Development of a DICOM server for the reception and storage of medical images in digital format.

FRANCISCO J. NOVOA, ANTONIO F. CASTRO, JAVIER PEREIRA, ALEJANDRO PAZOS
IMEDIR Center
University of A Coruña
SPAIN

Abstract: - Recent progress in digital medical imaging has ineluctably created a need for efficient management, for a mechanism that is able to gather and store all the modalities of medical images, regardless of the device that generated them. The DICOM standard defines a protocol for the communication between such devices. This article presents a software server that communicates with DICOM compatible machines and stores their studies. It has been validated and tested in real environments.

Key-Words: - Communications protocol, DICOM Standard, Server, Digital Medical Image.

1 Introduction
The digital image is omnipresent in today’s sanitary context. Diagnoses are most often backed by various types (modalities) of images, such as the Computerised Tomography (TC) or the Magnetic Resonance (MR). However, at a certain point the manufacturers of the devices that create these types of images started to use proprietary formats, which substantially complicated the storage of the images and their transmission between devices from different sources. In 1983, the ACR (American College of Radiology) and the NEMA (National Electrical Manufacturers Association) founded the first committee that was to tackle this problem. After testing several versions, in 1998 it presented the DICOM standard (Digital Imaging Communications in Medicine) in its third version, which was periodically upgraded. The use of this protocol has contributed significantly to the development of systems that are able to store and manage a large amount of images, also known as PACS (Picture Archiving and Communications Systems), by interacting directly with the generating devices. The IMEDIR Research Center at the University of A Coruña has developed a server that uses the DICOM protocol to communicate with DICOM machines.

2 Objectives
Considering the need for a software that could interact with various DICOM 3.0 compatible machines, our main objective was the development of a server that allows the reception of digital medical images generated by different devices and their transmission to a place of storage, in this case a PACS.

This server had to implement two of the operations (services) that can take place between two DICOM devices: Storage, during which the DICOM client requests the server to store the generated images, and Verification, an operation that confirms the existence of a connection between the machines. Since both services are connection-oriented, each operation consists of three phases: establishment of the connection, transfer of DICOM messages, and disconnection.
During the connection establishment, the devices negotiate about the information exchange and subsequently proceed to the data transmission. This software was validated and tested in several real environments with DICOM devices: the acquisition and storage tests took place at the Complejo Hospitalario Universitario Juan Canalejo, the Instituto Médico Quirúrgico San Rafael, both located in A Coruña city, and the Fundación Pública Hospital Virxe da Xunqueira in Cee (A Coruña province).

3 Material and Methods.
The DICOM server was developed according to the standard 3.0 version (2003 revision), and implemented with Java, because programmes codified in Java are portable, i.e. independent from the platform on which they are executed. It is also an object-oriented language that provides an extensive
Application Programmer’s Interface, with classes dedicated to the programming of network applications.

3.1 The DICOM standard
The DICOM standard defines a communication protocol (DICOM protocol) and a series of services (DICOM services classes) that provide various functionalities. It defines the TCP/IP protocol stack, and more concretely TCP, as transmission layer protocol.
DICOM is represented by an Entity-Relation diagram, also known as Information Model, that specifies the relation between the DICOM objects and entities such as Studies, Series, Images, etc, from the real world. The DICOM objects are defined by an IOD identifier (Information Object Definition) that can be either normalised or composed, depending on whether it represents one or several entities. The objects are associated to the services they can provide and as such form the Service Object Pair or SOP, which is divided into classes, each object being an instance of a class.
The DICOM standard defines the Application Entity (AE), a DICOM application that uses the standard to interact with other applications through a data network. The AE consists of a Service Class Provider (SCP), i.e. the server; and the Service Class User, (SCU) i.e. the client [1].
The standard also defines the Conformance Statements, which state the services and objects that are implemented by a device, and the Presentation Context, which is defined for each association in the course of the negotiation. It consists of the Transfer Syntax, that defines the codification of the basic elements, and the Abstract Syntax, and it allows us to negotiate what will happen during the transmission and how it will happen.

