Multifaceted, Collaborative Access to Cultural Heritage

Project IST-034984

D3.1 MOSAICA Distributed Content Management System - Reference Architecture

Spec. document

Editor: Asdrúbal Costa and Sérgio Marques
Contributors: Maria Teresa Andrade and Sílvio Macedo

Dissemination Level: internal
Version: 4.0
Date: 16-Apr-07

Executive Summary

Document Objective

T3.1 Distributed multimedia content management. (Leader INESC, with participation of XLAB; WP5 partners involved in terms of validation and conformance to the MOSAICA overall functional requirements and specification)

Dependencies: T3.1 is dependent of T5.1 and T5.2 and has links with T3.2 and T3.3.

The goal of this task is to define the high-level architecture of the p2p-cms and to specify its constituent modules and interactions thereby. This work was initiated with a study of the existing most advanced approaches to content management solutions based on the p2p paradigm and with an analysis of the different p2p models, among the variants of decentralised and hybrid. In parallel a thorough analysis has been performed on the open source project JXTA and on the possibility of using Distributed Hash Tables (DHT) to implement content search operations. Finally and having also as input the MOSAICA requirements and componential design, identified in deliverable D5.1, and the description of the role of each MOSAICA's sub-system and correspondent interactions, an approach for the p2p-cms of MOSAICA was defined and the overall architecture designed based on the Service Oriented Architecture (SOA). The present document describes the results obtained with this work.

Targeted Audience

MOSAICA technological partners

Prerequisite

IST-034984 MOSAICA DoW document

Summary:

This document provides the specification of the reference architecture of the MOSAICA distributed content management system. Accordingly it presents the principles and concepts adopted for the design of the high-level architecture of the P2P Content Management System (p2p-cms) and provides a description of its functionality in terms of constituent modules, use cases, collaboration/sequence diagrams and also class diagrams for some components. It builds on the deliverable D5.1 that provides the initial functional specification and componential design of the MOSAICA's sub-systems.

The p2p-cms is the MOSAICA subsystem responsible for enabling the access to the MOSAICA resources (content and ontologies), which are distributed throughout the Web. It ensures high availability of data and satisfactory performance transfers by embracing the P2P paradigm and practices such as the use of replication mechanisms and structured overlay networks.

This deliverable will be complemented or superseded by the final deliverable of task 3.1, deliverable D3.2 "MOSAICA Distributed Content Management System – Services and Protocols". D3.2 is due at month 27, and will provide the complete description of the MOSAICA distributed content management system

© 2007 MOSAICA

This document contains proprietary information of MOSAICA and shall not be reproduced or transferred to other documents, disclosed to others, or used for any purpose other than that for which it is furnished, without the prior written consent of MOSAICA.

The trademark and service marks of MOSAICA, including the MOSAICA mark and logo, are the exclusive property of MOSAICA, and may not be used without permission. All other marks mentioned in this material are the property of their respective owners

Contents

1. Introduction 5
   1.1. Overview 5
   1.2. Abbreviations and acronyms 5
   1.3. Definitions 6
2. High level architecture 8
   2.1. Concepts and principles 8
   2.2. The role of the p2p-cms within MOSAICA 9
3. Peers Architecture 14
   3.1. Overall architecture 14
      3.1.1. JXTA Layer 15
      3.1.2. Core Layer 15
      3.1.3. Interface Layer 16
   3.2. Component diagram 16
   3.3. Web Services interfaces 17
4. P2P-CMS Data 19
   4.1. Distributed databases 20
      4.1.1. Database Timestamp Conflict Resolution 21
   4.2. DHT 23
      4.2.1. Basic concepts of DHT 23
      4.2.2. CHORD Algorithm 24
   4.3. DHT versus Distributed Databases 25
5. Framework Logical View 26
   5.1. Class Diagrams 26
      5.1.1. Interface package 26
      5.1.2. Content and Metadata package 27
      5.1.3. Users and Groups package 28
      5.1.4. Utils package 29
   5.2. Sequence Diagrams 30
      5.2.1. Login 30
      5.2.2. Register Content 31
      5.2.3. Search content 32
6. Conclusions and future work 33

Bibliography 34

1. INTRODUCTION

1.1. Overview

This document answers deliverable D3.1 of WP3 - *D3.1" MOSAICA Distributed Content Management System -Reference Architecture"* document.

It reports the work being developed under task 3.1 "Distributed multimedia content management" addressing the investigation of the possible approaches to implement the MOSAICA's p2p-cms, combining the use of Web Services technologies with the p2p paradigm and concepts. Different models and current implementations of p2p architectures were analysed, among the several variants of decentralized and semi-decentralized models. A semi-decentralised two-layer architecture has been selected. The top-layer is based on Web Services and JXTA protocols and it implements functionality to support distributed metadata-assisted content searches, users and services management using distributed relational data bases and a Service Oriented approach. The lower layer deals with the distributed storage of the MOSAICA digital objects, addressing aspects of content caching and replication, routing and location.

The high-level architectural and componential design of the P2P Content Management Systems (p2p-cms) is formally described in this document using UML. It includes a high-level component diagram (illustrating the existing sub-systems and their interactions), use cases, sequence diagrams and class diagrams. Class diagrams provide a logical view of the peers' API.

The p2p-cms is the MOSAICA subsystem responsible for enabling the access to the MOSAICA resources (contents and associated metadata), which are distributed throughout the Web. It ensures high availability of data and satisfactory performance transfers by embracing the P2P paradigm and practices such as the use of replication mechanisms and structured overlay networks.

This document is organised as follows. Section 2 describes the high-level architecture adopted for the MOSAICA p2p-cms, the concepts and principles behind it. It also provides indications of each layer's functionality and how is used within MOSAICA's usage scenarios. Section 3 presents the layered software architecture of the MOSAICA peer, identifying constituent modules and their interactions as well as exposed services. Section 4 describes the different types of content and metadata that circulate in the p2p-cms and describes two different alternative solutions to implement metadata-assisted content search. Section 5 provides functional and implementation details in the form of sequences and class diagrams for the p2p-cms modules.

1.2. Abbreviations and acronyms

BNC	BitTorrent Network Controller
DB	Database
CMS	Content Management System
CVS	Concurrent Version System
DHT	Distributed Hash Tables

OPS	Operations
P2P	Peer to peer
RSS	Really Simple Syndication
SHA	Secure Hash Algorithm
SOAP	Simple Object Access Protocol
UML	Unified Modelling Language
URI	Uniform Resource Identifier
URL	Universal Resource Locator
VE	Virtual Expedition
XML	Extensible Markup Language

1.3. Definitions

Axis	Java platform for creating and deploying web services applications.
Bootstrap	It refers to a process where a node connects to the network, providing its capabilities to the overall services.
BitTorrent	A peer to peer protocol that enables secure and multi platform content sharing.
Downloader	A peer in the Bit Torrent network that has only some, or none, pieces of a shared archive.
JXTA	Sun Microsystems Inc. open source technology for open, generalized peer to peer protocols that allows any connected device (cell phone to PDA, PC to server) on the network to communicate and collaborate.
Leecher	Designation of peers that are downloading a file from the BitTorrent network.
Peer	Network node that provides the same functionality as any other node.
Rendezvous peer	A peer in the JXTA network acting as an entry point into the network. The addresses of rendezvous peers are well-known.
Seeder	Designation of a peer that contains all the different pieces of a complete shared archive in the BitTorrent network.
Service	System component designed to support machine to machine interaction over a network, in order to achieve some predefined task.

Torrent	Small archive that has the necessary information to share a specific content over the BitTorrent network.
Tracker	In BitTorrent networks is a computer that coordinates file distribution
User	Person or process that uses the value delivered by the system.

2. HIGH LEVEL ARCHITECTURE

2.1. Concepts and principles

The p2p-cms presents a two-layered architecture based on two major networks. The upper layer, implemented with JXTA technology and Web Services, is responsible for the provisioning of functionality associated with content search, management of users, distribution and access to MOSAICA services and registration of new content into MOSAICA. The lower layer uses the Bit Torrent protocol and is responsible for the distributed storage of content, its replication and the access/reconstruction of user selected digital objects. Figure 1 presents this logical high-level architecture.

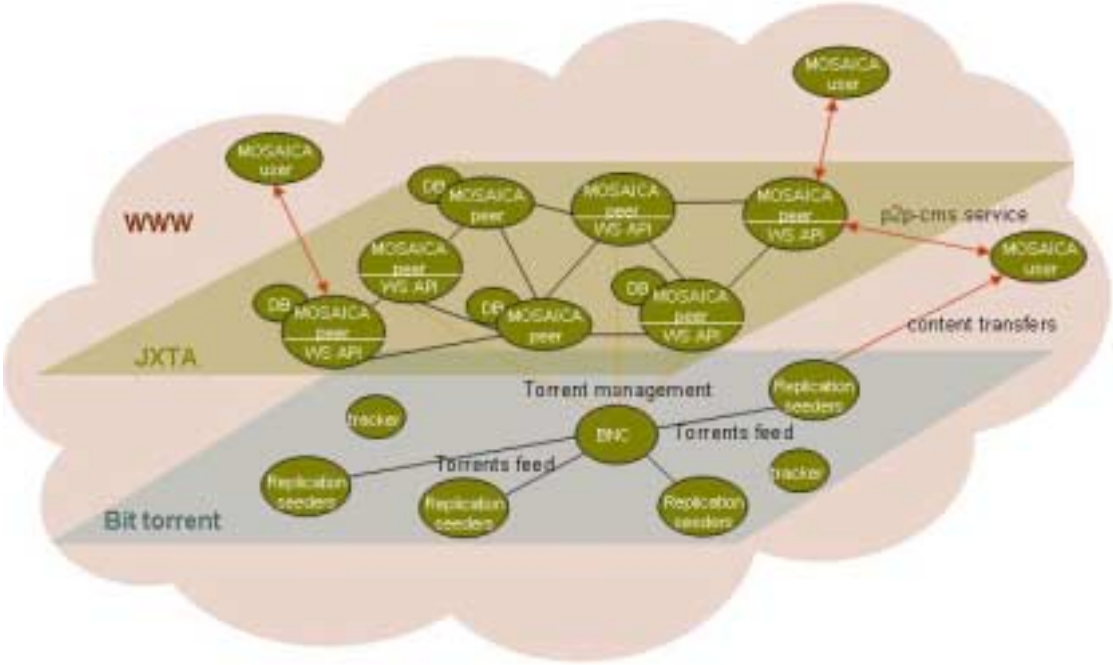

Figure 1 – p2p-cms architecture

The principle behind this architecture is that the operations related with the submission and search of digital objects using metadata and the access to MOSAICA services are functionally/logically separated from the operations concerning the distributed storage of the MOSAICA digital objects. Digital objects can be either media resources (e.g., videos, images, text documents, audio files) or ontologies.

