**Universidade de Aveiro**
**2013**
Departamento de Eletrónica, Telecomunicações e Informática

**TIAGO MARQUES GODINHO**

# DISTRIBUTED PACS: PERFORMANCE AND AVAILABILITY

# ASPETOS DE DESEMPENHO E DISPONIBILIDADE EM AMBIENTES PACS DISTRIBUÍDOS

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 Eletrónica do Departamento de Eletrónica Telecomunicações e Informática da Universidade de Aveiro

Este trabalho é dedicado à minha irmã, por todo o carinho, compreensão e amizade incondicional que fazem com que a próxima etapa seja apenas mais uma.

**o júri**

presidente

Prof. Dr. António Manuel Melo de Sousa Pereira
professor Catedratico, Universidade de Aveiro

Prof. Dr. Rui Pedro Sanches de Castro Lopes
professor Coordenador, Dep. Informatica e Comunicacoes da Estg do Instituto Politecnico de Braganca

Prof. Dr. Carlos Manuel Azevedo Costa
professor Auxiliar, Universidade de Aveiro

**agradecimentos**

Um agradecimento muito especial à minha família, por me terem sempre proporcionado alcançar mais esta etapa.

Aos meus amigos, que me acompanharam ao longo destes cinco anos, um muito obrigada, este documento também era impossível sem eles.

A todos os membros do grupo de bioinformática, pelo espetacular ambiente de cooperação. Em particular ao Luís Bastião, sem ele a qualidade e a pertinência deste trabalho não eram as mesmas.

Um agradecimento a todos os professores, que durante toda a minha vida contribuíram para o meu crescimento enquanto pessoa. Em particular ao professor Carlos Costa e ao professor José Luís Oliveira por terem aberto as portas à realização deste trabalho.

**palavras-chave**

Imagem Médica; PACS; Distributido; Web-based PACS; DICOM; Telemedicina; DICOM Router; Performance.

**resumo**

A imagem médica é hoje um meio complementar de diagnóstico fundamental nas instituições de saúde. Historicamente, estes meios têm custos muito significativos para as instituições, quer em aquisição de equipamentos, quer na manutenção da infraestrutura. Numa ótica da redução de custos operacionais e melhoria dos processos, as instituições médicas tem explorado os avanços na área das Tecnologias da Informação, com o objetivo de melhorar a aquisição, arquivo, distribuição e visualização dos estudos. Estes sistemas, denominados como PACS, começaram por se impor no interior das instituições. No entanto, a tendência atual é para interligar essas redes, criando ambientes de trabalho geograficamente distribuídos. O bom desempenho destas redes é fundamental para suportar a prática clínica, nomeadamente, no que diz respeito à latência associada às comunicações. Assim, é vital desenvolver tecnologias para melhorar o desempenho, segurança e robustez destas redes. Esta dissertação propõe e avalia um conjunto de soluções tecnológicas que objetivam melhorar a utilização de PACS em ambientes distribuídos.

**keywords**

Medical Images; PACS; Distributed PACS; Web-based PACS; DICOM; Tele-medicine; DICOM Router; Performance.

**abstract**

Nowadays, medical imaging is used as a primary method of diagnosis in healthcare institutions. Typically, those environments have huge costs related to acquisition equipment and infrastructure maintenance. In order to reduce costs and improve workflows, healthcare institutions have been exploring new information technologies to support the acquisition, storage, distribution and visualization of medical imaging studies. Those systems, denominated as PACS, are very used inside the institutions' networks. However, the actual tendency in PACS is to interconnect multiple institutional systems, thus creating geographically distributed medical imaging networks. The performance of these architectures must not delay or deteriorate the medical processes. As a result, the development of new technologies is fundamental to improve the performance, safety and reliability of these architectures. This thesis proposes and assesses a set of technological approaches that aim to improve PACS deployment and utilization in distributed environments.

# Index

# Figure Index



# Table List

# Acronyms

| Term | Description |
|---|---|
| **ACL** | Access List |
| **API** | Application Programming Interface |
| **CR** | X-Rays |
| **CT** | Computed Tomography |
| **DICOM** | Digital Image Communications in Medicine |
| **DIMSE** | DICOM Message Service Elements |
| **ePR** | Electronic Patient Record |
| **HIS** | Hospital Information System |
| **ICE** | Interactive Connectivity Establishment |
| **IT** | Information Technologies |
| **MR** | Magnetic Resonance |
| **NAT** | Network Address Translation |
| **PACS** | Picture Archive and Communications System |
| **PPSE** | Posterior Playfair Searchable Encryption |
| **QoS** | Quality of Service |
| **RIS** | Radiology Information System |
| **SCP** | Service Class Provider |
| **SCU** | Service Class User |
| **SE** | Searchable Encryption |
| **SIP** | Session Initiation Protocol |
| **SPoF** | Single Points of Failure |
| **TLV** | Tag Length Value |
| **US** | Ultrasounds |
| **WADO** | Web Access to Data Objects |

# 1. Introduction

This chapter provides an introduction about the thesis. It gives a glimpse on the main issues in the medical imaging networks and how they influenced the proposed work. Finally this chapter references the structure and the main goals of this document.

## 1.1. Overview

Over the last few decades information and communication technologies have played a key role in the way society interacts with the world, such as, the way people have access to entertainment, food, transports, information and other commodities in their daily life. Healthcare industry is not an exception to this trend and have followed the general evolutionary tendencies in the technologies and IT systems. Health related institutions have been increasingly providing new IT services to patients, such as ePR (electronic Patient Record). Moreover, other non-directly related to health industry corporations have also been showing great interest in the health information field and have been offering health related services to patients and healthcare institutions.

On an intra-institution level, health institutions have also been showing great interest in information systems to manage their business processes. Health information systems, such as, HIS (Hospital Information System) and RIS (Radiology Information System) have been increasingly used in the past decade. Medical Imaging related processes have also been spotted for digital information systems. According to [3, 4], in Portugal, 100% of the medical institutions use computers and internet connections to support medical practice.

PACS (Picture Archive and Communications System) manage the digital image workflow in institutions. The medical image workflow are associate with two major tasks, the archiving of images for posterior use and the distribution of images, since some physicians do not practice in the same department where images are acquired or archived.

As digital medical images produce huge amounts of data, image storage and distribution are often associated with significant costs both monetary and time related [5, 6]. These costs are severely aggravated as digital imaging usage is constantly increasing, and even small institutions may produce and consume a great number of medical studies. PACS have to deal with these structural constraints, in order to keep the lowest impact on the medical workflow's performance, without consuming significant institutional budget.

A current trend in PACS is focused on the disassociation of the system from the actual institution facilities, i.e. outsourcing the infrastructure that may run over cloud, reducing costs to organizations [7, 8]. However, PACS typically manages huge amount of data, which means huge communication overhead and lacks of performance in distributed environments, often resulting in poor QoS (Quality of Service) and no customer satisfaction.

Therefore, it is of paramount importance that medical repositories over the cloud care about fast communications supporting access to the medical data. This work will reflect these concerns as it will analyze a previously developed system at Universidade de Aveiro/Bioinformatics Group, i.e. the DICOM Cloud Router, searching for performance constraints in order to mitigate them [9]. A great effort has been made in the past years to develop a distributed PACS architecture to support regional environments. A distributed PACS architecture, as it will describe in further chapters, is a system intended to support the medical image

workflow across multiple institution locations. Extending these systems in a distributed manner means multiple PACS instances deployed at the same time probably serving more than an institution simultaneously. The second stage of this work will be based on top of other novel concerns in distributed systems, which are replication and fault tolerance, in order to increase the service availability in the overall system. Although this approach is rather focused in a specific system, the results produced by this work may be extrapolated to other systems, as constraints are similar.

This thesis explores these approaches as starting points to develop strategies to improve the overall quality of distributed PACS, taking performance and reliability considerations in order to propose a refined architecture.

## 1.2. Objectives and contributions

The main goal of this thesis is to provide a fast and effective environment for the medical image workflows across multiple institutions. Current solutions require significant investments both in infrastructures and in work force for network and environment configurations. Moreover they often disregard previous existing infrastructures leading to waste previous investments in resources. Summing up, it discourages the migration to a distributed environment despite its clear benefits.

DICOM Cloud Router is a software platform that requires minimal setup configurations and is designed to integrate previously existing resources such as PACS Archives and viewer applications. There are also minimal infrastructural requirements as it enables reuse of preexisting infrastructures, such as, Internet connection and PACS components.

Apart from providing easy set-up environment, DICOM Router architecture must be also competitive in terms of performance and availability, like any other Intranet solution. This dissertation intends to extend DICOM Router architecture to achieve high performance and availability.

Rather than focusing on specific solutions for the existent DICOM Router architecture, the taken approach starts by identifying the most relevant processes in a generic PACS workflow and then suggests solutions to mitigate found problems and correct erratic behaviors when extending these systems to a distributed environment.

Performance and QoS in data access often passes to provide replication of data. Replicated data has two very important aspects. Firstly, data is less likely to be lost during the system life cycle. Secondly, multiple sources exists making it possible to be retrieve data from multiple locations in parallel thus sharing the load of the system and contributing to lower delays and better speeds in data transferences.

In order to improve the image data handling process and boost the overall availability of studies, an effective cache system for medical images will be proposed. Moreover, the implemented approach will be integrated in the normal medical image workflow. Lastly a careful analysis of single points of failure (SPoF) will be conducted. The main goal is to create strategies in order to improve the overall availability of the system itself.

Combining these approaches, the ultimate goal of this project will be achieved not only by decreasing the response time of the proposed system to services, such as Query/Retrieve or Storage, but also by the improvement of the overall reliability of the system. Providing a better service to the medical imaging workflow is of course the main concern of this thesis.

## 1.3. Outlines

***Chapter 2:*** Provides a valuable description of the state-of-the-art scenario in the medical imaging distribution, as well as in technologies related to this thesis.

***Chapter 3:*** Provides a description of previous efforts that have led to the proposal of this thesis.

***Chapter 4:*** Provides a detailed description of the contributions carried out in this thesis. Namely there are described multiple methods proposed to improve the performance and availability of our distributed PACS Architecture.

***Chapter 5:*** This chapter presents the trials conducted to validate the proposed methods. Along with the trials results there is also a discussion about these methods contributions.

***Chapter 6:*** This chapter resumes the overall conclusions resultant from this thesis. Moreover, it appoints the directions of further contributions to our distributed PACS architecture.

***Chapter 8:*** An appendix with relevant diagrams to the full understanding of the proposed methods.

# 2. State of the Art

This chapter has the purpose of referencing the state of the art in systems and technologies related to medical imaging, such as, PACS (Picture Archive and Communication System) and DICOM (Digital Image Communications in Medicine). The current trends will be enumerated and discussed, both in industry and research in the areas of Medical Image Archives, Cloud Computing, among others.

The reading of this chapter is very important to understand not only the background environment of this document, i.e. storage and distribution of medical images and the workflow itself, but also to understand the taken decisions in each step of the project as they were very influenced by the constrains of medical imaging processes.

## 2.1. Digital medical imaging laboratory

Medical imaging is defined as the technique and process used to acquire visual representation of the Human body [10]. The medical imaging field is often associated with other subareas, such as Radiology or Nuclear Medicine, as they are techniques used to produce the medical images. Medical imaging has been used for quite long time as a primary and complementary method for diagnosis purpose. Among the most well-known modality types are, X-Rays (CR), Computed Tomography (CT), Magnetic Resonance (MR) and Ultrasounds (US).

Over the last century the medical images were printed in films, as most acquisition equipment were analogic. Nevertheless the Archive and Distribution of these images was associated with a few well know constraints. For instance, they must have been available for physician's access on-demand. Being an analogic media, the costs associated to storage logistics and the access times were both high. These constraints were imposed by the medical practice. As such they still apply to the current state-of-the-art paradigm of medical image Archive and Distribution.

In the developed countries it is almost impossible to find an individual who has not been subjected to a medical imaging examination. Frost & Sullivan [6] forecasted that 1 billion of medical imaging procedures will be conducted in United States of America in 2013. This impressive reality is only possible due to the great contribution of digital era, as PACS and digital acquisition equipment (i.e. modalities) started to appear in the market. The use of this equipment has been generalized, as prices tend to decrease, including the costs of PACS maintenance than are lower than analogic solutions.

Nevertheless the digital version of these systems provides a clear benefit when compared to the analogic ones. The current state of the art solutions are still trying to cope with the same constraints that were mentioned previously for the analogic systems, namely due to the tremendous volumes of data produced in our days and the distributed environments requirements. In fact, the state of the art solutions have not only to deal with these constraints but also with legal and security considerations about holding sensitive patient data in digital repositories.

The constraints imposed by the medical workflow itself are a key point to understand the importance of current research fields in the medical image archives and distribution systems. As

in the old approach with analogic films, the physician often requests examinations to diagnose patients implying that produced images must be moved from archiving system to the physician workstation, which might be at home. There are several medical imaging workflows that impose organizational constraints. For instance, it is hard to share studies with multiple organizations or even with different departments within the same organization.

The advent of digital acquisition equipment and the digital communication networks, such as, Internet and Cloud Based Computing have opened the door to PACS distributed environments and services outsourcing. As these approaches tend to reduce the medical workflow delays thus providing better service to patients and controlled costs to organizations because the exchange of previously performed studies can in many cases avoid purchasing a new one.

## 2.2. PACS (Picture Archive and Communication System)

As digital technological developments had been achieved in the field of medical imaging acquisition devices, the price and thus the cost-benefit ratio of these digital devices surpassed the analogical ones. Thus, with this growth, the need to find an information system that could bring the medical imaging archives and the distribution networks to the digital era grew as well.

PACS stands for Picture Archive and Communication System. It clearly defines a set of hardware, software and communications technologies for acquiring, storing, distributing and analyzing digital medical images in a distributed information system environment [8].

As [1] states, PACS may be spit in three major sequential steps: Acquisition, Distribution and Visualization. Figure 2.1 illustrates these steps.

Acquisition is the process of medical images production. There are two major methods associated with medical image acquisition. The first method is the one that the images are produced by digital acquisition equipment directly through examination procedures.

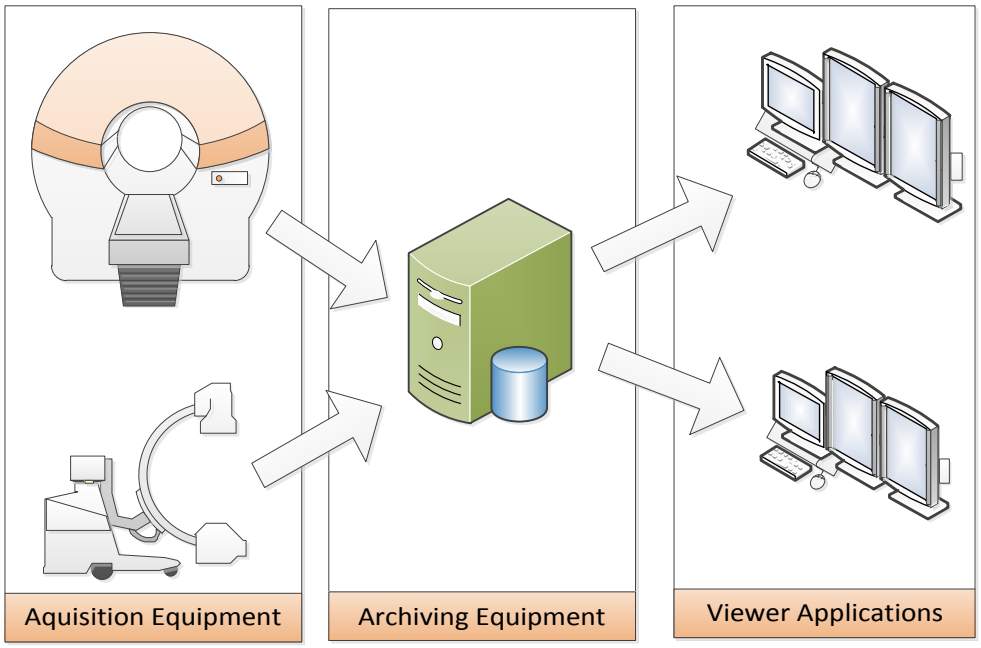



**Figure 2.1: Big Picture of PACS in the Medical Image Environment. Adapted from [2]**

Although this digital equipment are now industry standard, compatibility with outdated equipment and previously archived studies has made scanning mechanisms. These scanners produce a digital image from the analogical films produced by early equipment. This method continues to be very used in our days.

Distribution is the process of exchanging images or studies among the different PACS nodes. It is easy to understand that in most medical imaging studies need to be relocated inside the PACS, as in most situations physicians do not practice in the same department or center that acquired or archived the images. Moreover, the modalities and archive can be in distinct locations. The distribution process opens the possibility for the exchange of studies not only inside the same institution but also among different institutions as well thus creating opportunities to reuse of studies.

Visualization process is the frontend of any PACS. Image visualization is of major importance, as physicians must have an intuitive way of searching and reviewing studies. The use of third-party viewer applications, such as, Osirix [11] or K-PACS [12] is required. The visualization process often requires prior processes of acquisition and distribution of image objects.

After stating these three processes it becomes clear that PACS are a mix of repositories and information systems used to facilitate the workflows in medical imaging environments. It provides better integration for workflows of different stages as well as booting the overall performance of the medical practice. But the goods do not come without a cost. Medical images tend to generate a tremendous amount of data [6] which leaves the PACS Archives to deal with the storage of large volumes of data and the PACS Distribution Layer to keep the communication delays acceptable to the medical process. Moreover, the data overhead is not the only problem, legal issues and Human resistance to information systems also tend to delay the implementation of these systems.

As stated before, dealing with huge amount of data is a complex engineering task and raises some real issues directly related to cost and performance, such as, Backup, Redundancy, Security and Crash Recovery. With the spreading of digital equipment throughout medical institutions [13], including the small ones, these engineering problems are affecting even more entities. A great part of these institutions have economic limitations to support such IT infrastructure, including operational staff. As a result, outsourcing of PACS Archives to external IT contractors has been also a rising trend in the field.

Concrete integration of medical imaging repositories (PACS Archives) inside institutional workflows might vary greatly from an institution to another [1]. Although it follows some common stages, such as, patient registration in the HIS and RIS, the examination procedure, image analysis by physician and image archiving. In [8] the author proposed three general architectures for the workflow in a PACS, the Stand-alone, the Client-Server and the Web-based architectures. These workflows are actually being used in different healthcare units, described in sequel.

The stand-alone architecture involves a store and forward approach, as images acquired during procedures are immediately sent to the image archive and then forwarded to the previously registered workstations. These workstations are used by physicians to analyze and report the examinations. This architecture, although claimed to be very prone to study loses, arise some security considerations as studies are transmitted without asking and prone to be access not only by strictly necessary personnel but virtually anyone in the PACS.

The Client-Server Architecture is the most wide used and proven approach. Studies are uploaded from acquisition equipment to a central repository. Workstations retrieve studies only

when needed, without any pre-fetching strategies like, for instance, moving the studies before there are needed or during the previous night according with programmed events. This introduces a relative delay in the study review process as it wastes time while retrieving those studies, proportional to the available bandwidth for the transference. Nonetheless, it provides a more efficient access control to studies than the previous architecture.

Web-based architectures are (as in many others areas) the current trend in PACS Architectures. Web-based PACS integrate the central archive in a datacenter and provides a frontend (web-based viewer) to operate with images. It is probably the most robust solution In terms of security, bandwidth requisites and portability. Nonetheless, web-based architectures lack in compatibility as users are limited to the web browser environment that as some limitations, such as, 3D reconstructions and may not use any third-party software [14]. Moreover, when deploying a web-based architecture on public cloud environments a few legal issues may arise related to the exploitation of sensitive patient data.

## 2.3. DICOM (Digital Image Communications in Medicine)

In the early years of digital medical imaging, manufacturers of medical imaging equipment had developed their own communications protocol and image file formats. So exchanging images between different vendors equipment was a real issue and, in many cases, it was even impossible. Portability and studies exchange was then very hard to achieve. To solve this issue, in the mid-80s, a consortium formed by NEMA (National Electrical Manufacturers Association) and ACR (American College of Radiology) proposed the creation of standardization in file format, directory structure and communication's protocols for digital medical imaging equipment. The first draft of the standard was named ACR-NEMA 300 and is considered of major importance for PACS proliferation. Nonetheless this primary version had some issues and omissions that were rectified by latter versions of the standard [2].

The third version of this standard was named DICOM 3.0 [15] and its full version was presented in 93. Nowadays, it is the most important standard in medical imaging and PACS. DICOM 3.0 is constantly improved with the addiction of supplements to face the most recent issues in the medical imaging field despite always keeping compatibility with previous versions of the standard.

The proliferation of DICOM compliant[1] equipment has enabled the exchange of digital medical images among different equipment thus providing the possibility of implementing systems (PACS), such as, mentioned previously in this document. The importance of DICOM is recognized by not only IT personal but also medical personal as shown in [2] where a Professor of Radiology states "it has become the driving force behind the entire imaging workflow".[2]

### 2.3.1. DICOM Data

DICOM standard is able to represent all real world data based on the definition of DICOM objects. DICOM Objects are sets of Data Elements. These data elements are the actual representations of real world attributes. DICOM Objects follow the well know Object-Oriented approach and may enclose other objects thus providing extensibility and open-endness.

[1] DICOM Compliant or DICOM Ready is the common designation of equipment or software that support the DICOM Standard.

[2] Vassilios Raptopoulos, M.D, in [2].

DICOM Data Elements [16] are formed by three mandatory fields that seemingly follow a TLV (Tag Length Value) structure. The first field is a tag identifying unequivocally the DICOM Data Element. The second field is the length (in bytes) of the value field. Lastly, the value field encloses the binary data of the element. Encoding of the value field may be done using 27 different codifications called VR (Value Representation). VRs are previously defined in the standard (Part PS3.5 [16]) and are the primary data types in the DICOM standard. Figure 2.2 illustrates the DICOM data elements.

The Data Element tags are composed by two fields, the group identifier and the element identifier (within the group). Both fields are 16-bit unsigned values. The group field identifier identifies the group of the DICOM Data Element. In DICOM, Elements are grouped by similarity as [2] states. Groups often reflect a relation between DICOM Data Elements and real world Object, such as, Patients (0x0010) or Studies (0x0008). An illustrative example of DICOM Data Element Tags is the Patient ID (0x0010, 0x0020) that has the group identifier of the patients group and the 0x0020 as the identifier within the group. See Figure 2.3.

### 2.3.2. DICOM Data Dictionary

There are two possible ways of matching a DICOM Data Element to its VR type. The most common is using the DICOM Data Dictionary. The DICOM Data Dictionary, as its name indicates is an associative memory that matches DICOM Data Elements (by its tag) to a set of attributes including the Element VR, the name of the Element, its multiplicity and element data type. An example of a DICOM Data Dictionary is shown in [2].The second way is by setting an optional field in the Data Element called the VR Field with the identifier of the VR used. As illustrated in Figure 2.2.

