

MIFAR

Infrastructure for Managing Medical Imaging Research Data

Our laboratory has faced a growing need to standardize methods of storing medical image data along with other associated post-image processing files, physiologic records, lab notes, etc. The need for storing the information in a way to allow for data sharing amongst investigators is compounded by the increasing numbers of collaborators interacting with our laboratory from other institutions. This document will outline the system which is emerging to address our data archiving and retrieval needs. We have named the system MIFAR (Medical Image File Archive and Retrieval), and the system is currently in the pre-alpha development phase. Though we can demonstrate some of basic operations a full featured prototype will not emerge until mid summer of 2002. We currently have set down some principles by which our work on the MIFAR system is carried out and welcome comment. A first principle is that the system will be made available as open source software and will make use of, wherever possible, other open source software.

Design Principles

Data file format flexibility

Out of necessity researchers are constantly making up new file formats to store the results of their work. For image files themselves standard formats abound (DICOM, JPEG, PICT, MPEG, etc.). However analysis results often have to have their own format because nothing suitable already exists. Thus we cannot tie our system to any particular file type. We should provide extra support for established types by, for example, providing DICOM parsers that populate database fields but any file should be storable.

User permissions are enforced by the backend database.

This allows us to open the database up to arbitrary queries from the end user. It also allows for a wider range of client applications. In many research oriented online databases, permissions are checked at the web server. This means that only web browsers can directly interact with data stores.

Well documented server interface protocol

By documenting the server interface multiple developers can assist in writing clients for the MIFAR system. Currently we are working on a web site which is a client of the database and it's file storage tools. This is the most desirable client type since almost everyone has ready access to a browser. Another not so obvious client would be a DICOM transceiver. This client would take in DICOM streams and then upload them to the database or conversely turn a DICOM query into a MIFAR query. In the future it may be desirable to adapt the open source CTN (Central Test Node) or DCMTK (DICOM Tool Kit) software for this purpose.

Table layout flexibility

Different research trials require different data to be tracked. A study on lung cancer would need to track nodules (discrete and independent ROI's) while an emphysema study would need to track hole formation (where all ROI's are considered a single distributed entity.) However, both of these will have many common infrastructure needs. In MIFAR, data tables and server side functions come in two categories:

1. Framework level tables, procedures and programs that all studies may use.
2. Project level tables customized to each study.

The framework tables are used for tracking users, files, transactions, studies and patients. The project dependent tables are setup based on the specific needs of the project.

Image files are not stored in the database itself.

Storing files directly in the database tables creates three problems.

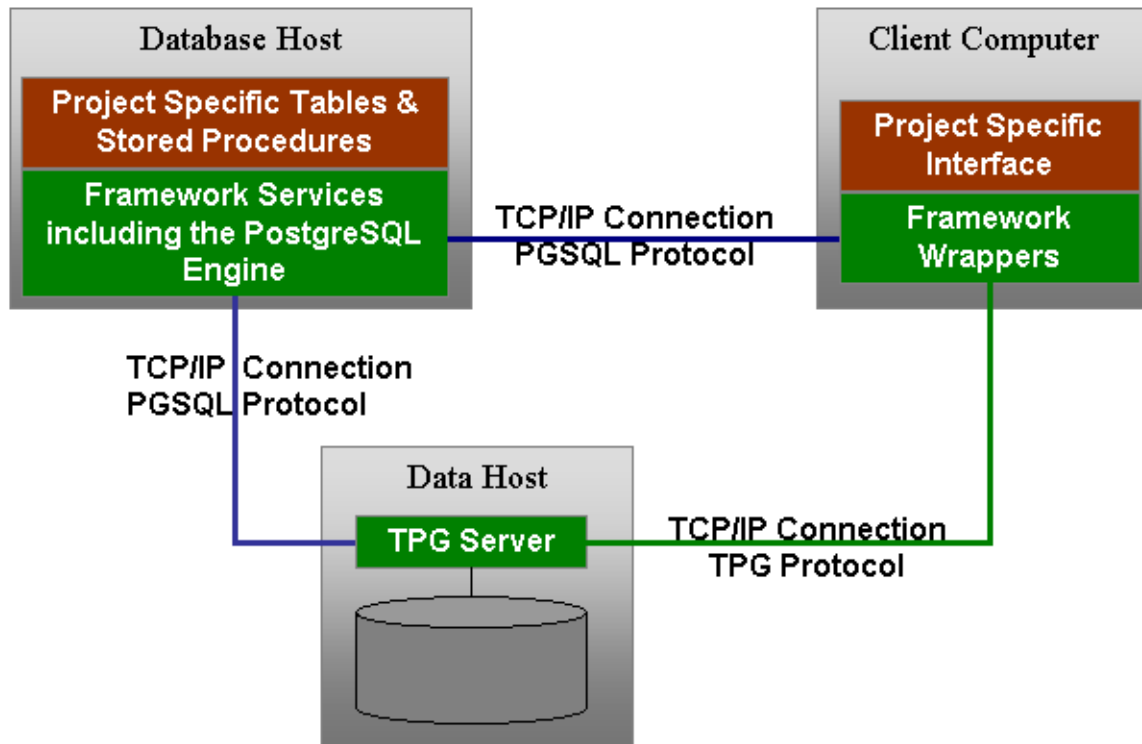
1. Since the files would not be stored in their native formats it would be very difficult to recover them in the extreme case of a catastrophic database corruption. We would like the piece of mind that comes from knowing all the files are stored on the file system in their native format.
2. It is much easier to backup the database if the images are not stored in the data tables because each piece to be backed-up (data tables, images) is a discrete, relatively small portion of an very large super-set.
3. Very large databases require a computer with 64 bits of address space and a large amount of RAM in order to efficiently address data tables over 2 Gigabytes in size. By not storing the images in the tables we alleviate the need to purchase expensive hardware in order to get short query return times.

