Grid enabled access to rich media content

Final Version of the Grid Middleware

Project Reference No.	**FP6-34363**
Deliverable No.	**D3.2: Final Version of the Grid Middleware**
Workpackage no:	**WP3**
Nature:	**P (Prototype)**
Dissemination Level:	**PU (Public)**
Document version:	**Final**
Date:	**11/07/2008**
Editor(s):	**ICCS, CYF**
Document description:	This deliverable reports on the development activities towards the delivery of the final prototype for the GREDIA Middleware services and subcomponents.

History

Version	Date	Reason	Revised by
01	20-06-2008	Initial Document	A. Asiki, K. Doka, A. Zissimos, I. Konstantinou
02	28-06-2008	Corrections, binding	D. Tsoumakos
03	07/01/2008	CYF contribution	B. Kryza
04	07/07/2008	Conclusions/Future Work and corrections	N. Koziris, D. Tsoumakos
05	11/07/2008	Final Document after Peer Review	N. Koziris, D. Tsoumakos

Authors List

Organisation	Name
ICCS	N. Koziris, D. Tsoumakos, A. Asiki, K. Doka, A. Zissimos, I. Konstantinou
CYF	B. Kryza, L. Dutka, R. Slota, J. Kitowski, M. Zuzek, C. Wisniewski, M. Talik, T. Swierczynski

Table of Contents

List of Figures 4

Executive Summary 5

1. Introduction 6

2. Data Replication – Persistence 7

2.1. Data replication 7

2.2. Metadata Persistence 7

3. Querying over Multiple Attributes 8

3.1. Complex queries 10

3.2. Keyword Searches 11

4. Data transfer Layer 14

4.1. Distributed Replica Location Service 14

4.2. GridTorrent Daemon Implementation 14

4.3. GridTorrent Control Interface 14

4.3.1. Implementation details 15

4.3.2. Deployment scenario 16

5. Framework for intelligent Virtual Organizations 18

5.1. Overview 18

5.1.1. Contract Ontology 18

5.2. Contract Negotiation Mediator 20

5.2.1. Contract Negotiation Component 20

5.2.2. Contract Negotiation Protocol 20

5.2.3. Starting negotiations 21

5.2.4. Ending negotiations 23

5.2.5. Contract Negotiation User Interface 24

5.3. FiVO Installation 25

5.3.1. Requirements 25

5.3.2. Installation of FiVO Contract Negotiation Service 25

5.4. FiVO evaluation 27

5.4.1. Scenario overview and goals 27

5.4.2. Actors and resources 27

5.4.3. Simplified Happy Bank Scenario 28

6. Conclusions – Future Work 33

7. References 34

List of Figures

Figure 1: Inserting and searching a metadata file in the Metadata overlay..9
Figure 2: A complex query in the *Metadata* overlay 11
Figure 3 Three objects annotated with the depicted keywords.. 11
Figure 4 List of inverted indices for the three objects... 11
Figure 5 List of inverted indices distributed over the nodes of a DHT. 12
Figure 6 The GridTorrent componets with the daemon and control connection interface features 16
Figure 7 GridTorrent deployment scenario 17
Figure 8: General structure of Contract Ontology 18
Figure 9: FiVO plug-in to Protege 2000 for contract negotiations 24

Executive Summary

This document describes in detail the steps taken towards the implementation of the final version of the Gredia middleware. Specifically, we discuss the finalization of the SFC method as well as how all different dimensions can be used in order to provide users with a powerful search mechanism. We describe data/system persistence schemes implemented in order for a robust operation in case of failures. Finally, we describe the implementation steps taken in order to provide a fully functioning GridTorrent client-server module that is fully integrated with the grid and DRLS services.

1. Introduction

The design of a generic middleware for efficient search and retrieval of annotated content in a distributed Grid environment has been motivated by the requirements posed by the GREDIA research project. GREDIA's main objective is the development of a Grid application platform, providing high-level support for the implementation of Grid business applications through a flexible graphical user interface. This generic platform will facilitate the provision of business services, which mainly demand access and sharing of large quantities of distributed annotated numerical and multimedia content. Furthermore, the GREDIA platform will exploit Grid technologies to enable access to the distributed content by mobile devices. The establishment of the described functionality faces great challenges regarding data management, security, authentication and authorization mechanisms.

In the heart of such a platform lies the middleware subsystem that essentially performs all data management operations for the system. We have thoroughly described the proposed middleware architecture in the GREDIA Deliverable D2.1: Middleware Architecture. In Deliverable D3.1 we presented the first version of the Middleware components: We described how we implemented the Data and Metadata overlays, our first efforts into the implementation of multi-attribute queries and our design and implementation of GridTorrent, a cooperative, Grid-enabled data transfer mechanism. In this section we will provide a brief overview of the final steps taken in order to present the complete Middleware component.

The proposed architecture deals with data management in a distributed environment consisting of resources belonging to different Virtual Organizations. A main characteristic of this environment is the heterogeneity of nodes in terms of computational power, storage capacity and bandwidth. Resources with different features, for example laptops, desktop computers, dedicated servers or even mobile phones, may participate in this platform. Moreover, we assume that some resources may remain off the system for long periods of time. Therefore, the design should cope with node arrivals and departures so as to provide a robust operation. In one part of this work we describe the steps taken in order to ensure smooth system operation even during unexpected node failures.

Another very important aspect of the Middleware is the indexing of the inserted files. The data in Gredia are annotated and these annotations are then used in order for the system to provide a powerful search mechanism. The annotations pass through our SFC component which maps them into a 1-dimensional ID and stores them into the Metadata overlay. In this work we describe the final steps into the implementation of the Space-Filling Curves method as well as our special method of dealing with keywords, as we anticipate they will constitute the vast majority of searches.

Finally, in the previous Deliverable we described GridTorrent, a fully cooperative data transfer mechanism that operates within Gredia in order to for any upload/download operation to take advantage of multiple locations of the same file. In this work we shall describe in detail how the GridTorrent clients and servers are integrating with the DRLS as well as our implementation issues in order for this component to modularly work with the other components.

This deliverable covers also the prototype description of the Framework for Intelligent Virtual Organizations, which allows VO participants to negotiate a contract and configure properly the middleware layer to work according to the rules agreed upon with respect to authorization and QoS.

In the following Sections we provide detailed descriptions of the steps taken concerning the implementation of the proposed architecture.

2. Data Replication – Persistence

2.1. Data replication

In this section, we describe the replication strategy we use concerning the storage overlay. For the purposes of the *GREDIA* platform we apply a static replication scheme. We consider that each organization has a pre-configured list of well known storage nodes within its premises, and a static replication factor *m* (the number of replica instances per stored item). Every time the server list or the *m* factor changes, storage clients are informed with a *PULL* mechanism. When a user wishes to share a local file that exists in its machine (e.g. a mobile phone), it uses the data service to randomly select a number of *m* available replica servers. When the list of the candidate storage nodes is decided, the client's GridTorrent session performs two actions: Firstly, it “advertises” the local file in the *DRLS* overlay, and secondly it remotely instructs the GridTorrent sessions of the selected storage nodes to start the file download. The transfer is being performed in a bittorrent like fashion, where the involved nodes participate in a bittorrent swarm[1]. After the transfer is completed, *m* storage nodes along with the client machine keep a replica of the file, and the *DRLS* overlay is informed about the new *m* replica locations. With this mechanism, the file is instantly available to other peers that wish to retrieve it, even before the replication procedure finishes.

Replication of crucial user data in dedicated servers is required both in the media and in the banking application. In both cases, we need to ensure that a new data item cannot be stored only in the user's machine, especially when this has limited internet connectivity or battery power.

2.2. Metadata Persistence

In this section we depict the mechanism we apply during leaves/disconnections of nodes that participate in the *DRLS* and *Metadata* overlay. Every p2p node comes with an sqlite database [2] which is used as its local storage back end system. The database consists of a table containing key-value pairs that are indexed by the local peer instances. The sqlite subsystem can be configured to store the indexed items in the main memory or in a .sql file in the local file system. Although the main memory is faster, as it does not involve frequent disk i/o operations, it has a main drawback: when the contents of the main memory are erased, e.g. during a restart, the indexed items are lost. Hence, sqlite has been configured sqlite to dump the indexed contents in a local file, after every successful item insertion or update. Every time a node starts the p2p application, it checks the contents of the sql file, and it re-indexes every item that is stored in this file.

What is more, the *kademlia* p2p protocol is configured to replicate every index instance to more than one neighboring nodes (using a static replication factor *m*). Metadata replication along with the sqlite local file dump ensures that an indexed item will be successfully located with high probability, even during situations with high network churn. The only situation where an indexed item cannot be located is when all *m* servers that index this item are off line.

[1] http://www.vodscape.com/bittorrent/BitTorrentSwarms.html

[2] http://www.sqlite.org/

3. Querying over Multiple Attributes

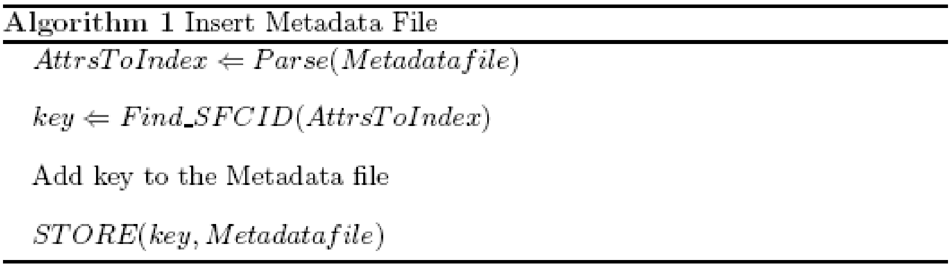
The search mechanism among the annotations of the contents is provided by the *Metadata service* hosted in the *Metadata* overlay. Each data file is described by a set of attributes following a proposed Metadata schema according to the pilot application. For example, an image can be tagged with attributes such as title, location, topic, keywords, creator name, duration, date, format, size. Some of these attributes (size, format, type, date) can be automatically completed by the Metadata client. The rest of them are filled in by the owner of the image or other authorized users inside a Web form. We have considered that only the most important attributes of the defined schema are indexed, in order to reduce the complexity of the indexing method and limit the search space for faster results.

