A QUICK INTRODUCTION TO VOMS

1.INTRODUCTION

Aim of this write-up is to give an overview on the current VOMS “affaire”: a description of the mechanism currently used on the LCG/gLite middleware, the relationships among various actors, and issues/remarks to be taken into account are presented.
The current organization of the LHCb specific VOMS server is also discussed at the end of the document with wished properties for each group/role that an LHCb member could potentially dress.

2. VOMS

Authorization in present Grid applications is based on the concept of **Virtual Organizations (VO).** VOs administer users, grant them permissions and establish agreements with Resource **Providers (RPs)**. A RP enforces local authorization in turn. In the framework of Globus, authorization relies on a simple mechanism: the “gridmap-file”. It consists of a simple list, resident on resources, of all authorized grid users expressed as Distinguished Names and associated with the corresponding local credentials (e.g. usernames on Unix systems). This mechanism would work for relatively small projects (=grids) and in any case it doesn’t answer the fine grained authorization requirements from VO that could have a complex, hierarchical structure with groups and subgroups.
Every user in a VO is then characterized by a set of attributes,
i.e. 3-tuples of the form (group, role, capability). The combined values of all of these 3-tuples form unique attributes, the so-called **Fully Qualified Attribute Names" (FQANs)**
In general an FQAN has the following form:

/VO[/group[/subgroup(s)]][/Role=role][/Capability=cap]

For example, the FQAN corresponding to the role *root* in group *production* of VO *lhcb* is:

```
/lhcb/production/Role=root
```

A first interim solution adopted in the early days of LCG consisted in using Lightweight Directory Access Protocol (LDAP) servers to manage the authorization information at the VO level, and developing the *mk-gridmap* utility to allow resources to automatically download this information and generate the "gridmap-file" locally on the resources.

VOMS **Virtual Organization Membership Service** Provides information on the user's relationship with his VO: his groups, roles and capabilities. It is a system to classify users that are part of a VO on the base of a set of attributes that will be granted to them upon request and to include that information inside globus-compatible proxy certificates (VOMS extensions).

VOMS consists of two main components:

- **VOMS** - includes the VOMS server and the VOMS client tools (e.g. *voms-proxy-init)*
- **VOMS Admin** - a Java server application (and UI servlet) used to manage users and their privileges for a VO

2.1 Voms server

The VOMS server is basically a simple account database, which serves the information in a special format (VOMS credential). The VO manager can administrate it remotely using command line tools or a web interface. It might be intended composed by the following parts:

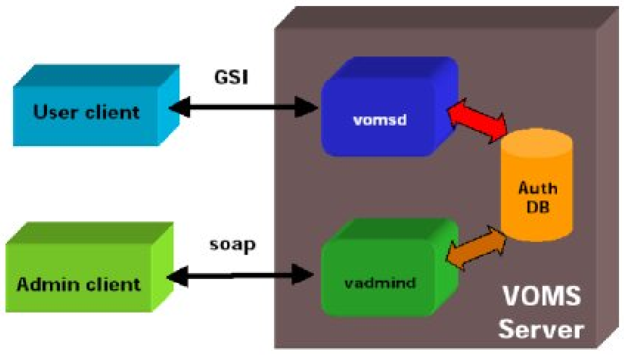

Fig.1

1. **User server** receives requests from a client and returns information about the user.
2. **Administration Server**: accepts the requests from administrative clients (that allow VO managers to add/remove/change users/groups/roles) and updates the Database.

2.2 Voms clients

The VOMS clients are used to both create a valid **grid-proxy** (once done via grid-proxy-init) and a **voms extension** that holds further attributes (granted to the user). The grid-proxy is used in the handshake mechanism for GSI authentication (PKI mechanism) while the voms-extension has to be intended as a rule for specifying further properties/privileges that a given user is granted to access on the resource. Using *voms-proxy-init* a user creates then a GSI proxy with special permissions that his VO entitles him to. Grid services that he will subsequently authenticate with may be configured to read these attributes from his proxy and perform decisions based on their values.