The search is based on the distribution of metadata associated to MOSAICA digital objects and on the use of the ontology alignment service. It uses the concept of distributed data bases and transparent replication and distribution of MOSAICA services. When new content is to be submitted to MOSAICA, the p2p-cms receives the metadata associated to the new digital object and stores it in the distributed data base implemented in the JXTA network

The storage of digital objects and retrieval of selected content is based on p2p concepts using the Bit Torrent protocol. Digital objects being submitted to the MOSAICA system are partitioned into identically sized pieces (usually 256 Kbyte long), which are then distributed, stored and replicated across a network of peers. The location of the different pieces and consequent re-assembling of the complete digital object is possible through the generation and

use of a file designated of “torrent”. This file contains information about the different pieces, and about the peer that coordinates the digital object’s distribution. A peer that wants to obtain a given digital object must first obtain a torrent of that object. This is accomplished as a result of a successful search and selection of content using the p2p-cms upper layer functionality. After being in possession of the torrent file, the peer connects to the tracker specified in the torrent. The tracker then provides a list of peers from where the various pieces of the file can be downloaded.

As illustrated in Figure 1, this architecture defines *MOSAICA peers*, which differ from *MOSAICA users*. *MOSAICA users* can be simple clients using the semantic layer applications of MOSAICA (semantic annotator, semantic browser, Virtual Expeditions browser and GIS browser) that, without the need for specialised p2p software, can access and consume the MOSAICA distributed services exposed by the *MOSAICA peers* using standard Web Services technologies.

2.2. The role of the p2p-cms within MOSAICA

The p2p-cms is the MOSAICA subsystem responsible for enabling the access to the MOSAICA resources (content and ontologies), which are distributed throughout the Web. It ensures high availability of data and satisfactory performance transfers by embracing the P2P paradigm and practices such as the use of replication mechanisms and structured overlay networks.

To perform this role, the p2p-cms implements a 2-layer architecture as introduced in the previous section. Through the higher layer, it provides the management of metadata and the access to MOSAICA services in a distributed fashion. MOSAICA provides semantic, GIS and Virtual Expedition services that are replicated/distributed via the p2p-cms to ensure high availability of these services. The p2p-cms provides the means to hide this distribution from the user and the MOSAICA application layer, by exposing in one single interface the services provided by each replicated server. This is illustrated in Figure 2.

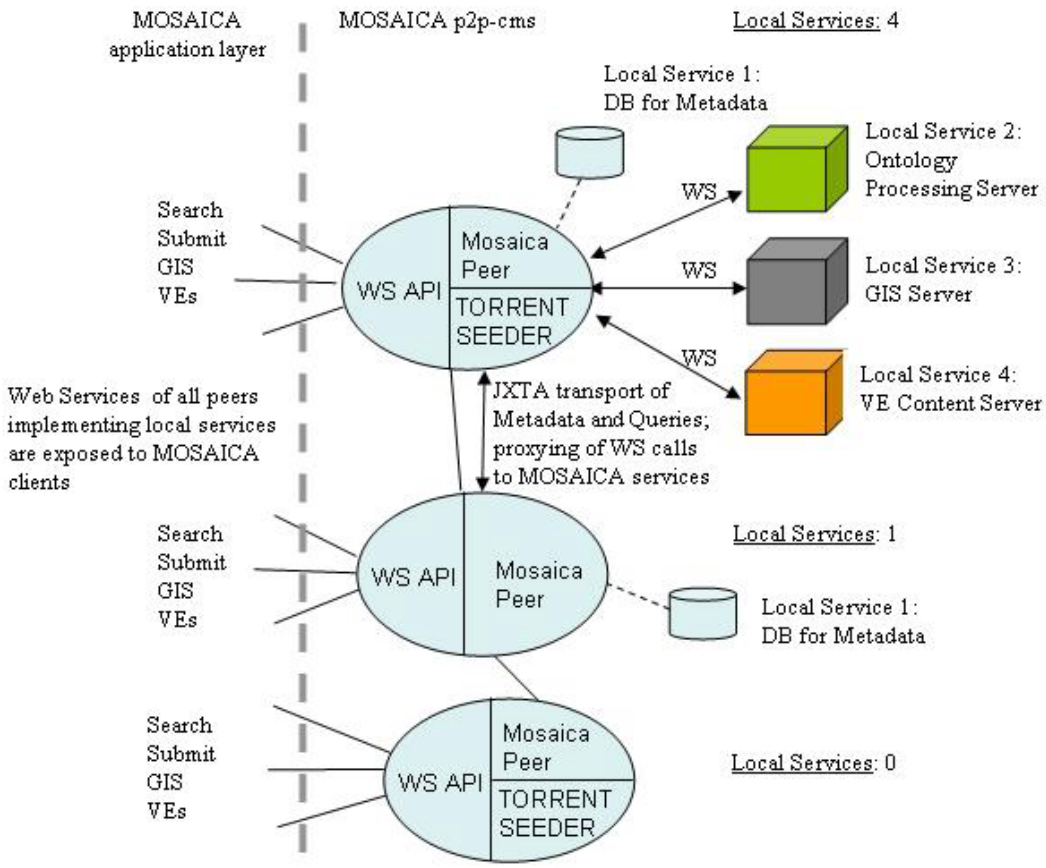

Figure 2 – Services exposed by the p2p-cms

As shown in figure 2, the p2p-cms exposes 4 different MOSAICA services to the application layer. The MOSAICA database that stores content-related metadata (semantic annotations and torrent information) to enable semantic based search in the p2p network and also users-related metadata to validate users into the system, is implemented in the form of distributed databases. This is a service that is directly implemented by the p2p-cms. The other 3 services are implemented by other MOSAICA sub-systems at the Semantic layer. Some of the MOSAICA peers in the JXTA network may have a local relational database that implements a part of the complete MOSAICA distributed database. Other peers may not have a local data base but may expose one or all of the other services. There may even be peers that do not have any of these 4 services but that may play a role at the Bit Torrent network level and at the JXTA level simply pass on to other peers the requests they receive.

For the definition of the services provided by the p2p-cms general use cases have been defined and the following relevant actors were identified:

- Administrator
- User
- Semantic browser
- Semantic annotator
- BNC (BitTorrent Network Controller)

The actors Semantic Browser and Semantic Annotator interact with the p2p-cms modules on behalf of the user, to allow him/her to register a new digital object or to initiate a semantic metadata-assisted search for content. For the sake of simplicity and considering that the ultimate consumer of the value delivered by the p2p-cms is indeed the user, in the remainder

of this section it will be assumed that the initiating actor is the user and not the modules of the Semantic layer.

The following **pre-conditions** are assumed to have been met for the successful implementation of each use case:

- A peer, providing a MOSAICA Web Service, is known outside the MOSAICA JXTA network;
- The user is identified
- The user has performed a successful login to the system

Figure 3 provides the diagram with the general use cases for the p2p-cms. These are then described in a succinct form in table 1, which provides also an indication of the initiating actor.

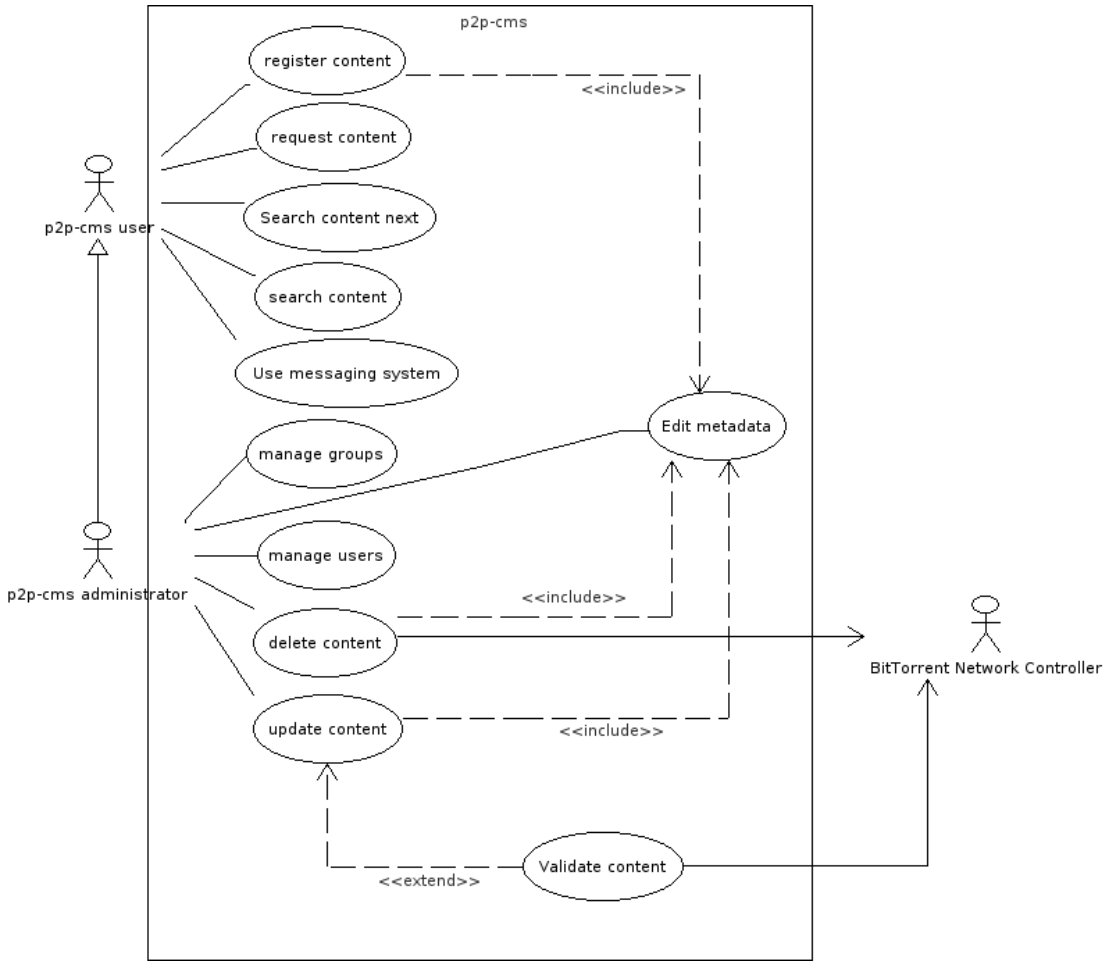

Figure 3 - General use cases diagram

Use case name	Overview	Initiating actor
Register content	The user inserts a torrent of the new content in the system. Torrent binary and content associated information is sent in XML. This torrent should be validated by an administrator. The torrent and semantic metadata associated to the content are uploaded to the distributed data base in the JXTA network.	User

Update content	The administrator could modify the information associated to the content. This could be non query-able (information present in the torrent), indexed data or annotations. Only administrators have the ability to change contents.	Administrator
Delete content	The user requests the deletion of existing registered content. The message must have a valid credential for content deletion.	User
Request content	The user requests content by its hash. The system should return a torrent file (only if the digital object referred in the torrent has been validated by an administrator).	User
Search content	The user sends a query composed by keywords and Boolean operators. XML description of the found contents should be returned	User
Search content next	Search next content	User
Manage groups	Groups management that includes creation, deletion, forum messages administrator and associated users and contents	Administrator
Manage Users	User management functionalities	Administrator
Edit metadata	User can insert, update or remove metadata indices from the system.	User
Validate	Administrator chooses to validate or deny content. The system provides him with an interface for that purpose.	Administrator

Table 1 - Use cases description

When a user wants to submit a new digital object that has been previously annotated using the MOSAICA semantic annotator, he/she should use a web application to create the torrent file and transfer data. The torrent file contains an "announce" section, specifying the URL of the tracker and an "info" section with the suggested names and length of the archive that contains the digital object, the length used for each piece in which the archive is partitioned and a SHA hash code for each piece.

