Implementing a Peer-to-Peer Web Browser for Publishing and Searching Web Pages on Internet

Heverson Borba Ribeiro, Lau Cheuk Lung, Altair Olivo Santin, Neander Larsen Brisola
Graduate Program in Applied Computer Science (PPGIa)
Pontifical Catholic University of Paraná(PUCPR), Brazil
e-mail: {heverson, lau, santin, neander}@ppgia.pucpr.br

Abstract

This paper presents an implementation of a P2P Web Browser, called Web2Peer[1], for publishing web pages on internet through peer-to-peer networks. The present proposal overcomes the problems involving entities dependency and content availability, by the user side, changing the conventional way used to publish web pages on Internet. The proposed solution allows anyone who has a computer connected to the Internet, even through domestic ADSL connection, to publish web pages on their own machines. By using a peer-to-peer web browser another mechanism for searching web pages is offered based on DHT (Distributed Hash Tables). The user interface, provided by Web2Peer, carries all functions needed for creating, editing, publishing, searching, downloading and replicating the web pages on the peer-to-peer network.

1. Introduction

Following the constant improvements in the internet access technologies, the publication of web pages has increased significantly, and billions of pages [1] can be found on the internet nowadays. Although these technologies for accessing the web have gained an expressive development, the main centralized structure of the internet remains the same, based on client/server model. Basically, nowadays there are three forms of web page publishing on the Internet: the first is using a rewarded service provider, the second is using a free web page provider and the third is creating your own web server. The first option is characterized by paying someone (a provider) else to manage the content which creates a dependent relationship. In the second, there are no fees but the limitation of resources is apparent. The last option is characterized by the direct compliance with the regulatory entities [2], which provide IP addresses, internet domains and internet links, resulting in a high-cost environment. If it was possible to reduce these costs the internet content publication could be even more democratic. Another important characteristic is that none of the options mentioned above can guarantee the web pages availability. Centralized structures like that act as a single point of failure, consequently the web page can be unavailable at any moment. There are applications where this situation can be considered reasonable, but it is rejected in others. Therefore the current model is distinguished by two potential problems, its dependency on regulatory entities and its vulnerability to failures resulting in the web page's unavailability.

In attempts to address these issues the emergent peer-to-peer architectures offer the possibility of creating a network layer over the existing topologies, eliminating the network's limits. In the peer-to-peer networks, all nodes are consumers and providers, resulting in the increasing availability of the resources. It is done by the passive replication that happens when a resource is accessed by a consumer node, so this node becomes a new content provider.

In this paper is presented an implementation of a distributed infrastructure for publishing web pages on internet using peer-to-peer networks. The present solution overcomes the problems mentioned before, involving entities dependency and content availability, by the user side, changing the current ways used to publish web pages. The proposed solution allows anyone who has a computer connected to the Internet, even through domestic ADSL connection, to publish web pages on their own machines. By using a peer-to-peer web browser, called Web2Peer, another mechanism for searching web pages is offered based on distributed hash tables. Distributed hash tables act as a server mechanism, just like Google does, providing a way for users to find pages start from subjects. The user interface, provided by Web2Peer, carries all functions needed for creating, editing, publishing, searching, downloading and replicating the web pages on the peer-to-peer network.

[1] This work is partially supported by CNPq (Brazilian National Research Council) through processes 481523/2004-9 and 506639/2004-5.

This paper is divided into the following sections: Section 2, an explanation about the peer-to-peer technology, JXTA protocols and distributed hash tables (DHT). In Section 3, the implementation of the infrastructure is presented and its main parts. Section 4, some considerations about the working environment and obtained results. Section 5, conclusions and future improvements are presented.

2. Peer-to-Peer and DHT

Using peer-to-peer technologies gives to the nodes the possibility of working in a scalable and self organized way without a centralized control, inside of a system where the number of nodes and computational faults varies often [3]. The JXTA project was initially presented by Sun Microsystems and it was later adopted by a group of academic institutions. When peer-to-peer services and architectures emerged in the past they usually did not interact with each other, one of the JXTA main objectives is to offer a set of protocols to build peer-to-peer networks that allow interoperability between distinct applications. JXTA defines the minimum requirements for developing these networks, conceding to developers the possibility of choosing the precise way of building their networks. The JXTA protocols establish an overlay network over the internet, allowing peers to interact directly with each other and auto-organizing themselves independently of their original network topologies [4]. JXTA specification was created to be independent of the programming language, operating systems, services and network protocols[5]. Another objective of the protocols is to hide the complexity of the network topologies providing communication between peers, even when carried out behind the firewalls and NATs [4]. The JXTA introduced some new concepts like peer groups, relay peers, rendezvous peers and pipes. Peer groups are responsible for isolating the applications inside the peer-to-peer networks. Relay peers and rendezvous peers are super-peers and have special capabilities such as relaying messages from isolated peers and message group propagation. Pipes are virtual communication channels used for sending and receiving messages between peers. If the pipe is sending a message it is called an output pipe and if it is receiving it is called an input pipe [6]. The JXTA has a set of six protocols [7]:

