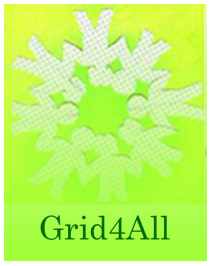

Project no. 034567

Grid4All

Specific Targeted Research Project (STREP)

Thematic Priority 2: Information Society Technologies

D4.3 Prototype API usage by Grid4All based applications

Due date of deliverable: 7st July 2008

Actual submission date: 17st July 2008

Start date of project: 1 June 2006 Duration: 30 months

Organisation name of lead contractor for this deliverable: UPC

Contributors: Grid4All Consortium

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)	X
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

Abbreviations and acronyms used in this document

Abbreviation/Acronym	Description
ACL	Access-Control List
API	Application Programmer Interface
CFS	Collaborative File Sharing
CNSE	Collaborative Network Simulation Environment
DCMS	Distributed Component Management System
DFS	Distributed File Service
EM	eMeeting application
FMS	Flash Media Server
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
RMI	Remote Method Invocation
SC	Shared Calendar
SIS	Semantic Information Service
SSO	Single Sign-On (authentication)
SWF	Shockwave Format for multimedia content
URN	Universal Resource Name
VBS	Virtual Block Store
VO	Virtual Organisation
VOFS	Virtual Organization oriented File System
VOMS	Virtual Organization Management System
WSRF	Web Services Resource Framework

Grid4All list of participants

Role	Participant N°	Participant name	Participant 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	SE
CR	4	Swedish Institute of Computer Science	SICS	SE
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

The following persons contributed to this deliverable:

Alicia Bou, Carles Maggi (Antares); François Berenguer (INRIA); Carlos Alario, Miguel Bote, Eduardo Gómez, Joan Manuel Marquès, Leandro Navarro, Zenon Perisé, Miguel Valero (UPC).

Leandro Navarro was editor, and Ruby Krishnaswamy (FT) was peer reviewer.

1 Executive summary

This is a deliverable part of the research project Grid4All (IST-FP6-034567). This document reports the work done in T4.2 until M24 in mapping end-user applications to the Grid4All API. It presents how the applications are developed in WP4 plan to exercise the infrastructure API, going from user requirements to a mapping, application by application, to the API functions provided by the infrastructure and a discussion on the changes, opportunities and advantages that can be obtained as a result at the end of the integration process, that will be reported in D4.5.

2 Introduction

This document reflects the work done in subtask T4.2: “Application design in Grid4All environment” (M6-M24)

> “Define and develop prototype an application API and programming models: mapping from application concepts, modules and calls to the infrastructure, may require redesigning and developing parts of existing or new applications.”

It describes how a collection of end-user applications are being developed to use in different ways the API provided by the modules developed within the project. These applications provide support for different types of end-user tasks (supporting groups [CNSE, CFS], group awareness [CNSE], sharing [CFS, CNSE, eMeeting], synchronous (same time) meetings [eMeeting], coordination [SC, CFS], and specific tasks [CNSE, gMovie]) relevant in the educational scenario identified in D4.6 and applicable to the wider range of scenarios of interest for the project. They are developed using diverse software technologies and development environments (e.g. an extension of the Mozilla browser framework for CFS, Globus for CNSE, Flash for eMeeting). They make use of different aspects of the Grid4All API, covering directly or indirectly most of the functions. This document illustrates how the applications have developed a plan (as integration tasks will be mostly performed in the following semester) to exercise the API to help achieve their specific user requirements, and improve the capability (functionally or non-functionally) of the applications, albeit both applications and API are not yet finalized and therefore the final details are not yet ready but they will be documented later in deliverable D4.5.

Section 3 illustrates the actual or intended mapping between each application and the Grid4All API. The following sections describe the applications in general and each in detail particularly in respect how they have been redesigned and modified to use this new infrastructure. Finally there are conclusions summarizing the lessons learnt from this process.

3 Applications

The following sections describe the applications that are being developed within the Grid4All project. For each application there is:

- Brief description,
- Relationship to end-user requirements,
- Application-specific issues in the mapping to the Grid4All API, including changes on the application to comply with it,
- Architecture of the application reflecting that mapping,
- New potential features and advantages as a result of the integration with Grid4all,
- Initial evaluation from the developer point of view (architectural perspective) on the usage of grid4all infrastructure for this application,
- and finally, a table summarizing the intended use of the Grid4All API by that application.

The following table summarizes the main characteristics of how applications are organized and can be used:

App	Instantiation	Install	Deploy	Use
CFS	One per user	Download and install an extension of the Firefox browser	Assumes/requires Grid4All infrastructure services (such as VBS+DFS for storage, security)	Opening a URN (creating or joining a shared folder) in Firefox+CFS extension
CNSE	One per student, one per teacher, several (centralized) services	Download and install a client tool (local Java application) for each user (teacher, student) to access the service	Requires deployment of CNSE services Assumes/requires Grid4All infrastructure services (such as storage, scheduling, security)	Teacher creates and configures the learning activity Each student starts a local Java application and joins a specific activity
EM	One per user, centralized server	Installation is loaded on-demand (dynamically loaded) by connecting to the eMeeting website	Requires a working eMeeting server Assumes/requires Grid4All infrastructure services	The initiator creates a meeting, the participants joins a meeting. For both, participation consists on visiting a web page, logging in and the application will be deployed on demand (Flash and Java based)
GMovie	Centralized server	Connect to the gMovie website	Assumes/requires Grid4All infrastructure services (such as VBS+DFS for storage, security)	Use the web based application
SC	One per user	Download and install a Java client	Assumes/requires Grid4All infrastructure services (such as VBS+DFS for storage, security)	Use the client application

The relationship between end-user requirements presented in D4.7, the Grid4All API and the end-user applications is illustrated by the following two tables. The first table presents the relationship between operational and functional requirements. Although many other relations can be considered, the gray elements indicate which operational requirements are considered most relevant (selected and described in D4.7) for each functional requirement. These relations or associations of functional and operational requirements are a particular characteristic of the project as the goal is to provide certain functionality for a number of scenarios under environmental factors for the target underlying computational infrastructure characterised by scale, churn, dynamics, heterogeneity and volatility. Therefore, considering certain functional requirements for the target scenario(s) under the environmental conditions considered in this project lead immediately to consider several of the operational requirements as the next table shows.

OPERATIONAL REQUIREMENTS	R.O.1 Performance	R.O.2 Scalability	R.O.3 Manageabilit	R.O.4 Dependabilit	R.O.5 Usability	R.O.6 Openness	R.O.7 Accessibility	R.O.8 Autonomy (o	R.O.9 Secure acce
			R.O.3.1 Manageme	R.O.4.1 Environme	R.O.5.1 Environme				
				R.O.4.1.1 Stability o					
FUNCTIONAL REQUIREMENTS				R.O.4.1.2 Executior					
R.F.1: Organizational									
R.F.1.1: End users are part of organizations (schools, universities), a									
R.F.1.1.1 Creation and set-up of virtual organisation					■		■	■	■
R.F.1.1.2 Lifetime of virtual organisation				■	■		■		
R.F.1.2: Individuals or organizations must be able to freely contribute									
R.F.1.2.1 Contribution of resources (computational or storage)				■	■		■	■	
R.F.1.2.2 Allocation of computational and storage resources			■		■				
R.F.2: Group									
R.F.2.1: Units of activity must be isolated from each other (among									
R.F.2.1.1 Access policy configuration				■	■				
R.F.3: Communication									
R.F.3.1 Awareness of related activity (event notification)					■				
R.F.4: Coordination									
R.F.5: Sharing									
R.F.5.1: People part of an organization or performing an activity									
R.F.5.1.1 Data services for remote mode (disconnected mode)				■	■				
R.F.6: Computation									
R.F.6.1 Running (scheduling) of tasks	■			■	■		■		
R.F.7: Task									
R.F.7.1: Group work with remote participants require supporting									
R.F.7.1.1 Appointment of meetings	■	■		■	■				
R.F.7.1.2 Creation of shared workspaces	■	■		■	■				
R.F.7.1.3 Sharing files	■	■		■	■				
R.F.7.2: Group work requires in some situations face-to-face, same-									
R.F.7.2.1 Remote multimedia conferencing	■	■		■	■				■
R.F.7.3: Task-specific activities such as those related to learning con									
R.F.8: Environmental									
R.F.8.1 Deployment of applications and task manager									
R.F.8.2 Awareness of environmental services					■	■			■
R.F.9: Development									
R.F.9.1 Development of manageable services and applications			■		■				
R.F.9.2 Continuous maintenance of applications					■	■			
R.F.9.3 Selective transparency				■	■				
R.F.9.4 Facilities for the development of collaborative applications					■				
R.F.9.5 Design and implementation of new market mechanism					■	■			
R.F.9.6 Deployment of applications and task management				■	■				
R.F.9.7 Scheduling of tasks					■				
R.F.9.8 Event notifications	■			■	■				

Mapping between operational and functional requirements (obtained from D4.7)

The following table shows a mapping between requirements (D4.7) and Grid4All modules including applications (relations: minor in gray, major in black), showing how the coverage of the requirements is the concern of different modules and applications across the project (each module covers multiple requirements, each requirement is supported by multiple modules or applications, most requirements are covered at least in some degree by the project work).

	VOMS	Inter-VO	Security	DCMS	Data	Executio	Informat	Applications:				
OPERATIONAL REQUIREMENTS								CFS	CNSE	eMeetin	gMovie	SC
R.O.1 Performance (or responsiveness)												
R.O.2 Scalability												
R.O.3 Manageability												
R.O.3.1 Management of apps												
R.O.4 Dependability (or non-disruption,												
R.O.4.1: The CSCL environment should be stable,												
R.O.4.1.1 Stability of the environment												
R.O.4.1.2 Execution of applications												
R.O.5 Usability												
R.O.5.1: The CSCL environment should be simple to												
R.O.6 Openness												
R.O.7 Accessibility												
R.O.8 Autonomy (or control)												
R.O.9 Secure access												
FUNCTIONAL REQUIREMENTS												
R.F.1: Organizational												
R.F.1.1: End users are part of organizations												
R.F.1.1.1 Creation and set-up of virtual organisa												
R.F.1.1.2 Lifetime of virtual organisation												
R.F.1.2: Individuals or organizations must be able to												
R.F.1.2.1 Contribution of resources (computatio												
R.F.1.2.2 Allocation of computational and storag												
R.F.2: Group												
R.F.2.1: Units of activity must be isolated from each												
R.F.2.1.1 Access policy configuration												
R.F.3: Communication												
R.F.3.1 Awareness of related activity (event												
R.F.4: Coordination												
R.F.5: Sharing												
R.F.5.1: People part of an organization or performing												
R.F.5.1.1 Data services for remote mode (disco												
R.F.6: Computation												
R.F.6.1 Running (scheduling) of tasks												
R.F.7: Task												
R.F.7.1: Group work with remote participants require												
R.F.7.1.1 Appointment of meetings												
R.F.7.1.2 Creation of shared workspaces												
R.F.7.1.3 Sharing files												
R.F.7.2: Group work requires in some situations face-												
R.F.7.2.1 Remote multimedia conferencing												
R.F.7.3: Task-specific activities such as those												
R.F.8: Environmental												
R.F.8.1 Deployment of applications and task												
R.F.8.2 Awareness of environmental services												
R.F.9: Development												
R.F.9.1 Development of manageable services and												
R.F.9.2 Continuous maintenance of applications												
R.F.9.3 Selective transparency												
R.F.9.4 Facilities for the development of												
R.F.9.5 Design and implementation of new market												
R.F.9.6 Deployment of applications and task												
R.F.9.7 Scheduling of tasks												
R.F.9.8 Event notifications												

These tables are helpful for implementers of API functions and applications to be able to realize what parts, and to what extent, are important with respect to the project requirements, and identify how best to coordinate support between developers and prioritize work during the integration phase. As work progresses, these tables will have to be revisited and updated to reflect the work being done actually.