Submission of a new digital object is then accomplished when the torrent file is uploaded to a peer inside the JXTA network. Its effective use as a sharable archive by MOSAICA clients becomes possible only after it has been validated by a MOSAICA administrator. Once the torrent file has been transferred to the BNC in the Bit Torrent network, replication servers are accordingly notified and the associated digital object can start to propagate from the notified replication servers, which thus act as the initial seeders for the newly submitted digital object. To increase performance and availability, several BNCs may be used to provide RSS feeds with torrent data to the replication servers.

As explained above, some of the MOSAICA peers in the JXTA network may have a local relational database, implementing a part of the full distributed MOSAICA data base. The information contained in each local database may not be completed or even updated. When a peer receives a query, it performs a local search into its own data base, if present, and forwards the query to other peers in the JXTA network to get additional information. The initiating peer re-assembles all the partial results and constructs the complete list of results to be returned to the user, matching his/her query. In case of conflict (e.g. different peers return different passwords for a user with unique ID) a timestamp is used to verify the most recent version of the associated information. The local database should then be updated with the most recent information received (identified by keys and tables).

Note that it is not necessary to have a database in all peers. Only peers explicitly configured to store and answer queries (normally *rendezvous* peers) should have a local database. The other peers send queries to the JXTA network and wait (a maximum amount of time) for answers.

Operations to insert, remove and update information on the database are propagated over the JXTA network using specific routing protocols.

3. PEERS ARCHITECTURE

3.1. Overall architecture

The p2p-cms has the goal of providing services for distributed content management using the p2p paradigm. To accomplish this task an API based on the JXTA suite of protocols and Web Services technologies has been design to implement the MOSAICA peers. With this approach, clients outside the MOSAICA JXTA network without the need to have specialised p2p software installed, communicate via Web Services to the MOSAICA peers, which in turn consume the JXTA p2p-cms services. On the other hand, this approach does not prevent the MOSAICA semantic-level applications (Browser, Annotator, VEs and GIS) to be deployed in the same machine as the MOSAICA p2p-cms modules, thus acting directly as a peer of the network.

MOSAICA peers are therefore peers belonging to the MOSAICA JXTA network and that enable the access to MOSAICA distributed services. MOSAICA peers may offer different levels of functionality, ranging from very simple peers that act as a sort of gateway, forwarding requests to more elaborated peers, to full-featured peers implementing/exposing all possible MOSAICA distributed services.

The functional architecture of a full-featured MOSAICA peer presents the a layered structure as illustrated in Figure 4.:

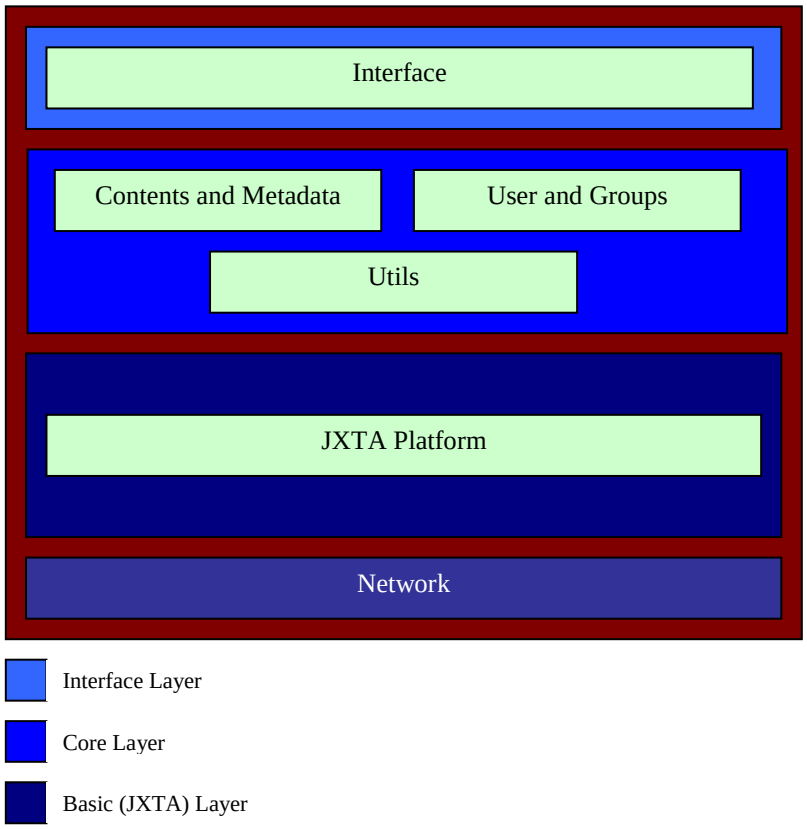

Figure 4 – Architecture of the p2p-cms peer's framework

The basic layer provides elements that are absolutely necessary within p2p implementation, with functionality that is not MOSAICA specific. In this layer the basic p2p protocols are defined and implemented using JXTA platform.

The core layer contains MOSAICA-specific logic for the manipulation of the data relative to users, groups, metadata and contents.

The third and upper layer is provided for the definition of interface classes that hide logic from lower layers. Web Services should use methods and functionalities from these classes.

3.1.1. JXTA Layer

In this layer, generic p2p functionality is implemented. Part of this functionality is even essential for running a p2p system, as is the case of the 'Transport' services. In many cases it will be possible to re-use existing modules available in the CVS repository of the JXTA platform project.

- **Security**: the security service permits secure communication and message authentication. It is crucial in services that need access control.
- **Groups**: the group service provides with an abstraction in the set of all the peers. Peers use the group service in order to organize them in groups of interest.
- **Transport**: it implements the message transport between peers. The principal network protocol that will be used is the TCP/IP but it is possible to use other network transport protocols such as HTTP or UDP.

3.1.2. Core Layer

This layer provides the functionality specific to the MOSAICA requirements that is not directly accessed by the user but that enables the p2p-cms to deliver the required value to its external actors (users or other MOSAICA subsystems).

- **Content and Metadata:** this module implements operations related with the management of MOSAICA content or digital objects (media resources or ontologies). As such it provides the required functionality to receive and process requests to register and upload new digital objects and queries to search for content. Since digital objects are represented by a torrent file, this module implements functionality to transfer, modify and delete torrents. It is also responsible for content validation administrative operations. Semantic metadata associated with the MOSAICA digital objects when registering new content or associated with user queries when performing a search, is passed to this module by the Semantic layer in XML format. In order to easily obtain keyword/value pairs from this XML, each keyword is associated to a list of values and related contents. This module is also used to establish communication with the BNC in the Bit Torrent network. It enables to exchange torrents (torrent binary encapsulated in XML) and additional info between JXTA peers and the BitTorrent network.
- **Users and Groups**: this module provides functionality required to the manipulation of groups, allowing the definition and identification of users and digital objects that are associated to specific groups. It enables for example, the assignment of user access rights and permissions to specific groups or forum

messaging. It also implements the management of users and of their security credentials.

- **Utils:** this module encapsulates all utility classes that implement functions common to several modules. For instance, XML validating functions are frequently used by all modules and are thus implemented within the Utils module. It also provides services for propagating queries in the JXTA network. Depending on whether the MOSAICA peer has or does not have a local database, this module provides the functionality to respond to queries by initiating a local search, or to pass on the query to other MOSAICA and peers and wait for a response. Hashing functions are also implemented in this module.

3.1.3. Interface Layer

The interface layer provides the external actors with the means to interact with the p2p-cms, thus allowing them to use the value delivered by this MOSAICA subsystem. It also hides lower layers' implementation details from the MOSAICA application layer. External actors will interact with p2p-CMS system using the classes implemented at the interface layer, for normal user and administrative operations. Interface with external subsystems may be implemented as Web Services using SOAP to exchange messages, as agreed within WP5.

3.2. Component diagram

The diagram in figure 5 represents the dependencies between the different MOSAICA peer's framework components.

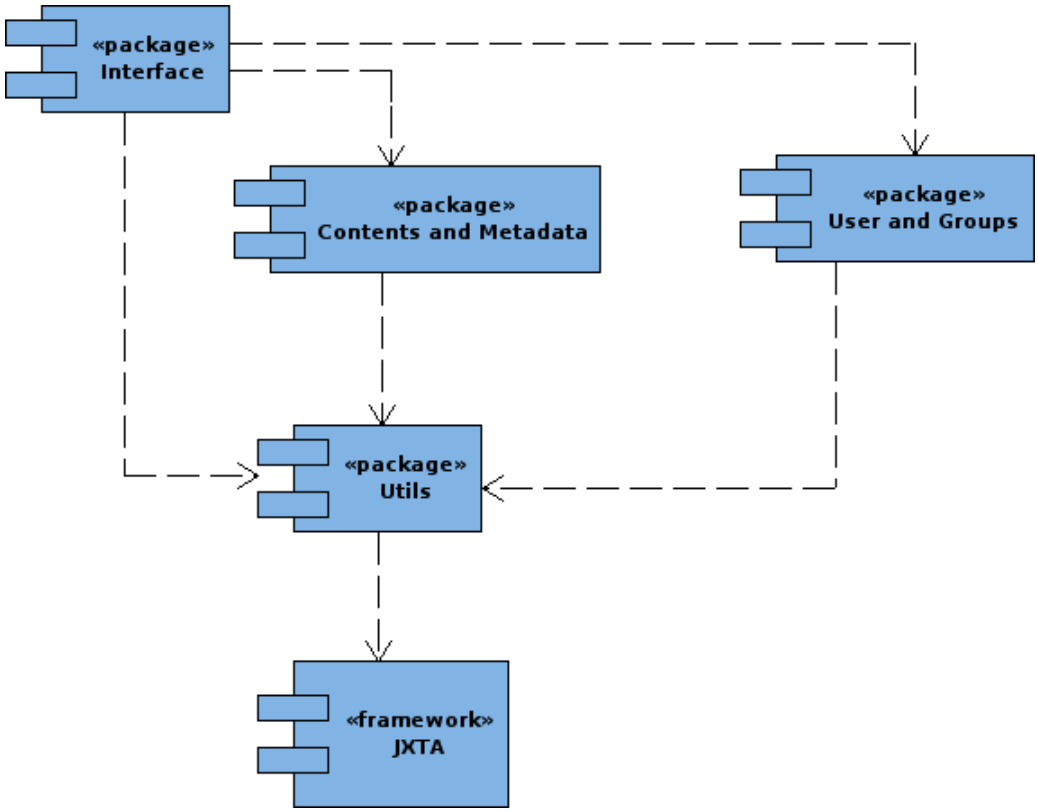

Figure 5 – p2p-cms framework component diagram

JXTA framework provides the p2p background technology that will be brought into MOSAICA. It provides specific p2p protocols that will be used to implement MOSAICA p2p networking functionalities. This will be properly documented in the final deliverable of WP3, with the description of the final implementation of the p2p-cms. The other components are modules that implement functionality specific to MOSAICA and that will therefore be developed in the course of the project.

