Evaluation of Two- and Three-Tier Database Connections for a Java Based Medical Image Viewer

F. Unglauben, W. Hillen, M. Murdfield

Medizinische Informatik
FH Aachen Abt. Jülich, Biomedizinische Technik
Ginsterweg 1, 52428 Jülich
Email: unglauben@fh-aachen.de

Abstract. Different methods to connect a Java-Applet for Medical Image Visualisation to databases are presented. Using Java Database Connectivity (JDBC) two-tier and three-tier applications have been developed. The middle-tier is built with Java-Applications and Java-Servlets connecting to the Applet using Remote Method Invocation (RMI) and the Hypertext Transfer Protocol (HTTP). The evaluation aims at a Java based visualisation and processing of medical images in a distributed environment with a direct connection of the Java-Applet to existing PACS.

Keywords: PACS, Teleradiology, Distributed Systems, DICOM, Image Information Systems, Java, Servlets, JDBC

1 Introduction

For archiving and retrieval of medical image data nowadays picture archiving and communication systems (PACS) are generally installed in large hospitals. In this environment diagnosis stations that are suitable especially for primary diagnosis and databases that store image and diagnosis data are connected through a Local Area Network (LAN). Picture creating devices as well as the PACS use the DICOM (Digital Imaging and Communications in Medicine) standard as their image format and standard interface. Viewing stations are spread all over the hospital to retrieve and view stored images . They allow a quick access to image and diagnosis data.

The Java technology allows to build viewing stations with no special software installed beside an operating system and a network browser. In this concept Java-Applets are transferred together with image data and diagnosis information from a web-server through the network to the client (viewing stations). The network browser that is installed on the client starts the Applet in its Java Virtual Machine (JVM). The connection of this client software to PACS is the target of the present evaluation. Possibilities to connect the clients (Java-Applets) to databases in the PACS as well as the transfer of image data from the PACS to the clients are presented.

2 PACS with distributed software

The basic concept of a PACS with an added web-server for distributed image visualisation is shown in figure 1. Medical images are transferred to the PACS where servers for storage and archival are connected to diagnosis stations. A web-server is connected to the PACS. This web-server transfers Java-Applets for medical image visualisation together with image and diagnosis data through Intra- or Internet to the clients that are located inside or outside the hospital. In the network browser that is installed on the clients the Applet is started and allows to view image and diagnosis data.

A Java-Applet for viewing and processing medical images in a network browser was presented by Hillen et al [1].

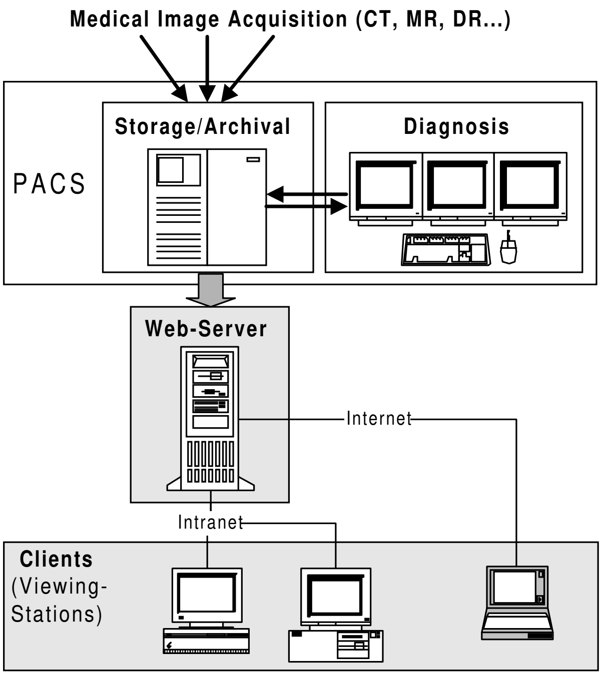

Fig. 1: Basic concept of a PACS with connected web-server for distributed medical image visualisation on clients.

