INTEGRATION OF A PEER-TO-PEER CLIENT IN A WEB SERVICE BASED APPLICATION

PREPARATION FOR THE MSc DISSERTATION

FINAL REPORT OF FRANCISCO A. BARBOSA

UNDER ORIENTATION OF M. TERESA ANDRADE (MSC) AND A. COSTA (ENG.)

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

FEUP

8TH OF FEBRUARY, 2008
<table>
<thead>
<tr>
<th>Index</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>3</td>
</tr>
<tr>
<td>Resumo</td>
<td>5</td>
</tr>
<tr>
<td>Chapter 1: Introduction</td>
<td>7</td>
</tr>
<tr>
<td>Chapter 2: Definitions</td>
<td>8</td>
</tr>
<tr>
<td>Chapter 3: Involved Technologies</td>
<td>9</td>
</tr>
<tr>
<td>Chapter 4: State of the Art</td>
<td>11</td>
</tr>
<tr>
<td>Web Services</td>
<td>11</td>
</tr>
<tr>
<td>JXTA</td>
<td>11</td>
</tr>
<tr>
<td>gSOAP</td>
<td>14</td>
</tr>
<tr>
<td>REST</td>
<td>14</td>
</tr>
<tr>
<td>Apache Axis – The third generation SOAP Implementation</td>
<td>16</td>
</tr>
<tr>
<td>Peer to Peer systems and technology</td>
<td>16</td>
</tr>
<tr>
<td>BitTorrent Protocol</td>
<td>19</td>
</tr>
<tr>
<td>Distributed Hash Tables</td>
<td>20</td>
</tr>
<tr>
<td>Mosaica</td>
<td>22</td>
</tr>
<tr>
<td>Chapter 5: The project</td>
<td>26</td>
</tr>
<tr>
<td>The Problem</td>
<td>27</td>
</tr>
<tr>
<td>The Solution</td>
<td>27</td>
</tr>
<tr>
<td>The Web P2P Client</td>
<td>28</td>
</tr>
<tr>
<td>Chapter 6: Work plan</td>
<td>29</td>
</tr>
<tr>
<td>Chapter 7: Conclusion</td>
<td>31</td>
</tr>
<tr>
<td>Chapter 8: References</td>
<td>32</td>
</tr>
</tbody>
</table>
Abstract

Peer-oriented computing allows several users in a network to share contents, like in server-client architecture, but in a scenario which every client (peer) is, at the same time, a client and a server.

A web service is a software application that allows distributed computing, exposing a group of functionalities using standard and common web technology.

These two concepts, although loosely coupled, serve the objective of create a more flexible and efficient way of communication. This report will introduce the concept of a peer-to-client client integrated into a Web Service, so that any user can connect to a peer-to-peer network, such as the MOSAICA network, using a normal HTTP Internet connection and retrieve contents distributed and downloaded in the peer-to-peer network.
Resumo

A computação orientada a peers permite a partilha directa de conteúdos entre múltiplos utilizadores, assumindo cada utilizador (peer) simultaneamente funcionalidades de cliente e de servidor.

Um serviço Web é uma aplicação que permite computação distribuída, oferecendo um grupo de funcionalidades que usam tecnologia Web standard.

Estes dois conceitos, embora apenas ligeiramente acoplados, servem o mesmo objectivo de criar uma comunicação flexível e eficiente. Assim, este relatório irá introduzir o conceito de um cliente peer-to-peer integrado num serviço Web, de modo a que um qualquer utilizador do serviço MOSAICA consiga, via internet, ligar-se ao serviço peer-to-peer e fazer download dos conteúdos que desejar.
Chapter 1: Introduction

The objective of this project is the specification and implementation of a Web based application, and its modules, to enable any user with an Internet connection and a web browser to access content distributed in a peer-to-peer network, with the possibility of searching and downloading the contents available in that network. This way, any user can access a peer-to-peer network, as for example a BitTorrent network, from any location, independently of software and hardware between the user and the network, such as firewalls or routers, taking advantage of distributed storage and download efficiency typical of peer-to-peer networks even though the user isn't logically associated with that network, without the need of having any peer-to-peer software installed.

The real peer-to-peer dient acts, to the Web application, as a Web server. It is responsible for returning lists of results to searches requested by the user through the Web application. Likewise, it is responsible for downloading specific content selected by the user through the peer-to-peer network and for sending it, possibly via HTTP or other defined protocol, to the user upon complete downloading.

This project is to be developed within the framework of the research project MOSAICA, an European project related with distribution and search of multimedia contents in the cultural heritage domain.
Chapter 2: Definitions

In the current report, some of the technologies, software or protocols involved will be referred by its acronym. So, I will introduce those acronyms and its meaning.

