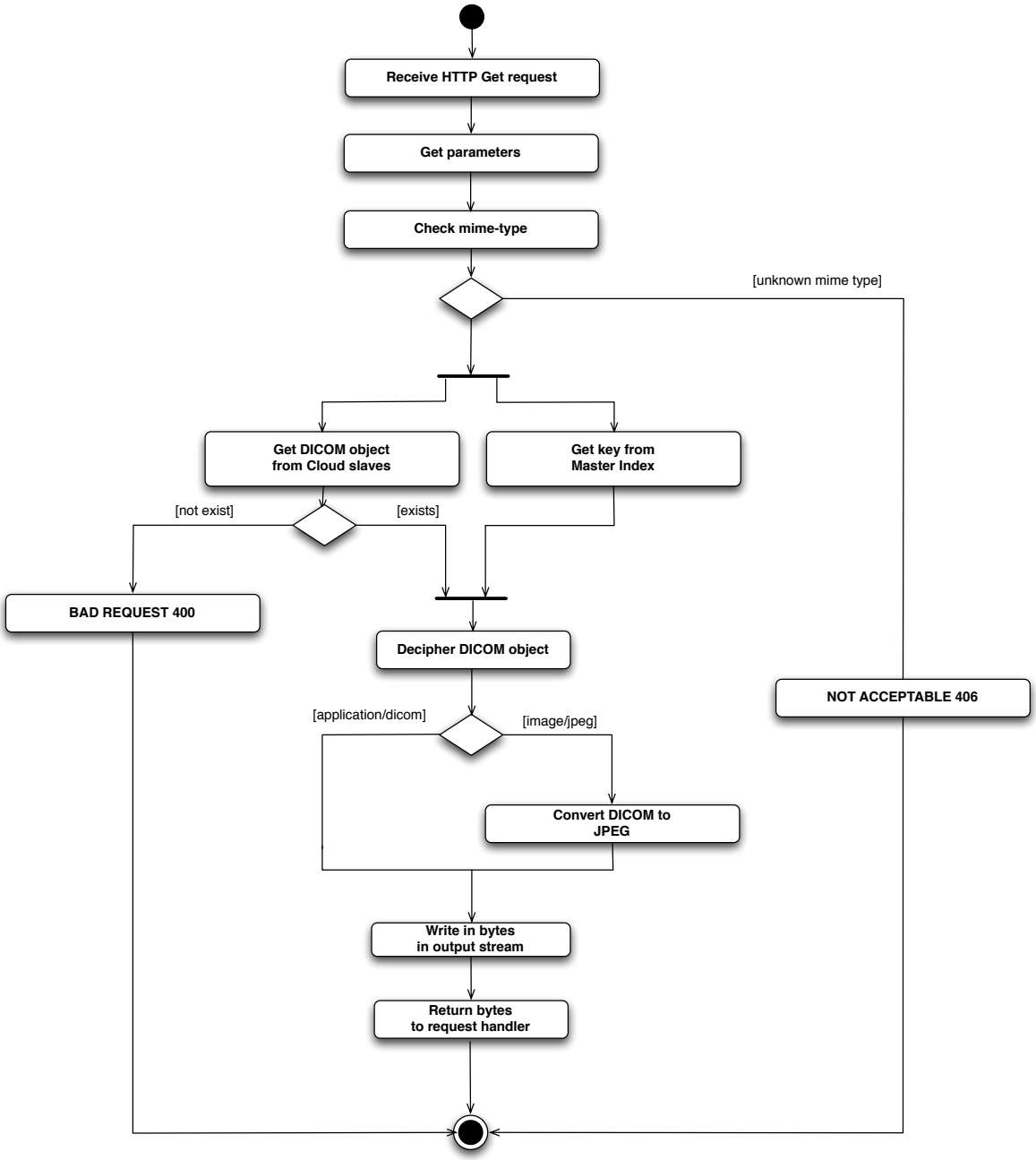


Figure 5.9: WADO activity diagram

We considered that WADO plugin can bring several benefits to web cloud applications because it supplies the medical images through the web. It is very interesting to integrate with other information systems (HIS) that already exist in the medical institution.

5.5 Security Model

While data outsourcing to the cloud alleviates institutions from the burden of local storage and maintenance, it also brings new security concerns. There are doubts about data integrity, maintaining privacy, reliability and unauthorized access to medical imaging studies in the cloud. Nevertheless, this chapter introduced some features to overcome such challenges.

5.5.1 Privacy and confidentiality

In this proposal, transferring sensitive patient data to a hosted service does not necessary mean losing control of data. Thus, all interactions between PACS Cloud [58] modules are securely protected through SSL (Secure Sockets Layer) channels, ensuring privacy and confidentiality of communication. However, the SSL by itself does not guarantee any privacy when we consider the cloud players as possible users with unauthorized access.

Our proposal to solve this issue was the introduction of a third party entity, the Master Index that is the core element of privacy in the proposed architecture. Since protected access is crucial, it must be under a trustable provider or located inside the end-user institution. The proposed system ciphers all studies and associated sensitive attributes before sending it to the cloud. PACS Cloud Gateway performs this process, which is also responsible for sending the secret keys and confidentiality data to the Master Index, although it is just a small part of information compared with the slaves database/repository. The huge amount of medical information is stored under cloud providers granting patient privacy.

The Master Index stores sensitive information, which includes patient names, *study instances uid,* the secret keys that grant access to studies, Cloud Slave provider references and respective authentication credentials. It also supplies control mechanism and request queues for each Gateway. The proposed approach provides a high level of privacy where a small part of identifiable attributes are stored in a trustable provider and such data stored in PACS Cloud Slaves are encrypted using symmetric key algorithms.

Possible unauthorized access to the Cloud Slave repository module does not jeopardise data privacy, since access to the repositories requires the right key to get image objects. Moreover, the data is ciphered in this associative memory and its key is detained in the Master Index entity. Also, the references to slave repositories, where image objects are stored, can only be accessed through information that lies in the Master Index/CloudController.

5.5.2 Integrity

The integrity of DICOM images is guaranteed in the proposed solution at different levels. Firstly, access to them is restricted by authorized PACS Cloud Gateways, and only the allowed users have permission to change such data. Secondly, the medical images stored in the Cloud are ciphered with secret keys using AES algorithms, but previously the system computes a Message Authentication Code (MAC) and it is stored under the Master Index.

Our approach does not guarantee that unauthorized users inside the cloud datacenter could not inappropriately modify the information. Although corruption is possible under the cloud provider, it is not possible for the intruder to change the information to desired data. Finally, the system detects lack of consistence comparing the MAC of DICOM image with the MAC stored into Master Index.

5.5.3 Authorization control

The Master Index owns the access control of the solution. It holds the accounts policy and allows or denies access to the medical data. Each PACS Cloud Gateway instance has to validate firstly through the username and password, which have to be registered previously in the CloudController by the PACS administrator.

Each PACS Cloud Gateway has to introduce access credentials at the start-up of the application. Thus, it will invoke a web service method to the CloudController, looking ahead to

login validation. The CloudController will retrieve a token that has to be sent in each web service method invocation to validate user authorization.

6 DICOM relay service supported on Cloud

"I do not fear computers. I fear the lack of them."
Issac Asimov

In this chapter a WAN (Wide Area Network) DICOM communications platform based on cloud computing will be described. The components were developed over the Service Delivery Cloud Platform (SDCP).

6.1 Description

Traditionally, the exchange of exams between medical institutions was processed through CD/DVD, email [59] or virtual private networks due to the lack of privacy of the DICOM protocol. Although the DICOM standards support SSL/TLS layers, there are many medical devices that do not support these features. Moreover, due to security concerns, DICOM communications are blocked by the firewall to exterior connections, preventing access to the PACS archive by users located the outside institution. If the previous chapter was focused on medical image repositories over the cloud, the driving idea of this one is to promote DICOM inter-institutional communications, allowing the exchange of services across them.

Nowadays, Cloud computing is much used to share files over the Internet and allow users to communicate with each other using external infrastructures. This technology allows the accomplishment of solutions that we have already presented in this thesis, which make healthcare information systems more reliable and scalable than through traditional approaches (e.g. Application Service Providers). The proposed DICOM relay service aims to be a communication broker, allowing search and store of medical images over a group of hospitals, in different sites. Furthermore, this solution fits well with the "Share medical image anytime anywhere" paradigm because it allows communication between several medical devices across multiple Intranet institutions. Additionally, this solution allows access to the internal PACS storage "anytime and anywhere", helping the remote reporting that is significant, for instance in radiology modality.

6.2 Architecture

This approach was implemented according to the architecture proposed in chapter 4 (SDCP). Although the component structure presents some similarity to the PACS Cloud archive, the involved concepts, the architecture design and the scope of the solution are different. DICOM relay service has two distinct components: DICOM Cloud Router (DCR) and DICOM Bridge Router (DBR) - Figure 6.1. The solution is easy to operate because it follows a well-know paradigm, similar to what Dropbox does to share files between the users. In our case, we forward/share the DICOM messages, as we will describe later in this section.

These concepts are related to well-known computer network knowledge. The TCP/IP has an addressing mechanism based on an IP address and port number. The basic idea of our DICOM Cloud Router is to use a new addressing mode beyond the TCP/IP. In our case, the forward technique was based on the normalized AETitle. A DCR analyses the AETitle in the DICOM messages and delivers it to the correct DICOM device, even if it is in another institution (via DBR). It checks the AETitle similar to what a real router does for an IP packet, i.e. check the header IP destination, verify what network it belongs to through routing tables and forwards to the correct network card interface. In our case, it checks the AETitle and sends it to the remote DCR that contains the addressed medical device.