4 CFS

The Collaborative File Sharing (CFS) application allows users to collaborate, interact and share information. The collaboration is asynchronous. CFS exports notions of files and forums. Workspaces may contain forums, files and other workspaces thus allowing a hierarchical organization. Users may access an existing item or create a new one. Access to items may be restricted by the creator. In addition, a notion of roles is carried. All users that can access an item have a role that determines what the user can do and this role may differ from item to item. This role may be changed at any time by the creator or the item administrators. Interaction and collaboration between participants may be either through messages posted in a forum or through file versions uploaded in a file.

4.1 User Requirements

The CFS application has some specific and detailed requirements, mainly coming from end-users and they are both functional and non-functional. They are discussed as follows:

In terms of operational requirements:

- **Usability** (R.O.5):
    - Ease of install and configure: The application might be easy to install and configure, by only copying files and updating some items on a configuration file.
    - Usability: The GUI of the application might be easy to use and similar to other already existing in collaborative environments such as BSCW.
    - Adaptability: The application might be easy to configure in a default behaviour but with powerful options to modify it.
- **Secure Access** (R.O.5): access to shared files should be restricted to the members of a certain group or work space.

Other operational requirements:

- In minor degree, ***Dependability*** *(R.O.4)*, in the need to ensure the execution of the application despite environmental factors (failures of other clients or disruptions of the storage service), ***Openness*** *(R.O.6)* regarding extensibility, ***Accessibility*** *(R.O.7)* as the use of VO storage services and resources reduces the cost of the process.

In terms of functional requirements:

- **Group** (R.F.2):
    - Multiple participants: The system allows multiple participants sharing and modifying the same set of data.
- **Sharing** (R.F.5):
    - Concurrent access: Participants may be able to concurrently open, read, modify, create and delete items on CFS.
    - Conflict detection / management / solving: Conflicts may appear when concurrent activity is performed. The system may detect and solve them automatically in some cases. In other cases, conflicts may be solved by participants.
    - Optimistic Collaboration: Users may have a different view of the shared workspace, but all views eventually converge.
    - Off-line collaboration: CFS may be used in off-line mode. The state of all replicas will eventually converge. In case of conflicting actions, the conflict management mechanisms must guarantee that the same resolution is applied in all instances
    - Data availability: Data from versions and content of messages is available for all nodes.
- **Task** (R.F.7):
    - Awareness: Participants might be able to view the list of actions (events) performed to each item.
    - Requirements related to forums:
        - Participants might be able to open and read messages.

        - Participants might be able to post a new message, including attached files
        - Participants might be able to download to their computer attached files.
    - Requirements related to files:
        - Participants might be able to open and download versions already uploaded.
        - Participant might be able to upload a new version to a file.
        - A conflict within a file appears when two or more versions are uploaded concurrently. Participants may decide which one is more representative. Discarded versions may still be available.
    - Voting: Participants might be able to rate items in order to give them some relevance.

4.2 Relation to Grid4All Infrastructure

The CFS application uses mainly two components of the Grid4All infrastructure: Telex and DFS. The application is installed in the user's machine and it uses the Grid4All infrastructure available for that machine.

Telex is used for managing the metadata of the system. Actions represent the CFS operations and the Constraints allow the application to specify the semantics between actions. Telex proposes a set of schedules of actions that satisfy the given constraints, i.e. sound schedules. The given constraints ensure that some conflicts will be automatically solved. For some others, the conflicts are solved by selecting a schedule and discarding the rest. The application will provide a procedure for selecting the most appropriate schedule

DFS is used for storing the content of file versions and messages. DFS implements a FUSE interface and it can be accessed using its POSIX interface. DFS ensures that data is eventually available for all nodes.

4.3 Architecture

The architecture of CFS can be view as a 2-layers pattern: an internal layer and different user interfaces. The internal layer provides the core functionalities and uses the Grid4All functionalities. It offers all functionalities to an upper layer that presents the information to the user in different formats. In figure 1 we can see how the internal layer interacts with the Grid4All infrastructure and how interfaces connect to that layer.

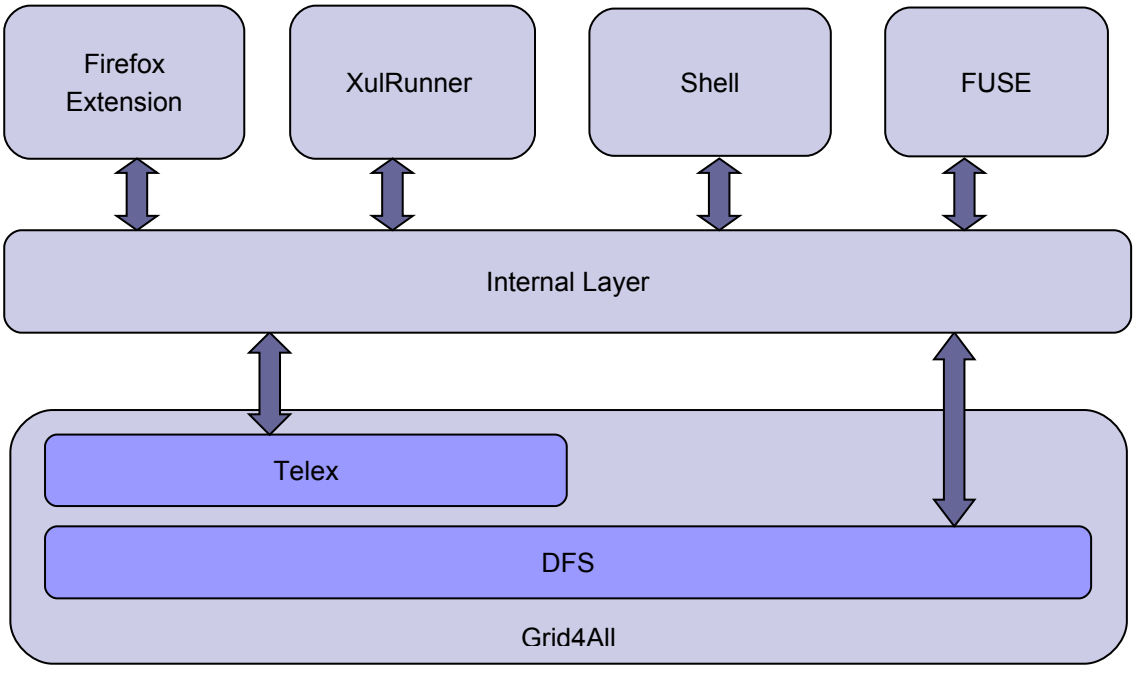

Figure 1. The architecture of CFS

The interfaces are:

- *Firefox Extension*: Provides an interface integrated in Firefox as an extension. It is coded in XUL language with connections to Java.
- *XulRunner*: It is presented as an stand-alone application with the xulRunner application. The xulRunner application is an engine that runs XUL applications out of the environment of Firefox.
- *Shell*: Provides a command-line interface used mainly for testing purposes.
- *FUSE*: Enables CFS to be used as a regular file system by regular applications. This is currently being developed.

In figure 2 we can see the integration with the Grid4All middleware. The users access a File or a Forum. The metadata of those items, for instance the *id* of the messages in a Forum and the relations among them, are stored in Telex. The content of the message is stored in DFS and it is only loaded when the user opens the message. This separation of data and metadata speeds up the displaying of the forum because there is no need of loading all messages in order to get their headers. In addition, by storing messages separately in DFS, the Telex system is not overloaded and keeps a good response time. For files and versions, the procedure is similar.

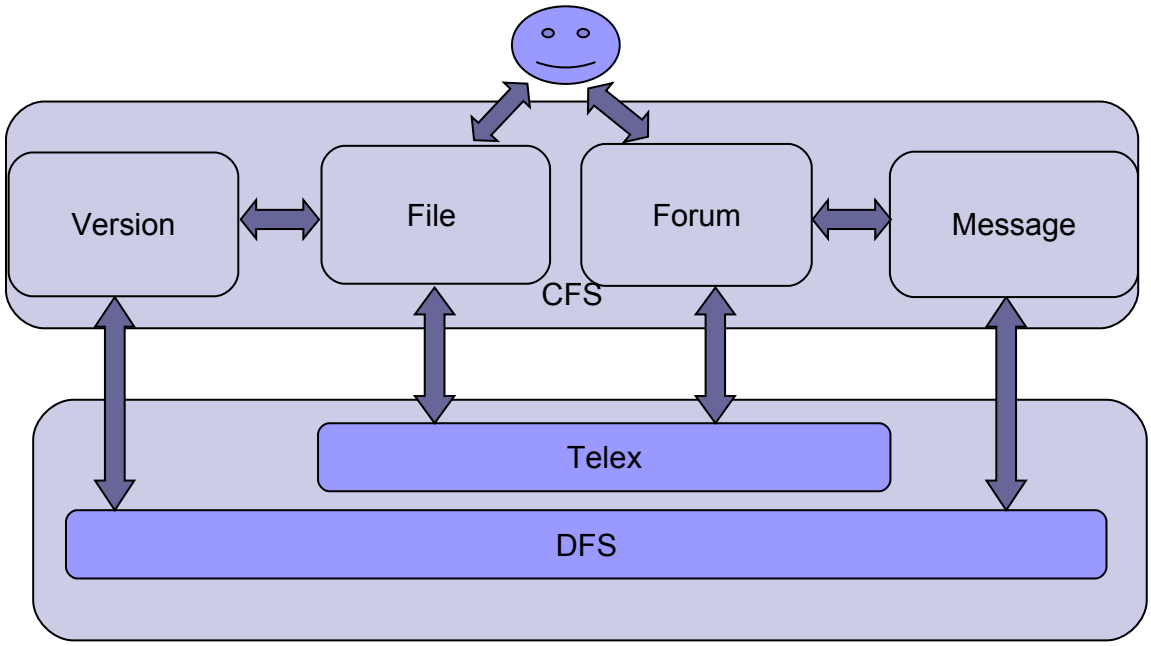

Fig 2. Integration with the Grid4All middleware

4.4 Integration

The CFS application has been designed and implemented from scratch to be used in the Grid4All infrastructure. It uses:

- Telex
- DFS
- Membership Service

CFS uses Telex for managing metadata. The application includes Telex as a library. Telex stores all actions received (either locally generated or remote) in a local multi-log. Telex proposes to the application a set of sound schedules of actions. The application may choose one of them and may execute it. All activity

generated in CFS is translated into Telex actions. Some constraints are also added to Telex in order to keep semantics and get correct sound schedules.

