

INFORMATION SOCIETY TECHNOLOGIES (IST) PROGRAMME



VICTORY

IST-6-044985-STREP

P2P-based Networking Stack

Deliverable No.	D3.1		
Workpackage No.	WP3	Workpackage Title	P2P network
Task No.	T3.1	Task Title	P2P-based Networking Stack
Authors	J. Trnkoczy, S. Dobravec, J. F. Tasič (UoL)		
Status (F: final; D: draft; RD: revised draft):	F		
File Name:	VICTORY-UoL-RD-WP3-V05-D3.1-P2P-based Networking Stack.doc		
Project start date and duration	01 January 2007, 30 Months		

List of abbreviations

2D	Two dimensional
3D	Three dimensional
API	Application Programming Interface
CPU	Central Processing Unit
DHT	Distributed Hash Table
DRM	Digital Rights Management
GRM	Graphics Resource Management
GUI	Graphical User Interface
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
J2ME	Java 2 Platform, Micro Edition
JXME	JXTA Java Micro Edition
JXTA	Juxtapose
MD5	Message Digest Algorithm 5
NAT	Network Address Translator
P2P	Peer-to-Peer
PC	Personal Computer
PDA	Personal Digital Assistant
QoE	Quality of Experience
QoS	Quality of Services
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VICTORY	Audio-Visual ConTent search and retrieval in a distributed P2P repositORY
W3C	World Wide Web Consortium
XML	eXtensible Markup Language

Table of contents

List of abbreviations.....	2
Table of contents.....	3
List of figures.....	5
Executive Summary.....	6
1 Introduction.....	7
1.1 The role of the networking stack in the VICTORY Project.....	7
1.2 Deliverable Objective.....	7
1.3 Deliverable Structure.....	7
2 Architectural and technological decisions.....	9
2.1 Requirements that influenced the design of the networking stack.....	9
2.2 Architectural decisions.....	10
2.3 Middleware technology selection.....	13
3 Networking stack architecture.....	15
3.1 VICTORY architecture.....	15
3.1.1 Desktop edge peer architecture.....	18
3.1.2 Mobile edge peer architecture.....	19
3.1.3 Super peer architecture.....	21
4 Implemented networking stack specification.....	24
4.1 Mechanisms for content advertisement, discovery and delivery.....	24
4.1.1 Description of components.....	25
4.1.2 Content advertisement, discovery and delivery sequence diagrams.....	29
4.2 Mechanisms for the effective interworking between the mobile and Internet-based segments of the VICTORY system.....	36

4.2.1	Problems and limitations of the VICTORY system mobile segment	36
4.2.2	Gateway functionality	37
4.2.3	Implementation technology.....	40
5	Conclusions.....	41
Appendix 1: Edge peer content advertisement, discovery and delivery APIs		42
Appendix 2: Super peer APIs.....		46
Appendix 3: Mobile gateway APIs.....		49
Appendix 4: Central database APIs		52
References.....		55

List of figures

Figure 1: The overall VICTORY architecture.	16
Figure 2: The desktop edge peer networking stack.....	18
Figure 3: The mobile edge peer networking stack.....	20
Figure 4: The super peer networking architecture.	21
Figure 5: VICTORY components allowing for efficient content advertisement, discovery and delivery.....	26
Figure 6: The sequence diagram for MultiPedia content discovery and delivery.	30
Figure 7: The sequence diagram for MultiPedia content advertisement.	32
Figure 8: Sequence diagram for P2P platform start up process.....	34
Figure 9: Sequence diagram for P2P platform shut down process.	36
Figure 10: The sequence diagram representing the interaction between mobile client, gateway and other system components.	39

Executive Summary

This document presents the networking stack that is being developed for the VICTORY project. It represents a report on the work accomplished in the scope of task T3.1. The main goal of this task is to design and develop a flat, scalable, efficient and robust networking stack, which will allow high degrees of (mobile) peer social autonomy serving the purposes of three dimensional (3D) content search, retrieval and manipulation.

The developed networking stack supports the functionalities that were identified in the VICTORY system requirements [1] and in the VICTORY architecture [2] documents. The specification and prototype implementation of the protocols and semantics for 3D content advertisement, discovery, and delivery, as well as for routing and efficient aggregation of content and context based queries were produced. The developed networking stack can be considered as a baseline communication mechanism upon which the application level services rely. On the other hand the content sharing and retrieval operations (which could be considered as application level services, too) represent a fundamental part of every Peer-to-Peer (P2P) system, and were therefore included in the set of core middleware functionalities. The resulting networking stack allows registered peers (both mobile and desktop edge peers) to have access to the distributed MultiPedia content repository and allows them to be able to contribute to the repository.

In the following chapters the networking architecture and the relationships between individual system components will be described. This architecture is further refined with regard to various network entities that represent a part of the networking stack. The technology used for the realization of the networking stack will also be described, and the functionalities provided by the stack will be elaborated in details. The provided functionalities are exposed through the application programming interfaces (APIs) that are used by the application level modules of the VICTORY system and are specified in the appendices of this document.

It should be noted that although this document is delivered according to the VICTORY plan in month 18, some of the functionalities of the networking stack will remain open issues until all the application level components reach their final version and all VICTORY modules are integrated. Further revisions of the system architecture and its modules are possible and they may influence the final version of the VICTORY networking stack. This document can therefore be considered as a specification of the core networking stack functionalities, but some modifications are likely to be expected.

1 Introduction

1.1 *The role of the networking stack in the VICTORY Project*

The project VICTORY aims at the creation of a distributed digital repository for 3D and associated multimedia content. The main goal of the project is to create the first distributed MultiPedia object repository (a repository of three dimensional objects along with accompanying two dimensional, textual and other forms of multimedia descriptions) to which any authorized user can contribute. The repository should allow for the utilization of the distributed data storage, distributed network bandwidth and also distributed computational power of peers in the network. Through the framework, users should be able to search, share, retrieve and visualize 3D and audio-visual data among peers around the world. The system should support desktop as well as mobile client devices.

The developed middleware described in this document therefore serves three main purposes. First, it can be considered a baseline communication mechanism, upon which the application level services are built. Second, it provides the content sharing and retrieval framework, which is considered as a core P2P middleware stack functionality. Third, it provides the functionalities of the VICTORY system to the mobile users. For the latter, the mechanisms allowing the connection between the mobile and Internet-based segments of the system were implemented.

The appearance of Peer-to-Peer (P2P) applications in recent years has demonstrated the significance of distributed information and workload sharing systems by offering the advantages of increased data availability and scalable storage space, network bandwidth and computational power sharing. These capabilities make P2P technologies a natural choice for the VICTORY networking stack. On the other hand some of the VICTORY requirements cannot be met by a pure P2P system. Therefore it was decided that a hybrid approach that combines an ordinary client-server solution and a P2P networking stack will be used in the project.

1.2 *Deliverable Objective*

The main goal of this deliverable is to report the work accomplished in task T3.1 of the work package no. 3 (WP3). The deliverable objective is therefore to provide the specification and implementation of the middleware that supports the MultiPedia content sharing, discovery and delivery. Furthermore the mechanisms allowing for the effective interworking between the mobile and Internet-based segments of the VICTORY network were developed and are described in detail in this document.

1.3 *Deliverable Structure*

The deliverable is organized as follows. **Section 2** identifies the requirements of the VICTORY system and provides various technological decisions that influence the design of the networking stack. The selected technology, upon which the networking stack is built, is briefly described. The system architecture has a fundamental influence on the design of the networking stack; therefore **Section 3** introduces the overall architecture of the system and describes the networking architecture of each individual component of the system. **Section 4** provides a detailed description

of the implemented prototype. The components that allow for content advertisement, discovery and delivery are presented and the mechanisms and modules for efficient interworking between mobile and personal computer (PC) based segments of the VICTORY system are described. Finally, in **Section 5**, the conclusions are summarized. The application programming interfaces (API) of the developed prototype are provided in several appendices.

2 Architectural and technological decisions

In order to select the appropriate architecture and technology for the implementation of the VICTORY networking stack, a detailed requirements analysis was performed. In the following chapters the requirements and constraints that influenced the design of the networking stack will first be outlined and explained. The decisions upon several important questions regarding the system architecture will follow. Finally the selected technologies used for the implementation of the networking stack will be described.

2.1 *Requirements that influenced the design of the networking stack*

The following main requirements that influenced the design of the communication mechanisms were identified:

Requirement 1: A novel content and context based MultiPedia object search and retrieval

There are two main differences between VICTORY and existing P2P search and retrieval systems. First, in VICTORY complex MultiPedia objects are being shared instead of plain files, and second, advanced content and context based multimedia search mechanisms are being used to search for these objects. The sharing of MultiPedia objects represents the sharing of virtual datasets each represented by a set of physical files (file containing 3D object together with accompanying files containing two dimensional (2D) images , textual files, video files etc.). The users can search for shared MultiPedia objects and retrieve any/all of the files in a dataset representing a particular object. The content based search uses low-level geometrical representation of an object to retrieve similar objects. The context based search is based on the ontological domain model. The complex search mechanism offered by VICTORY had a major influence on the architectural and technological decisions.

Requirement 2: Users able to contribute to the repository

The system should allow the users to add their own MultiPedia content to the repository (i.e. share their content with other users). Other users can then search and retrieve the newly added content. Furthermore, users may modify existing MultiPedia content or use that content to create new MultiPedia items and add them to the repository.

Requirement 3: Strict security and organization management procedures

The VICTORY application should cover both open communities as well as professional industrial applications. Professional applications often require a tight controlled architecture where users and peers have to be identified and authorised to get access to data. In open communities, on the other hand, users tend to exchange data in an uncontrolled way. Furthermore, the dynamic nature of open communities makes the task of controlling the access more difficult. A flexible identity management and security framework needs to be developed in order to be generic in the sense of the type of supporting application.

Requirement 4: Support for mobile peers

The developed system should support access from mobile devices. VICTORY will allow mobile users to search, retrieve, and finally use (display) extremely complex MultiPedia items as if they were using fully featured PC clients. Therefore the gap between fully featured, high performance PC clients and low capabilities mobile devices needs to be bridged. The infrastructure should be “flexible” enough in order to support the widest spectrum of devices independently of their hardware and operating system.

Requirement 5: Utilization of distributed storage space and bandwidth

While it is easier to develop a digital repository as a centralized solution, a distributed P2P based option solves the storage space and network bandwidth related problems of the centralized approach. This is especially true for a system that supports a large community of users that could build upon an open distributed repository of complex heterogeneous visual objects, accompanied with related textual documents, videos and images.

Requirement 6: Utilization of distributed processing power

The system should develop a scheme so as to utilize not only the distributed data storage, but also share the central processing unit (CPU) power of peers for the pre-processing, indexing, and representing of MultiPedia data.

Requirement 7: Copyright protection of content

The system should provide content copyright protection. Digital Rights Management (DRM) technologies allow intellectual property owners to express policies for content usage with confidence that these policies will be respected once the content is distributed in the network.

Requirement 8: Ensured quality of experience

The system should provide a satisfactory Quality of Experience (QoE) level to the end users. QoE will be realized as a combination of a multitude of Quality of Service (QoS) mechanisms. Communications quality, processing speed, 3D content rendering quality etc. will be monitored and provided to the users in order to impact the user experience. The system should therefore, amongst other QoE related mechanisms, provide means to connect the service consumers to the appropriate service providers dynamically, based on the current state of the system.

2.2 Architectural decisions

It is clear from the above mentioned requirements that in the VICTORY system there is a significant number of services needed. Well defined and efficient communication mechanisms between service providers and service consumers are therefore necessary.

Entities that form the VICTORY system (i.e. peers) may have different roles in different service communication scenarios. In some cases they might be providers of the services, in others they may

use services as clients. The provision of the services can either follow the client-server or the P2P paradigm. The P2P paradigm differs from the traditional client-server model in which communication between entities is relayed through the server. A pure P2P network does not have the notion of clients or servers, but only equal peer nodes that simultaneously function as both service consumers (i.e. “clients”) and service providers (i.e. “servers”) to other nodes on the network. In a pure P2P network, any node is able to initiate and complete the supported transactions with any other node. However, there exist some services whose provision using the P2P approach is not very suitable. In fact, pure P2P applications and networks are very rare in reality. Most existing P2P networks and applications actually contain or rely on some client-server elements. In order to optimally design the communication architecture and reach the decision on which services should follow the P2P approach and which should be client-server based, the following issues were taken into consideration:

- 1) **Technological capabilities of peers** need to be carefully studied, since they influence the choice of implementation technology. The devices that are the most problematic ones with respect to technological capabilities are, without doubt, the mobile devices. These devices have very limited resources (processing power, storage space etc.) compared to a desktop personal computer (PC). Due to limited resources some of the functionalities of mobile devices will have to be provided by other entities in the network on behalf of mobile devices.
- 2) **Peer visibility** is another important factor influencing the communication stack. Some peers may have public and static Internet Protocol (IP) addresses, while the majority of peers are usually hidden behind firewall and Network Address Translator (NAT) devices. Those peers cannot be contacted directly, so mechanisms supporting the availability of their services to other peers need to be developed. Another issue to consider is that peers can change their IP addresses when the users move from one location to another.
- 3) **Peer availability** is very important since it influences the services availability. End users applications may be often offline, meaning that their services are not available all the time. Another issue is that the users application might not have a chance to close properly (e.g. the end user turns off the computer before closing the application, network problems occur, etc.) and perform all the operations that are necessary to un-register from the system. Therefore presence control mechanisms need to be developed and integrated in the system.
- 4) **Communication efficiency** is another important issue. The efficiency of the communications protocols is important for the service provision. For example, high throughput streaming of rendered objects through P2P protocols proved to be difficult. P2P mechanisms introduce additional communication overhead, which demonstrates itself in a somewhat slower establishment of communication pipes (especially over firewalls and NATs) and in bandwidth utilization inefficiency. These protocols should therefore be used only for the tasks where they perform best.
- 5) **Service availability** is yet another issue to take into consideration when planning network architecture. Some services (e.g. file share and file download) can be seen as ‘core’ services and should be running on every peer in the system to enhance their availability. For the rest of the services, other parameters (security for example) could be the most influencing and thus requiring a different architectural model.