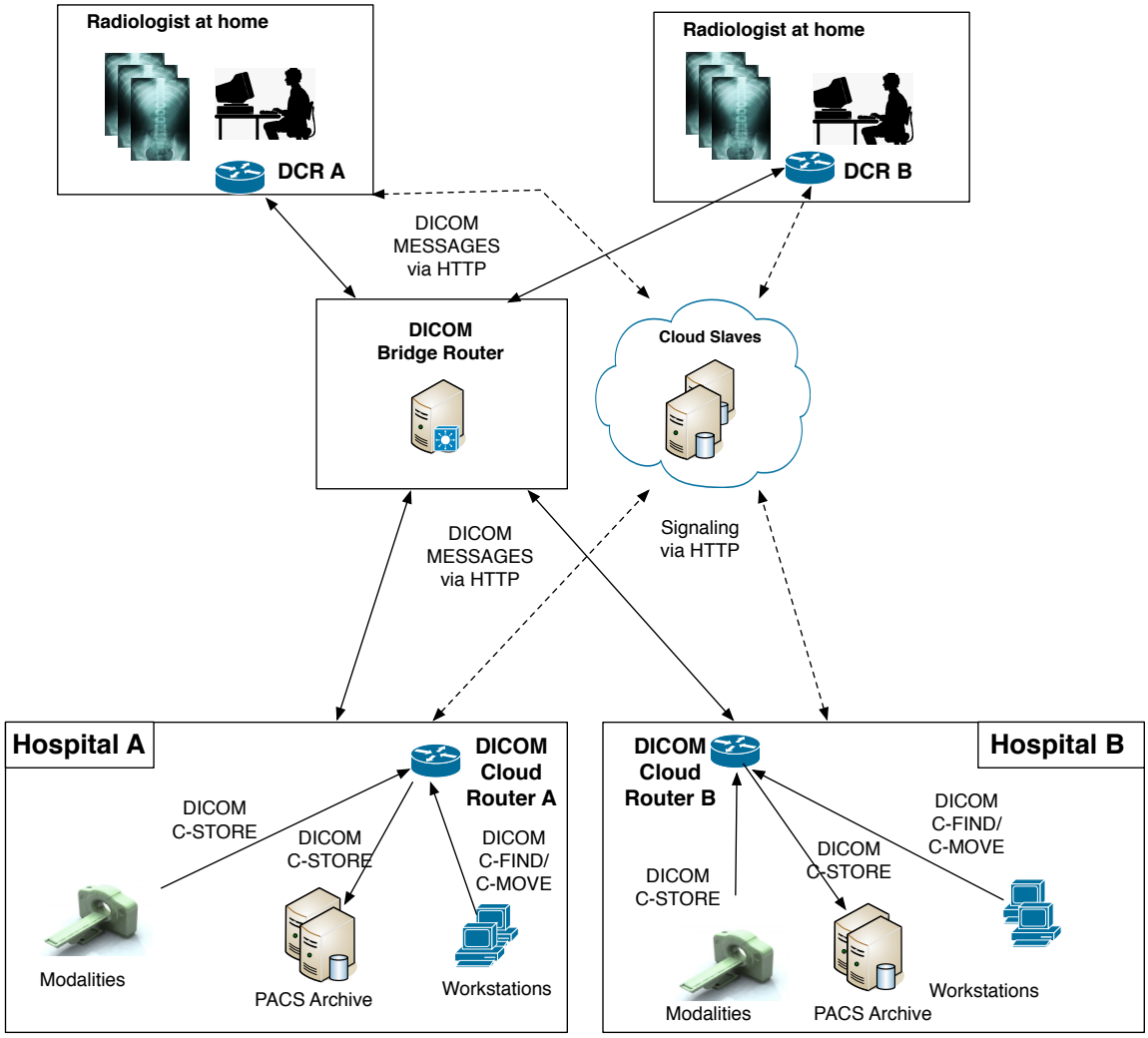


Figure 6.1: DICOM relay service architecture: allowing communication with external DICOM services

The proposed architecture can work with several hospitals, as long as they belong to the same trustable domain. The DCR was implemented as a Cloud Gateway plugin of SDCP. Firstly, each DCR has to be validated in the main infrastructure (Cloud Controller) to be allowed to communicate with agents that belong to the same domain. The architecture supports forwarding of two types of DICOM service messages: Storage and Query/Retrieve. The DICOM messages can flow in both directions, i.e. it can forward a message from a client (SCU) to a server (SCP) outside the institution, but it can also allow forwarding from remote DICOM device (SCU) to an archive

(SCP) inside a private network. Regarding the DBR, which is a temporary information system to forward the messages, it is implemented through the Cloud Controller plugin.

The DICOM relay service is based on the cloud resources. We take advantage of public cloud services to store information that we are forwarding to another router registered in the DICOM relay service. In this implementation, two cloud services were used: blobstore and signalling. The blobstore was used to pass information from one side to other, where the temporary information was stored. Aiming to optimize the DICOM message transfer, we used the cloud signalling service to perform actions in real time, if we have a new DICOM request.

A brief overview of the architecture was described in Figure 6.1. It is possible to observe in the proposed DICOM relay service that it supports several workflows. It allows telework sessions because a home user can have access to institutional PACS archives. Moreover, communications between different hospitals of the same trustable domain are allowed. For instance, hospital A and hospital B in Figure 6.1 belong to the same trustable domain. To add a host to the domain, and allow inter-institutional DICOM communications, PACS administrators need to register devices in respective DCR (hospital A and hospital B), namely PACS repositories, modalities and workstations.

As expressed, there are several use cases for this relay service, for instance, a radiologist can elaborate a report at home accessing to the hospital PACS archive (Figure 6.1). When a workstation wants to communicate with a DICOM host located in another institution, the DICOM message is directed to the local DCR (i.e. located inside the same institution of the workstation) that has the role of forwarding the messages to the correct remote DCR, via DBR, which will forward it to the final destination in the remote institution.

All these workflows require several business processes to communicate between the components of this architecture. Although DICOM standard defines the communication between medical devices and the DCR, it does not define communication through the web. So it was necessary to define the communication messages between the DCR, DBR and cloud slaves. Moreover, all the underlying communication to exchange messages was also considered, for instance, aspects to guarantee the privacy of the actors. All those features will be described in more detail further in this chapter.

6.2.1 DICOM Cloud Router

The DCR has the main responsibility of handling the DICOM services and forwarding messages to the correct place. To do so, it uses AETitle routing tables, i.e. for each AETitle belonging to the DICOM network domain, it contains associated information about the username of Cloud Gateway that is providing the respective service.

The DICOM standard does not provide a mechanism to auto-discovery of the DICOM nodes. Also, for security reasons, only allowed medical devices should access medical archives from outside the medical institution. So DCR has a graphical interface to setup the IP, port and the services available inside the medical institution.

Real world objects were mapped directly in the DICOM standard, for instance DICOM equipment is represented as a “Device” in the defined concepts of the standard. The DCR supports multiple devices (i.e. as many as are online in the WAN DICOM network), each one with a different AETitle and transfer syntaxes. To support bidirectional communications, DCR works as DICOM SCP (i.e. server) and SCU (i.e. client). On one side, the DICOM device that receives the requests is implemented as SCP (i.e. server), but instead of using local resources it will upload the DICOM messages to the cloud blobstore and send a signal to the other router, to get the messages

(Figure 6.1). This last operation is supported on the "Signalling-as-a-Service" abstraction of SDCP (section 4.2.4). All routers have to subscribe on advertisement channels to receive the notifications from the domain. There are several processes involved in this asynchronous communication, as we describe in Figure 6.2. In DICOM communication we have two different processes: first the association and then exchanging the commands of the corresponding service.

Besides the DICOM service listener (i.e. capacity to receive DICOM requests), to support forwarding of SCU (i.e. client) invocations, the DCR is subscribed to receive DICOM association, which is the first message of a DICOM communication (Figure 6.2). For instance, a workstation executes a query to a remote device and the local router will forward the message to another DCR. Thus, the first step is uploading the message to the cloud slaves, which works like a shared memory to exchange data between the routers. Afterwards, the local router sends a notification to the remote DCR through the signaling service channel, which means that the router has a new association request.

The "Association Check" is the module that waits for association requests. It is subscribed to the association channel, which means every time an association is opened, all DCR of the domain are notified. An association remits to a service, thus it will deliver those responsible to the correct checker entity (C-Store Checker, C-Find Checker or C-Move Checker).

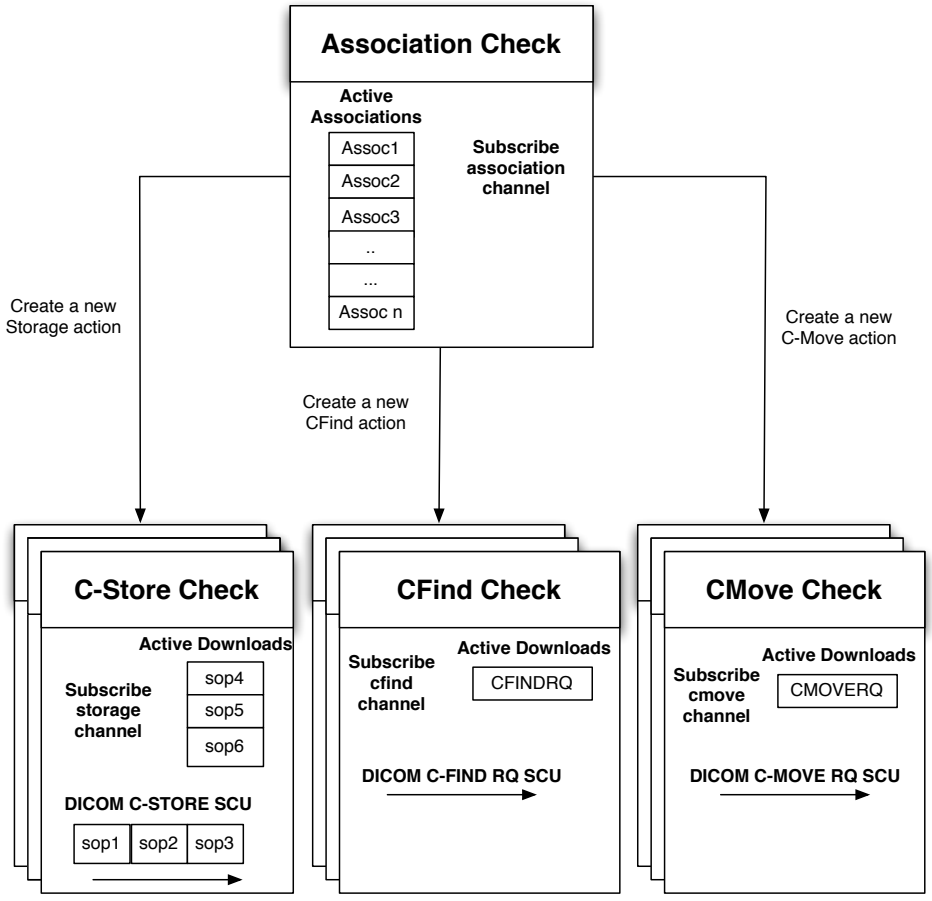


Figure 6.2: Asynchronous method to DCR receive requests from another DCR

The most complex process is the C-STORE because it supports multi-thread mechanism during the uploading of files to the cloud slave. When a C-STORE is invoked, it opens an

association with the remote SCP service, even without all files. In the meanwhile, the files are downloading with a multi-thread process. For each downloaded file the remote DCR sends a notification to the DCR that triggered the initial action, meaning that the file was already transferred.

Regarding the C-FIND and C-MOVE, the amount of information to exchange is smaller and we use single thread to do the process. Also, the C-MOVE takes advantages of the C-STORE relay service. We will describe the more complex workflows later in this paper (see section 6.3).

6.2.2 DICOM Bridge Router

The DBR entity works as a relay between different DCR disperse over several locations. The DBR is an important part of the architecture because it stores information about all devices (i.e. AETitles) and corresponding services supported. It is a system based on SDCP and it was built using the Cloud Controller service, as described in section 4.3. It should always be available over the Internet because DCR needs to write information in the DBR to provide communications. It can be deployed in several places, for instance, in a private cloud detained by a medical institution, Google AppEngine or any other cloud provider. Due to privacy concerns, we strongly recommend deployment of this component in a trustable provider or in-house (i.e. medical institutions).

It is a temporary information system that is accessible through the web service mechanism (RESTful). This component allows storage of the DICOM messages that come in the RESTful parameters. It uses the JPA datastore to automatically store java objects. This mechanism was supplied to store the AETitles of the DICOM network, to support create and delete operations, mainly in associations, C-STORE, C-FIND and C-MOVE, requests and responses respectively. Moreover, the DBR is a very important component because it stores the session key used to cipher DICOM messages of an association. Thus, it should be located in a trustable location, to safeguard the architecture.

6.3 Protocol specification and dataflow

In this section, we will detail the workflows involved in the DICOM relay procedure. The generic message flow from one DCR to another (when located in distinct institutions) will be described.

6.3.1 Dynamic routing table synchronization

It was assumed that in the same trustable domain, more than one DCR might be connected, which means that one or more institutions are connected. Each DCR contains a list of AETitles and services that it supports. Thus, it will reject the connections directed to the AETitle that are not signed in the DICOM relay network. Firstly, the DBR is configured and the information needs to be available to all DCR of same trust domain. Secondly, each DCR has to register the services it provides and wants to share with the group. This information is stored on the local routing table that is configured by the PACS manager for each DCR.

Each DCR sends its local routing table to the DBR, with the follow contents:

- AETitle: the AE Title of the service that the DCR are providing in their private network;
- Service: DICOM service that DCR are supporting for the AETitle. It can be storage SCP, query/retrieve SCP or both;
- Username: the Cloud Gateway agent that is providing the service.