As [2] states, the DICOM Standard has a standard Data Dictionary of around 2000 Elements. However a statically predefined Data Dictionary is of no use if vendors ever need a new Data Element to map new attributes introduced by their newest equipment, so private Data Elements can be inserted in order to meet the needs for new Elements.

### 2.3.3. DICOM Object Identification and Hierarchy

Following an Object-Oriented hierarchical approach DICOM Objects are abstract models for real world objects. A representation of a real world object is called an Instance, for example a DICOM Image instance is a DICOM Image Object with real values and attributes that exists in a concrete context (for example in a file system). A corollary of this definition is that the same real world image may have multiple instances in different contexts [2].

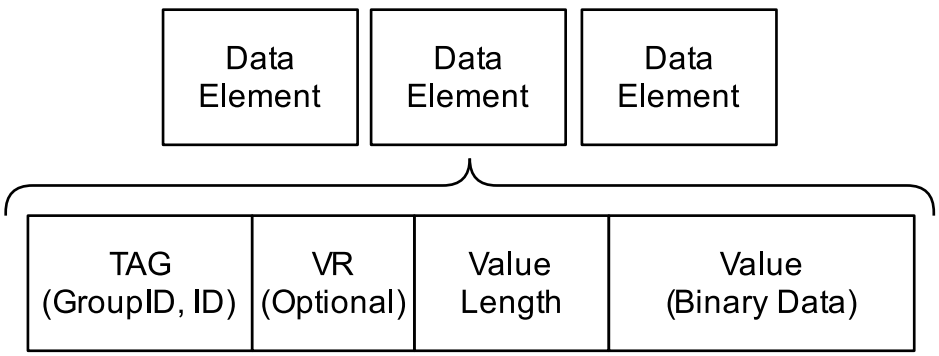



**Figure 2.2: DICOM Data Elements. Adapted from [1].**

In order to identify every instance unequivocally DICOM uses UIDs. UIDs are unique identification keys (numerical) assigned to each DICOM Object. UIDs are formatted in *<org_root>.<suffix>*, where the *org_root* is an organization unique identifier and the *suffix* represents the actual DICOM Object. The suffix normally has attributes that actually ease the process of identifying the object in the real word. This is well explained in [2] where is also given an example of a good identifier for a DICOM Object, based on the patient ID, study ID, date and time.

The identification of DICOM Files opens the door to explain how DICOM organizes information about real world objects associated with the medical practice, such as, patients and studies. O.S Pianykh [2] states that DICOM Objects are organized hierarchically by patients, followed by studies then series of images and eventually a single image, every node representing a DICOM Object (as shown in Figure 2.3). This hierarchy represents very well the real world constraints where a patient might have multiple studies and the single images are eventually related to studies and thus patients.

### 2.3.4.DICOM Upper Layer – Services and Commands

DICOM Communications between DICOM compliant applications is of paramount importance to digital medical imaging and therefore to PACS. Current DICOM standard specifies that the DICOM UL (DICOM Upper Layer Protocol) is located in the application Layer of the OSI Model. Therefore it has to use a transport protocol which in this case is the well-known TCP/IP. TCP/IP communication provides major compatibility not only between all sorts of equipment but also with existing communications infrastructures both inside and out of medical institutions.

DICOM UL identifies applications through the use of *AETittles*. Therefore applications are identified with the triplet *IP Address, Port* ***and*** *AETitle* which makes possible running multiple DICOM Applications in the same host machine. The DICOM UL also defines the concept of associations between pairs of DICOM Applications. Associations are bound with connection parameters, such as, image compression codecs that are negotiated between both applications [17].

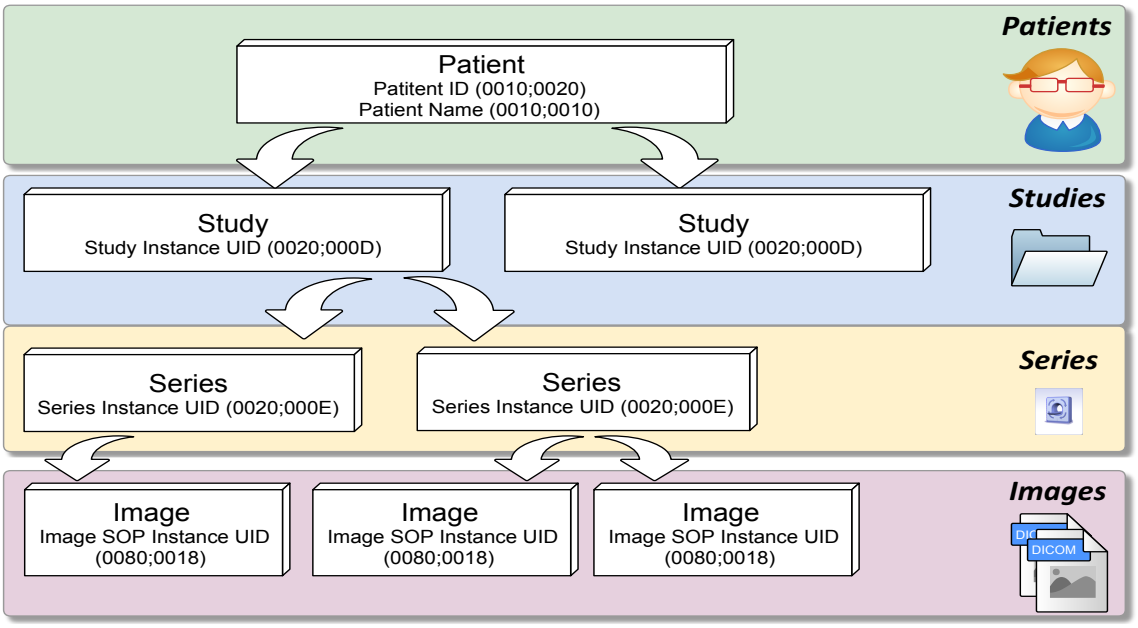



**Figure 2.3: DICOM Hierarchic Data Structure. Adapted from [2].**

DICOM standard follows a client-server architecture. It defines a set of high level services messages named DIMSE (DICOM Message Service Elements) (Part PS3.5 [16]). DICOM Applications can have one or two roles, which in DICOM both are defined as Service Classes. The Service Class Provider (SCP) is inherent from the client-server Architecture SCP (servers). The SCP provides a set of predefined services to other client applications. In its turn, the client applications are called Service Class Users (SCU).

DICOM Services are previously defined in IODs and are more easily defined as sequences of DICOM Commands. DICOM defines a wide range of services, such as Storage, Query, or Printing. Although for the sake of simplicity this document will majorly refer the Query/Retrieve and Storage services as they are the primary scope of the PACS Image Distribution Layer, and therefore they impose performance constraints to the system.

### 2.3.5.Storage Service Class

The Storage Service is typically provided by PACS archives. As the name suggests Storage Services goal is to store an image in the repository. A typical use case of this service is when acquisition equipment needs to send the recently acquired image to the central repository. Thus Storage Service Class is of major importance in PACS.

Storage Service is composed by a single C-STORE command per image. The workflow of Storage Services is easy to explain. Firstly the SCU sends the C-Store-Request message to the SCP (possibly the PACS Archive) containing the DICOM Object to be transferred, upon reception the SCP replies with a C-Store-Response message acknowledging the reception of the data. An illustrative example is shown in Figure 2.4.

### 2.3.6.Query / Retrieve Service Class

The Query-Retrieve Service is often used by physician's viewer applications to search and download studies from the PACS Archive to perform revisions. The Query Service supports the search of objects (Patient, Studies, Series and Images) inside a repository. They might be searched based on various DICOM Data elements, such as, patient name, study date, modality, accession number and few others. The Retrieve service is often executed after a Query Service. It aims to retrieve the desired objects from the repository based on a query, normally by objects UIDs.

Query Retrieve Service class is composed by two commands, the C-Find and the C-Move. A C-Find Command, as the purpose of query the archive (SCP) about the existence of studies or

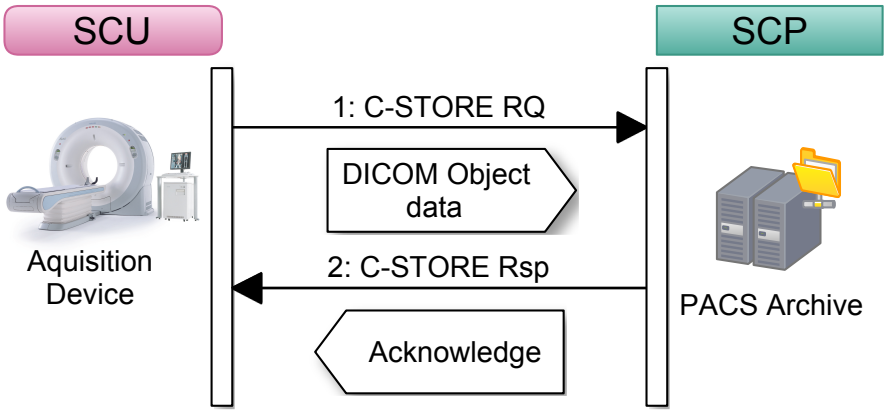



**Figure 2.4: DICOM Storage Service procedures. Adapted from [1].**

images with the desired properties specified by a C-Find-Request command, like for instance "PatientName=A*" for all patients that starts by the A character. One or more C-Find-Response would be sent back to the SCU for each matched object. Finally a C-Find-Response signaling the termination of the C-Find service will be sent in the end. This workflow is well represented in Figure 2.5.

The retrieval of studies uses the C-Move command. A C-Move-Request message is sent to the SCP identifying the desired objects. Then, the SCP issues a C-Store command for each object that needs to be transferred to the SCU. A C-Move-Response is sent when the last object is transferred signaling the termination of the process, as shown in Figure 2.6. Furthermore the Retrieve Service SCU typically provides a Storage Service in order to receive the objects via the C-Store. However, the C-Move Request can also move a study to third party machine, i.e. different machine that requested the study. It is important to keep in mind that a C-Store command is issued for each image to be transferred. As it will be shown further in this document this might not be the most effective way of transferring studies with a great number of images.

Note that for the purpose of performance optimization in a PACS environment the C-Move and thus C-Store commands have crucial importance. Query/Retrieve is not only one of the most used services in a PACS environment but also is normally included in middle of the medical practice workflow where the delays should be the lowest as possible.

### 2.3.7.DICOM WADO

Latest versions of DICOM Standard have introduced WADO (Web Access to DICOM Objects) [16]. WADO is the most recent initiative to take DICOM Standard to the web. As stated previously stated, DICOM Communications Protocol is on the application layer of the OSI model as such full connectivity can only be achieved if the network environment recognizes DICOM protocol as a legit application and thus provide full access to the network.

Unfortunately, most private network administrators tend to apply security measures due to the policy restrictions, such as, firewalls to better secure their network assets. DICOM Protocol is often not recognized and thus blocked in most private network environments.

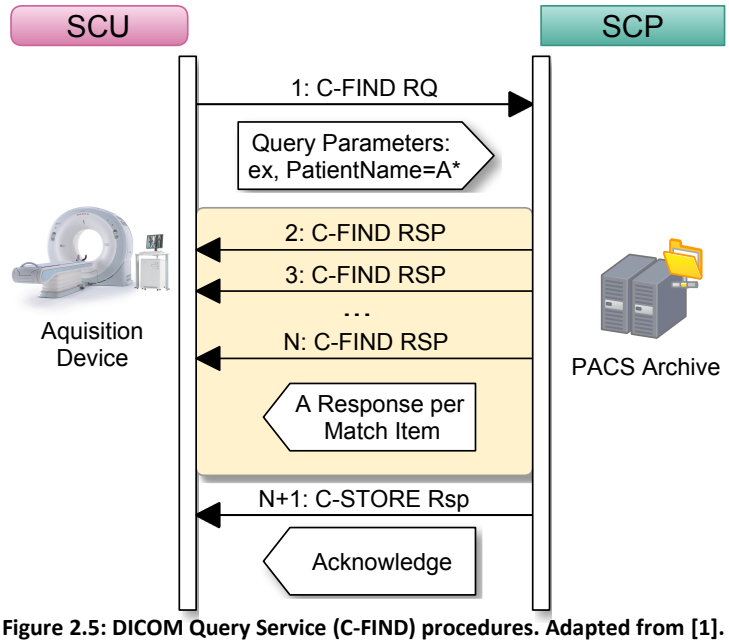



**Figure 2.5: DICOM Query Service (C-FIND) procedures. Adapted from [1].**

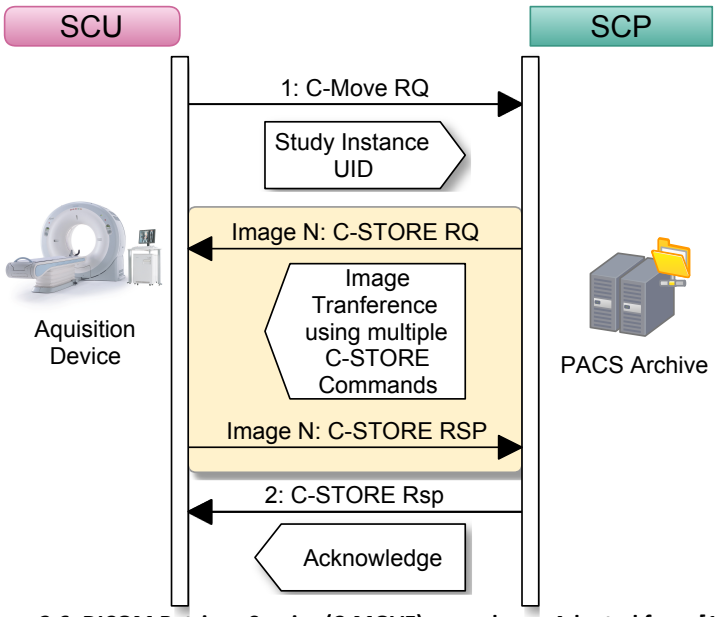



**Figure 2.6: DICOM Retrieve Service (C-MOVE) procedures. Adapted from [1].**

In spite of using standard DICOM Communication Protocol, WADO uses the well-known HTTP protocol to provide access to DICOM Objects and Services. HTTP Traffic is normally accepted in every network so DICOM information using HTTP protocol can effectively cross organizations boundaries.

As [18] states, WADO does not provide all the standard DICOM Services, namely content discovery analogous to C-Find standard command. Moreover, WADO implements other services related to the transformation and rendering of images which clearly indicates that WADO was not intended to be served as a web interface to access PACS Repositories but rather an extension to provide more functionalities for tele-radiology applications. The Related Work 2.6 section provides a brief description of WADA [19] which intended to extend the WADO enabling C-Find like queries to archives.

## 2.4. Cache Systems

Caches are widely used across every IT system. Cache consists in creating a temporary memory to store certain objects that have a high probability of being access in a short period of time compared with the normal repository. The idea is based on the assumption of retrieving an object from this temporary memory should be significantly faster than retrieving the same object from its original source. Caches have been used for in many scenarios, even before the Web 2.0 technologies took place over the IT field. These systems have been largely used namely in computer architectures.

Cache Systems are often associated with multiple hierarchical levels, as different storage devices often have different specifications. Namely storage capacity, persistence, retrieval speed and latency, as it is well represented in [20]. For example, hard disk based cache have greater capacity but severely lack performance when compared to memory (RAM) based cache.

Distributed systems have taken the use of caching technologies very seriously. Since communications inside a computational system are much faster than in LAN environments. Also

much faster compared with the WAN environments, and so, further cache levels were introduced combining cache instances deployed at different nodes of the system.

Distributed systems can be split in two major groups: client caches and distributed caches. Client caches (or local caches) are deployed locally to in each distributed system user. They provide extreme acceleration for retrieval of objects although they are often very limited considering the storage volume. Thus, they are only able to cache a very small portion of the distributed system data. Distributed Caches are deployed on the distributed infrastructure instead of its end users. Although they do not provide as much acceleration compared with local caches but they often cache more volume of content.

Nevertheless, Cache Systems are not only associated with benefits. Actually the exploration of caches involves difficult engineering problems. These problems are associated with the integrity of cached objects in the different hierarchical levels of the system and with strategies associated with the population and eviction of those objects.

According to [20] there are two major trends for data management in hierarchical multi-level caches. The inclusive management, where higher levels, with greater capacity, enclose the cached objects in lower levels. The exclusive management were cached objects are generally only present in a single level, providing better usage of cache capacity at the expense of harder population/eviction strategies.

Cache population strategies are associated with the insertion of objects into the cache. Cache population can be done expressly, resorting to pre-fetching techniques, or it can be executed along with the life-time of the cache, taking advantage of the client's work-flow. In its turn, Cache eviction is associated with the removal of less important objects, making room for more important ones. It is normally associated with eviction policies although users can perform it expressly. These policies are used to select the less important objects and therefore can be evicted. A Cache eviction policy example is the LRU (Least recently used) that evicts the least used objects.

The following subsections provide a brief description of caching technologies that can be used to fulfill the needs of the distributed system proposed on this thesis.

### 2.4.1.Java Caching System

JCS (Java Caching System [21]) is a caching system written in Java that provides the deployment of multi-level hierarchical caches in Java applications. As stated in [21] "It is intended to speed up applications by providing a means to manage cached data of various dynamic natures". JCS is highly configurable and therefore can be deployed with a wide variety of caches configurations. It supports the most common caches architectures for local and distributed caches.

JCS uses a concept of regions that can be seen as cache instances. Each region can be deployed with different combinations of plugins adding different levels with specific behaviors to the instance. The plugins are divided in four major groups Memory, Disk, Lateral and Remote. Memory and Disk plugins provide multi-level local cache support with common eviction techniques such as Least-Recently-Used (LRU) or First In First Out (FIFO). There is a JDBC [22] plugin in the disk group that provides persistent caching resorting to an ODBC [23] database, along with a specific plugin for Oracle's Berkeley DB (a key-value database for java).

The Lateral and Remote plugins provide connectivity across multiple cache instances (both local and remote) with distributed caching capabilities. There are multiple plugins which makes

the system highly tunable. Remote and Lateral plugins are often implemented on top of the TCP/IP transport protocol.

This framework is possibly the most complete caching solution for Java. There are other similar solutions for cache development, such as, Infinispan and Ehcache [24, 25]. The provided functionalities are more or less similar to JCS. However [21] claim JCS to be faster than Ehcache so that's the reason Ehcache description has neglected in this document. As for Infinispan, through the analysis of [24] it is driven to distributed caches than JCS as it makes almost no reference to local cache deployment.

### 2.4.2. MapDB

MapDB is an open source project with a few years, at the current date. It is an embedded key-value database for Java. Although it has some very particular features that make MapDB a unique database engine, suitable for storing data blobs persistently.

As [26] MapDB offers a set of concurrent associative memories (Maps and TreeMaps) to store serialized objects in both disk and memory devices. By using a key-value approach, query/retrieval of objects can be executed more efficiently than in other database engines (SQL or Document based).

Moreover, MapDB database engines offer encryption, transactions, caching and custom object serialization that may be used optionally to enhance the engine performance. MapDB is a memory cache designed for temporary storage of objects. Operations such as, read and write may not be performed directly to the disk. It can be configured with the Least-Recently-Used (LRU) eviction strategy, as well as with the desired in memory capacity. Management of cached objects is completely delegated to the database engine.

Compared to Java Caching System, MapDB is not a specialized cache provider. Nevertheless it offers a level of persistent disk storage and a level of memory cache that fits perfectly for our needs. As storage of binary data in common ODBC databases have poor performance, MapDB presents itself as better alternative than JCS as it is less complex, portable and faster for this kind of data.

## 2.5. Communication protocols

When aiming to achieve the best performance possible, sticking with the same transport protocol might not always be the best idea. Especially, if on top of the transference protocol we use a totally non-related application protocol to achieve a specific goal, such as, in our case to achieve connectivity across multiple private networks.

This section provides a brief description of techniques used to provide connectivity across multiple private network environments as well as a very efficient transport protocol that may be used to optimize communications in those cases.

### 2.5.1. UDT: UDT Based Data Transference.

UDT is a transport protocol aiming to mitigate the limitations of TCP algorithms on high-speed networks. In [27] the authors claimed that TCP congestion control, flow control and window control algorithms do not work well for high-speed networks with high latency. Actually in [27] it is given an example for an high-speed link of 10Gbps where the TCP could only achieve around 100Mbps effective bandwidth usage. It is certain that most medical institution do not

have such high-speed Internet connections. Nonetheless, UDT approaches data transference uses some good assumptions.

Firstly, congestion control, flow control and window control algorithms are tunable by the user application using a well-defined interface. Secondly, the protocol itself has enhanced bandwidth detection and uses a much optimized approach to acknowledgement of messages as described in [27]. Summing up, they might be configurable to our environment constraints.

Lastly, besides the client-server connection establishment, the UDT has a mode called Rendezvous Connection Setup. It takes advantage of the UDP protocol to setup the connection either by the server or the client. In the rendezvous setup mode, both hosts attempted to connect and listen for connections at the same time, making the connection possible even if one of the hosts is behind a firewall protected environment. Moreover this rendezvous mode promotes firewall hole punching. Firewall hole punching technique is used by some well-known peer-to-peer applications (such as Skype) to provide connectivity between works behind firewalls and NAT environments. It consists in connecting to an accessible host in order to open "breaches" in the firewall or NAT. As the UDP protocol supports various data streams per port these breaches can then be used to communicate with the desired host.

### 2.5.2. ICE: Interactive Connectivity Establishment

ICE (Internet Connectivity Establishment) is the latest effort of IETF (Internet Engineering Task Force) in NAT transversal [28]. NAT transversal is the concept of creating a communication channel between two hosts even if they stand behind a NAT system. It is a hot topic for IETF because of its interest in multimedia communications.

Multimedia communications in Internet often use out of band control protocols with rendezvous capabilities such as SIP (Session Initiation Protocol) [29]. ICE is very well enclosed in the SIP/VoIP environment and in multimedia communications in general. Its main goal is to find a direct communication path between two remote hosts using existent techniques, such as, TURN and STUN. STUN and TURN are two techniques previously used to transverse NAT. Their effectiveness varies according to the type of NAT implemented in both networks. Moreover, ICE evaluates all the found paths and orders them in terms of efficiency.

There are multiple libraries supporting ICE Protocol including one for Java, although it is not completely implemented and lacks of reviews. There are multiple stable releases of ICE libraries written in C language. There are no records of attempts made in order to use the ICE protocol in data transference situations but in the multimedia environment (VOIP and other SIP based Communications) it is frequently used.

## 2.6. Related Work

The typical implementations of DICOM Storage and Retrieve processes are not very efficient. The main propose of a DICOM network is to support exchange of medical image data between different nodes of the network. File download and upload are therefore two major processes. Nonetheless, creating strategies to improve this process is challenging due to few conceptual problems associated. Firstly there is no way of knowing how many images, nor its size, does a study have prior to its retrieval. The downloader will not be able to estimate the study download length, neither in terms of time nor in number of missing images. Secondly, there is no control in the transference procedure, in the downloader point of view, that allows it

to pause study retrievals. Lastly, storage requests are issued in sequence without any parallelism, leading to poor performance in high latency networks. These issues are well represented in [2].