As a side benefit of not storing files in the database tables we gain the ability to serve files from machines other than the database server.

Open Source Software License

When we begin using MIFAR for our data storage and query needs we will be highly dependent on it's proper operation everyday. There is ample evidence that software which is subject to peer review becomes much more reliable over the long run then closed source software [1]. By opening the MIFAR source we can benefit from contributions made by other groups if they use the software. Unlike the average user of consumer applications, potential MIFAR users are very likely to have skilled programmers on there staff which could provide valuable feedback and

code improvements. It is our intent to release the server side code under the General Public License (GNU GPL) [2] and the client libraries that communicate with it under the Lesser General Public License (GNU LGPL). The reason for selecting LGPL for the client code is to allow proprietary image analysis software to converse with a MIFAR server. By releasing the code under the GPL and LGPL we insure that improvements will make their way back to us. In addition there are a number of open source software packages that contain code that we would like to compile into MIFAR. Some that stand out at this time are Central Test Node (CTN) [3], DICOM Toolkit (DCMTK) [4], ImageMagick [5], Kerberos [6], OpenSSL [7] and Qt [8]. The GPL and LGPL licenses are compatible with the licenses of those packages.



Block Diagram of MIFAR Components for a General Purpose Client

MIFAR Components

PostgreSQL database engine

At the core of the MIFAR system is the PostgreSQL database engine, here after referred to as Postgres. General information on Postgres' capabilities can be found on line. [9] Postgres was chosen for the follow reasons:

1. The table handler is fully ACID compliant, meaning that transactions are Atomic, Consistent, Isolated and Durable. [10] Two other ACID compliant relational database

systems are Microsoft SQL Server and Oracle. SQL Server does not have the server side extensibility that we were looking for, and Oracle is just too expensive given our needs.

2. The database backend can be extended using stored procedures written in a variety of languages include C, Perl, Python Procedural SQL and TCL. In particular server side C functions have the ability to do anything that a standard C program could do: Including spawning new programs and communicating with mail servers. Server side flexibility cuts down on the code that must be written to automate various tasks. Instead of rewriting task automation code for each new client, all clients benefit from code implemented once in the database server.

3. Postgres is an open source program. Thus, if in the event that we need to modify it we can do so. One example where this may be necessary is if we want to use public-key authentication for MIFAR users as part of a larger participation in a Grid Virtual Organization. Though the current Grid Security Infrastructure [11] allows for the use of Kerberos tickets as an authentication mechanism and Postgres already supports Kerberos.

Postgres does in our estimation have one major draw back. By default all large objects, such as files, are stored directly in the database tables. This causes a slowdown in overall query processing speed as the database grows to over 10 Gigabytes in size. To get around this we only store file locations in the data tables.

Framework Level Database Layout

The MIFAR database implementation is broken into framework level and sub-project level tables and functions. What follows is a brief description of the major framework tables. We also have a set of sub-project level tables for our own use. However, these may not be applicable to the work of the LIDC. Soon we will document the framework level application programmer's interface and create a MIFAR administrators guide that will go into more detail.