In the synchronization process, a DCR provides its information to the domain but also receives information about AETitles and transfer syntaxes of all other DCR connected. This procedure unites the local tables with the external routing tables. Afterwards, DCR identifies AETitles that are providing services (i.e. servers) and will start to provide all those services to the local network. The DCR will run one DICOM device per AETitle and one single AETitle can support more than one service (e.g. Storage, CFind or CMove). Thus, the DCR can be contacted by the same IP-Port address and it will distinguish the requests by the destination AETitle.

In order to keep all AETitles synchronized, a channel was created for each domain, in the signalling mechanism through the Cloud (as described in section 2.2.3 and 4.2.4). Thus, if a DCR goes to offline mode, enters online or adds new DICOM services a message is sent to this channel. Every DCR of the same trust domain will receive this signal and update their tables. A separated thread keeps the routes synchronized and periodically updates the timestamp of the AETitle entry in the DBR. This update time can be setup by the PACS manager. This is a keep-alive method that allows the removal of offline services from the DICOM network. To avoid conflicts with different AETitles, an AETitle that already exists in the routing table cannot be registered.

6.3.2 Messages

The DICOM protocol and cloud services are not compatible as discussed in chapter 5. Thus it was necessary to endow this platform with a mechanism to make communication through the cloud considering the privacy of patients and clinical staff. The DICOM data flow between medical institutions requires jumping through a hop that needs to convert the DICOM messages to other message representation. The message that comes from the DICOM is passed through a XML serialization, which is saved in cloud blobstores. The DCR invokes a RESTful web service in DBR to register references to these resources stored in the Cloud, i.e. the DICOM message in XML design. All messages in these processes include the domain name to identify the group that the logged user is part of.

A DICOM association is a common process of all DICOM services and it must be opened before sending any sort of DICOM commands. Thus, for each request that the DCR receives, to forward to another medical institution, the first step is to create an association in the DBR. A DICOM association is represented by a message named *BridgeAssociation* that contains the following fields:

- From: the username of the DCR that is forwarding the service.
- To: the username of the DCR that contains the service that the message will be forwarded to.
- CalledAET: the AETitle of the destination
- CallingAET: the AETitle of the client that starts the association
- UID: a unique ID for the association that is pseudo-randomly generated. This UID will also be a seed to generate the key to cipher all the resources that flow over the cloud.
- Service: the service that the association stands for, which can take one of three values, CSTORESCP, CFINDSCP or CMOVESCP.

When a DICOM study is sent to another medical institution, we used external cloud providers to store information, reducing the overload of DBR. We created the *BridgeStorage* message, which contains the UID of association, the SOP Instance UID that it is transferring and also the list of resources to get the DICOM object. These resources refer to the cloud location where this DICOM object is stored. *BridgeStorageRSP* is the message that signals the end of the

transfer, which contains the UID of association. This message refers to the association closing the Storage service.

The DICOM C-FIND message is more complex due to variety of attributes it contains. Furthermore, the content of this message is also confidential to clinical staff. For this reason, the C-FIND message is ciphered and placed in the external resources (cloud slaves). The *BridgeCFINDRQ* stands for CFIND request, contains the UID and the link to the query resource. This external resource contains a query encoded over the XML file, which remains in a list of well-known DICOM tags and respective values (e.g. dates, string, wildcards, etc). CFIND Response is translated to *BridgeCFINDRSP*, which contains CFIND responses that are structured and organized in XML.

To move DICOM objects, the router platform supports C-MOVE command. In that case, the message is also simply called *BridgeCMOVERQ*, which contains the UID of association, query level and the instance UID that allows execution of the query to look up the DICOM images. The answer of CMOVE is *BridgeCMOVERSP* that basically is an acknowledgment to give feedback to the DICOM device that performs the C-MOVE command (similar to storage acknowledgment).

6.3.3 Storage

Storage relay service involves forwarding a DICOM object to a PACS storage server located in another medical institution, in a transparent way to DICOM devices. So each DCR binds DICOM storage service according to the available AETitles in DBR. For instance, the PACS manager sets up a modality device to send generated studies to another medical institution, using the AETitle of the remote archive and the pair (IP, Port) of the DCR of its medical institution.

Service storage is responsible for sending resources from one DCR to another (remotely located) one. As described previously, we have a bridge over the cloud to support this communication. In the storage service the resources are quite relevant because they represent a huge amount of volume to send over the Internet. Also, due to privacy and confidentiality issues the data to send over the cloud might be ciphered. Cloud blobstore is used to save the information temporarily and the signalling service to send a notification to the other remote party, alerting that they have a new DICOM request. Figure 6.3 is a data flow diagram that helps to understand all processes involved in the point-to-point storage service.

In the storage process the DICOM modality devices (e.g. CT Scanner) send a DICOM study to the other institution's archive. The device is configured to send the studies to the local router, i.e. the DCR 1 host (step 1 in Figure 6.3). The association process is the normal process compared to the local PACS because the DCR has the necessary configurations to handle standard association requests (i.e. AE Title and transfer syntaxes). For the association, the DCR 1 will create a *BridgeAssociation* message and will invoke the REST web service at DBR to start the association (step 2). To do that, it is necessary to check the routing table to identify the username that holds the other DICOM device. Meanwhile, the storage handler (DCR 1) receives the C-STORE requests (step 3 and 4) and starts to put the DICOM objects in the cloud blobstore (step 5). Then a new *BridgeStorage* message is sent to the DCB for each file, containing the links to the resources and also the corresponding UID (step 5). Thus, after this transaction the remote router (i.e. DCR 2) is notified and this happens for each DICOM object received (step 6).

DCR 2 has a thread for each *BridgeAssociation.* It is subscribed to the storage channel and is notified for each signal that it receives. Besides, it creates another thread to download the files from the Cloud slaves (step 7 and 8) that are added to the storage check thread pool. Afterwards, when it receives the first resource it starts to open an association with the archive storage SCP

through the Storage SCU (step 9). The client waits until the next DICOM object arrives from the cloud (step 10).

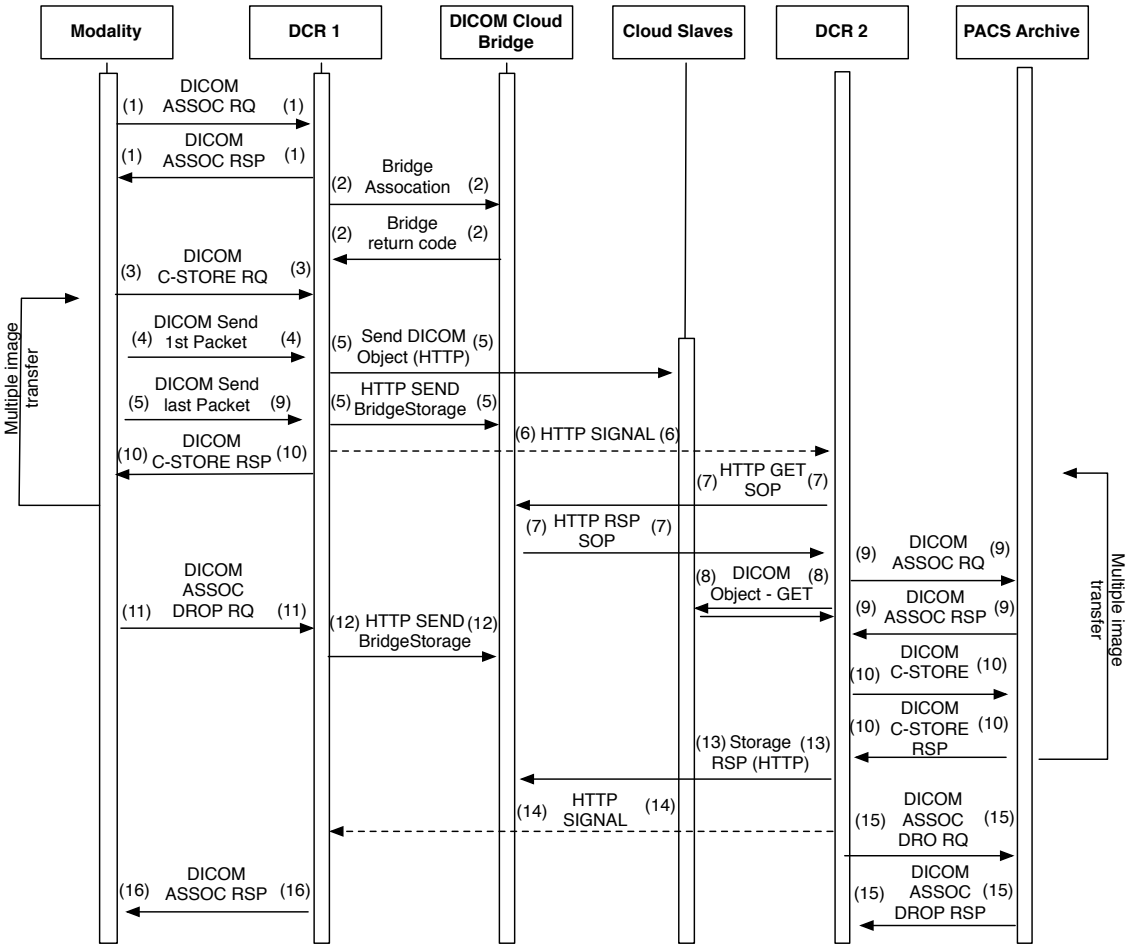


Figure 6.3: Storage service relay: data flow

Finally, the end of the operation will be detected because after download is completed the "*Storage Check*" checks if DBR contains a *BridgeStorageRSP* for the current association. Then, it closes the processes with the client and sends an acknowledgement to the DCR 1 (step 13 and 14). Both associations are closed in the two DCR (step 15 and 16).

6.3.4 Query/retrieve

Query/retrieve service contains two distinct commands: C-FIND and C-MOVE. First, we will explain the C-FIND data flow and then the C-MOVE flows.

C-FIND

The C-FIND command allows the user to perform search operations over a PACS archive. In the scenario presented, it allows the user to search over a remote PACS repository. Besides the common attributes that a DICOM command contains, C-FIND message has an IOD that refers to several DIM fields that are usually used by clinical staff. Once again, a data flow diagram (Figure 6.4) will help to follow the C-FIND operation over the cloud. So during the remote query procedure, the workstation needs to establish an association with the remote AETitle. Firstly, the message will be delivered to the first hop, which is the router (DCR 1) inside the medical institution (step 1 in Figure 6.4). In the association process DCR 1 will detect that it is a CFIND request and will create an UID for it. This information will be stored in DBR (step 2).

The DCR 1 receives DICOM C-FIND requests (step 3) and translates them to a non-DICOM message, i.e. *BridgeCFINDRQ* (step 4). Meanwhile, the query is exported to XML, ciphered and uploaded to the blobstore (step 4). The invocation of *BridgeCFINDRQ* and the store of the query in the cloud blobstore is an atomic transaction, which means that only both can happen and after that the remote router (i.e. DCR 2) can be signalled (step 5). After the DCR 2 receives the signal, it gets the resources from cloud slaves and DBR (step 6 and 7). The XML query is deciphered and, it will ask for the correct PACS archive based on its routing table (step 8 and 9).