**DHT**: Distributed Hash Tables

**ERP**: Endpoint Routing Protocol

**IP**: Internet Protocol

**HTML**: HyperText Markup Language

**HTTP**: Hyper Text Transfer Protocol

**P2P**: Peer-to-Peer

**PBP**: Pipe Binding Protocol

**PDP**: Peer Discovery Protocol

**PIP**: Peer Information Protocol

**PMP**: Peer Membership Protocol

**PRP**: Peer Resolver Protocol

**REST**: Representational State Transfer

**SOA**: Service-Oriented Architecture

**SOAP**: Single Object Access Protocol

**TCP**: Transmission Control Protocol

**UDDI**: Universal Description for Discovery and Integration

**URI**: Uniform Resource Identifier

**URL**: Uniform Resource Locator

**URN**: Uniform Resource Name

**W3C**: World Wide Web Consortium

**WS**: Web Service

**WSDL**: Web Service Definition Language

**XHTML**: eXtensible HyperText Markup Language

**XML**: eXtensible Markup Language
Chapter 3: Involved Technologies

Inherent to this project, there are several technologies which achieved different kinds of development. In this section, these technologies will be exploited, to have a global view of them.

The two fundamental technologies in this project are Web Services and Peer-to-peer computing. Both of these are said, by some authors, as being in an early stage of development, and what is done with them, nowadays, are just a small part of what can be done in the future.

Web Services are one kind of communication between a user and a certain service, enabling the development of systems and applications with a Service-Oriented Architecture (SOA).

SOA is a software architectural type whose goal is to offer application’s functionalities as services, and it is based in the principles of distributed computing, using the request/reply paradigm. In a higher level, SOA tries to make easier to find, define and manage the available services. These tasks are related, as we can see in Fig. 1.

All services available in a SOA are stateless, which means that those services do not depend on other services’ state at the moment they are invoked. Once invoked, an application is linked to service applications by the service provider, task which is called binding.

Discover and binding services is made dynamically by users, which results in independency of compile-time, improving maintainability.

The use of WS technologies enables to build interoperable SOA systems, which can be easily integrated with other applications, services or systems. In order to achieve this, WS standards use XML as a base communication language.

Web Service Definition Language (WSDL) and Single Object Access Protocol (SOAP) are two technologies whose base is Web Services, but it adds the use of XML.

Using a WSDL interface, a service can be registered, and later found, in a service directory using Universal Description for Discovery and Integration – UDDI – one other kind of WS technology.
Peer-to-Peer architectures’ main goal is the sharing of computer resources, which may be contents, storage and/or CPU cycles (distributed processing power). This kind of architecture has several advantages, namely, the sustainability of acceptable connectivity, good performance and content availability. Although in a P2P network, participating nodes may be subjected to failures or may randomly present ‘off’ and ‘on’ periods (i.e., they can stay in the network for a short or a long time, without any kind of pattern), P2P networks have the ability to adapt to failures and transient populations of nodes.


*Communication and collaboration* category includes systems providing infrastructures for real-time communication and collaboration between peers. Examples of this are programs like instant messaging software – Jabber or IRC.

*Distributed Computation* category are all the systems whose purpose is to gather as many peers as possible, and use each processor power all together as one giant supercomputer, dividing one heavy task into small pieces, each of which will be executed by a peer, and, in the end, returning the results to the central server. Example of this is the very well known Seti@Home.

*Content Distribution* category is the more popular one, and it’s composed by systems and infrastructures designed for sharing digital media between users (peers). This category groups programs like Napster or Gnutella.

Also, they give a very clear definition of what are peer-to-peer systems. According to them, "Peer-to-Peer systems are distributed systems consisting of interconnected nodes able to self-organize into network topologies with the purpose of sharing resources such as content, CPU cycles, storage and bandwidth, capable of adapting to failures and accommodating transient population of nodes while maintaining acceptable connectivity and performance, without requiring the intermediation or support of a global centralized server or authority".
Chapter 4: State of the Art

Web Services

First generation of Web Services contemplates the older Internet connections: they were not integrated with each other and were not designed to be easily integrated with third parties in a uniform way. Mainly, the first generation was written to demonstrate the validity of Web Services as a concept, only. SOAP is an example of the possibilities of WS, without concerning Quality of Service.

Second generation Web Services handled the basic scenarios in a basic view. It was focused on a limited set of scenarios, which were not completely implemented. It had synchronous and request-response support, and was efficient when sending moderate quantity of information over the web. Axis and Microsoft SOAP are examples of software that belonged to second generation.

