Distributed Search in P2P Networks through Secure-Authenticated Content Management Systems (CMSs)

Dimitrios Tsesmetzis, Manolis Solidakis, Vassilios Stathopoulos, Nikolaos Mitrou

Telecommunications Laboratory
National Technical University of Athens
Heroon Polytechniou 9, 15773 Athens, Greece
Email: {dtsesme, esolid, vstath} @telecom.ntua.gr
mitrou@softlab.ntua.gr

Abstract

Peer to Peer (P2P) networks become more and more popular nowadays. They offer the capability of locating and obtaining a file from all over the world very fast. As a result, P2P networks constitute the main cause for the network traffic. Besides their many advantages, these networks suffer from a very strong drawback: authentication-security. Content Management Systems (CMSs) on the other hand offer this capability, among others. A combination of these two architectures could be obviously very interesting.

1. Introduction

Today there is a rapidly emerging interest in P2P networks, as they provide an efficient way for locating and obtaining a file (data, audio etc.) from all over the world very fast. Though there are some limitations as we will see later. These networks can be categorized through the "keyword-searching" capability they offer.

There are implementations like Gnutella [1] , Kazaa [2] etc. that give the user the ability to search a P2P network without knowing the exact name of the file, or even knowing only some metadata about the file. This freedom of course causes a strong limitation: they cannot guarantee a certain possibility of successfully locating the file. This limitation is caused because the searching mechanism they use is based on – more or less - flooding and they are unstructured.

On the other hand, there exist architectures like Pastry [3], Tapestry [4] etc. that provide excellent results: they locate the requested file extremely fast (4-5 hops in a network with 10.000 nodes [3]) with a certain possibility. They are highly structured and the searching mechanism they use is based on consistent-hashing [5]. Nevertheless, these architectures have a very big disadvantage: they do not support keyword – searching. This means that a user should know the exact name of the file in order to get it, something that makes them of course difficult in use.

Besides their differences, the above implementations suffer from the same drawback: they do not support authentication-security. Every user can have full access in such networks completely anonymously. There is no way to control and manage a user's identity, which makes these architectures quite unreliable. Likewise, this anonymity does not enable enterprises to benefit from P2P networks in accounting matters. However, Content Management Systems (CMSs) can remedy all these problems. Although CMSs have been evolved in order to face the need of managing millions of files in the network, they can cooperate very efficiently with P2P architectures as well. In order for a system to be a content management one, it should fulfill some requirements, like [6]:

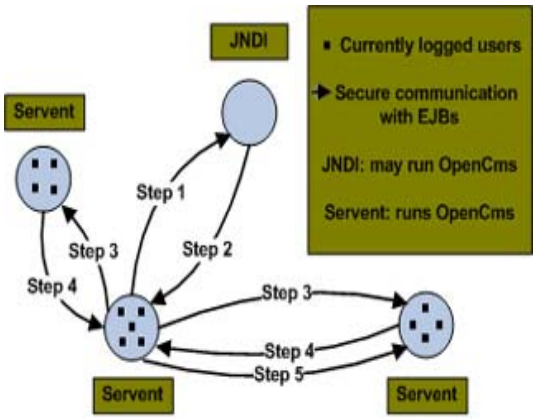
- Combine content components
- Separate content from design
- Manage workflow of content creation
- Make authentication of users

It is clear that if we could combine the benefits of CMSs –especially authentication- with P2P architectures in an efficient way, we could develop a very reliable P2P application.

2. Implementation

We have implemented an application for searching in a P2P network, where each servent (servent: client plus server) is a CMS. The CMS that we used is OpenCms version 5.01 [7] and is open source. It is worth mentioning, that although the application relies on this CMS, it is not tied with it. Likewise, although OpenCms offers the ability of searching locally for a resource, we preferred to build our own local search

mechanism, in order to support keyword – searching and searching with the *metadata* of the resource, something that is not supported from the current version of OpenCms. We will approach our implementation in five steps using the following diagram:



Step one: A user that has logged in to OpenCms, wants to search for a resource in the P2P-OpenCms network. In order to accomplish this, he should know his neighbors. Therefore, in the first step, the user registers his OpenCms-servent to the JNDI (Java naming and directory interface) node, which is responsible for the registration and the network shaping. Of course, after a successful registration, no other user of this servent will repeat this process. Additionally the communication between the servents and the JNDI node is achieved with the technology of EJBs (Enterprise Java Beans) and this system is completely fault-tolerant, as the JNDI node keeps permanently the status of the network structure in a distinct database. Thus, our proposal constitutes a robust solution in the sense that the system is considered as self recovered. Furthermore, the exclusive use of EJBs for the communication between the servents and the secure mechanisms they support (such as SSL protocol, access control etc.), make the system secure, robust and reliable.

Step two: The JNDI node, after a successful registration of the remote servent, replies with the list of neighbors, with which this servent will communicate.

Step three: The user sends his requests - queries to his neighbours. The search algorithm we used is based on flooding, as the expected number of the CMSs is considered to be not very large. This is true, as hundreds of users may register in every CMS. We should also mention that the search mechanism supports various ways for querying a resource, based on the resource name, title, description etc. and all of them support keyword-searching. These requests contain the identity of the user to the local CMS, which will be used in order to validate and authenticate the user to the remote CMS. This identity is correlated with the group that a user belongs to. In case this group is not supported at the remote CMSs, then user is classified as a member of the default one. In that way, the user will be able to query only for those resources that he is authorized to access.

Step four: The remote servents authenticate the remote user, search their local database and return any matching results, while concurrently forward the requests to their neighbours.

Step five: The user has found the resource that he was looking for and decides to transfer it to his local servent.

3. Conclusion-Future work

The extensive use of peer-to peer applications requires that there be specific mechanisms of certification and security in the system. In this paper, we present a way to accomplish these, using a CMS. We intend to create a more sophisticated authentication mechanism, in order to ensure the full identity of each user and servent. Furthermore, we plan to build a more efficient search algorithm, taking advantage of the network topology. To conclude, a testbed is under construction for the performance analysis of our application according to the standard benchmark.

4. References

[1] Gnutella. http://gnutella.wego.com

[2] Kazaa. http://www.kazaa.com

[3] A. Rowstron and P. Drushel. Pastry: Scalable, distributed, object location and routing for large-scale peer-to-peer systems, in *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)* (Nov 2001), pp. 329-350.

[4] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Tech. Rep. UCB/CSD-01-1141, Computer Science Division, U. C. Berkeley, Apr. 2001.

[5] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, R. Panigrahy. Consistent Hashing and random trees: Distributed caching protocols for relieving hot spots in the World Wide Web, in *Proceedings of the 29th Annual ACM Symposium on Theory of Computing* (El Paso, TX, May 1997), pp. 654-663.

[6] GartnerConsulting: The Emergence of Distributed Content Management and Peer-to-Peer Content Networks, *White Paper*, January 2001.

[7] OpenCms. http://www.opencms.org