CFS stores the content of versions and messages in DFS. DFS provides a POSIX interface since DFS is enabled as a file system by implementing FUSE interface. The application asks to the DFS for the data of the file versions and messages, leaving metadata to be managed by Telex.

The Membership Service is used by CFS for authentication of users. After authenticating a user, an application specific policy is applied.

The following table summarizes the intended usage of Grid4All modules and services by the CFS application (L/P: intended Level of Development/Priority[1]).

Domain	Modules & Services	Functionality	L/P	Description
VO Management: administrate a	Membership Manager	Authenticate members	1, 2	Authenticate users and VO Members
Security: manage security	Identity manager(VOMS)	Sign-on and create a proxy certificate with attributes	1, 1	Checking identity
	Policy Administration Point, Policy repository, Policy editor	Create/edit/store VO policies by VO admin	1, 1	Defining access policies
	Policy Enforcement Point , Policy Decision Point, Policy Information Point	Access to a VO resource(s) protected with a PEP(s)	1, 1	Checking permissions
Collaborative & Federative services: manage VO-wide file systems; provide support for collaborative editing of shared documents	Telex	Create/read/write/delete collaborative documents	5, 5	Telex based application
	VOFS Manager, User Agent	Maintain a VO-wide workspace		
	VBS+DFS File Service	Serve user files	5, 5	Access to distributed file storage

1 "Level of development" describes the intended objective for integration from 0 (lowest) to 5 (highest) being: 0 Out of scope; 1 Discuss (informally, relation to app); 2 Design, but not implement; 3 Minimal implementation; 4 Detailed implementation; 5 (4) + Evaluation.

"Priority" expresses the importance for the application developer: 0 Not considered; 1 If time left; 2 Good if included; 3 Should include; 4 Definitely include; 5 Essential.

5 Collaborative Network Simulator Environment (CNSE)

The CNSE is an application that aims at enabling the support of collaborative learning scenarios within the context of networking education. More specifically, CNSE allows students to carry out simulations of network scenarios described in simulation scripts. Such simulations generate trace files that are employed by CNSE to generate animations of the behaviour of the different elements of the scenario that can be collaboratively visualized by students. Interestingly, CNSE can be employed to carry out parameter-sweep simulations (i.e. simulations of the same network scenario varying the value of one or more parameters according to the instructions provided by the user). In many cases parameter-sweep simulations require a great computational power in order to finish simulations within a reasonable period of time as well as a large amount of storage resources to store the results of all simulations.

This need of computational and storage resources motivated the design and implementation of CNSE as a grid service-based application. This enables the deployment of CNSE within the context of service-oriented grids so that the computational and storage resources shared by different organizations in the grid can be employed by CNSE in order to perform parameter-sweep simulations.

In the context of Grid4All, we conceive a deployment plan where the CNSE executes on a dynamic and democratic virtual organisation. Here, computational and storage resources are typically contributed by the participants (teachers, students etc). The VO encompasses both the members and the resources that these members bring in to the organisation. As part of networking lessons conducted in this VO, the students are expected to prepare scripts and execute network simulations.

This deployment scenario is different from the precedent. In this scenario, the CNSE executes in an unstable and dynamic environment and uses idle Internet resources much in the way of desktop computing. This democratisation has advantages such as the lower to null resource costs and also disadvantages such as higher management costs. Even in this environment, the basic requirements need to be satisfied. Furthermore we need to ensure that using Grid4All infrastructure and functionality we are able to meet the management challenges.

5.1 Current application architecture

The current CNSE prototype has been designed and developed following a service oriented architecture and using WSRF compliant services. The design of any service-oriented application implies splitting up its functionality in a set of different services. The services that need to be composed to offer the CNSE can be seen in figure 3 and are described below.

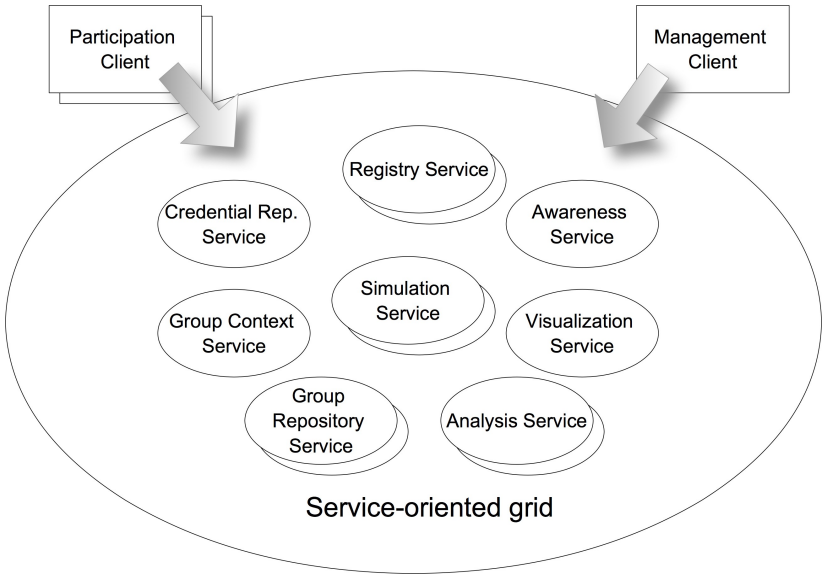

Fig 3. Services and clients of CNSE

CNSE has been developed reusing the well-known *ns2* network simulator and its visualization companion tool *nam*. Though these are not grid applications, their functionality is convenient to fulfil the objectives of the scenario, and thus they are offered in the grid as services. This choice determines somehow the data flow between the different services in CNSE. At this point, it suffices to know that *ns2* must be provided with an input file, which is a script containing the description of the topology to be simulated and some relevant parameters for the simulation, such as binary rate of links, the size of router queues, or the message rate of sources. After the simulation, ns2 produces an output file with the traces of the simulation (it is noteworthy that these files are generally very large). This file must be the input for *nam*, the visualization tool.

The script files and the output files employed by CNSE users are stored for later retrieval in the **group repository service**. This service will offer a group-view of the files stored (students in a group will see only the files of their group). The **simulation service** is in charge of carrying out the simulations of the networking scenarios described in the script files provided by users. The statistical analysis of the data generated by simulations can be carried out by the **analysis service**. The **registry service** is a registry typically employed in service-oriented grids to enable the discovery of the services shared by the participants in a grid. In the case of CNSE, the index service is used to find a shared repository, a simulation service or an analysis service whenever they are needed. The x-y graphs of the measures taken during simulations and the animations of the simulated network scenarios are generated by the **visualization service**. This service also implements the logic that allows sharing the view of a given graph or animation among the members of a group and that enables them to interact with it. The **awareness service** keeps the list of users that are online using the application. Besides, it is responsible for notifying the changes that may happen on the list of online users as well as the main actions made by them (e.g. upload a simulation file, perform a simulation, launch a collaborative visualization).

The use of collaborative services in CNSE such as the shared repository or the visualization service implies that a reference to the shared context of each group of students in each service must be available for the application. In this way, the application can put users in contact with their group mates and with their associated grid resources. Those references, as well as the identifiers of the users that belong to each group, are stored in the **group context service**. Finally, it must be taken into account that the access to services provided in a grid is typically done in a secure way. This is why CNSE, as many other grid-enabled applications, uses a **credential repository service** to store and retrieve valid user credentials that enable secure access to grid services.

The services that make up a grid-aware application must be coordinated so that they are accessed according to the functionality that the user demands from the application in every moment. This coordination can be made by light clients employed by users to interact with the application. In the case of CNSE, two such clients are defined: the **participation client** is conceived to allow learners collaborate using the simulator, while the **management client** allows educators to define the groups of students that will use the application. These clients are also graphically represented in figure 3.

5.2 User requirements

The CNSE application has several user requirements, both coming from the end-users (teachers and students) and from service developers. In terms of operational requirements:

- ***Performance*** *(R.O.1)*: this requirement is posed by the end-user and is **critical**, since the objective to use a grid to run network simulations is to achieve better performance, especially in parameter sweep simulations, so that students can use the results of simulations (e.g. in a discussion with other students of the same class) as soon as it is possible by parallelizing as much as possible the execution of the simulations (ideally, every simulation would be carried out in parallel in a different computer).
- ***Scalability*** *(R.O.2)*: this requirement is posed by the end-user and is two-fold. The application should be scalable in the number of users is a **moderate** requirement, since the number of students in a laboratory is normally bounded to 50 or 60. The number of computational resources that can be used by the application must also scale. This is an **important** requirement, since more nodes are required to run more complex parameter sweep simulations with similar performance.
- ***Manageability*** *(R.O.3)*: this requirement is posed by end-users and system integrators. It is considered to be **important** since educators should not have high technical skills to set up their educational scenarios, and the computational support should not fail under most circumstances, or the laboratory session will be spoiled. Nevertheless, failure does not imply damage to goods or

people and thus it cannot be considered critical.