3.1.1 The DICOM Association
The DICOM Association defines the connection that is established between two AEs to exchange information. The associations consist of three phases: establishment or negotiation, execution of the required service(s), and finalisation.
The first phase is used to determine which services will be used during the association and in what manner. The SCU provides a list that specifies all the SOP classes with their proper transfer syntaxes; the SCP responds by marking the classes it accepts; the required services take place, and the SCU breaks off the connection. In case of errors, the SCP can abort the connection.

3.1.2 The DICOM messages.
All the information circulates over the network in the shape of messages. As we can see in Figure 2, the messages consist of a Command Set and an optional Data Set. The command set consists of the Command Elements that each have three fields: Tag, Length and Value.

3.1.3 DICOM Services.
In part 4 of the DICOM standard [2] we can observe the complete list of all the services that can be used by a DICOM application. The developed server supports two basic services, whose objective is the exchange of information: verification and storage, classified by Oosterwijk [3] as first dimension services.

Another implementation concerns the “Upper Layer Services”, which allow the establishment and interruption of the connection between the AEs. The services are: A-Associate, to carry out the
association; A-Release, to release the connection; A-Abort, to interrupt the connection in case of error; and P-Data, for the information transmission between the AEs by means of messages.

3.2 The Java programming language.
The most significant characteristic of Java is that the bytecode that is generated after the source files compilation can be interpreted by any Java machine. This allows the applications to be executed in any hardware platform or in any operative system. Also, Java is an object-oriented programming language with three fundamental characteristics: encapsulation, inheritance and polymorphism. It eliminates the pointer arithmetics and the memory management, which take place automatically, and as such simplifies the programming and offers more robust software. Another important element of Java is that it provides multithread programming, which is fundamental for the server.

3.3 Design of the application.
The application was modeled with the Unified Modeling Language (UML) [4]. The following paragraphs describe the contents of the Class Diagrams and the Sequence Diagrams, created during the design phase in order to illustrate the functioning of the application.

3.3.1 Description of the application’s business logic classes.
We briefly describe the classes that were used for the application and their role; the classes that are only linked to the graphic interface were omitted for a better understanding of the servers’ functioning.
Main Class: Instantiates an object of the Timer class, another of the ServerSocket class for the creation of multithread associations, and another of the Log class for the storing of the server messages in a file.
Log Class: Stores the server messages in a text file.
ServerThread Class: An object of this class will be generated by each association the server tries to realize. It is linked to a socket through which all the communications between the DICOM and the server take place.
Operation Class: Superclass of the DICOM operations.
Associate Request Class: In charge of processing the association requests and creating the answer.
P-Data Class: This class processes the petitions of the implemented services.
Command Class: Superclass of the different DICOM commands implemented in the server.

Echo Class: Implementation of the C-Echo command, which is necessary for the verification service.
Store Class: Implementation of the C-Store command, which is necessary for the storage service.
ErrorReading Class: Manages the exceptions and errors that may be produced during the message processing.
Tag Class: Data dictionary.

3.3.2 Description of the application sequence.
After defining the classes that are part of the Server, we now describe how the objects interact to provide the functionality on the basis of the Sequence Diagrams. The most important actions are the following: initialisation of the server, establishment of an association with a SCU, performance of a service, and finalisation of the association.
Initialisation of the server: It is initiated when the programme, whose Main method is in the Main Class, is executed. The Main method creates the configuration window and an object of the Timer class; then it solicits an instance of the Log object. The method creates a new object of the ServerSocket class to observe the arrival of DICOM messages in a determined port (TCP 104). It then requests the Start of the Timer object, which monitors periodic observations of the association requests from the DICOM devices.

![Figure 3. Verification Sequence Diagram.](image)
**Association:** When a message transmits an association request from a DICOM device, a new execution thread is generated, the ServerSocket creates a ServerThread and calls its start method. This class is in charge of all the tasks required by the association. In this way, the thread checks which type of message has arrived: if the message is an association request, it creates an AssociateRequest object to process the request. Once it is correctly processed, the ServerThread returns the response message that is needed to make the association (A-Associate-AC). The next step is to send the response to the DICOM device that originated the association request.