As it has been described in Deliverable D3.1, we exploit the properties of the Hilbert Space Filling Curve (hence Hilbert SFC) to support multiattribute queries. In our multidimensional indexing scheme, we consider that the set of d-attributes to be indexed forms a d-dimensional space. Each point in this space represents a combination of values of indexed attributes. The points of the d-dimensional space are mapped down to a single dimension by the Hilbert SFC. A basic property of every SFC is its recursive generation. Initially, the d-dimensional space is subdivided into 2^d cells mapped into the First Order Hilbert Curve. In the next recursion, each cell is subdivided 2^d more times and is mapped into the Second Order curve. The number of recursions is indicated by the ***factor k*** called approximation order. This factor defines the number of space partitions and thus the precision of the algorithm. The derived values in the single dimension represent the keyset of the Kademlia-based DHT overlay. Each key is kd bits long and each node in the overlay manages data in one contiguous range of the SFC.

The mapping of a point in the d-dimensional space to its position in the SFC is executed by the *SFC component.* The SFC component provides the following operations:

- *Find_SFCID*: Maps the coordinates of a point in the d-dimensional space into its position in the SFC.
- *Find_Coordinates*: Maps a point in the SFC to its coordinates in the d-dimensional space.
- *Traverse_SFC*: Produces the SFC recursively enumerating the points s of the d-dimensional space along the SFC in order to find adjacent points.

The algorithms in *Find_SFCID* and *Find_Coordinates* use circular shifts and exclusive-or-operations on bytes, as introduced by Butz [5]. In order to insert a metadata file in the Metadata overlay, we parse the annotation provided by the user. The values of the attributes to be indexed are isolated and provided as input to the SFC component. The SFC component uses the Find_SFCID method to produce an ID. This ID is stored in the metadata file and is used as the key in the STORE RPC of the Kademlia protocol. The steps of the described procedure in order to insert a metadata file in the Metadata overlay is shown in Algorithm 1.

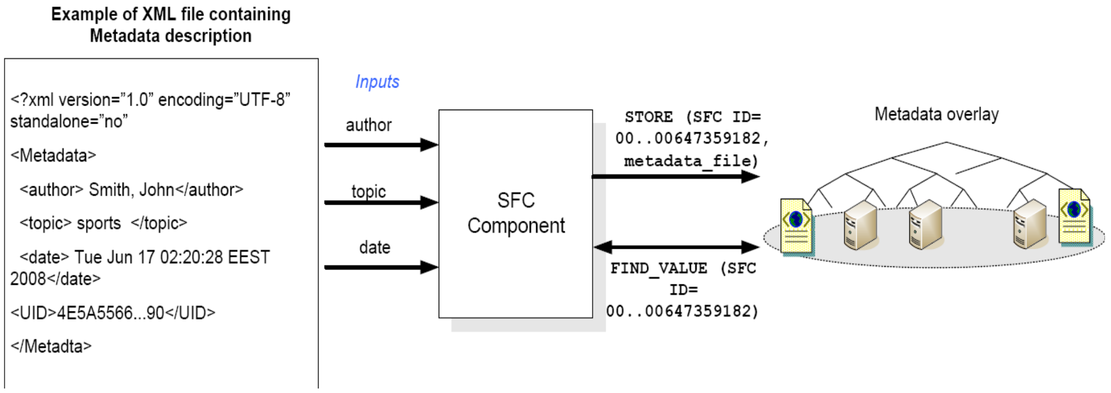


Algorithm 1 Insert Metadata File

```
$AttrsToIndex \Leftarrow Parse(Metadatafile)$

$key \Leftarrow Find\_SFCID(AttrsToIndex)$

Add key to the Metadata file

$STORE(key, Metadatafile)$
```

The search of a metadata file requires knowing the exact key used during the insertion of the metadata file in the DHT. This knowledge is acquired by using the SFC component. Therefore, the values of the attributes for the searched metadata file are given as input to the algorithm that maps the coordinates of a point to the derived key. The output is the SFC ID that has been used during the insertion phase of the specific metadata file, as described in Algorithm 1. The node instructed to lookup the specific metadata file in the network issues *alpha* parallel queries to the *kappa* closest nodes it is aware of. The node continues the process until the *kappa* closest nodes to the target key are found. Then, the node starts the procedure of retrieving the specified value with the Kademlia FIND_VALUE RPC.

An example of insertion and search of a metadata file is shown in Figure 1. In this example, the XML file contains a metadata description, where three attributes are going to be indexed. The author attribute is considered to be a string, the topic is categorical and the data is parsed, so that its numeric representation to be used. The XML file is enhanced with the UID of the data file as well. The values of the three attributes to be indexed are given as input to the SFC component. The produced ID is used to forward the XML file in the proper node of the *Metadata* overlay. In order to search the specific metadata file, the values of the author, topic and data attributes should be given to the SFC component. The output is the ID of the metadata file. The FIND_VALUE RPC of the Kademlia protocol is used for this ID, so that the metadata file to be found.


Example of XML file containing Metadata description
Inputs
author
topic
date
SFC Component
STORE (SFC ID= 00..00647359182, metadata_file)
FIND_VALUE (SFC ID= 00..00647359182)
Metadata overlay

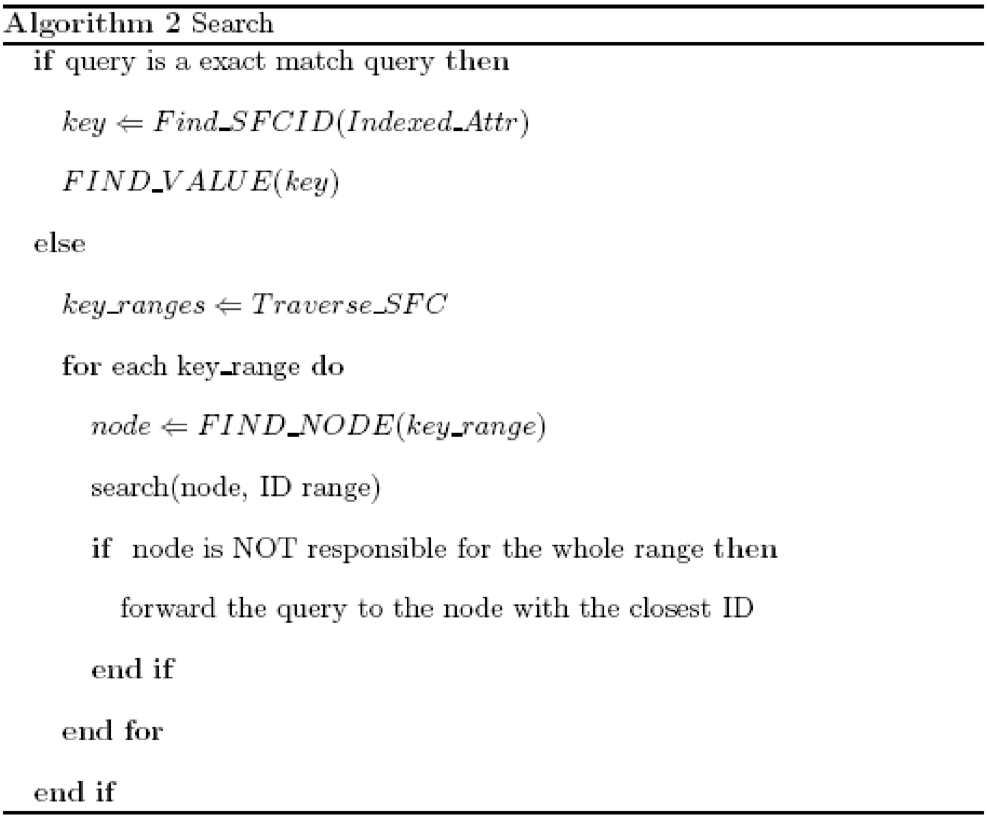
```
<?xml version="1.0" encoding="UTF-8" standalone="no"
<Metadata>
  <author> Smith, John</author>
  <topic> sports  </topic>
  <date> Tue Jun 17 02:20:28 EEST 2008</date>
<UID>4E5A5566...90</UID>
</Metadta>
```

Figure 1: Inserting and searching a metadata file in the Metadata overlay

Since the attributes of the metadata file are considered to form a multidimensional space, the answer to a query corresponds to a specific ID or to ID ranges. A single ID is the answer to an exact match query, if the user searches for a specific combination of the indexed attributes, as shown in the example of Figure 1. The processing of a query consists of two consecutive phases. In the first phase, the clusters of the SFC points answering the query are determined. To define the clusters of IDs of the metadata files answering a range query, we generate the Hilbert SFC recursively. We utilize this feature of the SFC in order to exclude areas of the SFC not containing relative IDs. Thus, we avoid traversing all the points of the SFC in a serial way to find the matching clusters of IDs. The traversal of the Hilbert SFC is done by *Traverse_SFC*. In this method, we use movement tables to produce the curve recursively beginning from the First Order curve [6]. The subdivision of a cell stops if it does not contain relative IDs.