- Peer EndPoint Routing Protocol;
- Rendezvous Protocol;
- Peer Resolver Protocol;
- Peer Discovery Protocol;
- Peer Information Protocol;
- Pipe Binding Protocol.

2.1 DHT

Distributed hash tables are similar to traditional hash tables in that they provide a mapping between keys and values. This was introduced in 2001 by CAN[8], Chord[9], Pastry[10] and Tapestry[11]. Distributed hash tables, known as DHT, are distinguished for presenting a partitioned table across de participating nodes. The data stored in the DHT is called value and a hash function is applied on it in order to obtain the key that indexes this value, creating the pair <*key, value*>[12]. Each node is responsible for storing and mapping a group of keys using internal routing tables. The DHT search is done by asking neighbor nodes and their neighbor nodes until the target node is found. This target node provides the value indexed by the key. The proposal presented in this paper makes use of a stable and already tested distributed hash table called OpenDHT Project [6]. OpenDHT is a public service that is currently performed in 700 nodes spread over approximately 300 sites across the world controlled by *PlanetLab* [13]. Each one of these nodes uses the Linux operating system and runs *Bamboo*[12]. *Bamboo* is an implementation of the Pastry protocols. *Pastry was first* presented in 2001, when the DHT architectures started to appear. The DHT nodes can execute inserts, query, removal and other message routing operations. Each openDHT node is also known as DHT-*gateway* because it allows the clients that are not part of the DHT to work as any other node present in the DHT. At this point, it's not been considered problems related to workload in the DHT nodes. By using the OpenDHT structure it is expected to have a proof of concept working environment.

3. Web2Peer Browser

The developed browser mixes some well known technologies and presents a new way for publishing, searching and accessing web pages on the internet. With this, every running browser becomes a node inside a peer-to-peer network providing its own web pages and replicated web pages from other peers. The implemented browser has the following main parts:

3.1 GUI

This block shows the components that interact directly with the user. The user interface was developed as a traditional java graphical interface, which was called screen loader, and an http library API, which provides the regular internet connection in order to preserve the compatibility with the traditional web pages used in client/server architectures. Working with the screen loader there is an event listener that is used for receiving all the user commands and actions. The content searcher is shown to the user in the GUI and it sends all the queries to the search mechanism that will be shown later. The html editor is the tool used for creating the web pages and only through this editor it is possible to publish web pages. All these pieces are shown in the graphical user interface and all of them interact with the engine parts.