The voms-proxy-init command contacts the VO's VOMS server the user belongs to (ex. lcg-voms.cern.ch:15003), authenticates to it using the "standard" proxy certificate, receives the VO-specific attributes, and creates a new proxy with these attributes appended. To specify the name of the VO to contact use the --voms optin, e.g.:

```
voms-proxy-init --voms lhcb
```

voms-proxy-init finds the address of the server for the given VO (i.e. *lhcbt*) by looking through a series of configuration directories, namely:

- /opt/glite/etc/vomses
- *$X509_VOMS_DIR* environment variable
- ~/.glite/vomses

Each of these directories may have files of the following format:

```
NAME SERVER_HOST SERVER_PORT SERVER_DN DESCRIPTION
```

For example:

```
[santinel@lxb2004 vomses]$ cat lhcb-lcg-voms.cern.ch
"lhcb" "lcg-voms.cern.ch" "15003"
"/C=CH/O=CERN/OU=GRID/CN=host/lcg-voms.cern.ch" "lhcb"
```

The administrator(s) of your VO(s) should give you the exact settings.
Note: VOMS has fairly strict permissions/ownership requirements on these data files, in particular:

- *$X509_VOMS_DIR* must be owned by exactly root:root
- The */opt/glite/etc/vomses* or the directory pointed by *$X509_VOMS_DIR* must have permissions exactly 0755
- The actual data files must have permissions exactly 0644

Otherwise VOMS tools might refuse to read these files, and you will get the following error:

```
VOMS Server for MyProject not known!
```

The syntax for explicitly claim to belong to a given group or to play a special Role/Capability is the following:

```
voms-proxy-init -voms <vo>:/<vo>/<group>/Role=<Role>/Capability=<cap>
```

Please be aware that in case the user is not permitted to have a special attribute he will get:

```
Error: lhcb: Unable to satisfy G/lhcb/sgm Request!
```

2.2 Operation

One strong requirement VOMS faced with, was to disrupt as little as possible – from the user's standpoint – the creation of the user proxy certificate. To achieve that it has been added the voms-proxy-init command to be used in place of grid-proxy-init.

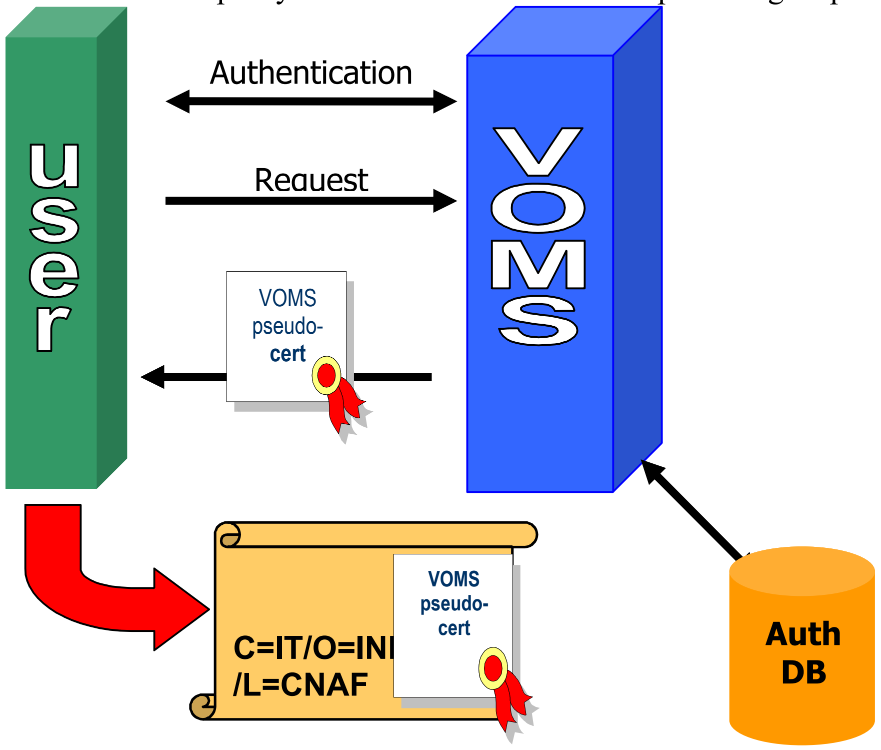

Fig. 2

This new command produces a user's proxy certificate – like grid-proxy-init does – but with the difference that it contains the user info from the VOMS server(s).
This info is returned in a structure containing also the credentials both of the user and of the VOMS server and the time validity. All these data are signed by the VOMS server itself. We call this structure a "Attribute-Certificate" (Pseudo-Certificate).
The user may contact as many VOMS's as he needs.
In order to use the authorization information, the *Gatekeeper*, in addition to normal certificate checking, has to extract the additional information embedded in the proxy done with appropriate LCMAPS plug-in although, as the VOMS info are included in a

non critical extension of the certificate, this can be used even by “VOMS-unaware” ga*tekeepers*, thus maintaining compatibility with previous releases.