In the next phase, lookup operations for all the cluster(s) start. Lookups for IDs ranges requires the modification of the Kademlia FIND_VALUE. Assuming that churn will not be an issue for our target applications, each node is aware of the node with the closest ID in the network and maintains an index towards it. Each time that a query for an ID range arrives to a node, it scans its keys and replies with the ones included in the ID range. If the node is responsible only for a subrange, then it forwards the query to the next node. The forwarding of the query continues until all the objects within the ID range are retrieved. The steps of the described search procedure are presented in Algorithm 2.

The number of bits used for value encoding in each dimension is equal to the approximation factor. We consider that there are three categories of attributes to be indexed, numerical, categorical and string attributes. For numerical types such as date, size, duration, we use their bit representations as input to the SFC component. The categorical attributes can be easily encoded due to the fact that the number of different values is limited. For string attributes, we use a hash function to map them into the available value ranges. A special case of string attributes is keywords. Our goal is to allow a user to describe a file with keywords without restricting their number. Moreover, keyword searches are the most popular in such a platform and the search mechanism should provide fast replies. For these reasons, we have decided not to include the keywords in the dimensions of the SFC. Each keyword is hashed and inserted in the *Metadata* overlay separately. The value for this (key, value) pair is the ID produced by the SFC component. Therefore, queries for keywords can be answered directly with FIND_VALUE operations in the DHT.



Algorithm 2 Search

```
if query is a exact match query then
  $key \Leftarrow Find\_SFCID(Indexed\_Attr)$
  $FIND\_VALUE(key)$
else
  $key\_ranges \Leftarrow Traverse\_SFC$
  for each key_range do
    $node \Leftarrow FIND\_NODE(key\_range)$
    search(node, ID range)
    if  node is NOT responsible for the whole range then
      forward the query to the node with the closest ID
    end if
  end for
end if
```

3.1. Complex queries

A more complicated scenario is to search data files according to their keywords and a range of values for one or more attribute(s). Such an example is shown in Figure 2. The user searches for metadata files containing two specific keywords, while it corresponds to a specific topic and is produced during a given period. At first, each keyword is hashed and lookup operations start for each generated ID. The inverted lists containing these keywords are found and the SFC IDs of the metadata files containing these keywords are returned. The various IDs are concatenated and arranged into ranges. These ranges are given as input to the SFC component. The SFC component receives also as input the values of the attributes indexed according to the SFC method. It traverses the Hilbert SFC and produces the ranges of IDs of metadata values that contain the given indexed values as well. Finally, these values are retrieved from the *Metadata* overlay and presented to the user.

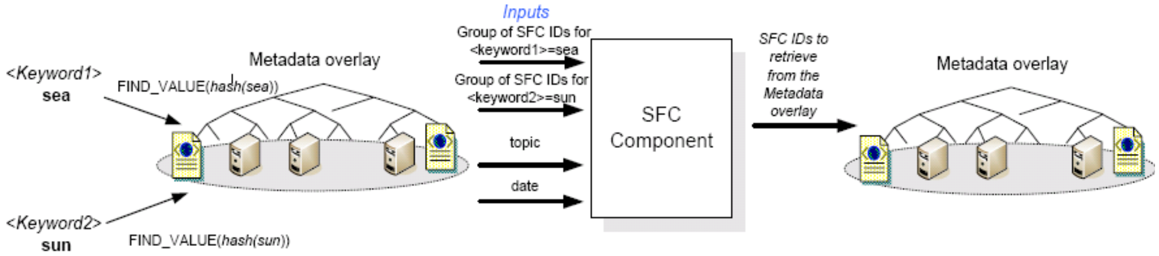


Figure 2: A complex query in the *Metadata* overlay

3.2. Keyword Searches

In a distributed search engine multiple keyword queries dominate the search workload and optimizing them is important for end user performance. Thus, there is a need to separately address the keyword search issue, in order to create a DHT-based storage system that can provide efficient keyword search functionality.

The most common way to implement keyword search in information systems is by inverted index. An inverted index is a structure that comprises of mappings between a keyword and a set of objects containing this keyword. For example, if we consider three files (a.jpg, b.txt, c.jpg) with corresponding keywords describing the contents of these files (Figure 3), the inverted indices' list is the one depicted in Figure 4. To find an object that contains a given set of keywords, one has to find the intersection of the sets associated with these keywords.

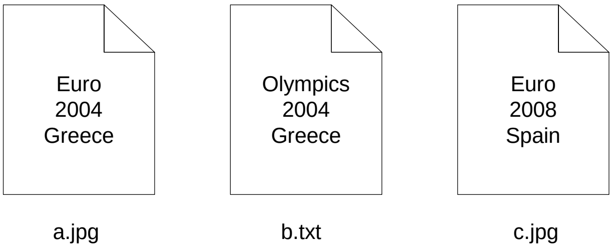


Figure 3 Three objects annotated with the depicted keywords.

2004	a.jpg, b.txt
2008	c.jpg
Euro	a.jpg, c.jpg
Greece	a.jpg, b.txt
Olympics	b.txt
Spain	c.jpg

Figure 4 List of inverted indices for the three objects.

To implement a keyword search over a P2P network, a distributed version of the inverted index can be built. A straightforward way to decentralize the inverted index list is to give each keyword a unique ID (by applying a hash function on it) and assign it to the corresponding node of the DHT. The DHT implementation, apart from the <fileID, value> mappings, must maintain <keyword, list of values> information.

When a user searches files using a keyword, the extended DHT forwards the query to the node that contains the location of the files annotated by that keyword. If a user specifies multiple keywords to locate a file, the underlying system should calculate the intersection of the results for each keyword before returning the result. This can be achieved in two ways. The first one is parallel query processing, where keywords are resolved separately and in parallel, with all results gathered in a node responsible for calculating the intersection of the various sets. This method reduces the time needed to resolve the query, due to parallelization, at the cost of greater bandwidth consumption, since all result sets should be transmitted to the node calculating the final result. The second option is the chained query processing, where each node hosting a keyword of the query is visited sequentially. Intermediate results are transferred from one node the next, where they are filtered accordingly. This method reduces the bandwidth consumed, since the intersection of the result sets is performed gradually, thus reducing its size. In our case, we have chosen the second approach.

As an example, let us consider the three files of Figure 3. Using the keyword as key for consistent hashing, we distribute the location information of the files. For file a.jpg, each of its keywords is hashed and assigned to a DHT node. Euro is assigned to N2, Greece is assigned to N3 and 2004 to N4. Since a query for any of these keywords should return a.jpg, its location information is stored at all three nodes.

If a user wants to find files containing both Greece and Olympics, the query message will be first routed to the node responsible for Greece. The intermediate results (a.jpg, b.txt) are then sent to the node responsible for Olympics, where the final result (b.txt) is generated by intersecting the intermediate results with the file list for Olympics.

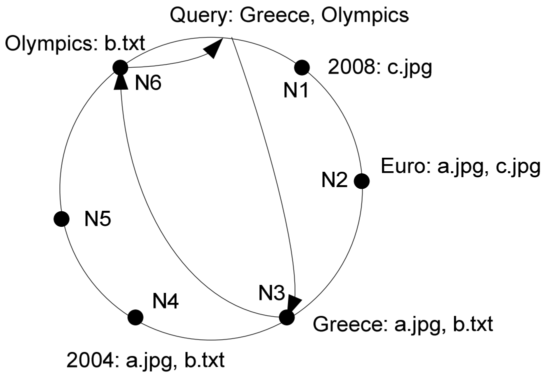


Figure 5 List of inverted indices distributed over the nodes of a DHT.

This approach however has some drawbacks. Some keywords that are common may be associated with a large number of files, while other keywords appear less frequently. Therefore, the various <keyword, list of values> mappings have different space requirements, leading to load imbalance. Moreover, a search query containing a common keyword will generate a large volume of traffic, since a long list of file location information must be transferred to the node responsible for the next keyword. What is worse, not all search results may be used to answer the user's query, since these common keywords are usually too generic and may thus contain a large amount of irrelevant information.

To tackle this issue, we describe a mechanism to optimize chained query processing. When querying with multiple keywords, identifying popular keywords and processing the query with uncommon ones first significantly reduces the amount of data transferred from one node to the next. Considering the previous query for Greece and Olympics, if the query processing begins from N6, responsible for the keyword Olympics, only the file b.txt will be transferred to the node hosting Greece, causing less network traffic. For this purpose we introduce a data structure held by every node of the overlay, which contains common keywords. We call this structure Common Keywords Dictionary. When a node responsible for a keyword determines that this keyword is common, i.e. the number of files associated with it exceeds a threshold, it registers it to the Common Keywords Dictionary. The content of this dictionary is propagated across the DHT nodes, in a fully decentralized manner. Query messages are used to piggyback the dictionary updates which are propagated to all DHT nodes after a time period. The storage overhead for each node is relatively small, since a zipf-like distribution of keyword popularity indicates that a small fraction of keywords is common and is thus inserted to the dictionary.

Revisiting our previous keyword search example, we describe the keyword search operation with the use of the Common Keywords Dictionary. When a user issues a query for Greece and Olympics, the initiating node first consults the local Common Keywords Dictionary, which indicates that the keyword Greece is common and is therefore not a good term to start the searching procedure with. So, the node decides to begin the keyword resolution from the node responsible for Olympics. The result set is then forwarded to the node responsible for Greece, where the intersection is calculated and returned to the requester.

4. Data transfer Layer

The purpose of this layer is to provide a data transfer mechanism that effectively deals with large file uploads and downloads, even when numerous requests rely on a single data source, maximizing bandwidth utilization. The proposed solution, GridTorrent, constitutes a decentralized approach that, unlike GridFTP, takes advantage of an incentive-sharing policy to boost aggregate transfer throughput. This section provides a short description of the current implementation status of GridTorrent.

