Communications and Network, **2013, 5, 529-543**
http://dx.doi.org/10.4236/cn.2013.53B2097 Published Online September 2013 (http://www.scirp.org/journal/cn)

A P2P Platform for Collaborative Aggregated Multimedia Sharing

Ines Fakhfakh, Hongguang Zhang, Marc Girod-Genet
UMR CNRS 5157, Telecom SudParis, Institut Mines-Telecom, Evry, France
Email: ines.fakhfakh@ it-sudparis.eu

Received June 2013

ABSTRACT

Peer-to-peer technologies have emerged as a powerful and scalable communication model for large scale content sharing. However, they are not yet provided with optimized heterogeneous aggregated content management functionality since they lack rich semantic specifications. To overcome these shortcomings, we elaborated a reference model of P2P architecture for a dynamic aggregation, sharing and retrieval of heterogeneous multimedia contents (simple or aggregated). This architecture was mainly developed under the CAM4Home European research project and is fully based on the CAM4Home semantic metadata model. This semantic model relies on RDF (Resource Description Framework) and is rich (but simple enough), extensible and dedicated for the description of any kind of multimedia content. In this paper, we detail and evaluate an original semantic-based community network architecture for heterogeneous multimedia content sharing and retrieval. Within the presented architecture, multimedia contents are managed according to their associated CAM4Home semantic metadata through a structured P2P topology. This topology relies on a semantically enhanced DHT (Distributed Hash Table) and is also provided with an additional indexing system for offering semantic storage and search facilities and overcoming the problem of exact match keywords in DHTs.

Keywords: P2P; DHT; RDF; Semantics; Metadata; Semantic Indexing and Search; Multimedia Content Sharing; Community Networking

1. Introduction

Peer to peer (P2P) systems provide scalable distributed mechanisms for data sharing and retrieval. There are two kinds of P2P topologies: unstructured and structured overlays [1,2].

Unstructured overlays, e.g. Gnutella [4], organize peers in a random graph and use flooding to propagate peer's queries. As both data and index items remain locally at peer level, each query is only evaluated against the local index. Unstructured networks are therefore not suitable for reducing lookup costs (the number of involved peers) by exploiting data correlations between data items shared in a P2P system. Moreover, flooding usually causes communication redundancy and network congestion. In addition, there is no central coordinator which has knowledge of all the shared data. Data is therefore not reorganized in the network and no global structure is maintained [2-7]. So, it's hard to provide semantic queries in an unstructured network.

Structured overlays, like kademlia [8], carry out a globally consistent protocol to determine where the distributed resources should be stored by the provider and cope with scalability and locality issues [1,2,8,9]. Those protocols rely on hashed identifiers to manipulate shared documents in an effective way. If structured overlays were developed to improve performances of data delivery, they are unfortunately more expensive to maintain (e.g. in terms of time and query number) and are not supporting efficient complex queries [2,3,10-12].

Another kind of P2P protocol is hybrid overlays where some functionality are still centralized, thus attempting to combine the flexibility of unstructured protocols with the lookup performance guarantees of structured ones [2,13-15].

There is unfortunately not yet a fully semantic-oriented P2P community network for wide area multimedia content sharing, even if some research woks have already been conducted in the P2P overlay world [16-20]. The only existing generic semantic metadata model for describing multimedia contents comes from the CAM4Home project [21,22]. However, this model was never combined with existing P2P systems for improving large scale multimedia content sharing and retrieval. Consequently peers, in existing systems, do not have a global knowledge about shared multimedia contents similarities and relationships.

Furthermore, peer's local knowledge about shared contents can actually be expressed in various formats such as, for example: Web pages (unstructured), text documents (unstructured), XML (semi-structured), RDF and OWL (structured), etc. This obviously depends upon the providers that generally handle the content in a proprietary way, which introduces heterogeneity. Therefore, in the context of efficient content aggregation and sharing collective knowledge composition, data knowledge must be expressed in a machine manageable/inferable format (e.g. RDF or OWL) and also in a generic and extensible way for heterogeneity management purposes. Otherwise, all the data that are not expressed in this generic format will have to be processed and converted in that generic format (metadata extraction). Let us note that metadata about data are mainly expressed in RDF format which is a W3C standard [22].

In our work, we focus on the interactive sharing and delivery of multimedia content in P2P community networks (a group). We investigate structured P2P networks using uniform hashing, often called Distributed Hash Tables (DHT). To address the flexibility and efficiency shortcomings, we combine their P2P mechanisms with semantic systems. We aim at proposing an enhanced semantic-based P2P architecture relying on generic structured descriptions for content heterogeneity management. The retained descriptions are logically the RDF-based CAM4Home semantic Metadata since they have been specified for describing any kind of multimedia contents (simple and aggregated) in a generic way [22].

To interconnect users in our P2P community, we rely on the commonly used structured P2P system called Pastry [23]. A Pastry system is a self-organizing overlay network of nodes which performs application-level routing and object location in a potentially very large overlay network of nodes connected via the Internet. It is a fully decentralized, fault-resilient, reliable and scalable building block currently used for in particular the design of large scale P2P file sharing and group communication systems [24,25]. On top of Pastry, we carry out a scalable application-level multicast overlay infrastructure called Scribe [25]. Scribe is a topic-based publish/subscribe (event based) system where any peer can create a topic and the other peers can then subscribe to the topic (*i.e.* register their interest in the topic). Any Scribe peer can therefore publish events and Scribe efficiently disseminates these events (application level multi-point delivery) to all the topic's subscribers [25].

On top of the two aforementioned systems (Pastry and Scribe), we finally introduce a new layer that carries out the following mechanisms for enhancing the DHT semantic functionalities: an original metadata indexing algorithm and an original metadata semantic searching mechanism (handling incomplete information searching). This layer not only extends the DHT with semantic search but also allows the overcoming of the problem of exact match keywords.

The paper is organized as follow. Section 2 will describe the proposed community network architecture. Then, Section 0 will detail the original semantic indexing and search algorithms added to this community. Section 4 will summarize the implementation of the proposed P2P community network. This community networks was integrated within the CAM4Home ITEA2 project pan-European test-bed. Finally, Section 0 will present some evaluation results of the proposed P2P community.

2. Overview of the P2P Community Network Architecture

2.1. General Architecture

The proposed P2P community network has been design as a subsystem of the CAM4Home network architecture ([21]) that relies on peer-to-peer networking technology. The subsystem consists of connected digital home environments, being comprised of user terminals (such as mobile phones, communicating terminals, tablets and PCs) and dedicated community network servers. The purpose of the proposed community network is to provide a service enabler for aggregated multimedia content management to distribute and retrieve content in a peer-to-peer network. This community network provides the community and user management with additional features such as eventing, social grouping and communication between the users. These functions or basic services are referred to as the P2P community network functions.

The proposed community network has two main components: the "Community Network P2P Modules and Functions" and the "Community Network Server". The Community Network P2P modules offer a set of functionalities to create peer groups for sharing and retrieving data in P2P overlays. The Community Network Server, also called Community Gateway, can be viewed as an extended "Community Network P2P Module" that allows non P2P clients and end users to access P2P community network functions. This Community Gateway is provided with a GUI and extended with a Community Network Service (CNS) module. The CNS module is designed in a SOA compliant way and provides non Community Network enabled components and processes with a basic set of interacting and interfacing functions with Community Networks. The CNS also implements eventing and provides publish/subscribe functionalities for handling the dynamics of shared multimedia content within communities.

Figure 1 summarizes the high level architecture of the Community Network. The "Community Network P2P Modules and Functions" depicted in this figure is composed of networking and application level routing functions relying on DHTs and on publish/subscribe paradigms. It implements functions for initiating and integrating a peer in a P2P community, as well as for publishing, distributing and sharing contents within a community. Pastry and Scribe have been selected to provide content and metadata storage and retrieval [24,26]. Pastry DHT, named PAST, is used as the Community Network DHT. Scribe is used, on top of Pastry, to build the groups forming the P2P communities.

The community Gateway depicted in **Figure 1** provides a generic community interaction and management interface, as well as content sharing and delivery to/from the P2P communities. Its main building blocks are the following:

- Community Network (CN) P2P Modules and Functions.
- Community Network Service.
- Community Gateway GUI: user-friendly graphical interface for end users of P2P communities, *i.e*. the community members.
- Search Aggregation Client: client of the Search Aggregation service that is designed and implemented, as a Web Service, for allowing the search of multimedia content over the Internet through a natural language and keyword-based query system. The search result is a description of a content that matches the search attributes, given in the CAM4Home Metadata model description format [22]. The Search Aggregation Client is also introduced for populating Community Networks with external content. For example, when a search operation performed by a community member fails, the Search Aggregation service could be automatically invoked and any external content retrieved can be stored in the community and provided to members.
- "Metadata Management Client": introduced for allowing the Community Network Server to validate Content Metadata descriptions that are published in our Community Networks. These Content descriptions need to be validated prior to their publication within communities in order to verify their conformance to the CAM4home Metadata model.

