International Conference on Complex, Intelligent and Software Intensive Systems

Evaluation of Clustering techniques for Efficient Searching in JXTA-based P2P systems

Fatos Xhafa
Polytechnic University of Catalonia
Department of Languages and Informatics Systems
C/Jordi Girona 1-3, 08034 Barcelona, Spain
Email: fatos@lsi.upc.edu

Enric Jaen Villoldo
Open University of Catalonia
Department of Information Sciences
Barcelona, Spain.
Email: ejaenv@uoc.edu

Thanasis Daradoumis
Open University of Catalonia
Department of Information Sciences
Barcelona, Spain.
Email: adaradoumis@uoc.edu

Leonard Barolli
Dept. of Information and Communication Engineering
Fukuoka Institute of Technology (FIT)
3-30-1 Wajiro-higashi Higashi-ku, Fukuoka 811-0295, Japan
Email: barolli@fit.ac.jp

***Abstract*—The efficient file searching is an essential feature in P2P systems. While many current approaches use brute force techniques to search files by meta information (file names, extensions or user-provided tags), the interest is in implementing techniques that allow content-based search in P2P systems. Recently, clustering techniques have been used for searching text documents to increase the efficiency of document discovery and retrieval. Integrating such techniques into P2P systems is important to enhance searching in P2P file sharing systems. While some effort has been done for content-based searching for text documents in P2P systems, there has been few research work for applying these techniques for multimedia content in P2P systems. In this paper we introduce two P2P content-based clustering techniques for multimedia documents. These techniques are an adaptation of the existing Class-based Semantic Search (CSS) algorithm for text documents. The proposed algorithms have been integrated into a JXTA-based Overlay P2P platform, and some initial evaluation results are provided. The JXTA-Overlay together with the considered clustering techniques is thus very useful for developing P2P multimedia applications requiring efficient searching of multimedia contents in peer nodes.**

Keywords: P2P networks, P2P multimedia clustering, Class-based Semantic Search, JXTA, content-based retrieval.

I. INTRODUCTION

Clustering in P2P systems is a key research area aimed to produce two clear benefits: On one hand increases the efficiency of the processes for document discovery and retrieval, and on the other hand increases the quality of the documents obtained. A lot of research has been conducted during the last years for searching and document retrieval from the WEB. Due to the fast development in P2P technologies [2] and their use in file sharing systems, researchers are paying attention to the implementation and evaluation of searching techniques [12], [14] other than brute force search.

Clustering techniques are developed in Data Mining domain to discover "structure" in the data, which yields to combining data into groups (clusters) according to some criteria of similarity; items in a cluster are similar to each other and items in different clusters differ among them according to the considered similarity criterion (the reader is referred to the book by Tan, Steinbach and Kumar [10]). Clustering techniques have attracted the attention of researchers from the P2P community due to their usefulness in developing efficient searching in P2P file sharing systems. Presently, three types of clustering techniques have been studied for P2P networks; locality-awareness clustering [4], clustering by file type [3] and clustering by content of textual files [11], [13], [5]. However, no studies has been carried out for clustering of multimedia files by content in P2P networks. There is, however, a large amount of work done in the area of content-based retrieval[1] (CBR) systems [6]. The purpose of this paper is to show how multimedia CBR systems and the existing P2P clustering techniques can be integrated to produce quality P2P networks, in terms of performance and quality of content-based retrieval.

To carry out this study, we have adapted the class-based semantic search system (CSS) for textual documents, to be able to search and retrieve any kind of multimedia files. More precisely, in this paper, we are concerned with image retrieval in P2P networks. We have adapted the CSS in two ways; a centralized version of the CSS, which has been integrated into our JXTA-Overlay platform[2], a JXTA-based P2P semi-structured platform, and a version of the centralized CSS that builds meta-classes. The objective is to evaluate both the centralized and meta-class versions and to evaluate their performance in a P2P network.

The paper is structured as follows: Section II gives a classification of clustering techniques for P2P systems. Section III introduces the main aspects of our JXTA-Overlay platform. Section IV describes the proposed CSS adaptations and their integration into JXTA-Overlay. The experimental study and some preliminary results are shown in Section V. Finally, Section VI concludes the paper and indicates directions for future work.