4.1. Distributed Replica Location Service

GridTorrent needs a file catalog, where information about file metadata can be found, as well as a dynamic list of peers that have the actual file data. GridTorrent has been implemented in a flexible and modular way, to support different types of catalog working in parallel. The types of catalog that are supported at the time are:

- Globus RLS, the Globus Replica Location Service that is supported, maintained and distributed by Globus.
- DRLS, a distributed replica location service that is implemented using the CHOROS mutable distributed hash table on top of the PeerPlatform infrastructure.
- WSDRLS, the Distributed Replica Location Web Service that is implemented for the purpose of the GREDIA project.

GridTorrent has a java interface to abstract all the catalog actions and multiple implementation of this interface, one per catalog type. There is also a java Factory, which looks up the catalog URL, and more specifically the protocol prefix and invokes the appropriate implementation. The prefixes currently supported are:

- rls://fully.quallified.domain.name:port for the globus RLS
- drls://fully.quallified.domain.name:port for the DRLS
- https://fully.quallified.domain.name:port for the GREDIA Web Service DRLS

4.2. GridTorrent Daemon Implementation

The GridTorrent implementation has been extended to be able to run as a service. Earlier implementations of GridTorrent where initiated by a user and remained attached to a console, not being able to run as a system service. The current implementation uses the Apache Commons Daemon interface, to start/stop the GridTorrent daemon. Using this interface GridTorrent can be detached from the user console and integrate with the underlying operating system. There is no need for GridTorrent to be attached to a console anymore, or for a user to initiate the server. All these functions are defined and handled by the Apache Commons Daemon package and are integrated with GridTorrent.

4.3. GridTorrent Control Interface

As a data transfer mechanism the GridTorrent Server should also have a control channel. Earlier implementations of GridTorrent where initiated by a user interactively and could not issue commands to a remote server. This capability is essential to GREDIA for two reasons:

First, the GridTorrent daemon implementation is a server application, which runs in the background as a service with no keyboard or console attached and thus a way for user interaction. So this Control Interface can be used to administer the GridTorrent service.

Second, in the context of GREDIA there are the requirements of data upload and data replication. The data upload may be a simple operation, but in a torrent-like environment it is a two step operation. More precisely, a peer can’t upload directly to a remote peer, but the remote peer must start to download the local file. This remote operation can be achieved through the GridTorrent control interface. As for data replication, the control interface can be used to build a mechanism for data transfer orchestrations. Consider the scenario, where multiple GridTorrent daemons are deployed throughout the GREDIA infrastructure. In that case, a GridTorrent control interface client can be used to orchestrate data transfers from daemon to daemon and develop a data replication scheme.

4.3.1. Implementation details

In order to implement the above features, GridTorrent was enhanced some components. As shown in Figure 6, the previous implementation is scaled up, by adding some extra layers to integrate the daemon interface, handle multiple file downloads and manage remote control requests. These components are the following:

- **GTManager,** is the main component of GridTorrent. This component interacts with apach commons daemon library to support daemon start/stop operation and invokes all the other components of GridTorrent. Following the remote requests coming from the control interface, GTManager invokes a PeerManager for every file download, which in turn initiates all the necessary components to execute the torrent download, as described in the previous deliverable.
- **GTCtrlServer,** is the server component that listen for new control connections. This component accepts the incoming connections and then executes the GSI handshake to authenticate and authorize the new control connection. If the GSI handshake is successful, the incoming connection is passed to the following component of GridTorrent, CtrlConnection.
- **CtrlConnection,** is the component associated with a control connection. As soon as a control connection is accepted, authenticated and authorized, it is associated with an instance of this component. This instance is responsible of receiving requests, process them and sending the results. The support commands are the following:
    - Start a new download
    - List active downloads
    - Stop an existing download
    - Delete a file from local file system
    - Request statistics about send/recv messages and throughput
    - Shutdown the server

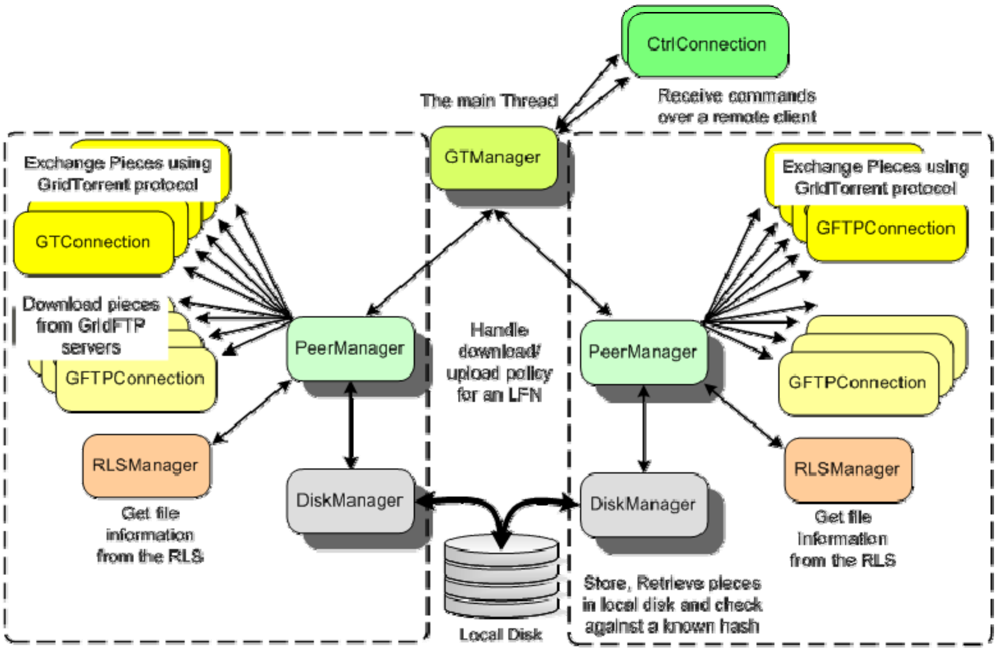

CtrlConnection
Receive commands over a remote client
The main Thread
GTManager
Exchange Pieces using GridTorrent protocol
GTConnection
Download pieces from GridFTP servers
GFTPConnection
PeerManager
RLSManager
Get file information from the RLS
DiskManager
Handle download/ upload policy for an LFN
Local Disk
Exchange Pieces using GridTorrent protocol
GFTPConnection
PeerManager
GFTPConnection
RLSManager
Get file information from the RLS
DiskManager
Store, Retrieve pieces in local disk and check against a known hash

Figure 6 The GridTorrent componets with the daemon and control connection interface features

4.3.2. Deployment scenario

Using the newly introduced features, we can describe a deployment scenario of GridTorrent. This scenario is depicted in Figure 7. In this scenario we have the following entities:

- The DRLS Web Service, which has the xml describing the file metadata, as well as a list with all the peers that participate in this file download
- GridTorrent daemons that act as seeds, meaning servers that have the actual file
- GridTorrent daemons that act as leechers, meaning servers that want the file
- GridTorrent client, meaning a thin client that can issue commands to remote GridTorrent daemons using the control interface

From this Figure, it becomes obvious that the data transfer component that is installed at various (if not all) sites of interest, interacts with two basic middleware components: The DRLS overlay in order to locate-update data location information and the storage servers themselves in order to get/put data.

This scenario starts with the seed having the file locally stored. When the seed initiate operations it will register the file to the DRLS. Then the GridTorrent client connects the three leechers and issues a “start download” command. Next, the leechers get a list of the active peers from the DRLS and connect to the seed to download pieces of the file. After a while, the leechers discover each other and they exchange the downloaded pieces with each other. Finally, the file is replicated to three more GridTorrent daemons, with a fast, secure, distributed and cooperative mechanism.

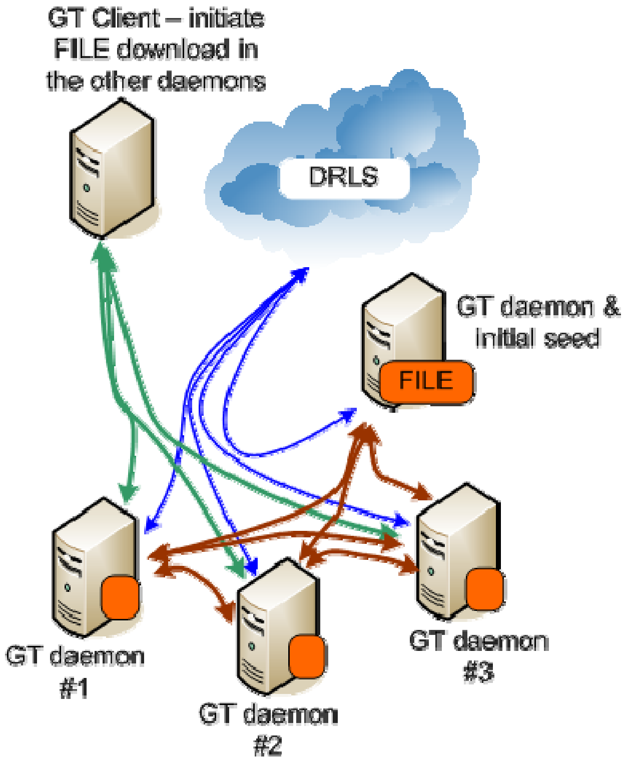


Figure 7 GridTorrent deployment scenario

5. Framework for intelligent Virtual Organizations

5.1. Overview

The major issues addressed by FiVO framework in the GREDIA project include:

- Provision of a unified semantic interface (at least at the service level) for discovery and management of all aspects of a Virtual Organization (including its members, agreements, resources, goals)
- Oriented strictly towards Service Oriented Architectures and Grid computing, thus assuming certain requirements on the infrastructures of organizations willing to participate in Virtual Organizations created on the GREDIA platform
- Allowing both static (manual) and dynamic (semi-automatic) creation and deployment of Virtual Organizations aiming to pursue some goal
- Support legacy information systems by mediating semantic queries into proper standards in which these systems are implemented

In overall the FiVO framework serves as a distributed system for negotiation, creating and managing resources and services within Virtual Organizations spanning heterogeneous real organizations. This is depicted in a sample deployment of FiVO framework within the GREDIA system in below.