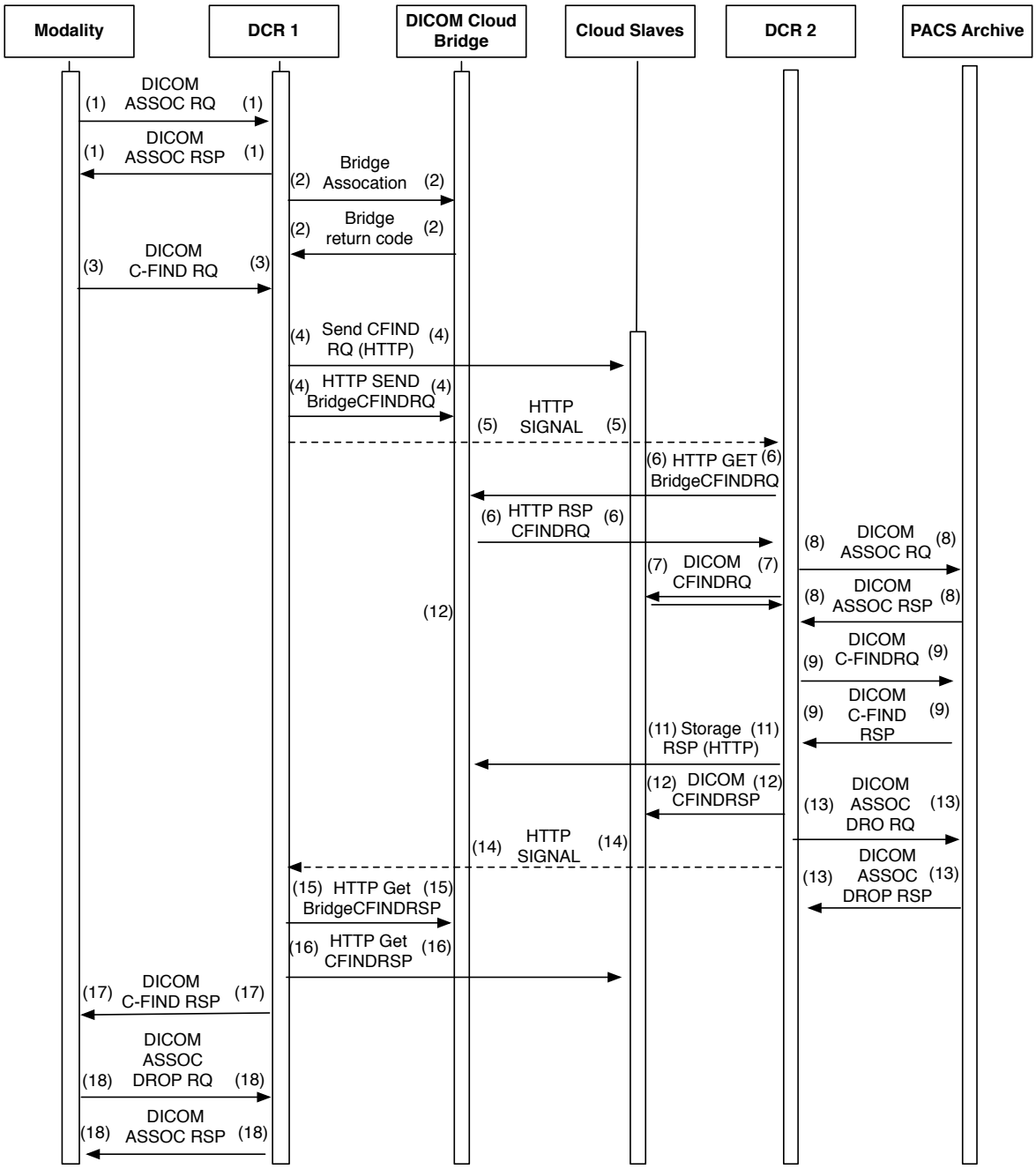


Figure 6.4: Query (C-FIND) – data flow

The PACS archive will return the responses with the match to the query (step 9). The response is also ciphered and the response is put in the cloud blobstore and the response message is

sent to the DBR that contains the resource address (step 11 and 12). After the transactional action the router DCR 1 also receives an asynchronous signal (step 14). It gets the message from DBR (step 15) and downloads the XML responses (step 16). After deciphering the answers, it will create the DIMSE C-FIND Responses and sent back to the workstation (step 17). Afterwards, the DICOM associations are closed (step 18) and meanwhile the association on the remote side was also closed (step 13).

C-MOVE

Once again, we will use Figure 6.5 data flow diagram to describe C-MOVE processes. Typically, C-MOVE action is performed by the workstations. They send a request to move a patient study, a series or more rarely an image (step 1 in Figure 6.5), from remote an archive to a local computer.

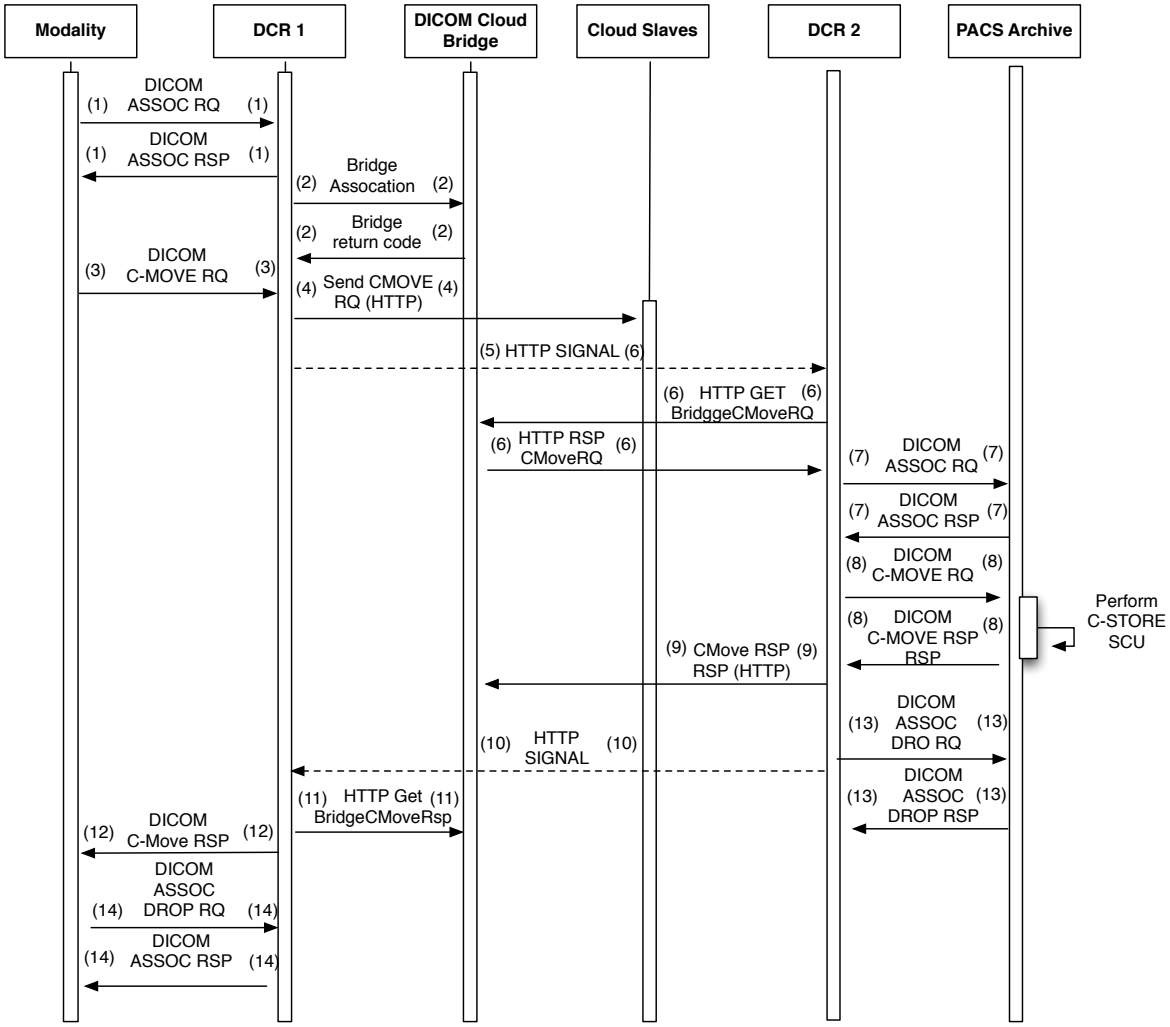


Figure 6.5: Retrieve (C-MOVE) data flow

Like in C-FIND, the C-MOVE process, creates an association identifier in the DBR (step 2). Then, the workstations will demand to move (C-MOVE request) to DCR 1 that receives the move action (step 3). The C-MOVE request is parsed and DCR will get the level and the instance UID. Next, the system creates a message *CMOVERQ* that posts to the DBR (step 4). Then, it sends a signal to the remote router - DCR 2 (step 5), which will receive and get the message to move based on the *BridgeAssociation* UID (step 6). Thus, it will perform the DICOM C-MOVE

command according with their routing table (step 7). On the other side, the archive server starts moving through the storage service that is common in the retrieve action (step 8). When the study is delivered to the requesting workstation, a *BridgeCMOVERSP* is sent (step 9). A signal is sent to the origin DCR 1 and the process is finished (step 10). Subsequently, the DCR 1 receives the *BridgeCMOVERSP* (step 11) and retrieve C-MOVE response (step 12). The associations are closed (step 13 and 14).

6.4 Assessment

The presented solution allows DICOM standard communication between different medical devices located in distinct institutions. The proposed architecture allows to creation of a federated DICOM network located over distinct medical institutions, creating a unique view of all resources.

It is a fact that other solutions exist, for instance, VPN and email. There are also some other alternatives that allow the exchange of DICOM communications via email. However, they do not offer any privacy with regard to the email provider.

DICOM relay service is easy to deploy in an institution and the end-user does not need complex setups to start communicating with external repositories, allowing interoperability with any the DICOM standard device. Besides, the required infrastructure is not excessive because it supports its main resources on the Cloud.

7 Results and Discussion

"The man of science has learned to believe in justification, not by faith, but by verification."
Thomas H. Huxley

This chapter shows the outcome of this thesis, presenting experimental trials and comparing them with traditional solutions. It also explains how we arrive at results and why there is an impact on the real deployment of this solution in the medical environment.

7.1 PACS Cloud Archive

To validate the proposed architecture we have deployed each of the PACS modules in two cloud storage providers, namely Amazon S3 [60] and Google Storage [61]. Amazon Simple Storage Service (S3) is a higher-level managed service of Amazon Web Services (AWS) and is used by popular products, for instance, the file synchronizer Dropbox. The Google Storage is a RESTful service for storing and accessing data using Google's infrastructure. We used several third party client workstations to test the cloud archive, namely OsiriX [62], dcm4che2 and dcmtk [63]. Moreover, in order to compare the developed solution with a traditional one an Intranet PACS archive was used, namely the Conquest [64].

The developed modules were tested with several study cases with a significant amount of DICOM images of different modalities, stored in both cloud providers. To evaluate solution performance and robustness, the PACS Cloud archive was tested using a data set of 7 studies, containing 1153 DICOM files, and including several modalities, namely XA, MRI, CT and Nuclear Medicine (NM).

We made storage and retrieval trials with well-known Amazon S3 and Google Storage providers. Moreover, the Master Index was also hosted in each Internet cloud provider. The cloud archive gateways, e.g. the DICOM Storage and Query/Retrieve SCU (Service Class User) nodes installed in institutions' Intranet, were tested with a connection with 24Mbits/s of downstream and 12Mbits/s of upstream. Finally, to have time references, similar operations were carried out with a traditional PACS archive located in the same Ethernet segment with a Gigabit connection. The average time to obtain results with trials is shown in Table 7.1 (storage process), Table 7.2 (query process) and Table 7.3 (retrieval process). It includes upload and download processes under HTTP, database interactions, and DICOM communication from/to PACS Cloud Gateway to DICOM devices.

In general, the trials showed that the solution is robust and that it was possible to store, query and retrieve all the desired studies without any problem or service interruptions.