- ***Dependability*** *(R.O.4)*: this requirement is posed by end-users, but it can be considered as **important** in this educational setting since both students and educators are likely to abandon the use of the application if it is not dependable: if performance is not even or the risk that tasks may not performed on time is significant.
- ***Usability*** *(R.O.5)*: end-users pose that usability is a **critical** requirement for this application, though this mostly concerns the user interface. System integrators also consider that is **important** that this application can be easily deployed and integrated, for which standard interfaces should be used.
- ***Openness*** *(R:O.6)*: at this point, no further functional developments of the application can be foreseen, and thus this requirement is rated as **unimportant**.
- ***Accessibility*** *(R.O.7)*: simulations are heavy tasks, but the distributed system should impose low resource demands on the client machines existing in the school laboratories in terms of the hardware and software required to use CNSE application. This requirement is **moderate**.
- ***Autonomy*** *(R.O.8)*: this requirement is mostly posed by resource providers that may want to join or leave independently of the application state. In this way, will prefer to have freedom to share their resources whenever they want. To engage more resource providers, this requirement should be **important**.
- ***Secure access*** *(R.O.9)*: the end-user does not give much importance to this requirement, but for the resource provider it can be seen as **important** (this may be relaxed if the grid is set up as a federation of just several educational institutions).

In terms of functional requirements:

- ***Organizational*** *(R.F.1 and particularly R.F.1.1)*: Students and the teacher(s) need to work on an organizational context (classes, school or group of schools participating on a project) and this organization has to be taken into account and reflected on the system.
- ***Group*** *(R.F.2 and particularly R.F.2.1)*: Collaborative learning is an activity where people work in groups and these groups shall learn independently from each other, therefore the system must provide isolation mechanisms among them, therefore users have to be checked and a policy for access should be defined *(R.F.2.1.1).*
- ***Communication*** *(R.F.3, and particularly R.F.3.1, R.F.9.8):* Asynchronous communication requires that people can interact at the same time or different times and therefore there must be ways to notify actions and changes (events) to be aware of the collective activity being performed by the rest of the participants on the session.
- ***Coordination*** *(R.F.4):* There is a need to support the synchronization of work in tasks where work has a defined ordering being performed by multiple participants.
- ***Sharing*** *(R.F.5):* People performing group activities such as the collaborative visualization of simulation results should be able to share those views even if they are in different places.
- ***Task*** *(R.F.7, and particularly R.F.7.1, R.F.7.3):* Computer supported group work require specific support for this specific activity of learning networking by simulation such as creating simulation scripts, sequencing the tasks for the group activity, launching group visualization sessions, collecting events about the activity of groups (R.F.7.3).

5.3 Relation to Grid4All infrastructure

The CNSE application can potentially profit from the functionalities offered by some parts of the Grid4All infrastructure, as follows:

- ***VOFS***: The CNSE will need to store and retrieve files containing the simulation scripts and the traces of the simulation (generated after it has been run, and useful for later visualizing the simulation). The trace files can be large. Ideally, these files should be transparently stored and retrieved in the grid, and for that the VOFS could be used.
- ***VOMS***: The services that make up the CNSE application can be run in several Virtual Organizations in the same grid, and in each case it should use and offer resources to each particular VO. For this, a VOMS is required, though the services of the CNSE application should be unaware of it.

- ***SS***: For parameter sweep simulations the CNSE application has to run many jobs (each corresponding to a single simulation for some given parameter values). Thus, the CNSE could benefit from some scheduling that determines the number of simulations that can be carried out in parallel in a given set of machines that can carry out the simulations.
- ***DCMS***: To increase manageability and dependability in volatile grids (grids where resources leave or fail more frequently than usual) it can be beneficial that CNSE services are monitored, replicated and remotely managed. For such scenario, some CNSE services can be partially redesigned to profit from the DCMS.
- ***SIS***: In a wider scenario, where the existence of the CNSE is not known in advance by educators, but where a pool of services exists in an educational grid, educators may need to query about what does exist to support their educational scenario and how can these services be integrated. For such a purpose, a Semantic Information Service could be used, and the application services described in an ontology managed by this SIS.

The current prototype of CNSE is not integrated yet with any of the above parts of Grid4All infrastructure, though some clear benefits could be achieved from this, as discussed later. In this sense, we have identified the main G4A functionality and services that could provide benefits to CNSE and improve its capabilities. The integration of CNSE with DCMS and SIS are currently being worked. More specifically, we are currently focussing on the usage of DCMS to improve the manageability of CNSE services and the SIS to enable discovery of CNSE services.

5.4 Potential application architecture over Grid4All

The architecture above does not consider Grid4All architecture, but because of its coarse-grain division into services it is quite simple to devise which parts would benefit of a specific grid infrastructure, such as that proposed by Grid4All. Figure 4 depicts the overall integration scheme.

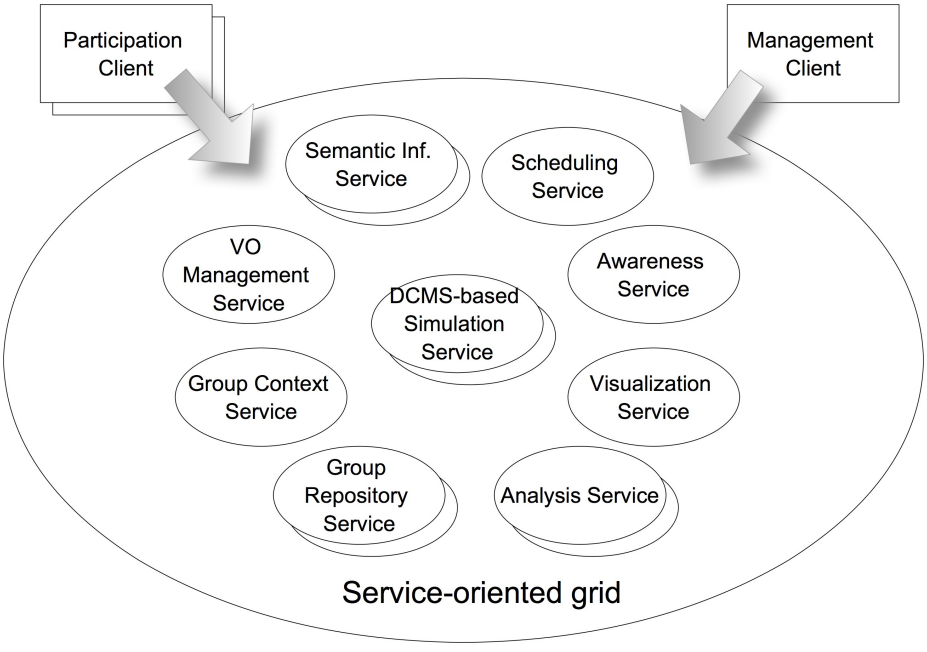

Fig 4. The CNSE architecture and its potential integration points with Grid4All infrastructure.

First, the Group Repository Service allows storing and retrieving files, and could thus be replaced by or developed on top of Grid4All Virtual Organization File System. As in CNSE it is expected that each group of students can only see their files, and not those of other groups of students, it seems more convenient to develop this service on top of the VOFS.

In addition, for parameter sweep simulations the Simulation Service should be split in three parts: the first one decomposes the parameter sweep simulation tasks in many individual jobs (each job is a single simulation), the second one schedules these many individual jobs (i.e. a decision is taken on what resource

will be used to carry out each single simulation), and finally the third one is in charge of carrying out each individual simulation. The second part can be played by Grid4All scheduling service. Moreover, if there are many instances of the simulation service, each one could be monitored to assess its progress or to detect its failure. If necessary, the simulation task could be re-launched in another simulation resource to hide the user any failure in the infrastructure. This could be done by Grid4All Distributed Component Management System that should put watchers attached to each simulation resource, aggregators collecting the watched variables and managers taking decisions.

Finally, the Semantic Information Service could be used to mediate between the management client and the Index Service. Currently, the management client queries the Index Service using given service names. With a SIS, the queries would use instead descriptions of services done according to a given ontology, and a semantic engine would be used to perform the searches.

5.5 Integration issues

The CNSE application has been developed following the service oriented paradigm, which is currently dominant in grid middleware. As such, it can be easily integrated with other software that offers or uses service interfaces, and will require more complex changes in other cases.

The integration with the Semantic Information Service can be easy if the search functionality is offered though a service interface. In such a case, the service could be queried with a string representing a description of the searched service according to the SIS ontology. This query would replace the query to the Index Service. For a richer user experience, the ontology could be used to visually browse and select available services, but this functionality is beyond the CNSE application, since it is much more general.

The integration with the Scheduling Service can be achieved in a similar way to the Semantic Information Service. It could be called though a service interface, with the list of jobs and resources, and then returning an assignment list.

The integration with the DCMS is necessarily more complicated since the design and development paradigms of this infrastructure and the current CNSE prototype are different. Providing the CNSE simulation service with self-* features requires that: the service be wrapped as a component, writing watchers and actuators to interact with that component; or, write the business logic of the simulation as a component, write watchers and actuators to interact with it, and wrap its functionality as a service to offer it to the rest of the application.

The following table presents the intended usage of Grid4All modules and services by the CNSE application (L/P: intended Level of Development/Priority).

Domain	Modules & Services	Functionality	L/P	Description
VO Management: administrate a	Membership Manager	Authenticate members	1, 1	Authenticate users and VO Members
Security: manage security	Identity manager(VOMS)	Sign-on and create a proxy certificate with attributes	1, 1	Checking identity
Collaborative & Federative services: manage VO-wide file systems; provide support for collaborative editing of shared documents	Telex	Create/read/write/delete collaborative documents		
	VOFS Manager, User Agent	Maintain a VO-wide workspace		
	VBS+DFS File Service	Serve user files	1, 1	Replace Shared Repository by this distributed service
Execution Management	Scheduling Service	Interacting with the Scheduling Service	2, 2	Schedule simulation jobs in different simulation services
Overlay Services – DCMS	DCMS	Executing Self-* Component-Based Applications	2, 2	Internal implementation of the simulation service as a set of Fractal components
Information Services	Matching Service	Advertise, Query market/service	2, 1	Look for simulation services using the SIS

6 eMeeting

Communication is key to any form of collaboration and eMeeting is an on-line synchronous collaborative tool that allows sharing not only voice and video but also all forms of documentation. eMeeting also provides tools for group charting and polling.

WP3 provides middleware and tools to construct tools for asynchronous communication applications. The CFS application described in section 5 allows participants within a project to share and edit documents. This rich application platform is a perfect companion to other Grid4All applications. Grid4All itself targets different collaborative tools and styles for Democratic Grids. eMeeting complements the other tools by enabling interactive collaboration styles.