[1]http://en.wikipedia.org/wiki/CBIR
[2]https://jxta-overlay.dev.java.net

II. Selection of Clustering Techniques

There are two kind of clustering categories that can be applied in P2P systems: Intra-node clustering, i.e. those techniques aimed to classify files inside a node. This category can be further divided into two sub-categories: tag-based clustering and content-based clustering. The former classifies files accordingly to some meta information such as file names, extensions, or user-provided tags, and the second one is based on the content of the files themselves. In this paper, we will focus on this second kind of clustering. On the other hand, inter-node clustering aims to perform clustering taking into account the information provided by multiple nodes. The next subsections explain these techniques in more detail.

A. Intra-node clustering

Content-Based Retrieval systems are queried with an example multimedia file and they retrieve similar files. These systems are classified accordingly the kind of multimedia file, so there are CBIRs, (Content-Based Image Retrieval) systems for images, MIRs (Music Information Retrieval) systems for music, and CBVR systems for videos.

B. Inter-node clustering: CSS

Class-based semantic search (CSS) is a clustering technique that creates classes of semantically similar documents in the P2P nodes, and that inter-connect virtually pairs of these classes accordingly to soft links (in case they are similar) or long links (in case the are not similar). CSS uses well-known vector-based algorithms for creating the intra-node classes, and defines two inter-node algorithms; one for adapting the topology, and another for doing the search of documents in two phases: a first phase of directed walk through long links, intended to locate the next node where to search for short-linked classes, and a second phase of flooding through soft links. In this phase all the short-linked classes are visited and their relevant files are returned. CSS is decentralized so that are the nodes themselves which maintain the links with their neighbors. To keep the network topology updated, clients periodically send prove queries that provide up-to-date information about the neighbor classes.

A key concept in CSS is the class vector of a given class, which is a centroid value representing the content of the whole class. This class vector is used to calculate the overall relevance score between two classes, which is a measure that determines the similarity between the two classes, and is used to determine the distance of the of virtual link.

III. Overview of JXTA-Overlay Platform

The JXTA-Overlay is a P2P platform for building mixed peer-to-peer applications, built on top of JXTA [1], [7], [8] The ultimate objective of the Overlay P2P network is to make Clients interact with other Clients by using four types of peers defined in JXTA: (a) minimal peer; (b) full peer; (c) rendezvous peer; and, (d) relay peer. The Overlay deploys a network of full peers (called Clients) and rendezvous peers (called Brokers). Clients interact with other Clients and they may or not provide a user interface. Brokers help Clients to discover other Clients and to collect information (such as files). JXTA-Overlay builds networks of peers and brokers. Peers (known as Full-featured Edges in JXTA) are endpoint applications virtually connected with other peers. Brokers (Rendezvous peers in JXTA) help peers to locate other peers, route their messages, and implement those parts of the middleware that require a whole knowledge of the Overlay network.

We show in Fig. 1 the architecture of the overlay network.

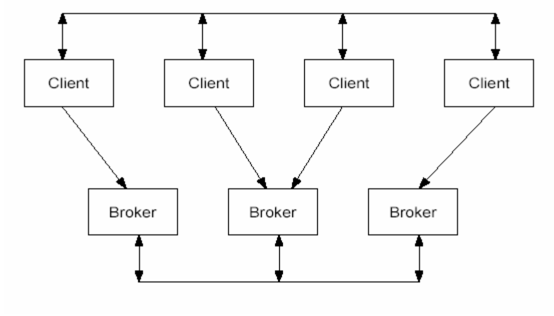

Fig. 1. The overlay network.

The Overlay layer offers a middleware interface to the client applications, consisting on a series of primitives and events classified in five functional areas, namely: discovery, messenger, file sharing, which can be used to develop distributed applications (remote execution of jobs to peer nodes), groupware tools, etc. The Control layer stands between the Overlay and the JXTA layer, and it's responsible to maintain JXTA peer groups, and offer high-level messaging functionality, separating therefore the Overlay from the JXTA details. The Broker layer implements the part of the primitives that require a whole knowledge of the network. Finally, there are two kind of Overlay clients: GUIClients that offer a GUI to the final user, and SimpleClients that perform application-specific computation in the background (composing a kind of grid).