7.1.1 Storage measurements

The PACS Cloud storage is a relatively complex process due to decoupling of the various modules and privacy issues. As expected, the traditional PACS archive image storage, based on

DICOM C-STORE command, is faster due to its simplicity and local operations. It is unquestionable that Cloud storage will always be slower than similar operations executed over traditional LAN PACS. However, the keynote is to analyse the solution's feasibility in a real world environment. The medical records are temporarily stored under PACS Cloud Gateway and their upload is happening in background. Meanwhile, the DICOM connection with Intranet images sender (e.g the modality) is released. The Cloud storage times presented in Table 7.1 are acceptable for a typical DICOM institution, because archiving medical image studies is executed just once and without impacting on the end-user interface.

Table 7.1 PACS Storage Process: Quantitative Measurements

Modality	Number of Files	Volume (MB)	Google Storage (sec)	Amazon S3 (sec)	LAN PACS (sec)
NM	1	1	3.9	5.33	0.4
NM	5	2	4.5	6.6	0.5
NM	6	8.2	16	30.8	1.3
MR	243	16.5	38.9	40.2	5.3
CT/PET	224	47.5	53.5	49.9	10.3
MR	611	206	228.1	163.2	31
XA	13	384	533.1	546.3	40

7.1.2 Query measurements

Queries are an important process in a regular PACS Archive operation mode. Table 7.2 is presents the measure of the DICOM C-FIND process, from the workstation executing the C-FIND request until it receives the C-FIND responses. The Cloud Gateway has an important role in this procedure because it executes queries on the Cloud Slave database (Amazon SimpleDB) and calls the Master Index to access the clear data. Finally, they collect the queries and return the results to the Intranet workstation in a C-FIND response. The results obtained (Table 7.2) proved that querying the PACS Cloud archive is a process with a very acceptable performance, i.e. the time measurements are not far from the local PACS Archive.

Table 7.2: PACS Query Process: Quantitative Measurements

Number of Results	Amazon SimpleDB (sec)	LAN PACS (sec)
1	0.39	0.13
2	0.42	0.14
4	0.46	0.17
8	0.66	0.18
16	0.83	0.28
32	1.09	0.45

7.1.3 Retrieval measurements

The analysis of studies' retrieval time is much more critical to validate the PACS Cloud solution because the image data can be accessed (and downloaded) several times in the same procedure and the physician is in front of a work station monitor waiting for the download to be completed. Once again, the traditional PACS image retrieval, based on DICOM C-MOVE and C-STORE commands, is faster than the Cloud solution. The LAN PACS retrieval delays are similar to storage times because the process time associated with the retrieve command (i.e. C-MOVE) is residual compared to the effective network data transfer (i.e. C-STORE). However, a good observation is that the retrieval time differences from Cloud to LAN are much lower than in the storage process (Table 7.3). Also, in the medical workflows, medical images may be fetched before they are needed, using the Modality Worklist, i.e. a system that move, the patient's exams earlier, using the healthcare information systems. We will discuss further details about optimization.

Table 7.3: PACS Retrieval Process: Quantitative Measurements

Modality	Number of Files	Volume (MB)	Google Storage (sec)	Amazon S3 (sec)	LAN PACS (sec)
NM	1	1	2.3	3.4	0.3
NM	5	2	3	3.7	0.4
NM	6	8.2	8.2	19.9	1.2
MR	243	16.5	19.9	20.6	2.3
CT/PET	224	47.5	26.4	28.3	6.4
MR	611	206	74.9	72.1	29
XA	13	384	640.9	538.2	39.5

7.1.4 Optimizations and performance measurements

To improve solution performance associated with PACS Cloud storage, query and retrieval processes, compression and cache mechanisms were proposed (described in section 5.3). Figure 7.1 shows the storage process, comparing the different approaches: PACS Cloud archive ("Standard", Compression, Cache) and the Intranet PACS.

The Figure 7.1 shows that utilization of the compression technique in PACS Cloud archive storage process has a significant improvement on performance. Despite the storage process not having any impact on the end-user, this optimization allows a reduction of the costs with storage and transmission to the Cloud. As expected, the cache mechanism has quite similar behaviour to LAN PACS.

In the query process use of the cache mechanism was also tried, namely a local DICOMDIR data structure that stores necessary information to answer queries regarding the cache local archive. We performed the same queries that we executed in previous tests and compared the results with local LAN PACS (Figure 7.2).

The results with an outsourced database in SimpleDB cloud provider were considered acceptable. However, querying is a process that users expect to be faster. Thus, this optimization will reduce the end-users' waiting time. There are significant differences using cache, and the values are very close to the local LAN PACS. Furthermore, the cache mechanism is a key point if a medical institution loses the Internet connection because the system still works for the majority of the operational functions.

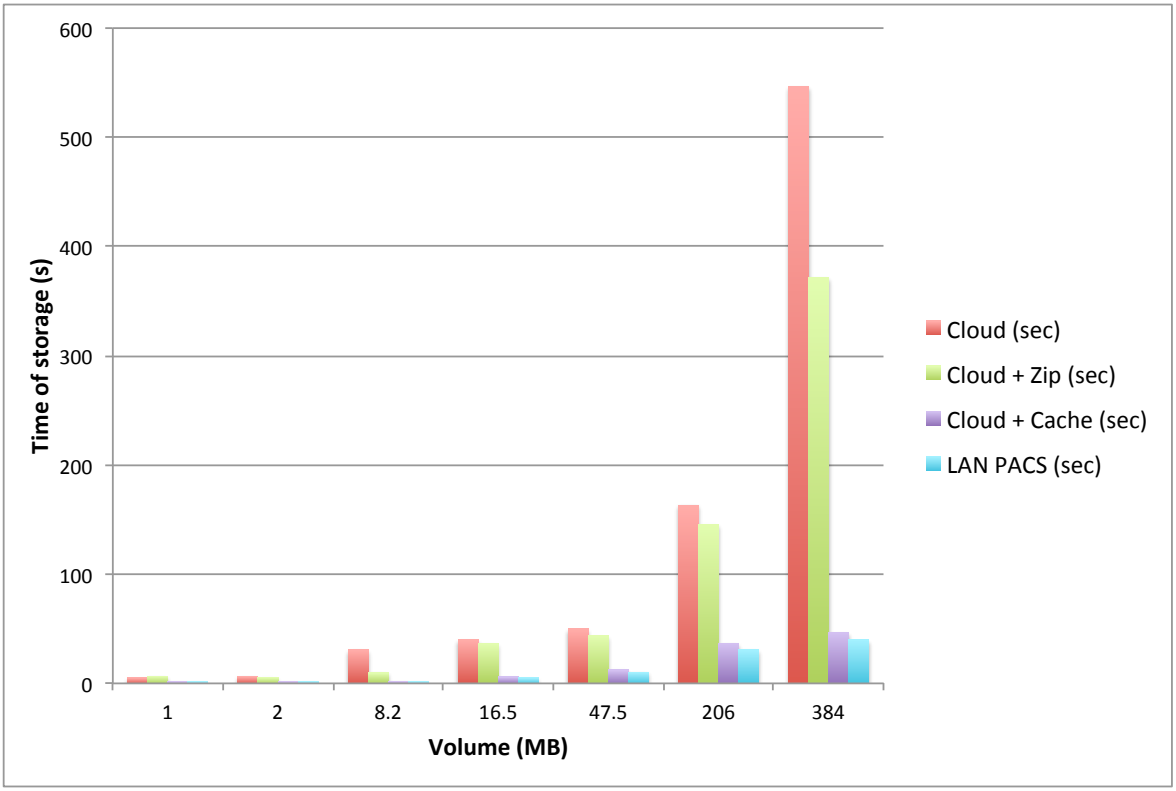


Figure 7.1: Comparing storage measurements

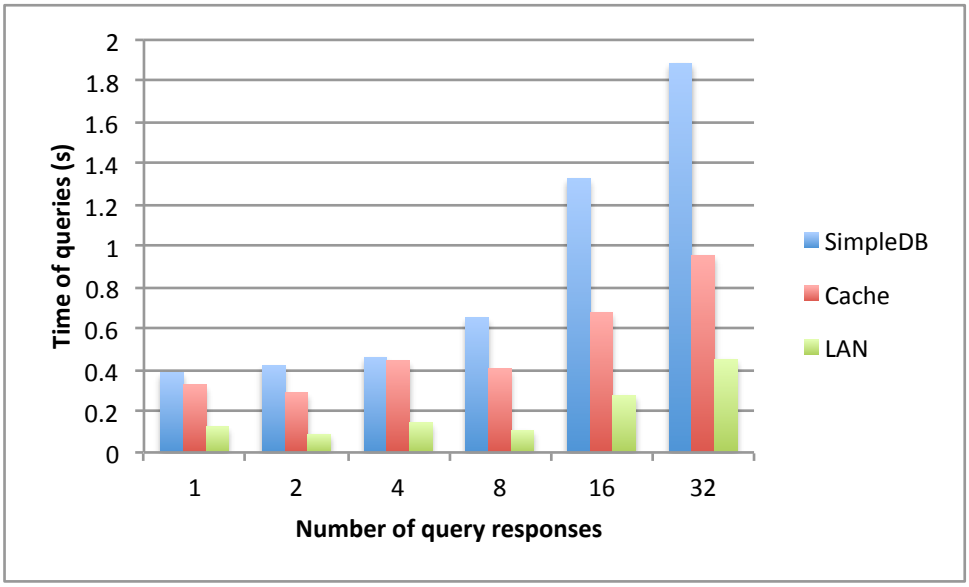


Figure 7.2: Comparison query measurements

Retrieval processes was one of the most analysed aspects in the PACS Cloud archive. The introduction of compression and cache mechanisms was primarily thought to have a great impact on this process. Figure 7.3 shows that the use of zip compression reduced the download times. All times were considered during the process, including the uncompressing procedure. For instance, unzip takes an average of 100ms to extract an image of 1.3Mbytes , which we considered a short time due to the measured time of a study with 1.3MBytes. The compression improvements are not impressive because some images used are in JPEG Lossy format. The use of zip compression with those images results in a file with a few bytes more than the original size.

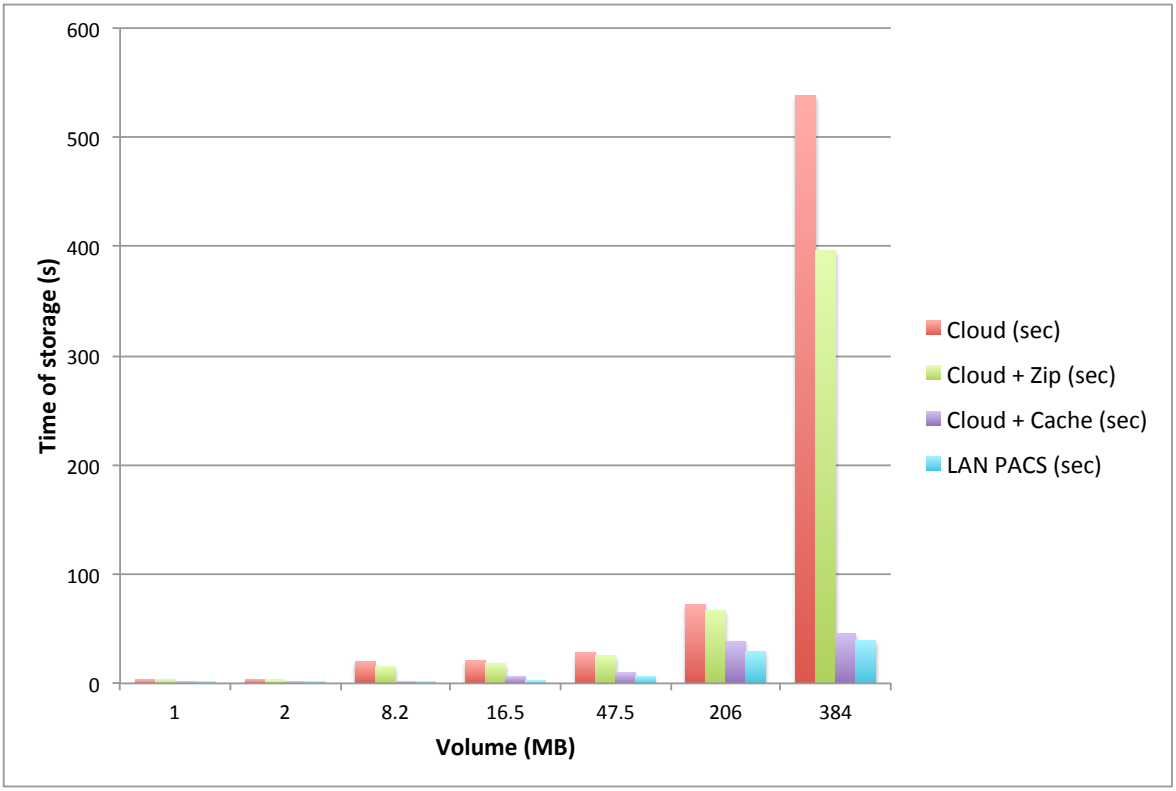


Figure 7.3: Comparison of retrieval measurements

Finally, compression use improves the storage and retrieval time, which has a strong effect on the solution. Furthermore, it also reduces significantly the amount of volume stored in the Cloud. Moreover, the compression should be applied because it has a performance and financial impact.

