Project no. 034567

Grid4All

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

Deliverable 6.7_VOFS: A peer-to-peer file system for people and applications.

Due date of deliverable: July, 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

Collaborative file sharing is a significant part of the workflows of Internet users and it should be available as a standard function to users. VOFS is a peer-to-peer file system that aims to provide simple file sharing for casual group collaboration. A user group is able to create shared workspaces spontaneously or in a more organised way over the long term. Users can quickly populate the shared workspaces with their own files or files from other users by linking them in. Workspaces also allow users to contribute to a storage pool where new files can be stored. VOFS keeps users independent from third parties by treating service providers as peers to the users rather than environment administrators. All sharing functions are available both interactively to users and programmatically to applications. VOFS conveniently caches files locally to be accessed while disconnected, and remains always responsive.

2 Introduction

We can distinguish two general utilities that dominate the everyday use of personal computers: the *file system* and the *Web*.[1] The file system is a traditional utility, long since taken for granted. The Web is a rapidly evolving place full of information, products, services and communities. People and machines at the edges of the internet are managing information between their file systems and the Web: downloading software, sharing documents and media, or running distributed applications.

Users are increasingly relying on the Web for their every day tasks and subsequently are depending more on third parties. We argue that at least for file sharing this is an unnecessary compromise. To avoid that dependency the ubiquitous traditional file system has to evolve and address the needs of modern users for simple file sharing.

We observe that the casual file sharing problem is not yet solved even in the Web. Users still use e-mail to exchange or share files or upload files to some kind of online repository. This activity either involves depending on a third party or expertly maintaining one's own infrastructure. Since a traditional local file system is always involved in the process, it provides a good candidate to incorporate a solution.

Our goal is to provide a flexible basic set of primitives for collecting and sharing one's own and foreign documents. To that end, we designed VOFS, a peer-to-peer file system that extends the traditional file system to answer the challenge.

VOFS does not focus on technologies for peer-to-peer storage or transfer. It assumes those technological solutions and links them together and brings them to the familiar file system environment. Users do not have to switch environments in order to utilise different technologies.

Even though we promote user independence from third parties, this is not against third parties. Third party services are essential to users but our approach ensures that

[1]The term *Web* is used for the World-Wide-Web and all other communication utilities such as e-mail, messengers and online collaboration tools, and generally the collective user interface of the Internet.

service providers do not control the environment more than necessary. In VOFS, a service provider would have to participate as a peer to other users.

3 Principles for a modern file system

We present the essential princples for designing VOFS as a modern version of the file system.

3.1 Personal Communications, Publishing and Reachability

E-mail is the simplest and most reliable way to reach a person on the Internet, and it is the standard means of communication among groups for collaboration. It is also the most common means to casually exchange documents outside controlled workflows (that use content management systems or revision control systems). As a technology, it is old, awkward, yet remains the most convenient choice.

The reason is that users need a simple communication endpoint that is strictly personal in management and control and is globally accessible. Taking this principle further, to document publishing, users will use and appreciate a simple personal utility that will host their published content (in the form of documents), and allow them access to the content of others.

Every user should be entitled to their own filesystem server as a standard service like e-mail, that will be always on the internet, communicating on behalf of the user. Filesystems should be easily created like e-mail accounts and files must be easily be published like sending messages. Modern developments and trends, such as *cloud computing* make it increasingly easier for users to maintain software agents on the internet, free from administrative concerns.

Personal publishing has two flavours, public publishing and publishing documents to specific users. Making documents available is more complex than sending messages in that the user's choices about access control continue to affect them or others past their initial action for publishing.

3.2 Collecting Documents

A common activity for users is collecting a relevant subset of their documents and documents of others in a single place to share them and use them in collaboration with others. For example, a worker may want to collect introductory material for a new co-worker, or create a collection of pictures to share with friends after a holiday trip.

These collections should be easy to create and dismiss and should not in any way interfere with each other or the repositories they derive from. These collections may serve transient needs, but some of them will be evolved into well-structured repositories and become reference points. This evolution should happen naturally through the usage of the system and not require any technical provisions or administration.

3.3 Manage Document Storage

The user needs to be aware and decide where his documents are stored so that he can make judgements regarding their safety and availability. For example, one will want to use resources from many machines, keep critical documents in more than one machines, or keep the frequently used ones in a more available machine.

However, a user not interested in this management is not burdened with such choices or concerns as the user's file server can also provide storage for its documents.

3.4 Work Offline

Although Internet connectivity can be taken for granted in many places, it is often unexpectedly unavailable, or even undesirable. Local resources should remain accessible regardless their associations across the network and users should not suffer from loss of connectivity more than necessary.

3.5 Maintain a Programmatic Interface for Applications

A unique feature of the filesystem utility is that it interfaces extensively with both users and applications. This provides great flexibility for the users in that they don't surrender the management of their documents to the applications and in that they can extract their work for use beyond application control, such as sharing or importing in other applications.

This feature is preserved in the modernised version of the filesystem, so that applications have access to the new primitives. This way, old and new technologies can hook into the basic workflows of users and provide tools for specific or more complex activities. Deliverable 6.7_Telex describes a library for creating collaborative applications that can use VOFS.

4 Overview of VOFS architecture

4.1 The VOFS network

VOFS is fundamentally a network of peers which are identified by a unique cryptographic 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.