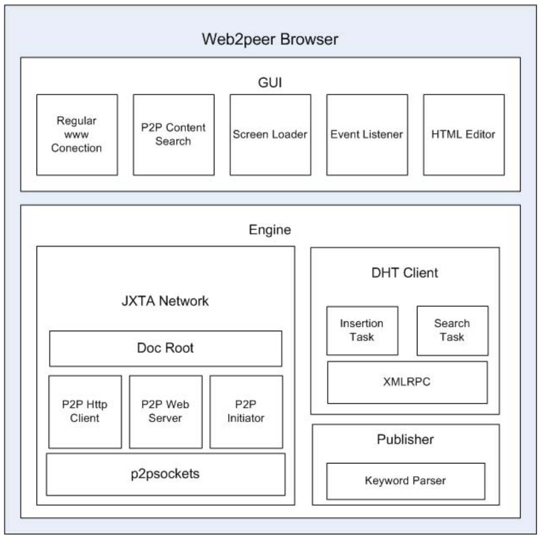


Figure. 1 Web2peer browser structure

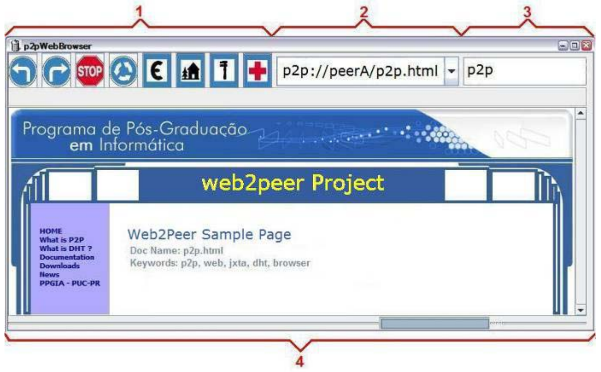
3.2 Engine

This block shows the mechanisms that provide all the capabilities of the browser. JXTA offers all the peer-to-peer functionalities including login in the JXTA network, obtaining a JXTA ID, publishing an output pipe through a peer-to-peer web server and others. JXTA has a Web Server which provides the web pages and a HTTP Client that accesses web pages from remote Web Server through the P2P network. Both are implemented using p2psockets [14] API which is a reimplementation of the java sockets to work inside a JXTA network. With this it is possible to transfer HTML files through the peer-to-peer network. *DocRoot* is an internal directory where the web pages created by the html editor and those replicated from other peers are stored. Moreover, the browser has a DHT Client to access a DHT in order to index the pages. The JXTA network has an internal protocol for searching resources, the PDP (*Peer Discovery Protocol*), but this search is done by flooding the network which reduces the response time. In order to solve this issue a DHT was adopted as an index service. Through this DHT, executing operations like insertion and searches are made possible. Another important mechanism is called publisher which is responsible for publishing web pages on DHT using the DHT client. The communication between the browser and the DHT is made by XMLRPC messages, as shown later.

3.3 Working Overview

When the browser is started, a connection to the JXTA network is immediately established using the p2psockets library. This task uses two methods (*signing* or *autosigning*) depending on the version of the p2psockets. After that, a JXTA ID is received from the JXTA network and the peer-to-peer browser will publish an input pipe advertisement. This pipe is used by the web server which will be listening for peer-to-peer connections. An open source JXTA web server called Jetty was used to provide this functionality.

Figure 2 shows the browser interface and its four main parts: 1: Control buttons, used for regular operations (backward, forward, stop, refresh, HTML editor, homepage, tools and preferences); 2: Address bar, used for either accessing regular internet addresses based on URL and URI or checking where the p2p document comes from; 3: Search bar, used for searching documents start from keywords[2]; 4: Screen area, where the pages are shown.



Web2Peer Sample Page
Doc Name: p2p.html
Keywords: p2p, web, jxta, dht, browser

Figure 2. Web2peer Browser Interface

There are three main functions performed by the Web2Peer browser: web page publishing, web page searching and web page replication. In summary, it is possible to describe the system's operation as follows:

When the system starts, it login a peer-to-peer network using JXTA, receives an ID and publishes an input pipe for receiving content's requests. The user creates its web page locally using the browser's editor and it publishes the web pages using keywords recorded as meta-tags during the web page creation. To publish the web page, the publisher mechanism creates a record in a DHT, indexing the keyword and the input

[2] Keywords are defined according to the subject of the web page during the web page creation.

pipe location (a P2P address) of the page. From this point, the web page can be found from other peers using the search mechanism provided by the DHT. Any user can use its own browser to search for any subject using keywords.

A list of documents found on the DHT using the same keywords is shown in the screen in order to allow the user to choose what page or from which source (using a P2P link) the user wants to download. Once the source is chosen the web page is downloaded using the JXTA network. The local internal HTTP client requests the remote internal web server to send the desired web page. When this web page arrives in the requesting peer, it is shown on the browser's screen, saved and delivered to the local web server. In this point, a new source for this web page is created and it needs to be indexed again to make the new source known to the others peers. To create this new entry, the publisher mechanism is invoked again and it creates a new key and value pair in the DHT using the requiring peer's input pipe and the same keywords saved in the web page's meta-tag.. Doing this, the web2peer browser increases the content availability of the web pages by replicating them through the peer-to-peer basic feature. Moreover, it provides a publishing and searching mechanism which makes the browser independent of any regulatory entity and enables anyone to publish a web page.

3.4 Web Page Publishing Process

The act of publishing a web page is completely independent. The web2peer browser offers everything needed to make a web page available on internet. The starting step is to create an html editor or to choose an open source editor whose code can be modified to implement specific functions (step 1 of the figure 3). In this work we create a very simple html editor with basic functions instead of using anyone. Figure 3 illustrates how the process of publishing was developed.

One important function added to html editor is the interaction to the publisher mechanism. After creating the web page, the HTML editor asks to the user for keywords, and saves them inside the web page in specific meta-tags in the default directory called *docroot* (step 1.1.1). The meta-tags are read again when the editor calls the publisher mechanism to publish the specified web page (step 1.3).