There are some fluctuations in the upload/download of DICOM studies that are mainly dependent on network traffic and on split/join, zip/unzip and cipher/decipher procedures. Moreover, the upload and download bandwidth on the Internet may vary and this was a factor to be considered during the storage and retrieval procedure. Finally, if the Internet connection is really fast, it may be that the zip/unzip process does not have an impact on the solution. Moreover, it can degrade the time performance of the solution due to the slowness of the zip process compared to Internet upstream/downstream.

7.2 DICOM Relay Service

In order to assess the performance of DICOM relay service, trials were performed with two different networks: in the University of Aveiro campus and a home network outside the campus (~60km away from the University of Aveiro campus). We accomplished the tests with an Intel Core 2 Duo 2.2GHz with 2GB RAM and an AMD Athlon 1.6 GHz with 700MB of RAM. In this section, we will compare the results supported on Amazon S3 cloud provider. The values with DICOM direct TCP/IP connections between the two sites were also analysed. Our DBR was deployed in Google AppEngine and the signaling provider was PubNub. We used a dataset with 7 studies containing 1153 DICOM files.

7.2.1 C-STORE relay measurements

We analysed the storage method from one institution to another, which we considered a key point in regional communications in telemedicine. We compared the values with DICOM

direct connection (Table 7.4) and, as expected, the storage to remote repositories is slower. This process may be optimization with the splitting the files. Moreover, the study threads parallelization may change the behaviour and improve the time measurements of the storage.

Table 7.4: DICOM relay service: C-STORE

Modality	Number of Files	Volume (MB)	Relay Cloud AWS S3 (sec)	Direct connection (sec)
NM	1	1	23.83	2.67
NM	5	2	29.13	2.95
NM	6	8.2	85.37	11.4
MR	243	16.5	53.62	14.23
CT/PET	224	47.5	112.55	39.91
MR	611	206	236.9	197.66
XA	13	384	796.6	385.35

7.2.2 C-FIND relay measurements

The query case study is very important because it is a process that is very close to the end-user of the workstation. This test was performed using the Dicoogle PACS [65] to execute remote queries over the network. Like the PACS cloud archive, we search over the remote repository using the same amount of results that we tried in the previous experiences. In that case, we consider that the times measurements are still slow. However, the direct connection does not supply any security mechanism and it is very difficult to implement in a real scenario.

Table 7.5: DICOM relay service: C-FIND

Number of Results	Relay Cloud AWS S3 (sec)	Direct connection (sec)
1	10.29	0.32
2	11.83	0.39
4	13.35	0.49
8	14.96	0.45
16	17.41	0.52
32	18.17	0.69

7.2.3 C-MOVE relay measurements

The retrieval process can be critical in urgent access cases. We consider that this time measurement depends on the bandwidth of the ISP that end-users are using. The direct connection is faster than the DICOM relay service presented. As discussed, the direct connection has associated privacy concerns. Moreover, this process can be optimized by splitting the huge files, with an automatic estimated value, depending on their size.

Table 7.6: DICOM service relay: C-MOVE

Modality	Number of Files	Volume (MB)	Relay Cloud AWS S3 (sec)	Direct connection (sec)
NM	1	1	35.87	3.12
NM	5	2	33.14	3.39
NM	6	8.2	98.49	16.35
MR	243	16.5	79.5	20.29
CT/PET	224	47.5	135.38	41.3
MR	611	206	245.12	185.3
XA	13	384	823.36	436.36

We consider the measurement times are a bit higher than what is reasonable in the tele-radiology scenario, although they can be improved in the near future, through the reduction of messages passed between the components of the architecture and also with some parallelization processes.

This solution has other benefits, due to easier application in a hospital and even in any computer that can work "anytime, anywhere". Moreover, several medical institutions still use conventional mail to transport CD/DVD, emails [59], etc. Thus, this solution will improve the workflow of this procedure.

7.3 Discussion

The use of Cloud computing utility has increased significantly in recent years and it appears as a natural evolution of the datacentre to perform more scalable computing and storage. With such a significant increase, the market is growing quickly and there are more companies providing new services with better features, including isolated services. So it is possible to rent a storage service from one provider and a database from another. Nevertheless, the cloud access interfaces have differences and those services are not compatible at all, i.e. the API is not the same. Indeed, the services provided by the cloud players have some areas in common, but it is not enough to ensure that clients can use the same procedure with different cloud platforms. For instance, Google Storage and Amazon S3 supply storage service, but the interface to store and access the files is different. We have developed an approach that deals with these differences and allows storage and retrieval of medical images from any kind of cloud storage service. Moreover, the solution is vendor neutral and allows storage of information in more than one provider at the same time. This possibility has special interest for backup strategies, to reduce download times and increase data availability. Even if one server is jeopardised, the solution might use the other provider to restore the information.

Nowadays, cloud providers proclaim their services in a philosophy of "always available" or "99.9%" available. But in such cases, data availability belongs to the cloud providers, and the availability of medical images might be questioned by hospital staff. In such cases, the architecture allows us to have a PACS archive repository with object instances replicated over multiple cloud providers, granting data redundancy. Thus, the availability of medical data will increase substantially compared with a single provider approach. It also increases the safety of medical data because the information can be stored in more than one provider, allowing a multi-provider backup strategy, i.e. the PACS Cloud Gateway sends the medical information to more than one provider at the same time, or scheduling these tasks for periods with less activity is a reasonable option.

Nonetheless, availability points nowadays to real-time access and not to download process waiting-time to visualize DICOM studies. In order to avoid the latency in access to studies, a cache mechanism was developed and integrated in the PACS Cloud Gateway module. This permit us to keep in cache the most used studies, those most likely to be consulted or based on pre-fetched systems, which reduces retrieval times and further enhances data availability.

Furthermore, deployment of the solution without the cache mechanism has some latency in accessing medical studies. Introducing a cache mechanism can solve the underlying problem and might be helpful in providing a reasonable workflow. Some literature discusses the cache issues and explains in detail why a cache is important, pointing out that a cache with 1-3 months of recent exams can have a high hit percentage. [33]. However, the cache size and lifetime should be adapted considering facts such as modalities, institution workflow, resources available, etc. A PACS Cloud Gateway may integrate a caching mechanism that allows storing studies locally during a period predefined by the administrator. Thus, retrieval of these studies during this cache period will be similar to local PACS. In addition, the Gateway can also download from the cloud the next required studies triggered by either DICOM Modality Worklist or recently performed studies, which enables having studies before they are needed. These strategies could have a positive impact on deployment of the solution in a real environment avoiding the latency in the retrieval process.

The presented architecture and its integration into the medical workflow could follow the same procedures as a traditional PACS archive server. In doing so the PACS Cloud Gateway represents the archive server entity to Intranet hosts. Integration with Hospital Information System (HIS) or Radiology Information System (RIS) has to comply with the usual practices.

Finally, the PACS Cloud archive proposed in this article implements two usual DICOM services: Storage and Query/Retrieval. However, other DICOM services can be implemented in the solution. For instance, retrieval through WADO was implemented in the architecture presented. All these extensions should be added to the PACS Cloud Gateway entity, which is the module that enables communication between the institution DICOM world and Cloud providers.

Regarding the DICOM relay service, it has multiple benefits in the regional PACS according to their needs for teleradiology communications. As we already verified, for this solution to perform well it is necessary to have a good Internet connection in both sites. We considered that moving information is less expensive than moving the patient, and despite the costs of this solution and the drawbacks in the time measurements, the final balance is still strongly positive.

The proposed architecture to relay DICOM services has an unquestionable benefit, which is the interoperability between medical devices. Radiologists can work at home, in the same way that they do in the hospital, without changing their methods. They will be able to access PACS in the remote hospital without needing to waiting for the exams to arrive by email or other mechanisms.

Furthermore, for a regional PACS/Tele-imagiologic services, the proposed architecture can provide the following benefits:

1. Tele-image center: a shared repository between a group of hospitals and the PACS cloud archive meets their needs.
2. Remote Query/Retrieval: radiologists can perform query and retrieve to a remote PACS archive.
3. Auto forward auto (1 centre repository): an institution that has several distributed hospitals can have one central repository in a single site, shared with other hospitals. So all medical institutions of the same group have access.

The developed solutions are flexible and can fit a very large universe of workflows depending on practice protocols. The results and trials show that tele-imagiology services over the cloud can perform well without any constrains due to the privacy and confidentiality issues. Moreover, we explored optimizations that allow working with a PACS Cloud archive in a similar way to a traditional solution, i.e. PACS archive inside the medical institution.

8 Conclusion and future work

"Experience is never limited, and it is never complete; it is an immense sensibility, a kind of huge spider-web of the finest silken threads suspended in the chamber of consciousness . . ."
Henry James

In this chapter we summarize the work developed in this thesis. We mention the problems tackled during the whole process. Then we list the main contributions presented by this thesis. Finally we suggest open problems for future research, focusing on the work developed work here.

8.1 Issues and controversies

To start developing a PACS archive solution we tackle many issues and controversies. In the beginning, we faced the problem regarding the cloud provider and platform to develop the solution. We define an abstraction and interfaces that the system needed, which allows us to develop a solution without knowing what cloud provider it was going to be deployed in. This has benefits for the solution because now it is a pluggable and generic solution.

In addition, we tackle several difficulties to deploy some components of the architecture, namely Master Index, based on AppEngine framework. For instance, JPA and JDO have several tricks to deploy over the AppEngine.

The DICOM relay service is a distributed system, which runs in different machines. The coordination of the processes between the DICOM services and the cloud resources has several complications to keep it synchronized.

Afterwards, we overcome all difficulties and develop a solution that meets the initial goals. In addition, we create independent modules that can be used in several use cases and distinct scenarios explored in this thesis.

8.2 Main contributions

We can identify 3 main contributions of this thesis: a multi-vendor platform to deliver services using cloud resources, PACS cloud archive and a DICOM relay service.

PACS-as-a-Service is a solution that provides storage of medical studies using cloud resources. The solution cipher the data over the cloud to avoid unauthorized access by third parties, e.g. Cloud providers. PACS Cloud grants interoperability with DICOM devices and acts as a traditional PACS archive from the client's perspective. Furthermore, on the administrator side, it greatly reduces IT infrastructure and brings new management facilities to multi-site institutions with shared PACS.