In [30] a study was conducted with a miscellaneous of medical imaging studies. It revealed that standard DICOM transference syntax (without any kind of compression) only used around 75% of the available bandwidth in LAN environments. It was also identified a problem with the agreement of transference syntax in DICOM UL associations. Citing [30] although DICOM supports various transference syntaxes with different levels of compression (both lossy and lossless) applications very often do not support the same syntaxes. Consequently leading to massive usage of the default transference syntax.

In order to mitigate this problem, the usage of interfaces between the DICOM applications to mediate the transference was proposed in [30, 31]. The interfaces would act as proxies between the two applications supporting the best transference syntax possible. Moreover, they provide parallel transference of images.

Conducted trials revealed the optimum number of parallel transferences as well as the most efficient compression codec for the transference syntax. With the proposed method Maani et. al. claim to have achieved 90% of network usage in LAN communications. The trials consisted in the transference of multiple studies at the same time with a significant data volume of 157MB. The proposed method managed to achieve considerable improvements both in LAN and WAN scenarios. The performance improvement in a WAN scenario was even better, as parallel transferences proved to be more effective.

Although this case study scenario is only focused in point-to-point connections the analysis of DICOM Protocol weaknesses proved to be very similar to the problem stated in this thesis. Compared to our proposed architecture, this technique cannot guarantee, “anytime anywhere access” as communications between interfaces use the normal DICOM protocol. Moreover they do not provide any security extensions to the standard protocol.

In [32] Bastião et al, proposes a PACS architecture that takes advantage of the cloud computing capabilities. Cloud providers offer huge amounts of storage space as well as optimal availability of data which makes them a very attractive platform for storing large volumes of data, key features of any PACS Archive. Costa et al [6, 33] claim that costs of storing large volumes of data on cloud providers tend to be smaller than regular in-house storage as it does not require any upgrades over time to keep-up with the amount of data. Furthermore cloud based PACS Archives can be effectively shared among different institutions and thus, effectively promoting study exchange and cost reductions [34].

Bastião et al, states that a PACS Archive might be divided into two different components, DICOM Object repository (storage) intended to store the actual objects and a indexer database for meta-data. Indexed meta-data enable faster responses to find requests. The proposed architecture intends to move both these components to cloud providers while keeping a secured master index inside the medical institution with the confidential patient meta-data. A gateway was also proposed. It works as a proxy for DICOM-Commands, translating them to web services requests. This approach provides compatibility with DICOM equipment way for both the master index and the cloud providers.

Performance tests were conducted on two different cloud providers Amazon S3 and Google Storage [35, 36]. They showed that this cloud-based PACS Architecture is significantly slower than regular PACS approaches (with archives inside the institution). However, even with poor performance, the proposed PACS-Cloud architecture opened the doors for the distributed DICOM network that will be mentioned further in this document. Its main ideas of sharing studies

among institutions and studies distribution based on web technologies were very helpful in following studies.

As expressed, cache systems are a hot topic in distributed systems. As such, previous works related to medical imaging and distributed PACS architectures have somehow incorporated cache components in order to improve their workflow. Nevertheless, there seems to be no proposal for generic software cache architecture for medical imaging repositories. Related works seems to focus either on pre-fetching techniques or very specific cache specifications.

In [37] was proposed a pre-fetching technique for medical imaging information systems (PACS, RIS, HIS). Its intention is to minimize the delay in operations associated with the retrieval of data from those systems by retrieving those objects prior to its request. This technique is focused on an institutional level rather than a distributed environment.

In [38] was presented a caching technique for a web-based PACS. The cache system was associated with a pre-fetching technique based on windows of interest. The method involved the calculus of a window of interest for each user. Cache population and eviction would then reflect the changes to the user window of interest. However great results were achieved, direct implementation in our architecture is infeasible as it is very specific for this use case scenario. Although we acknowledge that splitting images into smaller portions may help to improve the PACS users experience.

Another possible solution is the deployment of hardware caches. Essentially they are replicas of the actual image repository. Integration of these replicas within the PACS environment is often achieved by specific hardware or software. For example, in [39] Gutiérrez-Martínez, J. at al describes a cache architecture for a single hospital PACS. The proposed solution required specific hardware and infrastructure to be deployed. As the result system is impossible to be embeddable in a single software solution.

Steve Langer [40] presents a clever approach for improving the QoS of a PACS with an outsourced PACS Archive. His approach involved the deployment of a small intermediate cache archive, as such, the performance constraints introduced by the outsourced archive were reduced. In this thesis, we keep in mind this approach in order to propose our own cache strategy for reducing the footprint of foreign PACS Archives.

# 3. Case Study Assumptions

This chapter provides the assumptions and specifications about the current state of tow background projects for this thesis, namely the DICOM Router [9]. Firstly, it will be presented a general description about both projects followed by a complete description of current architecture.

## 3.1. DICOM Cloud Router

DICOM Cloud Relay is a relay service over the cloud that was proposed by Bastião et al in [9]. It was intended to provide DICOM Query/Retrieve and Storage services between remote locations over the Internet. The suggested approach enhances study exchange among institutions and tele-radiology by promoting seamless communication between DICOM compliant applications, both inside and outside institutions boundaries.

The proposed solution involved the deployment of two main components. The DICOM Routers, which are in charge of relaying communications between applications in different networks. Therefore, they are intended to be deployed in every DICOM island[3]. They are essentially DICOM nodes, providing DICOM Services to other applications, working as server unities and also as consumers. Provided services are essentially virtual because they are mediators to real services provided by applications over the distributed environment. They also have the responsibility of advertising the distributed environment about the services provided by appended applications. DICOM Routers act as proxies for these services enabling DICOM-compliant applications to access to remote applications (via router) and thus inter-connecting distributed environments.

The second component is DICOM Bridge Router. It works as a network coordinator storing centralized information about DICOM Router, services provided in the network and their location, also called routing tables in the paper. Moreover, DICOM Bridge Router provides also some security features, such as, user authentication.

The proposed solution also involved cloud providers in data transference processes. Cloud provider was used as middleware between routers. The studies are uploaded first to the Cloud providers and then downloaded by the receiving router. This was done in order to free the central DICOM Bridge Router of data flows keeping bandwidth needs and associated costs as reduced as possible.

## 3.2. Current scenario

The current scenario of DICOM communications environment in our research lab is derived from the [9] approach. However, some modifications have been made in order to make this distributed PACS more compliant with real-world deployments.

[3] DICOM island means a DICOM network without connectivity to other DICOM networks

The deployment scenario also involves the creation of a distributed PACS system supported over the Internet with DICOM-Ready communications. DICOM Communications (Services) are provided by a similar approach to [9] using DICOM Routers to inter-connect every DICOM island. DICOM islands are denoted to be networks were full DICOM Connectivity can be achieved, typically inside institutions or other private LAN environments. As DICOM routers are typically inside firewall protected networks, direct communication between routers may not always be achievable.

Apart from previous works (namely [9, 32]) the actual instantiations of the Cloud Router architecture does not rely completely on public cloud Services, neither for communications as [9] nor for storage (Image Archives). Rather, the platform relies on private repositories and the common Internet infrastructure for inter-institutional communication. More accurately it uses the well-known HTTP protocol. As the result of keeping the system's communication protocol fully web 2.0 compliant, DICOM Routers can effetely achieve connectivity in most private LAN environments.

There are two main reasons for PACS Archives not to be completely resident on cloud providers. The first reason, is that despite several attempts to secure the PACS Archive located in the cloud, legal constraints, namely in Portugal, France or Italy, tends not to make this migration pacific. Moreover, medical personnel have shown some resistance in relocating the whole Image Repository to a private contractor, again due to privacy and security considerations. The scenario where a PACS Archive would still reside in the medical organization boundaries has generally a better acceptance among the medical staff.

Taking into account both the legal aspects and the medical personnel valuable opinion, the current state of the art architecture preserves PACS Archives inside the institutions boundaries as in general PACS.

### 3.2.1. Main Architecture

The main architecture reflects the medical institutions needs to access medical image repositories from outside their institution boundaries. By not only making use of tele-radiology features but also in sharing studies resources across affiliated institutions. Figure 3.1 shows both these aspects inter-connecting a private institution with a physician at home and an affiliated institution. Both these DICOM islands representations are metaphors for real world use case scenarios.

The physician at home is a use case where the DICOM application residing on its own DICOM Island needs to access the PACS Archive (inside the private institution boundaries) normally for revision of studies. Multiple instances of these use case are supported, supported in its turn by the accounting system in the DICOM Bridge Router, which encloses the access to the distributed system to authorized personnel.

The Affiliated institution with its own local PACS Archive is the metaphor for remote access to the distributed PACS by physical locations not directly connected to the main institution facilities. Thus this use case encloses many possible scenarios including sharing of resources across multiple institutions, aggregation of facilities from the same institution etc. Again the access to the distributed PACS is subjected to the authorization given to the DICOM Router by the DICOM Bridge Router when it logs-in in the system.

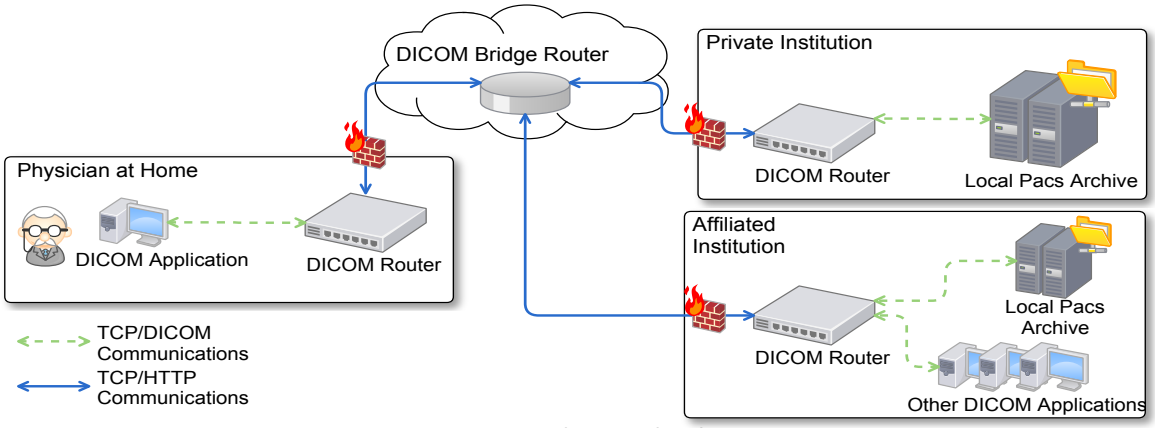



**Figure 3.1: Case Study general Architecture**

### 3.2.2.DICOM Bridge Router

DICOM Bridge Router serves a few noble purposes. One of its most important features is user authentication. As in any distributed system environment, user authentication prior to entrance is crucial to provide secure and armless services. User authentication is based on DICOM Routers credentials. Every router has to supply its credentials in order to access the system. These credentials are also used to identify the DICOM Router in the distributed system environment.

Although security is very important, it is not the main function of the DICOM Bridge Router. The main function of the DICOM Bridge Router is, as its name suggests, establishing a bridge between the DICOM Routers. It is used as a "man in the middle" for communication between pairs of DICOM Routers. For instance, router A wants to send some data (DICOM images) to router B. Minding the real world constraints imposed by firewalls and general NAT systems, router A may not have direct connectivity to router B. In order to overcome this inability the router A sends the messages to the DICOM Bridge Router, rather than directly to the router B. The DICOM Bridge Router acts as a relay to retransmit the messages to its proper destination.

The DICOM Bridge Router achieves total connectivity, as it is actually an HTTP Server listening for new HTTP Connections. Each established connection is used to the welfare of the distributed system. Every DICOM Router has at least one active connection that is established at the time of its registration in the Bridge. This everlasting connection can be seen as a service channel, in order to open new channels that may be requested. Note that this channel is very important because the DICOM Bridge Router cannot open new communications channels by itself, as it would eventually fall in the firewall constraints that were mentioned above. As the result the DICOM Bridge Router uses the service channel to send a command to the router asking the router to open a new HTTP Connection to the server.

### 3.2.3.DICOM Service Discovery and Registration

DICOM Communications are very similar with the SOA (Service Oriented Architecture) pattern where there are service providers which supply previously contracted services to clients. As stated previously in this document (section 2.3.4), DICOM compliant applications may provide certain predefined services to others, such as, storage or printing. In the DICOM standard, the service providers are called SCP (Service Class Provider). In our current architecture, DICOM Routers have the responsibility of making the virtual DICOM Services from appended DICOM Applications, i.e. applications registered in the router, available to the entire distribute PACS. This

is achieved advertising routes of those services, in the distributed environment, which are similar to network routers protocols.

In order, to become available, DICOM Services have to be previously registered in the DICOM Routers configurations, i.e. similar to a routing entry in network protocols. Routing entries are composed by the address of the DICOM application (*AETitle, IP Address, Port*) and the provided service class. Thus, DICOM Routers are capable of advertising their services to the distributed PACS.

Routes are advertised via DICOM Bridge Router to every DICOM Router in the system. Using this information, a routing table is constructed in each router. This enables the correct association between SCU requests and its SCP.

In addition to uploading its routing table to the DICOM Bridge Router, every DICOM Router also downloads the distributed routing table of the system. By doing so, DICOM Routers are capable of not only advertising provided services but also of effectively routing SCU (Service Class User) DIMSE commands to remote SCP. These techniques effectively allow remote services usage in the distributed system.

### 3.2.4.DICOM Routers

DICOM Routers communicate via the well-known HTTP/SSL Protocol. The choice of this protocol is easily explained by its broad usage that has conducted network security managers to allow this traffic to cross firewalls. Although HTTP protocol has the ability to pass through the majority of firewall configurations, HTTP protocol is inherently client server oriented which makes it difficult to achieve peer-to-peer communications. Moreover, the majority of institutions connected to Internet often use NAT (network address translation), disabling nodes (servers) inside the private networks to be reached by outside clients. As the result, achieving total connectivity between routers (in general) requires the usage of other connection techniques namely the use of a relay bridge.

As in [9], DICOM Routers have the purpose of representing all the DICOM applications inside of a DICOM island. In order to perform this feature, routers offer the same services as the DICOM applications supply. Currently, the DICOM Cloud Router only supports DICOM Query/Retrieve and Storage services. Figure 3.2 shows an example of the main functions of DICOM Router in offering Cloud Based DICOM Services (DICOM Retrieve Service) across different institutions. Although *router A* does not have any directly appended Query/Retrieve service, it acts like a proxy for the medical institution archive located in the other DICOM Island.

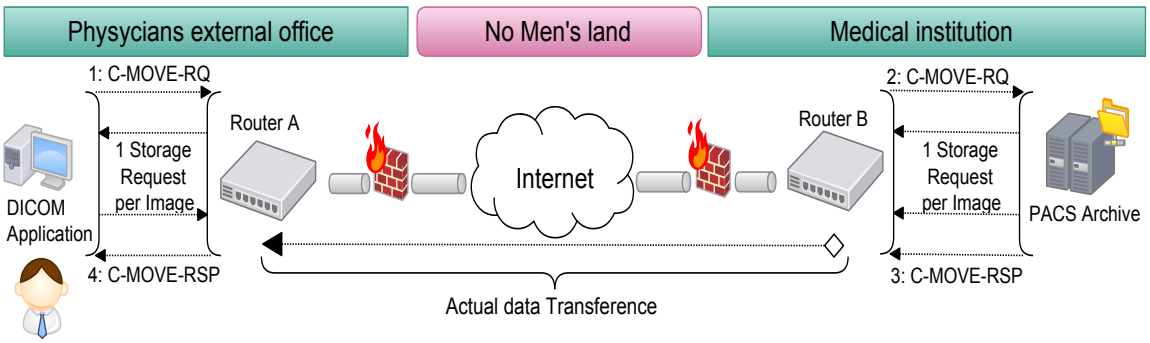



**Figure 3.2: Simplified Model of the Distributed PACS Environment.**

### *Data Transference: Uploading*

Data transference is a direct consequence of DICOM application calling a C-Store Command. In order to transfer DICOM Object over the cloud, firstly they need to be transferred from a repository application to the Router using the DICOM Retrieve service. Because this process occurs in a LAN environment, the transference is realized at a relatively fast pace.

Usually a medical image study has multiple images. DICOM Standard transfers them sequentially using multiple C-Store commands. Each C-Store Command, as it was previously described, has a C-Store-Request and a C-Store-Response which is sent for acknowledgement. No multiple C-Store commands are established in parallel and the images are transferred in sequential order with no overlapping.

Although in LAN environment this issue does not constitute a great problem for network usage. In WAN environment, bandwidth constraints and latency in communications makes the sequential transference of objects an inefficient approach. As in [30] the DICOM Routers make use of parallel transferences to boost the network usage and improves the transference performance in general. However, this process could still be to be improved, as it will be explored further in this thesis.

Figure 3.3 shows the workflow associated to a C-Store Command received by a DICOM Router. It is clear that upon receiving the file, the DICOM Router immediately compresses the file and opens a new connection channel to the destination router. The transmission starts immediately after the channel establishment. Since data transmission over LAN is much faster than in WAN (and specially with the overhead of establishing the connection) the rate of arrival of files is much higher than the rate of dispatched files which leads to a bottle neck effect on the Router, which in its turn leads to file gathering in the router. In order to keep the Router scalable the maximum number of opened connection channels was limited to 10. Channel limitations proved to be as dangerous as necessary, as it lead to some starvation problems, i.e. for two users using the service at the same time.

### *Data Transference: Downloading*

In order to accurately perform the DICOM Retrieve Service, it is mandatory that the DICOM Routers can exchange information correctly, namely DICOM Objects. As it is presented previously in this document, DICOM Standard uses a single major command to exchange DICOM Object between applications, the C-Storage command.

In DICOM environment, it is clear how transferences are performed. However, in an inter-institution level (communication between DICOM islands) DICOM must not be used for the sake

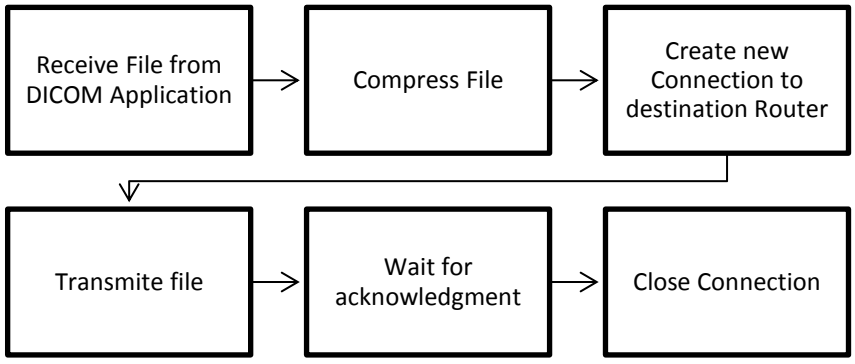



**Figure 3.3: DICOM Router incoming C-Store workflow**

of performance. The transference method in this architecture implied the creation of parallel communication channels between both DICOM Routers. The uploading Router is responsible to open these channels. An acknowledgement message needs to be sent by the downloading routers signaling the correct reception of each file. After the reception of this message, both routers may close the channel as it becomes futile.

Since data comes compressed from the up loader Router, decompression needs to be performed prior to sending the DICOM Object to the DICOM destination application. The DICOM Storage Service is used to send those objects. This workflow is shown in Figure 3.4.

DICOM Associations must be established between routers. They are created in the DICOM Standard whenever a DICOM Service/Commands need to be exchange between DICOM applications. In DICOM Router architecture, DICOM applications AETitles need to be unique not only in the DICOM island scope but also in the global distributed network. Moreover, associations are also identified by unique identifiers. These way DICOM Routers can match incoming DICOM Objects to its original DICOM Association and therefore its destination.

### 3.2.5. Final Considerations

The architecture described in this section is actually deployed, and capable of providing significant integration features as described above. Nonetheless a real word distributed PACS needs to provide not only features but also good quality of service.

As explained in the chapter 1, quality of service in a distributed PACS environment means: good performance and good reliability. The performance of the previous architecture was constrained by some architectural decisions namely the creation of a new channel for each image to be transferred. This adds a channel establishment delay for each image transference time, which scales up linearly with the number of images. As some study modalities may enclose hundreds of images [41] having an overhead in image transference is not an acceptable option.

Apart from those performance constraints, this architecture has two more problems. Firstly, the Bridge Cloud Router is a single point of failure. This means that, if the DICOM Bridge Router for some reason stops to operate, the whole system is compromised, as DICOM Routers cannot operate by themselves. Besides being a single point of failure, the DICOM Bridge Router is also a bottleneck as every communication channel between two DICOM Routers uses the DICOM Bridge Router. Its network bandwidth is shared by all the distributed system information workflows, both image data and control. This imposes a performance constraint and an infrastructural requirement, as the network connection of the Bridge Router needs to be able to support all the workflows.

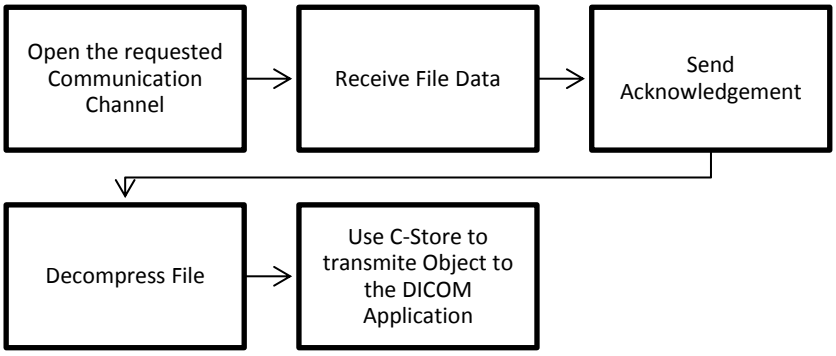



**Figure 3.4: DICOM Router incoming file workflow**

Lastly a major reliability issue occurs when the overall system fails to provide connectivity to its appended applications either if a DICOM Router or the DICOM Bridge Router fails. For instance, imagine that an application is trying to store a given image in an offshore archive and a failure of this type occurs. If there is no connectivity, the distributed PACS cannot provide the desired storage service nor a temporary storage service for the image. Thus if the application cannot provide this temporary storage herself this image would be lost.

In the next chapter it will be presented an extended architecture to mitigate these performance and reliability issues, as it is the main scope of this thesis.

# 4. Enhanced Performance and Reliability

This chapter provides an overview of this thesis proposed architecture. It describes the used techniques to achieve better performance and availability in the previously existing distributed PACS system (introduced in 3.2).

## 4.1. Overview

What does performance considerations mean in a distributed PACS? As it has been stated throughout this document, performance considerations mean reducing the footprint introduced by our system in the pre-existing medical workflow to the reasonable bare minimum. In this context, the term footprint refers to the modifications on previously existing workflow, imposed by an information system. For example, the raising of digital medical images and viewer applications changed the existing paradigm, i.e. where physicians analyzed image films, to one where physicians analyze images on computer screen.