When the publisher is called it reads the html file and using its parser mechanism it looks inside the html file searching for meta names called keywords and reads each saved keyword, as follows:

<META NAME="Keywords" CONTENT="p2p, web">

Each found keyword will generate one publishing message among web2peer and the DHT System. The keyword extractor code, called keyword parser in web2peer, is shown in figure 4.

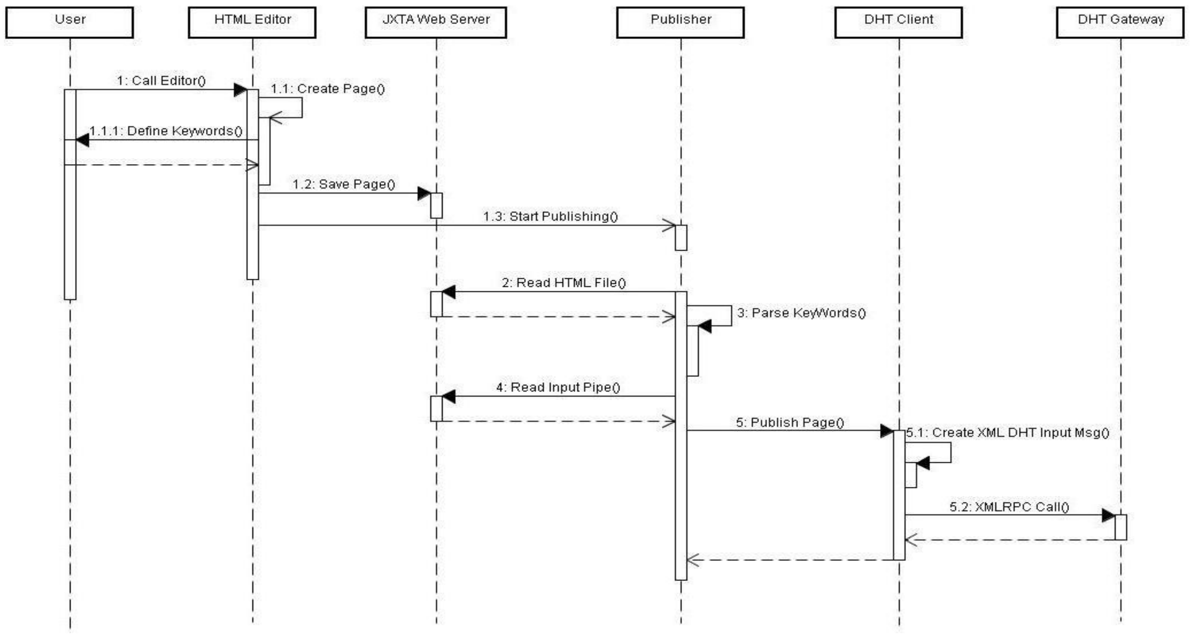
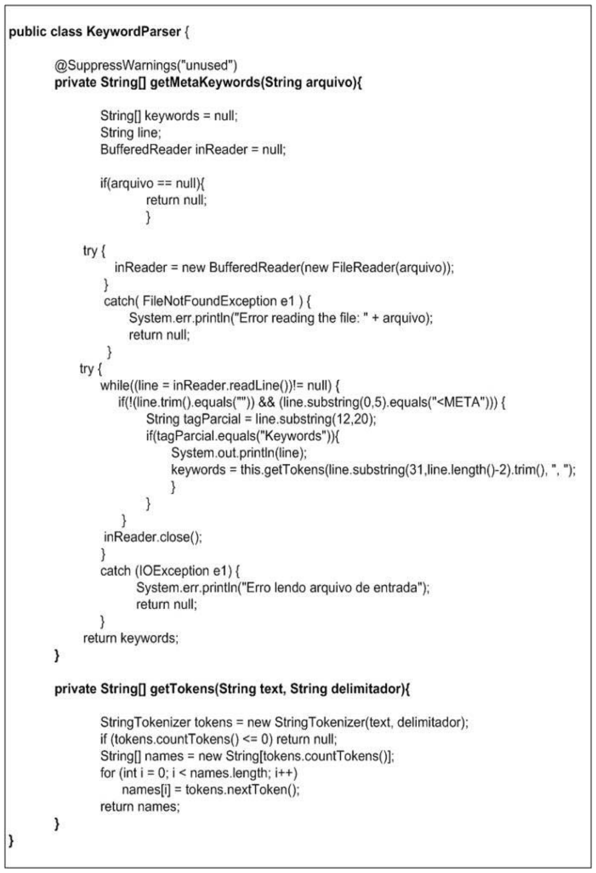


Figure 3. Web Publishing Process

Before creating the XML package to send to the DHT System, the publisher reads the input pipe name of the JXTA web server. This pipe name defines the location of the page inside the JXTA network. The keyword and the input pipe will form the pair <key, value> which will be recorded in the DHT.