3 Connecting Java Applets to PACS

The system concept shown in figure 1 has been simulated in a laboratory set-up. An Oracle database has been structured according to the DICOM standard [3]. This database keeps information about the patients' studies and the location of medical images produced during these studies. As medical images can be located on multiple storing devices the servername and path rather than the image itself is stored in the database.

Java's standard extension Java Database Connectivity (JDBC) has been used to analyse different ways of applications with the general aim to realise an application that is written in 100% Java and therefore is completely platform- and system-independent [2].

Realised database applications

Figure 2 shows two general concepts of the developed database applications. The upper part of figure 2 shows a two-tier application with a Java-Applet on the client and the database installed on the web-server. Image and diagnosis data are located on this web-server. The lower part (figure 2) shows several three-tier applications with Java-Applets on the clients, Java Servlets or Java Applications on the web-server (middle-tier) and the database and file system on different servers. Each application uses a JDBC-Driver of Type IV for database connection. This driver is written in 100 % Java and can be loaded from the web-server to the Java-Applets on the client side.

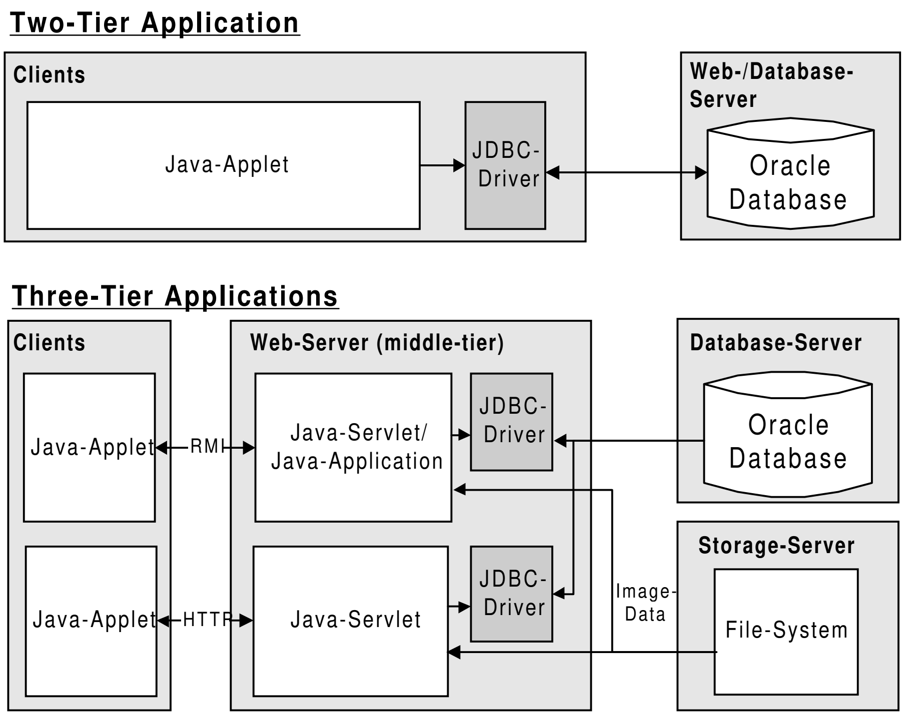

Fig. 2: Basic structure/concept of a two-tier application consisting of clients and a web-server in comparison to three-tier applications with a separate web-server in the middle-tier.

The middle-tier is realised by using Java- Servlets and Java-Applications. Java Servlets can be started interactively from clients and run in the virtual machine of the web server whereas an application has to be started directly on the server. Network communication with the database server was realised in all cases by the Oracle-JDBC-Driver. The network protocol between client (Java-Applet) and middle-tier (Java-Servlet/Java-Application) is used to send commands from the Applet to the Servlet and to transfer database query results and image data from the Servlet to the Applet. It was varied by using Java's Remote Method Invocation (RMI) as well as the Hypertext Transfer Protocol (HTTP). Both protocols can be used in Internets because they are accepted by firewalls and proxy servers. In three-tier applications image data are transferred from an additional storage server. Image data are transferred using I/O-Streams on different protocols (RMI and HTTP). Java offers the possibility to transfer the data-streams compressed with Lempel-Ziv-Algorithm. Furthermore in the three-tier application image data are copied to a temporary file on the web-server. This file is overwritten as soon as another image is loaded and transferred to the Java-Applet using the FileInputStream.