5.1.1. Contract Ontology

The Contract Ontology in Gredia constitutes of several ontologies that together provide necessary concepts and relations for defining contracts among partners within a Virtual Organization. The general overview of the Contract Ontology and how it can be used with domain ontologies for definition of a Virtual Organization is depicted below.

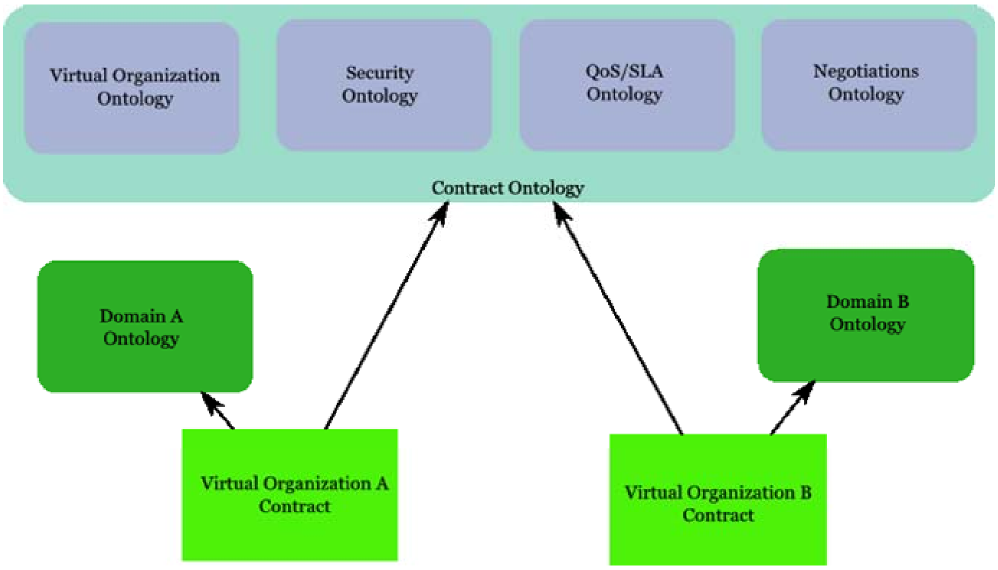


Figure 8: General structure of Contract Ontology

The Contract Ontology provides basic, general concepts that enable definition of a contract between partners in a VO, which combined with domain ontologies providing refinements and extensions of general concepts enables the contract to describe actual resources and functionalities within some domain to which the Virtual Organization is related.

Current ontology URL is:

http://zeus72.cyf-kr.edu.pl/gredia/ontologies/ContractOntology.owl

The main components of the Contract Ontology are:

⇨ VO Ontology

Provides generic concepts for description of Virtual Organizations such as VO, Organization or Actor.

Aspects covered by this ontology include:

- Allow to describe the concept of VO with its participants, administrators,
- Allow to describe organizations that will be parts of the organizations
- Allow to define roles of the organizations within a VO
- Allow to define users in the organization
- Allow to define resources in the organization

⇨ Security Ontology

This ontology provides generic concepts that are independent of existing security infrastructures, thus allowing to configure any security framework based on contract statements defined using these concepts.

Aspects covered by this ontology include:

- Allow to define general security assertions that can be translated to various security frameworks (e.g. CAS, PERMIS, Akenti, etc..), thus provide a generic super-set of assertions concepts that can be used in these systems. Currently the system is handles PERMIS authorization framework.
- Allow to refer to resources available in the system and to use concepts from domain ontologies (e.g. through URI's)

⇨ QoS Ontology

QoS and SLA ontology provides concepts that enable specification of non-functional requirements on resources sharing within a VO such as AvailabilityTime or ResponseLatency.

Aspects covered by this ontology include:

- Allow to define non-functional requirements on resource sharing such as e.g. performance metrics

⇨ Negotiations Ontology

This ontology provides concepts for the process of negotiations such as ContractStatement.

Aspects covered by this ontology include:

- Allow to define contract statements and their attributes

- Allow to track the negotiation process

5.2. Contract Negotiation Mediator

5.2.1. Contract Negotiation Component

Within the FiVO framework deployed in a given organization there may be many NegotiationsManager components. Each NegotiationsManager component is responsible for only one of the negotiations. FiVO component creates and destroys NegotiationsManagers and further directs the requests to proper NegotiationsManagers. It also receives and processes the requests when there is no NegotiationsManager component to which the request can be directed e.g. when the negotiations have not been created yet. An elected component from NegotiationsManager components, called NegotiationsManagerServer, works as a server for particular negotiations. ServerNegotiationsManager can be created in every FiVO component when one of the organizations starts the negotiations and takes the role of a VO manager. The server component stores the state of the negotiations, manages and controls the negotiations process. It receives all the requests from other NegotiationsManager components nad processes them. It also notifies other NegotiationsManager components of the changes in the negotiations state.

5.2.2. Contract Negotiation Protocol

This section describes the protocol for system activities related to management of negotiations. This includes management of negotiation tables, contract acceptance and GUI operations. A negotiation table represents a subset of the negotiation process in which the negotiations can be independent from other negotiation tables within the whole negotiation process of one VO. Thus each contract negotiation is performed on at least one negotiation table.

5.2.2.1 Creating negotiation table

Phase	Description
GUI Action	User creates the table
Request Creation	Request data sent: unique table name, table description (optional), table creator (organization)
Verification	Table name uniqueness
Processing	Table is added to the negotiations, the creator is stored

5.2.2.2 Cancelling negotiation table

Phase	Description
GUI Action	The creator of the table cancels the table with the intent to discard all statements (whether open or accepted) on the table
Request Creation	Request data sent: unique table name, table creator
Verification	Table existence, check if user (organization) is the creator of the table
Processing	Table is canceled, all statements on the table are removed

5.2.2.3 Accepting negotiation table

Table needs to be accepted by all table participants, that is all organizations that either created statements on the table or were referred to in the statements.

Phase	Description
Preliminary conditions	All statements on the table have to be accepted.
GUI Action	The table participant (organization for which there are statements on the

	table) accepts the table.
Request Creation	Request data sent: table name
Verification	Table existence, check if user (organization) is the table participant, concurrent modification prevention (check if the table has been modified before)
Processing	Table is accepted by the organization. If it is the last organization that has to accept the table, the table is permanently accepted. Statements from the table are added to the final contract.

5.2.2.3.1 Modifying contract during negotiation table acceptation

The table acceptation process has to be started over again if the contract has been modified (e.g. the statement added)

Phase	Description
Preliminary conditions	All statements on the table were accepted, some organizations accepted the table
GUI Action	User modifies the contract part related to the table (e.g. adds a statement to the table or modifies existing one)
Request Creation	*(same as modifying contract request)*
Verification	*(same as modifying contract request)*
Processing	*(same as modifying contract request)* and all previous acceptations on the table are invalidated. The table has to be accepted by all table participants again.

5.2.3. Starting negotiations

All activities performed when the negotiations are started

5.2.3.1 Creating negotiations

Actor	Action
User	Creates negotiations, specifies negotiations unique name and description. The name has to be unique within the organization
GUI	Forwards the request to FiVO container
FiVO container	Verifies the request to start negotiations (e.g. checks if the name is unique within the organization)
FiVO container	Creates Server FiVO component within FiVO container
Server FiVO	Initializes negotiations within Server FiVO. Assigns unique ID to negotiations (the ID is constructed from organization ID and (partially) unique negotiations name
Server FiVO	Notifies other FiVO components of new negotiations

5.2.3.2 Inviting participants

Actor	Phase	Action
GUI	Preliminary conditions	Negotiations created, domain ontologies loaded
VO Manager	GUI Action	Loads a list of participants
GUI	Request Routing	Forwards request to proper Server FiVO within FiVO container
Server FiVO	Processing	Adds participants to the negotiations

Server FiVO	Processing	Puts negotiations in the state of adding participants (no further ontology changes are allowed)
Server FiVO	Processing	Sends request to begin negotiations to FiVO containers of the organizations participating in negotiations. Sends negotiations data (name, description)
FiVO container	Notifications	Creates FiVO component for negotiations
FiVO	Notifications processing	Requests additional negotiations data from Server FiVO (i.e. participants list, domain ontologies)
Server FiVO	Notifications processing	Responds to FiVOs' requests for additional negotiations data
FiVO	Notifications processing	Notifies GUIs of new negotiations

5.2.3.3 Connecting GUI to FiVO

Actor	Action
GUI	Displays a list of current negotiations in which the organization is participating. Negotiations for which there is no GUI connection are marked.
User	Connects his GUI with negotiations.
GUI	Sends request to connect to negotiations
FiVO	Verifies request (checks if there is no GUI connection, or if current connection is not responding)
FiVO	Associates GUI with this FiVO component
FiVO	Notifies other GUIs that the negotiations have GUI connection
GUI	Connected GUI retrieves current state of the negotiations from FiVO
FiVO	FiVO sends to GUI the negotiations state (contract statements and their acceptation state, tables)

5.2.3.4 Disconnecting GUI from FiVO

Actor	Action
User	The user is participating in negotiations (GUI connected to FiVO). User decides to disconnect his GUI from FiVO.
GUI	Sends disconnect request to FiVO
FiVO	Disassociates GUI with this FiVO component
FiVO	Notifies other GUIs that there is no GUI connection for the negotiations

The GUI connection may be lost due to events not related to the system e.g. GUI crash, machine crash or network failure. In such a case there has to be a possibility of reconnection.