6.1 User Requirements

The eMeeting application has several user requirements, both coming from the end-users (teachers and students or their equivalent in other related scenarios) and from service developers. In terms of operational requirements:

- ***Performance*** *(R.O.1)*: this requirement is posed by the end-user and is **critical**, since the objective is to have synchronous meetings (at the same time): the more immediate the response the better for supporting the task.
- ***Scalability*** *(R.O.2)*: The application should be scalable in the number of users is a **moderate to critical** requirement, as the number of students in a educational activity is normally small (from groups of 2-5 people, up to typically 50 or 60 students per project or class), although there are some particular cases where an activity could involve many more (usually a uni-directional activity: a presentation).
- ***Manageability*** *(R.O.3)*: this requirement is posed by end-users and system integrators. It is considered to be **important** since educators should not have high technical skills to set up, adjust or maintain their educational scenarios, and the computational support should not degrade under most circumstances, or the interactive session will fail.
- ***Dependability*** *(R.O.4)*: this requirement is posed by end-users, but it can be considered as **important** in this educational setting since both students and educators are likely to abandon the use of the application if it is not dependable.
- ***Usability*** *(R.O.5)*: end-users pose that usability is a **critical** requirement for this application, though this mostly concerns the user interface. System integrators also consider that is **important** that this application can be easily deployed and integrated, for which standard interfaces should be used.
- ***Secure access*** *(R.O.9)*: this can be seen as a way to protect the privacy of communication and restrict it to the intended participants, and therefore it can be seen as **important**.

In terms of functional requirements:

- ***Organizational*** *(R.F.1 and particularly R.F.1.1)*: People need to have “same time” interaction, as a substitute for “same place, same time”, to have conversations to explore possibilities, discuss and structure work that otherwise would take too much time and overhead to do asynchronously (on “different time”).
- ***Group*** *(R.F.2 and particularly R.F.2.1)*: Conversations or sessions should be isolated from each other (among groups, classes, projects, etc) and roles should be honoured (e.g. instructors, learners), therefore users have to be checked and a policy for access should be defined *(R.F.2.1.1).*
- ***Communication*** *(R.F.3, and particularly R.F.3.1):* This application essentially provides communication support required for a group of people that interact, at the same time but on different places, through different channels and allows participants to be aware of the work being performed by the rest of the participants on the session.
- ***Coordination*** *(R.F.4):* There is a need to support the synchronization of work in tasks where a space is shared (e.g. a drawing space with a shared pointer or pencil and potentially conflicting actions, or group polling for decision support) among multiple participants.

- ***Sharing*** *(R.F.5, and particularly R.F.5.1):* People performing an activity together should be able to share things even if they are in different places and be able to act on them at any time, even concurrently.
- ***Task*** *(R.F.7, and particularly R.F.7.1, R.F.7.2):* Group work with remote participants require specific support for sharing annotations, drawings, documents, organize meetings, and particularly (R.F.7.2) face-to-face, same-time interaction with multimedia communication (in the educational scenario the channels can be audio, video, text or group chat, group whiteboard, shared documents) (R.F.7.2.1).

6.2 General Architecture overview

The eMeeting application has client-server architecture. The client-side is a front-end written using Flash. The front-end is connected to the Flash Multimedia Server that in turn connects to a Java-based application. This application implements the procedures and functions. The back-end application uses a database, where information related to users is stored.

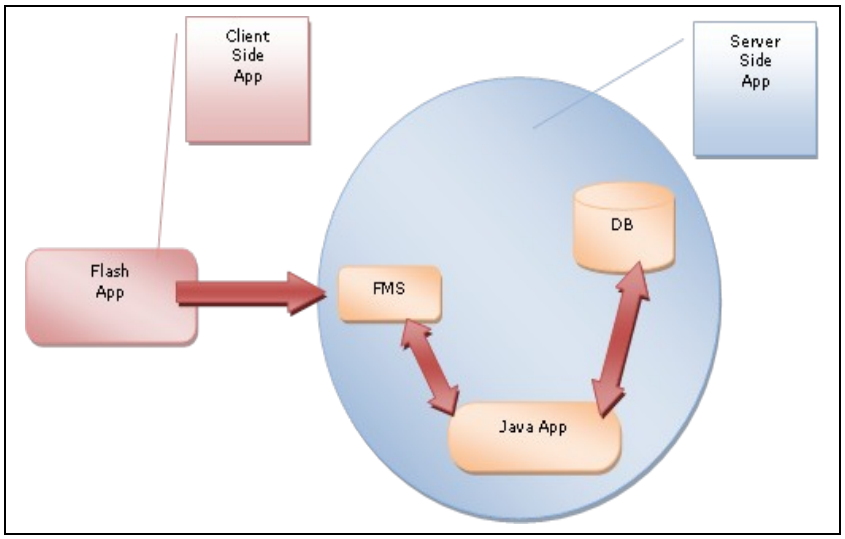

Figure 5. The original eMeeting architecture

eMeeting is a Web application, and it is installed in a web server with Apache Web Server 2.2.X, Tomcat Server 6.X, MySql Server 5.X.

In addition, it is required to use a Flash Media Server for video streaming. This Flash Media Server runs in another machine that is reachable from eMeeting application.

In order to run the application, it is required to install the application in a web server. Database model has to be created; application and database access must be set. Therefore, in the FMS server a component for eMeeting application must be installed. Finally the front-end must be installed in the Apache server, allowing communication with Flex via a PHP backend. At this point, the application is ready to run.

Once the installation process is done, the administrator must introduce the list of users – identified by username. Users willing to use eMeeting service must have the following environment installed in their client machines:

1. Web Browser (Firefox recommended)
2. JVM (Java Virtual Machine - for future integration, so far, there are no concerns about version)
3. Flash Player 9+ installed

This structuring of the application is simple but as a synchronous application it has to provide good response time (or a minimum responsiveness) as the number of participants or the load on the servers grows.

eMeeting could integrate with the Grid4All infrastructure in two ways:

a) The server side of eMeeting is currently executing on a stable environment such as high performance computer (multiprocessing machine). One objective (may have been) to decentralize this part and see how it could in fact execute on G4A like environments, i.e. on simpler machines that may be brought in by members of VOs or may be acquired from markets to the VO.

b) eMeeting is a service that executes somewhere on its own platform (like any other service executing on the Internet). Here what we want is that this service should be usable by members of a VO (VO such as the school VO usage scenario). Here the server and client parts are adapted to use Grid4All services and API so that they can be used by VO users.

The direction taken is (b), as (a) is currently not feasible as one important part of the server side relies on a commercial product (Flash Media Server) that cannot be modified and therefore cannot benefit from potential improvements in the scalability, manageability and dependability that could be built with DCMS. Nevertheless, given that in (b) clients and server are modified, the application could be used in a VO context, and it can benefit from other API functions, as being able to share files with DFS, authenticating users with VOMS, and coordinate agendas using the Shared Calendar application and indirectly Telex.

6.3 Integration issues

We are approaching integration considering eMeeting as a service that executes on its own platform. Thus, our objective is to set a service that could be used by members of a VO (such as the school VO usage scenario). That introduces some changes in client side processes and require some changes in general architecture.

This approach is likely the most sensible considering that eMeeting has a dependency on FMS. It is possible to approach the integration from another point of view, that is, re-architecture the whole application in order to execute it on G4A like environments, however, from the point of view of software development companies it is easier to encourage them to use G4A resources showing a case where integration is possible, changes are not dramatic and the application is improved.

Taking the stated premises, we have identified the key functionalities and services offered by the Grid4All software platform that improves the capabilities of eMeeting. In this approach we are not considering modifying the centralized eMeeting architecture. Our first objective is to adapt the application with minimal changes such that the eMeeting application can provide its services to members of a Virtual Organisation such as on the VO Learning scenario. Here the eMeeting service (back-end) will execute on its own computational resources (outside the VO) and will be accessible as a service to clients executing their front-ends on the compute nodes of the VO.

In previous deliverables, we had described how we envisaged integration of eMeeting with Grid4All services and middleware. The Virtual Organisation Membership Service and the Security service were to be used to authenticated users. The Shared Calendar application developed to demonstrate the Telex middleware was to be used to decide dates for eMeeting sessions. We also envisaged using the VO-aware version of the DFS+VBS to share documents and files. In particular the slides and video may be used by the participants during an eMeeting session.

The following sections explain each of the suggested eMeeting functionalities that are going to be improved with G4A functionalities.

6.4 Authentication

In its current version, authentication is based on passwords. Clients (of a session) provide their login name and password through the front-end graphical tool. The front-end sends this information to the authentication process in the Server Application via HTTP. The server application (Java Application) checks on an internal database, therefore an answer with authentication and proper authorization is done as you can see in the next figure.

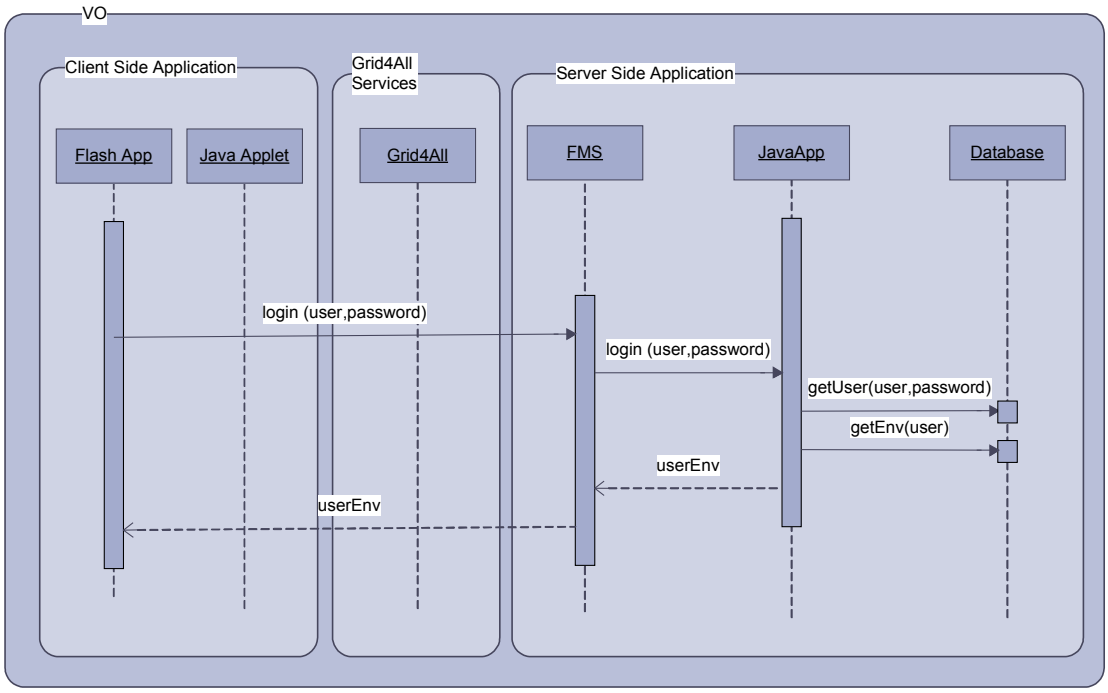

Figure 6. Authentication and authorization (original)

The VO membership service manages members and their associated user information. VO members will access eMeeting after having logged to the VO. It is useless to duplicate user information necessary to authenticate session participants also within the back-end. Moreover we would like to support Single sign-on. Hence our best choice is authenticate VO members joining eMeeting session at the client side.