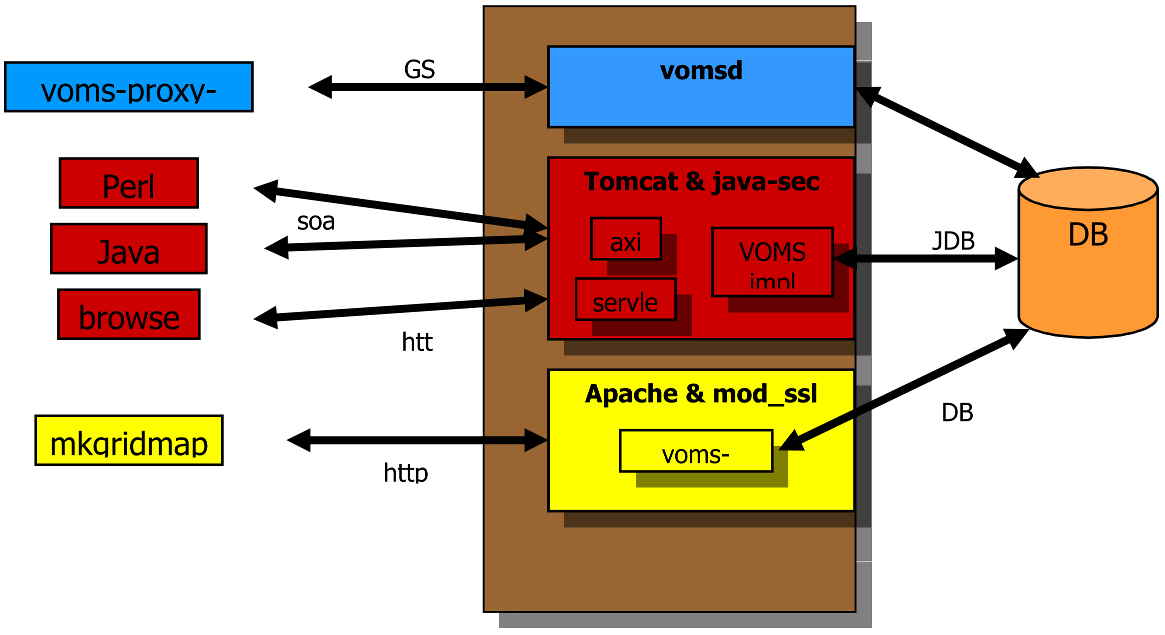

Fig.3

Another use of VOMS is the administrative one. Special clients (GUI and CLI) share a common server to modify the database. The server can be reached by the SOAP protocol. The server consists in three sets of routines, grouped into services:
The **Core**, which provides the basic functionality for the clients;
The **Admin**, which provides the methods to administrate the VOMS database;
The **History** which provides the logging and accountability functionality.

Finally (as evident from the figure 3) VOMS is used as LDAP servers were used in the past: as source of information for building old existing grid-mapfiles used as fall back solution for mapping user credentials once VOMS-LCMAPS mechanism does not work. Each service contacts then VOMS through cron jobs for updating static grid mapfiles We will go later on this aspect and rules used.