Nowadays, with an increasing need of transporting bigger quantities of information and with the introduction of Service Oriented Architecture, Web Services have special cares with several things, like be integrated with third parties in a uniform way and web extensions, such as security. This is the main focus of second generation web services. It uses structured languages, like XML or XHTML – which are standards and are defined according W3C -, and architecture type as REST or SOAP.

JXTA

JXTA is a peer-to-peer computing platform from Sun Microsystems, introduced in 2001, which was and it’s still being developed in an open-source context. This platform defines a group of protocols, sufficient enough to build almost any peer-to-peer application, maintaining protocols sufficiently flexible to be integrated to a great number of applications, and using platform independent language, like Java, it can be executed from different operative systems.

![JXTA Architecture](image-url)
Actually, JXTA defines six protocols. However, JXTA peers do not need to have all the six of them implemented, because only the peer’s capabilities are dependent on the number of protocols it has implemented, and it is just necessary one protocol to have the peer working correctly. These protocols are characterized by small overheads. They are:

- **Peer Discovery Protocol (PDP):** this protocol is used by peers to discover all published JXTA resources. This is the lowest-level protocol that can be found in JXTA, and it’s the more basic of the protocols. In highest-level protocols of resources discovery, we can find PDP as base for it.
- **Peer Resolver Protocol (PRP):** this protocol intends to standardize queries’ format, by making peers send generic queries to the network, and receive responses, also in a generic format.
- **Peer Information Protocol (PIP):** with this protocol, peers can ping other peers in the JXTA network. When a peer receives a ping message, it may reply with a simple acknowledge with its uptime, can reply with a full response, including its advertisement, or it can simply ignore it, since there can be peers capable of receiving messages but cannot send responses.
- **Peer Membership Protocol (PMP):** This protocol is used by peers to join and leave peer groups. This protocol recognizes four steps used by peers, defining messages for each process:
  - **Apply:** When a peer wants to join a group, it sends a membership request to the group membership authenticator. The authenticator responds with an acknowledge message to the peer.
  - **Join:** After apply step, a peer can choose to join the peer group.
  - **Renew:** a renew message is used by peers to update their membership information in the group.
  - **Cancel:** it’s a message sent by peers whenever they want to cancel their group membership.
- **Pipe Binding Protocol (PBP):** protocol used whenever a peer wants to create a new pipe, bind to an existing pipe or unbind from a pipe.
- **Endpoint Routing Protocol (ERP):** this protocol is responsible for helping peer routers query other peer routers about available routers for sending messages.

From the above protocols, ERP and PRP are lower level protocols, also known as core specification protocols. The other four, PDP, PIP, PBP and RVP are higher level protocols, also known as standard service protocols.

As JXTA tries to be as universal as possible, it needed to adopt a suitable representation for the majority of platforms, which came with XML. In fact, JXTA developers consider XML as “fast becoming the default standard for data exchange”. Although JXTA messages are defined in XML, JXTA itself does not depend on XML encoding, and a JXTA entity does not even need an XML parser, it can be an option.
JXTA, as a global system, comprehends many defined parts in the system:

- **Peers**: one entity in a network with one or more JXTA protocols implemented, that can be a computer, a mobile phone, a PDA or even a sensor. These entities existence is independent, and its communication is asynchronous.

- **Peer Groups**: when peers share common interests or contents, they can be grouped.

- **(JXTA) Messages**: provides the interoperability characteristic, as all communications in a JXTA network is made of messages. Messages can be encrypted at every endpoint's exit, which can grant security, integrity, authentication and confidentiality to JXTA systems.

- **Pipes**: virtual communication channels in the JXTA environment. Messages exchanged between peers are sent and received using pipes, which grants important abstraction, because the use of pipes allows peers to communicate without knowing their actual network address.

- **Services**: can be a peer service – when provided by a single peer – or peer group service – it’s available from multiple peers. Peer group services have more availability than a peer service, because if one peer fails, there are, usually, several others that can offer the same service.

- **Codats** (Code / Data): it's the content. Codats can be published, discovered and replicated.

- **Advertisements**: advertisements are represented as XML documents and are responsible for publishing and discovering JXTA resources, like peers, peer groups, pipes or codats.

- **Identifiers**: An identifier is defined as a URN (Uniform Resource Name), a kind of URI (Uniform Resource Identifier), and has to do with resources, not with physical network address. URN does not need special cares, but URI must stay unique, even after the resource disappears from the network.

- **World Peer Group**: is the group to which every JXTA peer belongs, even those who are disconnected. Every JXTA knows the world peer group and can join it when it goes online.