Thus, it is needed to change some of the business logic done in the server-side. Authentication is done in the client-side application using the VOMS.

Authentication of users who want to participate at eMeeting sessions would be done as follows:

1) Member logs into the VO and the VO authenticates the user.
2) The MemberId created by the VO is sent to the FMS (eMeeting back-end). This identity is stored in the data-base and is used to retrieve the user's environment.

The front-end of the eMeeting application will use the interfaces of the VO membership service to connect to it. If the user is authenticated, the VOMS returns the corresponding identity of the member. This string is then used by the front-end to login to the eMeeting server side application.

Access control may also be implemented through Policy Enforcement Points set at the client side. Using the flexible security architecture as described in D2.2, the VO may enforce control on the participation within sessions transparently to the back-end application.

The front-end executing at the client side is currently developed using Flash technology. The VOMS is developed in Java and exports its interfaces through the Java RMI registry. This is a major inconvenience since Flash does not have connectors to call via RMI. This introduces some changes to the current client-side architecture. These changes are described below:

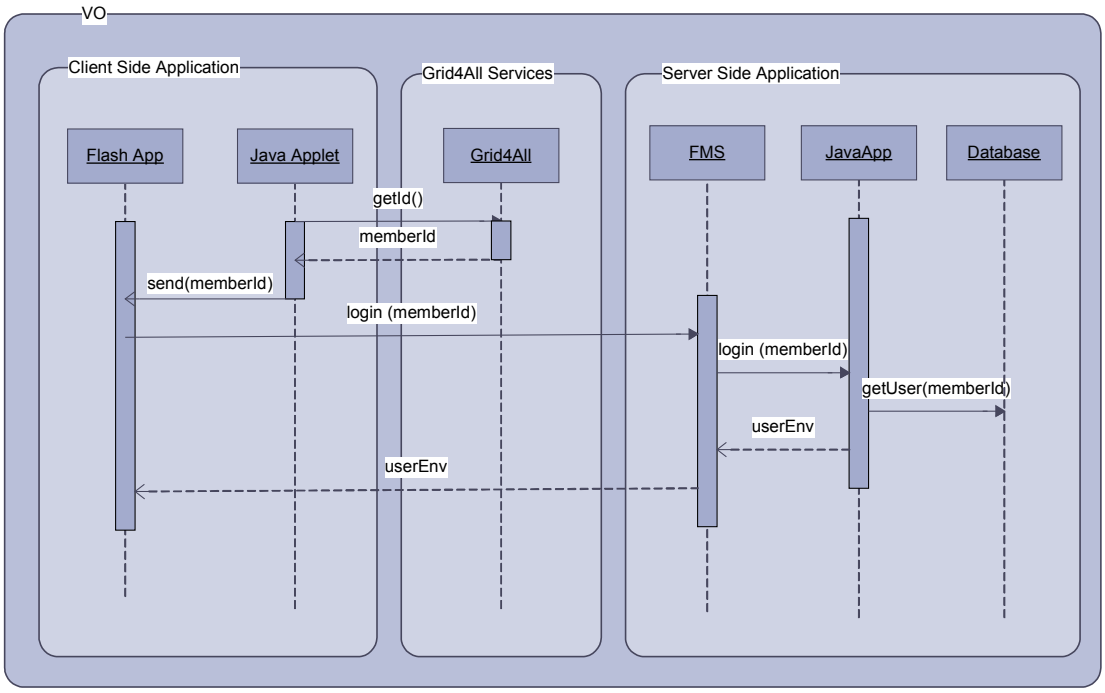

Figure 7. Authentication and authorization (proposed)

6.5 VOFS

The eMeeting application has a feature that allows participants to share files. Files are seen as a collection of resources that are available to the tutor, that is, the role that leads the eMeeting sessions. Files, such as presentations, could be shown to the rest of the participants in the eMeeting session using the slideshow component.

In the current application, files are stored in the server. So, a tutor must upload all the required files to the server before starting an eMeeting session. This is done using an FTP program. These files are kept in the server unless the administrator or the tutor deletes it; this is because these files can be used for another session. Tutor can upload as many files as required; but there's a limitation of space that it is set in function of a commercial agreement.

When a session starts, the tutor has all the available files to either share with others or show. It is in this scenario when collaboration with VOFS has been thought. The main advantage of using VOFS is to avoid loading files and being able to add new files when a session is already running. Hence our objective is to allow participants within a session to share not only the documents uploaded at the back-end but also, and mainly, the documents that they have placed within their own VOFS (within the VO), therefore adding value to the pre-existing application.

Currently, the application has the following model:

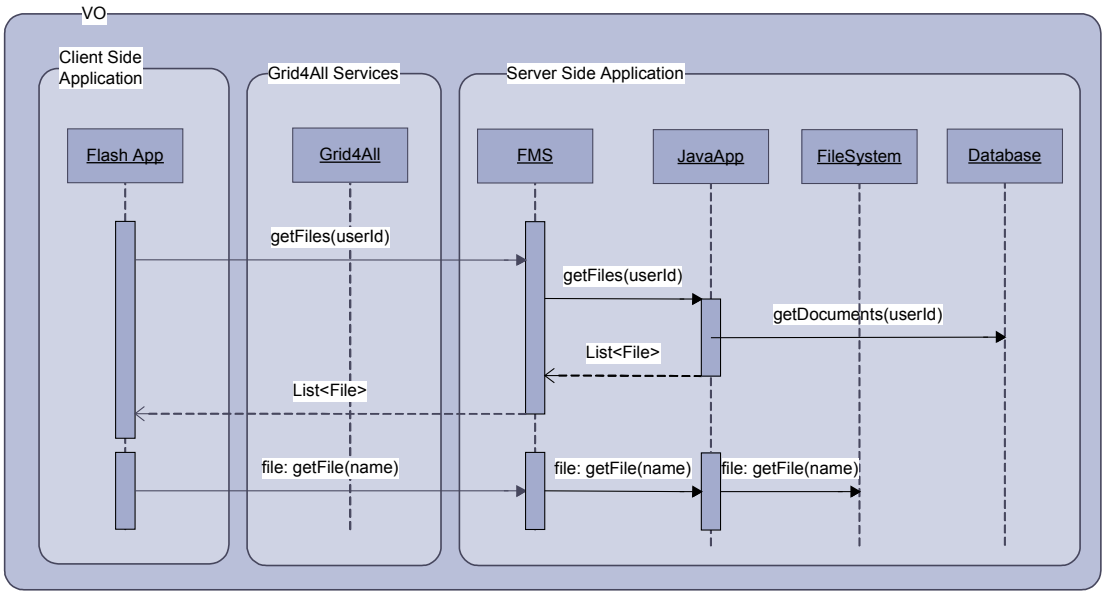

Figure 8. File access (original)

It is all managed by the server-side application (FMS). Server-side provides FMS with a list of all the available files (the ones uploaded previously); FMS in turn, opens shared or shown documents to the rest of users. The Tutor is the one controlling the use of files.

FMS has a major constraint, it cannot open a file which is not located in its local File System, and it is for that reason that in the Grid4All environment, it is needed to open them from the client side application – a web browser with a SWF file. The client Browser distributes any file event information to the rest of users, so when a file is opened or any event on the document is triggered, this is communicated to the rest of users. In addition, due to technological incompatibilities, it is required as it is in authentication, to develop a Java connector.

Attending to all the concerns, the final process would change to this one:

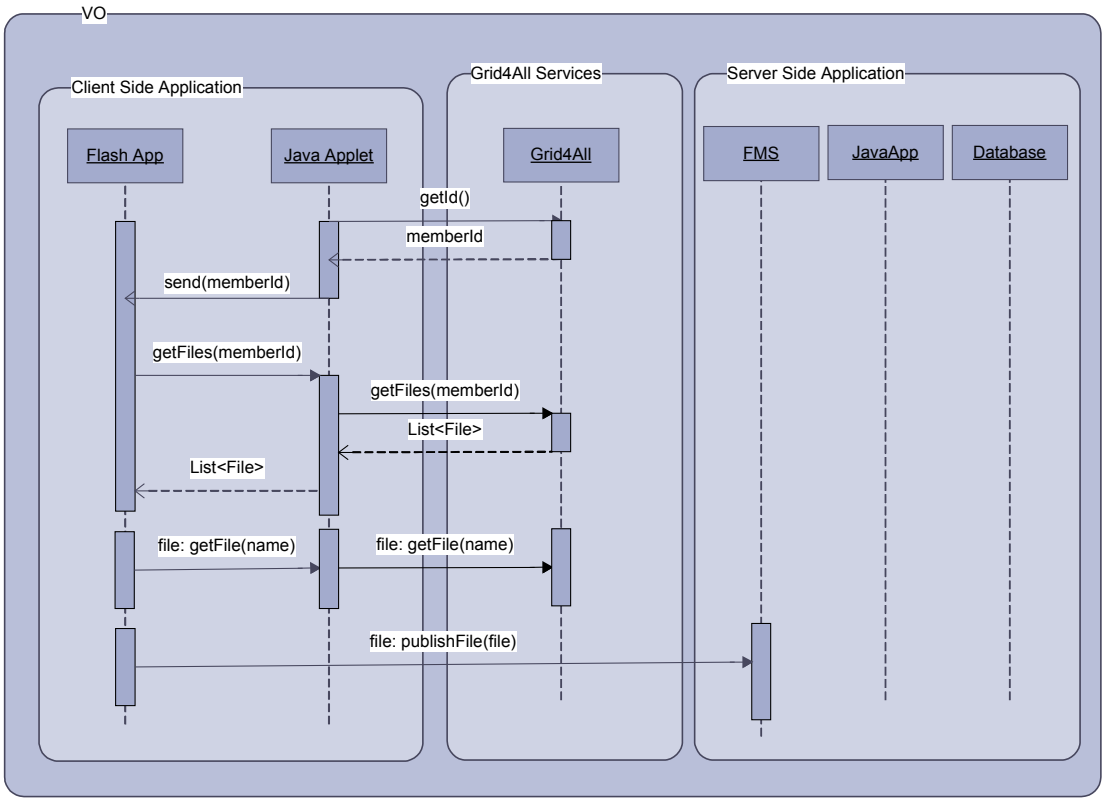

Figure 9. File access (proposed)

6.6 Collaboration with Shared Calendar

As has been introduced in the beginning, eMeeting application allows users to create session. Currently we do not offer tools to allow participants to fix the date according to their availability. An interesting and useful innovation to our product consists of offering session organizers the possibility to schedule sessions based on the availability of the participants as a function of their agendas.