5.2.3.5 Loading domain ontology

Actor	Phase	Action
GUI	Preliminary conditions	The negotiations were created but have not been started yet (No contract statement has been added, no participant except for VO Manager has been added)
VO Manager	GUI Action	Loads domain ontologies
Server FiVO	Verification	Verifies the ontologies
Server FiVO	Processing	Loads the ontologies

5.2.3.6 Starting negotiations

Phase	Action
Preliminary conditions	The negotiations were created, domain ontologies loaded, organizations participating in negotiations invited
GUI Action	Starts negotiations
Verification	The negotiations were created, domain ontologies loaded, organizations participating in negotiations invited
Processing	Begins negotiations

5.2.4. Ending negotiations

All activities performed when the negotiations are ended

Actor	Phase	Action
VO Manager	GUI Action	The VO Manager ends negotiations (either accepts the contract or discards it)
GUI	Request Routing	Sends the request to Server FiVO
Server FiVO	Verification	Checks if user is authorized (VO Manager)
Server FiVO	Processing	Blocks all requests, requests are discarded and errors are returned
Server FiVO	Processing	Additional processing specific to contract acceptation or discarding *(see below)*
Server FiVO	Notifications	Sends notifications to all FiVOs (including Server FiVO) together with the negotiations result (acceptation or discarding)
FiVO	Notifications processing	Notifies GUI that negotiations have finished
FiVO	Notifications processing	Disconnects GUI from FiVO
FiVO	Notifications processing	Requests FiVO Container to be disposed
FiVO Container	Notifications processing	FiVO specific to the negotiations is disposed

5.2.4.1 Contract acceptation

(additional activities for "ending negotiations" sequence)

Actor	Phase	Action
GUI	Preliminary conditions	All contract statements have to be accepted, all tables have to be accepted or canceled.
Server FiVO	Verification	Concurrent modification prevention (check if the contract has been modified before – the acceptation will cause acceptation of an unknown statement)
Server FiVO	Processing	Final contract is prepared, the result of negotiations is communicated to external systems, final contract is sent to external systems

5.2.4.2 Contract discarding

(additional activities for "ending negotiations" sequence)

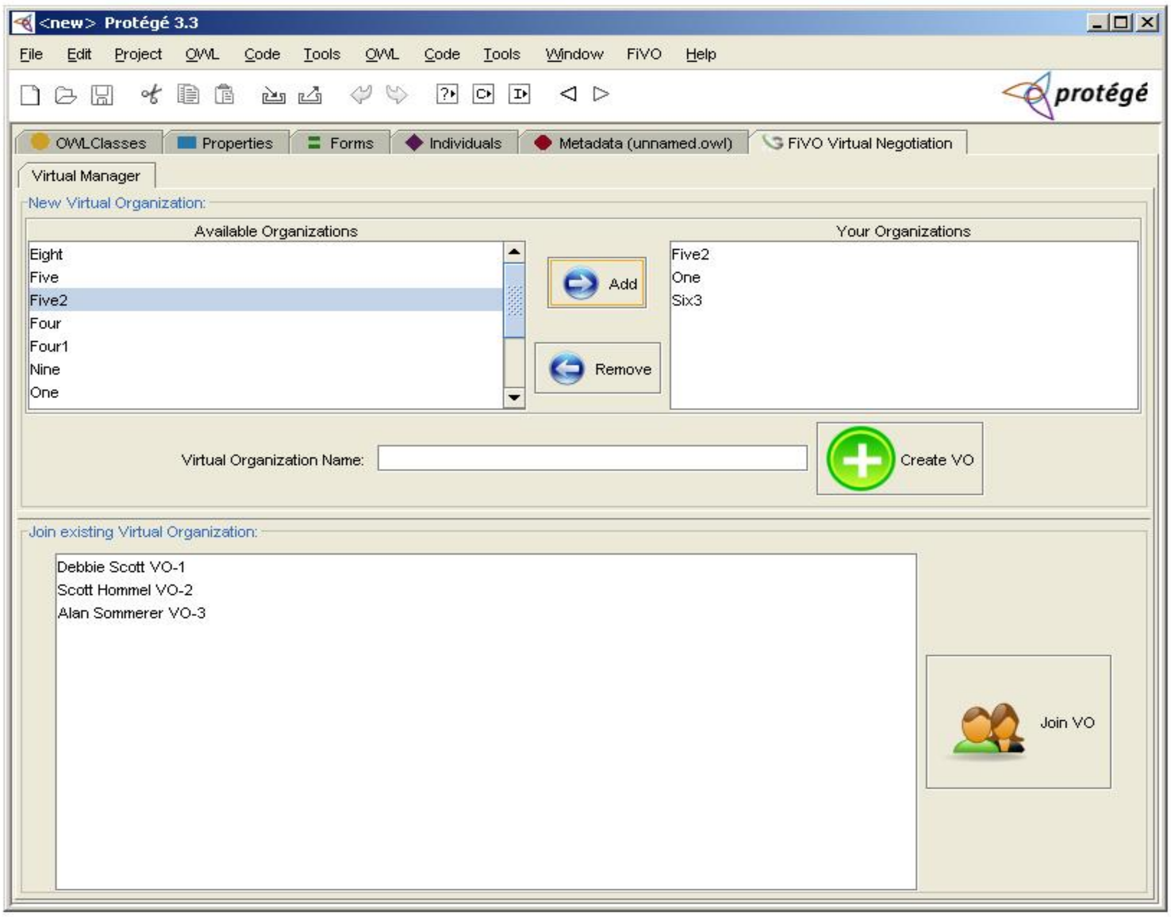
Actor	Phase	Action
GUI	Preliminary conditions	The statements and table can be in any state
Server FiVO	Processing	The result of negotiations is communicated to external systems

5.2.5. Contract Negotiation User Interface

Graphical user interface enables the user to utilize the functionality of the system. Graphical interface uses the interface provided by FiVO. FiVO component notifies graphical interface about events that take place during system operation, that influence the content presented to the user. Graphical interface communicates with NegotiationsManager during negotiations.

Though the overall system architecture is peer-to-peer, during the negotiations one component is elected to act as a server. In every FiVO component NegotiationsManager component is created that offers single negotiations functionality. In the organization of VO Manager, who created the negotiations, ServerNegotiationsManager is created that offers additional functionality of a server. All NegotiationsManager issue their request directly to the server.

The preliminary implementation of FiVO GUI is implemented as a plug-in for Protege 2000 ontology editor (Figure Figure 9).



Available Organizations: Eight, Five, Five2, Four, Four1, Nine, One

Your Organizations: Five2, One, Six3

Join existing Virtual Organization:

Debbie Scott VO-1
Scott Hommel VO-2
Alan Sommerer VO-3

Figure 9: FiVO plug-in to Protege 2000 for contract negotiations

5.3. FiVO Installation

This section describes the steps necessary in order to deploy a FiVO instance within a organization.

5.3.1. Requirements

Ant - download Ant from http://ant.apache.org/ (tested with version 1.7.0)

Java - download Java JDK from http://java.sun.com/ (tested with version 1.6.0)

JAXB - download JAXB RI from https://jaxb.dev.java.net/ (tested with version JAXB version 2.0). JAXB is only needed for generating XML configuration classes from package eu.gredia.fivo.component.config. FiVO component will not work when using JAXB API 2.1.

Tomcat - download Tomcat from http://tomcat.apache.org/ (tested with version 6.0.16)

Axis2 - download WAR distribution of Axis2 from http://ws.apache.org/axis2/ (tested with version 1.3). FiVO component will not work when using different versions of Axis2 (v.1.4 uses JAXB API 2.1)Install Axis2 by copying the axis2.war from the distribution to $CATALINA_HOME/webapps directory and restarting Tomcat.

Spring - tested with version 2.5.1

5.3.2. Installation of FiVO Contract Negotiation Service

This section describes the steps necessary for installation of FiVO Contract Negotiation service.

Building

1. Create local configuration file in FiVOComponent/config/<user.name>/build.properties file. <user.name> is the user name resolved by JVM
2. Use FiVOComponent/config/build-template.properties as a template and provide necessary properties in local build properties file
3. Use run deploy.libraries-for-axis ant task from FiVOComponent/build.xml to copy necessary jars to Axis2 web application.
4. Run all task from FiVOContractCore/build.xml. This should produce FiVOContractCore/FiVOContractCore.jar
5. Run generate.service task from FiVOComponent/build.xml
6. If you need to regenerate JAXB created classes do the following steps
7. Run generate.config.classes from FiVOComponent/build.xml

FiVO component configuration

1. Create FiVOComponent/resources/<user.name> directory
2. Find FiVOComponent/resources/templates directory and use template files in this directory further in the configuration process
3. Create organization.xml file in created directory defining the organization for which this FiVO component instance is deployed. organization.xml file specifies unique organization name and available resources. organization.xml should conform to FiVOComponent/resources/organization.xsd. Use organization-template.xml as a template
4. Create remote-config.properties file in created directory and supply appropriate properties. Use remote-config-template.properties as a template.