Taking into account the requirements, the technical characteristics of the VICTORY services, and the above mentioned issues, the following architectural decisions were taken:

- 1) A hybrid architecture between client-server and pure P2P system will be used in the VICTORY project. The networking stack will not follow the pure P2P approach. This decision is based on the following reasons:
 - a. The advanced content and context-based search is a lot more complex than the search mechanisms in traditional P2P systems. The distributed hash table (DHT) based solution is not feasible in this case. The unstructured overlays option, where every peer in the system provides its own search service was considered, but was omitted because of the complexity of the search service module. The installation of a complex and resource consuming search engine on every peer in the system is not feasible because of the resource constraints of the peer devices.
 - b. The need for strict management over performance and reliability makes the client-server architecture favourable over the pure P2P architecture, in which these requirements are difficult to ensure.
 - c. Security is particularly critical in the VICTORY system, since it has to support professional business applications. A centralized organization for management can tackle the authentication and authorization problems more efficiently, as the most accepted solutions rely on the existence of public key infrastructure and the privilege management infrastructure, which are fit to centralized systems and not to extremely decentralized P2P networks.
 - d. It is better to use the P2P networking only in situations where it can outperform client-server based systems. It has been already proven by other systems that the file sharing and the file downloading services benefit the most from the greater scalability and single-point of failure resilience brought by P2P systems. In VICTORY the P2P networking will be used mainly to support these services.
- 2) There will be basically three kinds of entities in the network: super peers, desktop edge peers and mobile edge peers. While desktop and mobile edge peers are considered as occasionally connected and unreliable peers (that may experience network failures and other problems that cause inappropriate disconnection from the system) the super peers are considered to be reliable and available all the time. In case one of the super peers fails the clients that are connected to this super peer will not have access to the system. They can however use another super peer, but the redirection will not be provided automatically, instead users will have to reconfigure the application manually. The reliability of super peers is therefore of critical importance.
- 3) The search engine in the VICTORY project will follow the Grid-like [14] approach rather than the fully distributed P2P search mechanisms. Only one indexing service per community of users will exist. This service will index all of the MultiPedia objects that are added to the repository by users belonging to the corresponding community. However the search service, which would present a significant performance bottleneck, if it was totally centralized, will be running on many peers in the network. Every search service will use a request broker that forwards the requests to the service that currently serves the smallest number of users. All of the search services will use the same index produced by the indexing service. The

mechanisms for index updating will therefore have to be developed. It is obvious that the reliability of the indexing service is of critical importance since in case of its failure the users, while still being able to discover and retrieve the content, will not be able to add new content to the repository.

- 4) Mobile devices will access the Internet-based segments of the VICTORY network through a dedicated gateway. The gateway performs various tasks on behalf of the mobile clients, whose processing power, bandwidth, and other capabilities are usually limiting the set of functionalities they can perform themselves. Furthermore the mobile gateway acts as a bridge between mobile clients, which are not capable of running a fully featured P2P communication platform, and the rest of the system. The gateway will be based on the World Wide Web Consortium (W3C) Web Services [3] standards technology.
- 5) All other services, except for the share/download services that are needed for distributed content storage and retrieval, will reside on centralized nodes. They will follow the client-server approach; therefore the centralized nodes that host these services need to have public and static IP addresses.
- 6) Edge peers that are hidden behind firewall and NAT devices will be accessed from other peers through "relay peers". Relay peers forward messages on behalf of peers that are not directly addressable from the perspective of another peer (due firewall or NAT environments), bridging the different physical and/or logical administration domains.

The development of the VICTORY networking architecture was influenced by the above mentioned decisions. A state of the art VICTORY networking architecture will be presented in Section 3.

2.3 Middleware technology selection

P2P has become an increasingly popular technology over the last decade. Along with the development of well known P2P applications (e.g. Napster, Gnutella, ICQ, SETI@home, Skype, just to name few of them), there have been considerable research efforts spent in this field. The majority of P2P research has focused on the low level aspects of P2P technology (routing strategies, search algorithms, etc.) and little work has been carried out on the actual design process for P2P application development. Thus, developing a P2P application often requires detailed knowledge of low level P2P protocols and mechanisms.

The main task of the VICTORY networking stack is therefore to provide a flexible framework that acts as an interface between low level functionalities of the underlying networking and client application developers. The middleware framework needs to provide a set of services for the application developers that can be used without detailed knowledge of their actual implementation.

When deciding on middleware technology, all architectural decisions had to be considered. It is clear that P2P technologies are a good candidate for the implementation of the VICTORY networking stack. The advantages of using P2P technologies are well known. They allow for greater scalability and provide single-point of failure resilience. These technologies are implemented as overlaid networks and therefore include mechanisms for traversal of firewall and NAT devices, thus enabling the delivery of messages to any peer in the network. Since in pure P2P systems every peer acts as both client and server (often noted also as 'servent'), at the same time these systems offer

capabilities of resource sharing between peers. On the other hand, the issues described in the previous chapter (Section 2.2) suggest that the use of a pure P2P approach is not a viable option for the VICTORY system. Therefore the VICTORY networking stack uses a combination of two types of protocols:

- High level P2P communication protocols and mechanisms, and
- Client-server communication protocols.

In the VICTORY project we selected JXTA [4] as the technology upon which the P2P protocols and services are built. The Java reference implementation (JXTA 2.4.1) was used. The JXTA project is an open source project originally conceived by Sun Microsystems Inc. and essentially provides specification of low-level protocols for building P2P virtual networks. The Sun's implementation of these protocols represents a programming library that can be used to solve a number of P2P related problems. This implementation allows for the establishment of a virtual network overlay on top of the Internet, without the need to implement all the P2P related functionality from scratch. The mentioned implementation is Java-based and therefore platform independent.

The standard Hyper Text Transfer Protocol (HTTP) communication is extremely reliable and well proven. However, the straightforward HTTP approach can only be used if the serving entity has a public and static IP address. Communication with services running on public and static IP addresses will therefore use HTTP based communications. It is true that P2P mechanisms could be used in this case as well but their use introduces additional overhead in communication and is therefore not meaningful. The part of the middleware framework that uses plain HTTP GET/POST request/response messaging will rely on the Apache Tomcat [5] server and Java servlet [21] technology.

Another important decision represents the selection of the networking technology for connecting mobile devices with the Internet-based segments of the VICTORY network. Since mobile devices differ much from one to another, developing a generic thick client application for all different kinds of mobile devices proved to be very difficult. Using the JXTA Java Micro Edition (JXME [6], a JXTA version for mobile devices) was considered but the conclusion was that this choice would be too risky. First, using JXME would considerably heighten the complexity of the client application. Second, there were some incompatibilities between the JXTA project and the JXME project in the past, making their interoperability difficult. The difference and incompatibility between mobile devices dictates to reduce the complexity of client applications on mobile devices as much as possible. Therefore it was decided that the mobile devices client will be a thin client, pushing as much functionality as possible to the dedicated gateway. For the communication of the mobile client with the gateway, the WS-I Basic Profile Version 1.1 [7] standards were chosen. Using such standards brings the independency between client and gateway implementation technology. The server technology that is used for the gateway implementation uses the Apache Tomcat [5] server with addition of Axis2 [8] module.

3 Networking stack architecture

In this chapter the networking stack architecture that was selected for the VICTORY project will be presented. To introduce the various system components and their services, the overall architecture of the system is presented first. Each of the components is then separately described in its own chapter. The description may occasionally refer to the terms and features related to the architecture implementation and at this point it might be misinterpreted. In such cases please refer to the section 4 of this deliverable, which thoroughly describes implementation aspects of the architecture.

3.1 VICTORY architecture

The main purpose of this section is to explain the overall VICTORY architecture and networking mechanism supporting the architecture. The architecture of individual components (desktop edge peer, mobile edge peer, and super peer) will be elaborated in detail in the following sections.

The overall architecture is presented in Figure 1. The VICTORY framework will be organized as an interconnection of “VICTORY islands” that form a VICTORY user community. Each island represents a group of edge peers connected to the super peer. Edge peers are running on end user devices and can be further divided (with respect to their capabilities) on mobile edge peers (MEP in Figure 1) and desktop edge peers (EP in Figure 1). While only one VICTORY user community is presented on the figure, the edge peers can in general be involved in several user communities at the same time. For more information on the overall architecture of the system we refer to [2].

As can be seen from the figure, four basic elements can be distinguished in the VICTORY system:

- Administration unit,
- super peers,
- desktop edge peers, and
- mobile edge peers.

The Administration unit represents the initial contact point of the system. There exists only one administration unit in the VICTORY framework. When the user wants to start using the system for the first time, he/she contacts the administration unit through a web page. The administration unit provides means for:

- Registering the user,
- configuring the client application according to the provided personal profile data,
- selection of desired community/ies, and
- downloading the properly configured client application.

The administration unit is in charge of configuring the client application, based on the user's profile data and current system state. For example, the selection of a "personal super peer", to which the edge peer will be connected, is performed. More information on this mechanism can be found in [9]. The administration unit also hosts the Digital Rights Management (DRM) server [10] and the ID Manager.

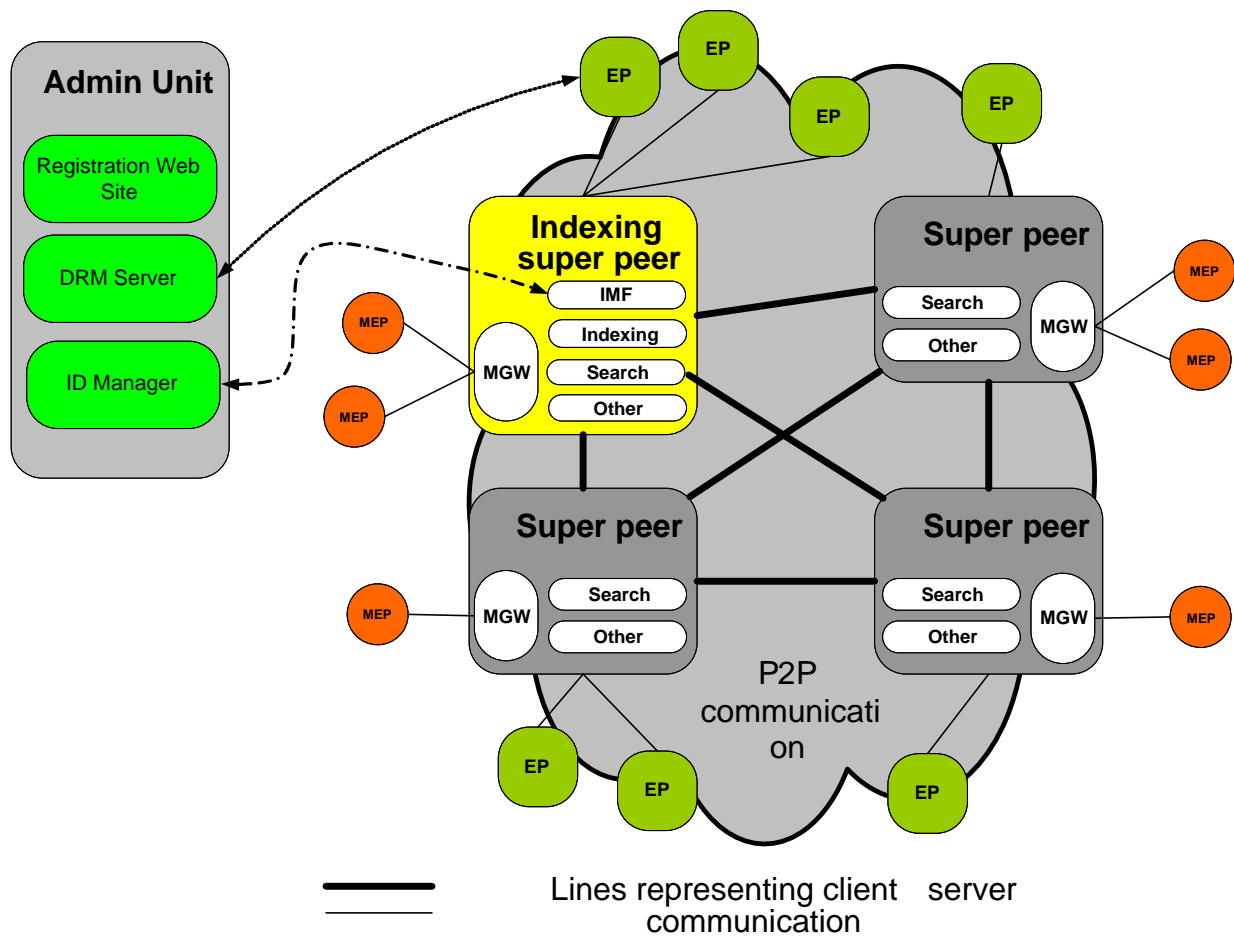


Figure 1: The overall VICTORY architecture.