Users The *user* table records include, login names, descriptive names, email addresses, phone numbers and optionally passwords. Only the database administrator has direct access to this table. For each entry in the user table a database view is created. SQL `select` and `update` permissions are granted on this view to the individual user so that they may maintain their own user information with out seeing protected parts of other users information. Server side functions assist the database administrator in maintaining this table and the associated views.

Groups The *group* table records user groups. In MIFAR a user is not directly granted permission on a table. Instead whenever a new *project* is created a read group and associated write group are automatically created. Then the database administrator adds

users to the project read and write groups as necessary. Server side functions make group creation, permission granting and deletion transparent.

Projects The highest level of organization in the MIFAR data hierarchy is the *project*. A project is an arbitrary set of data that is grouped for a purpose. For example the set of helical CT images and the analysis results that specify ground truth for the Lung Image Database could be a project. Data grouped into a project is treated as a common permission entity. An individual user can be given read/download permissions on a project or write/upload permissions on a project. Permission settings on one project do not carry over to another project. A user with write capability on one project may have read only permissions on another project and neither read nor write on yet a third. Though "finer than project" level permission settings are possible, they could become administratively untenable and should probably be avoided in most cases.

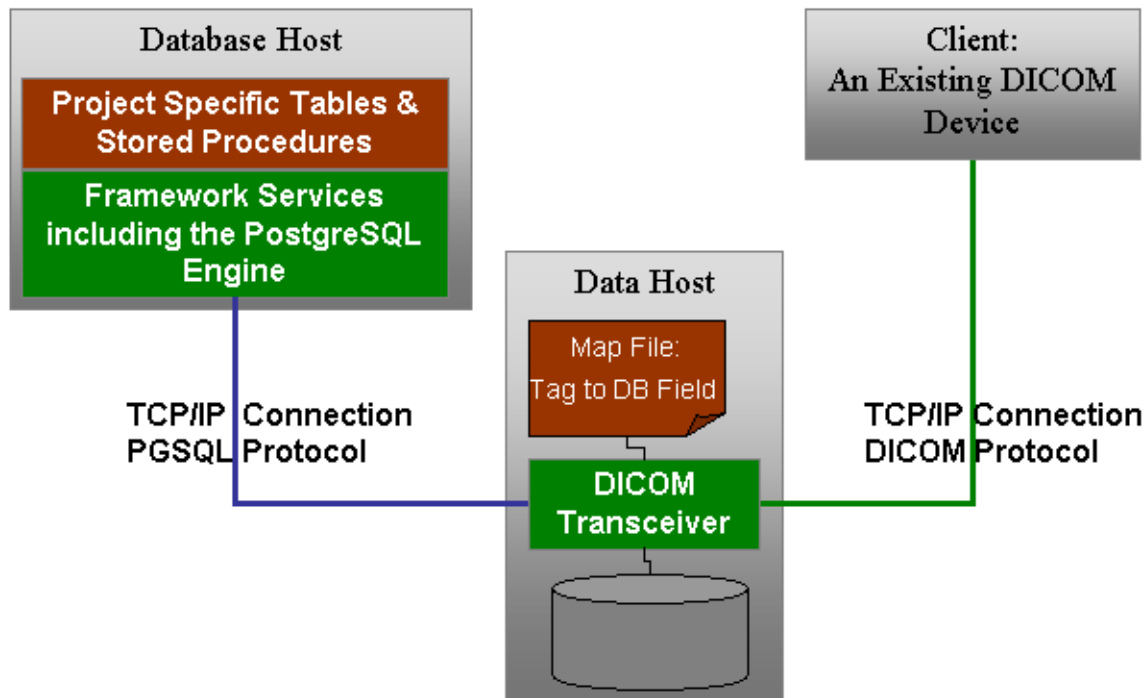
Sites and Hosts These tables help to organize where files are physically located. The *site* table represents an institution or research group that has file hosting capabilities. For each site there may be many file *hosts*. A file host is anything that can hold a copy of the data. The two major groups of hosts are computers, and offline media such as CDs and DVDs. Thus, even if a site is only responsible for keeping offline backups of images, the files they store could be tracked in MIFAR. Computers are file hosts that can also support file upload or file download operations. The host table records the projects for which a computer will accept uploads and/or downloads. Server side functions assist the host administrator (who is not necessarily the database administrator) in recording which projects may be hosted and how much server space those projects are allowed to occupy.