5. Create config.properties file in created directory and supply appropriate properties. Use config-template.properties as a template
6. Optionally create fivoComponents.xml file defining Web Service endpoint addresses of all known FiVO component instance (including this one). fivoComponents.xml must conform to FiVOComponent/resources/fivoComponents.xsd. Use fivoComponents-template.xml as a template. There can be only one fivoComponents.xml file for entire FiVO components network (but it has to be accessible for all components)

Deployment

1. Important Note: do not start Tomcat with FiVO component deployed. Deploy FiVO component after Tomcat startup. Use undeploy.service to undeploy FiVO component from Tomcat
2. Optionally run task deploy.fivo-config from FiVOComponent/build.xml
3. Run task start.rmiregistry to start RMI registry for communication between FiVO component and GUI (external RMI registry can be used provided that it has access to all class files)Optionally rmi registry can be started manually using: ‘rmiregistry 10991 &’
4. Run task deploy.service from FiVOComponent/build.xml
5. Run taks undeploy.service to undeploy FiVO component from Tomcat
6. Important Note2: do not undeploy and redeploy FiVO component with code changes. There exists a risk of old objects, cached by Tomcat, being still used.

5.4. FiVO evaluation

This section presents a sample evaluation of the negotiation process performed using the FiVO framework. Please note that it presents somewhat extended use case of HappyBank in order to allow for evaluation of all aspects of the negotiation process handled by the FiVO framework.

5.4.1. Scenario overview and goals

Let's assume that some bank is willing to provide a loan evaluation service to the clients. The bank cannot provide this service on its own, but has to outsource some aspects of this operation to various third-party organizations. The bank thus needs to create new Virtual Organization called HappyBank composed of many cooperating organizations providing services to each other. Cooperation of these organizations will eventually provide clients with the loan evaluation service.

5.4.2. Actors and resources

Euro Bank

This is the bank which invented the idea of HappyBank Virtual Organization and also the administrator of the negotiation process. It is going to make final decisions about Happy Bank. It is an international bank that is trying to invest some money in the local market. It can provide actual money for the clients' loans but has a few offices in which clients can handle their businesses and talk to the consultants.

It provides:
Money Source – actual money the clients receive will come from this bank

Easy Bank

This is a bank cooperating with Euro Bank. It has various services that Euro Bank doesn't have especially regarding customer services. It has many offices in which clients can talk to the consultants.

It provides:
Customer Care – Easy Bank can handle communication with clients. Clients can easily go to one of the Easy Bank's offices and physically sign the aggreements and get consultances regarding loans
Account Service – provides clients with their individual loans accounts tracking balance of their loans. It also gives online access to the account's history

National Bank

NationalBank is a huge bank with long traditions however with high administrative costs. The services of this bank are thus expensive.

It provides:
CustomerCare – it has many offices developed troughout the years of National Bank's operation. It can provide high quality customer care

Direct Consultants Ltd.

This company is cooperating with various banks. It collects offers from many banks and it is able to provide clients with offers that best suite their particular needs. It also provides convenient method for clients to handle their banking businesses by phone or direct consultations in the client's location.

It provides:

Direct Customer Care – clients can sing all the agreements and handle all banking related issues during appointed meetings with the consultants in the place selected by client

Credibility Assesment Center

This company specializes in the assesment of client's credibility during loans approval process. It collects data on loans and debts from third-party agencies and gives asking organization the estimate of how risky is particular loan aggreement.

It provides:
Loans Consultation – assesment of loans risk based on loans parameters provided by bank

Consulting Center

This is a competeing organization to Credibility Assesment Center. It also provides risk estimates for loans and collects loans and debts data from other organizations. It is a fairly new company so it is willing to provide cheaper services to encourage bank clients to buy its services.

It provides:
Loans Consultation – assesment of loans risk based on loans parameters provided by bank

National Loans Agency

This is a national agency collecting information on loans and debts. Its purpose is to help companies identify untrustworthy clients and limit the risk of giving various types of loans. Companies providing loans are not obliged to send their loans data to the agency. They sometimes do it in good faith to protect their businesses. On the other hand they refrain from disclosing confidential and business information. The information contained in the agency are therfore incomplete and not fully trustworthy

It provides:
Loans Registry Service – provides information on all recorded loans of a client.

Independet Loans Registry

This company provides similar service to the National Loans Agency. Banks are more willing to disclose their confidential data to this company as it is restricted by severe penatly fees in case of confidential data disclosure. It also provides better quality of service. The information contained in the registry are also incomplete (but they are not a subset fo those contained in National Loans Agency)

It provides:
Loans Registry Service – provides information on all recorded loans of a client.

5.4.3. Simplified Happy Bank Scenario

This sections contains simplified Happy Bank scenario in which only four organizations are negotiating. The number of organizations negotiating was limited for clarity and easier presentation.

Actors and resources

In this scenario only the following organizations are negotiating:

- EuroBank
- EasyBank
- ConsultingCenter
- CredibilityAssessmentCenter

Scenario sequence

Type	Author	Who Provides	What	Parameters
1	EuroBank	**--- Creates table (1) ---**		
EuroBank wishes to discuss bank related issues with EasyBank so it craetes new table for this purpose				

	EuroBank	EuroBank	**MoneySource**	MaxMoneyPerMonth=30M\$
				MaxMoneyPerClient=1000\$

EuroBank obliges itself to provide money for the loans. *It limits maximum amount of money it can provide for this purpose per month.* *It also limits maximum loan amount it feels comfortable with (the amount which, in the opinion of EuroBank, is not too risky)*				

	EasyBank	EuroBank	**MoneySource**	MaxMoneyPerMonth=40M\$
				MaxMoneyPerClient=2000\$
				ResponseTime=1day

EasyBank estimates on its own the needs of potential HappyBank's clients and, counting on more profits, rises the maximum amount of money clients can borrow from HappyBank. *This also enforces the rise of maximum amount of money EuroBank is providing for loans.* *EasyBank also states the requirement on EuroBank that it should provide money for individual loan in no longer than 1 d*				
EuroBank is content with the negotiations state regarding MoneySource and it proceeds with negotiating other resources needed by virtual HappyBank.				

	EuroBank	EasyBank	**CustomerCare**	ClientsPerMonth=30k
				MonthlyPrice=1M\$

EuroBank asks EasyBank to handle customer care for the purpose of new loans service. *It estimates numer of clients per month which will have to bo served by EasyBank's employees.* *It also states how much it is willing to pay for this service*				
	EuroBank	EasyBank	**AccountService**	
EuroBank asks EasyBank to provide individual accounts for clients who take loans. *It doesn't state the price for this service, EasyBank is supposed to provide the price which it feels appropriate for this kind of service.*				

	EasyBank	EasyBank	**CustomerCare**	ClientsPerMonth=20k
				PerTransactionCommission=20 %

In response to EuroBank's request EasyBank states that it cannot handle as mach clients per month as EuroBank would like. *It also states different kind of payment it wishes for this service. EeasyBank wishes to be paid a commission from the amount of every loan transaction*				
	EasyBank	EasyBank	**AccountService**	MonthlyPrice=1M\$
In response to EuroBank's requests EasyBank states the price for the individual accounts it is going to provide for clients. It takes into account the supposed amount of HappyBank clients.				
EuroBank is content with the negotiations state regarding AccountService and it proceeds with negotiating other resources needed by virtual HappyBank.				

	EuroBank	EasyBank	**CustomerCare**	ClientsPerMonth=25k
				MonthlyPrice=0.5M\$
				PerTransactionCommission=10 %

It turned out that EasyBank is wishing more money for customer service than EuroBank had expected. EuroBank is also dissatisfied with high commission as it would like to pay a fixed price for this service. So EuroBank lowers the commission and sets a lower fixed price for this service.				

EuroBank also rises the amount of clients per mounth EasyBank is obliged to handle, as it expects more clients to use loan services of virtual HappyBank.				

	EasyBank	EasyBank	**CustomerCare**	ClientsPerMonth=25k
				MonthlyPrice=0.5M$
				PerTransactionCommission=15 %

EasyBank finally agrees to the number of clients it is going to handle but it demands higher commission value. Other service conditions are left intact				
EuroBank is not very content with CustomerCare service pricing but, having no other option, is forced to accept conditions of EasyBank.				
	EuroBank	**--- Accepts table (1) ---**		
EuroBank doesn't have any other issues to discuss with EasyBank and agrees to all the statements on the table, so it accepts the table.				
	EasyBank	**--- Accepts table (1) ---**		
EasyBank sees that EuroBank accepted the table so it also accepts the table				

EuroBank negotiating with consulting companies

Type	**Table**	**Author**	**Who Provides**	**What**	**Parameters**
1	2	EuroBank	**--- Creates table (2) ---**		
1	3	EuroBank	**--- Creates table (3) ---**		
EuroBank would like to minimize the risks of giving loans to the clients so it wishes to pay for consulting services from consulting centers. There are two organizations providing this kind of service so EuroBank creates two tables, one for each organization					
	2	EuroBank	Consulting Center	**LoansRiskAssessment**	MaxResponseTime=1day
	3	EuroBank	Credibility Assessment Center	**LoansRiskAssessment**	MaxResponseTime=1day
EuroBank independently requests LoansRiskAssessment service from two organizations. It is going to choose the best offer. *EuroBank doesn't want virtual HappyBank's clients to wait too long for loan approval decisions so it specifies requirement on the service that the maximum time from sending loan data to receiving risk estimates is no longer than 1day*					
	3	Credibility Assessment Center	Credibility Assessment Center	**LoansRiskAssessment**	MaxResponseTime=3days AverageResponeTime=2days MonthlyPrice=1.0M$
EuroBank receives first reply from Credibility Assessment Center on table 3. *CredibilityAssessmentCenter cannot guarantee that they will always provide risk estimates in 1 day but they state that they can do it in 3 days. However on average their respone time can be not worse than 2 days.* *They also state their price for the service. Credibility Assessment Center have existed for a while now and they have stabilized position on the market. They can afford setting lower price to compensate longer response times.*					
	2	Consulting Center	Consulting Center	**LoansRiskAssessment**	MaxResponseTime=1day MonthlyPrice=2M$ GuaranteedAccuracy=90%