3.3. Web Services interfaces

Each MOSAICA peer (i.e., a peer in the JXTA network) will provide two interfaces implemented as Web Services. One is to be used by normal users whilst the other one by administrators. As explained above in section 2.2, the value delivered by the p2p-cms is to be ultimately consumed by normal users and administrators. However, most of the interaction of these users with the p2p-cms will be done via the MOSAICA Semantic layer using the GUIs of the Semantic Browser, Semantic Annotator, Virtual Expedition Browser and GIS Browser. Nevertheless for the sake of simplicity, it was decided in this document to refer only to the actors "user" and "administrator". The tables 2 and 3 present the methods that the Web Services will provide.

Method	Description
login(LUser, password)	Login method. It returns a session hash if user credentials are correct.
registerContent(LSession, contentData,contentGroups)	Groups and content data XML are sent. ContentID is returned.
requestContent(LSession,contentID)	XML describing content (title, torrent binary, etc.) is returned
searchContent(LSession,query)	Query contents using a XML query. XML list of found contents, for this user groups, is returned
changePassword(LSession,oldpass,newpass)	Change user password
listGroups(LSession)	Returns XML list of user groups
getGroupMessages(LSession,groupID)	Returns XML list of group forum messages
sendMessageToGroup(LSession , groupID,message)	Sends a message to group

Table 2 - User Web Services methods

Method	Description
login(LUser,LPassword)	Login method. It returns a session hash if user credentials are correct.
registerContent(LSession, contentData,contentGroups)	Groups and content data XML are sent. ContentID is returned.
requestContent(LSession,contentID)	XML describing torrent (title, torrent binary, etc.) is returned
changePassword(LSession,oldpass,newpass)	Change administrator password
getGroupMessages(LSession,groupID)	Returns XML list of group forum

	messages
sendMessageToGroup(LSession,groupID,message)	Sends a message to group
addUser(LSession,user,password,type,name)	Add a new user to the system
addUserToGroup(LSession,user,grouID)	Add a user to a group
associateContentToMetadata(LSession,filter,contentID)	Associate a content to metadata key/values, which are described by a filter
createGroup(LSession,groupData)	Create a group, using XML description
listUsers(LSession,filter)	Filtered list of users. The filter and return values are specified in XML
listContents(LSession,filter)	Filtered list of contents. The filter and return values are specified in XML
listGroups(LSession,filter)	Filtered list of groups. The filter and return values are specified in XML
listMetadata(LSession,filter)	Filtered list of metadata. The filter and return values are specified in XML
removeUserFromGroup(LSession,user,groupID)	Remove a user from a group
removeUser(LSession,user)	Remove a user from the system
validateContent(LSession,contentID)	Validate a content
putMetadata(LSession,metadata)	Insert provided XML metadata key/values
removeContent(LSession,contentID)	Removes a content from the system
removeGroup(LSession,groupID)	Removes a group from the system
removeMessage(LSession,messageID)	Removes a forum message
removeMetadata(LSession,filter)	Removes filtered metadata
desassociateContentFromMetadata(LSession,keyword,value,contentID)	Disassociate a content from a key/value pair

Table 3 - Administrator Web Services methods

4. P2P-CMS DATA

There are four basic classes of data that will be processed and stored in MOSAICA p2p-cms:

- information relative to groups and users;
- semantic metadata associated with MOSAICA digital objects (media resources and ontologies);
- structural metadata associated with the MOSAICA digital objects;
- MOSAICA digital objects.

The first class of data, enables the creation and management of different groups within the MOSAICA user community that may share different interests, characteristics or rights (as for example, geographical location, age or cultural interests, guest versus administrator, etc.). It thus allows to selectively enable the access to the p2p-cms functionality, depending on the group the user requesting that functionality belongs to. In particular it allows the implementation of administrator management operations such as group and user account creation.

Semantic metadata describes the digital objects available within the MOSAICA system, providing key/value pairs and a corresponding association to the described digital object. It thus constitutes the fundamental class of information used during content search.

Structural metadata provides information about the torrent file associated to a specific digital object. It conveys information concerning the partitioning made to the digital object when submitted to the p2p network, providing to mechanisms to identify the location of the different pieces that constitute the complete archive. It also includes contextual information about the digital object, such as title and state. Note that digital objects are assigned to groups, which means that normal users can only access and consume digital objects from the groups where they belong to.

Digital objects are the actual MOSAICA content, media resources and ontologies, that are to be shared by all MOSAICA users. This data is stored in MOSAICA peers in a distributed fashion using the Bit Torrent protocol.

Metadata information, the first three classes of data here referred, can be replicated and distributed within the p2p-cms using different approaches. We have studied distributed databases and the DHT schema. The next sections provide a description of these two approaches.

In spite the fact that deliverable D5.1 refers the use of DHT in the MOSAICA p2p-cms, the study conducted in task 3.1 led to the conclusion that DHT would not be the most efficient solution to perform semantic search of content in p2p networks. DHT allows to search for specific content identified by a hash key and to eventually perform Boolean operations upon the results of searches that used different keys. However, it does not allow to perform semantic searches based on proximity. As such it was decided to adopt a different approach for the search of content, based on the use of distributed data bases. Nonetheless, DHT will be use for the distribution of the content in the p2p network. The p2p-cms has adopted a 2-layer approach, implementing on the top layer, among other, functionality to perform the distribution of metadata using distributed data bases and accordingly to perform metadata-assisted distributed search for content. The lower layer provides the means to partition, distribute and retrieve content according to the p2p paradigm using the Bit Torrent protocol.

4.1. Distributed databases

In a distributed database schema, all metadata concerning groups and users management as well as content indexing, is distributed across a network of peers. Part of this information may be replicated in some peers, whilst part may reside in one single location, to achieve high availability and load balancing. In order to ensure data integrity, it is necessary to implement a database synchronization scheme across the existing replicas. Ideally, the database replicas should reside in a specific set of JXTA peers that have better capacity in terms of computer power, memory, bandwidth and trustfulness.

The main issue around database synchronization concerns updating operations of existing stored information. If a user intends to modify some table, all other replicas must be conveniently updated. There are various network solutions that may fulfil this synchronisation objective:

- **Master-Slave Replication**: in a master-server replication schema, queries are forwarded to the master server. The master server asynchronously sends data changes to the slave server. Slony-I is an example of this type of replication.
- **Statement-Based Replication Middleware**: with this kind of replication, a program intercepts a SQL query and sends it to one or all servers. Read-write queries are sent to all servers and read queries can be forwarded to just one server. Pgpool-I and Sequoia are examples of this type of replication.
- **Synchronous Multi-Master Replication**: in this configuration, each server accepts write requests and original data is transmitted from the original server to the other servers before each transaction commits. Read requests can be sent to any server.
- **Asynchronous Multi-Master Replication**: using asynchronous multi-master replication, each server works independently and periodically communicates with other servers to identify conflicting transactions.
- **Multi-Server parallel Query Execution**: this solution allows multiple servers to work concurrently on a single query. This is accomplished by splitting the data among servers, each server executes its part of the query and returns the result to a central server where they are combined and returned to the user. Pgpool-II has this capability.

The *statement-based replication middleware* seams to be the best alternative to exploit in MOSAICA. It could be implemented by wrapping SQL statements in messages protocol.

The idea is not to implement a true database synchronization procedure, thus augmenting the system scalability and reducing network and system load, but to provide a mechanism through which synchronisation is gradually obtained. In our approach, a set of peers (that can grow in number) has a local database. Every record stored in those data bases has associated to it a time stamp, indicating its most recent creation or updating time. Each time a record in a local data base is modified, its time stamp is also updated. When the system receives a search request, the query is propagated across the MOSAICA peers and the results obtained by each peer that performs a search into its local data base are fed back up to the initiating peer. Conflicting results are resolved with the aid of the timestamp. At that point, peers with local databases that have received the results of the search may update their records with the most recent information.

Every peer that has a database engine may join this service although the access control information should be stored only by trusted peers.

Figure 6 presents the relational database model diagram designed for the p2p-cms.