2. VOMRS

VOMRS stands for Virtual Organization Management Registration Service and it has to be considered as yet another service (with respect to VOMS).
Its main purposes are to offer a omni comprehensive set of services that facilitates secure and authenticated management of VO membership, grid resource authorization and privileges:

- ❑ implements a registration workflow providing means for collaborators to register with a Virtual Organization (VO)

- supports management of multiple grid certificates per member
- permits VO-level control of a member's privileges
- provides email notifications of selected events
- supports VO-level control over its trusted set of Certificate Authorities (CA)
- permits delegation of responsibilities within the various VO administrators
- manages groups and group roles
- is capable of interfacing to third-party systems and pulling or pushing relevant member information from/to them

Through VOMS admin APIs, any modification and reorganization of the VO membership are propagated to VOMS. A clear picture summarizing the system is given in figure 4

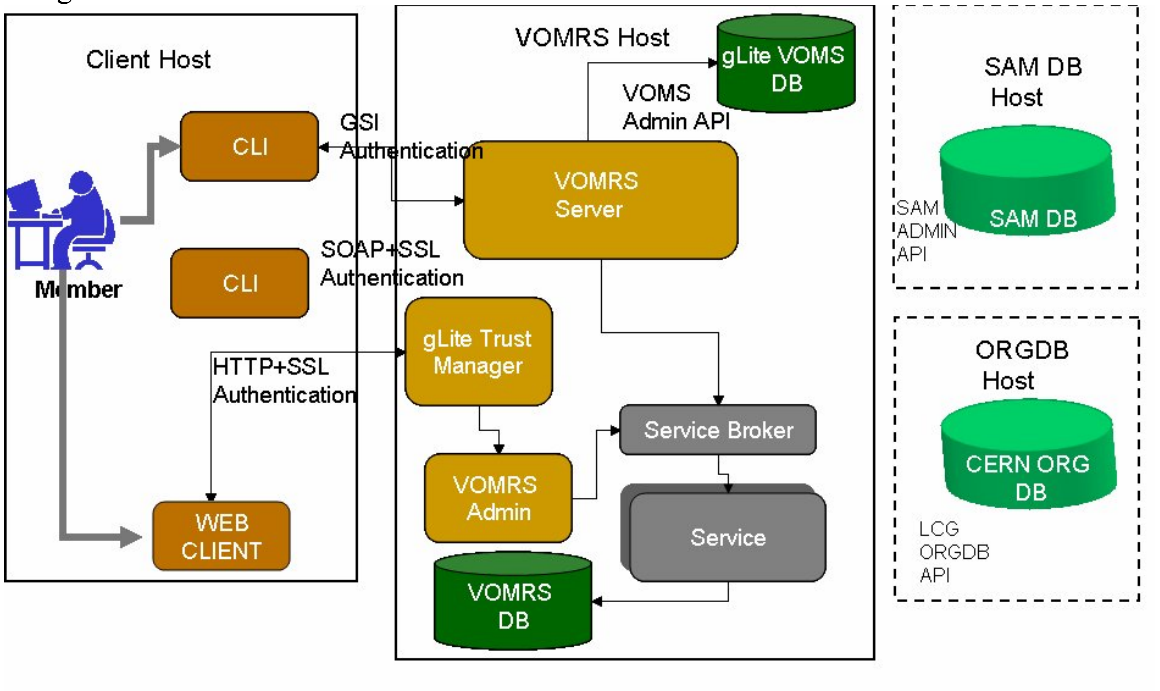

Fig. 4

3. LCAS/LCMAPS

In order to interface the middleware to the local fabric, LCAS and LCMAPS have been developed. **Local Centre Authorization Service (LCAS)** handles authorization requests to the local computing fabric. It represents a hook used by site managers to ensure that local policies are respected. The current version of LCAS allows for banning specific users, for putting time restrictions on job submission and for permitting only a restricted set of users accessing to the resources. After a given Grid User DN is authorized, it has to be mapped to local (UNIX) users. Operating systems have no specific knowledge of "Grid Users". Therefore it is needed to translate the idea of a grid user into that of a local user.

Local Credential Mapping Service (LCMAPS) provides all local credentials needed for jobs allowed into the fabric. LCMAPS is a framework that can load and run one or more 'credential mapping' plugins. Based on the user global credentials (more specifically the user's X509 certificate) and the job specification (JDL), the LCMAPS plugins have to perform either of these two tasks:

1. acquire local credentials or
2. enforce (apply) the local credentials

Local credentials are gathered (UNIX uids, gids, VO information, AFS/Kerberos (?) tokens) and are stored somehow.
LCMAPS has 2 different set of plugins. Let’s call them *VOMS* and *non-VOMS* aware

In the first set we have:

Non-VOMS aware plugins

- `lcmaps_localaccount.mod` : it collects the local account name from the *gridmap* file.
- `lcmaps_poolaccount.mod` : it collects a pool account name from the *gridmap* file (leases in $GRIDMAPDIR).
- `lcmaps_posix_enf.mod` : it enforces the local credentials in the running process by posix system calls (setuid(), setgid() etc.).
- `lcmaps_ldap_enf.mod` : it enforces the local credentials by setting the primary and secondary gids in the LDAP database that is used by the site as the source of account information for PAM or NSS.