2.2. P2P Content sharing and Delivery Architecture

2.2.1. Context and Related Work

There are several previous works on P2P architectures and RDF data indexing and retrieval [26], some of them are presented below:

- Felber *et al* [27] have proposed an architecture to access a file *f* using less specific queries. Every file *f* is described by an XML document and characterized by a most specific query *q*. The file *f* is stored in the node responsible for the key $k = \mathrm{h}(q)$ where $h()$ is the hashing function. The users can access the file by generating queries close to *q*. Unfortunately this approach limits the search to a small number of appropriate queries.
- Shen *et al*. [28] have adopted a super-node based P2P architecture and developed the "Building Hierarchical Summaries" algorithm for representing the semantic of a file by document's summaries called Ddoc. All Ddoc are held in a global group and peer index so that the query is efficiently transmitted to the most relevant peer. Unfortunately, this idea is only suitable to Super peer architecture.
- Zhu *et al*. [29] have proposed an approach to conduct efficient semantic search on DHT overlays. Their basic idea was to place indexes of semantically close files into the same peer nodes with high probability

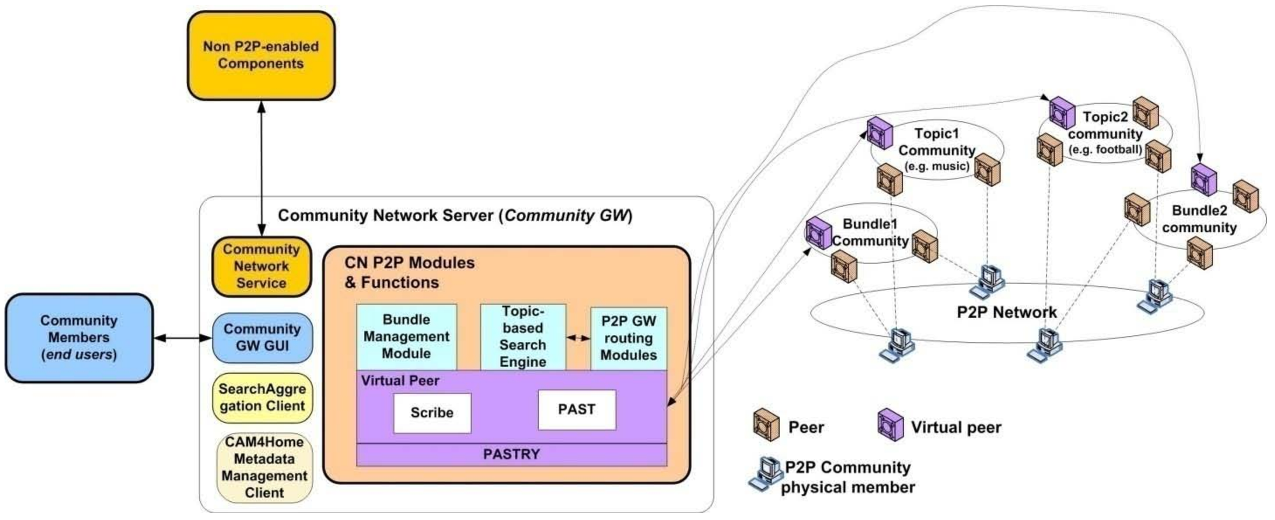

Non P2P-enabled Components
Community Network Server (Community GW)
Community Network Service
Community GW GUI
SearchAggregation Client
CAM4Home Metadata Management Client
Community Members (end users)
CN P2P Modules & Functions
Bundle Management Module
Topic-based Search Engine
P2P GW routing Modules
Virtual Peer
Scribe
PAST
PASTRY
Topic1 Community (e.g. music)
Topic2 community (e.g. football)
Bundle1 Community
Bundle2 community
P2P Network
Peer
Virtual peer
P2P Community physical member

Figure 1. Community Network high level architecture.

by adding major components on top of a DHT: extractor registry and semantic indexing and locating utility. Each file is identified by a semantic identifier "semID". However, this approach produces identical semID for similar files and queries.

- Liarou *et al* [30] rely on evaluating conjunctive Triple Pattern Queries over Large Structured Overlay Networks. They proposed the Query Chain algorithm (QC). QC main characteristic is that the query is evaluated by a chain of nodes. Intermediate results flow through the nodes of this chain and finally the last node in the chain delivers the result back to the node that submitted the query.

The distributed architecture that we have proposed and implemented within CAM4Home project (described in 0) aims at overcoming the weaknesses of the aforementioned approaches.

2.2.2. Proposed Architecture

We propose a distributed architecture based on semantic metadata for indexing multimedia contents. Our architecture relies on structured P2P networks and allows semantic queries on metadata. The structured P2P networks, usually referred as Distributed Hash Tables (DHTs), offers a lookup service similar to a hash table by providing a mapping between a pair (key, value). The goal of the proposed approach is to provide a fully distributed system that exploits the scalability and efficiency of DHTs in order to index and retrieve multimedia contents through their semantic metadata. It is worth mentioning that this scheme does not aim to build a distributed metadata repository but to index the metadata by the RDF triples that are distributed in the DHT-based storage for querying or searching. In terms of implementation, we use a structured P2P system based on Pastry [23]. Pastry provides self-organization, scalability and fault-tolerance. On top of Pastry we carry out an application-level multicast infrastructure relying on Scribe [25]. Scribe is a topic-based publish/subscribe system. Any Scribe peer can create a topic and other peers can subscribe to this new topic (*i.e*. register their interest in the topic). Any Scribe peer can publish events. These events are disseminated to all the topic's subscribers using Scribe dedicated functionalities. Scribe uses the DHT of Pastry to manage topic creation and subscription. For each topic, Scribe builds a multicast tree over Pastry to disseminate the events published in the topic. Each peer can act as an event publisher, a root of a multicast tree, a subscriber to a topic, a peer within a multicast tree, or any combination of these roles [25]. This is summarized in **Figure 2**. The Pastry DHT, named PAST, is used to distribute and share the multimedia content among the peers of the communities.

3. Metadata Indexing and Searching Within the P2P Community Network Architecture

In this paper, we propose two approaches for indexing and retrieve RDF metadata in a DHT. They are described below.

3.1. Metadata Indexing and Searching: First Approach

3.1.1. Metadata Indexing

In our first approach, we have chosen to index metadata using configuration file. In this file, we precise the relevant attributes that the system will use as keywords to hash in the DHT. For each RDF metadata file, we:

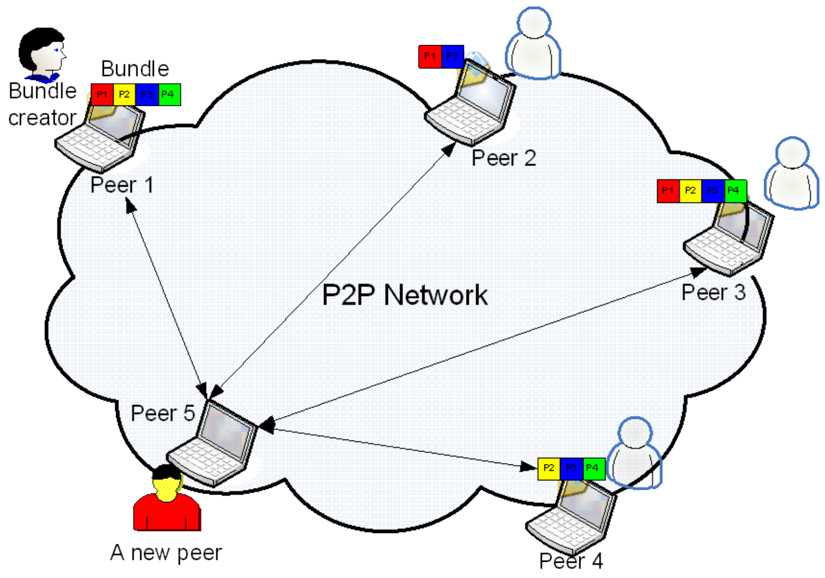


Figure 2. Scribe/pastry architecture overview.

- parse the file,
- pick out the keywords to use in the DHT storage, according to the configuration file,
- store the relevant information about the owner peer and the metadata file so it can be accessed from any other peer in the overlay network.

This indexing algorithm is mapped in **Table 1**.

Let $f(a, n)$ be the cost of running the algorithm, where a is the number of attributes in the configuration file, and n is the number of nodes in the DHT.

We have:

$$f(a, n) = \sum_{k=0}^{a}(1+\log(n))$$

$$f(a, n) = a + a.\log(n)$$

$$f(a, n) = a.(1 + \log(n))$$

$$f(a, n) = O(a.\log(n))$$

Therefore, the Execution Time of our first algorithm is in $O(a.\log(n))$.