Super peers are machines that are constantly connected to the system and offer both P2P and client-server functionalities. A super peer possesses / provides the following characteristics / functionalities:

- Super peers are reliable machines, running and connected to the network all the time. They have public and static IP addresses associated.
- Super peers act as servers hosting the services that are accessed through client-server communication.
- Super peers, just like edge peers, possess P2P functionalities, too. For example, they are capable to serve and to download content through P2P mechanisms and therefore play the role of the seeding servers in the system. Furthermore they act as P2P rendezvous and relay

peers, thus offering the advertisements searches and traversal of firewalls and NAT devices to the edge peers.

- Super peers are trustworthy machines from the security point of view.
- Each super peer manages a set of edge peers that are “directly connected” to it. In particular, each super peer knows all the edge peers that are currently connected to it, and the content they contribute (share) to MultiPedia repository.
- Super peers provide search and indexing services.
- Super peers, besides search, provide also other services: resources negotiation (Quality of Experience coordination services), identity management, rendering, visualization sessions management etc. These services can all be accessed in a typical client-server manner.
- Super peers also play a role of mobile gateways that connect mobile devices to the rest of the system.

Desktop edge peers represent client applications that are running on desktop PC machines. The desktop edge peers possess / provide the following characteristics / functionalities:

- Desktop edge peers are usually hidden behind NAT and/or firewall devices.
- Desktop edge peers can often be disconnected from the network. When they reconnect to the network, their IP addresses may change.
- Desktop edge peers register (or create a “connection”) with one and only one super peer, and provide it with all the information regarding the available (shared) contents, together with other general information (download pipe advertisement, heartbeat period, timestamp,...).
- Desktop edge peers provide low-level feature extraction functionality, annotation options and relevance feedback.
- Desktop edge peers act as content providers and consumers through P2P communication mechanisms.

Mobile edge peers represent client applications that are running on mobile devices. They possess / provide the following characteristics / functionalities:

- Based on their connection status, mobile edge peers can be seen as the most dynamic elements of the VICTORY framework. They are only occasionally connected to the network, and only for a short period of time. They connect to the VICTORY framework through mobile gateways provided by super peers.
- Due to limited resources of the mobile edge peers several functionalities that are usually available on desktop edge peers need to be provided on their behalf from other entities in the system.
- Mobile edge peers do not share content. They act only as content consumers.
- Mobile edge peers are using specialized servers (“rendering providers”) to render complex 3D content on their behalf. They are capable of participation in collaborative visualization sessions.

As can be seen in Figure 1 the architecture of the VICTORY system is not following the pure P2P approach, but is rather partially centralized. The system therefore uses a hybrid of P2P protocols and ordinary client-server protocols. The P2P mechanisms are mainly used for sharing and

distribution of the content and when the communication between entities that are behind NAT/firewall devices is needed. Communication with services that reside on public IP addresses is based on standard protocols such as HTTP.

3.1.1 Desktop edge peer architecture

Apart from super peers the desktop edge peers (running on home PC computers) are the most powerful (regarding processing speed, storage space and communication bandwidth) devices used in the project. A typical desktop computer is capable of running JXTA-based protocols and can therefore be connected directly into the P2P network by means of P2P communication stack. The architecture of the desktop edge peer networking stack is shown in Figure 2.

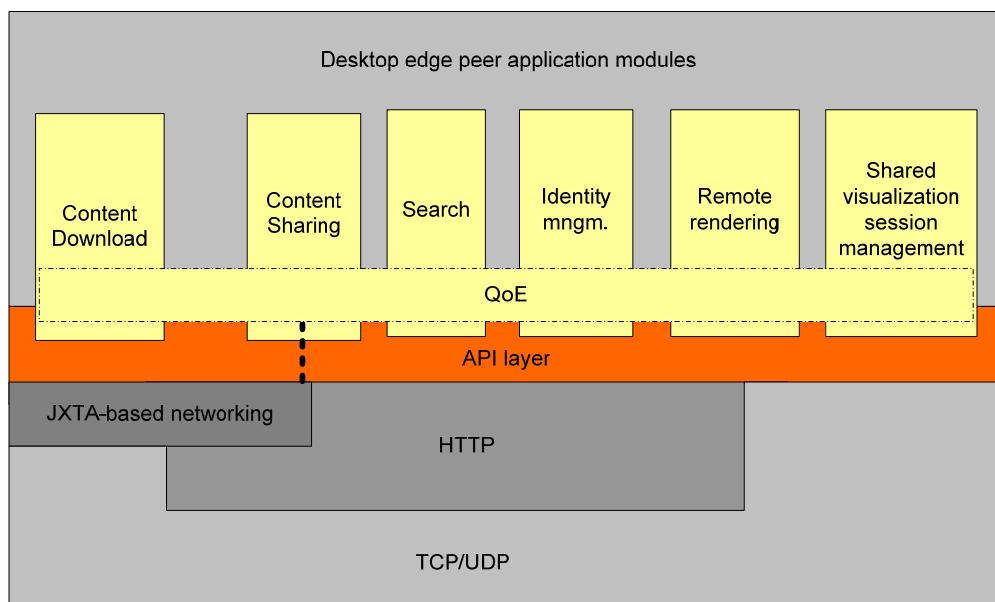


Figure 2: The desktop edge peer networking stack.

The desktop computer, as a high-end device, offers a wide set of functionalities, making the use and management of the distributed repository content easy. The desktop edge peers are capable of providing the following functionalities:

- publishing own MultiPedia items,
- searching for MultiPedia items using complex search queries,
- retrieving MultiPedia content from other peers,
- visualizing MultiPedia items, and
- modifying and re-publishing MultiPedia items.

These functionalities are achieved by calling different system services. The services are accessed through APIs of the networking stack and are called by application level modules. As can be seen in Figure 2 the desktop client application is composed of the following modules:

- the JXTA protocol stack, which provides low level protocols that enable NAT traversal and resolution of peers' endpoint addresses without knowing their physical address (IP, port). It is used to transmit messages using JXTA pipes mechanism. This protocol stack operates on top of mainstream HTTP and Transmission Control Protocol (TCP)/User Datagram Protocol (UDP) protocols.
- the content sharing module that enables sharing of files and handling of the accompanying information. It uses JXTA protocols (for advertising download services of the peers) as well as HTTP-based client-server communication (communication with central P2P database and indexing service).
- the content download module that handles incoming and outgoing download requests and serves/obtains content communicating with other peers. It acts as a client and server at the same time.
- the search module which is responsible for extracting low level features and formatting the eXtensible Markup Language (XML) document representing the search query. The query is then sent to the appropriate super peer and the response is collected. In case the client belongs to several communities the query is propagated to several super peers.
- the identity management module handles the user credentials in case authentication/authorization is required.
- the QoE layer that manages the quality of experience parameters.
- the rendering module which is responsible for sending rendering request and receiving incoming rendering streams.
- the visualization sessions management module that is responsible for joining and leaving collaborative rendering sessions, obtaining and releasing control tokens etc.

3.1.2 Mobile edge peer architecture

The use of mobile devices such as mobile phones and Personal Digital Assistant (PDA) devices allows accessing the VICTORY platform from any place, anytime, and in any context. But regarding the mobile devices capabilities there are some aspects to take into account. It is obvious that the small screen and keyboard of these small devices do not offer the same usability than a typical desktop PC. An overview of these limitations is:

- Small screen size/presentation space
- Small keyboard
- Low memory/storage space
- Low communication bandwidth
- Low processing power
- Different software environments (operating systems)

Due to the above mentioned limitations, the mobile devices are not capable of offering all the functionalities that desktop edge peers do. Some of these functionalities should therefore be

provided on behalf of mobile devices by the gateway. A set of functionalities that will be provided by the mobile gateway on behalf of mobile devices can be found in Section 4.2.1.

Because of mobile devices limitations, as well as software and hardware incompatibilities between different mobile devices, it was decided that the mobile client application will be as simple as possible, pushing the majority of the functionality to the mobile gateway. Another decision that was taken is that the communication with the gateway should use standardized web services based communication. The entire communication between the mobile client and the gateway therefore represents standard HTTP based client-server communication. Using standard web services the selection of mobile application technology will not depend on the gateway middleware technology. Therefore the mobile application can be built on the platform that is most suitable for the given mobile device. Depending solely on the Java platform could prove out to be risky as mobile devices have limited Java support that often varies over different mobile devices.

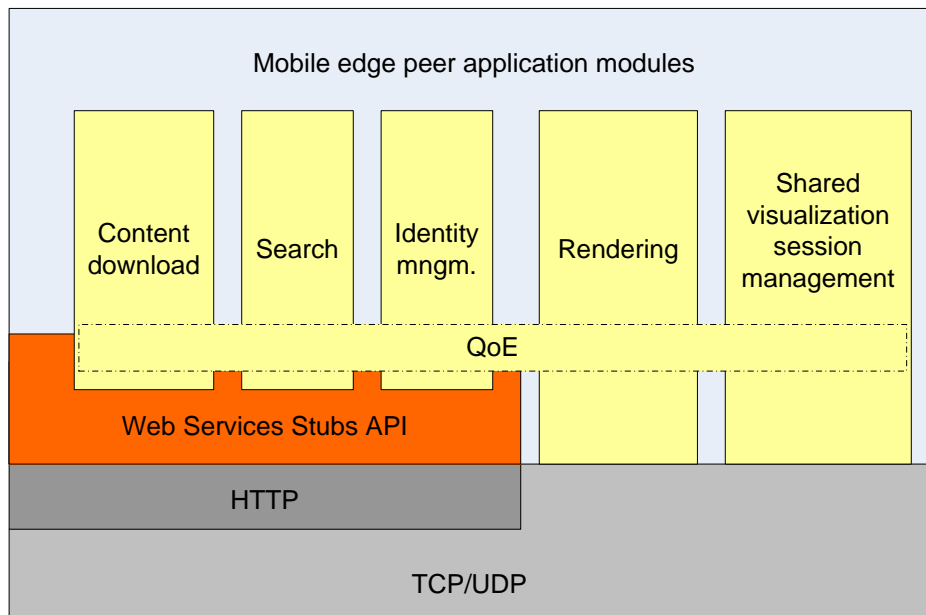


Figure 3: The mobile edge peer networking stack.

The architecture of the mobile edge peer is shown in Figure 3. The mobile client application is capable of providing the following functionalities:

- Content download through the gateway. The gateway downloads content from the P2P network on behalf of the mobile device and the mobile device downloads the content from the gateway using HTTP based client-server communication. Contrary to the desktop peer, the mobile peer does not provide the download service; therefore mobile devices are not content providers.

- Local rendering of 3D content of moderate complexity.
- Search for MultiPedia objects including composition of textual query, requesting low-level features extraction on behalf of the mobile device, concatenating the query with extracted low-level features, sending the query to the appropriate search engine and obtaining and showing the results.
- Identity management functions that handle the user credentials in case authentication/authorization is required.
- QoE layer that manages the quality of experience parameters.
- Negotiation of the remote rendering of complex 3D geometries that are too complex for local rendering.
- Shared visualization session management in case several users are participating in the visualization of an object. In this case only one user participating in a session is allowed to send commands that influence the visualization at a time.

3.1.3 Super peer architecture

The super peers provide a number of functionalities necessary for the integration of the VICTORY system into a functional whole. The core of the super peer represents its middleware, which consists of a number of functional modules. The architecture of the super peer is presented in Figure 4.

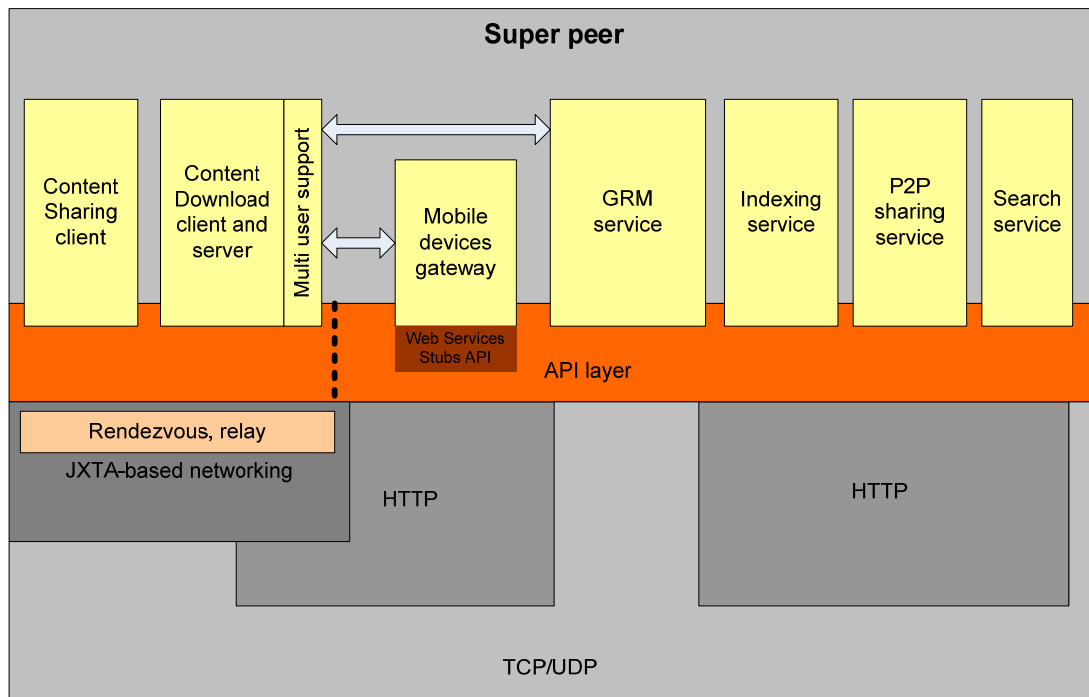


Figure 4: The super peer networking architecture.