VOMS-aware plugins

- `lcmaps_voms.mod`: it extracts the VOMS information from the user X509 proxy certificate.
- `lcmaps_voms_localgroup.mod` : it tries to find a local group Id (gid) based on the VO information and a *groupmapfile*. With the VOMS information it gathers primary and secondary GIDs by matching VO-GROUP-ROLE(-CAPABILITY) combinations in the so-called *groupmapfile* and by finding the corresponding local GID. Example of groupmapfile is given below.
- `lcmaps_voms_poolgroup.mod`: this plugin tries to find a pool group Id (gid) based on the VO information and a *groupmapfile* (leases in $GROUPMAPDIR)
- `lcmaps_voms_poolaccount.mod`: this plugin tries to find a pool account based on the VO information and a *gridmapfile* (leases in $GRIDMAPDIR)

```
 [root ce101 root]# grep lhcb /opt/edg/etc/lcmaps/gridmapfile

"/VO=lhcb/GROUP=/lhcb/ROLE=lcgadmin/Capability=NULL" lhcbsgm
```

```
"/VO=lhcb/GROUP=/lhcb/ROLE=lcgadmin" lhcbsgm

"/VO=lhcb/GROUP=/lhcb/ROLE=production/Capability=NULL" lhcbprd

"/VO=lhcb/GROUP=/lhcb/ROLE=production" lhcbprd

"/VO=lhcb/GROUP=/lhcb/Role=NULL/Capability=NULL" .lhcb

"/VO=lhcb/GROUP=/lhcb" .lhcb
```

3.1 Worth to know about mapping and VOMS

The fully VOMS-aware (=mapping through VOMS plugins) mechanism does not require any synchronization server-grid resource. The effect is immediate. What it is matched is the FQAN (or a list of FQANs) and then the plugin know how to deal with this FQAN. What has to change in case a given VO want a different behavior is the rule to be used for dealing with the FQAN. These rules are generally provided through YAIM configuration files and adequate YAIM scripts that build the group mapfile for LCMAPS accordingly. Other YAIM scripts are also used for building the mkgridmap script used for creating stating grid mapfile. Changing a rule for building the gridmapfile should be opportunely deployed.
(for instance if you want that the old grid-mapdfile built via VOMS maps all user in the group /lhcb/sgm into lhcbsgm account you have to modify the YAIM script at each site level; the default rule is indeed that users in Group= /lhcb group and Role=lcgadmin are considered to be mapped to lhcbsgm)

Note that the immediateness propagation of the mapping rules isn't true with the old grid-mapfile mechanism that requires LDAP central service (or VOMS server!) modification to be propagate everywhere. Be careful that the old mapping mechanism does require the sync.

If the VOMS-aware plugin doesn't find the match, any other FQAN that sits out of this list is ignored, LCMAPS fails and the old gridmap-file is used instead. The DN is checked against the old grid-mapfile and the mapping is done through non-VOMS aware plugins. This mechanism works for Computing Element, RB, FTS, Storage Elements. LFC and DPM do use yet another mapping mechanism based on propertary plugins.

The gridmap file generation is not longer done via LDAP since October 2006. It means that all grid-mapfile (at site level) will be generated using VOMS as source of information. Do not get confused: the grid-mapfile will need always a while to be sync with the new service (VOMS instead of LDAP). If you then change something in the VOMS server, you need to wait at least 8 hours until all sites will update their own grid mapfile!

LDAP servers (containing registered users) have been synchronized with VOMS servers. User mapped to normal pool account sit in to the default /lhcb groups w/o further Roles. User sitting in the old lcgadmin ldap servers, are intended belonging to Group=/lhcb group **and** Role=lcgadmin. The site **mk-gridmap** scripts will query LDAP and VOMS servers indifferently. The rule it uses for building the grid-mapfile is:

1. OR of the possible mapping is taken with the choice that falls into the most privileged mapping. (voms 1 says .lhcb, voms2 says .lhcb and LDAP says lhcbsgm the user DN gets mapped to lhcbsgm)
2. VOMS are queried with simple rules that **can be customized** via YAIM. The default is the one used for synchronizing VOMS with LDAP.