Migration of the current PACS archive to a PACS cloud archive can be a threat to real deployment of the solution because the entire infrastructure has already been acquired.

Nonetheless, this solution is a new opportunity for new hospitals and small centres that want to start without purchasing much infrastructures. The characteristics of the PACS Cloud presented ensure scalability and reliability of medical data, and can have a significant impact on healthcare institutions, by reducing local IT infrastructure. Integration within a healthcare institution is effortless and inexpensive since the solution is fully compatible with DICOM devices. Moreover, the implemented solution is easily expandable to other cloud providers due to a decoupling strategy that allows integration of new cloud drivers with a pluggable method.

DICOM relay service is considered PACS-as-a-Service because it allows distributing of medical images between multiple centres in different locations supported by the cloud resources. The modal presented is very important, mainly in the radiology modality. In this modality there is much remote reporting due to economic reasons. For instance, a hospital does not offer all specialists of all modalities and, in that case, a remote specialist can elaborate a report and store it in the PACS archive of the medical institution that holds the clinical case.

Furthermore, we have contributed to this research area with the following published articles:

- L. Bastião, C. Costa, and J. L. Oliveira, " A PACS archive architecture supported on Cloud services", International Journal of Computer Assisted Radiology and Surgery. Springer, 2011.(DOI: 10.1007/s11548-011-0625-x)
- L. Bastião, C. Costa, and J. L. Oliveira, " A Secure PACS Cloud Archive", in 29th International EuroPACS Meeting (CARS 2011), Berlin, Germany. 2011.
- L. Bastião, C. Costa, A. Silva and J. L. Oliveira, "A PACS Gateway To The Cloud", in 6th Iberian Conference on Information Systems and Technologies (CISTI 2011), Chaves, Portugal. 2011

We also submitted the "Tele-imagiology services platform over the cloud" to the ACM IHI 2012 conference and "A platform for delivering services over multi-vendor cloud resources" to the ACM SoCC conference.

We strongly believe that the solutions developed can be industrialized in European markets, in the near future. Besides, the contributed papers will have a relevant impact on this research area.

8.3 Further work

Despite the contributions listed in the previous section, still much work is left to be done in the future. Next, we will summarize what should be done to improve the work and expand the solution to other areas:

Performance improvements

While the adopted solutions in the storage and retrieval process are still statically assigned, we believe that the results could be improved in the near future through studying the parallelization of business processes. Moreover, the thread pools can be dynamically changing depending on the bandwidth connection available and depending on the size of chunks stored in the cloud blobstore. The chunks' size can also be dynamically assigned. A new mechanism must be developed in order to take advantage of the architecture proposed in this paper and allow end-users to perform operations faster.

Cache prediction

Our cache is based on a period of time and amount of volume that the PACS manager intends to store. We consider that cache is a key point in a PACS cloud archive. Thus, a cache that gets information from DICOM Modality Worklist is needed. In addition, a prediction cache can be done based on previous queries and retrieval actions performed. The model can fit the workflows of the medical institution automatically.

RIS supported on cloud services

One of the information systems in medical institutions is the Radiologist Information System, which contains the order of clinical trials. This system interacts frequently with PACS and is also very important to the institution. It has the same security concerns of the PACS archive (redundancy, backups, etc). Thus, we propose outsourcing RIS to the cloud to allow sharing in the inter-institutional scenario.

HL7 communication relay

While developing the work presented in this thesis we created a modular system. Thus, we propose a relay communication to another healthcare protocol named HL7 (Health Level Seven International). It is a different area from the content addressed in this thesis but can still be explored.

Telemedicine scenarios

The developed platform allows to storage medical records over the cloud. Also, distinct medical institutions can access remote DICOM archives. It can be expanded to other HIS executing queries to the remote information systems. Moreover, it is possible to create a federated view of all information systems for telemedicine and collaborative work.

9 References

1. **Google Health** [www.google.com/health] Available in: May 2011
2. **Microsoft HealthVault** [www.healthvault.com] Available in: May 2011
3. Costa C, Oliveira JL, Silva A, Ribeiro VG, Ribeiro J: **Design, development, exploitation and assessment of a Cardiology Web PACS**. *Comput Methods Programs Biomed* 2009, **93**(3):273-282.
4. **Prepare for Disasters & Tackle Terabytes When Evaluating Medical Image Archiving**. In.: A Frost & Sullivan Healthcare Article. http://www.frost.com.; 2008.
5. Huang HK: **PACS and imaging informatics: Basic Principles and Applications**, 2nd edn. New Jersey: Wiley & Blackwell, Hoboken; 2010.
6. Furuie S, Gutierrez M, Bertozzo N, Figueriedo J, Yamaguti M: **Archiving and retrieving long-term cineangiographic images in a PACS**. In: *Computer in Cardiology 1999; Hannover , Germany* 435-438.
7. Reiner BI, Salkever D, Siegel EL, Hooper FJ, Siddiqui KM, Musk A: **Multi-institutional analysis of computed and direct radiography: part II. Economic analysis**. *Radiology* 2005, **236**(2):420-426.
8. Benjamin M, Aradi Y, Shreiber R: **From shared data to sharing workflow: merging PACS and teleradiology**. *Eur J Radiol* 2010, **73**(1):3-9.
9. NEMA NEMA-: **Digital Imaging and Communications in Medicine (DICOM) - Part 3**. In: *Information Object Definitions.*
10. Seng WK, Kim MH, Besar R, Salleh F: **A Secure Model for Medical Data Sharing**. *International Journal of Database Theory and Application*, **1**(1):45-52.
11. Bennett S, Bhuller M, Covington R: **Architectural Strategies for Cloud Computing**. In.: Oracle White Paper; 2009.
12. Rosenthal A, Mork P, Li MH, Stanford J, Koester D, Reynolds P: **Cloud computing: a new business paradigm for biomedical information sharing**. *J Biomed Inform* 2010, **43**(2):342-353.
13. **Amazon Web Services LLC. 2009. Case Studies: TC3 Health** [http://aws.amazon.com/solutions/case-studies/tc3-health/] Available in: May 2011
14. **Gmail** [www.gmail.com] Available in: June 2011
15. **Dropbox Service** [www.dropbox.com] Available in: June 2011
16. **Google Docs** [http://docs.google.com] Available in: June 2011
17. Rimal B, Choi E: **A Conceptual Approach for Taxonomical Spectrum of Cloud Computing**. In: *Ubiquitous Information Technologies & Applications, 2009 ICUT '09 - Proceedings of the 4th International Conference Fukuoka*: 1-6.
18. **Amazon Elastic Compute Cloud** [http://aws.amazon.com/ec2/] Available in: June 2011
19. Linthicum DS: **Cloud Computing and SOA Convergence in Your Enterprise: A Step-by-Step Guide**: Addison-Wesley Professional; 2009.
20. **Windows Azure Platform** [www.microsoft.com/windowsazure/] Available in: June 2011
21. **Rackspace Hosting** [http://www.rackspace.com/] Available in: June 2011
22. **Amazon SQS** [http://aws.amazon.com/sqs/] Available in: June 2011
23. **PubNub** [http://www.pubnub.com/] Available in: June 2011
24. **Google App Engine (GAE)** [http://code.google.com/appengine/] Available in: June 2011
25. **Cloud Storage Initative** [http://www.snia.org/cloud] Available in: April 2011
26. Association SNI: **Cloud Data Management Interface v.1.0.1h**. In: *Overview of Cloud Storage.* 2011.
27. **Cloud Computing Interoperability Forum (CCIF)** [http://www.cloudforum.org/] Available in: June 2011
28. **Open Cloud Computing Interface (OCCI)** [http://occi-wg.org/] Available in: June 2011

29. **Open Grid Forum (OGF)** [http://www.gridforum.org/] Available in: June 2011
30. **Open Cloud Consortium (OCC)** [http://opencloudconsortium.org/] Available in: June 2011
31. Teng C, Mitchell J, Walker C, Swan A, Davila C, Howard D, Needham T: **A Medical Image Archive Solution in the Cloud**. In: *IEEE International Conference on Software Engineering and Service Sciences (ICSESS 2010); Beijing, China*: 431-434.
32. CIHI: **Health Care in Canada 2009: A Decade in Review**. In. Ottawa, Ontario: Canadian Institute for Health Information; 2009.
33. Wirth S, Treitl M, Mueller-Lisse UG, Rieger J, Mittermaier I, Pfeifer KJ, Reiser M: **Hard disk online caches in picture archiving and communication systems archives: how big is beautiful?** *European radiology* 2006, **16**(8):1847-1853.
34. Philbin J, Prior F, Nagy P: **Will the Next Generation of PACS Be Sitting on a Cloud?** *Journal of Digital Imaging*:1-5.
35. Chen C, Wang W: **Implementation of a Medical Image File Accessing System on Cloud Computing**. In: *Computational Science and Engineering (CSE) - 2010; Hong Kong, China*.
36. Yang C, Chen C, Yang M: **Implementation of a medical image file accessing system in co-allocation data grids**. *Future Generation Computer Systems* 2010.
37. Pohjonen H, Kauppinen T, Ahovuo J: **ASP archiving solution of regional HUSpacs**. *Eur Radiol* 2004, **14**(9):1702-1706.
38. Huang HK: **PACS and imaging informatics: Basic Principles and Applications**: Wiley; 2004.
39. Liu BJ, Cao F, Zhou MZ, Mogel G, Documet L: **Trends in PACS image storage and archive**. *Comput Med Imaging Graph* 2003, **27**(2-3):165-174.
40. Sharma A, Pan T, Cambazoglu BB, Gurcan M, Kurc T, Saltz J: **VirtualPACS--a federating gateway to access remote image data resources over the grid**. *J Digit Imaging* 2009, **22**(1):1-10.
41. Liu BJ, Zhou MZ, Documet J: **Utilizing data grid architecture for the backup and recovery of clinical image data**. *Comput Med Imaging Graph* 2005, **29**(2-3):95-102.
42. Smith EM, Migration D, Recovery D: **Storage Management: What Radiologists Need to Know**. *Applied Radiology* 2009.
43. Polónia D: **Electronic market for teleradiology services**. Aveiro; 2011.
44. **jclouds: multi-cloud library** [http://code.google.com/p/jclouds/] Available in: June 2011
45. **Amazon Webservices (AWS)** [http://aws.amazon.com/] Available in: 2011
46. **Amazon SimpleDB Documentation** [http://aws.amazon.com/documentation/simpledb/] Available in: June 2011
47. **Ajax Push Engine** [http://www.ape-project.org/] Available in: 2011
48. Fielding RT: **Architectural Styles and the Design of Network-based Software Architectures - CHAPTER 5: Representational State Transfer (REST)**. IRVINE: UNIVERSITY OF CALIFORNIA; 2000.
49. **Java Simple Plugin Framework** [http://code.google.com/p/jspf/] Available in: May 2011
50. Bieberstein N, Bose S, Fiammante M, Jones K, Shah R: **Service-Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap**. In.: IBM Press: 5.1.5.
51. DICOM-P3: **Digital Imaging and Communications in Medicine (DICOM), Part 3: Information Object Definitions**. In.: National Electrical Manufacturers Association; 2001.
52. Bastião L, Costa C, Oliveira JL: **A PACS archive architecture supported on Cloud services**. *International Journal of Computer Assisted Radiology and Surgery Springer* 2011.
53. Bastião LA, Costa C, Oliveira JL: **A PACS Gateway to the Cloud**. In: *CISTI*. Chaves; 2011.
54. **dcm4che sourceforge project** [http://sourceforge.net/projects/dcm4che/.] Available in: June 2011
55. **RESTlet framework** [www.restlet.org] Available in: June 2011
56. DICOM-P7: **Digital Imaging and Communications in Medicine (DICOM), Part 7: Message Exchange**. In.: National Electrical Manufacturers Association; 2009.
57. Schutze B, Kroll M, Geisbe T, Filler TJ: **Patient data security in the DICOM standard**. *Eur J Radiol* 2004, **51**(3):286-289.
58. L. Bastião CC, and J. L. Oliveira: **A Secure PACS Cloud Archive**. In: *CARS 2011 - International EuroPACS Meeting Berlin*.
59. Ribeiro LS, Bastião L, Costa C, Oliveira JL: **EMAIL-P2P GATEWAY to Distributed Medical Imaging Repositories**. In: *HealthInf 2010; Spain, Valencia*.
60. **Amazon Simple Storage Service** [https://s3.amazonaws.com/] Available in: June 2011
61. **Google Storage for Developer** [http://code.google.com/apis/storage/] Available in: June 2011
62. **Osirix DICOM Viewer** [http://www.osirix-viewer.com/] Available in: 2011

63. **dcmtk** [http://dicom.offis.de/] Available in: 2011
64. **Conquest DICOM Software** [http://ingenium.home.xs4all.nl/dicom.html] Available in: June 2011
65. Costa C, Ferreira C, Bastião L, Ribeiro L, Silva A, Oliveira J: **Dicoogle - an Open Source Peer-to-Peer PACS**. *Journal of Digital Imaging* 2010:1-9.