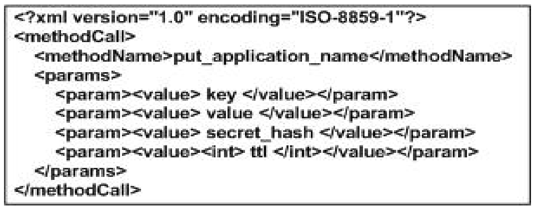
In order to index this pair in the DHT, the publisher calls the DHT Client (step 5), which builds a XML message to send to the DHT gateway (steps 5.1 and 5.2). This message is shown in figure 5 and it has four fields: key, value, secret and TTL (Time-To-Live). Key field represents the keywords (the subjects treated in the web page). Value is the location (represented by a P2P address) of the page in the peer-to-peer network. Secret is a password used to remove this entry when needed, this password is the result of a hash function applied in a user defined password. TTL is the expiration time of the entry, usually defined as the maximum amount. All these fields were defined according to the chosen DHT implementation, as mentioned before.



```
public class KeywordParser {

        @SuppressWarnings("unused")
        private String[] getMetaKeywords(String arquivo){

                String[] keywords = null;
                String line;
                BufferedReader inReader = null;

                if(arquivo == null){
                        return null;
                        }

            try {
                inReader = new BufferedReader(new FileReader(arquivo));
              }
              catch( FileNotFoundException e1 ) {
                  System.err.println("Error reading the file: " + arquivo);
                  return null;
               }
            try {
                while((line = inReader.readLine())!= null) {
                    if(!(line.trim().equals("")) && (line.substring(0,5).equals("<META"))) {
                        String tagParcial = line.substring(12,20);
                        if(tagParcial.equals("Keywords")){
                            System.out.println(line);
                            keywords = this.getTokens(line.substring(31,line.length()-2).trim(), ", ");
                            }
                        }
                    }
                inReader.close();
                }
                catch (IOException e1) {
                        System.err.println("Erro lendo arquivo de entrada");
                        return null;
                }
            return keywords;
        }

        private String[] getTokens(String text, String delimitador){

                StringTokenizer tokens = new StringTokenizer(text, delimitador);
                if (tokens.countTokens() <= 0) return null;
                String[] names = new String[tokens.countTokens()];
                for (int i = 0; i < names.length; i++)
                    names[i] = tokens.nextToken();
                return names;
        }
}
```

Figure 4. Keyword Parser

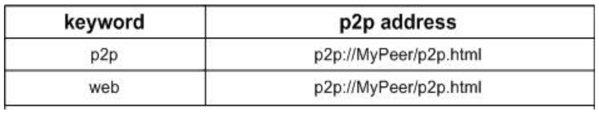
After creating the XML message, the DHT Client sends the message to the DHT Gateway which will perform the insert operation in the DHT.



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<methodCall>
   <methodName>put_application_name</methodName>
   <params>
      <param><value> key </value></param>
      <param><value> value </value></param>
      <param><value> secret_hash </value></param>
      <param><value><int> ttl </int></value></param>
   </params>
</methodCall>
```

Figure 5. XML Put Message

If the operation is successfully executed, a new entry will be added in the DHT, as shown in figure 6. In this example, a web page called p2p.html is provided for a peer that has an input pipe called MyPeer. The page has two keywords associated with it: p2p and web.



keyword	p2p address
p2p	p2p://MyPeer/p2p.html
web	p2p://MyPeer/p2p.html

Figure 6. DHT Entry

3.5 Web Page Searching Process

The searching process in Web2Peer makes it possible to find any published web page start from keywords. To search, find, download, and show a web page on the screen, two main steps are done, figure 9 has a detailed view about how it is done.

First, using the interface, as shown in figure 2 item 3, the user searches for documents related to specific keywords. The keywords inserted are sent to DHT Client which prepares the XML message to send to the DHT gateway. When the DHT gateway receives the XMPRPC call, it searches the indexes in the DHT and sends back a list of all peer-to-peer addresses with web pages associated with the same keywords. After receiving this list, the DHT Client mounts a local html page to show it to the user. Figure 9 shows the messages 1, 1.1, 1.2 and 1.3, which represent the XML message (Figure 7) being created and sent to the Gateway by DHT Client. This search service is supposed to work similarly to the way Google does nowadays. After the page is chosen, the second part of the search is initiated. It is responsible for bringing the desired web page from the source peer to the requesting peer. In the figure 9, a detailed view shows the messages exchanged, steps 2, 2.1, 2.2, 2.2.1, and their respective responses, to obtain this. The API, called p2psockets, is used to provide all the communication among Web2Peer and JXTA network. Selecting the source peer on screen, the user invokes an HTTP client that is built on top of p2psockets, as well as its web server, and can request and download the web pages.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<methodCall>
  <methodName>get_application_name</methodName>
  <params>
    <param><value> key </value></param>
    <param><value> maxvals </value></param>
    <param><value> placemark </value></param>
  </params>
</methodCall>
```