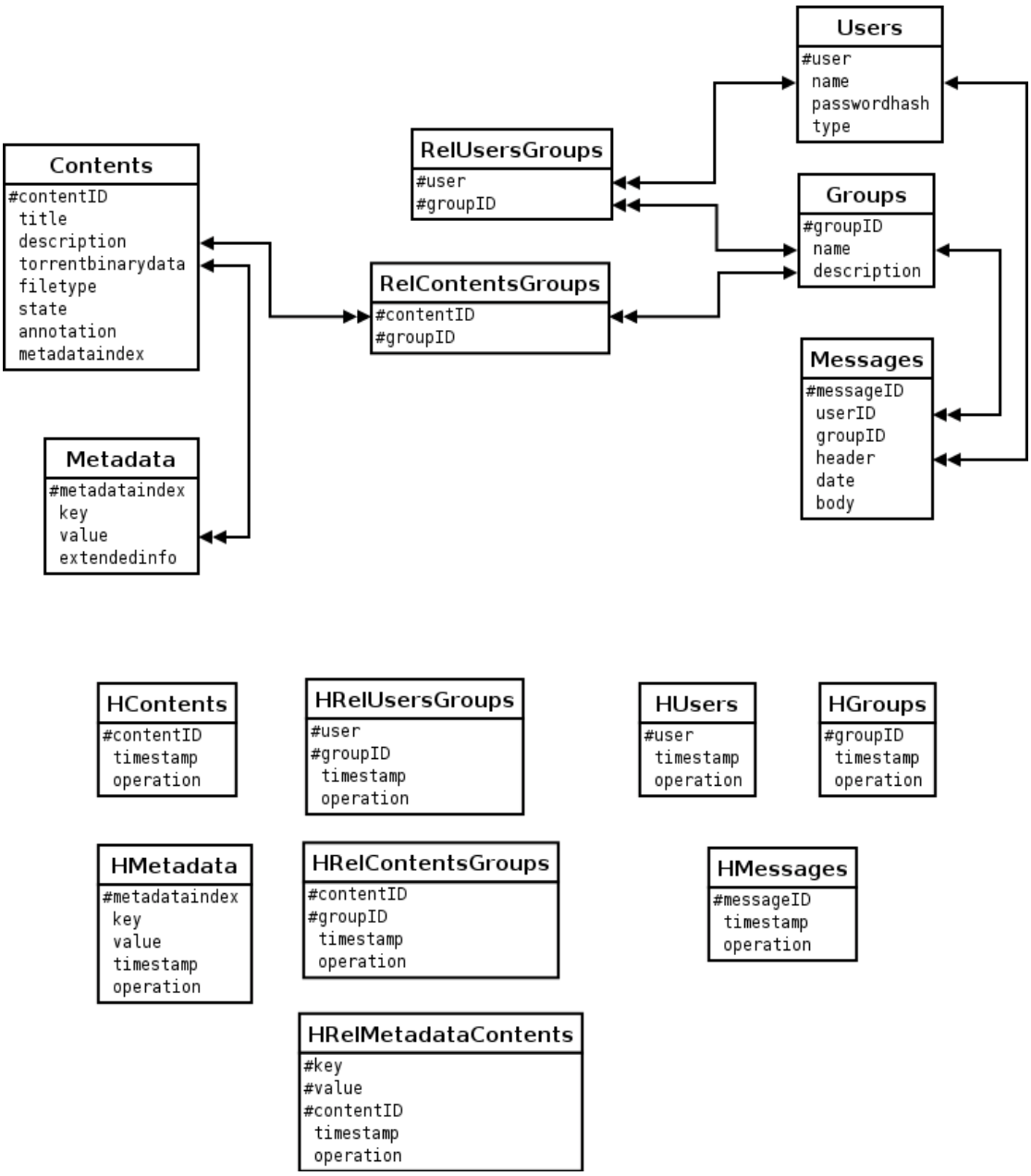

Figure 6: Database diagram

This database structure is oriented to answer queries related to users, groups and forums management. The design also enables requests for content associated to key/value pairs.

The complete model also includes relational tables that associate contents to users, groups and metadata key/value pairs as well as messages to groups (group forum).

For each table in the database schema there is one H-prefixed corresponding table that stores the table's write history. Those tables enable timestamp conflict resolution as explained in the exampled given below.

4.1.1. Database Timestamp Conflict Resolution

Consider, for simplicity, two peers (A and B) and a database table "Users". Suppose that the "Users" table in each peer has the values presented in Figure 7.

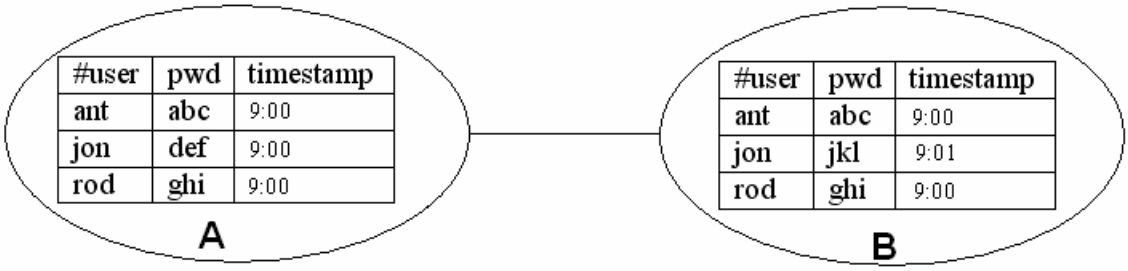

Figure 7: peer database table state

A user connects to peer A and the logon process queries the system with the SQL: "*select * from users where user='jon'*". The query is also sent to peer B and the user gets the following information:

#user	pwd	timestamp
jon	def	9:00
jon	jkl	9:01

Different values are returned for the same user. The conflict is resolved by using the time stamp information. The system selects the row with the most recent timestamp, which means that in this case the user gets the line:

jon	jkl	9:01

Now suppose that an administrator removes the user "jon", while server B is offline, in correspondence with timestamp 9:02. At that time, peers A and B would have the values:

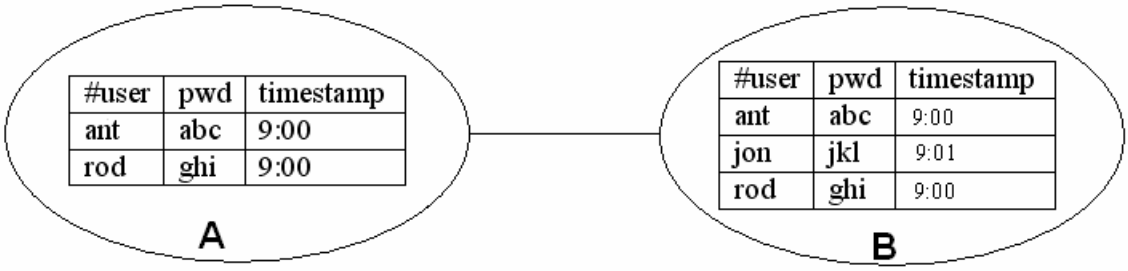

Figure 8: peer database table state

Meanwhile, peer B gets online and a new request in issued by a user with the SQL query "*select * from users where user='jon'*". Once the query is propagated to all peers, the requesting user will receive the incorrect result:

jon	jkl	9:01

This potential error is solved through the use of a history table that keeps track of operations performed on each row. The history table stores the last operation performed and the timestamp associated to it, for each key of the corresponding table. In the present example, the history table in peer A would have the row:

jon	9:02	deleted

The final result to the issued query would be achieved by crossing the results obtained by peer A using the history table with the results returned by peer B. As such, it would be an empty result.

It is useful to keep track of database write operations in order to schedule random database synchronization at regular intervals. A peer may schedule a task to select some random peers for querying the latest updates. That process could enable some database synchronization.

4.2. DHT

Distributed Hash Table (DHT) is a widely used algorithm in P2P computing. A DHT system assigns to each digital object a key generated by a hashing algorithm and maps the key to the node which also has an ID (e.g. hashed IP address). Normally this ID is the one closest to the key. In consequence, the storage and lookups of keys are distributed among multiple hosts. Chord, CAN and pSearch are representative DHT implementations.

Although the work developed within task 3.1 has included an analysis of current DHT implementations and features to assess the suitability of using DHT in MOSAICA, a parallel investigation was conducted on alternative solutions, notably based on distributed databases as presented in the previous sub-section. The outcomes of this combined study have indicated that it would be preferable to adopt in MOSAICA the alternative approach based on the use of distributed data bases. Nevertheless this deliverable provides here bellow an overview of DHT as a result of the study performed.

4.2.1. Basic concepts of DHT

The physical machines in the network are called **nodes** and the data items are called objects. **Namespace** refers to a set of names in which all names are unique. An **identifier** refers to the unique digital name of a node within a certain integer name space. In P2P system, it can be gotten from hashing the IP address of a node, such as identifier *PeerID* = *hash*(*IP*). A **key** is the unique identifier of a data item and can be gained from hashing the name of a data item *K* = *hash*(*V*).

Hash function can map keys to integers. **Hash table** is a dictionary in which keys are mapped to array positions by a hash function.

In DHT-based P2P systems, each node handles a portion of the hash space and is responsible for storing the objects related to some range of keys. For instance suppose that object O is associated with key K. Then, there exists a peer with identity id which is responsible for storing object O. After a lookup for the key K, the system will return the object O.

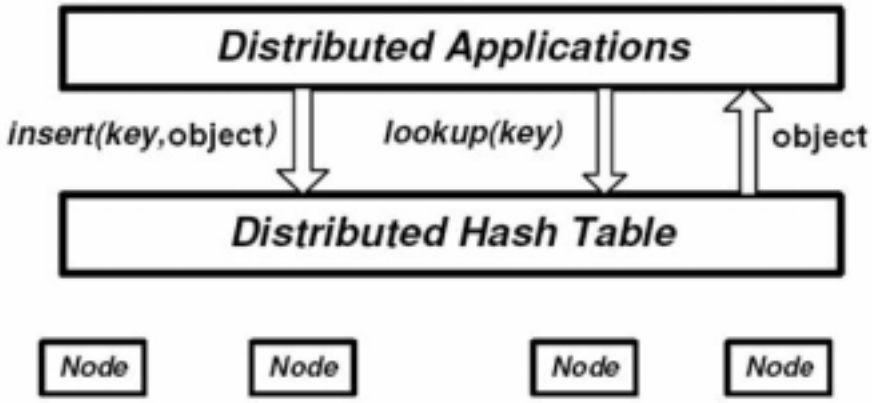

Figure 9 : Distributed Hash Table Functional Scheme

The core of the DHT systems is the routing algorithm. The DHT nodes form an overlay network which each node have several other nodes as neighbours. When a lookup(key) is issued, the lookup is routed through the overlay network to the node responsible for that key.

There are several systems that serve the DHT purposes, namely CHORD, PASTRY and TAPESTRY. Associated to JXTA which is the Mosaica's P2P core is the Meteor Service. Meteor implements two DHT algorithms: CHORD and CAN. Bellow it is provided a succinct description of the CHORD Meteor implementation, which is the one in a more advanced state of development.

4.2.2. CHORD Algorithm

The CHORD protocol supports only one operation: given a key, it maps that key into a node using a consistent hashing function. Depending on the application that uses CHORD, that node might be responsible for storing the value associated with that key.

The consistent hash functions assign each node and key an m-bit identifier using SHA-1 as a base hash function. A node's identifier is chosen by hashing the node's IP address concatenated with the time at the identifier creation while a key identifier is produced by hashing the key. We will use the term "key" to refer to both key and its image under the hash function, as the meaning will be clear from the context. Similarly, the term "node" will refer to both the node and its identifier under the hash function.

Consistent hashing assigns keys to nodes as follows. Identifiers are ordered on an *identifier circle* modulo 2^m. Key k is assigned to the first node whose identifier is equal to or follows (the identifier of) k in the identifier space. This node is called the *successor node* of key k, denoted by *successor*(k). If identifiers are represented as a circle of numbers from 0 to $2^m - 1$, then *successor*(k) is the first node clockwise from k. In the remainder of this paper, we will also refer to the identifier circle as the *Chord ring.*

If m=6 then we can have 64 identifiers. The following picture represents a Chord ring with m=6 and 10 nodes.

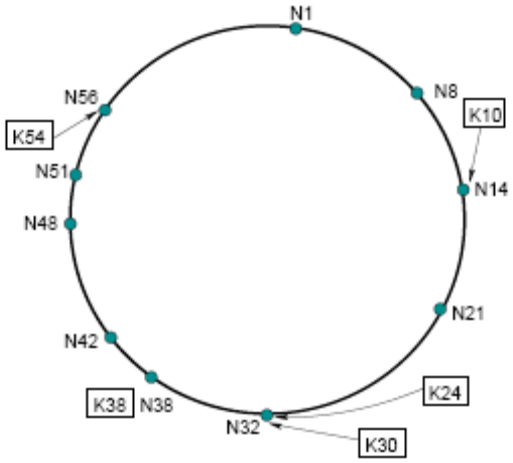

Figure 10: An identifier circle containing ten nodes and storing five keys

The successor of identifier 10 is node 14, so key 10 would be located at node 14. Similarly, keys 24 and 30 would be located at node 32, key 38 at node 38, and key 54 at node 56. When node n leaves the network, all of its assigned keys are reassigned to n's successor. No other changes in assignment of keys to nodes need occur. In the example above, if a node were to join with identifier 26, it would capture the key with identifier 24 from the node with identifier 32.