**Verification service:** Once the association has been made, the DICOM device requests a service. Figure 3 shows what happens when an association verification is requested. The server analyzes the message from the device and concludes that it is a P-Data, upon which it creates an instance of the class that deals with these messages (P_Data) and the server calls its processing method. Since the message is recognized as a verification request command (C-Echo-RQ), the server creates an instance of the class that is in charge of its message processing (Echo). After processing the message, the response is generated (C-Echo-RSP) and returned. As before, this response is sent to the source DICOM device.

**Storage service:** If the message is a storage request, the sequence is similar, except that the Store class is in charge of processing it.

**Finalisation of the association with a device:** As in the previous cases, an instance of the class that processes this type of message is generated (ReleaseRequest), the response generated by the server is returned and the association is interrupted.

4 **Results.**

The server was tested with two types of tests: the validation tests, that check the compatibility with DICOM and the correct functioning, and real environment tests to check all the requested characteristics.

Figure 4. Validation of the storage service.

**4.1 Validation tests.**

The first tests were carried out with the DICOM Central Test Node validation tools[5], which were designed in 1994 to be used in the annual RSNA congresses and to foment cooperative demonstrations. They were developed by the Mallinckrodt Institute of Radiology [6] of the Mallinckrodt Institute of Radiology and are at present in their version 3.0.6

We started by testing the storage service, which implied testing the association and disconnection between AEs. The server was started in the same machine and listening in port 104. The CTN, which acts as SCU, requests an association. The server responds by accepting the presentation context (storage of ultrasounds and Implicit Little Endian transfer). After establishing the association, the file is transferred in small packages (less than 17 Kbytes). When the transfer is completed, the CTN analyses the response of the Server and checks whether everything has gone well, as can be seen in Figure 4.
The next test was carried out against the verification service (Figure 5). At first we did not initiate the server, which generated an error (first ellipse). After starting up the server in port 104, we observed the validity of the verification service (second ellipse).

### 4.2 Tests in a real environment.

We carried out tests in various real environments in collaboration with the Hospital San Rafael [7] and the Complejo Hospitalario Universitario Juan Canalejo [8], in the city of A Coruña, and the Fundación Pública Hospitalaria “Virxe da Xunqueira”[9] in Cee (province of A Coruña). During the first phase we checked whether the services we pretend to provide function correctly with real devices: a helicoidal Hispeed TAC NX/i S 2247010 from General Electric Medical Systems at the Hospital San Rafael, an Acuson Sequoia echocardiograph from Siemens Medical Solutions at the Hospital Juan Canalejo, and a Sytec Sinergy TAC from General Electric Medical Systems at the FPH Virxe da Xunqueira; each device provided a positive result.

Also, since the three hospitals have similar network infrastructure characteristics (switched FastEthernet 100-Base-TX), and considering the supply of images of a determined modality that can be found in an average-sized hospital (according to a study by Torres in 2000 [10]), we create a charge test with compatible DICOM studies to carry out performance tests of the associations between AEs. We must remember that the amount of information that is transmitted through the links is, especially in certain cases, significantly larger than that of the original DICOM file, because the DICOM messages carry control information.

Table 1 reflects the average of the values of the three hospitals: it shows the size of the stored file, the amount of information transmitted by the stored file, the average transfer rate and the use rate of the link. All these characteristics allow us to define the efficiency of the data transmission.

### 5 Conclusions.

The advances in the field of medical technology have generated new formats in digital medical imaging and as such have led to the appearance of new problems. One of those problems, the transmission and storage of the digital images, was eliminated with the appearance of the DICOM standard and the PACS. The development of the server, one of the main components of the PACS, was the main purpose of this paper. The developed server is provided with the basic services that are required for its implantation.

We have created a modular design that allows future developments and that offers services to more than one DICOM device at the same time, complying, naturally, with the DICOM 3.0 standard.