3.1.2. Semantic Searching

Since a description file is stored according to the hash values of some keywords, the user is able to retrieve data by querying the desired ones. Search operation uses the DHT network to lookup metadata and to access to related data.

In order to search content and retrieve its RDF file metadata, the system:

- Searches metadata available in DHT that is related to user's selected keywords,
- Finds the physical location of the metadata,
- Retrieves the physical location of the Metadata and the data (image, video, text),
- And finally displays it on the remote Community Network Client Interface.

These operations are depicted in **Figure 3** which summarizes all the search mechanism steps. They are the following:

- Condition: Peer *A* publishes his content (a link to a CAM Element). This content will be routed to the responsible peer whose Id is closest to the file's identifier (Peer *B*).
- Another peer (Peer *C*) performs a search operation,

Table 1. Algorithm of Indexing Metadata using configuration files.

Precondition:	Peer disposing of configuration files
1.	for all attributes in the configuration file
2.	{
3.	*id* = ***hash***(O);
4.	***put***(*id*, *Information*);
5.	}

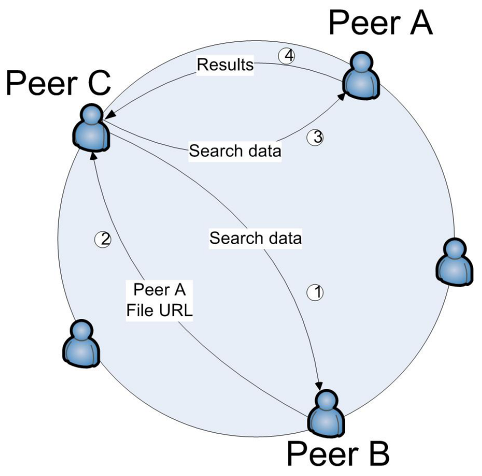


Figure 3. Steps of Search Metadata (first scenario).

- The peer responsible for the file Id (Peer *B*) returns the Metadata file URL,
- Peer *C* access to the Metadata file stored in Peer *A* using the file Id delivered by Peer *B*.,
- The Metadata file contains all the information related to a specific resource. So, Peer *C* can extract

The physical location from the Metadata file. The data is now reachable by Peer *C*.

In a traditional DHT, peers reference the multimedia content by its name. So, only users knowing exactly what they are looking for are likely to access it. In that context, our first investigation was to index the multimedia content according to some relevant attributes of the RDF metadata file. By the way, users are more likely to discover the RDF metadata description of the multimedia content which becomes easily reachable. However, the remote access to the metadata files increases the response time within the community Network. For this reason, we propose a second approach which consists of storing the whole RDF description in the DHT. In this way, we not only enhance the response time, but also the semantic aspects. In fact, since the entire RDF description file is stored in the DHT, the user can retrieve it from any of its attributes. The following section will present in detail this second approach.

3.2. Metadata Indexing and Searching: Second Scenario

3.2.1. Metadata Indexing and DHT Storage

The proposed metadata indexing scheme is composed of two steps. The first step carries out an RDF triple-based iterative indexing algorithm for multimedia addressing and discovery, thus providing a mapping for multimedia metadata spread among the distributed peers. This step is characterized by a hierarchical scheme in which the metadata are organized through RDF triple. The second step carries out an RDF triple-based storing scheme for metadata querying and searching.

An RDF document can be decomposed into a hierarchy set of atomic RDF statements. However, there is a discrepancy between metadata representation and storage model in a DHT-based infrastructure. The metadata is serialized as a hierarchical structured RDF/XML, while the data storage is codified in a flat pair of (key, value). To fill up this gap, we have proposed a RDF triple-based hierarchical indexing scheme. By using hierarchical triples, the CAM metadata can then be addressed iteratively and recursively. Let us take a CAM Object as an example. This object is decomposed into the triple tree T00; T10; T11; T12; T13; T14; T20; T30, as shown in **Figure 4**.

In order to index metadata, we iteratively put into the DHT overlay a set of (*key*, *value*) pair in the form (id_{mn}, id_{ij}). Here, id_{ij} is generated by hashing the triple T_{ij}, while id_{mn} is done by hashing each descendant of T_{ij}. The metadata is identified by id_{CAM}, calculated by the hash function on the metadata itself. The entity id_{CAM} is associated with T_{00} by (id_{00}, id_{CAM}). In order to support efficient queries on distributed RDF triples, we exploit an overlay structure to build a distributed index for these triples. Given the generic RDF triple T_{ij} with the form T_{ij} = (S_{ij}, P_{ij}, O_{ij}), it is registered into DHT three times by hashing S_{ij}, P_{ij} and O_{ij} respectively. In that way, triple T_{ij} can be queried through S_{ij}, P_{ij} or O_{ij}. Since the value of attribute "subject" and "predicate" must be a URI which is a string, the hashing function of DHT is directly used to map the values of subject and predicate to the m-bit identifier space. For "object", its type can be URI and literal. In case of a String type, the same hashing function is applied, while the locality preserving hashing used in [31] is applied to numeric types. The whole indexing algorithm is detailed in **Table 2**.

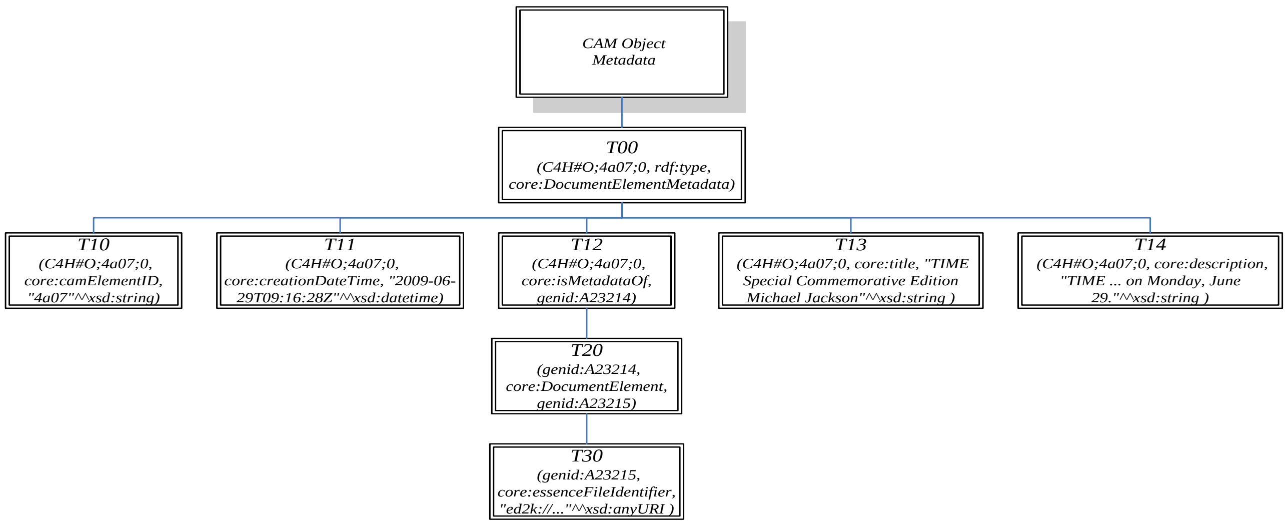

CAM Object Metadata
T00 (C4H#O;4a07;0, rdf:type, core:DocumentElementMetadata)
T10 (C4H#O;4a07;0, core:camElementID, "4a07"^^xsd:string)
T11 (C4H#O;4a07;0, core:creationDateTime, "2009-06-29T09:16:28Z"^^xsd:datetime)
T12 (C4H#O;4a07;0, core:isMetadataOf, genid:A23214)
T13 (C4H#O;4a07;0, core:title, "TIME Special Commemorative Edition Michael Jackson"^^xsd:string)
T14 (C4H#O;4a07;0, core:description, "TIME ... on Monday, June 29."^^xsd:string)
T20 (genid:A23214, core:DocumentElement, genid:A23215)
T30 (genid:A23215, core:essenceFileIdentifier, "ed2k://..."^^xsd:anyURI)

Figure 4. RDF triple-based tree Structure of CAM object.

Table 2. Algorithm of Indexing metadata by RDF Triples.

Precondition: RDF Metadata file has been decomposed into the triple tree like in **Figure 1**.

```
1: for all triples in the tree
2: {
3:     $id_{ij}$ = hash($T_{ij}$);
4:     do
5:         put($id_{mn}$, $id_{ij}$);
6:     while $T_{ij}$ has a descendant triple $T_{mn}$
7: }
8: $id_{CAM}$ = hash(CAM); /* Hash the metadata itself */
9: put($id_{00}$, $id_{CAM}$);
10: put( $id_{CAM}$, CAM);
11: for all triples in the tree /* Deposit triples by Subject, Predicate and Object */
12: {
13:     $id_{ij}(S)$ = hash(S); put($id_{ij}(S)$, $T_{ij}$);
14:     $id_{ij}(P)$ = hash(P); put($id_{ij}(P)$, $T_{ij}$);
15:     $id_{ij}(O)$ = hash(O); put(idij(O), Tij);
16: }
```

The decomposition of metadata into triples is very fast. In fact, the execution time of this algorithm is logarithmic. Let $f(n)$ be this execution time, where n is the number of nodes in the DHT.