The internal architecture of JXTA-Overlay follows a layered architecture, as shown in Fig. 2.

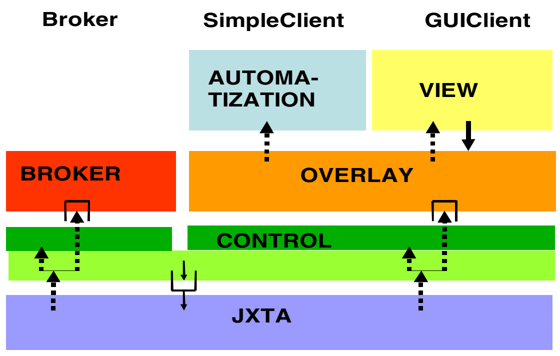

Fig. 2. Jxta-Overlay layers and peer types.

A. Control Layer

The control layer interacts with the JXTA layer, and is divided into two parts: a lower part with functionality common to any kind of peer, and a higher part with functionality specific to Brokers and Clients. The common part provides functionality for doing JXTA messaging, discovery and advertisement. On the other hand, the Broker specific part provides functionality for managing groups of Brokers and keeping broker statistics while the Client specific part provides functionality for managing groups of Clients, keeping client statistics, managing its shareable files, managing the user configuration and creating the connection with a Broker. Because this layer is the responsible to manage the JXTA peerGroups, this layer has been packaged under the namespace overlay.groups. Thus, as can be seen from Fig. 2, the lower part enqueues the JXTA messages to be sent. On the other hand, whenever a message arrives, the JXTA layer fires an event to the lower layer, which in turn fires a notifications to both the upper layer and the business layer. The starting point to instantiate the Control layer is the GroupManager class.

B. Business Layer

This layer implements the basic functionality of the Overlay. The layer is specific for Brokers and Clients, therefore it is divided into two packages, namely overlay.broker and overlay.client. At the Client side, this layer defines the Overlay API in the form of primitives and events (named fires), which establishes the middleware for future applications. Currently the API is divided into several areas for discovery (network information area), messenger (instant messaging area), files (file transfer area), execution (remote execution of task area) and Groupware tools area. The relationship between functions, primitives, tasks and fires are depicted in Fig. 3.

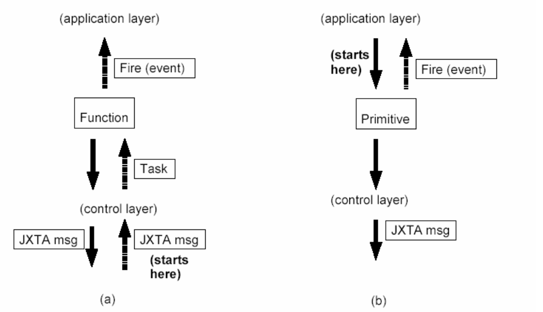

Fig. 3. Jxta-Overlay: (a) Receiving JXTA messages and (b) executing primitives.

IV. INTEGRATION OF THE CSS WITH JXTA-OVERLAY

A. Centralized CSS

Due to the mixed architecture nature of the Overlay platform, the CSS has been adapted so that Brokers maintain the CSS topology of the P2P network, instead of being distributed as in CSS. Clustering in the client peers is achieved via a CBIR system which is executed each time a user shares a file (an image in this case). Users can tune the cluster of files by giving a value to a relevance index, which is a threshold that will consider as similar those files with the relevance score above the cited index. For example, a relevance index of 80% will classify together images with a relevance score above 80%. Periodically, clients send to the Broker information about the classes, in particular, they send the centroid value of each class, which is not a class vector of text documents as in CSS, but is an image representing the whole class. Currently this image is just one of the images of the class, but in the future it could be an aggregation of content information obtained from the class files.

As in CSS, the centroid value is used to calculate the relevance score between two classes. The Broker, each time that receives an update-node message, recalculates the node neighborhood and the virtual links as defined in CSS. During this process the Broker calculates the overall relevance scores between the neighbor classes as it is defined in CSS. This is a costly process, as it requires to compare the centroid value with each neighbor class, which as explained before is an image in our case. The cost of this process will therefore depend on the number of neighbor classes and the size of the images. During a search operation, the client sends a query to its Broker, which processes the directed walk mode, meanwhile the flooding mode is carried out in the nodes containing the soft-linked classes. At the end of this search, the Broker returns to the client peer the list of relevant file paths and their node location. The client can therefore access those nodes and download the desired files.