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td>PercentLossWarranty=50%</td></tr>
<tr><td colspan="6"><i>Consulting Center sees the answer from Credibility Assessment Center and tries to provide a competing ofer.
It can guarantee that they will provide risk estimates in no longer than one day.
However, as it is a new organization, it cannot set prices lower than their actual operation costs, so they set highter price for the service
Trying to compensate for higher prices they are offering some warranties. They state that their service will be accurate 90% of the time, and if it isn't they are going to pay 50% of the documented losses.</i></td></tr>
<tr><td></td><td>3</td><td>EuroBank</td><td>Credibility Assessment Center</td><td><b>LoansRiskAssessment</b></td><td><i>MaxResponseTime=3days</i>
<i>AverageResponeTime=2days</i>
MonthlyPrice=1.0M$
GuaranteedAccuracy=90%
PercentLossWarranty=50%</td></tr>
<tr><td colspan="6"><i>EuroBank now has two options to consider. It is going to ask Credibility Assessment Center, which previously said nothing about warranties, about the same warranties they were given by Consulting Center</i></td></tr>
<tr><td></td><td>3</td><td>Credibility Assessment Center</td><td>Credibility Assessment Center</td><td><b>LoansRiskAssessment</b></td><td><i>MaxResponseTime=3days</i>
<i>AverageResponeTime=2days</i>
MonthlyPrice=1.5M$
GuaranteedAccuracy=70%
PercentLossWarranty=20%</td></tr>
<tr><td colspan="6"><i>In respone Credibility Assessment Center states that thay can provide warranties for their estimates however the guaranteed accuracy is lower and they cannot return much money in case their client will experience some losses making loans decisions based on their estimates.
Also the price for the service is higher.</i></td></tr>
<tr><td></td><td>3</td><td>EuroBank</td><td colspan="3"><b>--- Cancel table (3) ---</b></td></tr>
<tr><td colspan="6"><i>Seeing that Credibility Assessment Center is not willing to provide as high quality of service as Consulting Center EuroBank decides to finish negotiations with Credibility Assessment Center. So it cancels the table on which negotiations with Credibility Assessment Center were taking place
EuroBank continues to negotiate with Consulting Center as the only provider of LoansRiskAssessment service</i></td></tr>
<tr><td></td><td>2</td><td>EuroBank</td><td>Consulting Center</td><td><b>LoansRiskAssessment</b></td><td><i>MaxResponseTime=1day</i>
<i>MonthlyPrice=2M$</i>
<i>GuaranteedAccuracy=90%</i>
PercentLossWarranty=60%</td></tr>
<tr><td colspan="6"><i>EuroBank would like to have as accurate prediction as possible so they are trying to change conditions of the warranty for better ones</i></td></tr>
<tr><td></td><td>2</td><td>Consulting Center</td><td>Consulting Center</td><td><b>LoansRiskAssessment</b></td><td><i>MaxResponseTime=1day</i>
MonthlyPrice=2.2M$
<i>GuaranteedAccuracy=90%</i>
PercentLossWarranty=60%</td></tr>
<tr><td colspan="6"><i>Consulting Center agrees to give better warranty for higher price</i></td></tr>
<tr><td></td><td>2</td><td>EuroBank</td><td colspan="3"><b>--- Accepts table (2) ---</b></td></tr>
<tr><td colspan="6"><i>EuroBank finally agrees to the price and warranty conditions of Consulting Center so it accepts the table</i></td></tr>
<tr><td></td><td>2</td><td>Consulting Center</td><td colspan="3"><b>--- Accepts table (2) ---</b></td></tr>
<tr><td colspan="6"><i>Consulting Center also has to issue the final agreement on the statements on the table</i></td></tr>
<tr><td></td><td></td><td>EuroBank</td><td colspan="3"><b>--- Finish negotiations ---</b></td></tr>
<tr><td colspan="6"><i>Because all tables were accepted and HappyBank has all services it needs for operation EuroBank, as the administrator of negotiations, finishes the negotiations and accepts the contract</i></td></tr>
<tr><td colspan="6"><i>Negotiations participants now become virtual organization members and they can view the contract, they were negotiation on.</i></td></tr>
</table>

Failure Scenario

This is a scenario that happend before EuroBank's negotiations with EasyBank. EuroBank had the same expectations regarding CustomerCare service however it was trying to make an agreement with NationalBank. NationalBank is a huge bank with long traditions however with high administrative costs. The services of this bank are thus expensive.

<table>
<tr><th>Type</th><th>Author</th><th>Who Provides</th><th>What</th><th>Parameters</th></tr>
<tr><td>1</td><td>EuroBank</td><td colspan="3"><b>--- Creates table (1) ---</b></td></tr>
<tr><td colspan="5"><i>EuroBank wishes to discuss bank related issues with NationalBank so it craetes new table for this purpose</i></td></tr>
<tr><td></td><td>EuroBank</td><td>NationalBank</td><td><b>CustomerCare</b></td><td>ClientsPerMonth=30k
MonthlyPrice=1M$</td></tr>
<tr><td colspan="5"><i>EuroBank asks NationalBank to handle customer care for the purpose of new loans service.
It estimates numer of clients per month which will have to bo served by EasyBank's employees.
It also states how much it is willing to pay for this service</i></td></tr>
<tr><td></td><td>NationalBank</td><td>NationalBank</td><td>CustomerCare</td><td>ClientsPerMonth=30k
MonthlyPrice=3M$</td></tr>
<tr><td colspan="5"><i>NationalBank accepts number of clients per month however it completely disagrees with the price proposed by EuroBank. It demands 3 times as much.</i></td></tr>
<tr><td></td><td>EuroBank</td><td>NationalBank</td><td>CustomerCare</td><td>ClientsPerMonth=20k
MonthlyPrice=2M$</td></tr>
<tr><td colspan="5"><i>EuroBank tries to negotiate with NationalBank. It lowers the expected amount of clients per month lowering the monthly price at the same time</i></td></tr>
<tr><td></td><td>NationalBank</td><td>NationalBank</td><td>CustomerCare</td><td>ClientsPerMonth=20k
MonthlyPrice=3M$</td></tr>
<tr><td colspan="5"><i>Even lowering the number of expected clients per month would not lower high operation costs of the NationalBank. NationalBank clearly states this setting the price back to previous value</i></td></tr>
<tr><td></td><td>EuroBank</td><td colspan="3"><b>--- Abort negotiations ---</b></td></tr>
<tr><td colspan="5"><i>EuroBank cannot accept such a high price. Having no other partners to negotiate with it aborts negotiations. The requirements of future virtual HappyBank cannot be satisfied without a bank partner.
So now EuroBank searches for new partners on the local market and finally finds EasyBank.</i></td></tr>
</table>

6. Conclusions – Future Work

In this deliverable we described our final efforts towards the implementation of the Gredia middleware. Gredia addresses the subject of efficient and reliable multimedia content sharing and management in a Grid environment, where heterogeneous resources cooperate. In this context, we proposed a service-oriented middleware architecture that provides store, search and retrieve primitives for manipulation of annotated multimedia files. We introduced the idea of DHT overlays for metadata and data management, thus avoiding the use of centralized entities. Moreover, a multidimensional indexing scheme that speeds-up the searching procedure and facilitates range queries is used to enable user-customized searches. Finally, we presented GridTorrent, a transfer protocol resilient to flash crowd conditions and completely compatible with existing Grid middleware, which will be used for multimedia file uploads and downloads. We strongly believe that the proposed architecture, incorporating various P2P techniques, will provide a robust and scalable infrastructure for storing annotated data and searching over a large set of metadata attributes.

The implementation of the described platform is currently at a finalizing mode. In more detail and concerning GridTorrent, we moved towards two directions: First, we created a daemon-like version that is installed on all relevant servers in our system. Second, we implemented a communication channel through which GridTorrent-enabled servers will be able to participate in multiple uploads and downloads (in a fashion similar to the GridFTP channel). As far as SFCs go, we now utilize it using a number of dimensions that take numerical, categorical and keyword values (the ones that will be mostly used in Gredia queries). Moreover, we investigated optimization issues in various aspects of our design. One aspect we took into consideration is load-balancing issues which arise from the heterogeneity of nodes in terms of storage capacity and bandwidth, as well as the locality preservation that Space Filling Curves can offer. We also considered various replication and performance optimization techniques to enhance the performance of the proposed middleware.

In the near future we plan to perform a series of thorough tests which will, in succession, show the scalability, robustness and performance of our middleware both as a whole and individually as well. Any adjustments on the final version of the Grid Middleware components, which will be necessary after the execution of the field trials in the media and banking domains, will be incorporated in the next versions of the GREDIA Integrated Platform Deliverables (D6.2 and D6.3 respectively).

7. References

[1] Cohen, B. (2003), *Incentives Build Robustness in BitTorrent*, Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA.

[2] The official site of Globus Toolkit, http://globus.org/toolkit.

[3] Allcock, W., Bresnahan, J., Kettimithu, R., Link, M., Dumitresku, C., Raicu, I., Foster, I. (2005), *The Globus Striped GridFTP Framework and Server*, in Proc. of the ACM/IEEE Conference on Supercomputing, SC'05.

[4] Java CoG Kit, http://www.cogkit.org/

[5] A. R. Butz. Alternative algorithm for Hilbert's Space Filling Curve. *IEEE Transactions on Computers*, C-20(4):424–426, 1971.

[6] Guohua Jin and John Mellor-Crummey. SFCGen: A framework for efficient generation of multi-dimensional space-filling curves by recursion. *ACM Trans. Math. Softw.,* V31-1, 2005: 120-148.