The Shared Calendar is a decentralized tool to allow users to come to an agreement on meeting dates. It allows participants to plan their agenda. The advantage of this tool in comparison to existing calendar tools is that the Shared Calendar does not depend on a centralized calendar server.

Currently the initiator of a session selects the dates and the list of potential participants. The initiator (called tutor) submits this query to the FMS server which then generates the session. Creation of sessions does not take into account availability constraints of the participants.

In the current model all the business logic is implemented on the server-side application, so the client front-end retrieves the list of possible user guests and once the dates has been chosen, a submit to save data is done:

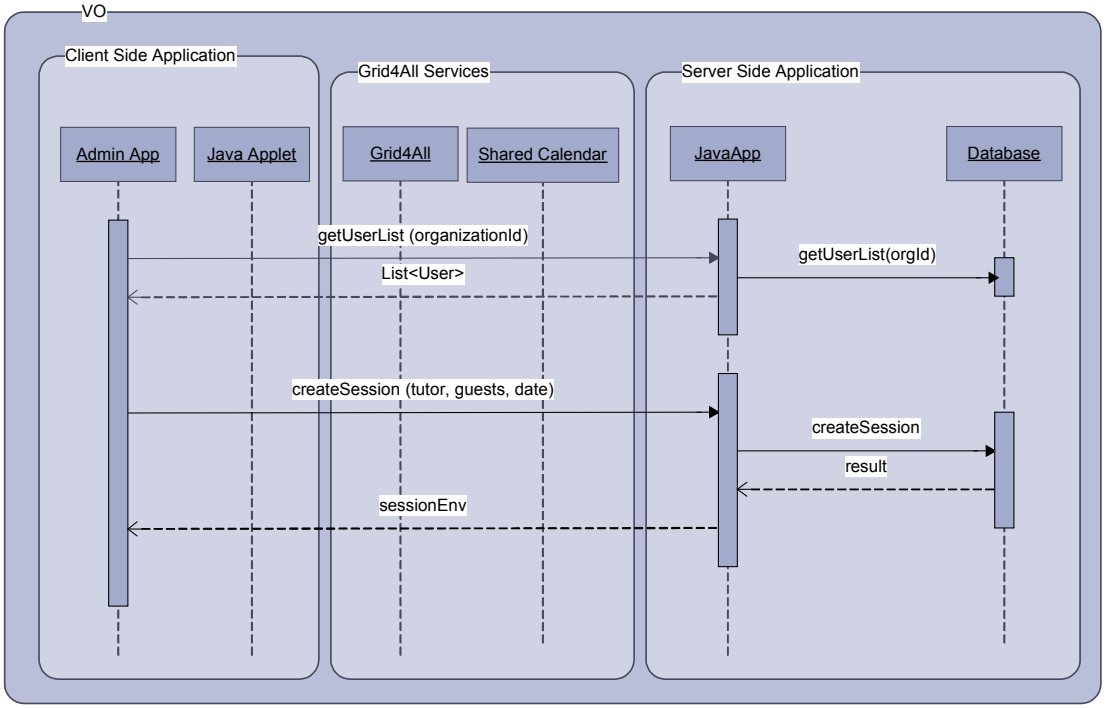

Figure 10. Selecting a date for meetings (original)

We envisage using Shared Calendar within the eMeeting application in the following way: eMeeting requires a fixed date before setting a session, studying the possibilities with shared calendar we suggest a solution for integration. In a collaborative environment, the user willing to meet other users should take into consideration other users availability. Thus, we suggest setting a Session as it is done by now, but this session would be set in a “WAITING FOR CONFIRMATION” status, then, a task is submit to shared calendar in order to sort all the constraints of potential users willing to meet. Once agenda conflicts are sorted out, the shared calendar is going to call a web service (located in the eMeeting server) changing eMeeting Session status or even eMeeting session dates. Users will know whether the meeting is arranged or not checking their calendar application.

6.7 Final Architecture

The final architecture model within the Grid4All environment can be described with the following figure:

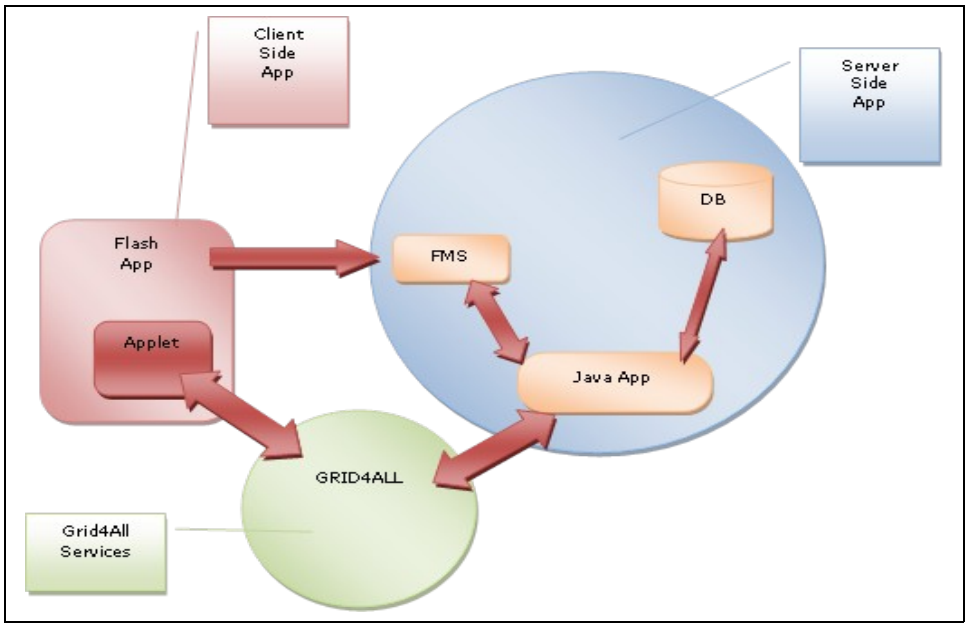

Figure 11. eMeeting architecture (proposed)

As shown in the figure, an applet in the client side Application has been considered to bind with other pieces of JAVA code (local and remote), in addition some operations will use the Grid4All API.

6.8 New potential features and advantages as a result of integration with Grid4All

The eMeeting application benefits from the use of the SC application, allowing to arrange a meeting taking into consideration all the problems related to user availability. This is going to be done automatically, and user just needs to check their calendar to know whether the meeting has been set or not. This integration improves eMeeting application sorting a problem that it is not sorted in current application.

In addition, eMeeting application also benefits from the VOFS file system. Currently the process of uploading files is not user friendly and could take a lot of time. Secondly, there are some limitation concerning to amount of space a user can use to load their files. Then, the following benefits are brought:

- ✔ The time required to upload files decrease, making easier for a non technical user to manage their files using VOFS facilities.
- ✔ The amount of space available to load files is set by the number of members added to VO, and it is not required to be limited as previously.

6.9 Initial evaluation from the developer point of view

Fitting an existing monolithic application (client/server architecture) to a distributed architecture environment is not an easy task. This application has all the business logic located in the server-side application. A distributed environment such as Grid4All, forces to do some changes on the already implemented business logic.

Furthermore, applications use Web browsers that use Flash technology as a part of the client environment. This brings additional issues of compatibility when using other technology such as Java, which is the case with Grid4All. We need to develop bridges and gateway components to interface with such services/components.

In conclusion, analysing different methods by which the eMeeting application could fit with Grid4All virtual organisations has been a useful exercise. We have shown how an existing centralized application can be used with relatively minor changes by members of a dynamic Virtual Organisations. However to leverage all

the benefits of the software results from G4A requires more work, in particular, we need to redesign our monolithic application to a set of modular components that can be executed within a distributed environment.

The following table summarizes the intended usage of Grid4All modules and services by the eMeeting application (L/P: intended Level of Development/Priority).

Domain	Modules & Services	Functionality	L/P	Description
VO Management: administrate a	Membership Manager	Authenticate members	2, 5	Authenticate users and VO Members
Security: manage security	Identity manager(VOMS)	Sign-on and create a proxy certificate with attributes	1, 1	Checking identity
	Policy Administration Point, Policy repository, Policy editor	Create/edit/store VO policies by VO admin	1, 1	Defining access policies
	Policy Enforcement Point , Policy Decision Point, Policy Information Point	Access to a VO resource(s) protected with a PEP(s)	1, 1	Checking permissions
Collaborative & Federative services: manage VO-wide file systems; provide support for collaborative editing of shared documents	Telex	Create/read/write/delete collaborative documents	1,1	Integration with shared agendas
	VOFS Manager, User Agent	Maintain a VO-wide workspace	2, 4	Manage files

7 gMovie

“gMovie” is a service accessible via a web interface, allowing distributed video transcoding on top of VO resources. Transcoding means changing the encoding and/or encapsulation format of the video. However we will also support some scale transformation.

The user will be provided with a list of available movie on a web page. Then he will choose in which format he wants to retrieve one of the videos. After transcoding, the user could be able to view the chosen movie on a cell-phone with small storage and reduced screen size for example.

7.1 User requirements

This demonstrator application aims at meeting several user requirements at the same time:

- ***Performance*** *(R.O.1)*: Transcoding a video file on top of VO resources should be faster than using only a single machine (the user's own one). In the worst case it should be done at a comparable speed (if the VO behind the service has only one computing node). The main bottleneck for performance could be the network, if it is used to upload the initial movie file from the user's computer, or to download the resulting movie. But such a transcoding service would be completely relevant in case of Fiber-To-The-Home scenario, even if the user is uploading his own movies, and still relevant in a ADSL scenario in case user's upload capacity is very limited, the website would then propose a list of already available movies (no possibility to upload for the user).
- ***Scalability*** *(R.O.2)*: it is expected that the transcoding process should adapt transparently to the number of nodes within the VO (new nodes joining, nodes leaving because of failure or end of availability). However, it is expected that this service will have an upper limit regarding efficiency on the number of participating hosts: the video can be split into parts but too many parts would incur that the start-up time of transcoding compared to the transcoding time is too high. So cutting the file into fewer parts would be more efficient.
- ***Usability*** *(R.O.5)*: we claim that using such a service should be very easy from the user point of view, we will hide from the user the fact that the resources are VO-provided. We believe that while the user should pay in order to access such service and/or should be registered in order to use it (if movies have a copyright owner, etc.), the fact that resources are provided by a VO is irrelevant to the end-user.
- In minor degree, ***Manageability*** *(R.O.3)* in the need to manage the transcoding process, ***Dependability*** *(R.O.4)*, in the need to ensure the execution of the transcoding process, and ***Accessibility*** *(R.O.7)* as the use of VO resources reduces the cost of the process.