B. Meta CSS

We have designed a new approach of the CSS that takes advantage of the centralized information of the CSS topology available in the Broker. This new approach avoids calculating class links (which is a costly process) but instead creates meta classes. Meta classes are classes containing similar classes. The similitude between two classes is measured through their relevance score. As with file clustering, the user can specify a relevance index to cluster the classes. At the client side there isn't difference with respect the centralized CSS in the sense of class creation and information sent to brokers. The Broker, however, classifies the received classes accordingly their relevance score with the other classes, just in the same way as it is done with the files clustered on the clients. The advantage of this approach is that the classes are automatically classified so that there isn't need to create topological links, and therefore much less relevance calculations are needed. Another advantage is that this approach returns a large number of occurrences with respect CSS. Section V compares experimentally the two approaches.

C. UML models

This section describes partly the UML models representing both techniques. We show in Fig. 4 the UML diagram of the centralized CSS model.

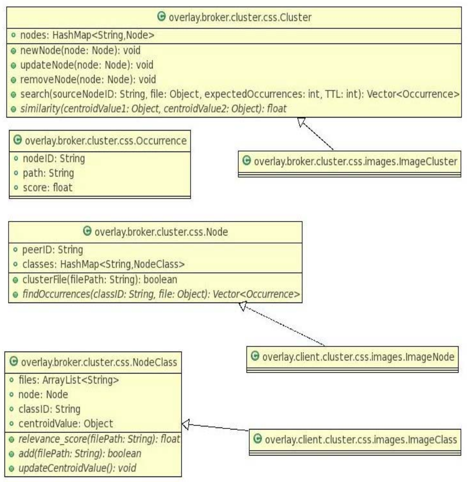
overlay.broker.cluster.css.Cluster
nodes: HashMap<String,Node>
newNode(node: Node): void
updateNode(node: Node): void
removeNode(node: Node): void
search(sourceNodeID: String, file: Object, expectedOccurrences: int, TTL: int): Vector<Occurrence>
similarity(centroidValue1: Object, centroidValue2: Object): float
overlay.broker.cluster.css.Occurrence
nodeID: String
path: String
score: float
overlay.broker.cluster.css.images.ImageCluster
overlay.broker.cluster.css.Node
peerID: String
classes: HashMap<String,NodeClass>
clusterFile(filePath: String): boolean
findOccurrences(classID: String, file: Object): Vector<Occurrence>
overlay.client.cluster.css.images.ImageNode
overlay.broker.cluster.css.NodeClass
files: ArrayList<String>
node: Node
classID: String
centroidValue: Object
relevance_score(filePath: String): float
add(filePath: String): boolean
updateCentroidValue(): void
overlay.client.cluster.css.images.ImageClass

Fig. 4. UML model for the centralized CSS.

The Cluster class, which resides in the Broker, defines methods invoked by the clients to either upgrade the topology (addNode, updateNode, removeNode) or to perform searches. Clients send a Node class serialized to the Broker, containing a set of NodeClasses and the JXTA peerID. It has a clusterFile method to allow the client classify a file, and a findOccurrences callback method which is invoked by the Broker to find Ocurrences in that class. Notice that this is an abstract class. The ImageNode class implements the findOccurrences method for images. A NodeClass contains a list of file paths, the Node which belongs to, a classID and an abstract centroidValue, which in the case of the ImageClass is a buffered image. The ImageClass class implements the methods to calculate the relevance score for images, a method to add an new image to the class, and a method to update the centroidValue. The search method receives the nodeID of the client, the template file to search, the number of expected occurrences and a time to live. Notice that the similarity method is abstract, and it is implemented by the ImageClass class which finds the similarity between two images. As a result, the search operation returns a series of Occurrences, indicating the location (nodeID), the file path inside the node, and score of the occurrence.

Fig. 5 depicts the classes that define the Meta CSS. Notice that the Cluster class shares the same interface as the centralized CSS, but adds a new set of MetaClasses.This abstract class is specialized for images via the ImageCluster. The MetaClass defines a set of NodeClasses, which are described above. This abstract class is specialized for images via the ImageMetaClass.

