



Project no. 034567

# Grid4All

Specific Targeted Research Project (STREP)
Thematic Priority 2: Information Society Technologies

# Deliverable 3.6: Implementation of VO-aware filesystem.

Due date of deliverable: June, 2009.

Actual submission date: .

Start date of project: 1 June 2006 Duration: 37 months

Organisation name of lead contractor for this deliverable: (ICCS)

Revision: final

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)

**Dissemination Level**

| | | |
|---|---|---|
| **PU** | Public | √ |
| **PP** | Restricted to other programme participants (including the Commission Services) | |
| **RE** | Restricted to a group specified by the consortium (including the Commission Services) | |
| **CO** | Confidential, only for members of the consortium (including the Commission Services) | |

# Table of Contents

# Grid4All list of participants

| Role | Part. # | Participant name | Part. short name | Country |
|---|---|---|---|---|
| CO | 1 | France Telecom | FT | FR |
| CR | 2 | Institut National de Recherche en Informatique en Automatique | INRIA | FR |
| CR | 3 | The Royal Institute of technology | KTH | SWE |
| CR | 4 | Swedish Institute of Computer Science | SICS | SWE |
| CR | 5 | Institute of Communication and Computer Systems | ICCS | GR |
| CR | 6 | University of Piraeus Research Center | UPRC | GR |
| CR | 7 | Universitat Politècnica de Catalunya | UPC | ES |
| CR | 8 | ANTARES Produccion & Distribution S.L. | ANTARES | ES |

# 1 Executive Summary

This report accompanies the final VOFS prototype, a file system aimed at groups of casually collaborating users, proposing a new way to think about sharing documents. VOFS is documented in various other deliverables cited below, and a practical guide to acquire and install the software is included.

The most prominent features of VOFS are: one file-serving peer per user, web-like, cross file system links for federation, controlling file storage through storage pools, responsiveness despite network failure and disconnected operation, flexible access policy mechanisms, and easy extensibility.

# 2 Introduction

The VOFS prototype is essentially an implementation of the core functionality of the VBS+DFS architecture as documented in Deliverables 3.2 and 3.3. An overview of VOFS and its benefits to the user is provided in Deliverable 6.7. An evaluation of VOFS from feedback provided from the use of the VOFS prototype is available in Deliverable 5.3.

The VOFS prototype allows users to create filesystems of their own and serve and access them over the network, and collaborate with other VOFS users through shared workspaces.

The goal of VOFS is to provide a simple and independent environment for collaborating users without the need for externally administered services or infrastructure. Instead, users create their own structures according to the workflows and third party services are presented to users through peers in the network.

Traditional peer-to-peer approaches such as block stores and file systems on DHTs, focus on providing technology to manipulate storage on peer networks, but do not address the file sharing from the practical user perspective. VOFS makes its contribution on a higher level than this and specifies an architecture where this technology can be put to good use from the user perspective.

Simplicity in sharing is the target of many web-based solutions such as `dropbox.com`, `box.net`, `wuala.com`, but all of them rely on a central service to function properly.

Section 3 provides a practical guide to acquire the software and documentation and proceed with its installation. Section 4 outlines the prototype in terms of architecture components that were implemented. Section 7 shortly discusses the basic workings of the prototype and describes its basic features and their usage.

# 3 Availability

The prototype software and its installation and usage manual are available at

`http://www.cslab.ece.ntua.gr/vofs/`

The usage manual details the procedures of installation, configuration and launching of the software, which are outlined in this section.

The package can be extracted to a hierarchy of files under a directory named `vofs`. In this directory there are:

- The usage manual for the software, named `vofs-manual.pdf`
- The installing script `setup.sh` (a symbolic link to it)
- A directory `src` which contains the software

### 3.1 Installation

VOFS is run from the extracted directory after proper set-up. To set-up, run the script `setup.sh`. The script will make the necessary compilations, set up a peer identity and a corresponding workspace configuration, and will create a mountpoint and customised launching script at the designated locations.

After running the script, VOFS is launched using the `launch-vofs` script and the filesystem is available under the mountpoint that was configured during the setup.