In performance terms, the main footprint introduced by a distributed PACS architecture has its main component in the image retrieval. The retrieval of medical image studies using a PACS is usually more efficient compared to analogical competitors. Nevertheless, it does not mean that delays introduced to the medical practice are completely inexistent. Especially, in a distributed environment where communication delays are introduced not only by the potentially gigantic data flows (4.2.1), but also by the distributed communication infrastructure itself. Resuming, the performance footprint of a distributed PACS can be measured by the studies loading time in the different use case scenarios supported by the system. This automatically gives extreme importance to data transference and to architectural options that could improve data exchange processes.

Availability in a distributed environment is the capacity of the system to make services accessible. In a distributed PACS, such capacity is related not only to the system's performance footprint but also to the provided quality of service (QoS). Also, it reflects the system reliability, which can be looked as the ability of the system to make regular medical processes keep functioning, disregarding external conditions, especially IT related issues. In this proposed architecture, it is mainly offered remote DICOM Services (Query/Retrieve, Storage) which means that the distributed PACS environment should be always accessible, otherwise the medical practice may be severely compromised. The availability of imaging studies is a working subject of fault tolerance and replication areas of the information technology. In this chapter, it will be proposed a method for PACS Archives replication, as a similar approach was taken in other non-related distributed systems.

The proposed architecture derives from the previous distributed PACS architecture introduced in section 3.2. In order to explain our proposal, Figure 4.1 will be used as an illustrative example. The proposed architecture places the DICOM Bridge Router in the same network as the PACS archive, i.e. inside the private institution network boundaries, in order to improve the overall system performance. Thus, the communications between the Bridge and the DICOM Router associated to the PACS Archive are performed in a LAN environment, having typically a better performance, when compared to the deployment of the bridge in an external

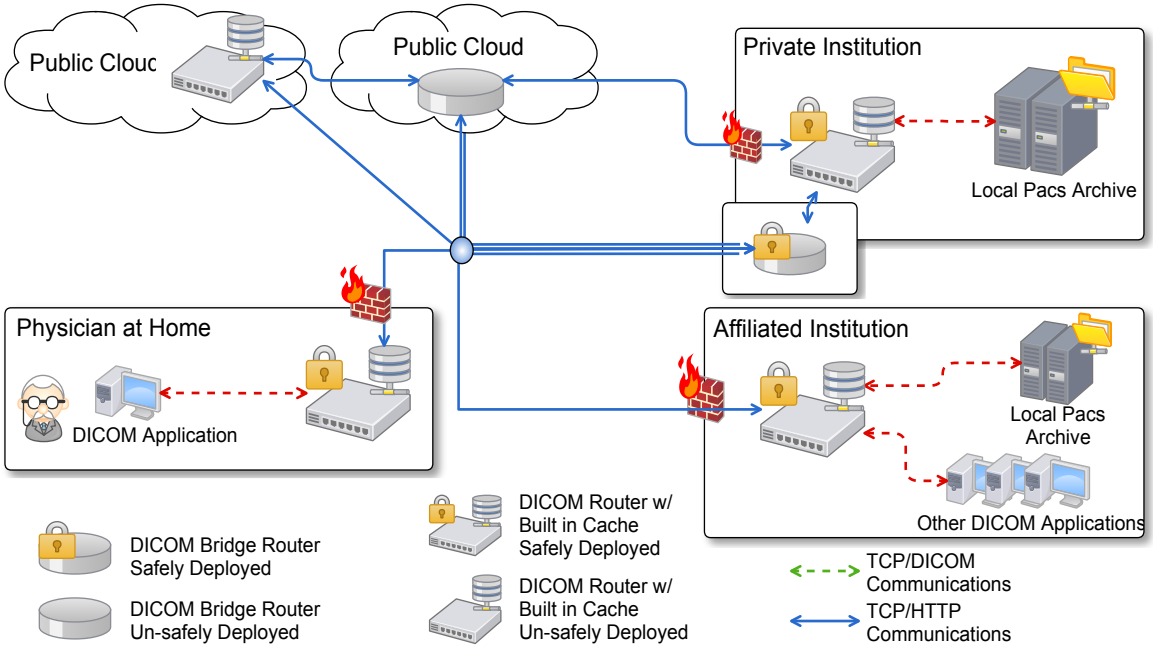



**Figure 4.1: Purposed architecture**

environment. In a LAN environment, it will be possible to benefit from minor round trip times (RTT) as internet connections are associated with higher latency.

In order to improve the overall study retrieval performance, the Controlled Channels method is proposed in ***section 4.2***. The Controlled Channels are high level control mechanism for router-to-router transferences. This control mechanism reflects many novel performance considerations about the previous implementation which make the enhancement possible. This thesis also proposes to include a Cache Module to the DICOM Router ***(section 4.3)***. This module provides caching capabilities for DICOM images, either complete or split in chunks. It also provides direct support for image retrieval as other workflows may use its synchronization capabilities.

The cache module will provide local and distributed persistent caching. Local caching will be implemented on regular routers, in order to avoid repeated transferences of images. Whenever a DICOM Router is presented with the need to download (DICOM C-Move) an image, the local cache module will be asked if the image is available. If data is in the local cache, its transference will be avoided and the DICOM Router automatically responds to the application, achieving a very high transfer rate gain. The distributed caching mechanism is our architectural solution to provide both higher availability and better transference performance. They should hold a (statistically) representative set of images from a PACS Archive, creating multiple locales where studies would reside. Multiple data sources will ease data flow overhead on both DICOM Routers and DICOM Bridge Routers as data retrieval operations may be balanced between multiple components ***(section 4.3.3)***.

In previous architecture, DICOM Bridge Router was identified as a single point of failure that might compromise the whole system. In order to eliminate this effect, the proposed architecture opens the possibility to instantiate multiple bridges in the distributed environment. DICOM Routers may connect to as many bridges as desired, favoring also load balancing of the distributed system. Several changes have been made to the DICOM Bridge Router communications layer to incorporate the multiple bridge paradigms, described further in ***section 4.4***.

Finally in ***section 4.5***, we propose that some DICOM Bridge Routers may act as Security Managers protecting sensitive data in public locales. Moreover, distributed cache routers instances are intended to be deployed inside private institutions boundaries but also on public cloud infrastructures. So, it is crucial to protect sensitive data, such as, patient identification. Therefore the concept of safely deployed components was introduced. The unsafely deployed components mark the instances that should protect these sensitive data. Both DICOM Routers and DICOM Bridge Routers may be deployed unsafely on public providers.

## 4.2. Improvements in image transference procedures

Chapter 3.2.4 provided a small introduction on how image transference was handled by our proposed distributed system, more specifically on how images are transferred from a DICOM Router to another.

A few problems were identified regarding not only the performance of the communication channels used in image transference, but also with their management. The creation of a new communication channel for each image proved to be an inefficient approach. Each channel creation has a fixed establishment delay associated. As the result, the transference's overall channel establishment delay scaled linearly with the number of images to be transferred. Moreover, the synchronization of multiple active channels, in both downloader and uploader routers, required a bothersome algorithm that imposed performance constraints on its own, as described in 3.2.4.

In order to solve previous issues, we propose a technique that re-uses communication channels. Moreover, the actual performance of the image transference channels also needed to be improved. Taking into consideration the distinct data profiles of medical imaging modalities transferred (4.2.1), we present a proposal to boost the overall performance of study retrieval. Both these efforts culminate in the proposal and implementation of the Controlled Channels method, aiming to decrease the study retrieval time by combining channel re-usage with a file splitting technique, described in sequel.

### 4.2.1. Data profile of medical imaging studies

In order to improve the performance of any data transference, it is essential to understand beforehand which are the profiles associated with the data to be transferred. For instance, the average volume of data, whether or not it is divided in smaller pieces (files or chunks), the average size of each piece, among others. Different data profiles may benefit differently from different flow control algorithms. The performance requirements for each profile may also vary. For example, for small files, latency in communications is more important than raw transference throughput.

Distinct medical imaging modalities produce studies with different data profiles [40, 41]. They vary from small studies, with one or two images, to studies that enclosed several hundreds. Moreover, images itself vary from a few hundred KBs (Kilobyte) to a GB (Gigabyte) per object file. This diversity is well represented in our study dataset (see Table 5.1 on chapter 5). In our use case scenario, different data profiles are even more relevant than in usual point-to-point transferences. Due to the fact that transferring images from router-to-router often requires its previous transference from the PACS Archive to its appended router. This transference is completely serialized (non-parallel), as a result, images will arrive to the router at different rates, therefore severely shaping the posterior router-to-router transference.

Achieving the best performance in every circumstance is tricky. As such, our flow control algorithm needed to be adaptable. The next sections will describe our proposal to reduce the effect of such a diversity of data profiles, by using an adaptable flow control mechanism along with a data normalization technique.

### 4.2.2. Controlled Channels

Controlled Channels method is the primary contribution of this thesis for enhancing medical imaging transference performance in distributed PACS using the DICOM Router platform. The main concern in Controlled Channels is to provide a very high-level communication channel for router-to-router image transference. There are a few specifications that make Controlled Channels a unique proposal.

Firstly, they implement an application layer flow-control algorithm that is intended to accelerate performance regardless of the transport protocol beneath it. Some transport protocols, such as, TCP/IP, may also have flow-control algorithms incorporated. However, in our case, router-to-router communications cannot rely on them, as they are connection oriented and router-to-router communications are relayed by the DICOM Bridge Router (i.e. they are not direct connections). For example, TCP/IP's flow-control algorithm can only guarantee the delivery of messages from DICOM Routers to the DICOM Bridge Router and not from router to router. This was also performed by previously implemented channels through the use of acknowledge messages. However, its impact on the channels performance was not taken into account.

Secondly, Controlled Channels are not bound exclusively to DICOM data. In fact, they are completely generic regarding the type of transferred data and its usage. This makes them actually usable in other systems. In our particular use case, it allows the re-usage of a channel for multiple images transfer. Moreover, the use of the Controlled Channels method allows the data transference channels to be oriented to its origin-destination pair of routers, as opposed of being oriented to the image as the previously approach.

Lastly, the Controlled Channels are adaptable and flexible. They are bound to three transference parameters making possible to tune the channels behavior according to the data profiles of studies being transferred and the channel congestion rate. This enables improvements of performance regardless of these factors.

In order to develop this method, we carefully analyzed which metrics associated with the data transference channels would have a higher impact on the overall study retrieval performance. We identified the transference delay, the communication channel throughput and the channel establishment delay (previously referred in this document). The transference delay is the total time needed to download a study from one router to another. It very well reflects how long it takes to issue all the storage requests from a C-Move call, in our distributed environment. The communication channel throughput is the relevant data transference rate of the communication channel, discarding the channel's overhead, such as, acknowledges and other notification messages. This reflects how much of the channel is wasted with non-relevant data.

As previously referred, transference of medical images has to deal with huge amounts of data but also with different data profiles. This produces different combinations of study size, number of images and average image size, which severely shape the conditions imposed by the previously described metrics. For instance, studies with many small images will generate several control traffic and benefit of a small transference delay. On the other hand, in big files the connection establishment delays is meaningless and thus maximum throughput is desirable even if it means higher establishment delays.

As a result, Controlled Channels implements a data normalization procedure. It is intended to provide an abstraction to the studies data profile. Thus, the communication channel can optimize the transference performance based only on its own parameters (described next).

### *Study data profile normalization*

In order to normalize the data profiles, two approaches were taken. The first approach is to split image data into chunks. The process of splitting is easy both conceptually and computationally. It consists of dividing the binary data of file into portions called chunks. In order to ease the chunk handling with a minimum data overhead, along with each chunk, it is also produced a chunk descriptor containing information to identify the chunk unequivocally. It includes the chunk size, chunk number (indication the position of the chunk in the original file) and the total number of chunks produced from the file. Holding both the chunk data and the chunk descriptor, any application (local or remote) can recombine the data and reproduce the original file. Figure 4.2 shows an example of this technique.

Chunk splitting is implemented in the DICOM Router prior to data transference. In addition to chunk splitting, the Controlled Channels implement the concept of Bulk transference where a predefined number of files (bulk size) are transferred sequentially, without acknowledging each file, but rather the whole group. In order to accomplish this, Controlled Channels have an inner queue where files wait for its turn to be transferred. When the inner queue reaches the required number of files, a new bulk containing those files is dispatched. However, there is a maximum waiting time in which an incomplete bulk will be sent in order to prevent long waiting times to files that are in the queue of an available channel.

In Figure 4.3, it is represented a Controlled Channel associated with the length of time needed to establish the communication channel, i.e. the channel establishment delay. It is also represented two bulks, one with three chunks (representing a full bulk) and an incomplete bulk with just one chunk. Along with the Controlled Channels representation, there is also a representation of the previous transference paradigm in order to give contrast to differences between both.

Using Controlled Channel with File Splitting technique, it is possible to normalize the data

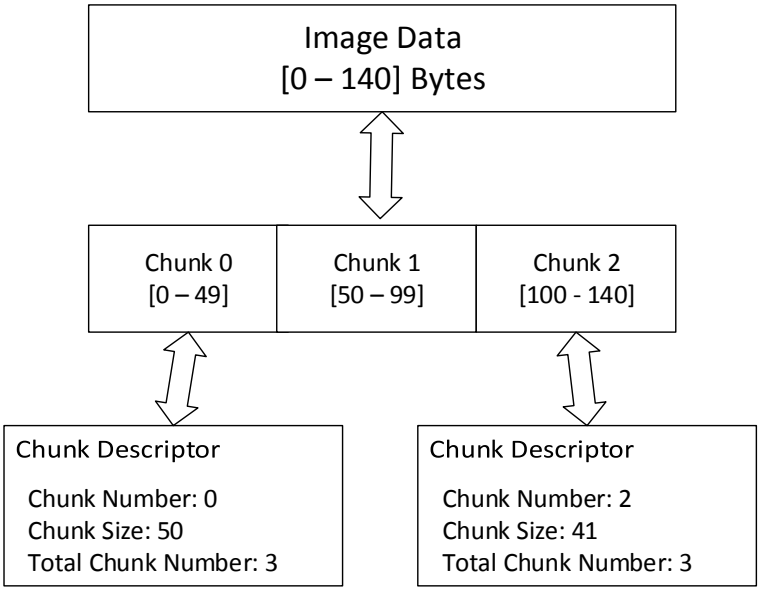



**Figure 4.2: Image spliting example**

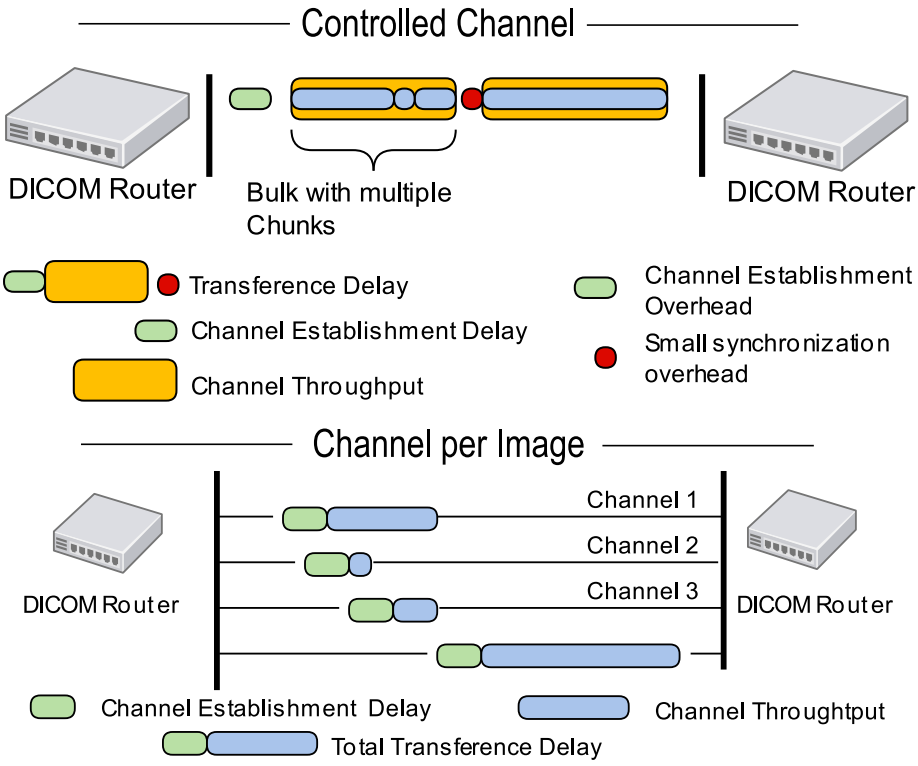



**Figure 4.3: Controlled Channels Representation**

profile, as files are primarily split with a maximum size, i.e. the chunk size. Then, from study to study, the average chunk size would not differ, only the number of chunks is different. This will make Controlled Channel queues to fill at different rates. On one hand, studies with large average file size will produce large number of chunks at slow rates (as images take more time to sequentially arrive from the DICOM application). On the other hand, studies with low average image size and high file count will produce a constant stream of few chunks.

The solution, to cope effectively with both scenarios, resides on the configuration of the Controlled Channels parameters, i.e. the Bulk Size and Maximum Pool Time. Increasing Bulk Size will reduce acknowledge overhead and, theoretically, increases the throughput of the channel. This should be performed when the channel has a high chunk count in the queue. However, there is a cutoff point where the throughput of the channel cannot be increased further. Then, the solution is to open another parallel channel. Decreasing Maximum Pool Time and Bulk Size will reduce latency between bulk transferences, and thus it will provide more response to small studies and images (see section 8.1).

Controlled Channels provide two interfaces which main goal is to audit channel performance and reconfigure its parameters on the fly. Performance Probes takes into account major factors of the channel operation, such as, the amount of data transferred by the channel since its creation, a time stamp denoting how long the channel have been active and, lastly, the channel throughput. The configuration of the Controlled Channels can be performed via a well-defined interface aiming channel reconfiguration over time. Ideally, each channel would be automatically configured taking into account study modalities characteristics and the channel probe information, such as, the queue length or average waiting time.

### *Implementation*

The Controlled Channels were developed along with the DICOM Router Project. It used a previously developed component denominated as *Communication Manager* which core functionality is to manage of the establishment of communication channels (router-to-router). As the result, communications are still performed using the same set of messages and channel creation and deletion is accomplished through the same procedures.

Apart from the *Communication Manager*, the Controlled Channels have essentially other two key components, as illustrated in Figure 4.4. The Stream Controller is used to provide a flow-control algorithm on top of the transport layer in order to support reliable connections between two points that are connected through a Bridge. There are two kinds of Stream Controllers, an *Input Stream Controller* used in outgoing channels (upload) and an *Output Stream Controller* used in incoming channels (download). These controllers are responsible by the enhancement of transference performance, namely the implementation of the bulk transference capabilities described above.

The flow-control algorithm makes use of three distinct messages. The header message is used to transmit all the bulk chunk descriptors in a single message. Its correct reception is signaled by an acknowledge message. It does not carry with any information at all, although it is crucial for the maintenance of channel. The absence of these messages will cause the channel to fault and become useless to the system. The chunk data is handled by chunk messages that are transferred in sequence (similar to its descriptors). The bulk transference is finished with an acknowledge message that makes the channels available to issue another bulk. This procedure is described in Figure 4.5.

Stream Controllers provide blocking read/write operations. Thus, each stream needs to be associated with a driver to be used in a concurrent environment. Stream drivers are essentially threads that manage the queue of the channel and perform account operations. They also provide an abstraction from the blocking nature of the stream to a non-blocking approach more suited for concurrent environments. There are two kinds of drivers for incoming and outgoing channels. BlockingBuffer is a very important component inside upload drivers. It manages the minimum waiting time of the channel by scheduling a maximum time for the channel queue to be flushed upon the entrance of a new chunk in an empty queue. This scheduled operation may be canceled if the queue reaches the bulk size in the meantime.

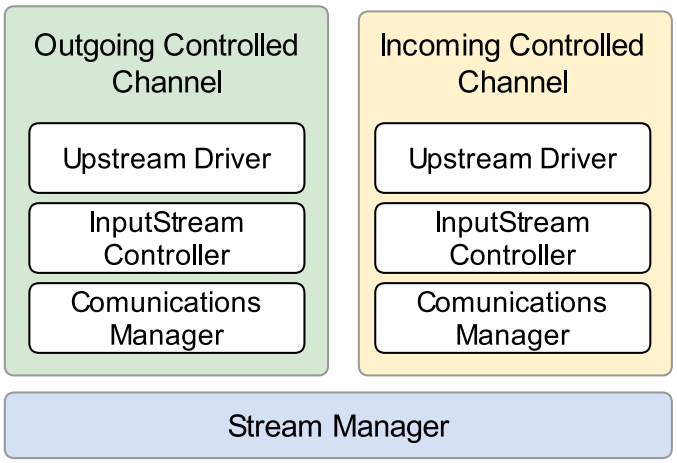



**Figure 4.4: Controlled Channels Architecture**

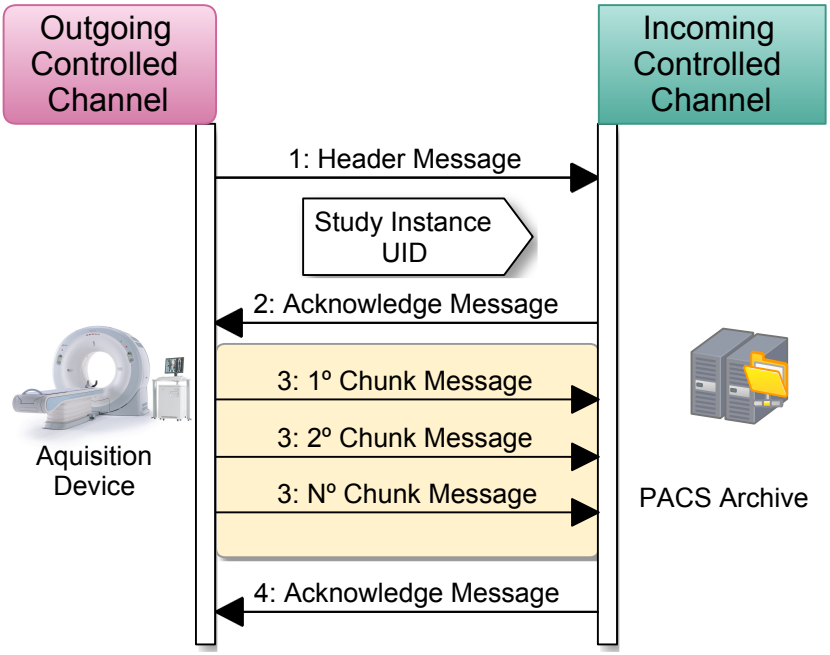



**Figure 4.5: Controlled Channels flow control signalling**

### *Integration in the DICOM Router*

The integration of Controlled Channels in DICOM Router involved certain adjustments, mainly in communication control procedures. Firstly, a manager module had to be implemented, in order to manage the creation, deletion and performance of the channels. With Controlled Channels, before the transfer of each file, a channel needs to be retrieved from this module. The possible creation of new channels will be performed if needed, i.e. only if the channels are too busy or there is no channel established to the destination router. Comparing with the previous approach, this is somehow analogous, because a new channel is always created to transfer an image.