File Instances There is one row in the file *instance* table for each copy of a file that is known to MIFAR. Since copies of a file may exist in multiple places there may be many file instances for any one file that is tracked in MIFAR. Each file instance is considered to be in a project.

Files, File Types: For each file there is one row in the *file* table. Among other things the MD5 hash of the file is recorded so that the integrity of each file instance can be checked. Each *file* row references the file *types* table. An example of where the types table is useful would be during setup for a DICOM transmission as not all files can be moved using the DICOM transmission protocol. In the future we may list which programs or functions can understand a particular file type in this table.

Patients, Studies: Patients are not immediately grouped into projects. A patient is an independent top-level object in the database. Patients and Projects are related to each other through the study table. This is because a single patient may be involved in

multiple projects and their case history would need to be linked to them all. At this point we have not figured out how we intend to handle permissions on the patient table.



**Client-Server Design Allows for Multiple Data Host Types:
Component Diagram for a DICOM Storage Host**

Data Transport programs

MIFAR does not store data files directly in the database tables. Because of this, the native large object transport functions built into the Postgres client libraries are not useful for moving files from one machine to another. Some other means had to be used. As far as we could tell no existing programs would be useful for this purpose. On the surface it looked like FTP would be an obvious choice. However, there are a few problems with using a standard ftp server:

- 1.** In order to keep file permissions aligned with project permissions every MIFAR user would also have to have a system account and every MIFAR group would have to have a corresponding system group. In addition the file system that held the files would have to support access control lists for file permissions. The standard user, group, other permissions would not be sufficient to mirror the MIFAR permission set. This would restrict the types of computers that could act as data hosts and make administration of those hosts more difficult.
- 2.** FTP servers are not in the habit of logging their activities to a database. So a client

program could upload or download a file without informing the database at all. This would result in inconsistencies between what files are on the disk and which files are tracked in MIFAR.

To overcome these problems we created a simple network protocol for sending and receiving files called TPG (Trivial Put and Get). This protocol includes information to assist in tracking the file in a MIFAR database as well insuring file integrity. We are currently working on a set of programs to implement the TPG protocol. This route definitely seemed easier than altering an existing FTP server.

In the future we will investigate altering an existing DICOM server such as CTN or DCMTK so that it can be used as an interface to the MIFAR database. This looks possible because different AE titles can be used to determine the project associated with a dataset.

First Client: A Web site using Apache and Jakarta Tomcat.

Since web browsers are ubiquitous our first MIFAR client is a web site. This web site is served by an Apache web server which uses the Jakarta Tomcat servlet engine to render pages from Java Server Page code. Since project level tables have yet to be defined this client currently only offers features that make use of framework level functions and tables including:

File upload and download Upload and download does work but for now files are stored on the web server itself since the TPG server and client are not yet finished. DICOM and Analyze image files do have extra support in that the analyze header or DICOM tag information is parsed and used to fill out some of the database fields. The DCMTK package is used to parse DICOM fields. Region of Interest and Nodule definition files are not treated to any extra parsing at this point.

User management Users can view their information and change their password and contact information. They can also view their project permissions however they cannot change these as doing so requires database administrator rights. Creating a user account that whose information is static can be done by simply revoking the SQL update permission on the that user's information view in the database.

Online image preview Uploaded DICOM Images that are converted to a stack of JPEGs so that they may be viewed online via a Java applet. This conversion makes use of the DCMTK package. The conversion does not happen at upload because doing so would force the user to wait longer for an upload confirmation page. Instead a cron job on the web server converts the images to JPEGs in the background. If in the future, the web server does not have the processing power to do this while also servicing HTTP requests,

conversion could be farmed off to another machine. In the future we will investigate using the ImageMagick package to create JPEGs of Analyze images.

General SQL queries One of the benefits of having user permissions enforced by the backend database is that we can allow users to execute arbitrary SQL `select` queries.

The web site reads the output of these queries and looks for any column headers that it recognizes. If it sees one it knows, it dynamically creates links to the object referenced in the query output. Currently the framework level column `file_id` is in the link creation watch list. So if a random query produces output that includes a file reference, links are created to that file's JPEG stack. If the user has read permissions on the linked file they can preview the image online. Project specific search pages have not been implemented at this time.