<table>
<thead>
<tr>
<th>Switched FastEthernet Network</th>
<th>Real Size (MBytes)</th>
<th>Data TX (Mb)</th>
<th>Time. (mm:ss)</th>
<th>Rate. (Kbps)</th>
<th>Bandwidth Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1- CT complete cranium.</td>
<td>82,93</td>
<td>85,63</td>
<td>01:11</td>
<td>9,879,62</td>
<td>14,50%</td>
</tr>
<tr>
<td>2- CT neck</td>
<td>45,20</td>
<td>46,65</td>
<td>00:33</td>
<td>11,580,22</td>
<td>16,23%</td>
</tr>
<tr>
<td>3- CT head A</td>
<td>12,03</td>
<td>12,42</td>
<td>00:10</td>
<td>10,178,18</td>
<td>16,45%</td>
</tr>
<tr>
<td>4- CT head B</td>
<td>31,08</td>
<td>32,10</td>
<td>00:10</td>
<td>10,113,14</td>
<td>13,72%</td>
</tr>
<tr>
<td>5- CT thorax</td>
<td>20,06</td>
<td>20,71</td>
<td>00:16</td>
<td>10,602,35</td>
<td>13,39%</td>
</tr>
<tr>
<td>6- NMR cranium</td>
<td>2,10</td>
<td>2,36</td>
<td>00:01</td>
<td>16,082,01</td>
<td>18,50%</td>
</tr>
<tr>
<td>7a- Hemodynamic A (comprimido) *</td>
<td>1,26</td>
<td>10,79</td>
<td>00:01</td>
<td>46,778,46</td>
<td>62,60%</td>
</tr>
<tr>
<td>7b- Hemodynamic A NoMultiframe (C)</td>
<td>1,26</td>
<td>11,91</td>
<td>00:07</td>
<td>13,938,53</td>
<td>23,03%</td>
</tr>
<tr>
<td>8a- Hemodynamic B *</td>
<td>14,76</td>
<td>14,80</td>
<td>00:03</td>
<td>40,417,15</td>
<td>64,10%</td>
</tr>
<tr>
<td>8b- Hemodynamic B NoMultiframe</td>
<td>14,76</td>
<td>16,52</td>
<td>00:12</td>
<td>11,274,53</td>
<td>18,00%</td>
</tr>
<tr>
<td>9a- Echocardiography *</td>
<td>10,13</td>
<td>10,16</td>
<td>00:02</td>
<td>41,602,85</td>
<td>57,05%</td>
</tr>
<tr>
<td>9b- Echocardiography NoMultiframe</td>
<td>10,13</td>
<td>10,55</td>
<td>00:05</td>
<td>17,285,25</td>
<td>25,68%</td>
</tr>
</tbody>
</table>

Table 1. Transfer measurement results between the SCU and the SCP.
The server is provided with a log file that stores the events from the start and is structured in such a way that it can easily be read or imported. Finally, we used free and inexpensive tools for the development of this project, which is integrated into the major project of developing a DICOM PACS [11].

The modularity that results from the server development allows the future support of more DICOM services: consultation/recovery, storage compromise, and printing management. The implementation of these services would enable the server to carry out nearly all the important tasks of the commercial PACS’ servers, and provide various types of hospitals with a complete service.

Another important step will be to add the possibility to configure the server over the web, providing the server manager with more mobility and speed. We should also consider the possibility to add more configuration elements, apart from the already existing ones (e.g. the maximum size of the received package); or we could add a database with all the possible devices that can be connected to this server, and as such improve its speed and efficiency.

Where the security is concerned [12], the codification of the transmissions would allow the development of an assistential telemedicine system. A medical expert could then consult images received from DICOM devices in remote hospitals.

Finally, the server should also comply with the other standards that are being implemented by the hospitals. An interesting example would be the IHE standard (Integrating the Healthcare Enterprise)[13], which applies to the hospital in general rather than to datatransmission alone.

References:

Acknowledgements
This work was supported in part by Spanish Department of Science and Technology (MCYT) TIC 2000-0120-P4-03 & TIC2003-07953 and "Xunta de Galicia" (PGIDT 03PXIC10504PN & PGIDT 04PXIC10503PN)