The super peers offer services in an ordinary client-server manner, but they also act as P2P entity and are therefore capable of serving and retrieving content from the P2P network. Typical client-server services offered by super peers are:

- Rendering services,
- search and indexing services,
- services running on gateway for mobile devices,
- identity management services, and
- QoE services.

The P2P stack installed on super peers is needed mainly for four reasons:

- Super peers act as content seeding servers.
- Indexing super peers need to index textual attachments of MultiPedia objects. The textual attachments contain a lot of useful information that can be used when searching for MultiPedia objects. In order to index the documents containing text (.pdf, .txt, .html attachments) they need to be downloaded on the server where the indexing service is running.
- Super peers host rendering services that need to retrieve the content from P2P network before it can be rendered.
- Super peers act as rendezvous and relay peers.

As can be seen from Figure 4 a super peer contains the following modules (note that some of the super peers might not host all of the presented services/modules):

- the JXTA protocol stack: Provides the communication basis for the content sharing and content download modules. Additionally super peers include the JXTA proxying layer (rendezvous and relay), which enables the resolution of peers endpoint addresses (based on peerID) and communication and exchange of content to the edge peers hidden behind firewall and NAT devices.
- the content sharing module enables sharing of physical files that were downloaded to the super peer. Note that super peers are not capable of publishing new MultiPedia objects, neither do they need to search for them. They are only capable of sharing physical files that were downloaded from the P2P network.
- the content download module handles incoming and outgoing download requests and serves/obtains content to/from other peers. It acts as a client and server at the same time.
- the gateway for mobile devices hosts services that perform certain tasks (perform search, download content items from the P2P network on request, inform the mobile device about the status of ongoing downloads, serve the downloaded files to the mobile devices, etc.) on behalf of the mobile application.
- the graphics Resource Management (GRM) service provides rendering to devices that are not capable of visualizing the complex 3D content themselves. Additionally it manages the protocol of sending and receiving the collaboration tokens and other collaborative visualization session messages. For more information on GRM we refer to [9].

- the search service receives the queries from edge peers, performs search and returns the results. The search service includes the resource broker module which propagates the queries to the machines that currently offer the quickest response time.
- the indexing service is used when a new MultiPedia object is shared by one of the users of the system. The MultiPedia object metadata (low-level features, annotations, textual attachments) needs to be indexed by the search engine. There exists only one indexing super peer per community of users. All other super peers that provide search functionality use the same index provided by the indexing super peer. The indexing super peer also needs to access the content sharing and content download modules of the P2P stack in order to download and index textual attachments of the MultiPedia objects. The P2P network appears as a virtual file system for the search engine.
- the P2P sharing service is used when a new MultiPedia object is shared by one of the edge peers or when downloaded files are automatically shared in order to be accessible to other peers in the network.

4 Implemented networking stack specification

The prototype networking stack implemented within the VICTORY project serves two main purposes. First of all it implements the protocols and semantics for MultiPedia content advertisement, discovery and delivery. This enables the access to the distributed MultiPedia content repository and allows for contribution of content to the repository. Secondly, the prototype implementation includes the mechanisms for interworking between the mobile and Internet-based segments of the system. For this purpose a mobile devices gateway, which enables the access to the system from mobile phones and PDA devices, has been developed.

It is important to note that while the Annex I [11] proposes the development of mechanisms for identification of peer neighbours, this functionality was not implemented. The identification of two kinds of neighbours was proposed. The community neighbours were meant to represent a group of peers that share common interest fields, while the generic neighbours were meant to represent a group of peers that are separated with the least physical hop counts between them. It was supposed that the proposed functionality would significantly improve the performance of the search engine. The omission of these mechanisms can be treated as a consequence of the chosen architecture that was selected for the VICTORY project. The identification of neighbours proves meaningful only if the search mechanism in the system is based on unstructured overlays. In this case the search service is installed on every peer of the system. In such systems the search queries are propagated from one search service to another according to different strategies. The strategy can be either flooding [12] or random walks [13] or the algorithm for query propagation can implement some kind of propagation intelligence. This intelligence can, for example take into account peer neighbours in order to determine to which peers the query should be propagated. Since the VICTORY project adopted a partially centralized architecture, in some respects similar to Grid systems [14], the query propagation mechanism of this kind is not needed. In VICTORY the edge peer always sends the query to the same search service which then uses a request broker that forwards the requests to the service that currently serves the smallest number of users.

4.1 Mechanisms for content advertisement, discovery and delivery

The P2P mechanisms in the VICTORY project are mainly used as a method of distributing files over a network. Regarding the functionality, the P2P stack has both client and server properties. As a server, giving access to the shared files allowing for the other peers to download them, and as a client, accessing and downloading the shared files of the other peers. Using P2P software, a client can receive files from other clients, after a successful search, based on their metadata descriptors. In VICTORY, the users are actually searching for MultiPedia content, i.e. sets of files, each set being composed from 3D object and accompanying information (textual, 2D, video and audio files). Once a MultiPedia object is found the user can select the files he/she wants to download from the repository. The MultiPedia content advertisement, discovery and delivery therefore consist of three basic functionalities:

- 1) Sharing the MultiPedia content so it is available to other peers.
- 2) Searching for MultiPedia content available in the system.
- 3) Downloading of the physical files that constitute the MultiPedia objects.

MultiPedia content sharing is the activity of making MultiPedia objects (sets of files) available to other users in the network. This procedure therefore includes indexing at two levels. First of all the metadata describing these objects needs to be indexed by the search engine. Based on the indexed metadata, the objects can be found by submitting a query that is mapped to metadata descriptors. When the MultiPedia objects are found the user gets the results list together with information describing which physical files constitute the MultiPedia object (the unique file identifier is returned for each file). In a P2P model, the files are stored on and served by personal computers of the users. Therefore in order to download a certain file (with the given file identifier) the client application needs to obtain information about the peers that store the file in question. This information needs to be published and needs to be accessible to the client software. The second step of content sharing therefore represents the publishing of this information in the network.

The procedure of searching for MultiPedia objects contains the generation of the search query and the submission of the query to the appropriate search service. The search service computes the similarity between the query and the metadata of MultiPedia objects that are stored in the search engine index and returns a ranked result list (more information on the VICTORY search engine can be found in [15]). In VICTORY the search services are not running on every peer in the network but only on super peers. This allows for use of ordinary HTTP-based communication between edge peers and search services and amongst search services themselves. The workload distribution is achieved by means of a request broker that propagates the queries to the search service which is currently serving the smallest number of users.

Most people who are engaged in MultiPedia object sharing are also downloading physical files that constitute the MultiPedia objects, and are offered by other peers in the system. Once a MultiPedia object has been discovered and some/all of the physical files that constitute this object are selected for downloading, the files can be downloaded to the local hard disk drive. The edge peers are usually located behind firewall and NAT devices, therefore in order to start downloading from these peers, mechanisms for NAT/firewall traversal were incorporated into the system.

4.1.1 Description of components

VICTORY developed a P2P stack for content advertisement, discovery and delivery built on top of the HTTP and JXTA protocol suite. The stack represents a cross platform application written in Java (compatible with Java 1.5.x and higher). The stack components are shown in Figure 5. Note that for simplicity reasons only one super peer is shown on the picture, while in a real system setup several super peers usually exist. Also the components of the client part of the P2P platform, which are also running on the super peer, are not shown on the picture, since they are very similar to the components of edge peers.

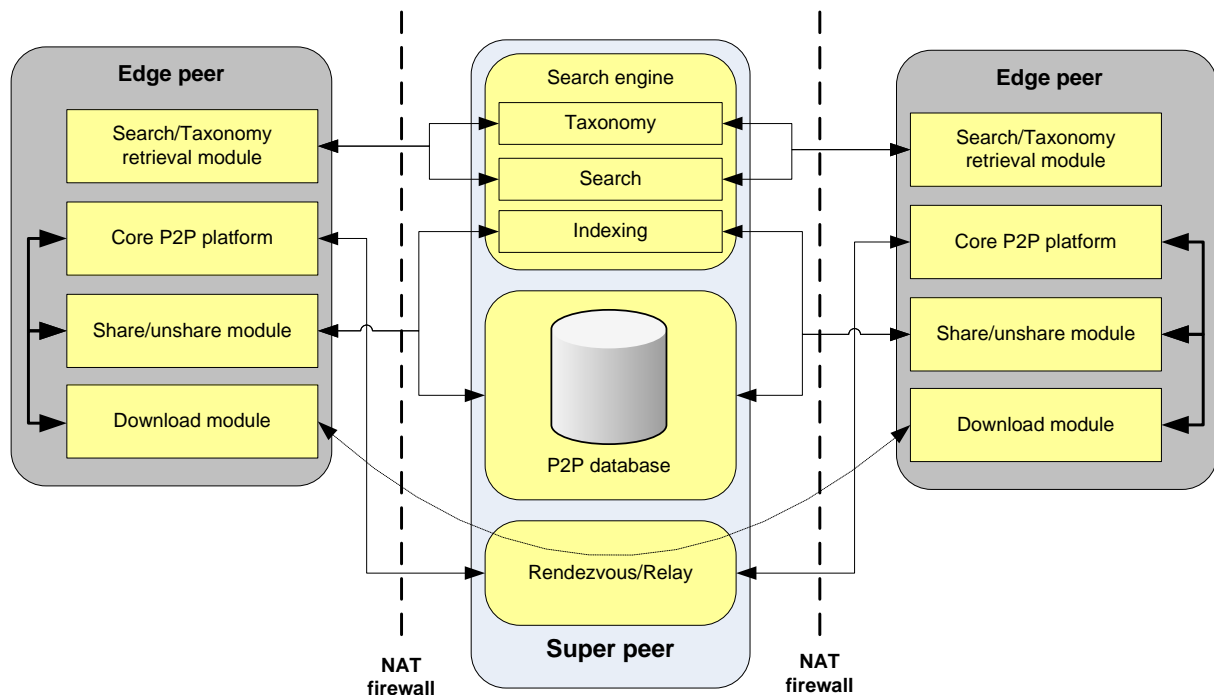


Figure 5: VICTORY components allowing for efficient content advertisement, discovery and delivery.

4.1.1.1 Edge peer components

Core P2P platform: This module implements the basic P2P platform functionality. It uses Java-based libraries of JXTA 2.4.1. The following actions are performed when the P2P platform is started (for more details see chapter 4.1.2.3):

- Automatic configuration operation,
- start up of core P2P services (discovery, rendezvous/relay, download),
- registration of peer data in the central P2P database,
- start up of module responsible for sending heartbeat messages to the central P2P database,
- local index initialization,
- consistency check on files that were shared by the users in previous application run-times,
- sharing of the consistency-checked list of files.

The Search/Taxonomy retrieval module deals with searching for shared MultiPedia objects, collecting responses containing references (that are later used to issue the actual content delivery) to shared files that constitute the objects and retrieval of MultiPedia objects taxonomy from the search engine. The module is responsible for low-level features extraction (in case of query by example), combining low-level features with the "high level" user formulated query in a complete query xml,

sending it to the search engine and collecting the search results. Additionally this module is used to retrieve the taxonomy of MultiPedia objects shared in the repository. This taxonomy is shown in the graphical user interface so that the users are able to select the categories while formulating the query. Only the objects matching the low-level features, user provided textual query and at the same time belonging to selected categories are returned as search results.

The share/unshare module is responsible for sharing/unsharing MultiPedia objects (i.e. making them searchable and retrievable from the P2P network). Sharing a MultiPedia object consists of two steps. First of all, an XML file that contains metadata descriptors (low-level features, user annotations etc.) is created and indexed by the search engine. Second, the information about the peer offering the files that constitute the MultiPedia object is inserted into the central P2P database. For each shared file a pair of unique file identifier and unique peer identifier is inserted. The unique file identifier is obtained by calculating the Message Digest algorithm 5 (MD5) checksum [16] of the first several KB of a file.

The download module represents the download service (both client and server) that deals with the actual media delivery from one or more locations in the P2P network. The download service therefore serves locally shared files and it downloads files from other peers. Logically it can therefore be divided into the download manager (client part) and download request handler (server part). The download manager keeps track of currently ongoing downloads (the end-user can issue several download requests in parallel) and reports on their statuses through a listener interface. Each individual download is started in a separate thread. The file that needs to be downloaded, is divided into data blocks (chunks) and those are retrieved simultaneously from several peers and assembled at the client side. The download manager sends requests for data blocks to the remote peers (their download request handler) that currently serve the requested files. The download manager is also handling the assignments rounds of data blocks. If a new peer serving the file in question connects to the P2P network some of the data blocks will be requested from this peer, too. When a remote download manager asks for a block (start index, end index) of data given by a unique identifier the local download request handler accesses the local index (based on Lucene [19] technology), obtains from it the Uniform Resource Locator (URL) path (path to the file on local hard disk drive) to the file with the given file identifier, extracts the requested block from the file on that URL, creates a download response message containing the block and sends it back to the requestor. The download manager also provides the mechanism for restarting interrupted downloads. If a file has been partially downloaded in the past, only the parts that are missing are downloaded the next time. Note that in VICTORY the downloading and sharing operations are actually linked together, meaning that every downloaded file is also automatically shared and therefore offered to other peers.