Kerberos Ticket Server

MIFAR is envisioned as a network transparent system where multiple programs and computers work together to provide services for the end user. For example performing a file upload requires access to a client, a central database and a data receiving program. Each program may be running on a different physical computer. Requiring password authentication for each program would diminish the usefulness of the system. Currently this problem is solved by each program automatically sending the user's MIFAR password to the next program in the chain as needed. This is insecure as a single compromised program can steal a user password. In the future we will use a more secure system such as Kerberos [15]. Kerberos tickets do not contain a users password thus one compromised program can not be used to gain access to all parts of the MIFAR system. In addition to being more secure, Kerberos is compatible with the Grid Security Infrastructure. It is our intention to keep MIFAR compatible with the evolving Grid Computing initiative.

How can it help the LIDC

There are quite a few aspects of the MIFAR design that will assist the mission of the Lung Image Database Consortium. [12] One of the most important is that the MIFAR system is currently under construction. Thus any recommendations made by the LIDC can be incorporated from the "ground up." Another overall benefit is the open source code license will allow participating institutions to have direct input by way of code contributions on the system's capabilities and maintenance. The following references numbered points from the LIDC mission statement. Through participating in the development of MIFAR (or another similar system which adheres to open source principles) the LIDC will have not only provided a lung nodule image database with associated ground truth, the LIDC will have (possibly more importantly) laid the ground work for future such image database projects to be sponsored by the NIH.

(2.1.2) One of the purposes of the lung image database is to store "CT images of representative cases selected from lung cancer screening studies or diagnostic studies." These studies could make up one project with read permission granted to the public. For each institution developing CAD methods a different project could be created. Permissions could be set such that the creator or tester of the CAD method would have administrative control over the project with the ability to specify where data is stored. Since MIFAR will not require the computer serving the data files to also be the database host, participating groups could host their own data or optionally have it stored on a central file server.

(2.1.2.d) The database is expected to have "a flexible query system to allow for the evaluation of other future performance parameters for CAD, other image processing methods and related observer studies." By using the database engine to enforce user permissions we gain the ability to allow the user to execute arbitrary SQL queries. For most search parameters a web interface for selecting search criteria is desirable. However if an individual wants to perform a search for which there is not yet an interface and if they know SQL they can issue an arbitrary search without requiring alteration of any of the web forms. If it turns out that a particular query method is useful the web front end could be expanded to include the new search method at a later date.

(2.2.a) The database should be capable of storing "raw CT image data to permit ... alternative image reconstruction methods." This has been our intention from the beginning. Since raw image files can be quite large a couple of design Decisions have been made to facilitate this capability. Images are not stored in the data tables even though this is possible with multiple modern database engines. Multiple machines can cooperate in hosting the data files while still providing single point of contact for the end user. This is ideal for storing large raw image sets as different machines and even different institutions could cooperate to provide the disk space and bandwidth to supply raw data on demand.

All of the above points reference implementation objectives. However, a principle goal of the LIDC is to "establish standard formats and processes for managing lung images and related technical and clinical data for use in developing and testing computer aided diagnosis." The reason we began developing the MIFAR system was to solve this type of problem. We have a wide variety of data that is not sufficiently organized nor easily sharable with our collaborators. Hence the need for extensible data management system targeted toward medical image researchers. Since it is an open source project, MIFAR provides a platform whereby the LIDC members can collaborate to meet their goals while providing reusable tools that future consortia can use to further their goals.

References

1. <http://www.openhealth.com/en/opensource.html> Access date 4/25/2002
2. <http://www.gnu.org/licenses/licenses.html> Access date 4/25/2002
3. <http://wuerlim.wustl.edu/DICOM/ctn.html> Access date 4/25/2002
4. http://www.offis.uni-oldenburg.de/projekte/dicom/soft-docs/soft01_e.html Access date 4/25/2002
5. <http://www.imagemagick.org> Access date 4/27/2002
6. <http://web.mit.edu/kerberos/www> Access date 4/28/2002
7. <http://www.openssl.org> Access date 4/25/2002
8. <http://www.trolltech.com> Access date 4/26/2002
9. <http://www2.ca.postgresql.org/features.html> Access date 4/28/2002
10. <http://developer.postgresql.org/pdf/transactions.pdf> Access date 4/28/2002
11. <http://www.globus.org/documentation/incoming/butler.pdf> Access date 4/28/2002
12. Lung Image Database Consortium - Mission Statement.