Let m be the number of triples in the RDF file.

We have:

$$f(n) = m.\log(n) + \log(n) + \log(n) + 3.m.\log(n)$$

$$f(n) = \log(n).(m + 1 + 1 + 3.m) = 4.m.\log(n)$$

Since $m << n$, we have finally the following execution time:

$$f(n) = O(\log(n))$$

3.2.2. Semantic Searching

Metadata search consists in locating CAM Element given one or more of their attributes, while metadata query corresponds to the extraction of some required information from metadata. We have focused on an approach that extends P2P community system with searching capabilities by exploiting both the triple-based indexing algorithm and the knowledge-based RDF query languages. Based on semantic query languages like SPARQL and the triple-based indexing scheme, searching multimedia in the proposed system can be easily achieved as follows: since RDF triples are distributed on P2P DHT overlay, we resolve RDF queries by the way as presented in [32]. As querying results, RDF triples are formed to retrieve CAM Bundle (or CAM Object) on which they have been indexed. **Table 1** presents the pseudo-codes of metadata searching algorithm.

Let f(n, m, r) be the complexity of the a Search algorithm, where n is the number of nodes in the DHT, m is the number of triples in the RDF tree and r is the number of results. The complexity of this algorithm is calculated as follows:

$$f(n, m, r) = \sum_{k=1}^{r}(1+\log(n) + \sum_{l=1}^{m}(1+\log(n))$$

$$f(n, m, r) = \sum_{k=1}^{r}(1+\log(n) + m(1+\log(n))$$

$$f(n, m, r) = \sum_{k=1}^{r}((1+m)(1+\log(n)))$$

$$f(n, m, r) = r.(1 + m)(1 + \log(n))$$

$$f(n, m, r) = O(r.m.\log(n))$$

If we suppose that $r<m$ then we finally have:

$$f(n, m, r) = f(n, m) = O(m \log(n))$$

Common RDF query patterns are described in terms of relationships. For instance, if the users wants to know everything about a particular resource, it is S-Query in the form of (S, P?, O?). Together with similar P-Query and O-Query, they are called atomic queries. Complex queries can be resolved based on atomic query. For example, PO-Query (S?, P, O) is resolved by P-Query and O-Query to filter the intersection triples. In the above algorithms, RDF triples indexed by Subject, Predicate and Object are stored three times in three peers. Through this way, the knowledge in semantic metadata can be distributed over DHTs, which makes any query be resolved. However, it also results in “hot spot” where a specific peer responsible for a popular index holds too many triples. What’s more, the routing cost for resolving complex query is multiple to that of atomic query. Thirdly, we are not aiming to use the RDF triple-based scheme to distribute metadata but to provide metadata search capabilities based on RDF query. Consequently, information in the index scheme should contain only the relevant subset of data in order to create an efficient index structure. Essentially, the searching algorithm presented previously is for querying on leaf triples in O-Query or PO-Query, and then recursively for getting the root triple in order to retrieve the entire metadata. Therefore, we optimize to store the leaf triples without descendant by hashing O and PO. **Figure 5** summarizes the new steps of the search mechanism.

Table 3. Metadata Searching Algorithm.

```
Algorithm 2: CAM Bundle or Object Searching Algorithm
Precondition:
a) Metadata are indexed using Algorithm depicted in Table 2;
b) Search metadata with the social tag value “Mark Knopfler”
1: RDF Query
2:     SELECT ?SocialTag WHERE {?SocialTag core:tagValue
   “Mark Knopfler”^^xsd:string}
3: for all resulting triples
4: {
5:     key = hash(SocialTag core:tagValue “Mark Knopfler”^^xsd:string);
6:     Triple = get(key);
7:     do
8:     {
9:         index = hash(Triple);
10:        Triple = get(index);
11:    } while Triple has the parent
12:    CAM = get(Triple);
13: }
```

Condition: Peer *A* publishes his content (the RDF metadata file of the CAM Element). The content will be routed to the responsible peer who Id is closest to the file’s identifier (Peer *B*).

- Another peer (Peer *C*) performs a search operation.
- Peer *C* performs a search operation by Subject *S*, Predicate P, Object O, (S, P), (P, O), (S, O) or (S, P, O),
- The peer responsible for the file Id (Peer B) returns the metadata content,

The metadata file contains all information related to a

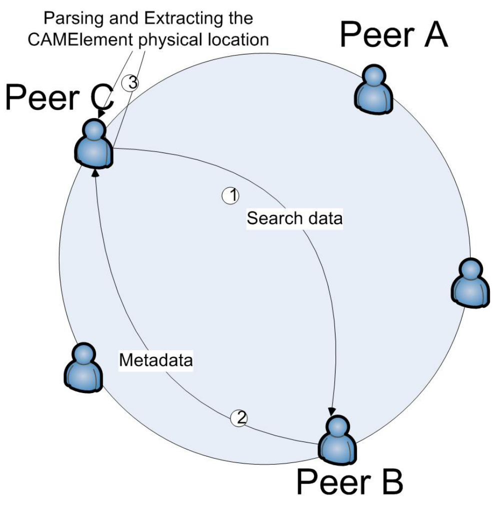


Figure 5. Steps of Search Metadata (second scenario).

specific resource. So, the peer *C* can extract the physical location from the metadata file. The data is now reachable by the peer *C*.

3.3. Incomplete Information Semantic Search

Peer-to-peer Distributed Hash Table (DHT) systems greatly improve the scalability and exact-match accuracy of P2P systems. Those systems make it simple to discover specific data when their complete identifiers-or keys-are known in advance.

In practice, however, users looking up resources don't know the exact match identifiers stored in P2P system. Since our proposed approaches of RDF descriptions indexation and search are based on DHT storage, this DHT shortcoming is still a problem. Indeed, although the metadata file can be localized based on one of its attributes, user still need to introduce the correct sentence. The goal of this approach is to provide a distribute system which can make a mapping between user's incomplete information and keywords to resolve the problem of the search by exact keywords. This approach introduces modifications in the indexation and in the search of data. Those modifications are presented below.

3.3.1. Indexation

In our proposed indexation approach, we include a mechanism to do the spelling which is based on distributed indexes build when storing the metadata in the DHT. In fact we include two additional indexes:

- The first index is indexing the relevant keywords stored in the DHT. Those keywords correspond to the object of each triple (S, P, O) in the RDF file description. This index contains the exact match keywords that a user is able to need for his search.
- The second index is introduced because the exact match keywords could be single words or whole sentences. However, to do the spelling, we need to verify if each word is well spelled. So, we create a second index regrouping the exact match keywords split into words.

3.3.2. Search

When performing a search operation, a user could misspell data or provide only partial information. In that case, we have included a mechanism to correct and complete user's keywords.

Before searching in the DHT, we make sure that we are using a keyword already stored. In fact, we perform the following steps:

1) The system corrects the user' keywords, if they are misspelled, and finds the corresponding exact match keywords stored in the DHT. In fact, we make sure that the user have spelled correctly a keyword or a part of keyword stored in the DHT (we will use the first index). The system returns all the possible words closest to the user's proposition.

2) Find the exact match keywords stored in the DHT (using the second index). Find the whole keyword that contains a part of user's proposition.

3) Search in the DHT and find identifiers of the CAMElements stored in the system.

This approach is incorporated in the search operation to improve the flexibility of our system. We have introduced a spell checking step into the search algorithm (according to the indexes inserted into the indexation stage) for finding the exact keyword from partial information, *i.e.* for finding the RDF metadata description from incomplete or incorrect information.

As the second approach is semantically richer than the first one, we will execute the semantic searching using our SPO Algorithm with additional step to suit the partial or misspelled words to the correct one as illustrated in **Figure 6**.

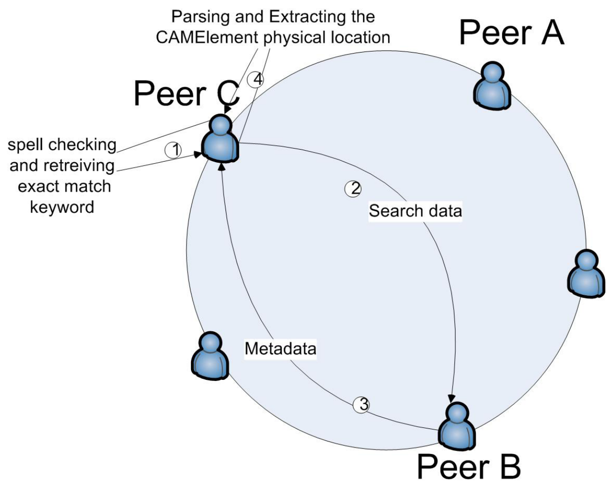


Figure 6. Steps of Search Metadata (second scenario).