V. EXPERIMENTAL EVALUATION

We have compared experimentally the two centralized CSS approaches described before. In this section we present a preliminary evaluation.

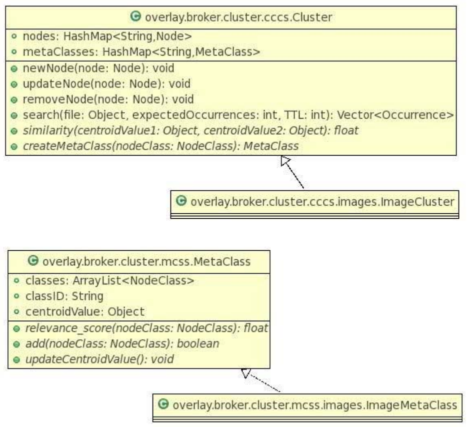
overlay.broker.cluster.cccs.Cluster
nodes: HashMap<String,Node>
metaClasses: HashMap<String,MetaClass>
newNode(node: Node): void
updateNode(node: Node): void
removeNode(node: Node): void
search(file: Object, expectedOccurrences: int, TTL: int): Vector<Occurrence>
similarity(centroidValue1: Object, centroidValue2: Object): float
createMetaClass(nodeClass: NodeClass): MetaClass
overlay.broker.cluster.cccs.images.ImageCluster
overlay.broker.cluster.mcss.MetaClass
classes: ArrayList<NodeClass>
classID: String
centroidValue: Object
relevance_score(nodeClass: NodeClass): float
add(nodeClass: NodeClass): boolean
updateCentroidValue(): void
overlay.broker.cluster.mcss.images.ImageMetaClass

Fig. 5. UML model for the Meta CSS.

A. Experimental setting

The tests have been carried out with 4 and 8 nodes. Each node shares 9 JPG images of about 50 Kb and 300 Kb size.

We have reproduced a test scenario where a node joins the network, then the Broker recalculates the topology and finally the user does a search query. For this case the upgrade is not done periodically as the upgrade happens just once. This scenario has been repeated with different file sets in order to produce different amounts of classes. Both the relevance index that determine the file clusters and the index that determine the class clusters have been set to 50%.

The evaluation measures the following parameters:

a) Upgrade time: During topology update it is measured the upgrade time required and the total number of relevance scores (i.e. similarities).

b) Broker's time for search query: Another measure is the search time employed by the Broker to perform the user query. The query is submitted to its Broker, which processes the query and returns to the client peer the list of relevant file paths and their node location. The following output is an example, where we can see five occurrences found at four nodes:

```
Occurrence: nodeID=urn:jxta:uuid-...6A032 path=/.../img01.JPG score=0.9441511,
Occurrence: nodeID=urn:jxta:uuid-...6A032 path=/.../img02.JPG score=0.5895395,
Occurrence: nodeID=urn:jxta:uuid-...6A033 path=/.../img03.JPG score=0.54662615
Occurrence: nodeID=urn:jxta:uuid-...6A030 path=/.../img03.JPG score=0.54662615,
Occurrence: nodeID=urn:jxta:uuid-...6A031 path=/.../img03.JPG score=0.54662615
```

c) Number of relevant results: finally, it has also been measured the quality of the results in terms of number of relevant occurrences found.

To carry out the evaluation we have run a Java Overlay Broker in a Linux 8-core server. Each core consists of a 3GHz CPU, and the total amount of memory is 4GB. The client nodes have been emulated by creating files containing the serialized classes that would be sent by the client nodes to the Broker in a real scenario.

To cluster the files and calculate the relevance score between classes we have reused the Lire Java package[3], which is a free source CBIR system. This package keeps the clustered files in a database index and provides a method to search similar documents that returns a list of relevance files and their relevance score.

Calculating a single relevance score is costly, as it needs to create a new database index with the two images to be compared and, then to do the comparison. It has been measured this time and it takes about 3 seconds to compare two 50Kb images, and about 4 seconds to compare 300Kb images.

B. Computational results

a) Relevance scores and upgrade time.: We show in Fig. 6 the graphical representation of the number of relevance scores calculated during the topology upgrade for the four different test cases.