The components that were described above represent the components of the desktop (PC based) client application. The functionalities of these components are accessed from other client modules (graphical user interface for example) by calling respective Java methods. The API in a form of Java methods is defined in Appendix 1.

4.1.1.2 Super peer components

Search engine service can be divided into two subcomponents; the search and the indexing service. The search service is running on every super peer of the system. All super peers use the same index produced by the indexing service for the search purposes. Therefore the search service periodically requests index updates. When the user submits the query (formatted as an XML document) to the search service running on his "personal super peer", the search service takes the query and uses the request broker to propagate it to the search service that is currently processing the smallest number of requests. The selected search service processes the query and returns the results in a form of an XML document.

The indexing service of the search engine is running on the indexing super peer. In VICTORY, there is only one indexing super peer per user community. This service is used to index the metadata of new MultiPedia objects shared by the users. The communication messages between the edge peer and the indexing service are formatted as XML documents.

Additionally, as explained in Section 3.1.3, the indexing service needs an interface to the download functionality of the super peer. This was implemented by adding a new P2P connector to the connector framework of the search engine. In this way the P2P network appears as a virtual file system to the search engine.

The central P2P database stores information about peers that are currently in the P2P network and the files they are sharing. This information is used during the share and download procedures. A special mechanism for the identification of failed peers was developed. The failed peers are the peers that did not manage to unregister from the P2P network successfully. Usually, this situation occurs if the user switches off the computer before the VICTORY application is closed, or because of network failures. In VICTORY, the MySQL [17] database technology is used. The central database is accessed through a HTTP GET/POST request/response interface, which is specified in Appendix 4.

Rendezvous/relay services represent a core part of the P2P network. Rendezvous peers forward discovery requests to help other peers discover resources. When a peer joins the P2P network it automatically seeks a rendezvous peer. Each rendezvous peer maintains a list of other known rendezvous peers and also the peers that are using it as a rendezvous. Edge peers send discovery requests to their rendezvous peer which in turn may forward requests it cannot answer to other known rendezvous peers. A relay peer maintains information about the routes to other peers and routes messages to peers. Relay peers also forward messages on behalf of peers that are not directly addressable from the perspective of other peers (due to firewall or NAT environments), bridging the different physical and/or logical administration domains.

Client P2P components are running on super peers, too. While it is not evident from Figure 5, the super peers, in addition to providing services that are not running on edge peers, act also as edge peers by themselves. Each super peer hosts a slightly modified (multi-user support) P2P stack. The super peers need the P2P stack for content advertisement, discovery and delivery due to the following requirements:

- Super peers act as seeding servers. They represent reliable machines that are always connected and have good network connectivity. Therefore it is meaningful if super peers are content providers since the content availability is enhanced in this way.
- The indexing service that is running on super peer needs to be able to download textual attachments of MultiPedia objects in order to index the text they are containing.
- Rendering services that are running on super peers and render the content on behalf of other peers need to download the content before it can be rendered.
- Mobile gateways that are running on super peers need to download the content from the P2P network before it is served to mobile devices.

The APIs for accessing the super peer platform are implemented as HTTP/GET-POST request/response message exchange and are described in Appendix 2. The whole platform of a super peer is running in an HTTP container on the Jetty [18] server. In this way independence between the implementation technology of the modules calling the super peer platform and the implementation technology of super peer is achieved. The only requirement to the calling modules implementation technology is that they are capable of sending HTTP requests to the platform.

4.1.2 Content advertisement, discovery and delivery sequence diagrams

In this section the interactions between components involved in the process of content advertisement, discovery and delivery will be presented. The interactions will be presented in a form of sequence diagrams. Note that in order to present the interactions as clearly as possible the identity management procedures were not included in the sequence diagrams. Every service (except the download service that does not involve the identity management since the content is protected by DRM mechanisms) is performing the identity management procedures as part of its normal functioning. The sequence diagrams for identity management interactions can be found in [10].

4.1.2.1 Discovery and delivery sequence diagram

The sequence diagram of components interactions that are performed during discovery and delivery of MultiPedia content is presented in Figure 6. In the presented scenario the user first searches for a MultiPedia object and when it is discovered the download of the selected files that constitute the MultiPedia object is requested. The steps of the scenario are as follows:

- 1) The user prepares the search query. This includes low-level features extraction from provided 3D, 2D object or sketch, selection of category from taxonomy, formulation of user query string etc. The result is a search query in a form of an XML document.
- 2) The client sends the search query to "its personal super peer" (Super peer 1 in this case). The address of the "personal super peer" is stored in configuration file that was produced by the administration unit during user registration.
- 3) The query is propagated by the request broker to the search service that is currently handling the smallest amount of requests.
- 4) Search results (in a form of an XML document) are returned to the super peer.

- 5) Search results are returned to the client and presented to the user. The user then selects the files to download. Due to its general concept and architecture any kind of files can be exchanged through the networking stack. The request manager of the module then handles the individual file downloads separately.
- 6) Client sends a query for the download sources (i.e. the addresses of the download services of those peers that are sharing a certain file).
- 7) The request is sent to the central P2P database located on one of the super peers (Indexing super peer in this case).

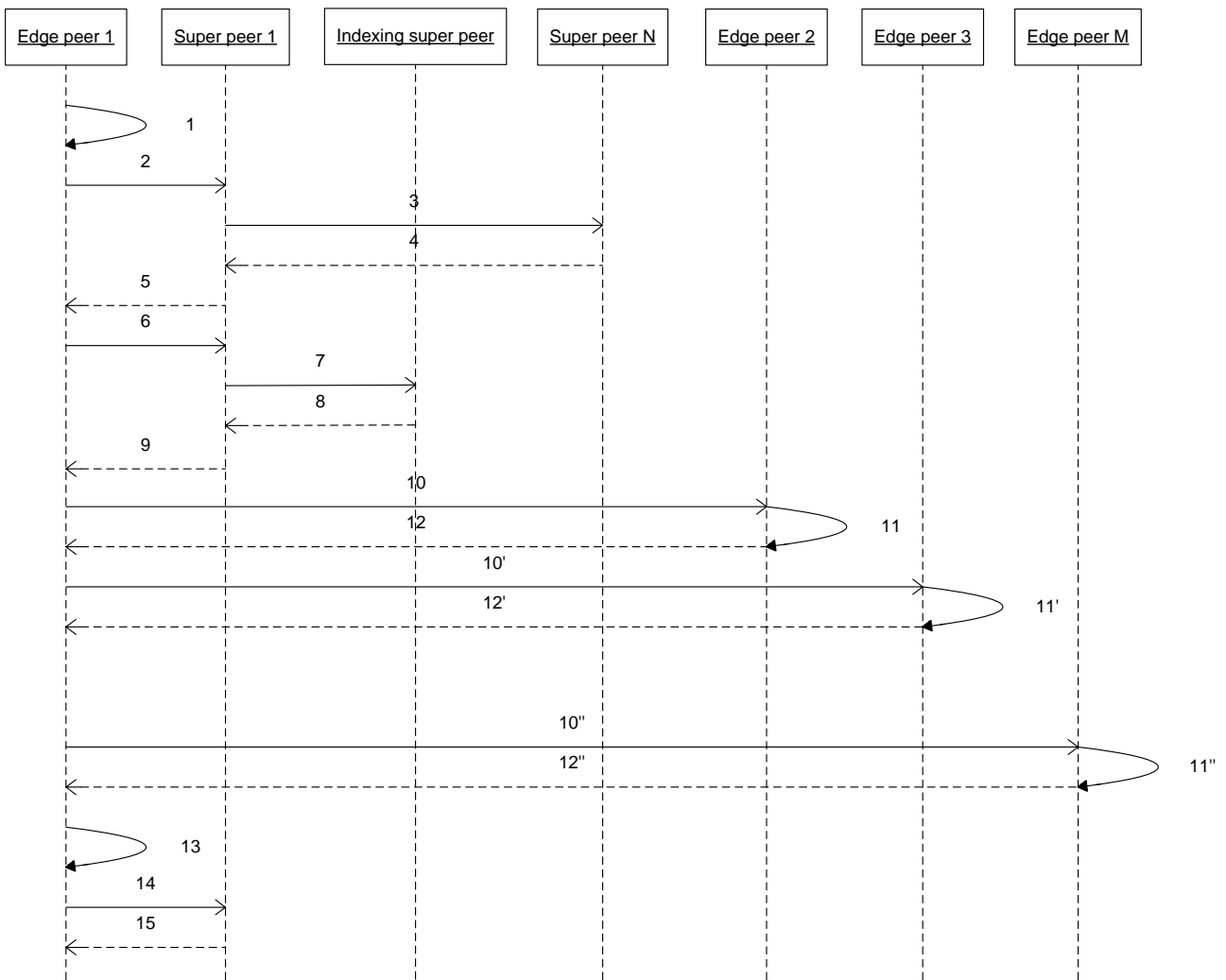


Figure 6: The sequence diagram for MultiPedia content discovery and delivery.

- 8) The database responds with the result. A list of download pipe advertisements of peers sharing the requested file is returned. The list has the form of an XML document.
- 9) A list of sources is returned to the client.
- 10) The client's request manager requests a certain data block from a certain edge peer. Every serving peer has a request handler module that is constantly waiting for requests for data blocks. The data transfer is based upon the JXTA pipes mechanism. The rendezvous peer is used to resolve the download pipe advertisement to the physical endpoint address of the serving peer. Once resolved the data block transfer request can be sent. In case the serving peer is located behind a firewall/NAT device, the transfer of the request is relayed through the relay super peer.
- 11) The serving request handler module receives the request. It retrieves the path to the file (URL of the file on the local hard disk drive) from the local index. For security reasons these paths are not known to the P2P network. The requests for data blocks contain only file identifier of requested file which is then resolved to the URL path locally.
- 12) The request handler opens the file, extracts the requested data block and serves the requested data block to the requesting client. Note that request manager sends data block requests to the serving peers in parallel (i.e. in separate threads). If serving of data block on given serving peer fails then the request manager requests this data block from the next available serving peer. The request manager keeps querying for the newly joined peers during the download. If a new peer joins the network and if it shares the requested file then the request manager starts sending requests for data blocks to the newly joined peer too.
- 13) When all data blocks of a given file are received and assembled into a complete file the client application locks the file. The locking of a file is needed in order to prevent the modification/deletion of a shared file during the application run-time.
- 14) For better content availability, every downloaded file is shared automatically. The information about the file is therefore inserted in the local index and the central P2P database. If the user does not want to share the file then it needs to be unshared manually.
- 15) The confirmation of the successful file sharing is sent to the client. From now on other peers are able to download data blocks of the file from this peer too.

4.1.2.2 Content advertisement sequence diagram

MultiPedia content needs to be advertised in order to be found and retrieved from the P2P network. This procedure requires the indexing of the MultiPedia objects metadata and registering the information about the peers that store the content in a central P2P database. The sequence diagram of components' interactions that are performed during the MultiPedia content advertisement is presented in Figure 7. The steps of the content advertisement scenario are as follows:

- 1) The client prepares an XML document containing MultiPedia object metadata. This step includes the extraction of low-level features and the acquisition of user annotations (including options that are provided to the user by the system).

- 2) The client sends the metadata to its "personal super peer" (Super peer 1 in this case). The address of the "personal super peer" is stored in a configuration file that was produced by the administration unit during user registration.
- 3) The metadata is propagated to the indexing super peer (there exists only one such super peer per community of users).
- 4) The metadata is indexed by the search engine.
- 5) Confirmation about indexing success/failure.
- 6) Confirmation about indexing success/failure.

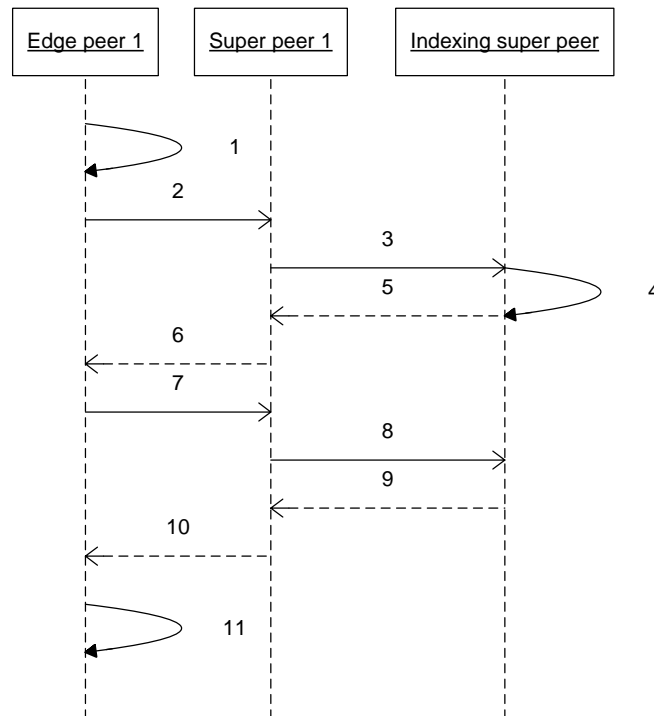


Figure 7: The sequence diagram for MultiPedia content advertisement.