- **Net Peer Group**: it is a group of peers in a local network. Usually, network administrators configure a peer group so that every peer on that network can join.

- **Rendezvous Peers**: special peer that stores information about other peers it knows. That information has to do with advertisements caching of other peers. Rendezvous peers can help peers to discover other peers in the network and forward requests to other rendezvous peers.

- **Endpoints**: destinations in the network. It can be represented by a network address. Peers use, usually, endpoints through pipes, which are built on top of endpoints.

- **Routers**: peers that moves packets around in the JXTA network. Not all peers need to be routers, but those who aren’t, need to find a router to route their messages.

JXTA platform uses also a CMS – Content Manager Service – allowing it to share and retrieve contents. CMS is represented by a unique content ID within a peer group. CMS has content advertisement that provides metadata about the content.
gSOAP

gSOAP is a C/C++ approach of a SOAP application, presented by Robert van Engelen and Kyle Gallivan [1]. The developed compiler enables C/C++ and Fortran applications to share computational resources and information with other applications, and between different platforms, language environments, networks and even different organizations. This implies an increase of intercommunications and interoperability.

The strongest points in this approach have to do with support for native data types, due to pre-compiled routines, increasing performance of restructuring compiler techniques, causing them to be able of optimizing kernel’s routines or reducing memory access latencies. This implies minimizing memory operations, using just application’s native data structures, and consequently, minimizing memory use, which, being an advantage is also consequence of a limitation problem that commonly, exists in small devices, such as PDAs. Stand-alone client and server executables have a small memory footprint, typically less than 150 kB.

gSOAP’s runtime library includes a customized parser that parses XML on demand without keeping parts of a XML document in memory. This fact, that implies less use of memory, as said before, causes XML to be parsed more efficiently.

Finally, it enables the integration of Fortran legacy applications within SOAP clients and services through existing C-to-Fortran bindings, provided by the Unix linker or through Microsoft Windows dynamic-link libraries (DLLs), which makes it platform independent, causing SOAP clients to be able to run under Linux, Microsoft Windows, PocketPC and embedded systems.

REST

REST is the acronym for Representational State Transfer, exploited by Roy Fielding in his Ph.D. dissertation, which describes an architecture style of networked systems. REST is not a standard, as it’s not a technology, it’s a style, a way of making Web Services. However, it uses standards, like HTTP, XML or mime types.

In order to create Web Services in a REST way, it’s necessary to, first of all, identify all of the conceptual entities that will be used as a service. Next, it has to be created URLs to each resource, and it’s desired that URLs are designated by names and not by verbs. Resources are desirable to be accessible through HTTP GET command. When a client wants to modify resources, he can use HTTP POST, PUT or DELETE commands.

This architecture type also defines that revealing data should be a gradual process. In other words, a single response should only reveal a part of the information, and provided hyperlinks should be enable to reveal more details.
REST’s elements are summarized in the table below.

<table>
<thead>
<tr>
<th>Data Element</th>
<th>Modern Web Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>resource</td>
<td>the intended conceptual target of a hypertext reference</td>
</tr>
<tr>
<td>resource identifier</td>
<td>URL, URN</td>
</tr>
<tr>
<td>representation</td>
<td>HTML document, JPEG image</td>
</tr>
<tr>
<td>representation metadata</td>
<td>media type, last-modified time</td>
</tr>
<tr>
<td>resource metadata</td>
<td>source link, alternates, vary</td>
</tr>
<tr>
<td>control data</td>
<td>if-modified-since, cache-control</td>
</tr>
</tbody>
</table>

**Table 1 - REST’s Data Elements [6]**

Resources are divided into application state and functionality and one resource is uniquely addressable using a universal syntax for use in hypermedia links - URI. All resources share a uniform interface for the transfer of state between client and resource, and are constituted by a constrained set of well-defined operations and a constrained set of content types, that may support code on demand.

REST’s architecture claims several benefits:

- Improved response times and less load delivered to the server, due to caching possibility;
- Improved server scalability by reducing the need to maintain communication state;
- Requires simpler client software, because one browser can access more than one resource/service;
- Doesn’t require a separate discovery mechanism, since all services can be available through hyperlinks;
- Has longer-term compatibility and evolvability, since HTML or XML language can evolve without breaking backwards or forwards compatibility.
Apache Axis – The third generation SOAP Implementation

Axis is an implementation of SOAP, and one of the best known applications is Apache Axis, currently on version 2.2. It is the result of an attempt of obtaining SOAP engine much more flexible, configurable, able to handle both the SOAP and XML protocols. The Beta 2 release includes a SOAP 1.1 fully compliant engine, providing also partial support for some SOAP 1.2 functionality. Its main features include:

- Flexible configuration/deployment system;
- Support for "drop-in" deployment of SOAP services;
- A type mapping system for defining new serializers / deserializers;
- Automatic serialization/deserialization of Java Beans;
- Customizable mapping of fields to XML elements or attributes;
- Automatic two-way conversions between Java Collection and SOAP arrays;
- Providers for RPC and message-based SOAP services;
- Automatic WSDL generation from deployed services;
- WSDL2Java and Java2WSDL tools, allowing the creation of Java classes from WSDL documents and WSDL from Java classes.