In order to balance the load of each channel, a round robin algorithm was implemented. It works by changing the selected channel whenever it has its inner queue full (has more chunks than its Bulk Size). In case every previously established channel has its inner queue full, the algorithm tries to open another if the maximum number of active channels had not been previously met. This workflow is represented in Figure 4.6.

On the receiving router, the application has to wait for all the chunks of the same file to become available, otherwise the image file will not be valid to enter in the DICOM stream, i.e. DICOM streams can only send complete images. As chunks from the same file can arrive from multiple channels, it is crucial to have a centralized waiting module to all the channels. Further in this document, it is proposed a Cache System Architecture that, along with the faculties of caching DICOM Objects, provides the resources for the channel synchronization.

The complete integration of the Controlled Channels in the DICOM Routers workflow is represented in Figure 4.7. Along with the processes described above, it is important to refer that images are compressed before being split. This assures an optimal compression ratio. Of-course this requires images to be decompressed in the downloading router.

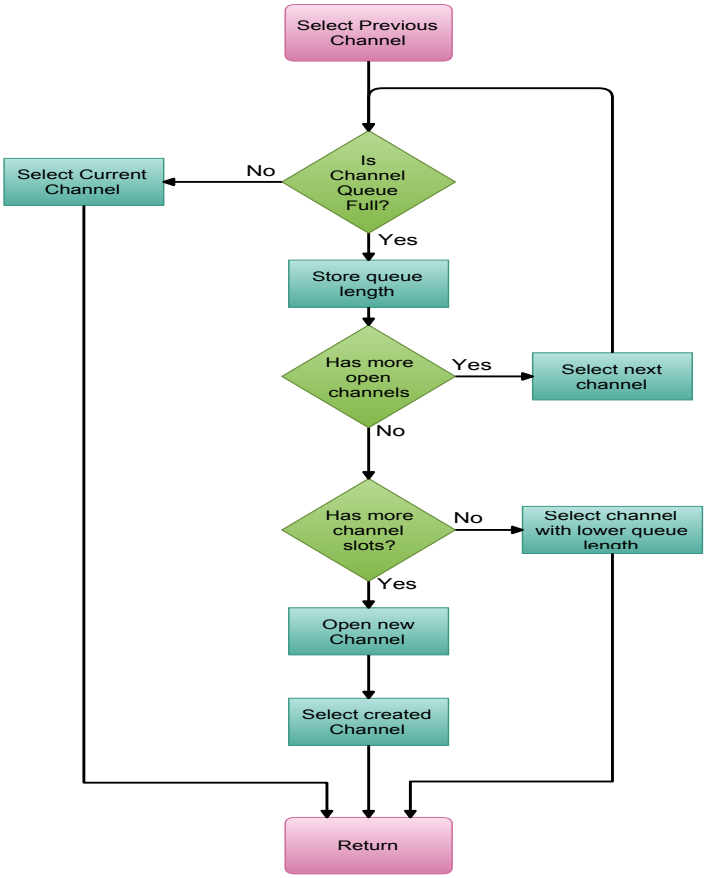



**Figure 4.6: Retrieving an available channel workflow**

## 4.3. Caching DICOM Objects

There are two main motivations for developing a cache system to support distributed workflows in medical imaging repositories. Firstly, the high availability of any PACS is extremely important in the medical workflow. Even small data access interruptions may severely affect the medical practice [42]. For this reason, in a distributed PACS environment high availability is crucial. Therefore the deployment of content across multiple nodes minimizes the effects of connectivity losses. For instance, if the medical institution loses the connection to a remote PACS repository, a local cache may contain the exams for the last months, which are often requested by the physicians. Secondly, there is the possibility of using data redundancy to increase the performance of the distributed PACS. Hence, load balancing techniques and multi-sourced services will be exploited in further sections.

This section proposes a novel Cache System Architecture for medical imaging repositories, purely software based. It aims to be a standalone module usable, embeddable in any java application, as a cache of any PACS archive. We will also explain how to take advantage of data redundancy provided by caches instances in our distributed PACS.

After analyzing the functional requirements of a Cache System Architecture for a PACS Archive, we noticed that two basic functionalities were required: the query and retrieve. As the names suggest, the query functionality allows external users to query for specific content that exists among the cached objects (studies, images, etc). In its turn, the retrieve functionality

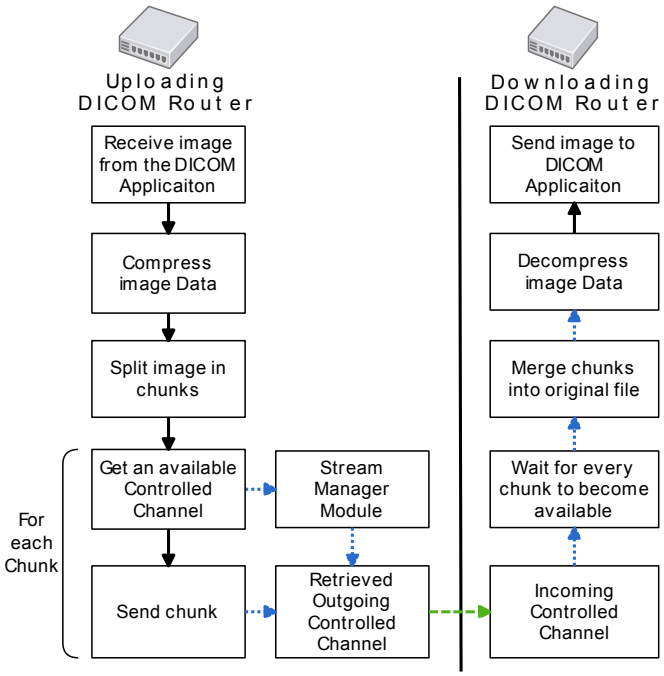



**Figure 4.7: Controlled Channels Workflow integration**

allows their retrieval from the cached objects and also from the overall system. These functionalities enable the cache system integration in an external workflow. Both interfaces names where carefully selected to reflect the well-known DICOM Query/Retrieve Service Class (described in chapter 2.3.6). In fact, a cache system has to provide a similar interface to the DICOM Query/Retrieve Service, commonly available in PACS archives. This means that a basic cache system may be viewed as a PACS Archive containing only a subset of studies and having a specific population strategy.

However, to support production requirements many other aspects must be worked. Cache Systems need to provide extended features to support intelligent cache population and eviction, as described in chapter 2.4. The idea of the proposed architecture is to provide a set of services that allows the creation of efficient modules to manage the cached data, improving the performance of exams retrieval while doing a smart management of cache storage volume.

Moreover, Cache population does not often involve storing whole objects but rather representative portions of them, for instance, store only one image per study. In fact, in our case study storing whole studies is not the smartest option. It has advantages related with availability of all study but it might need to store several hundreds of images (as described in section 4.2.1). Moreover, with some modalities, such as, CT producing 1GB image files storing whole images is not also a recommended option. Both of these approaches might quickly drain the cache capacity. The file splitting technique described above resolves this problem, although it creates the need to handle both images and image chunks in the system.

Cache eviction is also a major challenge for cache systems, as described in chapter 2.4. This functionality is not available in common PACS archives. Cache eviction policies often require meta-data to be collected during the cache lifetime, such as, hit-ratio, exams access statistics, and objects lifetime. This requirement varies with the implemented policy of distributed environment. We noticed that different clients would expect different behaviors from their cache system. Therefore, our architecture is intended to support multiple eviction strategies, including third-party.

### 4.3.1.Cache System Architecture

Taking into account the previously defined specifications, we propose an agile and modular architecture. It aims to provide an easy module replacement and supporting flexible policy implementations. Moreover, it is expected to be easily deployable in any PACS-DICOM environment. Our cache system architecture decouples the various aspects of a cache system into modules. It is divided in three layers of functional modules: storage management, meta-data management and the service layer modules, as illustrated in Figure 4.8. A description of each module will be provided in sequel.

#### *Storage management*

The storage management modules are responsible for the direct storage of objects in a persistent medium, such as, file-system or a database. Moreover, it provides low-level methods for querying about the existence of specific objects, images or image chunks, mainly to perform integrity checks. This layer encloses two modules, namely the Cache Persistence and the Big Memory Manager. The Cache Persistence module is the lowest level functional module in the system. It only provides simple (put, get, remove, contains) methods for objects in the persistence medium. Actually, in spite of handling DICOM Objects directly, it handles data following a Key-Value pattern, where keys production is delegated to another module and the value is binary data. The Key-Value pattern usage is very important in order to support seamless storage of images and image chunks.

In its turn, the Big Memory Manager module provides an abstraction for DICOM Objects (images and image chunks) on top of the simple Cache Persistence module. The Cache Key Translator module (Service Layer) is used to translate objects into keys prior to storage or retrieval. Events executed in this module may be recorded in order to collect important cache usage meta-data. This is accomplished by triggering events from the Cache Plugin Interface module (Service Layer).

#### *Meta-data management*

The meta-data management modules are responsible for coping with the meta-data of cached objects. There are two major tasks associated with this purpose. Firstly, the DICOM Objects meta-data has to be stored separately from the object. Indexing DICOM objects meta-data is increasingly common in nowadays archives, since it enables faster responses to queries and better information extraction capabilities [43]. Moreover, since we might not have whole

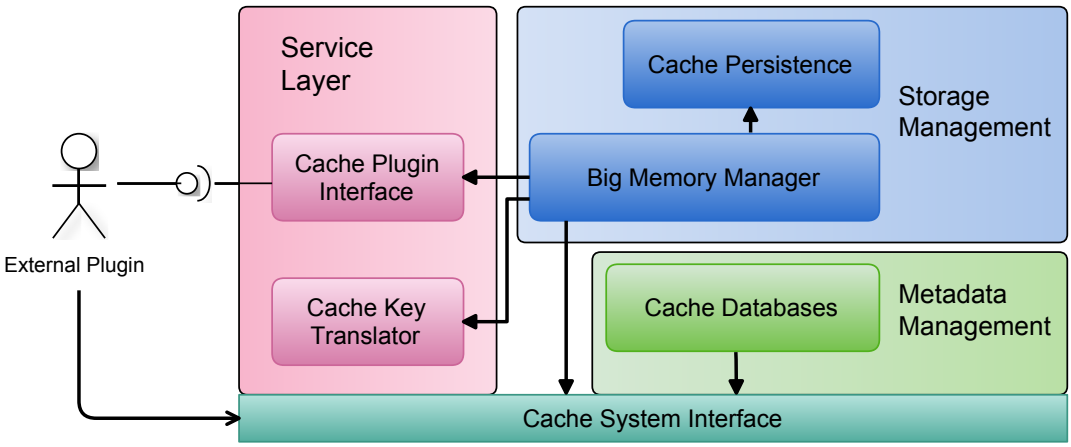



**Figure 4.8: Cache System Architecture**

objects cached we must manage this data independently.

Secondly, this module has to ensure a persistent mapping between images, image chunks and chunk descriptors (Figure 4.9). Image chunks have also to be linked to the DICOM Objects meta-data. Moreover, this information is crucial to know if the image is completely stored on cache or not.

### *Service Layer Modules*

The service layer modules provide extended functionalities to other modules, namely object-key translation and external plug-in support. The Cache Key Translator module provides object-Key translation. Translation works similarly to a hash function creating a link between a given object and its key. As described above, translation is used to keep the implementation of Storage Management modules as simple as possible by removing the need to work with both images and image chunks.

As described in chapter 4.3, the implementation of cache eviction policies is often associated with the maintenance of different meta-data. Cache eviction policies are supported via an external plugin system. Cache plugins are responsible for managing the meta-data associated with its policies. Furthermore, most policies also need meta-data associated with the cache usage, such as the cache hit-ratio or the least recently used object. Plugins may collect this data using the Cache Plugin Interface module through its event listener interface. Notifications are not only related to Storage and Retrieval operations but also with storage capacity limitations. Plugins are also encouraged to acquire meta-data from other external sources.

As a result of using an external plug-in system, our architecture is free of the burden of supporting meta-data management for multiple policies. Consequently more policies are supported and the system is more flexible. External plugins are encouraged to take actions upon the system via its public API, described below.

### *Cache System API*

We propose a modular architecture and so, a single API (Application Interface) is offered to its users. It acts as a wrapper bundling all the modules together. By using a single API the usage of our cache system is much more simplified. Moreover, it is easier for developers to implement and deploy their own version of the system as it is less likely to create inconsistencies between

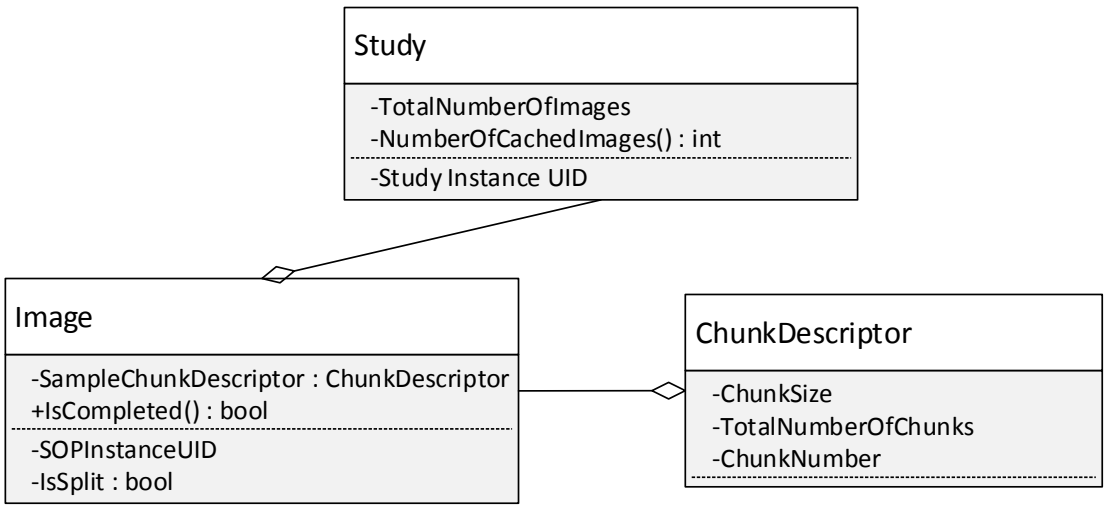



**Figure 4.9: Simplified meta-data management module class diagram**

the different modules. The API also provides an abstract implementation of the system including the correct linkage of all modules. As such, developers can concentrate on building their own modules without worry about their integration. Moreover, it encourages the usage of third-party modules. Section 8.3 provides the class diagram of our cache system architecture where it is clearly described the abstract modules that should be overridden.

The cache system API provides essentially two sets of functionality. Lower level functionalities more related to the Storage, Query and Retrieval of specific objects and higher level functionalities related to query of DICOM Objects. These functionalities are thoroughly described in section 8.2.

### 4.3.2. Technical implementation specifications

This section explains how the previously described modules were implemented. Moreover, it provides a brief description of the technologies used. Figure 4.10 shows the framework perspective of our cache implementation.

The storage management module is backed by a Key-Value embeddable database for Java. Typically, a Key-Value database provides quicker responses to queries compared with standard relational databases, because they do not have to perform extensive join operations [44]. Moreover, they are more suitable for our data profiles, i.e. binary blobs. Storing our data, i.e. image and image chunks, directly in the file-system was another option. In fact, it is the most common approach to store data blobs. We also tried this option and, inclusively, was implemented a solution. However, it presented two major flaws in our scenario. Firstly, storing each image chunk separately would lead to major overhead in file creation. The deployment of this approach proved to have intolerable performance constraints. Secondly, if the option is based on the storage of each chunk in its final position within a single file image, this involved the usage of random access read/write operations to maximize its performance. Although it avoided the overhead of file creation, it raised another issue with the storage capacity utilization. Since chunk eviction did not always led to actual release of storage space, due to file systems specifications. Taking these issues into account we have decided to use the MapDB framework as it offered embeddable Key-Value storage, an integrated cache system for stored objects and a custom object serialization, as described in 2.4.2.

The meta-data management modules were developed using two distinct technologies. For indexing DICOM objects meta-data, a Document-Oriented database was used, named Lucene [45]. The options is based on reports of success story in other projects, such as Dicoogle [43]. In order to ensure persistent mapping between images and image chunks, the DB4Ojects database

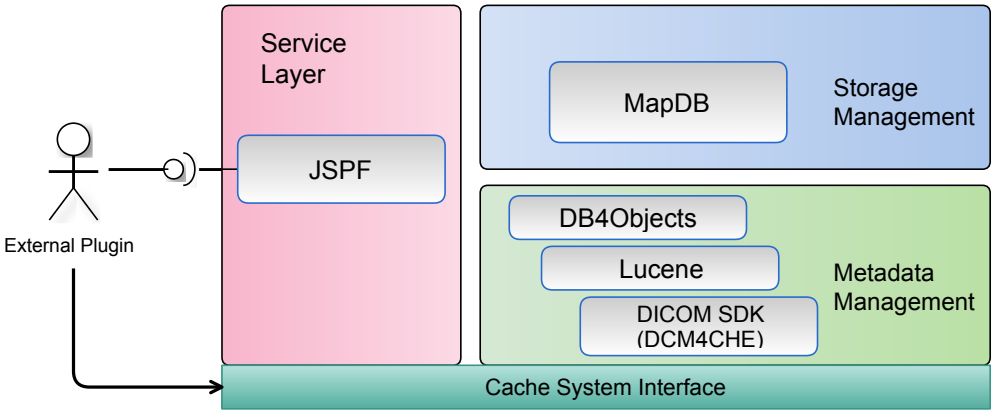



**Figure 4.10: Cache System Frameworks**

was used. DB4O [46] permitted a quick migration of our previously developed data module into the database. As opposed to the usage of a SQL database, that required its migration to a relational paradigm. Nevertheless, in terms of performance the usage of DB4O proved to have no relevant impact on the performance of image population (around 300ms for our study dataset). Compared to the previously used Sqlite4java [47] database wrapper, the DB4O proved to be faster. In our opinion, this had to do with the fact that DB4O's engine also provides some caching functionality for objects. It somehow seems not to be present is most embeddable SQL databases.

The plugins support was achieved easily through the usage of the JSPF framework [48]. It provides a very fast and seamless integration of third-party plugins in Java applications. Moreover, this framework also simplifies the development of plugins, as developers only have to follow the defined interface.

The integration of proposed cache system architecture in the DICOM Router was easily achieved. Although it involved some modifications since the previously images were stored exclusively on the file-system. Using our cache system, objects (images or image chunks) are imitatively stored in the router local cache, independently of its origin, i.e. they can come from an external application or from another DICOM Router (via a Controlled Channel). If there is meta-data available, chunk descriptor or DICOM meta-data, it is also cached in the system.

### 4.3.3. Multi-level cache to support multi-source Query/Retrieve Services

The proposed cache system allows the deployment of a hierarchical multi-level caching system in the DICOM Router environment. To support a distributed PACS environment, we propose the deployment of two major levels, local and distributed.

The local cache is deployed along with each DICOM Router. Its configuration in terms of capacity is subjected to the routers user computational system. Therefore its primary function is not to cache PACS archives contents, but rather support the routers workflow. Whenever a C-FIND or C-Move request is issued by an appended application, the Router forwards it to its local cache primarily. The images cached inside the system are immediately forwarded to the client's application consequently avoiding loading them from a remote location.

The local caches population is intended to be done along with the routers life-time, taking advantage of information collected from previous C-Store-Requests. This favors the application of pre-fetching techniques either by external cache plugins or external entities, such as, other Routers or Caches in the distributed system. As a result, contents can be cached through the normal Storage Service.

Distributed caches are intended to favor the distributed system as a whole, providing extended availability for archives and content replication. Distributed Caches are not intended to be populated along the life-time of the DICOM Routers but rather with pre-fetching techniques appointed by system administrators. The Distributed caches present a great asset to the distributed PACS, not only by increasing overall study availability but also by enabling the deployment of replicated archives on public cloud locations. This extends the features of the distributed system increasing its appealing to medical institutions. An example of usage of the distributed caching capabilities would be to deploy a cache on a public cloud provider containing the studies conducted in the previous week or month. This would enable a better performance and availability for study retrieval, external to the institutions archive local network.

The implementation of distributed caches is similar to the local caches, as it involves the deployment of a DICOM Router that advertises a specified PACS archive. The router does not

have to be connected to the archive, as it was initially intended. Nevertheless, it is able to handle all requests that will be answered by its cache system. Routing requests to these DICOM Routers is exactly the same as for normal routers (with appended archives). As such, these routers need to advertise the desired archives *AETitles* even though they are not appended to them. These routes are configured normally through the routers configuration interface. Although it is not strictly required, these routes should be marked as *Cache Routes* for better handling of Query Requests.

The existence of multiple DICOM Routers advertising the same PACS archive required a few changes on how the DICOM Query and Retrieve Services are issued by the distributed system. Previously, these services were supported in a very unicast manner, with communication essentially between a pair of DICOM Routers as shown in [17]. Now, the distributed PACS can include several temporary repositories (caches) and, potentially, many replicas of the same information. The main PACS archive has always all studies. As a result of having seemingly multiple instances of the same archive across, the communications paradigm changed from unicast to multicast. As the communications are intermediated by the DICOM Bridge Router, this change of paradigm will be abstracted from the router's point of view and the modifications have been performed in the DICOM Bridge Router, explained bellow.

### *Query Service*

DICOM Query Service with multiple instances of to the same archive is supported relying on a hierarchical approach. The main idea is to forward the C-Find-Request iteratively to each instance until a positive match is returned. Therefore, the routes should be organized in a priority queue. The first route would lead to the actual PACS archive router, if available. These workflows are illustrated in Figure 4.11.

C-Find-Request arrives to the SCU application router and, if local cache is available, it is forwarded to this entity. In parallel, the request is also forwarded to the distributed system. As it eventually arrives to the DICOM Bridge Router, a small C-Find session is created. The bridge then starts the forwarding procedure. Iteratively, it forwards the request to each route in the queue described above. The bridge analyses each response from the inquired DICOM Router. In case it could not be fulfilled, the bridge iterates and forwards the request to the next Router. When a fulfilled response is returned, the C-Find session terminates. The response is then re-routed to the original router. After receiving the response, the router forwards it to the requester DICOM application.

All this extra-work is preferable to responding directly from the DICOM Routers cache. Because DICOM studies can have new series inserted after the study creation. By doing this, we are giving the chance to DICOM Routers to check if their cache meta-data is updated, enabling more accurate responses if the actual distributed system becomes unavailable.