VOMS-aware LCMAPS plugin tends to get as many GIDs as possible. Once sgm and prod accounts will not be longer static account but pool accounts like lhcbsgm[001-050] will be in place the current organization of LHCb could be potentially dangerous. Having groups-only mapping (w/o extra roles specified) might be a problem because the privileges will be per UNIX group (and not longer GID/UID Unix pair) and anyone registered in a VOMS group sgm, **unwillingly**, inherits privileges automatically. This because the FQAN presented via VONS extension (that are always listed) are matched on the groupmapfile. Even if the user (through voms-proxy-init) explicitly asks for some other group it gets also GIDs competing to this (these) super-privileged group(s) in the list.
Using Roles explicitly (like the trick we thought of using the "user" role) prevent this kind of situation; there is not chance indeed to present a FQAN for being mapped to sgm group unless you explicity require the Role="user" (which is not by default in the list of possible default FQANs) . This enforces another pillar of the security schema: the minimum risk principle. The mnemonic schema for this rule of thumb is:
GROUP→ Unix 'su' command
ROLE → Unix 'sudo' command
The Role is on demand and super privileges are not used accidentally but are always explicitly required.
The Group is there for free, just because I belong to! The Group should be rather considered an activity

VOMS extensions cannot be renewed via my-proxy mechanism. This is not an issue once the job has been submitted to the CE and gets its own slots. The RB indeed does not need the FQAN but just the old proxy component for authenticating users against gatekeepers and gridftp servers (for sandboxes upload and download). So even if the voms extension is not longer valid because LCG RB does not support its renewal **(however in place with the gLite WMS)** the job will keep running and upload its output.
The maximum expiration time of a voms extension may be configured on the server.
It might be worth to note that the current mechanism used by the LCG RB to monitor jobs running in the site is per DN. In a fully VOMS aware CE the grid-monitor process that runs on the CE and that talks with the gaph server of the LCG RB (via GASS protocol) runs under different users for different VOMS proxy used. This turns out that

the LCG RB could not handle two concurrent jobs on the same CE by the same users by with different VOMS proxies used. This issue disappears with the new gLite WMS.

4. GENERAL OPERATION

The last picture summarizes the general mechanism for getting a grid service. A user gets via voms a proxy certificate and its voms extension. Via the proxy certificate he *authenticates* him self against any grid service. The *authorization* goes then through LCAS mechanism and finally LCMAPS (or some alternative systems like the one used in LFC and DPM) *maps* the GRID user credentials into local Unix account/group that reflects the granted privileges to that user.