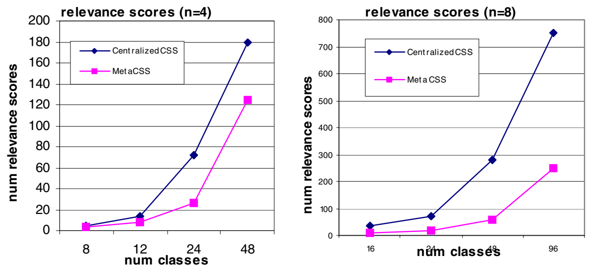

Fig. 6. Number of relevance scores for 4 and 8 nodes.

As can be seen from the figure, the number of score calculations increases exponentially with the number of classes, and increases also exponentially with the number of nodes. This may lead to scalability problems. The Meta CSS approach in any case requires less calculation that the Centralized CSS. Taken into account this, Fig. 7 shows the time taken by the Broker to upgrade the topology when a node joins the network, or when updates its classes.

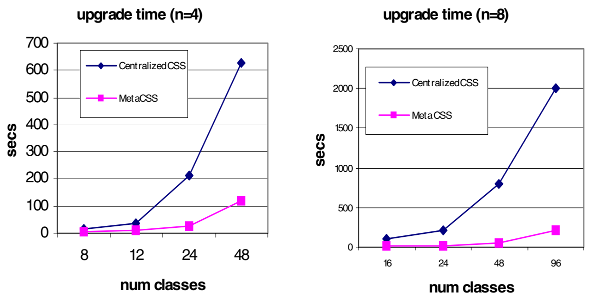

Fig. 7. Upgrade time for a network with 4 and 8 nodes.

It is clearly seen that the Meta CSS scales much better than Centralized CSS. For example, when the cluster topology has 48 classes and there are 4 nodes, the Centralized CSS takes about 625 seconds (approximately 10 minutes) to upgrade a node, meanwhile it takes 125 seconds (approximately 2 minutes) in Meta CSS, yielding thus an important performance improvement. These times increase with the number of joined nodes, so for example, with 8 nodes the upgrade time for 48 classes raises to 33 minutes and 3 minutes respectively. Although considerable, this time is transparent to the users as it takes place in the Broker (in the background).

[3]Lucene Image REtrieval library at http://www.semanticmetadata.net/lire/

b) Query search time.: The next measure analyzed is the query search time. We show the graphical representation of the results in Fig. 8. Surprisingly, the search times are almost the same in both techniques. The search time increases with the number of classes, and it must be reminded that it involves access to the client nodes in order to find the relevant files of those similar classes previously selected by the Broker. The search time, however, is not related to the number of nodes, as Fig. 8 shows. As an example, it takes about 37 seconds to get a response when the network has 4 nodes and there are 48 classes, but it takes about 17 seconds when there are 8 nodes.

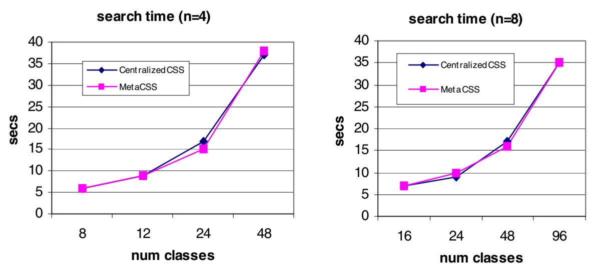

Fig. 8. Search time for a network with 4 and 8 nodes.

c) Number of image occurrences.: An interesting measure is the number of image occurrences that the user receives. As can be seen from Fig. 10, Meta CSS returns always the 100% of the possible similar images, meanwhile in Centralized CSS this percentage is slower.

Finally, we show in Fig. 9 the upgrade time, the search time and number of relevance scores for different size images.

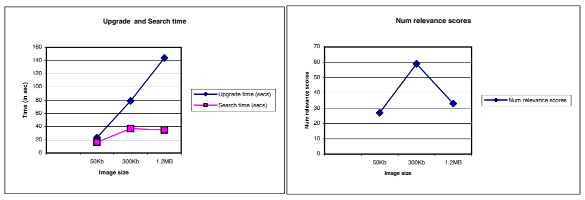

Fig. 9. Upgrade and search time (left) and number of relevance scores (right) for different size images.