In order to accelerate a lookup for a key, each chord node maintains routing information besides the successor node organized in a table with up to m entries as described in Figure 11.

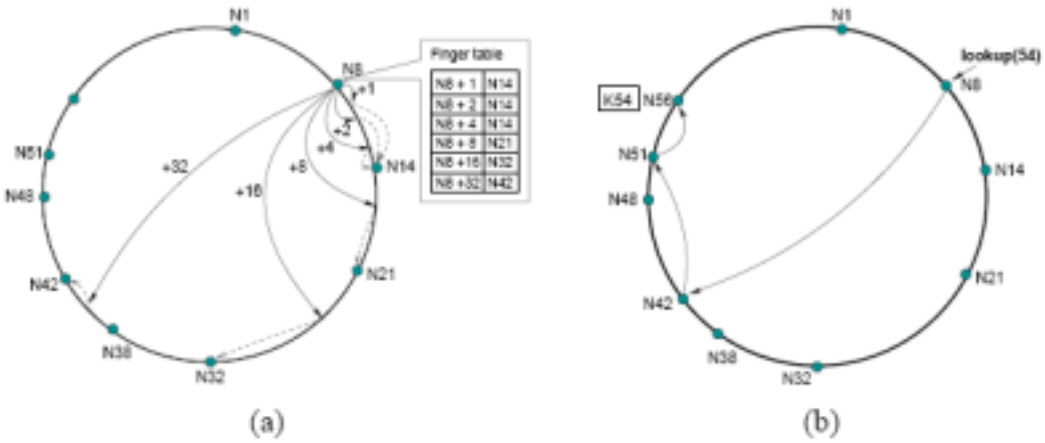

Figure 11: The finger table entries for node 8. (b) The path a query for key 54 starting at node 8 travels, using the Chord algorithm.

Every CHORD node sends the query to the node in its finger table which is nearest to the correspondingly key successor.

At bootstrap time, a joining node sends a successor request using another node in the network. When its successor is found, it is sent a join message that permits the new node enter the DHT network.

4.3. DHT versus Distributed Databases

All peers can participate in DHT service but there are some drawbacks:

- Queries in DHT are very simple (only keys of the hash table are queried). In MOSAICA context complex queries should be implemented, with performance and consistency constrains.
- Different groups of information are related in MOSAICA (e.g. users belong to groups). Consistency problems could arise when information is updated in DHTs. Relational databases solves this problem, using keys and referential integrity.
- Considering that DHT entries are XML documents and that there may be an enormous number of digital objects associated to a key/value pair, thus greatly increasing the size of the XML file, it may result in bad performance or even become impracticable to parse those files..

In this specification we address the database solution for groups and users management and indexing service although the main interfaces remain practically unchanged if we adopt the DHT schema for one or both services.

5. FRAMEWORK LOGICAL VIEW

This section further details, through the use of UML, the functional specification of the MOSAICA peer' software architecture. Class diagrams have been created for each package or functional sub-system that composes the MOSAICA peer as illustrated in Figure 4 and Figure 5 of section 3. These diagrams include the list of methods implemented by each sub-system thus providing in a formal and concise manner the indication of the functionality implemented therein. Additionally, sequence diagrams for the main use cases are also included. They provide an indication of the role played and value delivered by each sub-system towards the fulfillment of the specific use case. These formal specification tools provide a clear and direct guidance for the development of the corresponding software. As the implementation phase progresses and intermediate tests are conducted, it may be possible that these functional specifications may need to be revisited and modified according to the indications obtained. This is in fact part of the iterative development process intended to be adopted within MOSAICA and accordingly described in deliverable D5.1.

It should be noted that UML class diagrams are not provided for the JXTA sub-system, as this is considered as background knowledge or third-party developments brought into the project. JXTA is an open source project. Some of the existing JXTA classes will be directly used in MOSAICA whereas others will be modified or even completely re-written, even if the original functionality and concepts may be maintained. The final deliverable of WP3 .1, D3.2 "MOSAICA Distributed Content Management System – Services and Protocols" will provide a clear indication of all the developments made.

5.1. Class Diagrams

5.1.1. Interface package

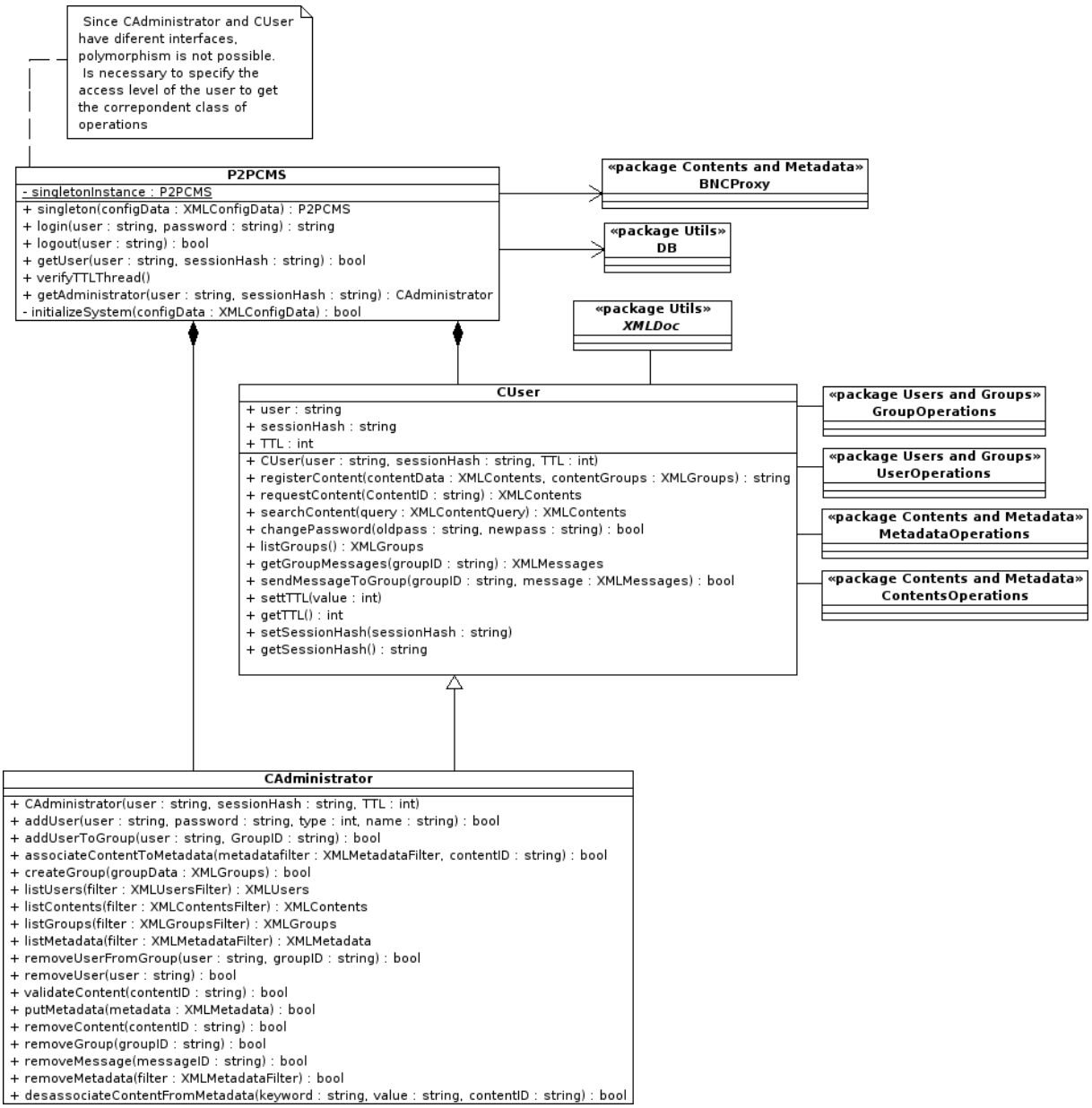

Figure 12: Interface package class diagram

The *P2PCMS* is the main class which represents the entire system. This class is implemented with singleton pattern., where can be only one instance per peer, which is used to initialize all services and user sessions creation and management. Each peer stores the state of every logged user. A thread is started to monitor all sessions in the session pool in order to remove the expired ones and to decrease every TTL at regular periods of time. Every time a session expires, the user must login again in the system.

The class *CUser* implements normal user operations and the class *CAdministrator* inherits them and provide additional administrator functionalities. Correspondent user objects of these classes are obtained by p2pCMS::getUser() and p2pCMS::getAdministrator() methods respectively.

5.1.2. Content and Metadata package

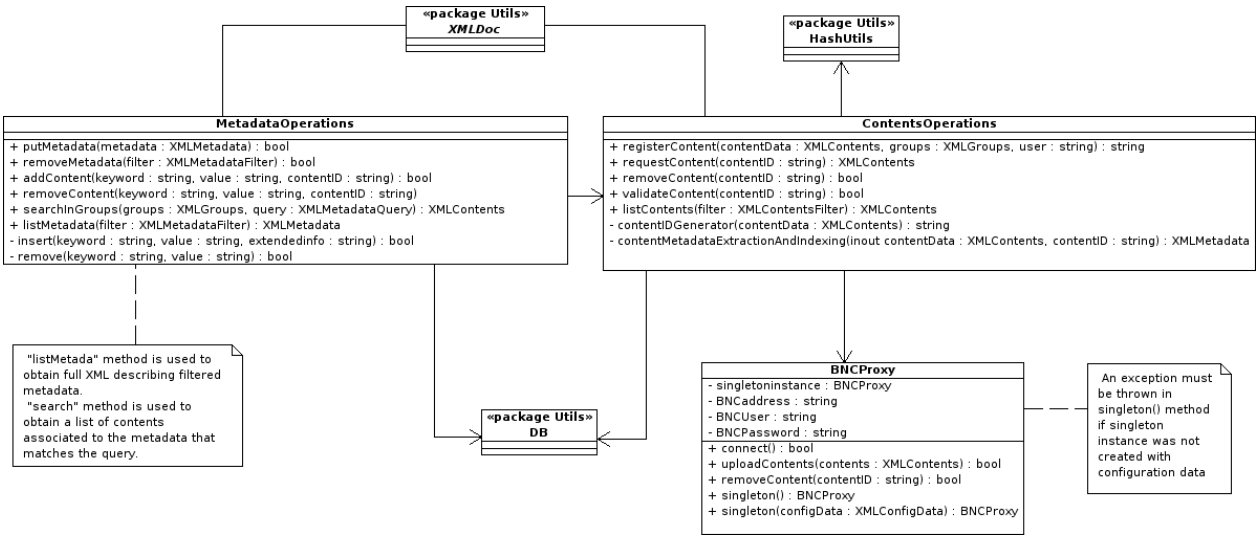

Figure 13: Content and Metadata package class diagram

The *MetadataOperations* class permits to manage metadata indices stored in the database and to implement metadata based search queries. The *ContentsOperations* class is used to implement all content management functionalities overseen by the use cases.