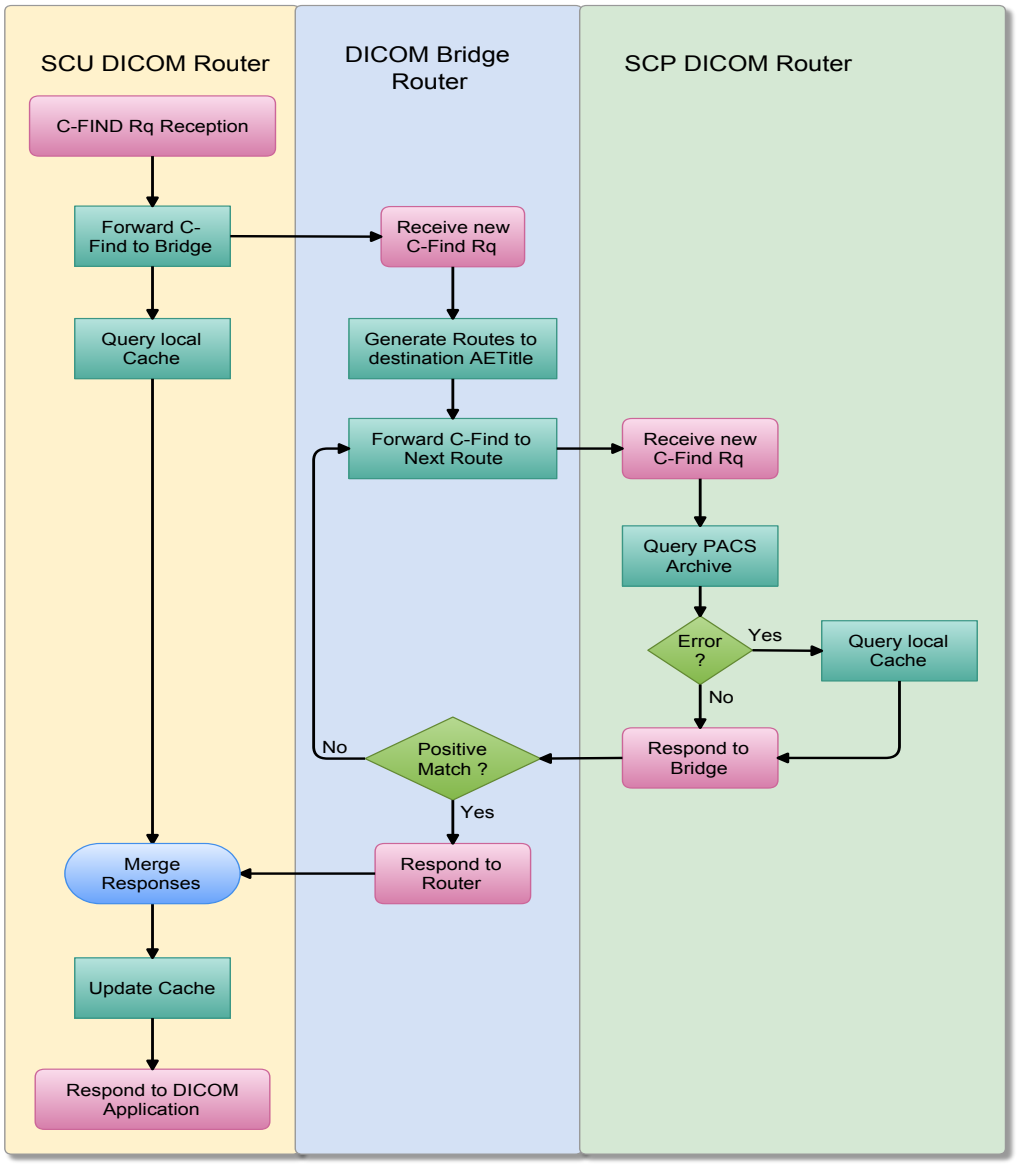



**Figure 4.11: Multi-Source Query Service**

### *Retrieve Service*

The support for the C-Move Command in the distributed environment could be done in a similar way as the C-Find Command. However, the deployment of caches in the DICOM Routers permits, not only to increase the overall availability of PACS Archives but also the performance of study retrieval, exploring the possibility of downloading DICOM Objects from multiple sources.

In order to take advantage of this multiple sources, it is proposed that the DICOM Retrieval Service could be issued by more than one provider in parallel, including the DICOM Routers local cache. The main idea is to split the objects to be retrieved by multiple providers so that each provider would only have to upload a subset of the study and not the whole study. In order to achieve this, we had to perform a huge refactoring in the previous implementations of C-Move commands, as shown in Figure 4.12.

Firstly, the SCU application Router receives the C-Move-Request. Next, the request is forwarded, in parallel, to the router's local cache *(1.1)* and to the distributed environment *(1.3)*.

The objects presented in the router's local cache *(1.2)* do not need to be retrieved from distributed environment. Thus, a ***"prune work-list***" containing those objects is transmitted along with the request message to the DICOM Bridge Router. These objects are immediately forwarded to the application.

In order to manage the multi-source retrieve process, the bridge router creates a small retrieve session upon the reception of the C-Move-Request along with the ***"prune work-list"***. This session will enclose the work-flow described further. After creating the session, the request message, along with the "***prune work-list***" is broadcasted to every DICOM Router advertising the given SCP (archive) *(2)*. As such, in order to be illegible to issue C-Move requests, DICOM Routers have to advertise the PACS Archive *AETitle*, as if they have the archive appended.

Then locally, each router follows the same procedure in order to answer the request. Firstly, it forwards the request to the archive. If the archive proves to be unreachable, the request is forwarded to the routers local cache. This allows distributed cache Routers (which are not appended to the archive) to respond to C-Move-Requests. In order to respond to the bridge router, each DICOM Router has to merge the "***prune work-list***" with its own response. The idea is to produce a list of DICOM Objects that are not presented in the SCU router local cache. The merged response is then sent to the DICOM Bridge Router *(3)*.

Meanwhile, in the bridge router, the retrieve session is waiting to receive the responses from all routers. When that happens, it will perform the scheduling of the upload process. For each DICOM Router, a list containing the objects (images or image chunks) that the DICOM Router has to upload to the C-Move SCU router is created, also known as "***upload work-list***". The ***"upload work-list"*** is sent back to every router in the session *(4.1 – Router B and C)*. Along with the "***upload work-list***", a list of the Image SOP Instance UIDs involved in the transference is also produced. This list is sent back to the SCU application router *(4.2)*. It serves as a close acknowledgement message, as well as enabling the router to know beforehand which images will be involved in the transference.

Upon receiving the ***"upload work-list"***, each router starts uploading the objects to the SCU Router, as described in section 4.2. In case the objects are not presented in the routers local cache, a C-Move-Request is issued to the PACS Archive in order to make them available in the Routers local cache.

The algorithm for producing the ***"upload work-lists"*** uses a greedy approach, i.e., it calculates every list for every router at once, based on the current state of the system. It tries to balance the load of the distributed system by counting how many objects each router has to upload. In larger studies, the complexity has to be taken into account as it scales linearly with the number of objects.

For this reason, the most used approach for scheduling is described in [42]. It uses centralized ***"upload work-lists"*** instead of a distributed, which are our case. Uploading nodes poll a new item from the list each time they complete their last task. As such, the scheduling is performed along the transference time minimizing the computational requirements at the expense of some communications overhead. However, in our scenario this communication overhead would lead to worse performance, as communications latency between DICOM Router and the DICOM Bridge Router tend to be significantly high (in an internet scenario).

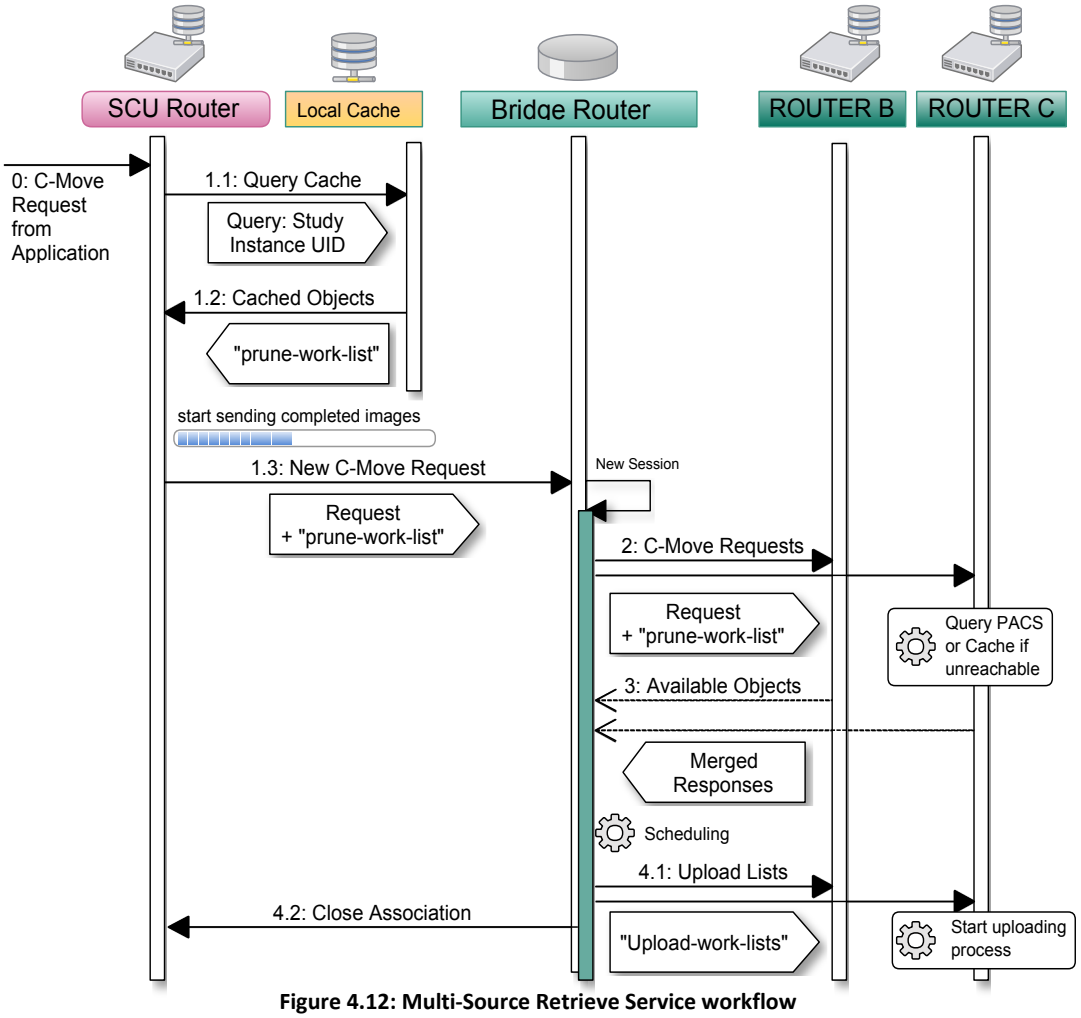



**Figure 4.12: Multi-Source Retrieve Service workflow**

## 4.4. Bridge Replication

The replication of DICOM Bridge Router is fundamental to reduce its single point of failure effect (SPoF). In fact, bridge replication represents the creation of multiple points of failure. Multiple points do not mean that problems will cease to exist, but it will be reduced the probability that those failures compromise the entire system. Generally, public cloud providers have extremely high availability ratios, bigger than private datacenters [5, 49]. Therefore, the deployment of components on public cloud providers will benefit from almost total infrastructural availability. The combination of both replication and cloud deployment results in an almost flawless system.

In this distributed PACS architecture, the DICOM Bridge Router supports all distributed communications (as described in 3.2.5). It not only represents a single point of failure, but also a possible performance bottleneck. Consequently, the DICOM Bridge Router is a good candidate for replication along with the implementation of load balancing techniques. This section proposes the integration of these methods in our distributed PACS environment.

### 4.4.1. Proposed Approach

As described previously, DICOM Bridge Router is a single point of failure, but also a bottleneck on PACS distributed system. Although the Bridge Router replication was pointed as a solution, it raises some issues regarding their deployment, described in sequel.

Each DICOM Bridge Router requires a good network bandwidth, both upstream and downstream, in order to relay communications efficiently. This requirement is usually not compatible with institutions private networks, as they support a great number of workflows with non-specialized network infrastructures. However, this solution is preferable taking into account the security concerns, as these components are deployed in trusted environments. These security considerations will be presented and discussed in chapter 4.5, along with the concept of safely deployed components.

The proposed architecture is based on multiple DICOM Bridge Routers that may be deployed in multiple locations, private or public. According to its deployment location, system administrators may configure each bridge router to be a safely deployed entity or not. To support this feature, a configuration interface for the DICOM Bridge Router was developed. This interface uses common XML configurations files. As multiple DICOM Bridge Routers are deployed in the system, administrators might want to identify each bridge router. This is supported via the definition of bridge identifiers (*Bridge ID*), also configured in the previous interface.

The replication of bridges can be used to increase the performance of the distributed system. It was introduced a concept of load factor to take advantage of Bridge replication mechanism. It is a metric associated to every DICOM Bridge that represents how many services a given bridge is issuing at the moment. So, it will be easy to know which bridge is the best to issue the service, when the router requires it. This metric is based on the number of active communication channels (control and data) in the bridge. In order to better represent the bridges load, each channel may, as well, be associated with a weight factor. For example, data channels (such as controlled channels) may have twice the weight factor of control channels, as they require more resources, due to the data transfer requirements. Moreover, a base load factor might be defined to statically tune each bridge priority.

Furthermore, DICOM Bridge Routers can be linked together. The inter-bridge communication channels were created to support this feature. These channels use the same communication module as normal communication channels, and they have the same properties, such as, encryption. Bridge Links may be configured through the previously defined interface. The Inter-bridge communication protocol was thought to enable the creation of groups of bridges, with functionalities shared among them, such as, user authorization. Moreover, this protocol was intended to support redirection of messages between bridges, thus supporting services redirection. However, this approach was abandoned as it consumes more DICOM Bridge Routers resources than concurrent techniques (see 4.4.2.

Although we specified an inter-bridge communication protocol, our replication technique does not necessary involve the multiplexing of a single service between a set of bridges. Our technique proposes to balance the load of the distributed system, instead of balancing the number of services being issued by each DICOM Bridge Router. Moreover, this allows complete compliance with the previous versions of our architecture.

### 4.4.2. Routers working with multiple Bridges

In order to support multiple DICOM Bridge Routers, we changed the architecture presented in previous sections. Our major commitment was to minimize its performance

footprint in the system. Additionally, as also described above, we tried to keep compatibility between this version and previous ones. Thus, we realized that it was better to take this management overhead off the DICOM Bridge Routers and put it into the DICOM Routers.

Firstly, DICOM Routers are responsible for connecting to multiple bridge routers. These DICOM Bridge Router addresses are configured via the DICOM Router configuration XML. DICOM Routers connect to every bridge router simultaneously. There were other possible approaches, such as, making the router to be connected to one bridge at a time. However, this required DICOM Bridge Routers to manage the service connections across multiple bridges. This could be achieved by resorting the redirection messages, although it imposed more performance constraints by requiring more Bridge resources and message exchange. So, it was decided to adopt the simultaneous approach. Even though having multiple control channels connected to different bridges also wastes some resources, we consider it minimal. Moreover, being connected to multiple bridges at the same time provides better service in case of a bridge failure, as other bridge router may be used without prior processes.

An important issue is to detect which is the best bridge to use. In fact, DICOM Routers are the responsible for balancing the load of DICOM Bridge Routers. As expressed, load balancing is achieved resorting to the DICOM Bridges load factor that is announced in Keep-Alive messages. DICOM Routers keep track of the most recent load factor for each bridge, creating an “extended routing table”. When a new channel needs to be established, the router selects the bridge with the lowest load factor. Using this technique, we can achieve load balancing without any significant computational requirement.

## 4.5. Security Model

Healthcare information systems tend to be associated with very tight security considerations. They often manage sensitive patient data, such as, patient names, contacts, diseases and many others. If these kinds of systems are deployed on public cloud providers, the security considerations need to be even tighter, as sensitive patient data might be stored, processed or even consumed by untrustworthy computational nodes. In fact, public cloud providers are known to follow an “honest-but-curious” approach regarding their client’s data. They comply integrally with the contracted service, SLA (Service Level Agreement), although any guarantees are given that they won’t misuse their client’s data, either by selling it directly or by applying data mining techniques [42].

Zhang et al [50] identifies the common issues associated with the deployment of Healthcare information systems on cloud environments. The most important for our scenario are information ownership, authenticity and authentication, patient consent and authorization, and data integrity and confidentiality. The distributed PACS environment proposes to support DICOM connectivity across multiple locales. So, the ownership of information and patient authorization is out of the scope. It is therefore delegated to other institutional application the management of both these issues. DICOM Bridge Routers provide user authentication in our system. Authenticity of those users is also guaranteed through the usage of the HTTPS protocol in communications and a credentials system based on username and password. These features are migrated peacefully to unsafely deployed components.

The problems associated with the deployment of components on untrustworthy locations reside, essentially, on data confidentiality and integrity. Data confidentiality means that sensitive patient data is stored safely, therefore only visible to trustworthy entities. In our architecture it means that unsafely deployed DICOM Routers must not store any sensitive DICOM meta-data.

Thus, unsafely deployed DICOM Routers may not index objects meta-data, at least in clear text. Further in section, it will be presented a technique for supporting the cache Meta-data management modules of unsafely deployed routers.

DICOM Objects meta-data can also be retrieved from the actual objects itself. Therefore, the storage of these objects is also a critical process in terms of security. We propose to use the MapDB encryption capabilities to securely store these objects. The cached objects cannot be retrieved by a third-party workflow without the right pass-phrase.

### 4.5.1. Handling sensitive meta-data on unsafely deployed components

As stated in section 4.4, our cache system indexes the DICOM Objects meta-data in order to respond to the queries. In its turn, the cache query interface enables DICOM Routers to answer to DICOM Query Services (*C-FIND commands*) without requiring direct connectivity to the PACS archive. This is a crucial feature to proposed architecture, as it allows distributed cache routers to operate with complete autonomy, i.e. they are able to support DICOM Query/Retrieve Service by themselves.

In order to support this feature on unsafely deployed DICOM Routers we used a novel Searchable Encryption (SE) technique, named Posterior Playfair Searchable Encryption (PPSE) [51]. SE techniques provide query capabilities over ciphered data. Therefore, it is possible to have indexes encrypted in untrustworthy repositories. However, those indexes continue to be used in search operations. In our scenario, we will have indexes of DICOM Objects meta-data.

Other approaches were identified. In [32] decouples the meta-data index from the actual object storage. However, it requires the deployment of two components on distinct locations because the meta-data index must be stored in a trustable place. In our case, it would reduce the DICOM Routers autonomy, which is not desirable, due to requirements to tackle with failures of network links.

The proposed distributed PACS architecture provides enough components to deploy the PPSE algorithm. The PPSE algorithm works by ciphering content using a secure key and an unsafe provider may index this codified data. In order to query contents in this provider, the query phrase also has to be firstly ciphered. As a result of having ciphered data indexed, the query response has to be deciphered itself, in order to retrieve the plain text data. Therefore, the PPSE algorithm requires a secure location for encryption and decryption operation to be performed. With this approach, it was possible to implement the concept of safely deployed DICOM Bridge Routers.

#### *Untrusted meta-data index*

Our proposal is not to handle sensitive patient meta-data on unsafely deployed components. Instead, we aimed to handle this data on safely deployed components and let unsafely deployed components handle only ciphered data, through the usage of the PPSE algorithm. Figure 4.13 illustrates how unsafely deployed DICOM Routers index this data.

Images arrive to DICOM Routers via a controlled channel (described in section 4.3). They arrive compressed and split in chunks, as a result no meta-data can be extracted from each chunk. However, chunk descriptors carry the needed information to assemble the multiple chunks back to one piece. This information is not critical, as it does not enclose any patient data. When the complete image is available, DICOM Routers assemble the multiple chunks into the DICOM Object in order to extract and index its meta-data. Unsafely deployed routers should not perform this action, as it would expose the meta-data in memory. In fact, they never try to

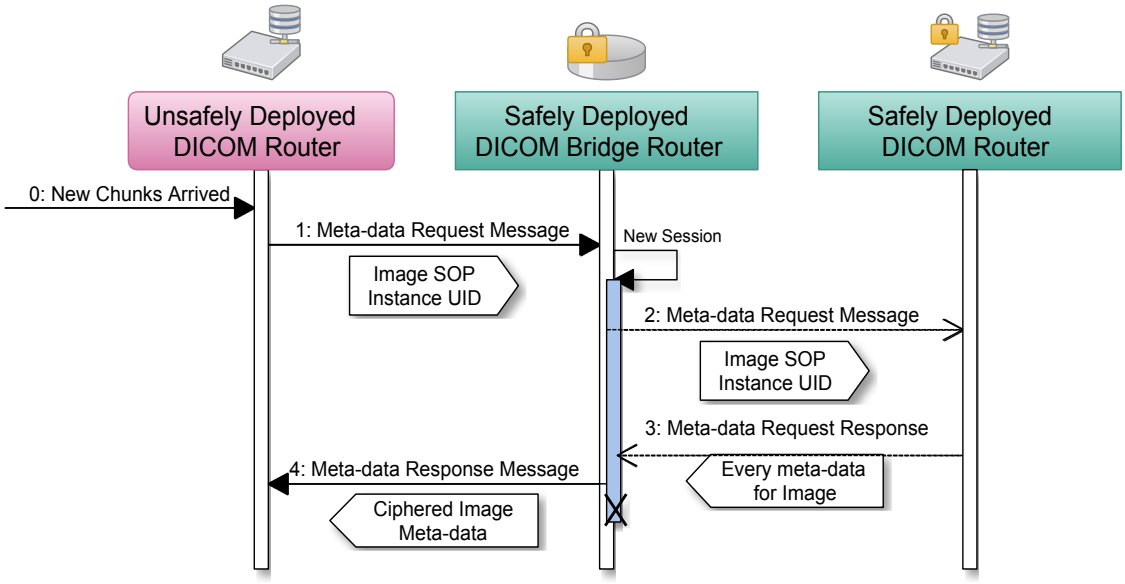



**Figure 4.13: Indexing Meta-data on unsafely deployed routers.**

assemble any image. Each time an image chunk arrives from a controlled channel, the router immediately checks if it has cached the meta-data of that image. This is done using the Image SOP Instance UID from the chunk descriptor. If there is no cached meta-data for that image, the router requests it via a safely deployed DICOM Bridge Router.

Safely deployed DICOM Bridge Routers provide encryption/decryption of meta-data but they do not index it. It would be a very complex task to index every DICOM Objects meta-data, due to performance components constrains. In our distributed PACS architecture, the image meta-data is indexed at the DICOM Routers cache modules (as described in section 4.3). As a result, DICOM Bridge Routers create a Meta-data Request Session.

The meta-data request session is very similar to the C-Find Session described in 4.4.3. The DICOM Bridge Router forwards a Meta-data Request message to each DICOM Router iteratively until a positive answer is returned. However, DICOM Routers do not respond to the Meta-data Request message with multiple DICOM Objects. Instead, they respond with a Meta-data Response message, holding the indexed meta-data for that image. Unsafely deployed routers cannot cache plain DICOM meta-data. Therefore, the DICOM Bridge Router uses the PPSE algorithm to produced searchable ciphered data. This data is then forwarded to the requesting router, making the session termination. Note data some DICOM meta-data is not ciphered, mainly the identification fields such as Study UID or Image SOP UID. This information is not considered sensitive as it does not expose any patient information.