Apache Axis also includes preliminary security extensions, support for session-oriented services, via HTTP cookies or transport-independent SOAP headers, preliminary support for the "SOAP with attachments" specification, an Enterprise Java Bean (EJB) provider for accessing EJBs as web services and a standalone version of the server, with HTTP support.

Peer to Peer systems and technology

Peer-to-peer technologies, commonly known as P2P are currently up to the third generation. The first generation of P2P appeared as a decentralized mechanism for any arbitrary person on a network, to simply request and share files, and with basic mechanisms of discovering files and its location. This was based in a point-to-point scheme.

Second generation P2P systems improved the concept of "share". Anyone could download a given file from anyone who was hosting and sharing it, simultaneously. The download process split the file into smaller pieces, and the client who is receiving data can assure that he requests all the parts of the file, and can verify each received piece and the whole file.

The third generation of P2P services is characterized by gaining new horizons, such as business and research use of it.

As peer-to-peer systems become more sophisticated, new functionalities are being created and integrated into software and into peer-to-peer networks. Features like security, anonymity, fairness scalability, performance, resources management and organizational capabilities are in constant development.
In what concerns to security issue, data integrity and authenticity is guaranteed by a simple method that forbid unauthorized entities of changing data or adversaries substitute a forged document for a requested one. Privacy and confidentiality may be assured by methods that ensure that data is only accessible to authorized peers and that there is control over what data is collected. Methods that ensure only authorized users have access to required data give availability, persistence and stability properties to the network.

Scalability of system is assured by methods that maintain system’s performance independently of the number of nodes available at a certain moment.

Fairness methods assure that users offer and consume resources in a balanced way, granting also performance to the network.

In what concerns to the information itself and its search and find capability, it can be grouped by schemes like semantic grouping, which is based on the content, can be grouped by location, based on network distance, or even grouped on organization ties.

When talking about network centralization, peer-to-peer networks can be categorized as:

- *Purely Decentralized Architecture*, where all the nodes in the network perform exactly the same tasks, being, at the same time, clients and servers, all with equal load. The searching method consists of nondeterministic searches, because there is no way of knowing where the requested files are. But sending requests to the network would result in flooding the network. So, requests have in the header a field that limit the spread of this kind of message – TTL, or Time To Live – that, once being zero, make the message to be dropped.

- *Partially Centralized Architecture*, which is quite similar to purely decentralized architecture, but there are some nodes that act as local central indexes for files shared by peers. These *supernodes* are automatically assigned, being this an automatic process, many times based mainly on CPU cycles available in each node. If one of these nodes fails, the network will replace it with another other. In networks with this kind of architecture, discovery times are small when compared to purely centralized systems.

- *Hybrid Decentralized Architecture*. In this type of architecture, there is one central server that maintains directories of metadata, describing the shared files. This central server helps the interaction between peers, but all exchanges of data occur only between peers. This architecture is of quite simple implementation and, due to server existence, location of files is quick and efficient. However, this is vulnerable to censorship, legal action, surveillance, malicious attacks and/or technical failures.

**FIG. 3 - TYPICAL HYBRID DECENTRALIZED PEER-TO-PEER ARCHITECTURE**
Peer-to-peer networks can be unstructured or structured. In unstructured networks, contents need to be located, because its storage is not defined according any algorithm or according to the content itself. Searching methods can be such as brute-force search or queries spread sent over the network, or a more sophisticated method based on random walks or routing indices. Although this fact, it's the appropriate type for networks with highly-transient population. Structured networks try to solve scalability problems that exist in unstructured types. Here, files or file pointers are stored at a specific location, and a distributed routing table routes queries to the nodes with the desired content. However, it's very complicated to manage the structure in order to have good performance when it's populated with very transient nodes. In this type of networks, exact match search are greatly efficient, but it needs to know the exact identifier of data in order to locate the node that store it. There are also networks whose content's location isn't completely specified, but queries may be affected by routing hints – Loosely Structured Networks. Freenet is an example of a loosely structured network.