It is possible to independently launch VOFS peers for more control. For details refer to the usage manual.

### 3.2 Software Requirements

- Linux Operating System
- Python 2.5
- Fuse-2.7 and standard C library development files
- SQLite 3 and Python bindings
- C development files

## 4 Implementation Overview – Software Components

The prototype is an experimental implementation of the VOFS architecture that is mainly directed towards end-users. There are two important reasons for this focus. First, experimenting with the VOFS architecture from the application or the service provider point of view is a process that needs effort in development. Second, the biggest contribution of VOFS is to the end users. Since extending VOFS and supporting applications (i.e. Telex) was demonstrated earlier, a simple tool for collaborating users was deemed a more meaningful focus.

## 5 Implemented Components

The VOFS software bundle consists of the components:

- Networking back-end. It handles connections to remote peers and transfers.
- Generic storage back-end. It stores persistently peer state including file hierarchy and file metadata.
- Data storage back-end. Stores partial or whole files and keeps track of local changes.
- Generic peer. Implements the protocol for file and data serving.
- Generic client library. Implements the core client logic that interacts with file servers and storage providers, manages cache and disconnected operation.
- POSIX client library. Implements POSIX calls on top of the core library. It also implements the virtual files.
- FUSE client library. It bridges FUSE with the POSIX client library to provide a mountable file system.
- Setup and workspace-creating scripts. Provide a higher level setting up and controlling VOFS over low-level virtual file access.

# 6 Components not Implemented

Basic components that were not implemented are:

- Cryptographic support for peer identities and communication encryption. Cryptographically secure systems are difficult to engineer and this effort is well out of scope.
- Global Identity Index. This is not strictly required as peers can always identify themselves with their hostnames or IP addresses. A special identity index makes sense only if cryptographic support is also implemented.

## 6.1 Integration

The software package does not include plugins and extensions that were implemented to demonstrate integration with other technologies (such as alternate storage back-ends) or other Grid4All modules (such as the Security Infrastructure). The goal of this software package is to provide users with a simple experimental tool. Since an integrated prototype for Grid4All is out of scope, it is only useful to maintain the extensions only for demonstrative purposes.

# 7 Overview of main features, capabilities and usage

## 7.1 Overview of the workings

VOFS peers create a network of file systems which are identified by a unique identity that can be used in authentication and authorisation of transactions among them. A peer's name can be either its identity or a name through which the identity can be retrieved. Each peer may assume one of three roles all supported by a unified protocol: file server, storage provider and client.

The file server is the authority to serve a user's files to the network. File servers keep the file hierarchy and metadata for each file. Each file is identified by the peer's identity followed by a path unique to each file server. Clients may explore a file server's hierarchy. There are symbolic links that can be followed accross the network, effectively linking the filesystems together in a WWW-like fashion.

File data are not normally kept in the file server. Instead, a list of storage chunks is kept with the file metadata. These storage chunks are kept and served by storage providers, similarly to file metadata.

Clients create files by first allocating data chunks with their data and then creating or updating the file in the file server with the new chunk list. Clients are responsible for the allocation of data chunks on their own. However, the file server may refuse to accept a file if there has been implemented a policy that refuses the file's storage chunk list.

A set of of key-value pairs, the *attributes* is associated with each file. Attributes are inherited from parent directories so that large hierarchies can be conveniently managed. POSIX attributes (like access times) are encapsulated in those attributes. These attributes may be used privately by clients or they may be given special semantics by extensions to VOFS.

An example of a standardized attribute is the "pool" attribute. This attribute holds a list of storage providers available so that clients know where to allocate storage chunks for file data for this filesystem (or a subset of it, since "pool" as an attribute can be independently be set for any file).

Clients may cache the attributes of files and keep an additional local set of attributes. The local attributes can be used by users or applications to configure the behaviour of the client.

For example, the contents of the *forward* local attribute for each file is automatically attached by the client to every request that is made to the file's server. Client- and server-side extensions may use this mechanism to exchange credentials without any client modifications.