### *Query/Retrieve on unsafely deployed Routers*

The DICOM Query Service is issued seamlessly on unsafely deployed routers, as the PPSE algorithm enables queries on ciphered data. As queries also need to be ciphered, the DICOM Query service can only be issued by safely deployed DICOM Bridges. The procedure to issue the queries is similar to the described in 4.3.3. The only difference is that the bridge router has to know beforehand if each router is safely deployed or not. This is done through the usage of keep-alive messages between DICOM Routers and DICOM Bridge Routers. Keep-Alive messages are not only used as described in section 4.4.2, they also enclose a flag indicating whether the router is safely deployed or not. By using this information, the DICOM Bridge Router knows whether or

not it needs to cipher the query and decipher the responded meta-data. This workflow is illustrated in Figure 4.14.

Concluding, encrypted index of unsafely deployed DICOM Routers are only accessible (i.e. searchable) through a safely deployed DICOM Bridge because they have the PPSE keys to encrypt/decrypt queries. However, unsafely deployed DICOM Routers may answer to C-Move Requests normally. DICOM Retrieve Services usually rely on objects unique identifiers (e.g. Study Instance UIDs) that are not ciphered in the routers cache.

### *Assessment*

The technique adopted to support secure deployment of components is robust. Moreover, it does not compromise the performance of the distributed system, because safely deployed components have their workflows favored.

However, there is a minimal possibility for accessing DICOM Objects on unsafely deployed components. It involves dumping the Java virtual machine memory in order to collect every image chunk before they are cached securely. We do not give too much importance to this minimal leak as it involves taking massive memory dumps, identifying each image chunk and then use the correct decompression method. Moreover we took a major caution to evict those objects from memory, as soon as possible. Therefore, many dumps would be necessary to catch a whole image.

There is also the possibility of a third party guessing either the MapDB's database pass-phrase, or the PPSE algorithm key. If such a thing happens, either the pass-phrase, or the key should be replaced as soon as possible. The MapDBs databases would have to be totally wiped and the cache module meta-data index would have to be ciphered with another key. The PPSE algorithm key needs to be shipped with every safely deployed DICOM Bridge Router. This key is unique and every DICOM Bridge Router should have the same key to achieve maximum compatibility. In our point of view, this is in fact the only limitation of the PPSE algorithm as it discourages its usage in scenarios where there are not only multiple consumers but also multiple producers.

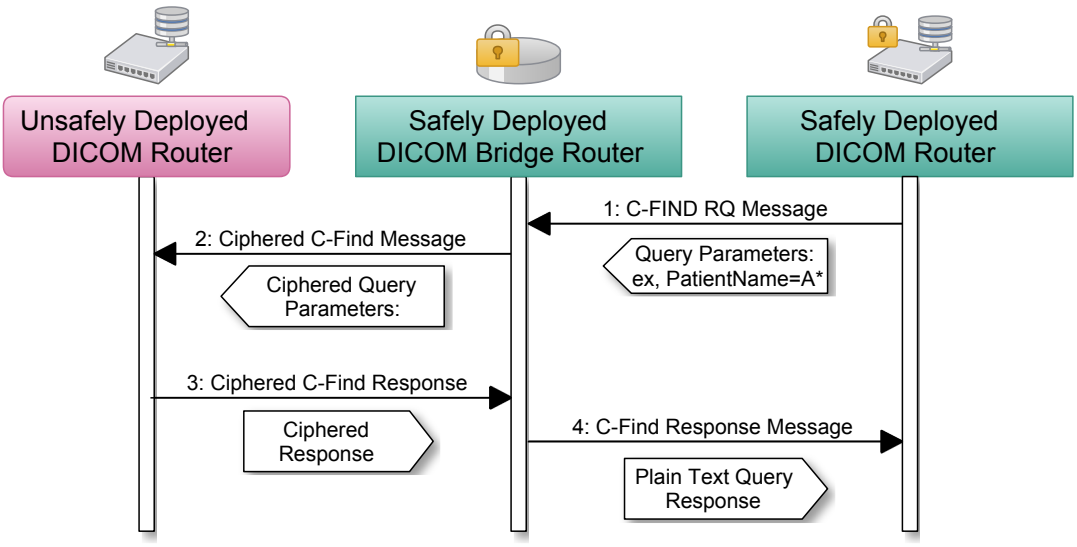



**Figure 4.14: Queries over unsafely deployed routers.**

# 5. Results and Discussion

This chapter presents performance trials conducted with the different proposed techniques. It aims to show their effectiveness, comparing different approaches to achieve the same goal, measuring and assess the benefit of each solution.

The used dataset was carefully selected to represent multiple studies and data profiles. It encloses 7 medical imaging studies from 4 different modalities, summing a total of 863 images with a volume of 682 MB. In Table 5.1 is represented the multiple studies with its volume (expressed in MB) and the average image size. By analyzing this table, it is possible to verify the impact of the different modalities in the data profiles. It is easily to check the difference in the average image size, even within the same modality.

## 5.1. Controlled Channels

This section provides a description of our performance trials for the controlled channels method. We deployed a test bed scenario, involving a distributed PACS environment with a PACS Archive and a remote storage application (simulating a viewer application). Dcm4che was used to deploy both applications. The applications were deployed in a remote Internet environment. The PACS Archive was served by a 10 Mbps upstream connection, while the application had 30 Mbps downstream bandwidth. This simulation represents carefully a real world use-case scenario of a distributed PACS.

Trials were performed with three transference methods. Firstly, using a VPN connection to link directly both networks as suggested in [9]. The second method used normal communication channels in our distributed PACS as explained in chapter 3.2. Lastly, the Controlled Channels method was tested in the distributed PACS architecture (see chapter 4.2.1), as proposed in this thesis. The controlled channels were configure to use a maximum of 3 parallel channels, 10 chunks per bulk (bulk size), each chunk with 50KB (chunk size) and 1 second of maximum waiting

**Table 5.1: Study Dataset Table**

| Study Modality | Number of images | Volume (MB) | Average image Size (KB) |
|---|---|---|---|
| NM | **1** | **1** | **1.000** |
| NM | **5** | **2** | **400** |
| NM | **6** | **8,2** | **1.367** |
| PT | **244** | **16,3** | **67** |
| MR | **223** | **47,1** | **211** |
| MR | **369** | **206,1** | **559** |
| XA | **15** | **401,6** | **26.773** |
| Total | **863** | **682,3** | **791** |

time.

Our distributed PACS was supported by a DICOM Router along with each application. The DICOM Bridge Router was deployed in the PACS Archive machine. This configuration is based on a common deployment scenario also used as a case study. The Controlled Channels deployment required a few changes to the image handling process, i.e. the process that deals with the image in the transference (as described in 4.2.1). We tried to minimize the performance effects of this change. Therefore, every DICOM Router version involved in the test had its image handling process changed. In the introduced process, images were handled completely in memory. As the memory offers almost real-time performance, fewer differences in performance were introduced by each different approach. Consequently, more accurate results were produced regarding the Controlled Channel efficiency.

The time needed for the system to issue each study storage request was registered by the dcm4che application (see Table 5.2). Analyzing the results, it is easily to verify that the Controlled Channels provide a clear performance improvement compared to the concurrent methods. Moreover, better improvements are achieved for studies with higher average image size which are normally associated with higher transference times. For smaller studies the archived improvements are not significant when the system is only supporting a single user.

## 5.2. Cache Module

The benefits of our cache module are unquestionable because it provides major features to our distributed PACS architecture, such as, supporting an offline archive without any transference over the network. Despite of the clear benefit, it also introduces a disadvantage, i.e. the DICOM Router needs to have a predefined storage space to support the cache, which is customizable in our case. The performance benefits of caching PACS Archives content will be demonstrated supported on controlled trials. To achieve this goal, we deployed a PACS Archive and a storage provider in our distributed PACS architecture. Once again Dcm4che [52] was used in both applications.

In order to account the benefits of the usage of our cache module under different network

**Table 5.2: Controlled Channels trial results**

| Modality | Number of images | Volume (MB) | VPN (s) | DICOM Router with normal channels (s) | DICOM Router with Controlled Channels (s) |
|---|---|---|---|---|---|
| NM | **1** | **1** | **2,9** | **2,4** | **2,6** |
| NM | **5** | **2** | **5,7** | **5,2** | **3,6** |
| NM | **6** | **8,2** | **12,2** | **6,5** | **5,5** |
| PT | **244** | **16,3** | **27,4** | **137,6** | **33,5** |
| MR | **223** | **47,1** | **59,0** | **126,7** | **32,8** |
| MR | **369** | **206,1** | **264,4** | **210,0** | **82,4** |
| XA | **15** | **401,6** | **622,8** | **549,3** | **350,4** |

configuration, two distinct network scenarios were produced. In the first scenario the c-move application host was served by a 30 Mbps downstream bandwidth. While in the second scenario this bandwidth was reduced to 5 Mbps[4], simulating a more restrictive environment. In both scenarios the PACS Archive host was served by a 30 Mbps upstream bandwidth.

The final version of our distributed PACS architecture was deployed, using all methods proposed in this document. A DICOM Router was deployed along with each application, ensuring connectivity between both environments with the controlled streams. A DICOM Bridge Router was also deployed along with the PACS Archive and its router (similar to 5.1). The Controlled Channels configuration was the same used in the previous trial, 3 parallel channels, 10 chunks per bulk (bulk size), each chunk with 50KB (chunk size) and 1 second of maximum waiting time.

The trials aimed to show the performance speed up achieved by combining multiple cache population scenarios in both DICOM Routers. The external location router was populated with 25%, 50%, 75% and 100% of each study. Meanwhile, the PACS Archive Router had the entire study or no images cached at all. In this case testing intermediate scenarios is pointless, as they all require a C-Move Request to the archive. Thus they not create any real advantage to an institution. Each study retrieval time, without any image cached in both routers, was taken into reference to calculate the speed-up. The reference times are presented in Table 5.3.

The speed-ups were calculated independently for each study of our dataset in each of the cache scenarios. In Figure 5.1 are presented the average speed-ups for each scenario, i.e. the mean speed-up of all studies for the given percentage of the study in cache. The results show consistent improvement of the transference performance for all scenarios. Even with little study portions cached the speed-up is considerable. The effect of the PACS archive router local cache on the transference time is even more incredible. Studies cached in both routers benefit from a speed-up, as the C-Move Request to the Archive is avoided.

These trials represent a real-world scenario, where an institution might want to cache content of foreign repositories to improve the quality of service of their users. Moreover, they justify the adoption of study pre-fetching techniques by institutions as a mean of getting the

**Table 5.3: Cache Module trial results**

| Modality | Number of Files | Volume (MB) | Completely non-cached study (S) | |
|---|---|---|---|---|
| | | | 30 Mbps | 5 Mbps |
| NM | **1** | **1** | **3,45** | **5,43** |
| NM | **5** | **2** | **4,23** | **7,24** |
| NM | **6** | **8,2** | **5,57** | **11,63** |
| PT | **244** | **16,3** | **13,81** | **35,94** |
| MR | **223** | **47,1** | **19,09** | **52,22** |
| MR | **369** | **206,1** | **45,96** | **144,80** |
| XA | **15** | **401,6** | **200,75** | **670,73** |

[4] The 5 Mbps connection is considered the average connection speed in Portugal by Akamai [53].

maximum potential of their cache routers. It is proved that caches may have an outstanding role in the overall performance of the system.

Figure 5.1 also illustrates very well the effects of cached content in different network conditions. It is easily perceivable that for low speed bandwidths, the effect of the archive router's local cache does not have the same importance as in higher speed bandwidth configurations. This is the result of a lower disparity between the router-to-router transference throughput and the available images throughput (via PACS Archive in this case), that keeps the Controlled Channels queues from emptying. However, for high cached percentages (>75%), its effect may be considered of major importance. By analyzing both charts, it is also clear that caching has major speed-up benefits, especially in lower bandwidth networks.

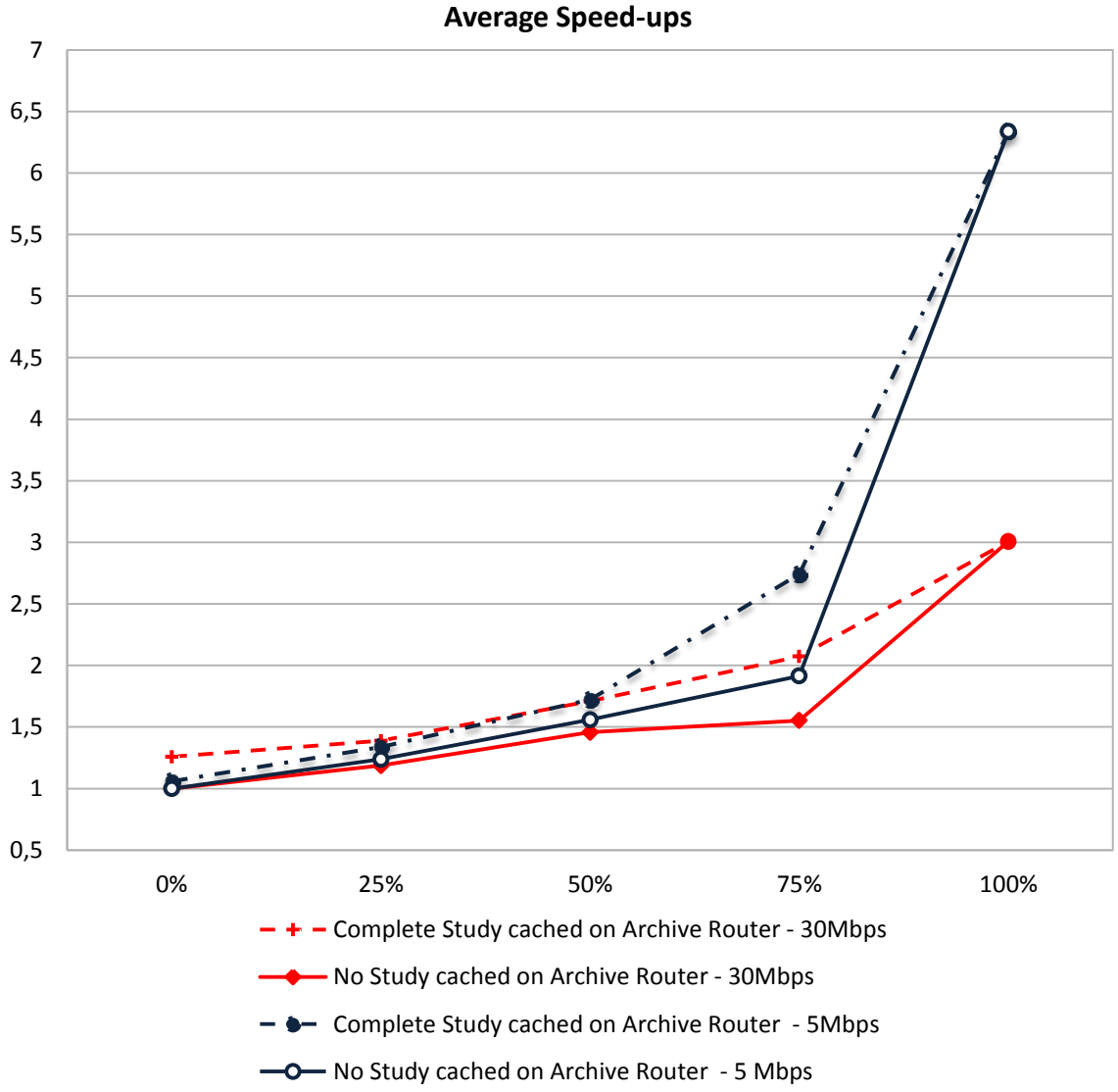



**Figure 5.1: Comparison of cache module average speed-ups in different scenarios.**

# 6. Conclusions and future work

## 6.1. Conclusions

Nowadays medical imaging environments tend to be migrated to web-based PACS solutions. One of its main features is the capability to integrate multiple institutions in the same PACS providing external access to the resources [8]. Moreover, this particular type of PACS architecture relieves medical institutions of supporting complex IT infrastructures. However, this approach has also some challenges. Web-based PACS architectures do not usually provide compatibility with previously existing institutional resources, such as, previous PACS Archives, acquisition image devices or DICOM compliant applications. Besides, migration to web solution imposes some legal constrains related to sensitive patient data exploitation. Common web-based PACS still face strong resistance within the medical community due to some lack of functionality, such as 3D reconstruction or other complex features that are not typically available on web viewers. Nevertheless, they desire a solution that provides integration of multiple institutions, including previous equipment, reduction of cost with IT infrastructures and requiring minimum configuration efforts. Furthermore, the data sources should always be available and it is fundamental to have access to medical imaging studies in a rapid way.

Our distributed PACS Architecture complies with this specification by interconnecting multiple institutions with DICOM compliant communications. Moreover, it provides effective re-usage of previous institutional resources, such as, Internet, PACS Archives and visualization workstations. In fact, a previous version of our architecture is currently deployed and has received good acceptance among the medical community. However, there were also some issues associated with this version. Firstly, the performance of medical study retrieval could be optimized. This problem is aggravated with the fact that Portuguese Internet connections usually provide more downstream than upstream bandwidth. Secondly, this architecture relied completely on standard institutional PACS Archives, clearly opposed to the normal web-based PACS architecture. Lastly, the previous system had a single point of failure that compromised the overall reliability of the system.

In order to solve these issues, this document proposes multiple improvements to previous architecture. Namely, replication of components to increase the overall system reliability (see section 4.4), an improved data transference protocol to better cope with the medical imaging data specifications (see section 4.2), and techniques to support common web-based PACS features, such as, cloud deployment of archives (see section 4). The result was an even more versatile system. It provides a distributed web-based PACS, with optimized features and processes, preserving the specifications mentioned above. Despite the obvious raise of architecture complexity, it is not perceivable to PACS users. .

When analyzing the pertinence of proposed methods, we find that each solves a specific problem. The Controlled Channels method was proposed in order to make the data transference more compliant with medical imaging studies data profiles. As a result, section 5.1 shows that improvements in study transference are notorious. Moreover, they require less routers resources than the previous method. As the Controlled Channels are endpoint oriented, they also contribute to the scalability of the system.

The proposal of our cache system for medical imaging repositories proves to be a major contribute of this thesis. Our cache system has made possible the improvement in the DICOM

Routers image handling process. Combined with the proposed Controlled Channels and image splitting technique, this cache system provides replication of PACS Archives. The result is a better performance in image transference, through the usage of multi-source image transference, and a better availability of studies. Moreover, our distributed PACS architecture supports the deployment of repositories on public cloud providers, like common web-based PACS.

The proposed security model provides effective guarantees that sensitive patient data is not exposed in cloud environments. The model has merit by itself, as it preserves the system performance and it does not introduce any further components or excessive complexity. Better system reliability and load balancing was achieved using the Bridge Replication technique. This was also a major improvement, as it eliminates single points of failure existing in the previous architecture.

Concluding, all these techniques combined provide a better quality of service to medical institutions. The proposed system was designed to increase data availability and reduce data access latency in distributed PACS environments. It was instantiated over a Web DICOM Routing mechanism, demonstrating an easy integration due to its architecture and DICOM interface. Moreover, the solution proved to be useful in other distributed medical imaging processes. For instance, the Routing mechanism has been used to support tele-radiology sessions where physicians are remotely reporting examinations that were produced by distinct institutions. Compared to other low cost competitors, such as VPN connectivity, it provides much more features with better performance. This version of our architecture should be deployed in our case study environment in a short term.

From the scientific point of view, this thesis as produced a conference paper regarding the usage of the Controlled Channels method. The paper was presented in CISTI 2013 [17]. Moreover, another article named ***"A DICOM cache mechanism to support distributed PACS environments"***, giving a special focus to our cache system architecture, was recently submitted for revision at IEEE Journal of Biomedical and Health Informatics.

## 6.2. Future work

During this thesis, we completely fulfilled the objectives defined in the proposal. As such, we opened the door for further work and research directions by taking advantage of these thesis conclusions and also using the developed frameworks.

The Controlled Channels performance may be improved by the automatic reconfiguration of their intrinsic parameters. The study of automatic algorithms to perform this task is possible through the developed framework. We believe this is a very interesting area for future research directions which may further increase the performance of Controlled Channels, eventually leading to less resource consumption in DICOM Routers.

Two technologies were introduced in chapter 2.5. They aim to provide direct connectivity between DICOM Routers. Since the proposal of the Controlled Channels technique, we realized that these frameworks may be adaptable in the proposed architecture. The idea is to free the DICOM Bridge Routers from the burden of relaying data transference channels (Controlled Channels) whenever possible.

In terms of the overall system's features, it is important that further contributions are made in order to keep the attractiveness of our architecture. An example is the automated deployment of cache routers on public cloud providers. This would enable the scalability of our architecture in terms of storage capacity but also in performance terms. Regarding our cache

system, it provides a great framework for the development of a wide variety of cache strategies. The development of new strategies also may be an interesting research field. Moreover, the use of prediction techniques may result in further intelligent population of both local and distributed caches, maximizing the economic efficiency of our PACS instances.

# 7. References


[1] L. A. B. Silva, “Medical imaging services supported on cloud,” 2011.

[2] O. S. Pianykh, *Digital Imaging and Communications in Medicine (DICOM): A Practical Introduction and Survival Guide*: Springer, 2008.

[3] I. N. d. Estatística. "Proportion of hospitals using computers (%) by Geographic localization (NUTS - 2002); Biennial 2012," February, 2013; http://www.ine.pt/xportal/xmain?xpid=INE&xpgid=ine_indicadores&indOcorrCod=0002289&contexto=bd&selTab=tab2.

[4] I. N. d. Estatística. "Proportion of hospitals connecting to the Internet (%) by Geographic localization (NUTS - 2002); Biennial," February, 2013; http://www.ine.pt/xportal/xmain?xpid=INE&xpgid=ine_indicadores&indOcorrCod=0002289&contexto=bd&selTab=tab2.

[5] C.-C. Teng *et al.*, "A medical image archive solution in the cloud." pp. 431-434.

[6] A. F. S. H. Article. "Prepare for Disasters & Tackle Terabytes When Evaluating Medical Image Archiving.," 2008; http://www.frost.com.

[7] C. Costa, A. Silva, and J. Oliveira, "Current Perspectives on PACS and a Cardiology Case Study," *Advanced Computational Intelligence Paradigms in Healthcare-2*, Studies in Computational Intelligence S. Vaidya, L. C. Jain and H. Yoshida, eds., pp. 79-108: Springer Berlin Heidelberg, 2007.

[8] H. Huang, *PACS and imaging informatics: basic principles and applications*: Wiley-Blackwell, 2011.

[9] L. B. Silva, C. Costa, and J. Oliveira, “DICOM relay over the cloud,” *International Journal of Computer Assisted Radiology and Surgery*, pp. 1-11, 2012/08/01, 2012.

[10] McGraw-Hill, and S. P. Parker, *McGraw-Hill Dictionary of Scientific & Technical Terms*, 6E ed.: The McGraw-Hill Companies, Inc., 2002.

[11] "O*sirix DICOM Viewer.*," 2013.

[12] A. Knopke. "*K-PACS*," 2013.