Peer-to-peer can also be divided into five distinct models:

- **Atomistic**: It's the earliest and simplest approach to P2P. It involves direct client-to-client connectivity, without any mediation by servers. However, as no server is needed to mediate, it's not used to establish connections based on data availability;
- **User Centered**: This model uses a directory to provide an efficient way for users to make connections with others;
- **Data Centered**: This model allows users to search and access data that is stored on other user's system;
- **Web Mk2**: This model results from the above three models' convergence with actual web architecture.

**Compute Centered – Distributed Processing**: This model is designed so that an application’s processing task can be splitted between multiple clients, each of which takes responsibility for a to-be-processed task, and a central server coordinates the split to pieces and gathers the final result.
BitTorrent Protocol

BitTorrent is a protocol for sharing files and allows users downloading data to upload simultaneously, the data that they already have downloaded. A BitTorrent file distribution consists of:

- Web Server
- Static ‘metainfo’ file (which is commonly known as ‘torrent’ or ‘torrent file’)
- BitTorrent tracker
- ‘Original’ Downloader
- End user web browser
- End user downloader

Ideally, there are several end users for a single file, which implies a great availability for it.

Web server is used to store the torrent file or a pointer to it, from which a user can download the torrent, and start the downloading process with his BitTorrent client. The torrent file is generated using the complete file to be served to users. It includes the keys:

- **Announce**: it's the tracker's URL.
- **Name**: this key suggests the name to save the file or directory as.
- **Piece Length**: number of bytes in each piece of the file is split into. Usually, piece length is a power of two, most commonly, 256kB.
- **Pieces**: maps a string whose length is multiple of 20. Each string is the SHA1 hash of the piece at the correspondent index.
- **Length**: whenever length key is present, it means that the download is a single file. In this case, it maps the length of the file, in bytes. Otherwise, the key will be `files`, representing a set of files that will be saved into a directory.

BitTorrent protocol operates over TCP, and its connections are symmetrical, this is, messages sent in both directions look the same, and data is allowed to flow both ways.

When a user is downloading a file, the peer protocol is responsible for partitioning file in pieces by index, which is described in the torrent file. When a peer finishes downloading a piece, the protocol checks the hash and, if matches, announces to all other peers that it has that part, allowing the peer to perform the upload of that piece.

BitTorrent protocol connections start with handshaking. Handshake consists of the character nineteen (in decimal), followed by ‘BitTorrent protocol’ string. After these, eight reserved bytes are sent, and are all zero in current implementations of the protocol. Then, it’s sent the twenty bytes sha1 hash of the info value in the torrent file. Next, it’s sent a twenty-byte peer id, so that it can make part of the peers list that trackers broadcast to peer clients. This finishes the handshaking.

Web servers don’t store any information about the content location. They only store torrent files, and associated them with the URL of a tracker. Trackers are responsible for helping downloaders find each other, and they use simple protocol layers on top of HTTP. In detail, each downloader sends status information to trackers, and receives reply with lists of contact information for peers which are downloading the same file.
As can be seen in figure 8, one user goes firstly to a web page, downloads the torrent file, which contains the file length, name, hash and tracker’s URL. Using this URL, user contacts tracker, and tracker returns a list of peers from which the user can download the file.

**Distributed Hash Tables**

Distributed Hash Tables are a class of decentralized distributed systems that provides a lookup service similar to a hash table. It can also be understood as a virtual database maintained by nodes communicating with each other. Name/value pairs are stored in the DHT and participating nodes can retrieve the value associated with a name. In peer-to-peer systems, DHT gives a mapping from a set of data elements to a dynamic set of hosts. Hosts are mapped using a hash function and so are elements. DHT can also distribute responsibility for data storage, being each node responsible for some section of the space.

DHTs are good for distributed storage of files with known names, have high scalability, distributing automatically load to new nodes, are quite robust against node failures, except for bootstrap nodes and data automatically migrated away from failed nodes and it’s self organizing, not needing a central server. However, DHTs continue to have some security problems, like verifying data integrity. Also, its performance under attack decreases with number of nodes, as can be seen in figure 9.

DHT provides fairness by allowing all roles perform the same role and can join and leave the network at any time, and scalability, because the cost of maintain a network is relatively low even for bigger networks.

Actually, DHTs can be found in BitTorrent clients as a distributed tracker to provide rendezvous between clients downloading a particular file.
In what concerns to DHTs design, by convention, it’s assumed that keys are mapped to a single peer, being that peer the one responsible for storing a value associated with the key, such as the content of a file with a given name. To support this kind of design, DHTs must consist of two components:

- A consistent hashing over a one-dimensional space;
- An indexing topology, so that navigation through this space can be quick.