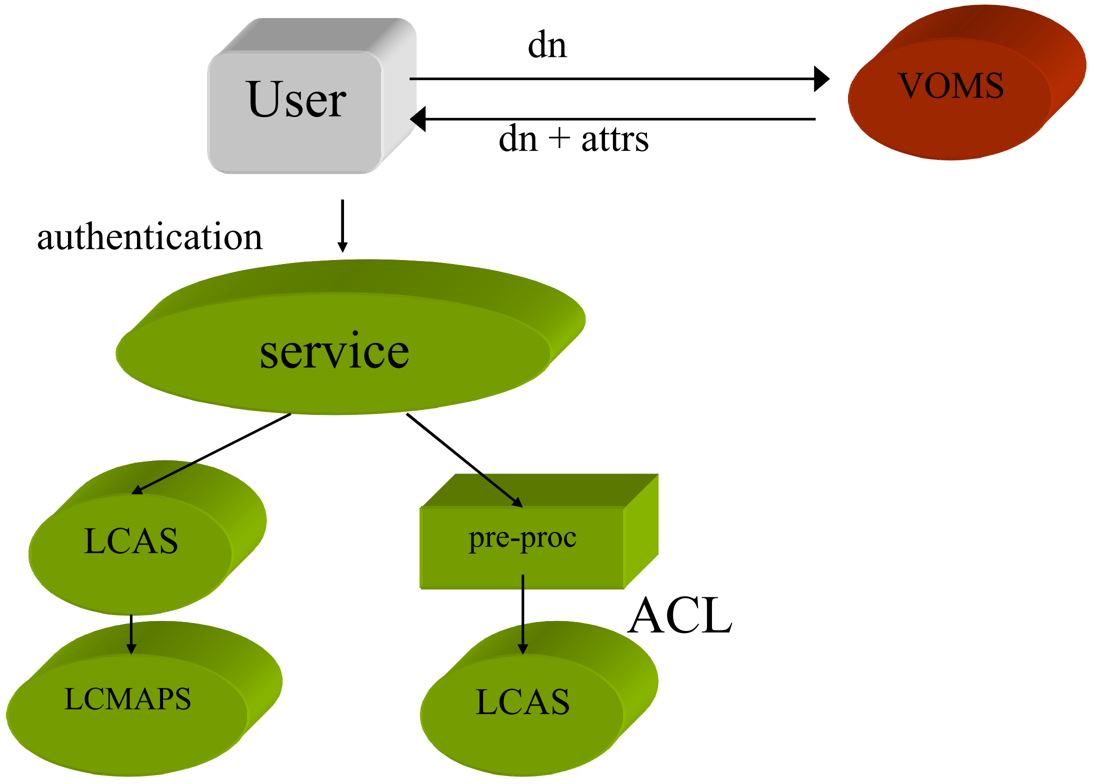

Fig. 5

5. VOMS and LFC/DPM

While the WMS relies on the LCMAPS mechanism for getting local UID/GID on the user VOMS proxy presented basis, the DMS implements the concept of Virtual Ids. They are site independent identifiers and do not require any administrator action. A DN is mapped to a virtual uid and a VOMS group or role to a virtual gid. They are currently used by LFC and DPM. Only te virtual uid and the primary virtual gid (i.e. the ID

mapped to the primary group as intended in [chapter for VOMS]) are used to control access to the File Catalog entries or the file on the Storage Elements.
Unless changed with ***chown*** the files are owned by the DN of the user who created the file; the group ownership depends on the *S_ISGID* value of the parent directory: if it is set, the file group ownership is the same as the parent one, if not the primary group of the user is used.
The LCG Data Managements is also designed for supporting secondary groups. Secondary groups are useful as explained by this example: Let's suppose a given user dressing the "production" role needs to access some collective files whose group ownership is "lhcb". Let suppose these files are not world readable. In this case, if secondary groups were not supported (and then the user is not seen from within the "lhcb" group), - unless not explicitly set by an accessing ACLs giving "lhcb/Role=production" as supplementary group - **the user cannot access these files**.
(This is what Joel ran on at beginning of October):
There are two different "level" of ACLs: base and extended. The former map directly to standard UNIX permission (owner, group owner, others), the latter correspond to lists of supplementary users/groups. The existence of supplementary users/groups does require also an ACL mask be defined. Two different types of extended ACLs: access and default:

• Access ACLs: can be set on both files and directories and are used to control the access
• Default ACLs can be set on directories and are inherited as access ACLs by every file or sub-directory underneath unless explicitly specified differently. The default ACLs are also inherited as default ACLs by every sub-directory. LFC and any SE (HPSS) with SRM v2 interface support Posix ACLs.

Permission to delete a file depends on the *S ISVTX* bit in the parent directory: one always needs to have write permission on the parent directory. Furthermore, if the *S_ISVTX* bit in the parent directory is set, one further needs to be the owner of the file or write permission on the files.
Hint In case of user analysis files, in order to guarantee privacy, it is suggested to set the S ISVTX bit on parent directory and the mode of the file to 0644 or even to 0600 if the file must only be seen by the owner (DN).
Conversely, in case of production files, the bit of the parent directory must be set and the permission mode to 0644
(the file must been accessible at least by the members of the group).
A special VOMS role "*lhcb_data_admin*" can finally be defined to grant users with this role, accessing all data of the Virtual Organization. This can be realized in two different ways:

1. An ACL entry is opportunely added to every file or directory to give the role "*lhcb_data_admin*" all permissions on them.
2. The role "lhcb_data_admin" can be recognized by all LCG Data Management services as being a special role to get all permissions on files or directories owned by the VO (without using ACLs).