[13] S. S. Furuie *et al.*, "Archiving and retrieving long-term cineangiographic images in a PACS." pp. 435-438.

[14] H. K. Huang, “From PACS to Web-based ePR system with image distribution for enterprise-level filmless healthcare delivery,” *Radiological physics and technology,* vol. 4, no. 2, pp. 91-108, 2011.

[15] ACR-NEMA, "Digital Imaging and Communications in Medicine (DICOM)," National Electrical Manufacturers Association, 1993.

[16] A. C. R. Nema, "Digital imaging and communications in medicine (DICOM) Part 3: Information object definitions," Part, 2007.

[17] T. Godinho *et al.*, “Enhanced regional network for medical imaging repositories,” in CISTI, 2013.

[18] P. Lipton, P. Nagy, and G. Sevinc, “Leveraging Internet Technologies with DICOM WADO,” *Journal of Digital Imaging,* vol. 25, no. 5, pp. 646-652, 2012/10/01, 2012.

[19] G. V. Koutelakis, and D. K. Lymberopoulos, “WADA service: an extension of DICOM WADO service,” *Trans. Info. Tech. Biomed.,* vol. 13, no. 1, pp. 121-130, 2009.

[20] R. Appuswamy, D. C. van Moolenbroek, and A. S. Tanenbaum, “Cache, Cache Everywhere, Flushing All Hits Down The Sink: On Exclusivity in Multilevel, Hybrid Caches.”

[21] T. A. S. Foundation. "JCS - Java Caching System," January, 2013.

[22] G. Hamilton, and R. Cattell, “Jdbc: A java sql api,” *Sun Microsystems,* vol. 1, pp. 997, 1996.

[23] Microsoft. "ODBC-Open Database Connectivity Overview," february, 2013; http://support.microsoft.com/kb/110093.

[24] JBoss. "Infinispan," February, 2013; http://www.jboss.org/infinispan.

[25] Terracota. "Ehcache," February, 2013; http://ehcache.org/.

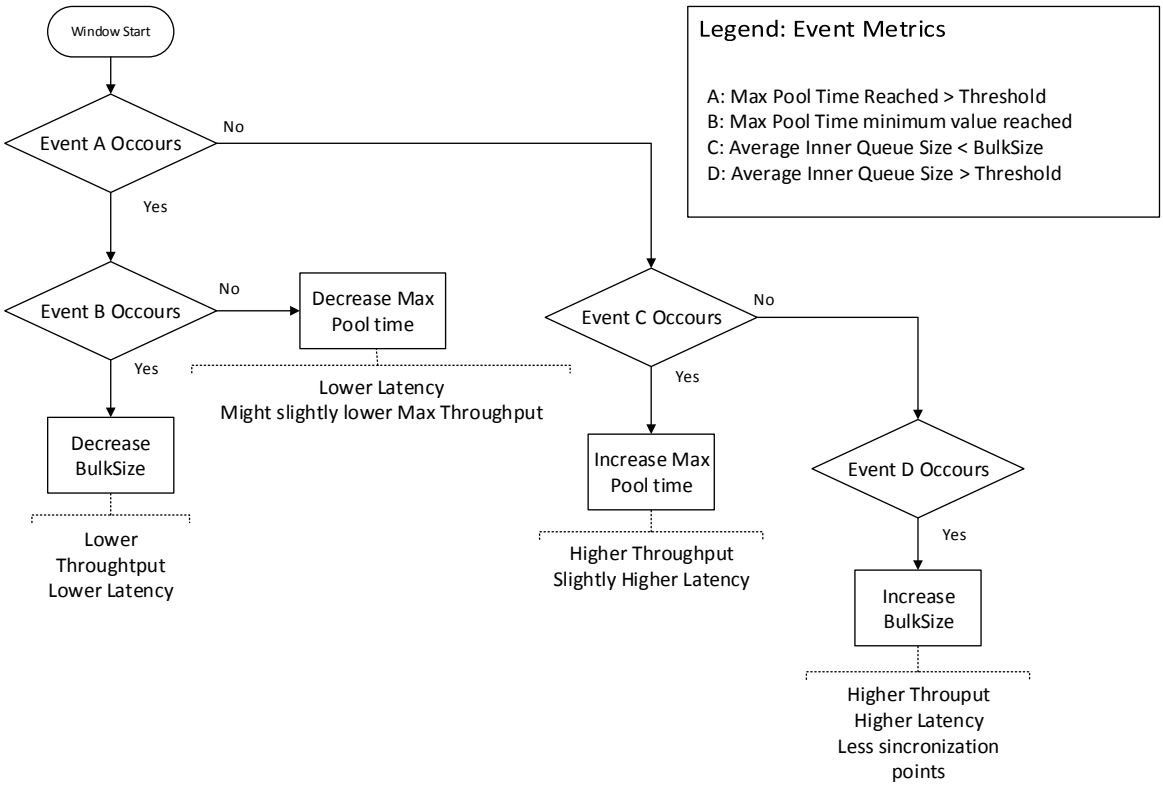
[26] J. Kotek. "MapDB," 2013.

[27] Y. Gu, and R. L. Grossman, “UDT: UDP-based data transfer for high-speed wide area networks,” *Computer Networks,* vol. 51, no. 7, pp. 1777-1799, 2007.
[28] I. m. group. "ICE: Internet Connectivity Establishment," February, 2013; http://www.internetsociety.org/articles/interactive-connectivity-establishment.
[29] H. Sinnreich, and A. B. Johnston, *Internet communications using SIP: Delivering VoIP and multimedia services with Session Initiation Protocol*: Wiley. com, 2006.
[30] R. Maani *et al.*, “A practical fast method for medical imaging transmission based on the DICOM protocol,” pp. 76280M-76280M, 2010.
[31] R. Maani, S. Camorlinga, and N. Arnason, “A Parallel Method to Improve Medical Image Transmission,” *Journal of Digital Imaging,* vol. 25, no. 1, pp. 101-109, 2012/02/01, 2012.
[32] L. B. Silva, C. Costa, and J. Oliveira, “A PACS archive architecture supported on cloud services,” *International Journal of Computer Assisted Radiology and Surgery,* vol. 7, no. 3, pp. 349-358, 2012/05/01, 2012.
[33] C. Costa *et al.*, “Design, development, exploitation and assessment of a Cardiology Web PACS,” *Computer Methods and Programs in Biomedicine,* vol. 93, no. 3, pp. 273-282, 2009.
[34] M. Benjamin, Y. Aradi, and R. Shreiber, “From shared data to sharing workflow: Merging PACS and teleradiology,” *European Journal of Radiology,* vol. 73, no. 1, pp. 3-9, 2010.
[35] Google. "Google Cloud Storage," 2013; https://cloud.google.com/products/cloud-storage.
[36] Amazon. "Amazon Simple Storage Service (Amazon S3)," 2013; http://aws.amazon.com/s3/.
[37] A. A. Bui *et al.*, “Problem-oriented prefetching for an integrated clinical imaging workstation,” *Journal of the American Medical Informatics Association,* vol. 8, no. 3, pp. 242-253, 2001.
[38] A. Descampe *et al.*, “Prefetching and caching strategies for remote and interactive browsing of JPEG2000 images,” *Image Processing, IEEE Transactions on,* vol. 16, no. 5, pp. 1339-1354, 2007.
[39] J. Gutiérrez-Martínez *et al.*, “A Software and Hardware Architecture for a High-Availability PACS,” *Journal of Digital Imaging,* vol. 25, no. 4, pp. 471-479, 2012/08/01, 2012.
[40] S. Langer, “Issues surrounding PACS archiving to external, third-party DICOM archives,” *Journal of Digital Imaging,* vol. 22, no. 1, pp. 48-52, 2009.
[41] K. J. Dreyer *et al.*, *PACS: A Guide to the Digital Revolution*: Springer, 2005.
[42] L. S. Ribeiro, C. Costa, and J. L. Oliveira, “Clustering of distinct PACS archives using a cooperative peer-to-peer network,” *Computer methods and programs in biomedicine,* vol. 108, no. 3, pp. 1002-1011, 2012.
[43] C. Costa *et al.*, “Dicoogle - an Open Source Peer-to-Peer PACS,” *Journal of Digital Imaging,* vol. 24, no. 5, pp. 848-856, 2011/10/01, 2011.
[44] P. J. Sadalage, and M. Fowler, *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*: Addison-Wesley Professional, 2012.
[45] T. A. S. Foundation. "Apache Lucene," June, 2013; http://lucene.apache.org/.
[46] J. Paterson, and S. Edlich, *The definitive guide to db4o*: Apress, 2006.
[47] Almworkds. "Sqlite4java - Minimalistic high-performance Java wrapper for SQLite," December, 2012; http://code.google.com/p/sqlite4java/.
[48] R. Biedert. "JSPF: Java Simple Plugin Framework," June, 2013; http://code.google.com/p/jspf/.
[49] M. Armbrust *et al.*, “A view of cloud computing,” *Commun. ACM,* vol. 53, no. 4, pp. 50-58, 2010.
[50] R. Zhang, and L. Liu, "Security models and requirements for healthcare application clouds." pp. 268-275.
[51] L. S. Ribeiro *et al.*, “XDS-I outsourcing proxy: ensuring confidentiality while preserving interoperability,” *IEEE Journal of Biomedical and Health Informatics*, 2013, to be published.
[52] "dcm4che2 DICOM Toolkit," February, 2013; http://www.dcm4che.org.
[53] D. Belson, T. Leighton, and B. Rinklin. "Akamai: State of the Internet Q4 2012," June, 2013; http://www.akamai.com/stateoftheinternet/.

# 8. Appendix

## 8.1. Controlled Channels automatic reconfiguration agent architecture

Based on the knowledge gathered about the modalities data profiles and the Controlled Channels technique, we have made a sketch about a possible future implementation for a reconfiguration module. The following diagram intends to aid possible further efforts by representing a software agent for this module. This agent is intended to select the best parameter to tune, in order to improve the performance of its controlled channel. It follows an hierarchical architecture very common in reactive agents. The agent is supposed to work in time windows with fixed size. For each time window, the agent calculates the world state which we think that is well modulated by four event metrics (shown in diagram). Based on the world state the appropriate behavior is selected effectively tuning the Controlled Channels parameters. This would be a very good starting point for further improvements to our distributed PACS.



## 8.2. Cache System API Functionality Table

The following table thoroughly states the functionality provided by our Cache System API. The methods are sorted by primary scope. We consider the primary scope of each method to be the module where it is majorly issued.

| Method | Description | Primary Scope |
|---|---|---|
| **Query** | **Queries the cache system about DICOM Objects with certain attributes. These attributes are specified in the query string. The query string is formatted according to the Lucene's term query format, as it is used very often across the IT field.**<br>**The response is formatted according to the C-Find response specifications of the DICOM Standard.** | **Meta-data management modules.** |
| **Query as String** | **Same as the above but it converts the responses to a Key-Value paradigm using a hash map.** | **Meta-data management modules.** |
| **Index DICOM Document** | **Indexes in the Cache engine the supplied DICOM Object. It can be supplied directly or by an input stream.** | **Meta-data management modules.** |
| **Index Transformed Document** | **Same as the above but indexes the DICOM Objects attributes provided by an Key-Value paradigm with strings.** | **Meta-data management module.** |
| **Retrieve Images for Study** | **Retrieves the Image SOP UIDs of the images enclosed in the specified study. The study is specified by its Study Instance UID.** | **Meta-data management module.** |
| **Register Study** | **Forces the registry of a new study in the cache system. It may be useful to cache study information without supplying an actual DICOM Object.** | **Meta-data management module.** |
| **Resolve Data for Study** | **Retrieves every Descriptor of cached objects (image or chunk) belonging to the desired study.** | **Meta-data management module.** |
| **Add Chunk** | **Caches the given chunk data.** | **Storage Management and Meta-data management modules.** |
| **Add Image** | **Caches the given image data.** | **Storage Management and Meta-data management modules.** |

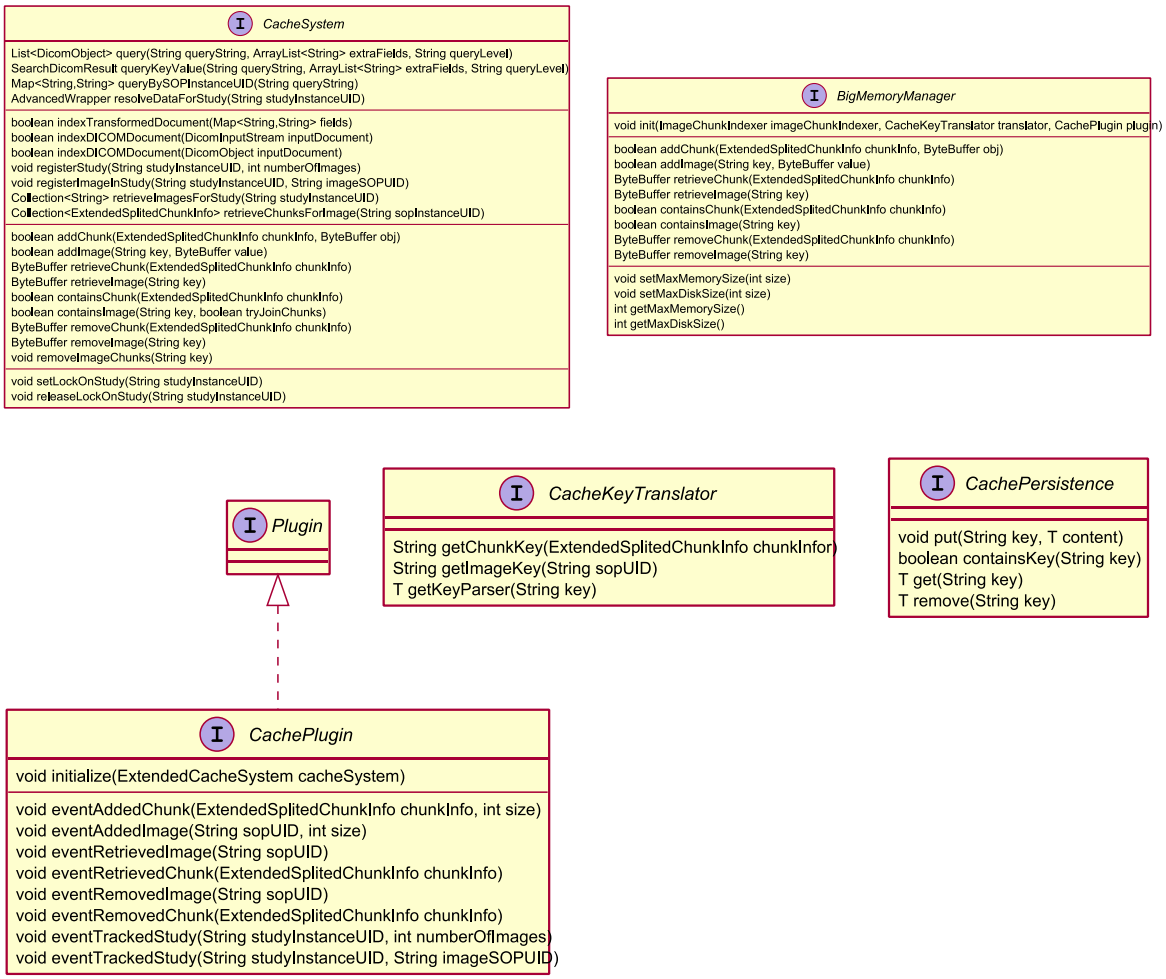
| Method | Description | Primary Scope |
|---|---|---|
| Register Image In Study | Forces the registration the specified Image SOP UID in the given study. | Meta-data management module. |
| Retrieve Chunk | Retrieves the chunk identified by the given Chunk Descriptor | Storage Management modules. |
| Retrieve Image | Retrieves the image identified by the given SOP Instance UID | Storage Management modules. |
| Retrieve Chunks From Image | Retrieves the descriptor of the cached chunks for given image. | Storage Management modules. |
| Contains Chunk | Checks if the given chunk is cached. | Storage Management and Meta-data management modules. |
| Contains Image | Checks if the given image is cached. | Storage Management and Meta-data management modules. |
| Remove Chunks | Evicts the specified chunk from the system. | Storage Management and Meta-data management modules. |
| Remove Image | Evicts the specified image if only it is completely cached. | Storage Management and Meta-data management modules. |
| Remove All Image Chunks | Evicts the specified image completely, even if the image is not complete. | Storage Management and Meta-data management modules. |
| Set Lock on Study | Sets a new lock on a given study. | Service Layer |
| Release Lock on Study | Releases the previous lock on the given study. | Service Layer |

## 8.3. Cache System Class Diagram

The following diagrams show the internal class structure of our proposed cache system architecture. They present a great starting point for developers who want to embed our system in their applications. The modular architecture is very well represented in the diagrams. Moreover, it is easily seen the role played by the abstract modules referenced in section 4.3.1. In every diagram pay a close attention to the class contractors and to the initialize method, as they are used to link correctly the different modules in the abstract implementations class.

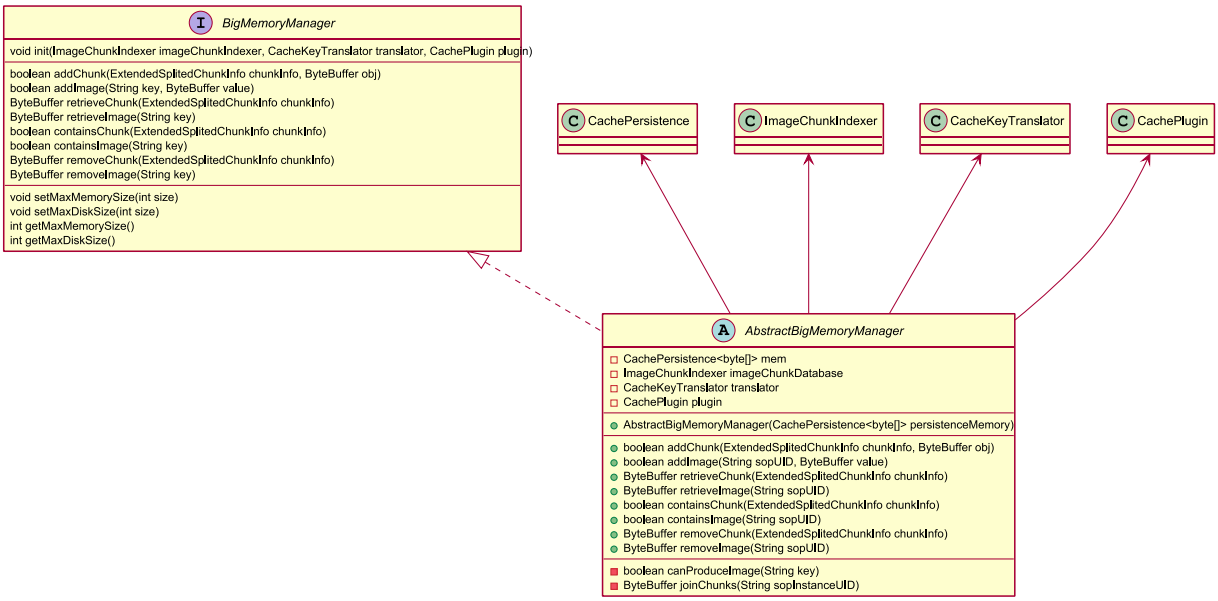
### 8.3.1. Module interfaces diagram

The following diagram represents the interfaces used to describe every module in our cache system architecture.



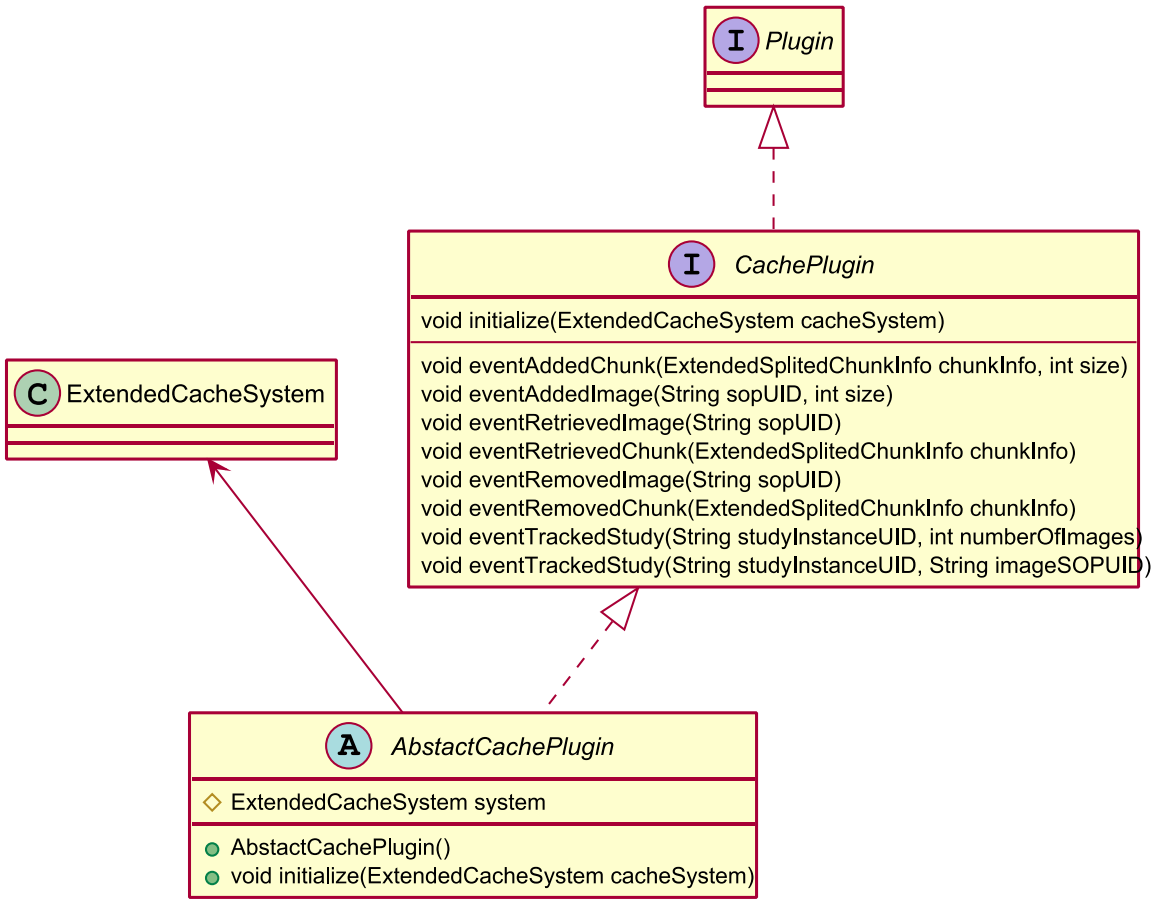
### 8.3.2.Storage management modules

This diagram presents the BigMemoryManager class that plays a key role in the Storage management modules.



### 8.3.3.Service Layer modules

The following diagram represents the AbstractCachePlugin class that plays a key role in the implementations of third party plugins.



### 8.3.4.Cache System class diagram

Lastly this diagram shows the AbstractCacheSystem that should always be used as a base for new systems implementations.