Figure 7. XML Get Message

The HTTP Client can find the source peer on the JXTA network using the name of the input pipe published by the source peer, which was put in the DHT at publishing time, as discussed before. To reach this input pipe, the HTTP client uses two JXTA protocols: PDP (*Peer Discovery Protocol*) and PBP (*Pipe Binding Protocol*).

When this input pipe is found a direct connection is established between HTTP Client and the Web Server, and the requested page is transferred and loaded in the screen of the Web2Peer browser. The query receives the required arguments (*peergroup*, *input pipe*, *page name*, *peer identification*, etc) to perform all tasks needed to reach the chosen source page. First, by using the *signin* method the peer connects to the JXTA network using the identification arguments. After that, the *p2p* protocol is registered to allow the communication with remote web servers. Following, other received arguments are put in a list called *url* which is used by two other methods. The first one, called *executeMethod*, searches for the specified input pipe and the second, called *getResposeBodyAsString*, downloads the web page code directly from the target remote peer, putting it in a local variable called *responseBody*. Finally, the content of the variable *responseBody* is saved in the *DocRoot* directory and it is presented to the user

3.6 Web Page Replication Process

One of the main features of peer-to-peer networks is the possibility for contents replication. To build the replication process in this environment, very few lines of code were added. When the remote web page arrives at a local peer, it needs to be saved in the right place, which means saving the file inside of the *docroot* directory of the Web Server.

So, the downloaded web page is also available in a new source inside the peer-to-peer network. But it still does not reflect the real state of the indexing service. In order to replicate the page in the DHT, a new invocation of the publisher is needed. Calling the publisher to create a new entry in the DHT provides the possibility for other peers to find the same web page in other sources.

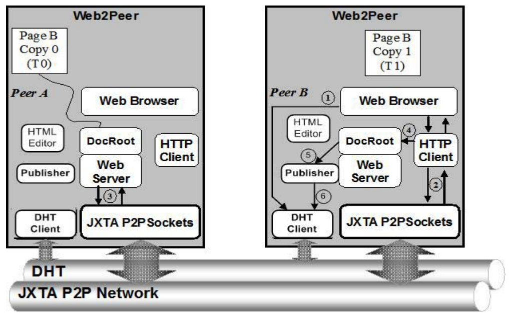


Figure 8. Web Page Replication Process

In summary, the replication process is divided in saving the page locally and calling the publisher. Figure 8 shows how a regular replication is performed. In the example, Peer B searches for specific keywords in the DHT (steps 1) and receives a list with possible sources for Page B, which is the page that contains the required keywords. This page was originally created by Peer A (Page B Copy 0).

If the user decides to download the page from Peer A, the Peer B will receive the page through the JXTA network (steps 2 and 3) and it will also have a copy of Page B (Copy 1). pAfter that, the replication will be performed by saving the page B locally, in the *DocRoot* directory (step 4). Then the publisher is called, and it will update the index at DHT (step 5 and 6). So, the replication process is done, it makes Peer B a new source peer of the page B.

4. Environment and Results

The prototype was developed using the IDE Eclipse 3.1 and the Java Virtual Machine version 1.5. The environment has two edge peers on different computers running Windows XP operating system. Moreover, a peer relay was built running on UBUNTU Linux operating system, version 6.06 with kernel 2.6. This relay peer, connected to the JXTA network, is needed to allow peers inside networks with firewalls to access the JXTA network. The tests were performed with two edge peers connected from different physical networks through the internet. One edge peer connected on a domestic ADSL connection and the second connected to the distributed systems' laboratory.

The relay peer was connected straight on the internet and it was listening on ports 80, 22, and 9701, as shown in the figure 10. A web page was created by one edge peer and keywords were associated with the page, and published. Next, the other edge peer searched by one of the used keywords and the list of the source peers is shown.