BNCProxy is a singleton that should be instantiated by *P2PCMS* to configure proxy connection parameters

5.1.3. Users and Groups package

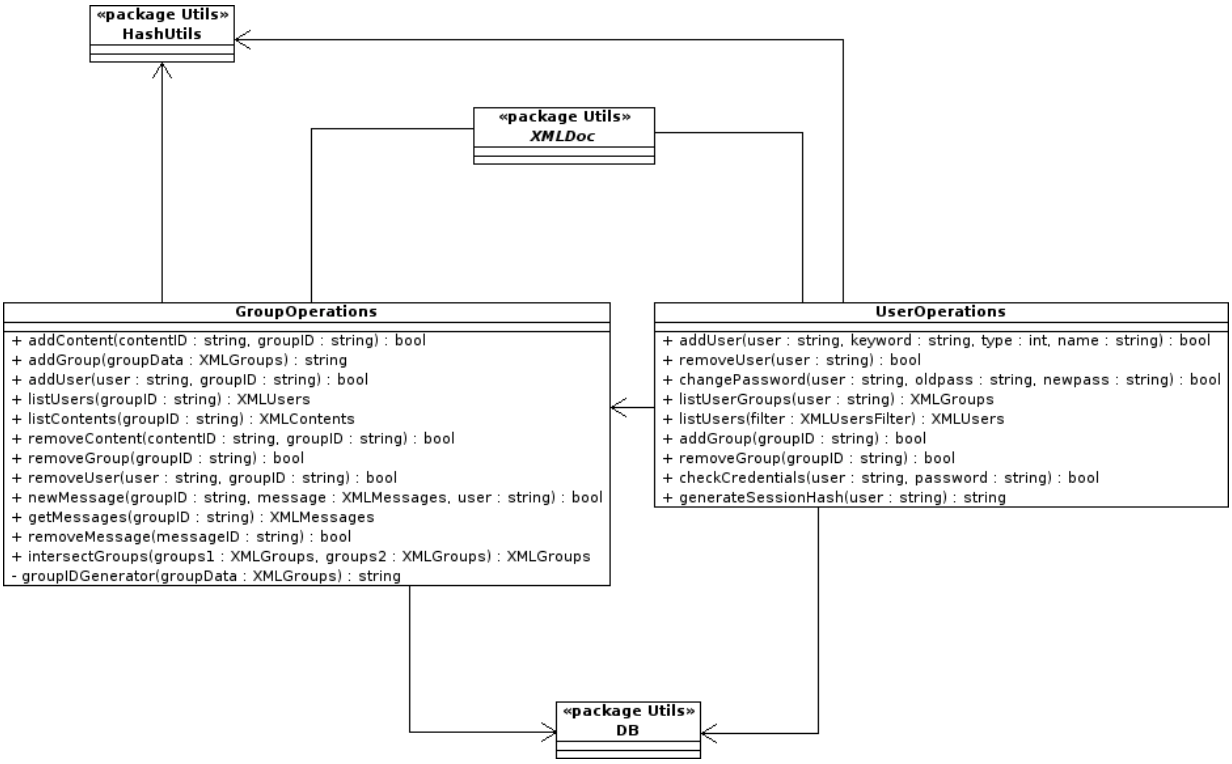

Figure 14: Users and Groups package class diagram

This package classes enable group management and concede group access to specific user. Methods to list users associated to a group and list groups that a user belongs to are provided too.

5.1.4. Utils package

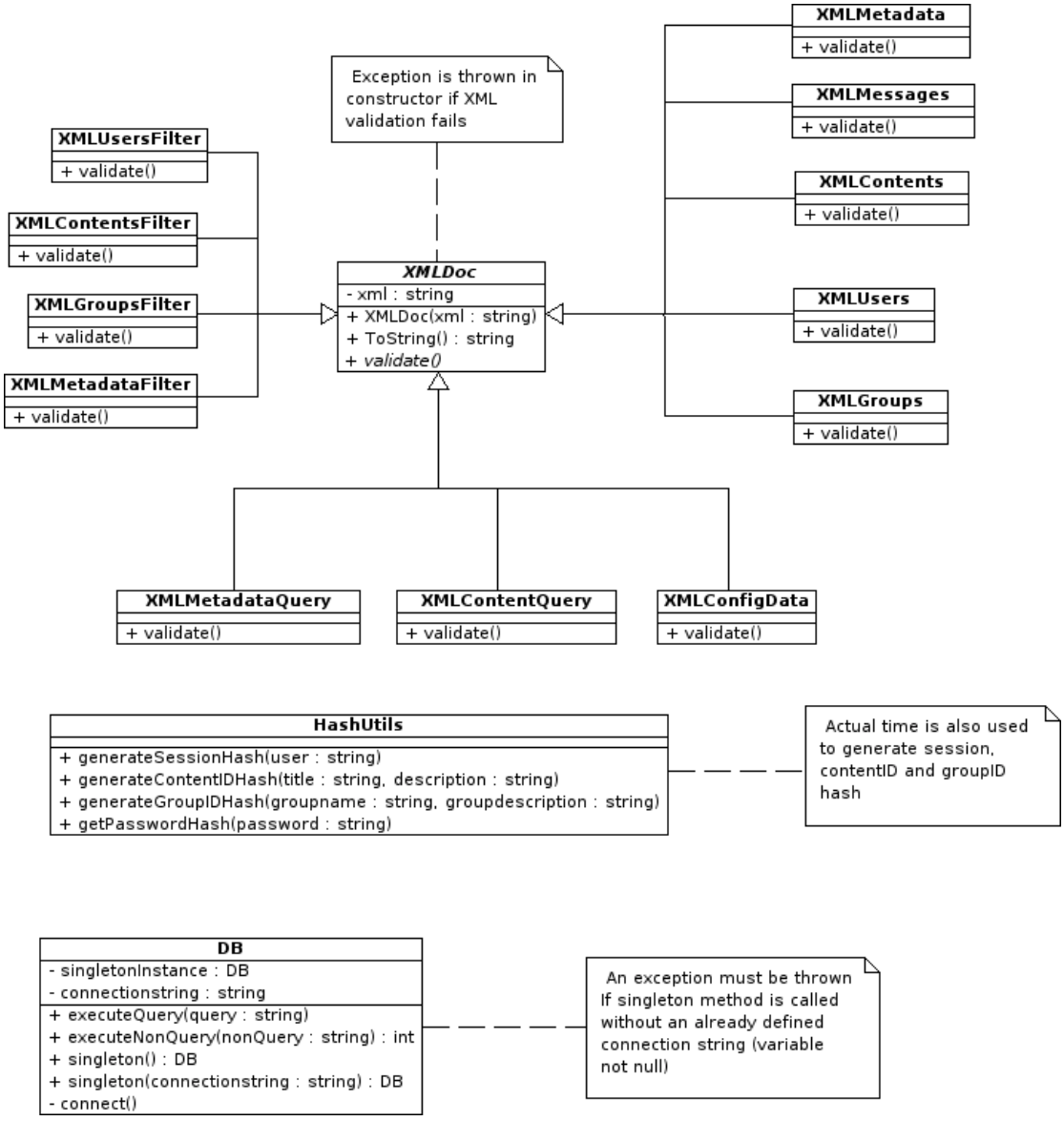

Figure 15: Utils package class diagram

This package provides container classes of each type of XML document that is passed as argument in other packages class functions. All XML validation classes have a common abstract parent class and must implement virtual method *validate()* to test correspondent XML schema.

DB class is a singleton, instantiated by *P2PCMS* with connection parameters, that provides methods to execute query's and other SQL operations

There is also the *HashUtils* class that implements some useful functions for generating unique IDs and session strings.

5.2. Sequence Diagrams

This section provides sequence diagrams representing the logic involved between classes and objects towards the implementation of the most relevant use cases.