To make VOFS features available over POSIX file system calls, VOFS implements *virtual files*. Each path in VOFS may have an additional path component separated by a `@` character:

```
/normal/path@virtual
```

The virtual path refers to a virtual file that is specific the normal path. For example, `file@data` displays what storage provider hosts the data for the specific file.

## 7.2 Links across file systems

The most important virtual file (actually a directory) is `directory/@/`, which represents the whole VOFS network as a directory. This is a global virtual file and can be accessed from any directory. Users can chdir in a peer address there to browse that peer's files:

```
ls -l @/peer:fs/
```

or link documents across filesystems:

```
ln -s @/ICCS:fs/docs/profile.pdf profiles/ICCS.pdf
```

## 7.3 Support for applications

Traditionally, the file system interface is used by both users and applications. This has been preserved in VOFS. In addition to all that is available to users, applications may subscribe to file servers for events happening to files. Applications may also trigger a message-passing `PUBLISH` event that can be subscribed to and used for group communication based on a file as a publishing point. The list of subscribers to this publishing point file is a special attribute and is cached by clients, limiting the disruption from the disconnection of its file server.

## 7.4 Disconnected Operation

Clients keep a local cache with copies of files and storage chunks. User requests are served from this cache to reduce latency and overcome connectivity problems.

Disconnection is expected by VOFS and users can actually force it. Disconnectedness is implied by communications timing out and the state is maintained per host. Users and applications may force files in and out of the state of disconnectedness.

Local changes to a client are immediately stored in the local cache. The local cache is synchronised periodically or when required by the user or a system operation. VOFS does not guarrantee any consistency as the last request to be served by the file server will supersede all previous ones.

## 7.5 Flexible Access Control

Access control in VOFS is based on filtering access requests. Each request has an origin and a recipient peer, a target resource and a specific access that is requested. Based on this design, access control may be implemented as simple as setting a password for files (or whole hierarchies), or as complex as filtering the requests through a sophisticated external security infrastructure. Whatever the access control method employed, users just have to place their credentials in their `forward` virtual files, as described earlier.

# 8 Conclusion – Future directions

The VOFS prototype provides a simple tool for collaborating users to create shared workspace and share their files. The prototype is far from complete product, but im-

plements all functionality needed to prove that if effort is invested it will be usable and useful.

Integration with the user's environment was always a concern — this is why POSIX was selected as a standard interface. But the setting VOFS addresses is quite fluid. Since the conception and design of VOFS, various trends have matured and emerged in dominance, such as *cloud computing*, and new technologies are being developed by the industry, such as the *Google Wave* and a new generation of lightweight web-centered operating systems.

We find that the changing landscape does not make VOFS obsolete. Instead, it justifies its design and we expect that it would thrive in the future, where clouds will make it easy for users to run their VOFS peers. For more about the applicability of the VOFS design in the future refer to Deliverable 6.8

Level of confidentiality and dissemination

By default, each document created within Grid4All is © Grid4All Consortium Members and should be considered confidential. Corresponding legal mentions are included in the document templates and should not be removed, unless a more restricted copyright applies (e.g. at subproject level, organisation level etc.).

In the Grid4All Description of Work (DoW), and in the future yearly updates of the 18-months implementation plan, all deliverables listed in Section 7.7 have a specific dissemination level. This dissemination level shall be mentioned in the document (a specific section for this is included in the template, both on the cover page and in the footer of each page).

The dissemination level can be defined for each document using one of the following codes:

**PU** = Public.
**PP** = Restricted to other programme participants (including the EC services).
**RE** = Restricted to a group specified by the Consortium (including the EC services).
**CO** = Confidential, only for members of the Consortium (including the EC services).
**INT** = Internal, only for members of the Consortium (excluding the EC services).

This level typically applies to internal working documents, meeting minutes etc., and cannot be used for contractual project deliverables.

It is possible to create later a public version of (part of) a restricted document, under the condition that the owners of the restricted document agree collectively in writing to release this public version. In this case, a new document code should be given so as to distinguish between the different versions.