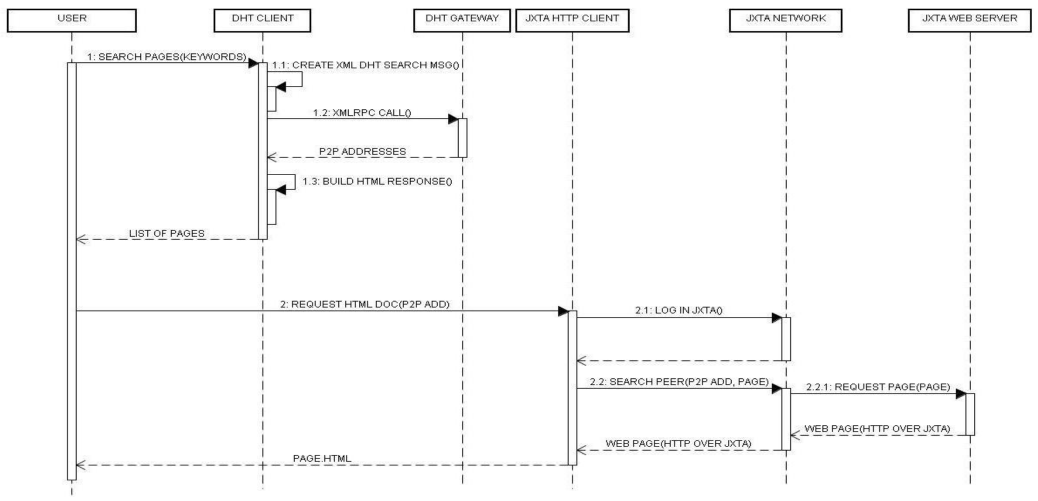


Figure 9. Searching Process

By clicking on it, the page was downloaded through the JXTA network, as expected. When the page was shown in the requesting edge peer, a new search was done and it was possible to verify that the same file was available from two different peers now, certifying that the publication, searching and replication were working as expected.

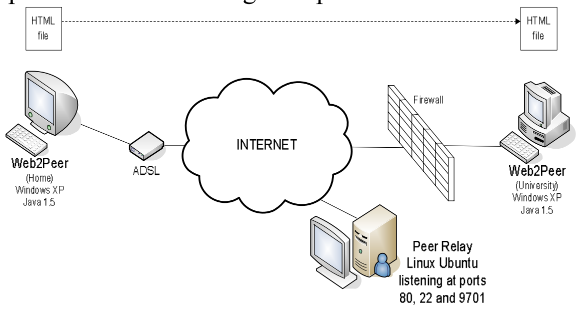


Figure 10. Working Environment

Believing in the feasibility of the practical use of this infra structure, the response times observed during the publishing and searching tests have shown values similar to the ones obtained today on search mechanisms such as Google.

To observe the time consumed by the publishing process, a set of web pages were published several times. During the practical tests, ten web pages were created and published ten times, and the time used to publish these pages did not present relevant variations. The time spent to each publication is shown in figure 11.

The searching time test was performed querying keywords in the DHT, each query presents 10 results, such as Google. The time consumed was, in numbers, inferior to Google. Even so, the obtained results were fairly good, considering that the Google architecture is different in terms of hardware and software, justifying the usability of our system. Figure 12 shows the times obtained in 10 queries with a different number of keywords.

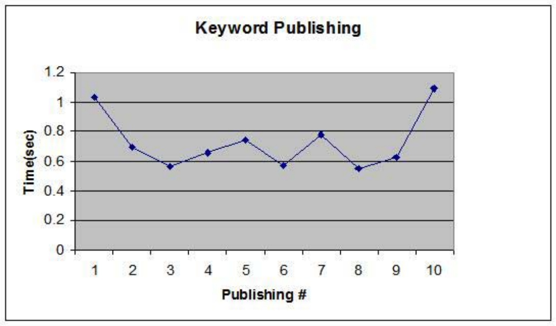


Figure 11. Publishing Time

The presented results were compared with Google just to emphasize the usability of the web2peer. Of course, it is not expected to replace the great indexing/searching mechanisms used by Google nowadays. Our intention in this comparing is to show to the future users that the times spend by using Web2Peer are similar as current popular mechanisms.