In terms of functional requirements:

- ***Organizational*** *(R.F.1 and particularly R.F.1.2)*: People working together could pool resources for the activity of the group, and therefore can contribute and use resources that applications such as gMovie can use.
- ***Computation*** *(R.F.6 and particularly R.F.6.1)*: There is a need to perform *heavy* tasks such as scheduling network simulation (bag of) tasks for diverse groups of students on the pool of allocated resources.

7.2 Relation to Grid4All infrastructure

gMovie uses several grid4all components, listed as follows:

- ***VOFS***: it is possible that we use the VO File System in order to exchange data between computing nodes and the transcoding service server (movie input parts, transcoded results). This will depend on the availability of the VOFS. However, even without it, the XtremWeb middleware can handle data transfer.
- ***VOMS***: the grid4all Reservation Manager is used to reserve and acquire resources.
- ***SS***: the Scheduling Service is used to provide completion time estimations.

- ***DCMS***: the gMovie XtremWeb workers are grid4all-Jade enabled (a DCMS layer) in order to be self-deployed on grid4all nodes.

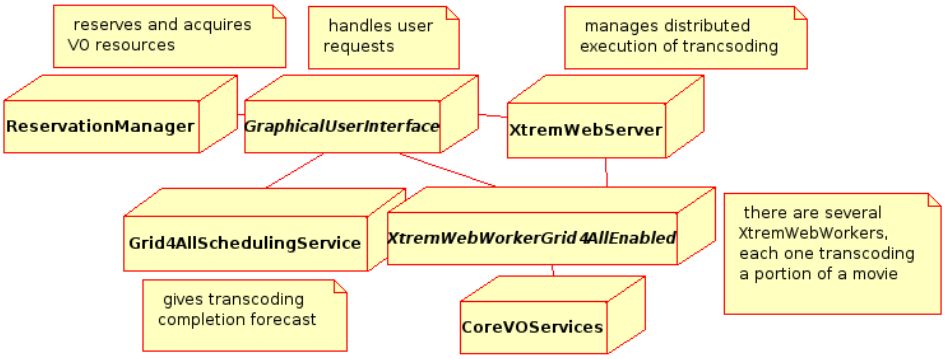

Figure 12: Deployment diagram for gMovie and grid4all components

7.3 Architecture

The gMovie architecture is shown in the above figure. One server (called XtremWeb server) is connected to several clients (called XtremWeb workers). The server distributes data (movie parts) to clients, and then each client transcodes sequentially all the movie parts it was given. Finally, the application gathers all the transcoded movie parts and merges them together to recover the movie in the user-requested format. XtremWeb workers are wrapped as Fractal components (DCMS layer) and deployed through the VO deployment service.

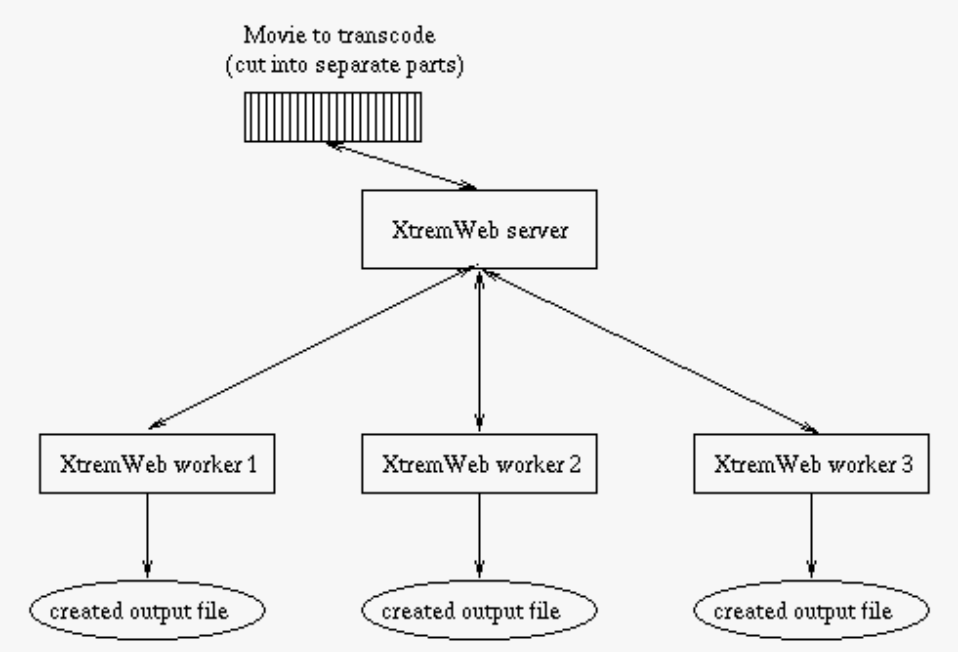

Figure 13: gMovie transcoding architecture

7.4 Authentication

Currently, as gMovie is only a service demonstrator for grid4all, we don't plan to integrate authentication.

In the eventuality of a real deployment, users accessing this transcoding service should have specific accounts allowing them to use resources provided by a VO (either they use their VO account or they pay if they are not VO members).

7.5 Integration issues

The ability to use the VOFS is conditioned by its maturity (documented installation procedure or shell command listing at least, possibility to write a file from a node and read it from another, stability of this procedure).

The following table summarizes the intended usage of Grid4All modules and services by the gMovie application (L/P: intended Level of Development/Priority).

Domain	Modules & Services	Functionality	L/P	Description
VO Management: administrate a VO, its members, its resources	Membership Manager	Authenticate members		
	Resource Manager	Discover, Allocate, Donate, Monitor resources	4, 4	Discover and allocate resources
	Deployment Manager	Deploy applications	4, 4	Deploy our application on previously allocated resources
	Reservation Manager	Reserve resources	4, 4	Resource reservation
Inter-VO services: allocate or offer leases for computational resources by brokering at resource markets; maintain and advertise information on resources	Market Information Service	Query, subscribe information		
	Market factory	Select, deploy market		
	Currency and payment manager	Transfer currency		
	Agreement Manager	Settle, Distribute agreement		
	Negotiator	Negotiate with markets	4, 4	Buy resources
	Auction service	Register, Ask, Bid, Obtain feedback	4, 4	Buy resources (probably through usage of the Reservation Manager)
Execution Management	Scheduling Service	Interacting with the Scheduling Service	4, 4	Execution time forecasting from the SS, even if coarse grained
Overlay Services – DCMS	DCMS	Executing Self-* Component-Based Applications	4, 4	Deploy our software, it is currently encapsulated as a grid4all Jade component

8 SC

SC provides users a way to manage their activities. Each user has his own and independent calendar filled with private events. Additionally he can organize meeting with other collaborators. He creates a “meeting document”, shares it, and notifies invitees of their invitation. For that purpose, he creates an operation (action) on their respective calendar. Thus, he has to import the invitees’ calendar.
When one receives an invitation he can collaborate to hold the meeting: he can invite other users, and modify the meeting time. For that purpose he creates operations (actions) on the corresponding “meeting document” concurrently with other invitees. Consequently conflicts may appear. As we are in an optimistic replicated environment, those actions are “tentative” until they are “committed”. In case of conflict some of them are “aborted”.

9.1 User Requirements

The main user requirement is to be able to collaboratively crate and manage events, detect conflicting events (double booking) and achieve eventual consistency (R.F.5 regarding unrestricted sharing and specifically for the task R.F.7.1.1). This occurs in a certain organizational context (R.F.1.1) with groups of people (R.F.2.1.1) and to the extent the information in part or all may be private, there is a need for secure access (R.O.9).

9.2 Relation to Grid4All Infrastructure

SC application is build on top of Telex. Telex is used for managing calendar information of the system. Telex computes which actions are going to be executed and in what order. All conflicts are detected and eventually solved with Telex.

9.3 Architecture

The architecture of SC is described in deliverable D3.1 “Requirements analysis, design and implementation plan of Grid4All data storage and sharing facilities” chapter 1.

9.4 Integration issues

SC application is fully integrated with Telex since it has been developed on top of it and using it as an essential component.

Currently it is not possible to collaboratively change the time of an event. This will require extensions to Telex to associate the time updates with some user invitation to detect a double booking, which is future work.

The following table summarizes the intended usage of Grid4All modules and services by the SC application (L/P: intended Level of Development/Priority).

Domain	Modules & Services	Functionality	L/P	Description
VO Management: administrate a	Membership Manager	Authenticate members	3, 1	Authenticate users and get user status
Security: manage security	Identity manager(VOMS)	Sign-on and create a proxy certificate with attributes		
	Policy Administration Point, Policy repository, Policy editor	Create/edit/store VO policies by VO admin		
	Policy Enforcement Point , Policy Decision Point, Policy Information Point	Access to a VO resource(s) protected with a PEP(s)	3, 1	VOFS+Telex manage basic security
	PEP, PDP, PR, PIP, PAP	Set/view ACL for a VO resource.	3, 2	Set ACL for calendars
Collaborative & Federative services: manage VO-wide file systems; provide support for collaborative editing of shared documents	Telex	Create/read/write/delete collaborative documents	5, 5	SC is Telex based application
	VOFS Manager, User Agent	Maintain a VO-wide workspace	5, 5	Access to storage through Telex

9 Conclusions

This document describes the mapping and coverage between user requirements, end-user oriented applications and the Grid4All API, and the implications and priorities for the phase of integration. The choice of a collection of diverse applications is seen as helpful to show how a wide range of applications could also make use of the Grid4All API. As part of the integration process there is also a task for the refinement of the details of the API that will be done in coordination between users and developers of these API operations. That will raise the opportunity to abstract out and find higher level functions and therefore propose transparency mechanisms that hide lower level details of the API, thus facilitating porting applications to the Grid4All environment.

This document is delivered in parallel with D4.4 describing the status of development of end-user applications by M24, and signalling the start of the integration phase, where developers of applications and infrastructure will work together to enable and proceed with the integration. This work will be reported in D4.5.

10 References

D3.1 Requirement analysis, design and implementation plan Grid4All data storage and sharing facilities

D4.7 Measurable requirements

D4.3 Prototype API usage by Grid4All based applications, 2008.

D4.4 Initial Prototypes of applications, 2008.

D4.5 Integrated Prototypes of applications (M30)