- 7) The client sends information about the files constituting the shared MultiPedia object (in form of an XML document) to its "personal super peer". The information includes clients peer identifier and file identifier, file size and file name of every file constituting the shared MultiPedia object
- 8) The client's "personal super peer" forwards the metadata of files to the central P2P database which is located on one of the super peers (Indexing super peer in this case).
- 9) Confirmation about success/failure of database information insertion.
- 10) Confirmation about success/failure of database information insertion.

- 11) The files have been successfully shared so they need to be locked in order to prevent their deletion/modification during the application run time. Also the information about file size, URL path to file on local hard disk and timestamp for every shared file is inserted into the local index. From now on other peers are able to download data blocks of the files that constitute shared MultiPedia object from this peer, too.

4.1.2.3 P2P platform start up sequence diagram

As already explained the P2P platform acts both as client and as server at the same time. The platform therefore provides some services that have to be started and need to run as long as peer is connected to the system. Additionally the insertion of shared files information in the central database needs to be performed at platform start-up. The sequence diagram of the components' interactions that are performed during the P2P platform start up is presented in Figure 8.

The following actions are performed in sequence in the process of starting the P2P platform:

- 1) The client platform is configured as an edge peer that uses a given rendezvous server. The platform needs to publish its peer advertisement and download the pipe advertisement on the rendezvous peer. If the client is not on a public IP, it is also configured to use a given relay server. Following the configuration all client-side P2P services (discovery service, rendezvous service, download service) are started. A local database, in the form of a Lucene index, is also initialized. Each peer contains a local index that holds information about shared files (fileID/fileURL/fileSize/timestamp/isOurOwnFile). This information is used by the download request handler. When the download request handler receives the request for a certain block of file with the given file identifier it needs to find out where on the local hard drive this file is located. It finds this information in the local index. The local index is used also during the application start up (see step 13 of this sequence diagram).
- 2) The client connects to the rendezvous service that is running on its "personal super peer" (in this case super peer1) and waits until the connection is accepted.
- 3) Confirmation of successful connection to rendezvous service.
- 4) The client sends its peer information (username/ peerID/ downloadPipeAdvertisement /peerStatus /heartbeatTimestamp /heartbeatInterval) to its "personal super peer".
- 5) This information is propagated to the central P2P database that is running on the indexing super peer.
- 6) Confirmation of success.
- 7) Confirmation of success.
- 8) The client opens a separate thread and starts sending heartbeat messages (confirming its presence in the network) to the central P2P database. This is necessary to detect those peers which were not properly stopped. Since these edge peers did not have the chance to "unregister" from the P2P network, a mechanism for automatic unregistering of such peers was developed.

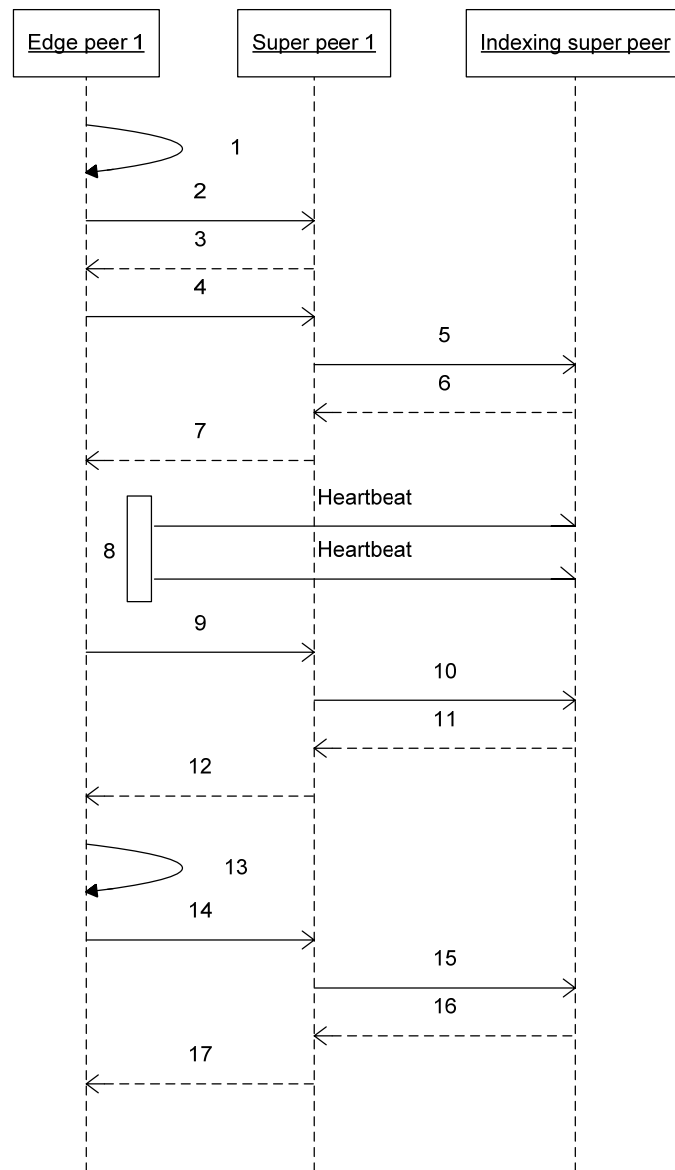


Figure 8: Sequence diagram for P2P platform start up process.

- 9) The client requests the unsharing of all its files from the central P2P database. The request is sent to its "personal super peer".
- 10) The request is propagated to the central P2P database and all entries for this client are deleted from the shared_Files database table.
- 11) Confirmation of success.
- 12) Confirmation of success.

- 13) The consistency check of files that were shared in previous application run times is performed. For every file whose metadata is found in the local index, two steps are performed. First, the client application checks if the file is still located on the hard disk drive of the user. Second, the client application checks if the file was modified during the application shut down (using timestamp and MD5 checksum metadata). If the file does not exist anymore or was modified then its metadata is deleted from local index.
- 14) Every file that passed the consistency checking needs to be shared at this moment. The client sends the sharing information of every file (in the form of an XML document) to the "personal super peer".
- 15) The XML document is forwarded to the central P2P database.
- 16) Confirmation of success.
- 17) Confirmation of success.
- 18) As the last step every file that was shared is also locked to prevent its modification/deletion during the application run time.

4.1.2.4 Platform shutdown sequence diagram

When the user closes the application several actions need to be performed. First of all, the files shared by the user need to be unshared, since they will not be available when the peer application is turned off. The local index needs to be stopped appropriately and all the peer services need to be stopped. The sequence diagram of the components' interactions that are performed during the P2P platform shut down is presented in Figure 9.

The steps of the P2P platform shut down process are as follows:

- 1) The client unshares all the files it shares from the P2P network. All peer identifier / file identifier entries for this peer need to be deleted from the central P2P database. The request for deletion is sent to the client's "personal super peer".
- 2) The request is forwarded to the central P2P database (in this case located on the Indexing super peer) where the appropriate entries are deleted.
- 3) Notification of deletion success/failure.
- 4) Notification of deletion success/failure.
- 5) After the files are successfully unshared the lock upon them is released.
- 6) A request to unregister the peer from the central P2P database is sent. Peer unregistering means the deletion of the peer identifier / download pipe advertisement information from central P2P database. This request is sent to the "personal super peer" of the client.

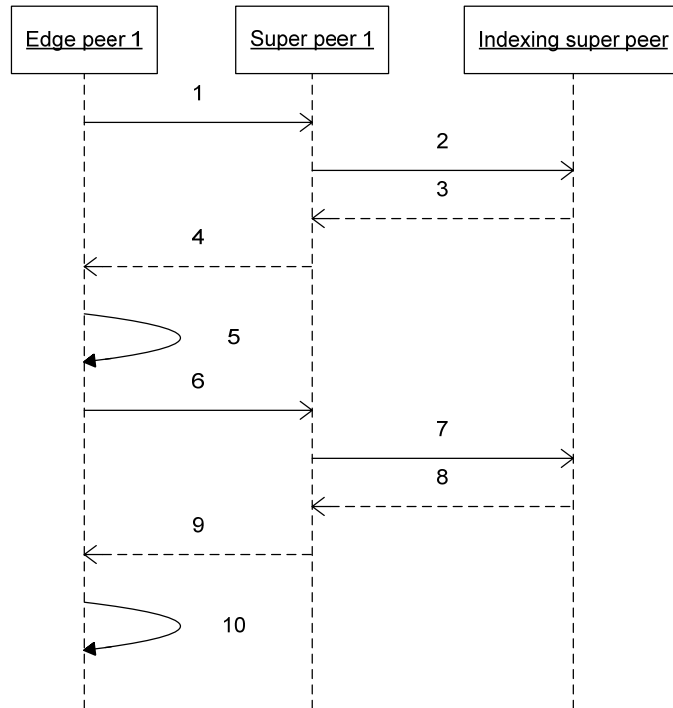


Figure 9: Sequence diagram for P2P platform shut down process.

- 7) The request is propagated to the central P2P database.
- 8) Notification of deletion success/failure.
- 9) Notification of deletion success/failure.
- 10) The local index is stopped properly. Heart beat message sending is also stopped.
- 11) All the P2P services (rendezvous, relay and download) are shut down.

4.2 Mechanisms for the effective interworking between the mobile and Internet-based segments of the VICTORY system

The interworking between mobile devices and the VICTORY system could be implemented in several ways. In the following chapters it will be first explained why a gateway based implementation was chosen. Then the presentation of the mobile client application will follow and the functionalities of the gateway that support the mobile client application will be explained. Finally the implementation technology of the mobile gateway will be presented.

4.2.1 Problems and limitations of the VICTORY system mobile segment

While developing the VICTORY mobile devices application, the simplest solution would be to port the entire Java-based application that was developed for the desktop edge peers, to the mobile device. In this case the mobile application would be a thick client offering similar functionalities as

the desktop client. Unfortunately, this solution is not possible because of the mobile devices limitations mentioned in Section 3.1.2.

The process of MultiPedia content advertisement, discovery and retrieval includes several tasks that cannot be performed on the mobile device. These include:

- The extraction of low-level features from 3D objects. This is a processing power consuming task which cannot be performed on mobile clients due to their low processing power.
- The visualization of complex 3D geometries that cannot be performed on mobile device, again due to resources constraint.
- Running the entire edge peer P2P stack on mobile devices proved to be difficult if not impossible. Mobile devices such as mobile phones, smart phones and PDAs cannot run the full JXTA middleware, because of the relatively high requirements that the JXTA middleware is imposing. Even though a mobile JXTA version has been developed (JXME - JXTA for Micro Edition [6]), porting a JXME based P2P stack to many different versions of mobile devices proved to be extremely difficult.

4.2.2 Gateway functionality

In order to understand what kind of services the gateway needs to provide, the functionalities of the mobile client application needs to be analyzed first. The mobile devices will not offer entirely equal functionalities as the desktop devices. This is mainly due to their limited processing power and limited storage resources. The following functionalities will be provided by the mobile client:

- Search for MultiPedia content in a particular community based on a number of parameters. The input to search can be a query by example (i.e. 3D object, 2D picture or sketch).
- MultiPedia content downloading to the mobile device.
- Rendering/visualization of 3D objects. The objects that need to be rendered can be of low, moderate or high complexity.
- Participation in a remote visualization session where multiple users are able to concurrently share the same data representation.

As explained, current Java profiles installed in the majority of existing mobile devices are not compliant with the requirements imposed by the VICTORY desktop edge peers. As we nevertheless wish to offer access to the system to mobile devices, the gateway that performs some functionalities on behalf of the mobile device had to be developed. It is desirable that the client application is a thin client. Therefore as many of the edge peer functionalities as possible should be outsourced and provided on behalf of the mobile devices. The following functionalities will be outsourced:

P2P file content retrieval: Since the mobile client is not capable of running the P2P stack the downloading from P2P network is provided by the gateway. The downloaded file is stored locally on the gateway and the mobile device accesses the file through the HTTP based file transfer.

Low-level features extraction: When a mobile device performs search by example the low level features need to be extracted. The extraction does not happen on mobile device itself, instead the

example 3D object is sent to the gateway where low-level features are extracted and returned to the mobile client. The low level features are then combined with the rest of the user query and sent to the search engine.

Rendering/visualization of the 3D content of high complexity: Objects of high complexity cannot be managed by a local approach; both the local rendering hardware and storage capability of any handheld device available on the market are not sufficient to manage objects described by millions of textured polygons. When the model cannot be visualized locally in an effective way, the rendering engine can remotely compute a flow of still images (or a video stream) for edge peers. A client-server approach is therefore needed. The VICTORY infrastructure will provide mobile users a set of special providers named “rendering providers”. Besides remote rendering the "rendering providers" are also in charge of shared visualization sessions management. It is important to note that the rendering providers are separate, stand alone servers that are not part of the mobile gateway.