4.2 Creating files

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, in similarly to file metadata. VOFS can be extended through storage plug-ins to handle other protocols for storage than its own (FTP, for example).

Clients create files by first allocating data chunks with their data and then writing creating or updating the file in the file server with the new storage 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.

4.3 File attributes

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. File metadata (e.g. 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).

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.

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.

4.4 Cache and 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. Clients may cache the attributes of files and keep an additional local set of attributes.

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.

4.5 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.

4.6 Interface through Virtual Files

To make VOFS features available over traditional 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.

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
```

5 An example scenario

Assume that three persons from three institutes, ICCS, KTH, INRIA, collaborate in the context of Grid4All, and that each maintains their institution's profile document which needs to be shared among all. The result of their collaboration is a profile document for the whole Grid4All group.

Each collaborator has their own VOFS peer and for simplicity, that they all maintain the same file hierarchy: `profile/[institution].txt`.

First, they launch a new VOFS peer named `G4A` to facilitate a shared workspace. This peer may be set up to use the security infrastructure of a Grid4All VO in order to allow access to users. The workspace in `G4A` has a directory structure:

```
profile/contribs/ICCS.txt
profile/contribs/KTH.txt
profile/contribs/INRIA.txt
profile/G4A.txt
```

The contributions in the `contribs` directory are symbolic links to the respective instition's peer, where the file is actually served. The `G4A.txt` is the result of the collaboration and is hosted in the `G4A` peer.

Each collaborator has access to all contributions through the workspace. Files are locally cached so network disconnections don't affect them besides leaving them with older versions of the other's contributions. Even if the workspace peer is offline, they can still access each other because the workspace is cached and they can follow the cached links to the fresh contributions.

`G4A.txt` is a public document and access can be enabled for all specifically for this file. The contributions are private among Grid4All so the `contribs` directory is protected. Each of the users should also protect their contributions requiring Grid4All credentials if they want to comply with the agreed policy.

Note a significant detail about VOFS: federating the contributed files in the workspace does not deprive collaborators control of their files, as they can freely edit or allow access to their contributions at will. However, what files are linked in the workspace is in total control of the `G4A` peer and despite users having total control on their resources, they can never name them Grid4All resources by themselves.

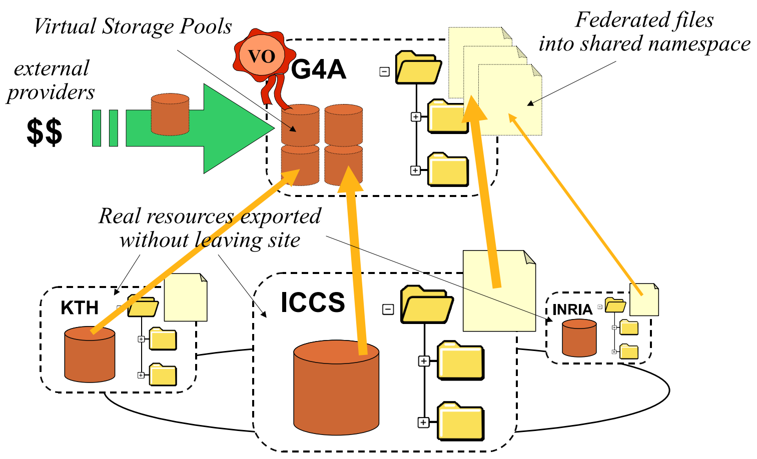
Virtual Storage Pools
external providers
$$
VO
G4A
Federated files into shared namespace
Real resources exported without leaving site
KTH
ICCS
INRIA

Figure 1: A typical workspace setup using VOFS.

6 VOFS and similar systems

The main features of VOFS as described are:

- Integration with the OS
- No expertise for setup or maintainance
- Independence from third parties and priority to one's own resources
- Simple sharing with peer-to-peer exploration, linking and federation
- Responsiveness and disconnected operation

- Storage management with pools
- Support for applications

Traditional peer-to-peer file systems are not relevant to discuss because their goal is purely technical: to provide a distributed image of a file system on a peer-to-peer network. Such technology does not compete with VOFS and can actually be used to support its storage or communications layer.

Collaboration products for enterprises are also out of scope since they require professional installation and maintainance, and are complex and difficult to use.

Most relevant to VOFS is an emerging class of products in the Web, that mainly address the need for personal online storage. An indicative list is Dropbox (getdropbox.com), box.net, wuala.com, wizehive.com, zumodrive.com, Skylive (skydrive.live.com).

All of them support integration with the OS and are available via the web, except Wizehive which is purely web-based. All of them provide server-based storage in centrally owned resources. Wuala employs a supplemental peer-to-peer storage for performance optimisation, and Zumodrive uses *amazon S3* in addition to their servers.

The basic primitive of sharing for these systems is providing the file owner with a link to their files that he can share with others. Wuala promotes a social networking approach to file sharing allowing users to publish their files in centrally organised indices. Box.net and Wizehive offer additional collaboration applications, such as discussions and task organisers.

Zumodrive features disconnected operation similar to VOFS.

None of the systems, however, allows the user to create a symbolic link to a remote file. VOFS uses peer-to-peer linking as a simple and effective means for exploration and offers a unique feature for federation of files.

Moreover, none of the mentioned systems is independent from central, third-party servers. In contrast, the VOFS design requires service providers to be peers with users. This means that users are able to use services without allowing service providers to control anything more in the collaborative environment.