Making a consistent hash, both peers and keys are hashed into a 'ring'. Keys are then assigned to the nearest peer in the clockwise direction.

Servers are connected to their neighbors in the ring and searching for a key reduces to traversing the ring. Fast searches are enabled through additional overlays edges spanning larger arcs around the ring. [7]

DHT is also the base for Document Routing Model (DRM), one of P2P common discovery algorithm. In document publishing, the algorithm routes the peer whose ID is the most similar to the document ID and repeats the process until the nearest peer ID is the current peer's ID. Discovery algorithm search for peer whose ID is the most similar to document ID, repeating the process until the document is found.

DHTs can be measured by:

- *Cost of join/leave:* The service should be able to accept peer's state changes easily;
- *Congestion:* No server should be a bottleneck to service's performance. Load resulting from lookups should be distributed equally by participating servers;
- *Lookup path length:* The forwarding path of a lookup service should involve as few servers as possible;
- *Fault tolerance:* Even when some servers fail, the service should function as well as possible;
- *Dynamic caching:* When a server has the ability of caching popular data, it reduces the chance of causing bottleneck at and around its location. [9]
Caching ability, and more specifically, the dynamic caching protocol, should satisfy four properties:

- Prevent Swamping: since each server should handle as few messages as possible;
- Keep the cache small: Although storing all data in cache prevents swamping, it would have a lousy performance in terms of memory use;
- Reduce latency: Due to caching protocol, delays in obtaining data can occur, but they have to be controlled and reduced to minimum as possible;
- Keep update time low: although being stored to cache, data is constantly changing, which requires that caching protocol must handle changed items in cache.

DHT architecture is based in layers, as seen in figure 11.

![DHT Layered Architecture Diagram](image)

**FIG. 7 - DHT LAYERED ARCHITECTURE [9]**

Lookup services made using hierarchy can reduce number of hops, especially when nodes have heterogeneous availabilities, can reduce latency when groups can cooperatively cache popular files and can facilitate large scale deployment by providing administrative autonomy to groups.

**Mosaica**

Mosaica is an European funded research project, developing a peer-to-peer content management system for enabling the access to cultural heritage multimedia content. Mosaica takes advantage of peer-to-peer's technology, so it has no need of a centralized repository, it has high-availability of data and quite satisfying performances in file transfers, due to its' replication mechanisms, structured overlay networks and partition and distribution of resources among participating peers. Mosaica can be characterized as extensible, scalable, modular, flexible, of low cost and heterogenic in metadata schemes. In the current version, Mosaica supports semantic searches in peer-to-peer networks, has a service-oriented approach, resulting in an easy and efficient discovery of offered services, and it uses Distributed Hash Tables (DHT) to build structured peer-to-peer systems.
DHT allows efficient location of distributed contents, because indexes to files are distributed across the network, and all documents are identified by a hash key with an exact match lookup.

![FIG. 8 - TWO-LAYERED ARCHITECTURE OF MOSAICA [3]](image)

Mosaica is built on a two-layered architecture, as can be seen above, in figure 3. The upper layer is implemented with JXTA technologies and Web Services, being this layer responsible for the provisioning of functionality or services associated with content search, management of users, distribution and access to services. The lower layer uses the BitTorrent protocol, and is responsible for the distributed storage of contents, its replication and the access and reconstruction of digital objects. This layer is based on distributed hash tables.

In Mosaica system, normal users can login, register content, request content, search for contents, change password, list groups, receive messages from a group or even send a message to a group.

![FIG. 9 - UML USE CASE FOR DIFFERENT](image)
Administrators can perform operations related with manage users and groups, like adding a user or adding a user to a group, can manage content in Mosaica, like validate or remove contents, can manage metadata, which includes adding metadata, removing or list it and can coordinate metadata with contents – associate or disassociate content from metadata.

When a Mosaica user wants to download some of content from the network, first of all he has to search for what he wants, introducing words that can lead to the desired content. A semantic search is then initiated, and, hopefully, will return at least one match. The search process is made according to the scheme presented in figure 5.

FIG. 10 - SEARCH PROCESS IN MOSAICA [3]
The user, then, has to download the torrent to his computer, storing it locally. After getting the torrent file, the peer-to-peer client, using the torrent, start the process of downloading the data that user wants.