5.2.1. Login

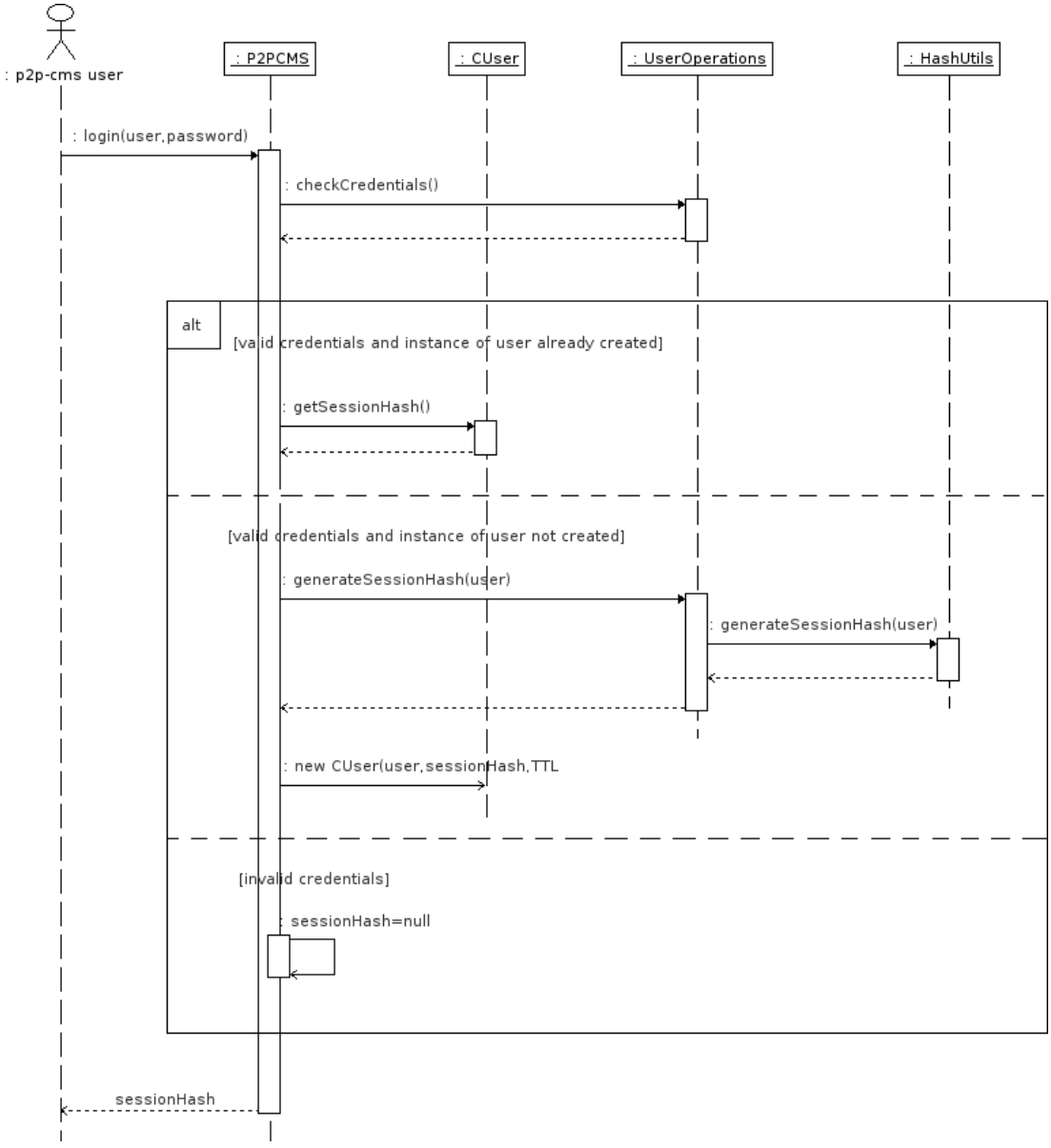

Figure 16 - Login sequence diagram

5.2.2. Register Content

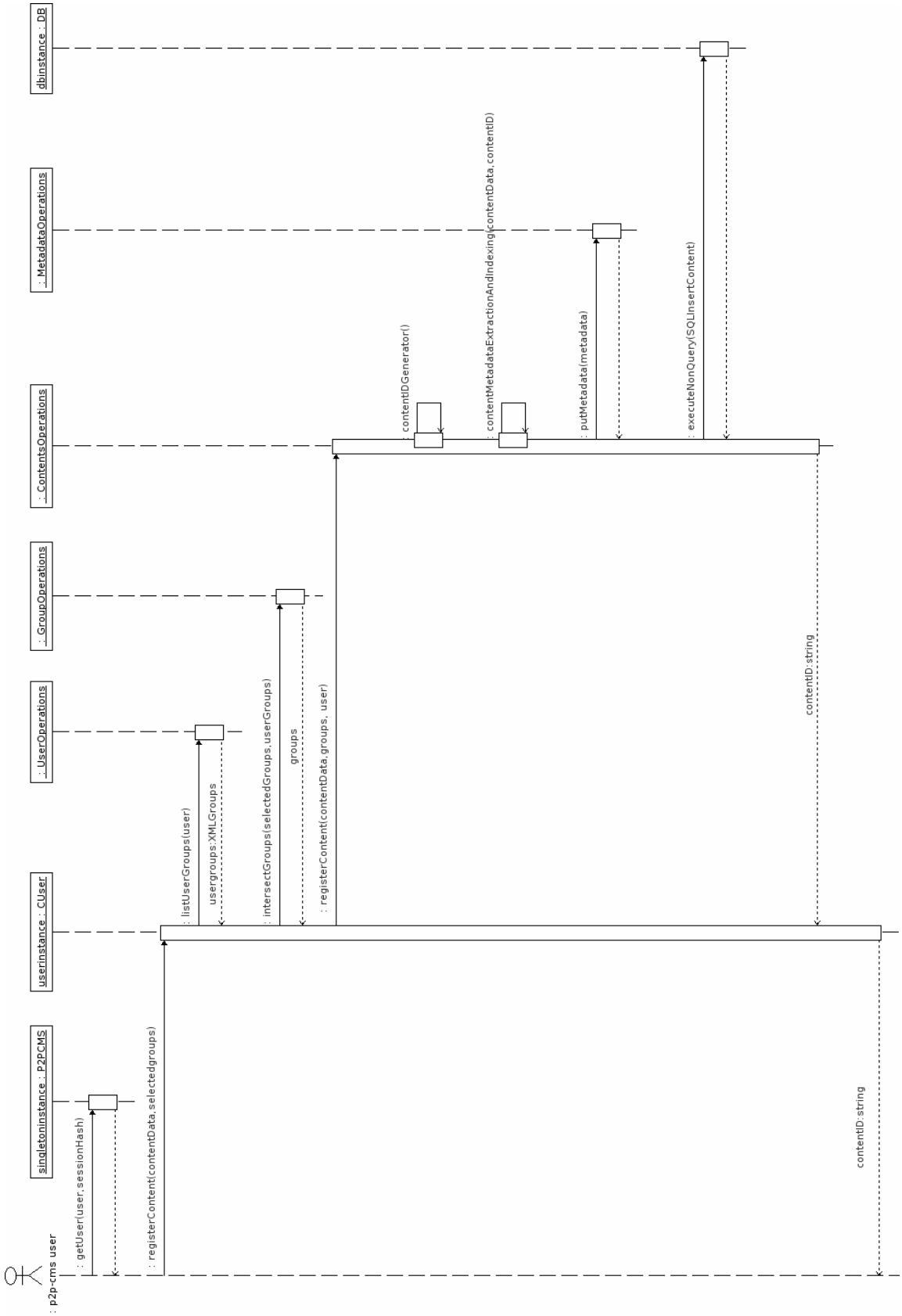

Figure 17 - Register content sequence diagram

5.2.3. Search content

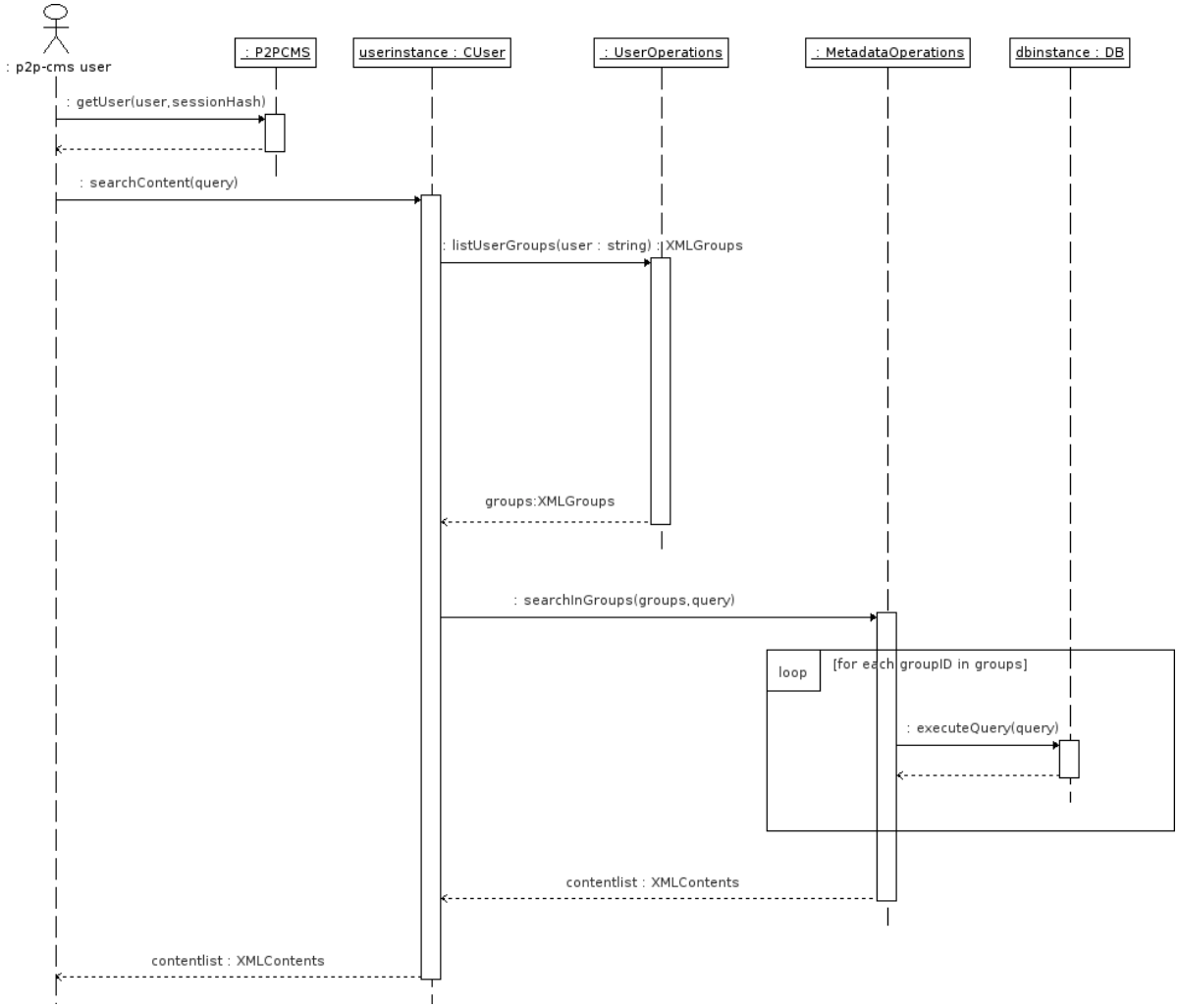

Figure 18 - Search content sequence diagram

6. CONCLUSIONS AND FUTURE WORK

This deliverable has provided a thorough description of the work conducted during the first 4 months of activity of task 3.1.The outcome of that work is the formal specification of the subsystem *p2p-cms*, the distributed content management system of MOSAICA based on the use of p2p technologies

The analysis performed on the current state of the art in p2p technologies and solutions and the investigation conducted on emergent research work on semantic-based search in p2p systems, have guided the approach taken and the concepts adopted for the specification of the MOSAICA p2p-cms. Accordingly it was decided to support the semantic-based search using distributed data bases. Moreover, it was decided that the system should be implemented as a 2-layer architecture where the high-level functionality would be based on the open-source JXTA platform and the lower layer on the Bit Torrent network. In this way the operations related with the partitioning and distribution of the MOSAICA resources or digital objects across the network and conversely the retrieval of the different pieces and reconstruction of the digital objects would be de-coupled from the application-level MOSAICA-specific operations. The former include generic operations performed in any p2p network and would then be implemented by the most efficient p2p technology currently available. The later includes the implementation of specific functionality to support the MOSAICA requirements and would be based on one of the most versatile open source platforms for p2p networks.

BIBLIOGRAPHY

JXTA v2.0 Protocols Specification.
http://spec.jxta.org/nonav/v1.0/docbook/JXTAProtocols.pdf; accessed in February 21, 2007

Stoica, I., Morris, R., Karger, D., Kaashoek M. F. and Balakrishnan, H. *Chord: A scalable peer-to-peer lookup service for internet applications.* MIT Laboratory for Computer Science. http://pdos.csail.mit.edu/papers/chord:sigcomm01/chord_sigcomm.pdf; accessed in February 21, 2007

Xie, M. *P2P Systems based on distributed hash table.* Computer Science, University of Ottawa. http://www.site.uottawa.ca/~mxie/academic/bak/DHT.pdf; accessed in February 21, 2007