A Distributed Heterogeneous Image Server

Stefano Loddo, Gavin Brelstaff, Gianluigi Zanetti

BioMedical Applications, CRS4, Cagliari, I-09123 Italy

Digital image transmission is now ubiquitous across computer networks and thus there is increasing pressure to allow access to medical image data at sites remote from PACS locations. In fact, it may soon make economic sense to outsource medical image services - to dedicated service providers at geographical locations outside of the traditional radiology department or HIS's. The technical challenge faced by system developers is to produce client-Viewer/server-PACS configurations that can realistically span the network. In particular, the systems must provide the performance usually expected by client medics and it must be flexible to the needs of the service providers.

At CRS4 - BioMedical Applications - we are integrating various technologies derived from the www-intranet field, and object-oriented middleware to prototype technological solutions that address both the issues of performance and flexibility. These are discussed in turn below; then we provide an overview of our system

Performance issues

Medics accustomed to viewing images on their local work station expect those accessed over the network to be of equivalent quality - which can mandate large volume 16 bit image data being losslessly transmitted from server to client, on demand. Clearly this will generally require high bandwidth telecommunication networks otherwise the response times expected by the medics will not be met. Some degree of remote controlled cropping and sampling of the images at the server before transmission may ameliorate the situation. Futhermore, the overhead of any network protocol must not be allowed to slow down pixel tranfer. So called blob streaming [1] between TCP/IP sockets may be the preferred pixel transport mechanism.

Flexibility issues

Providers need to have flexible control to improve the operation of their services - especially in a price-competitive environment. They should be able to incorporate a new PACS into their system without undue technical development. We are addressing this issue through adopting the DICOM 3.0 standard [2,3] -we are encapsulating a CTN PACS [4,5] in our prototype service. Providers should also be able to deploy the latest versions of their software at minimal cost. Java applet software can address the client-side deployment issue by providing a sophisticated image viewer that can be downloaded to a wide variety of computer platforms. Providers should also have the freedom to shuffle the data around their storage sites to optimise performance. This should be achieved by a Name Service that avoids the need to explicitly tell clients about network address changes.

Overview

CORBA [6] is the natural choice for system integration between our proposed image client written in Java and the CTN image server written in C/C++. Its interface definition language (IDL) permits true distributed computing based on objects defined in heterogeneous object-oriented programming languages. Adopting an object-oriented approach can reduce the complexity of the programming task and thus promote the realistic low cost reuse of software in follow-on applications. CORBA is the open standard that will effectively allow us to write our client and server software as a unified system and not have to manage two separate entities. In comparision with web-based HTTP/CGI network protocol it can provide a much richer, more sophisticated, more responsive level of communication [7].

Dynamic network addressing requires a dedicated Name Service so that:

- an image acquisition may be named before it gets sent from an exam site to the service provider (e.g. to facilitate DICOM moves).
- the service provider can achieve reasonable operational load balancing by moving data to other disks, network nodes, or storage media.

The basic operation of a Name Server is shown in Fig 1. Although CORBA has specified its own Naming Service [8], we have adopted a simpler, easier to manage approach that involves registering of the name of available imagery within an ORACLE database table through a Registry interface that might run on a separate network host.

As shown the Name Service mediates requests between *Image Client* and *Image Server* software:

1. The *Owner* within the Image Server registers the `name` of an image as being located at a given `address`.
2. The *Registry* then stores the pair (`name`, `address`) in the database table.
3. Next, the Image Client contacts the *ImageBoxBroker* within the *Name Server* which then can retrieve the address of the ImageBoxBroker within the ImageServer.
4. This address is then used to bounce the client’s request to the said ImageBoxBroker, which constructs an appropriate *ImageBox* object by accessing the PACS.
5. This ImageBox is then available to provide details of the image to the client - such as its dimensions, bits per pixel etc.
6. The Image Server is then requested to send the image data (pixels) to the Image Client.
7. Finally, the Image Client then receives and views the image.