4. Implementation

The implemented P2P Community Network is depicted in **Figure 7**, including the Community Network GUI. The main components of this architecture are the Community Network Peer, the Community Network Service and the Community Network GUI. The community Network Server is a specific Community Network Peer allowing non P2P-enabled entities to interact with P2P communities through the Community network architecture. It acts as a community Gateway and implements additional functionalities for serving non P2P clients. These components are fully specified in [33] and are therefore only summarized below, mainly in terms of implemented functionalities. All the aforementioned mechanisms of indexing and searching (see paragraph 3) have been implemented and tested in real conditions on the large scale demonstrator of the CAM4Home project [21]. The P2P Community Network implementation also relies on AJAX and AJAX-Push/Comet framework [34] and ICEfaces Framework [36].

The Community Network Server, also called P2P Gateway, enables a non-P2P Client to interact with P2P Community networks for creating or joining communities, publishing and discovering bundles (*i.e*. aggregated multimedia content) in the P2P network. The developed Server takes into account the heterogeneity of multimedia content (audio, video, text) and includes a publish/ subscribe model that notifies users about bundle elements' changes. The functions provided by the Community Network Peer and the community Network Server are:

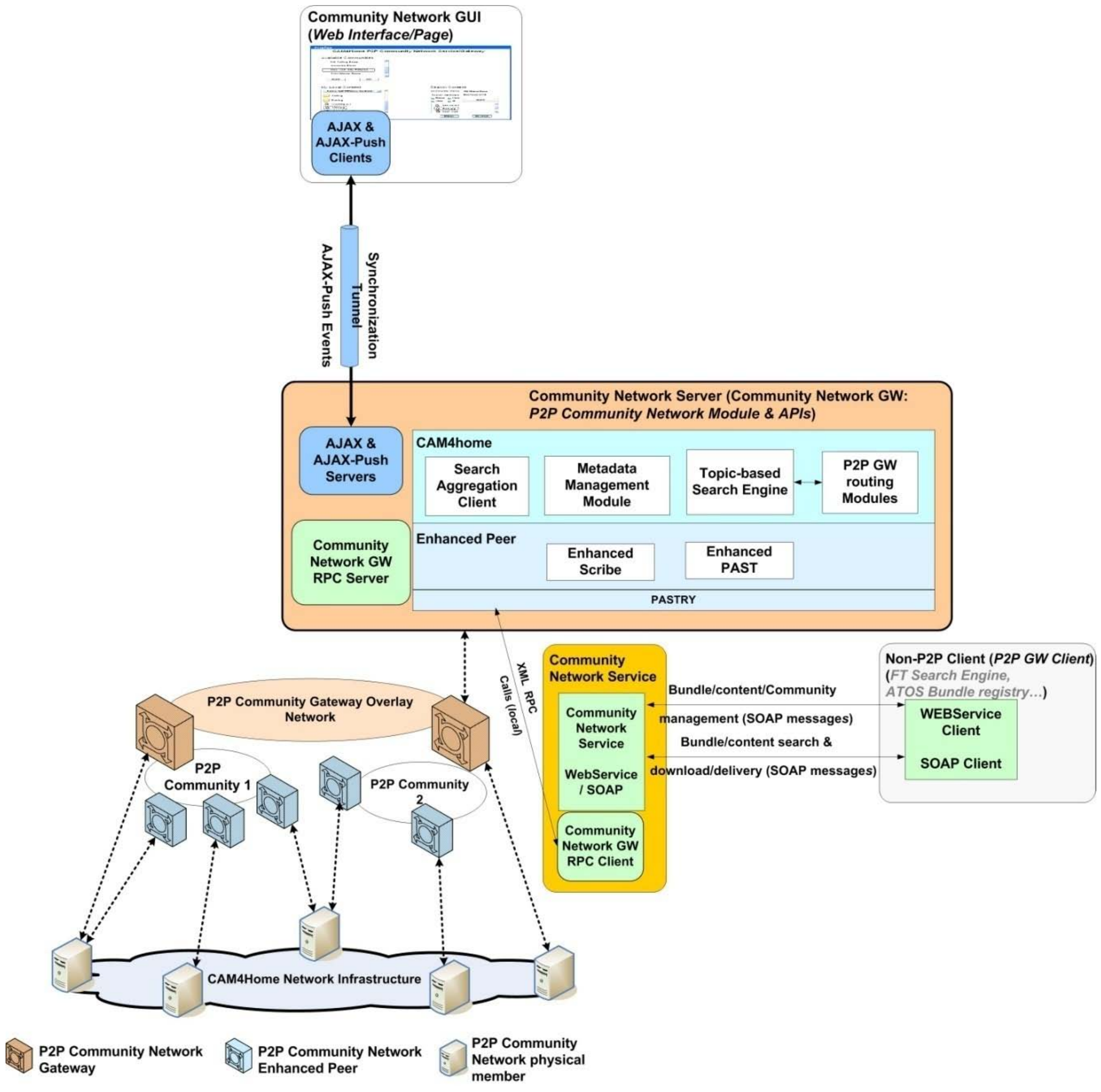

Community Network GUI (Web Interface/Page)
AJAX & AJAX-Push Clients
Synchronization Tunnel
AJAX-Push Events
Community Network Server (Community Network GW: P2P Community Network Module & APIs)
AJAX & AJAX-Push Servers
CAM4home
Search Aggregation Client
Metadata Management Module
Topic-based Search Engine
P2P GW routing Modules
Community Network GW RPC Server
Enhanced Peer
Enhanced Scribe
Enhanced PAST
PASTRY
XML RPC Calls (local)
Community Network Service
Community Network Service
WebService / SOAP
Community Network GW RPC Client
Bundle/content/Community management (SOAP messages)
Bundle/content search & download/delivery (SOAP messages)
Non-P2P Client (P2P GW Client) (FT Search Engine, ATOS Bundle registry…)
WEBService Client
SOAP Client
P2P Community Gateway Overlay Network
P2P Community 1
P2P Community 2
CAM4Home Network Infrastructure
P2P Community Network Gateway
P2P Community Network Enhanced Peer
P2P Community Network physical member

Figure 7. Community Network and GUI: overall architecture.

- Create Node

To Create a Scribe Node, we use the Pastry Create Node functionality and message. The address and the port of an existing node (the bootstrap node) have to be specified as inputs, and the system returns the Node Id of the new node. This Node Id is automatically generated by the P2P system. The created node can also be used afterwards as a bootstrap node.

- Create community

Our architecture offers to users the possibility of creating communities of interests. A P2P user can create a community with has a significant name called community topic. This would allow other users interested in that topic to subscribe in the corresponding community and share data.

To create a community called *Community Name*, a P2P node sends a *CREATE* (*Community Name*) message to the Scribe system. The message is received by the node having the closet ID to *Hash* (*Community Name*) (*Hash* beeing the hashing function). This node is called the Rendezvous node of the said community.

- Join Community

When a user is interested in a given community, she/he can subscribe and join it. The user would be able to receive the notification messages from community members. A P2P client has to perform a join operation to subscribe to the chosen community.

For this purpose, scribe node sends a JOIN message to the Group Id corresponding to the community it wants to join. Pastry routes this message to the rendezvous point using the forward method. There are two distinct situations: if the intermediate node is already a forward node, add this node as a child, if the intermediate node is not a forward node, create a child table of the group and add the node.

- Leave Community

When a user is no more interested by a community, he can leave it and triggers at its Scribe peer a subscription removal. When a Scribe node wants to perform a subscription removal from a peer group, it records locally that it left the group. Then, it checks if there are no other entries in its children table. If so, it sends a LEAVE message to its parents in its multicast tree. The message then travels recursively up the multicast tree until it reaches a node that has no more knowledge, in its routing table, of the node being removed.

- Search Community

Every user connected to the system can perform a search operation to discover the communities already created in the system. The user can join an existing community or choose to create a new one.

- Publish Metadata

This function allows users to publish and share their contents within a specific community. When publishing a multimedia content within a community, the system will execute all the aforementioned indexing algorithms for storing its associated metadata information with different hash keys within the DHT.