10 Appendix

10.1 PACS Cloud archive

10.1.1 PACS Cloud Gateway architecture

The PACS Cloud Gateway architecture contains 4 different packages, as described below:

- **Server DICOM**: Contains DICOM Services: Verification, Storage and Query/retrieve. It also encompasses the auxiliary classes to provide interaction with Cloud services.
- **Core:** It stores all settings and POJO (Plain Old Java Object) classes. It contains a thread mechanism to monitor new requests/signaling from Master Index, but it also holds download manager in order to overcome the download single thread.
- **GUI (Graphical User Interface):** PACS Cloud Gateway runs graphically on the system tray (platform independent). It contains the system tray graphic material and configuration window; it is able to add/edit or remove DICOM nodes.
- **Clients: A**ll classes that communicate with Master Index, and Cloud Slaves are implemented in this package.

We have defined an abstract wrapper to communicate with Master Index dubbed MasterIndex (Figure 8.1). It does not supply any implementation, but holds the interface to the method implementation, as known as adapter pattern. Thus, the real implementations were done using RESTful client web services through the RESTlet framework, in MasterIndexImpl. Following this principle, the interface can be implemented using other frameworks or other kinds of providers, for instance SOAP (Simple Object Access Protocol).

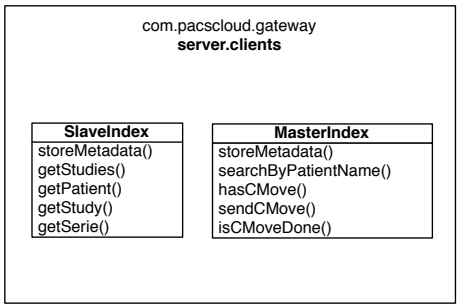


Figure 8.1: Class diagram – client package

One of the biggest concerns in the architecture was the support of multiple providers. In the first stage it looks like a technological problem, but it is not. In order to solve this problem, we have defined our own class definition, so that it would become easier to

implement it, in other cloud players. In client package, the class SlaveIndex (Fig.5) is just an adapter to the real implementation, like MasterIndex.

On the server side, there are a couple of classes focusing on creating DICOM services listener (Figure 8.2), then forwarding the messages or DICOM objects to the Cloud Slaves blobstores and respective metadata for Slaves databases. It also needs to communicate with the gatekeeper, i.e., Master Index, to request other kinds of information like PatientName, study session keys, etc.

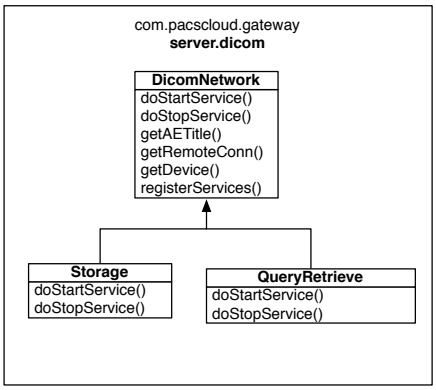


Figure 8.2: Class diagram - DICOM package

Besides Storage and QueryRetrieve classes, there are also other classes such as FindRSP and CMoveRSP, which are DIMSE responses. These responses were created using the current state of the PACS Cloud databases. The solution also implements upload and download in multithread towards a better performance in the storage/retrieval of medical data. It contains a thread poll to limit the number of threads alive and this value might be changed on gateway configurations.

In order to improve request-handling to/from slave database, an abstraction to DIM structure was created (Figure 8.3), which is maintained in a separated project (PACS Cloud Framework).

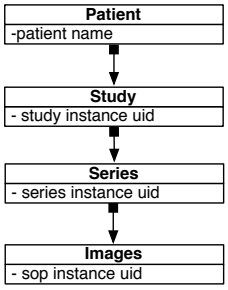


Figure 8.3: DIM - class representation

The initial settings are loaded from a class named start up, living in the core package. In the first stage it will validate gateway login, and then request settings, e.g. PACS's AE Title.

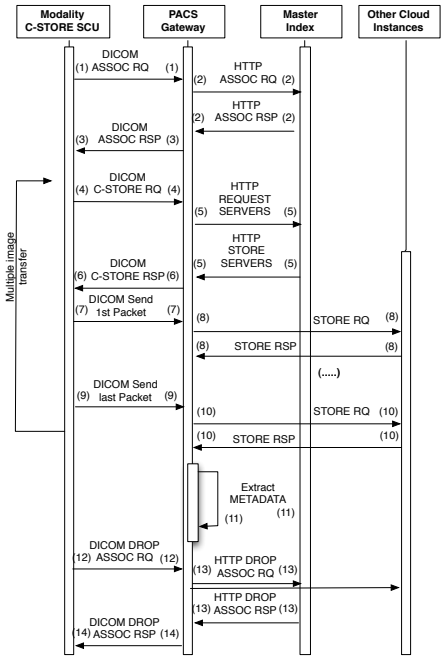
Finally, the gateway, by itself, encompasses configurations. For instance, the TCP ports where DICOM services listen, the username and password, and MasterIndex endpoint. All those configurations can be changed through the GUI or in the config.xml that is created automatically in the first time the application start.

10.1.2 Workflows

In this section we will describe each workflow: storage, query and retrieve.

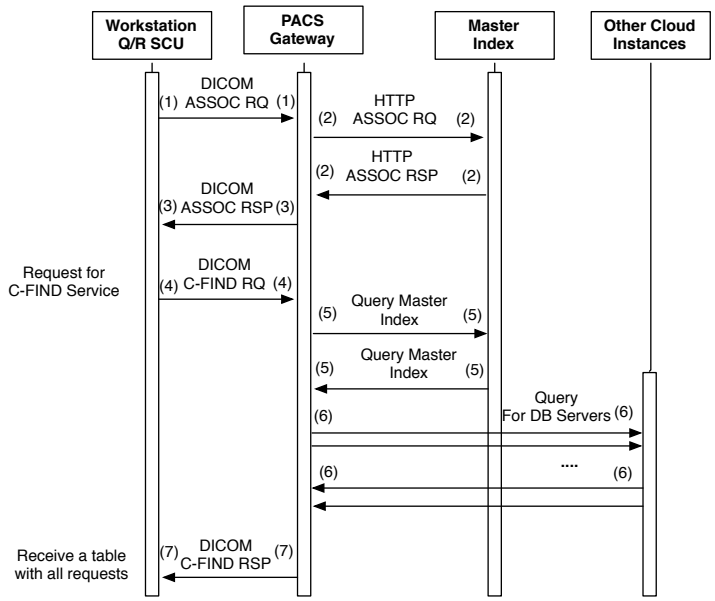
Storage Workflow:

1. Modalities produce studies and send the study using DICOM communications with C-STORE command. Before this happens the modality needs to know which kind of studies are supported by PACS. The first step of this procedure is to send a "DICOM ASSOC REQUEST" to verify if server supports SOP Instance UID.
2. PACS Gateway receives a DICOM Association Request and will ask for core system about settings. Master Index will check the configurations and retrieve the answer. This configuration can be already stored in cache, i.e. transfer syntaxes and SOP classes.
3. PACS Gateway with gathered information collects a DICOM Association Response command.
4. The invoking modality (SCU) issues a C-STORE service request to the PACS Gateway (SCP).
5. PACS Gateway invoking issues a C-STORE service request (using HTTP/XML protocol) to the Master Index/Cloud Controller. This message also asks for a Token (i.e. Server, Protocol and Credentials) where information will be stored. Master Index receives C-STORE service request and issues a C-STORE response to the PACS Gateway. The message to servers to store information is sent. Once again, this settings can be already stored in local cache.
6. C-STORE Response is sent from PACS Gateway to Modality/DICOM Client.
7. DICOM files are sent from Modality to PACS Gateway. The image file is locally stored.
8. DICOM chunks are stored in previous servers. When a chunk is stored acknowledgment message is sent back.
9. Last DICOM packet is sent to PACS Gateway.
10. DICOM chunks are stored in previous servers. When a chunk is stored an acknowledgment message is sent back.
11. PACS Cloud Gateway sends the metadata to the DB Servers (i.e. Master Index and Slave Databases). A chunk of data will have a reference to the server where it was stored.
12. Modality closes Association sending a Request.
13. PACS Gateway closes session with Master Index. Master Index sends an acknowledgment back.
14. DICOM Association response is sent and all connections are closed.



Query Workflow:

1. The first step of this procedure is to send a "DICOM ASSOC REQUEST" to verify if the server supports service.
2. PACS Gateway receives a DICOM Association Request and it will ask for Master Index about settings. Master Index receives the request for C-FIND Service, with specific AE, and it will accept or reject association. A response message is sent back to PACS Gateway.
3. PACS Gateway sends DICOM Association Response with the answer to previous message.
4. The workstation's Q/R application issues a C-FIND request to the PACS Cloud Gateway.
5. Master Index receives the C-FIND request from the querying workstation and sends a query for the Master Index.
6. PACS Gateway sends queries for each instance of servers. Other instances of Cloud/Database servers in cloud will answer with results.
7. PACS Gateway receives the list of results (of other instances), collect the results from Master Index and Cloud slaves and returns it in a DICOM C-FIND Response. Afterwards, the association is closed.



Retrieve Workflow:

1. The first step of this procedure is to send a "DICOM ASSOC REQUEST" to verify if server supports service.
2. PACS Gateway receives a DICOM Association Request and it will ask for Master Index about settings. PCCS receives the request for C-MOVE Service, with specific AE, and it will accept or reject association. A response message is sent back to PACS Gateway.
3. PACS Gateway sends DICOM Association Response with the answer to previous message.
4. The user at workstation's Q/R application selects interesting studies and issues a C-MOVE request to the Master Index.
5. PACS Gateway receives C-MOVE request and issues Master Index with a MOVE REQUEST.
6. Master Index sends a signal/asynchronous event to the PACS Cloud Gateway with the specific AETitle (received in previous C-MOVE request). It signals the gateway has data to get in Master Index. The servers where PACS Gateway will download data are sent in signal message.
7. PACS Gateway downloads the images from servers.
8. PACS Gateway issues the AETitle with a C-STORE SCU to retrieve the images. All images are transferred to destination.
9. When all data are transferred "PACS Gateway receiver" sends a message to Master Index, and Master Index sends the same message to Gateway.
10. C-MOVE RSP is sent to finish the process.
11. After transferring the last image Workstation issues a "dropping association request" to terminate the process.
12. DICOM association terminates.