To keep several connections simultaneously open to the database a hashtable with connection objects has been created. This so called Connection Pool manages and controls multiple connections to databases. Any Applet that connects to the middle-tier gets its own connection. This connection is checked whether it is free and valid.

The Java-Applet for medical image visualisation that is loaded to the client has interfaces to act either in a two-tier or in a three-tier environment. These interfaces are each realised in one Java-class. Only this class has to be changed if the database structure changes. This facilitates the connection to any type and any structure of database in a PACS environment.

4 Results

The time it takes to open a connection and to query the database in two-tier applications is determined by the processor power of the client. Connection and transfer on a Pentium II 266 take less than half of the time they take on a Pentium 133. Looking at three-tier models, only the performance of the web-server influences all database interactions. The client's Applet only gives commands that are executed by the web server's Servlet/Application. However all results of database queries have to be transferred from web server to client. This transfer is dependent on the performance of both parts (client and web server) as well as on the used network protocol. HTTP is twice as fast as RMI in transferring complex data.

Using Applets in a two-tier model means that databases as well as image data have to be located on the web-server the Applet is loaded from. This is due to Java's security restrictions. With a three-tier model Java-Servlets and -Applications are allowed to contact any server in an Intranet. Therefore databases and image data can be distributed on different machines. The transfer of image data is strongly dependent on the network protocol. Again HTTP has shown better performance. However transferring image data through an input-/output-stream leads to memory overflows with large images (>2.2 MBytes). Compressing the streams with Lempel-Ziv-

Algorithm takes a lot of time and only moves the overflow problem to another borderline (>3.5Mbytes). Copying the image data in a temporary file and loading this data directly to the Applet using a FileInputStream has shown the best performance and no memory overflow even with large images (tested up to 12Mbytes in MS Internet Explorer).

Java-Applications have to be started on the middle-tier (web-server). When using RMI the "RMIRegistry" has to be started as well. Servlets can be started interactively by accessing their URL. This can be done in a browser or by the Java-Applet itself. The "RMIRegistry" in this case is done in the Servlet's superclass to avoid multiple inheritance as any Applet has to inherit from HTTPServlet.

Connections can be opened at one time and kept in a hashtable. The Servlet manages the number of connections and makes sure that each connections remains valid. In the test environment only a small advantage in performance was noticed.

5 Conclusions

The concept of using Java-Applets on clients offers the possibility to distribute medical images stored in a PACS through networks via a web-server as shown in [1]. One class can build the interface to databases and can be changed if databases' structure changes.

Two-tier database applications as realised in this project are only useful in small environments. The web-server must contain the datbase as well as all the image data as Applets are only allowed to contact the server they are loaded from. In real PACS environments where image and diagnosis data is distributed on several servers the use of Java Applets can only be realised by using three-tier applications with a web-server in the middle-tier. This gives the possibility to query databases on any server and to retrieve image data from any storage device in the network. Java's Remote Method Invocation (RMI) is a useful and secure mechanism to transfer data through network but compared to HTTP it has less performance.

Connection Pools are able to manage multiple connections to databases. Their function and advantages have to be further tested in real environments.

6 References

1. Hillen W., Jansen N., Unglauben F., Indefrey R.: Multimediale Darstellung und Verarbeitung medizinischer Bilddaten in Rechnernetzen. Bildverarbeitung für die Medizin 1998: 373-377. Springer-Verlag, Berlin, 1998:
2. Unglauben F.: Implementation of an Oracle database with a Java interface for medical image visualisation, Master Thesis, Coventry University, August 1999
3. Murdfield M.: Anlegen und Verwalten einer Oracle Datenbank für die medizinische Bilddarstellung unter Java, Diplomarbeit, FH Aachen Abt. Jülich 1999

This project was supported by the "Innovationsprogramm Forschung" of the "Ministerium für Schule und Weiterbildung Wissenschaft und Forschung" (NRW-Germany).