- Search

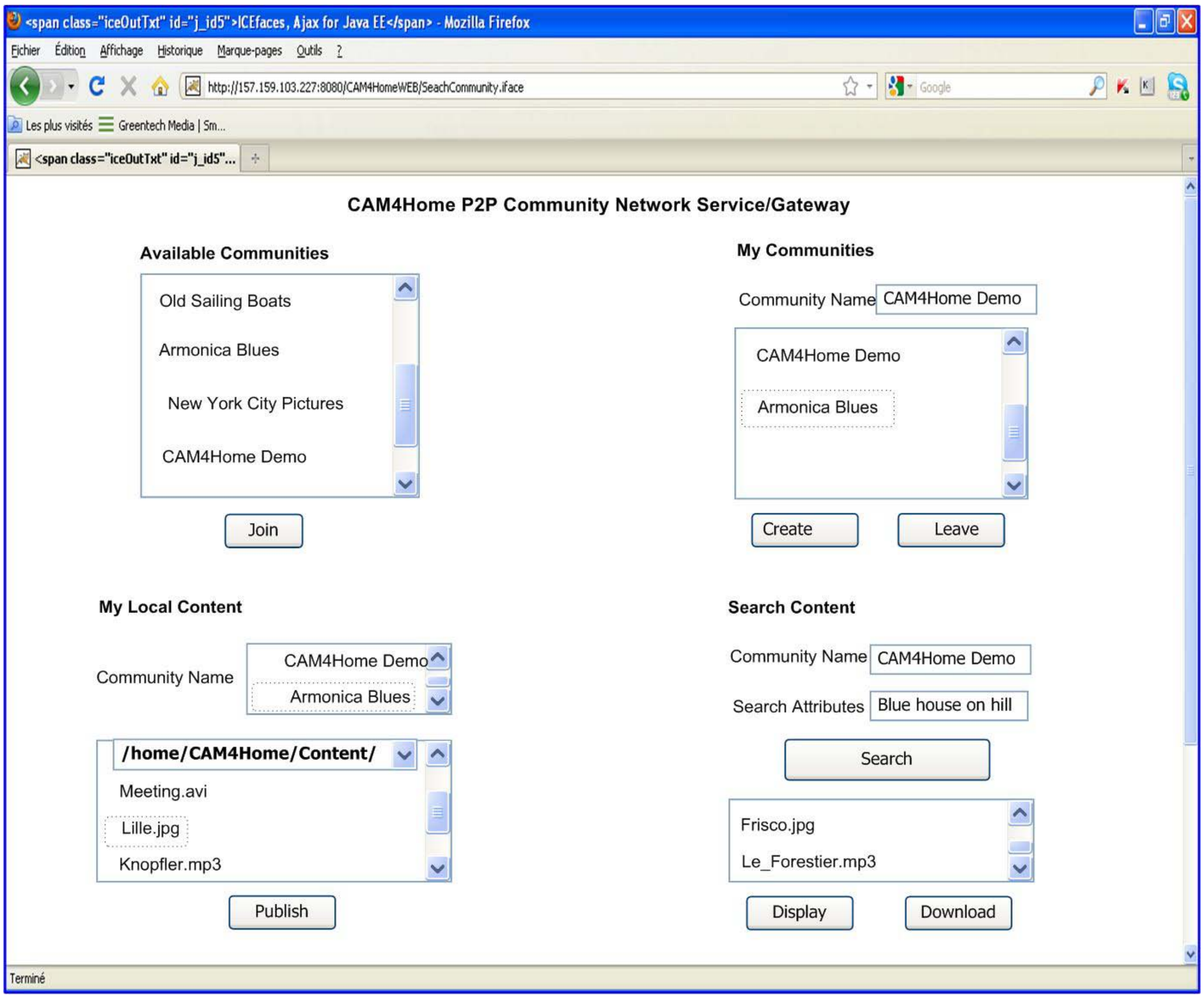
The search functionality is implemented as already described in Section 3.3: the search is performed using semantic queries expressed in XML and is simplified by the use of the indexation mechanism.

The Community Network Service is the Server-side application allowing any component or application to remotely interact with communities through the Community Network Gateway. It is designed in a SOA compliant way and implements Web Service components. For modularity reason, it is not directly implemented with the Community Gateway, but interacts with this latter through an RPC interface and the “P2P Module XML RPC Server” of the Community Gateway.

The Community Network GUIs, designed as modular web interfaces using AJAX framework [35], are synchronized in real time with Community Network GWs through AJAX push events and synchronization tunnels established between AJAX Push Servers of Community Network GWs and AJAX-Push Clients of Community Network GUIs. Let us recall that the synchronization between Community Network Gateways is carried out using the eventing mechanisms embedded in the Scribe/Pastry P2P framework. When the Community Network GUI is loading, a first web page is displayed to allow the end user to enter is User Id and access credential. This step is required for Authentication and Authorization purposes since only registered and authorized user can interact with Community Networks. If and only if the authentication step succeeds, the user will be redirected to the Community Network GUI main Web page. This main Web page, depicted in **Figure 8**, is decomposed in 4 main sub-parts, showing available communities, end user communities, end user local content, and retrieved content.

Through the Web interface, an end user can:

- Select an available community from the list of available communities for joining purposes,
- Create a community,
- Leave one of its belonging communities,
- Select one of its available contents and publish it in one of its communities,
- Search for a content in one or any of the existing communities. The end user just enters a natural language query in a dedicated field and click on the Search button. All the retrieved content will be automatically displayed,
- Select content from the “Search Content” sub-part and click on the Download button. This content is then downloaded locally in a dedicated directory.



<span class="iceOutTxt" id="j_id5">ICEfaces, Ajax for Java EE</span> - Mozilla Firefox

http://157.159.103.227:8080/CAM4HomeWEB/SeachCommunity.iface

CAM4Home P2P Community Network Service/Gateway

Available Communities

Old Sailing Boats
Armonica Blues
New York City Pictures
CAM4Home Demo

My Communities

Community Name: CAM4Home Demo

CAM4Home Demo
Armonica Blues

My Local Content

Community Name: CAM4Home Demo, Armonica Blues

/home/CAM4Home/Content/
Meeting.avi
Lille.jpg
Knopfler.mp3

Search Content

Community Name: CAM4Home Demo
Search Attributes: Blue house on hill

Frisco.jpg
Le_Forestier.mp3

Figure 8. Community Network GUI Web page.

5. Evaluation

P2P systems are popular to be scalable and fault tolerant. However, they lack of semantic which make search in P2P systems (especially DHTs) difficult for users. Our first method already introduces more semantic than common DHTs. However, this semantic is not so rich. We have therefore carried out our second method that implements a DHT based on enriched semantic. Only this second approach will be evaluated in that section.

Our implemented architecture relies on Scribe and Pastry. Those frameworks were already fully evaluated in [24,37] where the authors have shown their large scale scalability and efficiency. They will therefore not be reevaluated in this section where we will only validate our architecture by analyzing the impact of introducing our semantic storage/search mechanisms on the storage/search times and on the efficiency of our community network architecture.

Our community Network architecture is designed to handle complex multimedia contents which regroup several multimedia objects (video, image, text…) in a single bundle. Each bundle is described by an RDF documents identifying all necessary elements to reconstruct a complex multimedia content. By the way, we could find many photos and videos characterizing the same topic or event in barely 20 seconds as described in **Figure 9**. To evaluate our architecture, we use well established measures to calculate the response time needed to restore multiple bundles from the DHT in the same time. We find out that our system is accurate enough to restore all bundles related topic based on a single mundane keyword or less on an incomplete or misspelled keyword.

The search time needed for the system to get results varies according to the keyword introduced by the users. To have an accurate result, we calculate the mean value of search times. For the same result, we calculate the search time needed by the system to retrieve bundles

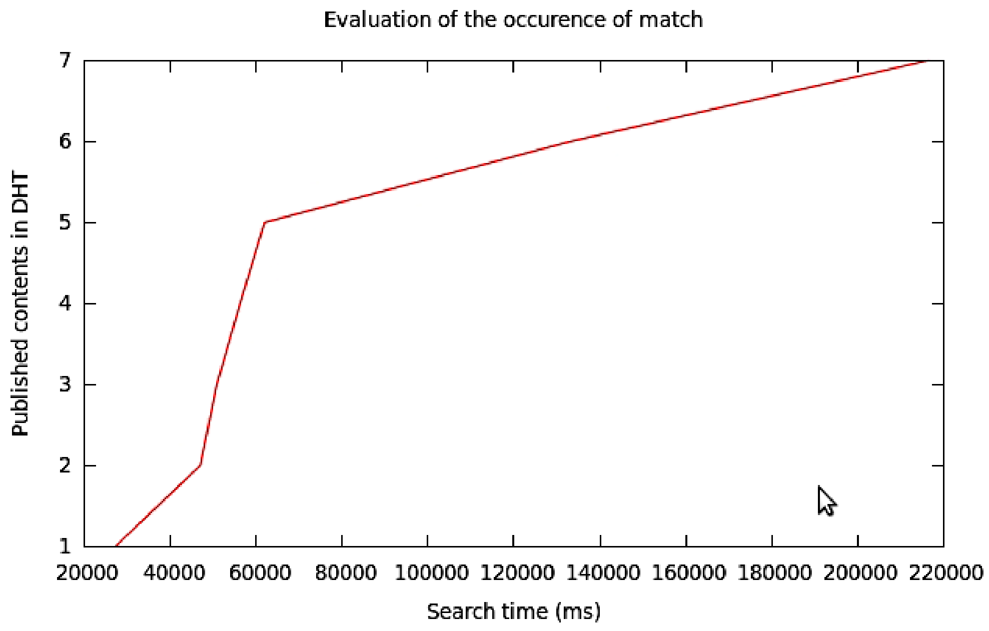


Figure 9. Evaluation of the occurrence of match.

based on different keywords. We have done the following experiments:

- We search all documents published in a specific community. We find out that the research time rises moderately.
- We search one or more documents related to a specific keyword.
- We search one or more documents according to incomplete or misspelled keywords. The research time is more important because of the size of the index used.