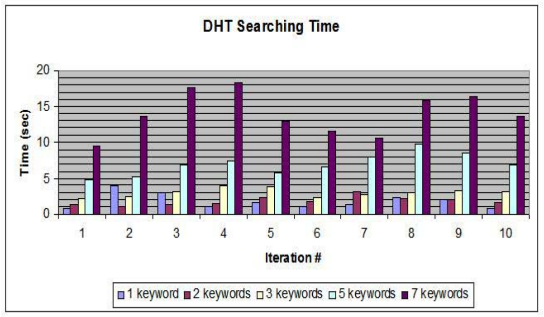


Figure 12. Searching Time

5. Conclusion and Future Work

Although some research has tried to present similar ideas, none has reached the set of functions that our project did.

In 2001, the Freeweb[15] project introduced itself as an alternative for publishing web pages on the internet preserving the anonymity and censorship. On the other hand, Freeweb relied on a poor searching mechanism based on flooding. The DHT, used in our approach, produced the faster responses of the two. Moreover, Freeweb is derived from Freenet[16] architecture which is IP address dependent, JXTA approaches are more flexibly because of its independence from platform, programming language, topologies, and architectures.

In 2003, a web browser based on JXTA architecture was presented [17]. But it was a collaborative web browser (for groupware) based on peer-to-peer networks providing the possibility of sharing the screen and controls of the web browser but it did not provide any capability of publishing web pages inside this JXTA network. The work presented in this paper has two important contributions, first improving the availability of the web pages published on the internet, second making the publishing process easier by breaking the dependency of the regulatory entities. In order to reach these objectives the JXTA protocols and a set of other existing technologies and mechanisms were put together providing a user-friendly web browser that works both on the current internet and on the new peer-to-peer internet.

There are some key points that must be improved or developed to make the solution even more interesting such as: Security mechanisms that allow a user to digitally sign web pages providing authenticity and integrity; Cryptographic mechanisms for providing confidentiality; Reputation mechanisms for scoring peers and their behavior on the network; Access control mechanisms based on public key infra-structure (PKI). A new feature is almost finished and it will be part of the project soon, it is called *Updating Page Control,* and it will provide a way for controlling updates made in the web pages, similar to *Control Version Systems*. The *Web2Peer* can be found at http://www.ppgia.pucpr.br/~lau/Web2Peer for testing.

References

[1] B. T. L. J. Li, J. Hellerstein, F. Kaashoek, "On the Feasibility of Peer-to-Peer Web Indexing and Search," presented at IPTPS, 2003.

[2] "ICANN - Internet Corporation for Assigned Names and Numbers", www.icann.org, Access: [Jan/2007]

[3] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," *ACM Computing Surveys*, 2004.

[4] L. Gong, "Project JXTA: A Technology Overview," Sun Microsystems Inc 2002.

[5] B. Traversat, A. Arora, M. Abdelaziz, M. Duigou, C. Haywood, J. C. Hugly, E. Pouyoul, and B. Yeager, "Project JXTA 2.0 Super-Peer Virtual Network," *Sun Microsystem White Paper*, 2003.

[6] S. Microsystems, "JXTA v2.3: Java Programmer's Guide," 2005.

[7] B. J. Wilson, *JXTA*. Indianapolis: New Riders Publishing, 2002.

[8] P. F. S. Ratnasamy, M. Handley, R. Karp, "A Scalable Content Addressable Network," presented at ACM SIGCOMM, 2001.

[9] R. M. I. Stoica, D. Karger, M. F. Kaashoek, "Chord: A scalable peer-to-peer lookup service for internet applications," presented at SIGCOMM, 2001.

[10] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Lecture Notes in Computer Science*, 2001.

[11] L. H. B. Y. Zhao, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz, "Tapestry: a resilient global-scale overlay for service deployment," *IEEE Journal on Selected Areas in Communications*, 2004.

[12] B. G. S. Rhea, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu, "OpenDHT: a public DHT service and its uses," presented at SIGCOMM 2005, 2005.

[13] P. L. Consortium, "PlanetLab Consortium , http://www.planet-lab.org/, Acessed: Jun, 2006," 2006.

[14] "p2psockets", www.p2psockets.org, Access: [Jan/2007]

[15] "FreeWeb Website", Freeweb, http://freeweb.sourceforge.net/ Access: [Jan/2007]

[16] O. S. I. Clarke, B. Wiley, and T. W. Hong, "Freenet: A Distributed Anonymous Information Storage and Retrieval System," presented at Workshop on Design Issues in Anonymity and Unobservability, 2000.

[17] J. M. M. Nakamura, K. Chiba, M. Shizuka, and Y. Miyoshi, "Design and implementation of a P2P shared Web browser using JXTA," presented at AINA, 2003.