The simplest use case scenario of the mobile application represents the case in which the 3D object that belongs to the MultiPedia object is rendered locally on the mobile device. This use case describes all the functionality that is required from the mobile gateway. The sequence diagram representing the components interactions involved in this use case is presented on Figure 10. The remote rendering and shared visualization session management that are needed in other use cases are handled by "rendering providers" servers and not by the mobile gateway. For more details on this functionality we refer to [20]. The components' interactions that are performed in the above mentioned use case are as follows:

- 1) The mobile client initiates a session creation. It provides username and password to the gateway.
- 2) The gateway creates a unique session identifier and returns it to the mobile client. The session details are stored in a list of currently ongoing sessions on the gateway side.
- 3) The mobile client requests the taxonomy of MultiPedia objects in order to populate the graphical user interface (GUI). The request is sent to the gateway.
- 4) The gateway forwards the request to the search engine running on the same super.
- 5) The taxonomy is returned to the gateway service.
- 6) The service returns the taxonomy to the mobile client.
- 7) The client wants to perform the search operation. The search operation is (also) content based, so low level features need to be extracted from the provided content. Since the mobile client is not capable of running the feature extraction algorithm the provided content needs to be uploaded to the gateway first.
- 8) Notification of content upload success/failure.
- 9) Request for feature extraction from the uploaded content is sent to the gateway.
- 10) The low level features are extracted.
- 11) The low level features are sent back to the client.
- 12) The client now formulates a complete query (combines low-level features with textual query from the user) and sends it (in the form of an XML document) to the gateway.

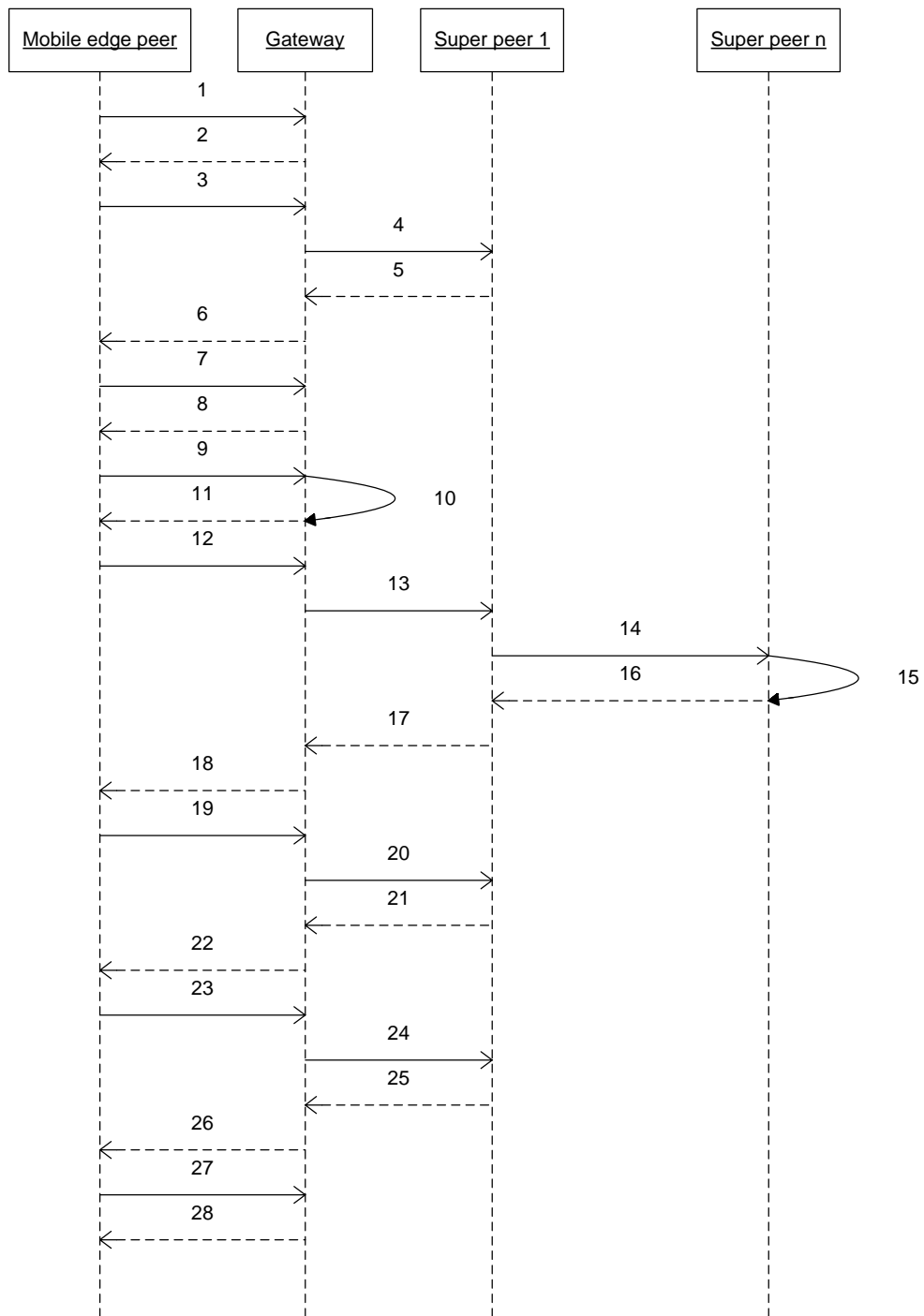


Figure 10: The sequence diagram representing the interaction between mobile client, gateway and other system components.

- 13) The gateway forwards the query to the search engine that runs on the same super peer.
- 14) The search engine request broker propagates the search query to the search service that is currently serving the smallest number of requests (service running on super peer n in our case).
- 15) The search service formats the results list.
- 16) The results list is sent back to the search service running on super peer 1.
- 17) The results list is propagated to the gateway.
- 18) The results list is propagated to the client.
- 19) The user is now able to select the file(s) to be downloaded to the mobile device. For every wanted file a request for file download is sent to the gateway.
- 20) The gateway requests a file download from the P2P platform running on the same super peer.
- 21) Notification of download starting success/failure.
- 22) Notification of download starting success/failure.
- 23) If the file download was successfully started the client needs to monitor its status. The client therefore periodically sends requests for the download status to the gateway.
- 24) The gateway asks the P2P platform of the super peer about the download status.
- 25) The information about the file download (including file identifier, status of the download, download percentage, download byte rate and the URL of downloaded file (once the download is completed)).
- 26) The client is informed about the status of the download. The steps 23, 24, 25 and 26 are performed periodically until the download reaches the status finished or error.
- 27) If the download status is "finished" the client starts the HTTP based download from the gateway.
- 28) The downloaded file is delivered to the mobile client and can now be locally visualized.

4.2.3 Implementation technology

The gateway for mobile devices is implemented as a set of standard W3C Web Services following the WS-I Basic Profile Version 1.1 [7]. Using the web services approach, the mobile devices, wanting to invoke one of the functionalities, invoke a local function which forwards the function call to a remote gateway. The function executes on the remote machine and returns the results to the mobile device. The web services are deployed to the Apache Tomcat [5] server running the Axis2 [8] module. The API that allows the access to the gateway functionality is presented in Appendix 3.

5 Conclusions

This document presents the VICTORY flat, scalable, efficient and robust P2P-based networking stack, which allows high degrees of peer social autonomy serving the purposes of 3D content search, retrieval and manipulation. The document represents the specification and implementation description of the protocols and semantics for 3D content advertisement, discovery, and delivery. The developed prototype allows client applications to have access to the distributed MultiPedia content repository and be able to contribute to the repository. It presents a baseline communication mechanisms between clients and services that are involved in the VICTORY framework. The developed prototype also provides mechanisms that allow the effective interworking between the mobile and Internet-based segments of the VICTORY system. The prototype is described in details and the interfaces to the various components are provided in appendices.

It is important to note that the development of the networking stack and its interfaces is an iterative, fine-grained and progressive procedure towards the final version. This is a normal procedure that every software development process should follow (opposing to the waterfall model in which all the functionality is defined and implemented in one step). Minor modifications of the networking stack are therefore expected, until all system subcomponents and subsystems are built and all VICTORY modules have been integrated together.

Appendix 1: Edge peer content advertisement, discovery and delivery APIs

<u>Method Name</u>		
startP2P		
<u>Brief description</u>		
Starts the P2P platform. The actions executed during the start up of the platform are described in the sequence diagram in Section 4.1.2.3.		
<u>Inputs Parameters</u>		
Name	Type	Description
username	String	The username of the user.
password	String	The password of the user
<u>Return Parameter</u>		
void		

<u>Method Name</u>		
stopP2P		
<u>Brief description</u>		
Stops the P2P platform. The actions executed during the shutdown of the platform are described in the sequence diagram in Section 4.1.2.4.		
<u>Inputs Parameters</u>		
none		
<u>Return Parameter</u>		
void		

<u>Method Name</u>		
getTaxonomy		
<u>Brief description</u>		
Retrieves the taxonomy of MultiPedia objects from the search engine. The taxonomy is used for the user interface population.		
<u>Inputs Parameters</u>		
Name	Type	Description
taxonomyQueryXML	String	The XML representing the taxonomy query that is sent to the search engine.
<u>Return Parameter</u>		
String representation of the retrieved taxonomy (in the form of an XML document).		

<u>Method Name</u>		
search		
<u>Brief description</u>		
Retrieves the search results (MultiPedia objects metadata) that matches the given query from the search engine.		
<u>Inputs Parameters</u>		
Name	Type	Description
searchQueryXML	String	The XML representing the search query (including low level features).
<u>Return Parameter</u>		
String representation of the retrieved results list (in a form of XML document).		

<u>Method Name</u>		
share		
<u>Brief description</u>		
Shares a list of MultiPedia objects in the system. The metadata of objects is inserted into the search engine index and in central P2P database.		
<u>Inputs Parameters</u>		
Name	Type	Description
fileIdFileURLPairs	HashMap<String, String>	A list of file identifier/ file URL pairs for each file to be shared.
shareXML	String	A metadata describing the MultiPedia object to be shared (in a form of an XML document)
<u>Return Parameter</u>		
String representing the response of the search engine.		

<u>Method Name</u>		
unshare		
<u>Brief description</u>		
Unshares a given file (belonging to the client that issues this request) from the P2P network. When the file is unshared it cannot be retrieved from this client by other clients any more (i.e. the client that issued this request is no longer a resource provider for this file).		
<u>Inputs Parameters</u>		
Name	Type	Description
fileID	String	The file identifier (MD5 sum) of the file to be unshared.
file	File	The object representing the file to be unshared.
<u>Return Parameter</u>		
void		

Method Name		
getNumberOfSources		
Brief description		
Returns the number of online peers that are currently sharing a file with given file identifier.		
Inputs Parameters		
Name	Type	Description
fileID	String	The file identifier (MD5 sum) of the file.
Return Parameter		
The number of sources (type: long).		

Method Name		
getListOfSharedFiles		
Brief description		
Returns the list of shared files the client is currently sharing.		
Inputs Parameters		
none		
Return Parameter		
The list of metadatas (FileMetadata objects) describing the shared files (type: ArrayList<FileMetadata>).		

Method Name		
getFiles		
Brief description		
Downloads the files that are specified to be downloaded. Three options exist:		
(1) the file/s with given file identifier/s was/were already downloaded in the past and is/are already on clients hard disk		
(2) the file/s with given file identifier/s is/are currently being downloaded (previous request for downloads)		
(3) new download/ need(s) to be started for the file/s.		
Inputs Parameters		
Name	Type	Description
fileIdDownloadParametersPairs	HashMap<String, DownloadParameters>	A list of file identifier / DownloadParameters pairs that describe the files to be retrieved from P2P network. The DownloadParameters object contains a DownloadListener object and a flag specifying if the file needs to be shared automatically when it is shared or not. The DownloadListener object is an implementation of DownloadListener interface and is notified about the status of the download.
Return Parameter		
void		

<u>Method Name</u> stopDownloads		
<u>Brief description</u> Stops the downloads of the specified files.		
<u>Inputs Parameters</u>		
Name	Type	Description
fileIDs	HashSet<String>	The list of file identifiers of files whose downloading needs to be stopped.
<u>Return Parameter</u> void		

Appendix 2: Super peer APIs

Every super peer is acting also as a P2P client (i.e. it is capable of downloading and sharing files from the P2P network). The P2P platform that is running on super peers is similar to the edge peer platform, providing additionally the Rendezvous and Relay functionality. Moreover, the super peer acts as a proxy between the edge peer application and various system services. The edge peer application is always connected to its "personal super peer" and this super peer then forwards the requests to the appropriate servers in the system. The interface to the P2P platform functionalities and to the proxy is exposed as a set of HTTP GET/POST requests to the Java servlets [21] running in a container of the Jetty [18] HTTP server.