C. Evaluation

As can bee seen from the experimental results both approaches have a similar response time to the user. However, Meta CSS returns more occurrences and scales better in the Broker, meanwhile the centralized CSS behavior doesn't scale.

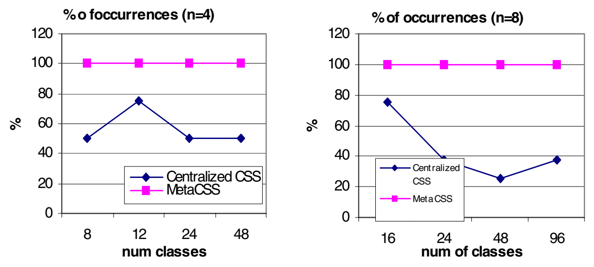

Fig. 10. Percentage of occurrences found for 4 and 8 nodes.

VI. Conclusions and Future Work

In this paper we have presented and analyzed two approaches for clustering by content multimedia files in P2P networks. These techniques take advantage of the previous work done in P2P clustering using classes and content-based retrieval systems. To this end, we have integrated content-based semantic search systems (CSS), namely centralized CSS and Meta CSS, into our JXTA-Overlay P2P platform and have evaluated the performance of the proposed techniques. The obtained results suggest that it should be used Meta CSS rather than the centralized CSS as the latter does not scales well in a centralized environment due to the cost of creating virtual links.

In our future work we will perform a thorough evaluation of the proposed approach by deploying our JXTA-Overlay in a large P2P network. Also, we plan to use the implemented techniques integrated in our JXTA-Overlay to develop groupware tools that need efficient searching of multimedia files in P2P networks.

Acknowledgements

This work has been partially supported by the Spanish MCYT project TSI2005-08225-C07-05.

References

[1] D. Brookshier, D. Govoni, N. Krishnan, and J. Soto. *JXTA: Java P2P Programming*.

[2] J. Crowcroft, T. Moreton, I. Pratt, and A. Twigg. Peer-to-Peer Technologies. In Foster and Kesselman, eds, *The Grid: Blueprint for a New Computing Infrastructure*, chapter 29, 593–622. Morgan Kaufmann, 2003.

[3] K.P. Gummadi, R.J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *SOSP'03*.

[4] F.L. Fessant, S. Handurukande, A.M. Kermarrec, and L. Massoulie. Clustering in peer-to-peer file sharing workloads. In proc. of IPTPS 2004.

[5] J. Huang, X. Li and J. Wu. A Class-Based Search System in Unstructured P2P Networks. In Proceedings of the 21st international Conference on Advanced Networking and Applications (May 21 - 23, 2007). AINA. IEEE Computer Society, Washington, DC, 76-83. 2007

[6] M.S. Lew, N. Sebe, C. Djeraba and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. ACM Trans. Multimedia Comput. Commun. Appl. 2, 1 (Feb. 2006), 1-19.

[7] S. Li. *Early Adopter JXTA*. Wrox Press, 2003.

[8] S. Oaks, B. Traversat, and L. Gong. *JXTA in a Nutshell*. O'Reilly, 2003.

[9] H.T. Shen, Y. Shu, and B. Yu. Efficient semantic-based content search in p2p network. IEEE Transactions on Knowledge and Data Engineering, 16(7):213-236, 2004.

[10] P.N. Tan, M. Steinbach, V. Kumar. *Introduction to Data Mining*. Addison-Wesley 2005

[11] H.F. Witschel. Content-oriented topology restructuring for search in P2P networks. Technical report, University of Leipzig, Germany, 2005.

[12] B. Yang and H. Garcia-Molina. Improving search in peer-to-peer networks. In Proceedings of the 22nd IEEE International Conference on Distributed Computing (IEEE ICDCS'02), 2002.

[13] Y. Zhou, W. B. Croft, and B. N. Levine. Content-based search in peerto-peer networks. Technical report, University of Massachusetts, 2004.

[14] Y. Zhu, X. Yang, and Y. Hu. Making search efficient on gnutella-like p2p systems. In Proceedings of the 19th IEEE International Parallel & Distributed Processing Symposium (IPDPS'2005), Denver, Colorado, Apr 2005.