Moreover, the search time varies according to the depth of the RDF tree metadata file describing the multimedia bundle. The more the RDF file is deep, the more the Bundle is rich. **Figure 10** illustrates the variation of search time depending on the tree depth, in other words, depending on the complexity and richness of the multimedia content.

In this work, we are interested in introducing a semantic search mechanism to DHT. To have a flexible search, we have added a mechanism to correct misspelled or incomplete user keywords. We calculated the search time to extract the same content based on exact match keywords and on incomplete information. The slight extra time needed to do the search, shown in **Figure 11**, depends on the index size, which is bigger since users publish lots of contents. Moreover, as soon as the user introduces a wrong word, the correction mechanism will generate several possibilities of exact match keywords. This will generate in turn different combinations to recover the RDF metadata file from the RDF triple-based tree.

However, when we choose the exact match keywords of a single specific content, the research time is constant whatever the number of metadata content stored in the DHT is and it increases moderately when the keywords only belongs to several metadata files as illustrated in **Figure 12**. The yellow curve corresponds to the research times of a single content using the exact match keyword, whereas the green one corresponds to the research times of all the contents using the wildcard search attribute.

Moreover, the relevance of the answers is an important evaluation criterion. Thus, in our implemented architecture, the user could choose the precision of correcting words he wants (from 0.1 to 0.9 and 1). This precision influences the accuracy of results but not the response time. A good tradeoff between search flexibility and response time can be obtained by fixing the precision to 0.7. The precision of 1 corresponds to an exact match search.

In conclusion, introducing metadata and semantic aspects in the DHT ease the user's search, since the multimedia content is reachable from any mix of complete/incomplete information specified in its metadata file.

6. Conclusions

In this paper, we have presented and evaluated an original semantic-based community network architecture for heterogeneous multimedia content wide area sharing and retrieval. The use and the storage of CAM4Home semantic Metadata as content descriptions within the P2P Community (distributed in a semantic DHT) ensure that all members shared a common understanding of content.

We have enhanced the P2P communities with novel indexing techniques for indexing the data stored in the peer-to-peer network (indexes are distributed across the

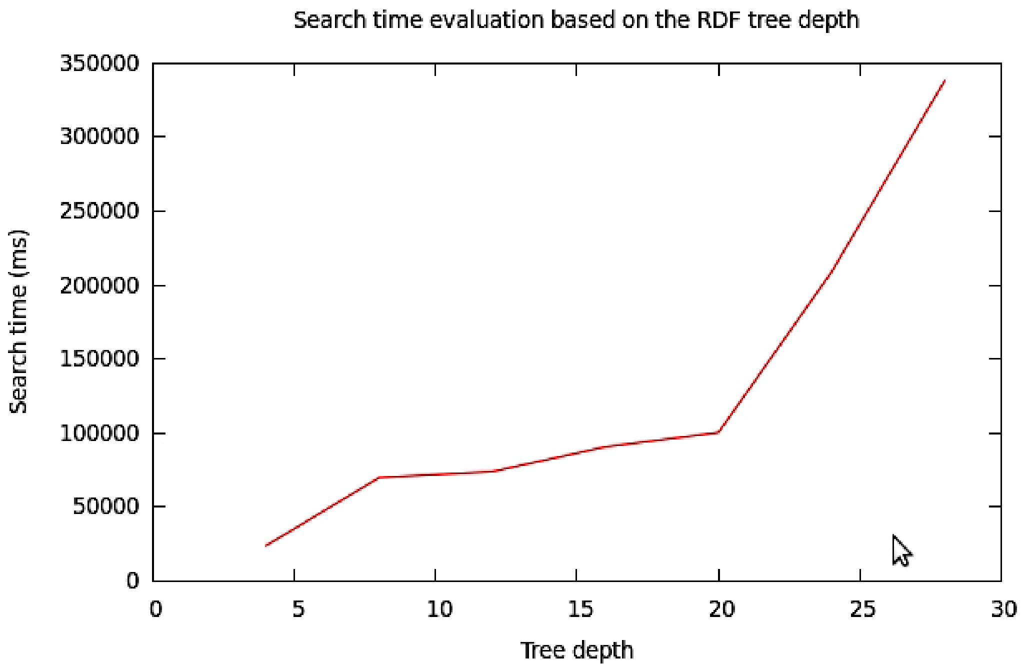


Figure 10. Search Time evaluation based on the RDF tree depth.

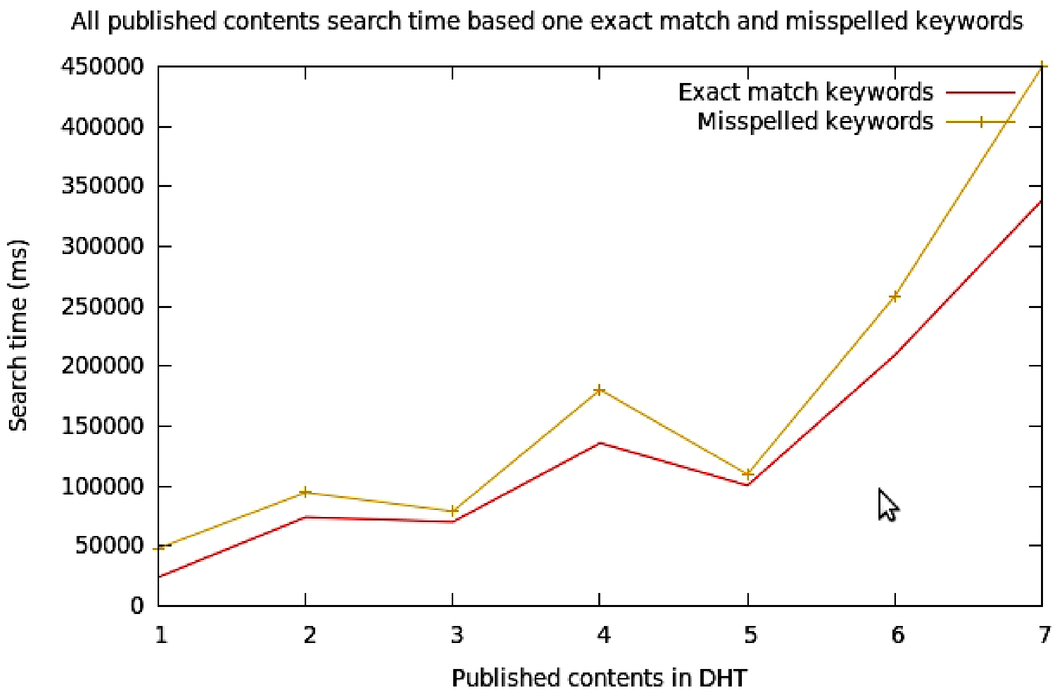


Figure 11. Additional search time for Misspelled keyword correction.

network and contain key-to-key or query-to-query mappings). Besides, the exact match lookup needed for search in DHT has been overcome by the introduction and use of two new indexes in the search mechanism. The community Network service design has also moved towards a more semantic oriented framework based on Subjects/Predicates/Objects search algorithm.

The proposed mechanisms allow our P2P Community to provide efficient multimedia content indexing (distributed) and retrieval mechanisms, at peer and Community Gateways level, thus offering full semantic distributed storage and search functionalities to users. The evaluations and tests conducted and depicted in this paper show that those mechanisms can be carried out in a scalable way.