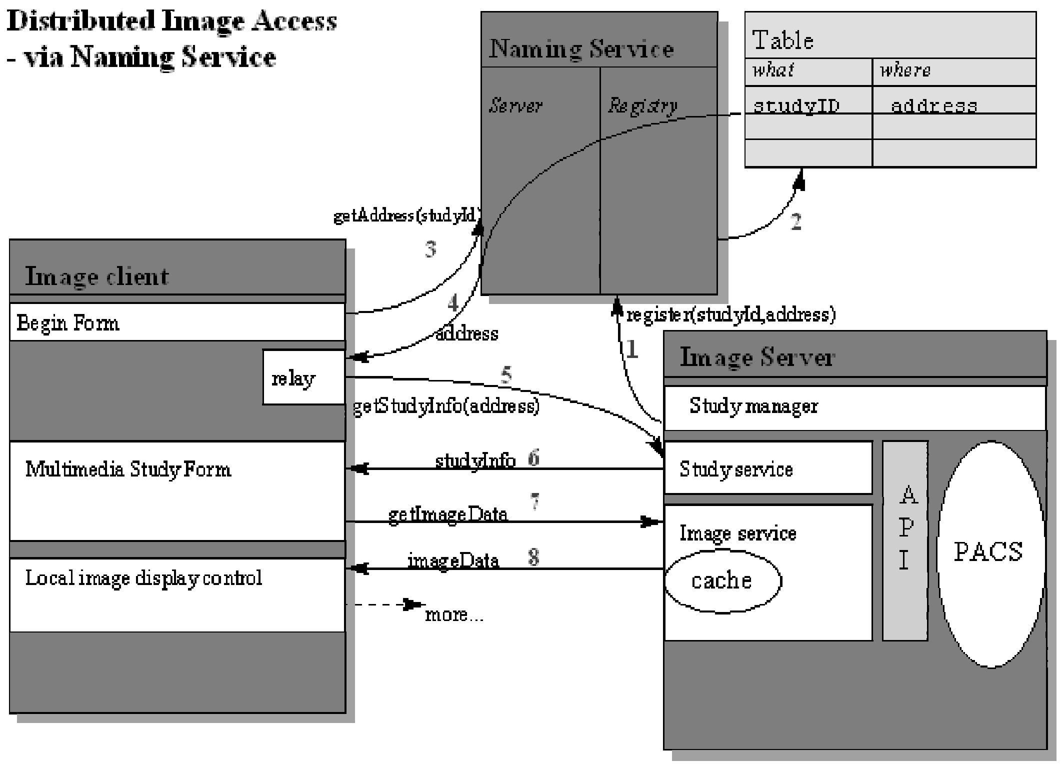

Figure 1 ***Name Service diagram showing the sequence of events.***

The details of how the pixels are transmitted to the client should depend on the performance criteria discussed above. Our early prototypes are using CORBA IDL to specify the transmission interface. If this proves to have too greater overheads then stream sockets may be also tested. In any case, some sort of encryption and compression is expected to be deployed in an operational imaging service. Although significant compression can be provided by JPEG/JIF - and we are also providing for its transmission - clearly the inevitable data loss embodied in JPEG is not really acceptable for Medical Diagnostic usage.

State of the Work

At this time we have completed the IDL design and the C++ implementation of the ImageServer and the NameService systems. Implementation of the low level parts covers only some DICOM modalities but the design makes extension to other formats easy to do. Pixel's data are, for the moment, transmitted as CORBA IDL parameters while the more performant blob streaming solution is a planned extension.

For the development of the server's components we have worked on an *IBM RS6000 OS AIX 4.1* platform using the native xlC C++ compiler. For the persistent data we haveused the Oracle Server 7.3.2 database, while the CORBA ORB daemon is the Orbix daemon v2.01 for AIX 4.1 C++.

On the client's side the platform chosen is the SUN JDK 1.02, while we have used the OrbixWeb IDL compiler v2.01 for the IDL to Java mapping.

References

[1] Pyarali I., Harrison T.H., & Schmidt D.C. *Design and Performance of an Object-Oriented Framework for High-Speed Electronic Medical Imaging*, USENIX, Vol 9, No. 3, 1996.

[2] ACR/NEMA, The Digital Image Communications in Medicine (DICOM 3.0) Standard, 13 vols, No. 905013, NEMA Washington D.C, 1994.

[3] Revet, B. *DICOM Cook Book for Implementations in Modalities*, chap 1 &2, version 1.1, Architecture Re-Use Communications, Philip Medical Systems Holland 1997.

[4] Moore S.M., Hoffman S.A., Beecher D.E., *DICOM Shareware: A Public Implementation of the DICOM Standard*, Medical Imaging 1994-PACS: Design & Evaluation, Gilbert Jost R. (ed) Proc SPIE 2165 pp. 722-781, 1994.

[5] Giachetti, A & Petri C. *Medical Image transmission on heterogeneous networks*. EUROPACS 97 (this volume) 1997.

[6] OMG, *CORBA: Architecture and Specifications*, 1996

[7] Orfali R., & Harkey, D. *Client/Server Programming with JAVA and CORBA*, John Wiley: New York, 1997.

[8] OMG, *CORBAservices*, COSS1 1993.