<p><u>Servlet Name</u></p> <p>DownloadServlet</p>
<p><u>Servlet description</u></p> <p>This servlet is called when a P2P download of a certain file to the server (hosting servlet) is required.</p>
<p><u>Request parameters</u></p> <p>fileID</p> <p style="padding-left: 40px;">This is the unique file identifier of the file to be downloaded (MD5 sum).</p> <p>clientID</p> <p style="padding-left: 40px;">This is unique client identifier of the client that requested the file download.</p>
<p><u>Response</u></p> <p>Response string is "true" if the download successfully started or "false" if not.</p>

<p><u>Servlet Name</u></p> <p>GetDownloadStatusServlet</p>
<p><u>Servlet description</u></p> <p>This servlet is called to report on the download status of a certain file.</p>
<p><u>Request parameters</u></p> <p>fileID</p> <p style="padding-left: 40px;">This is the unique file identifier of the file that we want to obtain download status of.</p>
<p><u>Response</u></p> <p>Response string is ";" delimited string with information about file identifier; status of the download; percentage; byte rate; URL of downloaded file. The file identifier is MD 5 sum of file. Status of the download can be -1 (FILE_UNAVAILABLE), 1 (DOWNLOAD_IN_PROGRESS), 0 (DOWNLOAD_FINISHED), 2 (DOWNLOAD_ERROR). Percentage represents how many percentages of file have already been downloaded. The byte rate is expressed in number of bytes per second. URL of downloaded file is the URL path to the file on server (once downloaded).</p>

<u>Servlet Name</u> StartRdvRelayAndClientP2PServlet
<u>Servlet description</u> This servlet is used to start the P2P platform (including Rendezvous server and Relay server) on the server. It is called during server start up process
<u>Request parameters</u> none
<u>Response</u> Response string is "true" if the P2P platform started successfully and "false" if the initialization of the P2P platform failed.

<u>Servlet Name</u> SubmitIndexingRequestToSearchEngineServlet
<u>Servlet description</u> This servlet is used to submit metadata of a MultiPedia object (in a for of an XML document) to the indexing service of the search engine.
<u>Request parameters</u> InputXML This is the metadata of MultiPedia object to be indexed in a form of an XML document
<u>Response</u> Response string is containing the response of the search engine's indexing service.

<u>Servlet Name</u> SubmitSearchRequestToSearchEngineServlet
<u>Servlet description</u> This servlet is used to submit search query (in a for of an XML document) to the search service of the search engine.
<u>Request parameters</u> InputXML This is the search query in a form of an XML document
<u>Response</u> Response string is containing the response of the search engine's search service.

<u>Servlet Name</u> SubmitTaxonomyRequestToSearchEngineServlet
<u>Servlet description</u> This servlet is used to obtain a taxonomy of MultiPedia objects (in a for of an XML document) from the search engine.
<u>Request parameters</u> InputXML This is the query for taxonomy in a form of an XML document
<u>Response</u> Response string is containing the taxonomy of MultiPedia objects in a form of an XML document.

Appendix 3: Mobile gateway APIs

This is a servlet that allows for file upload to the server. Uploading large files through web services is not meaningful therefore the upload is HTTP/servlet based.

<p><u>Servlet Name</u></p> <p>MultipartFormDataReceiverServlet</p>
<p><u>Servlet description</u></p> <p>This servlet is used for file upload from the mobile client to the server.</p>
<p><u>Request parameters</u></p> <p>The request should be formatted as a valid multipart/form-data HTTP POST request. An example of form is:</p> <pre><FORM action=http://localhost:8080/VictoryServletsGateway/MultipartFormDataReceiverServlet enctype="multipart/form-data" method="post"> <P> Title: <INPUT type="text" name="title" />
 Description: <INPUT type="text" name="description" />
 Keywords: <INPUT type="text" name="keywords" />
 Crid: <INPUT type="text" name="crid" />
 File to send : <INPUT type="file" name="files" />
 <INPUT type="submit" value="Send" /> </P> </FORM></pre>
<p><u>Response</u></p> <p>Response string is "true" of "false" depending on the fact if the file upload succeeded or not.</p>

The following are the W3C Web Services standards-based web services that run on Apache Axis2 in the Apache Tomcat container. The functionality of the mobile gateway is therefore exposed as web services.

<p><u>Service name</u></p> <p>SessionHandlerService</p>
<p><u>Service description</u></p> <p>This service handles the currently ongoing user sessions. It creates a unique identifier for each session and stores this session ID together with user credentials (username, password) in a list of currently ongoing sessions. When the session is terminated it is removed from this list.</p>
<p><u>Service methods</u></p> <pre>public String createSession(String username, String password) Input: username and password (i.e. user credentials) Output: a unique identifier of the session public boolean destroySession(String sessionID) Input: sessionID - a unique identifier of the session Output: booleann value "true" of "false" depending on the fact if the session has been successfully removed or not public String getCurrentSessions() Input: void Output: String of current sessions IDs separated by ";"</pre>

<u>Service name</u>
GetTaxonomyService
<u>Service description</u>
This service will be called in order to retrieve the available categories of MultiPedia objects for a specific community. The taxonomy is used to populate the graphical user interface and is used during the search operation.
<u>Service methods</u>
public byte[] getTaxonomy(byte[] taxonomyRequestXML) Input: The request for taxonomy that is submitted to search engine (in a form of XML document transformed into byte[]) Output: The taxonomy of MultiPedia objects (in a form of XML document transformed into byte[])

<u>Service name</u>
GetObjectDescriptors
<u>Service description</u>
This service will be called in order to retrieve the low level features of a given object. These features cannot be calculated on mobile client therefore the file must first be uploaded to the gateway (see MultipartFormDataReceiverServlet above) and then this service can calculate the descriptors and return them to the client.

<u>Service name</u>
SearchForMultipediaObjectsService
<u>Service description</u>
This service is used to retrieve the ranked results list of MultiPedia objects that match our query.
<u>Service methods</u>
public byte[] submitSearchXML(byte[] xmlAsByteArray) Input: The search request that is submitted to search engine (in a form of XML document transformed to byte[]) Output: The taxonomy of MultiPedia objects (in a form of XML document transformed to byte[])
<u>Service methods</u>
Double[] getDescriptors(String pathToFile) Input: The path to the file from which the descriptors need to be extracted Output: The calculated descriptors.

<u>Service name</u> StartP2PDownloadOnSuperPeer
<u>Service description</u> This service starts the P2P download of a given file on the gateway.
<u>Service methods</u> public String startDownloadingFile(String fileID, String sessionID) Input: the fileID of a file to be downloaded and a sessionID that is used to identify which download belong to which session Output: string "true" or "false" depending on the fact if the download started successfully or not

<u>Service name</u> GetSuperPeerDownloadStatusService
<u>Service description</u> This service responds with the download status of a given file.
<u>Service methods</u> public String getDownloadStatus(String fileID) Input: The fileID of a file that we want to retrieve the download status for. Output: String that contains the download status. It is ";" delimited string with information about file identifier; status of the download; percentage; byte rate; URL of downloaded file. The file identifier is MD 5 sum of file. Status of the download can be -1 (FILE_UNAVAILABLE), 1 (DOWNLOAD_IN_PROGRESS), 0 (DOWNLOAD_FINISHED), 2 (DOWNLOAD_ERROR). Percentage represents how many percentages of file have already been downloaded. The byte rate is expressed in number of bytes per second. URL of downloaded file is the URL path to the file on server (once downloaded).

Appendix 4: Central database APIs

<p><u>Servlet Name</u></p> <p>GetDownloadSourcesServlet</p>
<p><u>Servlet description</u></p> <p>This servlet is used to obtain list of sources (download pipe advertisements of peers) that offer a specific file.</p>
<p><u>Request parameters</u></p> <p>fileID</p> <p>This is the file identifier of a file for which we want to retrieve a list of sources.</p>
<p><u>Response</u></p> <p>Response string is containing a list of download sources. For each of the download sources a peerID and download pipe advertisement is returned. The response has a form of an XML document.</p>

<p><u>Servlet Name</u></p> <p>RegisterUsernamePeerIdDownloadPipeAdvServlet</p>
<p><u>Servlet description</u></p> <p>This servlet is used to register a peer in a database. The username, peerID, download pipe advertisement and heartbeat period data is inserted into the database.</p>
<p><u>Request parameters</u></p> <p>username</p> <p>This is the username of the user that is registering.</p> <p>peerID</p> <p>This is the peerID of the peer application that is registering.</p> <p>downloadPipeAdvertisement</p> <p>This is the download pipe advertisement of the download service of the registering peer.</p> <p>heartBeatInterval</p> <p>This is the interval (in milliseconds) for sending heart beat messages to the central database. The server unregisters peers that fail to send heart beat messages from the network (these peers did not disconnect from the network appropriately so they need to be unregistered by the server).</p>
<p><u>Response</u></p> <p>Response string is "OK" or "ERROR" depending on the fact if the registering succeeded or not.</p>

<p><u>Servlet Name</u></p> <p>ShareFileOrFilesServlet</p>
<p><u>Servlet description</u></p> <p>This servlet is used to share a give file or files in the P2P network. For each file its fileID, file size, file name and peerID is inserted into the database.</p>
<p><u>Request parameters</u></p> <p>sharedFilesXML</p> <p style="padding-left: 40px;">This is the XML containing data of each individual file to be shared.</p>
<p><u>Response</u></p> <p>Response string is "true" or "false" depending on the fact if the sharing succeeded or not.</p>

<p><u>Servlet Name</u></p> <p>StartCleaningDatabaseProcessHeartbeatMessagesServlet</p>
<p><u>Servlet description</u></p> <p>This servlet is used for two things. When the servlet is first called it starts the process that unregisters failed peers (the ones that did not unregister successfully and did not send heart beat message for a specific period of time) from the database. The servlet also processes the heart beat messages sent by peers – it inserts the "new" timestamp of heartbeat message into the database.</p>
<p><u>Request parameters</u></p> <p>period</p> <p style="padding-left: 40px;">After a "heartbeat" message is received the peer is marked as "alive" in the database. If the next heartbeat message does not arrive in the interval defined by the period, the peer is considered to be failed peer. In this case the peerID/downloadPipeAdv pair of this peer is removed from the database.</p> <p>peerID</p> <p style="padding-left: 40px;">The unique identifier of a peer that sent the heart beat message.</p>
<p><u>Response</u></p> <p>Response string is "OK" or "ERROR" depending on the fact if the heart beat message processing succeeded or not.</p>

<p><u>Servlet Name</u></p> <p>UnregisterUsernamePeerIdDownloadPipeAdvServlet</p>
<p><u>Servlet description</u></p> <p>This servlet is used to unregister a peer from a database. The username, peerID, download pipe advertisement and heartbeat period data is removed from the database</p>
<p><u>Request parameters</u></p> <p>peerID</p> <p style="padding-left: 40px;">The unique identifier of a peer that will be unregistered from the database.</p>
<p><u>Response</u></p> <p>Response string is "OK" or "ERROR" depending on the fact if the unregistering process succeeded or not.</p>

<u>Servlet Name</u> UnshareFileOrFilesServlet
<u>Servlet description</u> This servlet is used to unshare a given file or files from the network. For each file its fileID, file size, file name and peerID is removed from the database.
<u>Request parameters</u> peerID The unique identifier of a peer that shares the file to be unshared. fileMD5Sum The unique file identifier of a file to be unshared.
<u>Response</u> Response string is "true" or "false" depending on the fact if the unsharing of file/files was successful or not.

References

- [1] VICTORY Deliverable D1.1: Report on user requirements and Use Cases.
- [2] VICTORY Deliverable D2.1: VICTORY architecture analysis and building blocks analysis.
- [3] W3C Web Services recommendations; retrieved from <http://www.w3.org/2002/ws/#documents> on 9.Jul.2008.
- [4] Project JXTA official web page; retrieved from <https://jxta.dev.java.net/> on 9.Jul.2008.
- [5] Apache Tomcat official web page; retrieved from <http://tomcat.apache.org/> on 9.Jul.2008.
- [6] JXTA JXME project official web page; retrieved from <https://jxta-jxme.dev.java.net/> on 9.Jul.2008.
- [7] WS-I Basic Profile Version 1.1; retrieved from <http://www.ws-i.org/Profiles/BasicProfile-1.1.html> on 9.Jul.2008.
- [8] Apache Axis2 official web page; retrieved from <http://ws.apache.org/axis2/> on 9.Jul.2008
- [9] VICTORY Deliverable D3.2: Quality of Experience (QoE) management framework.
- [10] VICTORY Deliverable D6.1: DRM and Identity management solutions for the VICTORY environment.
- [11] VICTORY Annex I – Description of Work.
- [12] Shudong Jin, Hongbo Jiang; Novel approaches to efficient flooding search in peer-to-peer networks; *Computer Networks: The International Journal of Computer and Telecommunications Networking*; Volume 51, Issue 10 (July 2007), pp. 2818-2832
- [13] C. Gkantsidis, M. Mihail, A. Saberi, Random walks in peer-to-peer networks, in: *Proc. of IEEE INFOCOM*, vol. 1, March 2004. pp. 130-140
- [14] Foster I., Kesselman K., *The Grid 2: Blueprint for a New Computing Infrastructure* (The Elsevier Series in Grid Computing., Morgan Kaufmann Publishers 2003
- [15] Empolis: Technical White Paper – e:Information Access Suite
- [16] RFC 1321: The MD5 Message-Digest Algorithm; retrieved from <http://tools.ietf.org/html/rfc1321> on 10.Jul.2008.
- [17] MySQL official web page; retrieved from <http://www.mysql.com/> on 10.Jul.2008.
- [18] Jetty web server official web page; retrieved from <http://www.mortbay.org/jetty-6/> on 10.Jul.2008.
- [19] Apache Lucene project official web page; retrieved from <http://lucene.apache.org/java/docs/> on 10.Jul.2008.
- [20] VICTORY Deliverable D6.4: Shared Visualization sessions protocol development.
- [21] Java Servlet Technology official web page; retrieved from <http://java.sun.com/products/servlet/> on 10.Jul.2008.