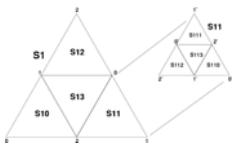
Software tools to easily store, share and present scientific data: MCS, a new approach to data treatment in astronomical projects

Giorgio Calderone, Palermo & Luciano Nicastro, IASF-INAF, Bologna, Italy



In brief

MCS is a set of C++ high level, easy to use, classes aimed at implementing an application server, that is an application that provides a service over the network. Its main features, amongst others, are the possibility to customize the server behaviour through the derivation of some classes, and the usage of a well defined communication protocol (the MCS protocol).

With MCS you can easily implement custom services on top of which different users (say scientists, technicians and other people) can perform requests to a common database through several tools obtaining different types of data, depending on the tool used and the user permissions. The MCS protocol and software tools let user's application obtain data in a well defined binary fashion or plain text, so that they are ready to be processed further.

So MCS and its protocol are for software applications, what a web server and HTTP are for the WWW: a simple way to access data. In this comparison customizing the MCS server is like writing a web page.

In a typical scientific experiment we have an instrument producing data, a main storage system, a set of software tools to perform analysis, and people with different needs who wish to access the data. In this poster we'll analyse the components of an informative system based on MCS, applied to such an experiment.

Instrument

The software controlling the scientific instrument can inform the MCS server that new data are available using the `Client` class to connect to it and send all data and/or files through the connection. The software can also receive commands from the MCS server, for example if it is a telescope, the coordinates to point it at or the filter to use. Data sent to the server can be stored in the database or in a dedicated directory or relayed to another MCS server over the internet.

Database

The database is used to store all application specific data, for example a log of what has been done by the instrument, the list of files produced by it, house-keeping information, etc. It is worth noting that the data can be either left in files (in any format) or reported in BD tables. In the latter case it could be worth implementing *DB engines* which allow a transparent I/O on these tables as if they where files with the original format. MCS foresees FITS and VOTable engines at the moment. Once the data have arrived to the MCS server the `LocalThread` class can be used to perform a quick-look of the data and store the results in the DB as well.

External programs

If you already have programs or routines to analyse the data, you can customize the `LocalThread` class so that, as soon as they are ready, they are automatically analysed using that software and the results made available to the (authorized) users. External programs can also be executed as a response to a user custom commands (processing on demand).

MCS

MCS is the core of the informative system. Its classes are designed to hide all low-level aspects of implementing an application server like socket handling, multi threading, etc…Users should only provide the code to implement specific tasks. The service offered by MCS is similar to an operative system shell which allows you to execute commands. Customizing the server means implementing new user commands. Note that a set of Astronomy oriented tools are already (or will soon be) available as a *user contributed library*. It includes the `HTM` and `HEALPix` libraries used for sky pixelization and objects indexing, as well as the astrometric `NOVAS` library for date and coordinates conversion as well solar system bodies information.

A schematic example of a customisation to perform a fit over some data stored in the Database:

```
If (Command == MAKE_FIT) {
  GET data from db;
  FIT data;
  RETURN data to user;
}
```

Similarly if, for example, you wish to make a sky map of survey data.

Server side

Client side

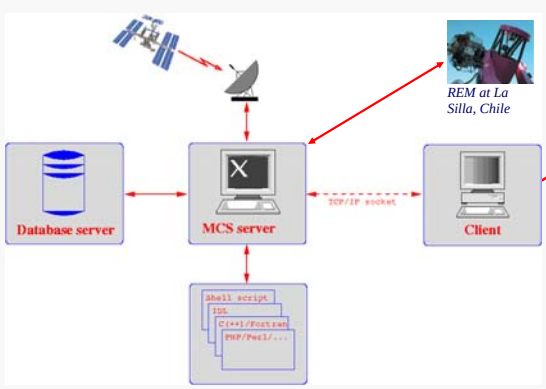
Web server

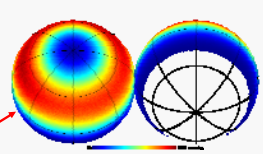
A web server can easily be turned into an MCS client using the provided PHP interface. This way you can build dynamic web pages which can show the status of an experiment or Observatory, as well as create *real-time* outreach information. This can be done already by using other libraries/tools but you'll probably need to put together different data produced by independently developed s/w. MCS offers a single homogeneous environment which can easily host existing s/w with little changes.

Users

MCS provides a number of interfaces to different programming languages: C/C++, Fortran, IDL, PHP. Using any of these interfaces you will get the full potentiality of MCS and its binary protocol. New interfaces will be implemented in the near future: Perl, Python, Java. The data transferred (in both directions) can be in a binary format, but can also be in the form of a FITS, XML or VOTable file. Finally MCS has also a "text only" mode by which you can access the server using a simple `telnet` client. This interactive mode is very useful for quick checks on the DBs and to perform simple tasks.

Flow diagram for an MCS Server/Client system. One Server accepts multiple connections from Clients on the internet performing the requested tasks.





This diagram summarises the way MCS works. A Server collects data coming from an Experiment (ground or space based) and saves log information and data into DB tables.

A Client sends requests (using any language capable of socket connection) and retrieves info/data. Eventually require the Server to perform specific tasks on the data getting the results back, for example a sky map. The programs can be written in any language.

http://spora.ifc.inaf.it/mcs/

MCS is open source and it is downloadable from the Web or requested to us. Documentation is included in the distribution and it is available as `doxygen` HTML pages. A descriptive document (TeX+PDF) is also included. The distributed tarball provides the typical and simple `configure/make/make install` sequence. The only required external libraries are `libmysqlclient` and `libpcre`. A full test suite is included together with demo programs in the various supported languages. If you wish IDL/PHP compatibility you need these packages to be installed too. Please note that the library is in continuous improvement so we ask the interested people to get in contact with us. We are seeking collaborations and we plan to have a distribution mailing list. The current version of MCS is 0.2.1. Get it from `spora.ifc.inaf.it/mcs/mcs-0.2.1.tar.gz`

A demo describing the example reported in this poster is accessible at `ross.iasfbo.inaf.it/demo_mcs/`