FIG. 11 - UML USE CASE FOR END USER OF MOSAICA [4]
Chapter 5: The project

It is desired that someone in a computer, and with an Internet connection can establish a connection over TCP/IP network, and in port 80 (HTTP), to a MOSAICA P2P client, who is a peer in MOSAICA P2P network. This connection is made from a web browser, who requests a session in MOSAICA P2P client. In order to better understanding, steps of a download process can be seen below.
The Problem

The great variety of network topologies and configurations makes it difficult for someone with an Internet connection to gain access to the content distributed across a P2P network. Apart from the fact that it is necessary to install dedicated P2P software, network administrative restrictions may block all the P2P traffic using firewalls or misforwarding the traffic in routers. Also, actual users are in constant movement, and may not access internet and P2P network from the same workstation, which implies that, if the user starts downloading something from one workstation, in order to resume or conclude the download later, he has to do it from the same workstation he started downloading.

The Solution

If there’s a machine connected to the Internet that can access a P2P network without any kind of problem or limitation, using port 80 of TCP/IP protocol, physical problems are overcome. Since HTTP traffic in a network is always properly forwarded by routers and isn’t blocked by firewalls, connection from a user somewhere in the Internet, using a Web browser, is never rejected or blocked. Also, this ‘Web’ user isn't doing any kind of P2P traffic, just HTTP traffic. So, network policies whose objective is to block P2P traffic do not apply to. Considering the use of the MOSAICA system, the download of the desired content could be requested to a MOSAICA peer, located within the P2P MOSAICA network. When the MOSAICA peer would finish the downloading of the requested content, it would transfer it to Web user as HTTP traffic and not P2P traffic.
The Web P2P Client

Conceptually, the parts of the system will be connected as represented below.

![Network Topology of the System](image)

When users access remotely to Mosaica P2P client's webpage with a browser, they will have an interface, made with Java, where they can work as they would in a regular peer-to-peer client. This implies that users must have Java Virtual Machine already installed. However, most operative systems nowadays have it already installed, so the fact of using Java does not raise any problem, maintaining the principle that users do not have to install anything so they can use this peer-to-peer web application.

The bridge between this interface and MOSAICA peer client is made with Java and XML. Java is responsible for translating user's request to XML, and will be XML code that will be sent to MOSAICA peer client, which, running on top of Axis2, already works as a web server, being accessible all over the Internet. Messages exchanged between both ends will use HTTP protocol.

Returned lists of results from web server will be sent to web client, who will receive them as XML code, being once more translated to the user by Java program, displaying it in the interface.

After download is complete in the web server, data will be transferred, possibly by HTTP, to the client, being the process monitored by the application made.
Chapter 6: Work plan

The following work plan has been devised as appropriate for the development of the thesis project. It should however be seen as a tentative planning, which will be updated upon completion of its first suggested phase.

Phase 1: Reflection and synthesis of the bibliographic research conducted during the preparation of the thesis. Duration: 4 weeks.

The outcomes of this phase of work should provide clear indications on the technologies to use and on the added-value with respect to the current state-of-the-art implementations.

The determining elements for the selection of technology include:

i) availability of open source tools;

ii) interoperability/compatibility with tools and technologies in use within MOSAICA;

iii) adequacy to the objectives of this thesis;

iv) usability (easy-of-use).

Simple installation and usability tests should be conducted with potential candidate tools and development environments.

This phase should conclude with a detailed study and “hands-on practice” of the selected technologies and tools.

Phase 2: Analysis and Specification. Duration: 3 weeks.

This phase will comprehend the following steps:

i) Identification and analysis of the possible application scenarios;

ii) Identification and characterization of dependencies with the MOSAICA system;

iii) Elicitation of requirements and of corresponding functionality to implement;

iv) Formal specification of the system adopting a UML approach.

This phase should conclude with the specification of the WSDL of the system to be developed.
Phase 3: Development. **Duration:** 8 weeks.

Phase 4: Integration and tests. **Duration:** 3 weeks.

Phase 5: Writing of the thesis report.
Chapter 7: Conclusion

After all the research and reading done in the context of Preparation for the MSc Dissertation to the current project, I can conclude that peer-to-peer technology is in a younger stage, several aspects of it can be developed, and new services like a web client are new approaches that can help this technologies become even more popular and more complete.

Although not being directly related to this project, DHT lookup operations are also an important subject, and several attempts to improve semantic searches with it are currently being tested and developed. This fact comes from the direct match returned from key/value pairs that do not allow direct semantic searches, implying the use different approaches. However, none of them are standard and its efficiency is still being tested.

Knowing these facts, I conclude that a peer-to-peer client integrated into a Web Service is a new, interesting and easy way of using a web-based peer-to-peer client.
Chapter 8: References


[28] X. Li, J. Wu. “Searching Techniques in Peer-to-Peer Networks”