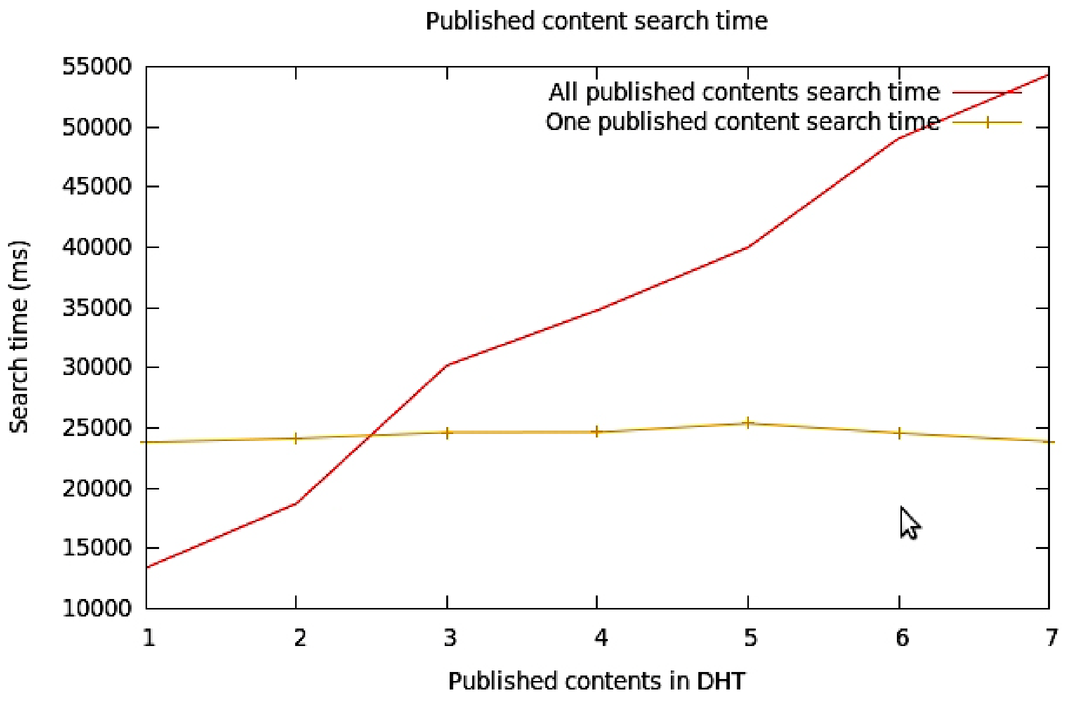


Figure 12. Constant search time to find an exact match content.

REFERENCES

[1] K. Lua, J. Crowcroft, M. Pias, R. Sharma and S. Lim, "A Survey and Comparison of Peer-to-Peer Overlay Network Schemes," *IEEE Communications Surveys & Tutorials*, 2005.

[2] M. Castro, M. Costa, A. Rowstron, "Peer-to-Peer Overlays: Structured, Unstructured, or Both?" Technical Report MSR-TR-2004-73, Microsoft Research, Cambridge, 2004.

[3] M. Castro, M. Costaand and A. Rowstron, "Debunking Some Myths about Structured and Unstructured Overlays," *Proceedings of NSDI*'05, Boston, May 2005.

[4] D. Stutzbach and R. Rejaie, "Characterizing Today's Gnutella Topology," *Proceedings of ACM SIGMETRICS* 2005, Alberta Canada, June 2005.

[5] Y. R. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham and S. Shenker, "Making Gnutella-Like p2p Systems Scalable," *Proceedings of SIGCOMM*'03, *Karlsruhe*, August 2003.

[6] Q. Lv, P. Cao, E. Cohen, K. Li and S. Shenker, "Search and Replication in Unstructured Peer-to-Peer Networks," *Proceedings of ACM ICS* 2002, New York City, June 2002.

[7] S. Saroiu, P. Gummadi, S. Gribble, *et al.*, "A Measurement Study of Peer-to-Peer File Sharing Systems," *Proceedings of ACM MMCN* 2002, San Jose, California, January 2002.

[8] P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-Peer Information System Based on the Xor Metric," *Proceedings of IPTPS* 2002, Cambridge, March 2002, pp. 53-65.

[9] S. El-Ansary and S. Haridi: An Overview of Structured P2P Overlay Networks," *Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks*, Aurebach Publications, 2006.

[10] M. Castro, M. Costa and A. Rowstron, "Performance and Dependability of Structured Peer-to-Peer Overlays," *Proceedings of IEEE DSN*'04, Washington, 2004.

[11] S. Wang, D. Xuan and W. Zhao, "On Resilience of Structured Peer-to-Peer Systems," *Proceedings of IEEE Globecom*, San Francisco, December 2003.

[12] Y. J. Joung, C. T. Fang and L. W. Yang, "Keyword Search in dht-Based Peer-to-Peer Networks," *Proceedings of IEEE ICDCS*'05, 2005, pp. 339-348.

[13] C. Xie, G. Chen, A. Vandenberg and Y. Pan, "Analysis of Hybrid P2P Overlay Network Topology," *Computer Communications*, Elsevier, Vol. 31, No. 2, 2008, pp. 190-200.

[14] H. Han, J. He and C. Zuo, "A Hybrid P2P Overlay Network for High Efficient Search," *Proceedings of ICIFE* 2010, Chongqing, September 2010, pp. 241-245

[15] Y.-F.u Yu, P.-J. Huang, Q.-J. Chen, T.-L. Huang and K.-C. Lai, "On the Design of Semi-structured Multi-Star Hybrid-Overlays for Multi-attribute Range Queries—Advances in Grid and Pervasive Computing," *Computer Science*, Springer, Vol. 6104, 2010, pp. 451-460.

[16] A. Kementsietsidis, M. Arenas and R. J. Miller, "Mapping Data in Peer-to-Peer Systems: Semantics and Algorithmic Issues," *Proceedings of ACM SIGMOD* 2003, San Diego, June 2003, pp. 325-336. http://dx.doi.org/10.1145/872757.872798

[17] M. Cai, M. Frank, B. Pan and R. Mac Gregor, "A Subscribable Peer-to-Peer rdf Repository for Distributed Metadata Management. *Journal of Web Semantics*, Vol. 2,

2005.

[18] A. Crespo and H. Garcia-Molina, "Semantic Overlay Networks for p2p Systems," Technical Report, Stanford University, January 2003.

[19] A. Mawlood-Yunis, M. Weiss and N. Santoro, "From P2P to Reliable Semantic P2P Systems," *Peer-to-Peer Networking and Applications*, *Springer Journal*, Vol. 3, 2010.

[20] T. Cerqueus, S. Cazalens and P. Lamarre, "Reducing the Semantic Heterogeneity of Unstructured P2P Systems: A Contribution Based on a Dissemination Protocol," *Transactions on Large-Scale Data- and Knowledge-Systems* VII, Springer, Heidelberg, 2012, pp. 62-95.

[21] CAM4Home ITEA2 Project. www.cam4home-itea.org/

[22] CAM4Home Deliverable D2.1, "Metamodel Definition". http://www.w3.org/

[23] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," *IFIP*/*ACM International Conference on Distributed Systems Platforms* (*Middleware*), Heidelberg, November 2001, pp. 329-350,

[24] M. Castro, P. Druschel, Y. C. Hu and A. Rowstron, "Topology-Aware Routing in Structured Peer-to-Peer Overlay," *Future Directions in Distributed Computing*, Springer-Verlag, 2003, pp. 103-107. http://dx.doi.org/10.1007/3-540-37795-6_19

[25] A. Rowstron, A. M. Kermarrec, M. Castro and P. Druschel, "Scribe: The Design of a Large-Scale Event Notification Infrastructure," In: J. Crowcroft and M. Hofmann, Eds., *Networked Group Communication*, 3*rd International COST*264 *Workshop* (*NGC*'2001) (Nov. 2001), Vol. 2233, pp. 30-43.

[26] I. Filali, F. Bongiovanni, F. Huet and F. Baude "RDF Data Indexing and Retrieval: A Survey of Peer-to-Peer Based Solutions," INRIA, 00540314, Sophia Antipolis, Novembre 2010.

[27] P. A. Felber, L. Garcés-Erice, E. W. Biersack and G. Urvoy-Keller, "Data Indexing in Peer-to-Peer DHT Networks," Institute EURECOM, 06904 Sophia Antipolis, France.

[28] H. T. Shen, Y. F. Shu and B. Yu, "Efficient Semantic-based Content Search in P2P Network," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 16, No. 7, July 2004.

[29] J. Y. Zhu and Y. Hu, "Efficient Semantic Search on DHT Overlays," *Journal of Parallel and Distributed Computing*, Vol. 67, No. 5, 2007, pp. 604-616. http://dx.doi.org/10.1016/j.jpdc.2007.01.005

[30] E. Liarou, S. Idreos and M. Koubarakis, "Evaluating Conjunctive Triple Pattern Queries over Large Structured Overlay Networks," *Proceedings of* 5*th International Semantic Web Conference*, Athens, November 2006.

[31] M. Cai and M. Frank, "MAAN: A Multi-Attribute Addressable Network for Grid Information Services," *Proceeding of* 4*th International Workshop on Peer-to-Per Systems*, February 2003.

[32] M. Cai and M. Frank, "RDFPeers: A Scalable Distributed RDF Repository Based on A Structured Peer-to-Peer Network," *Proceedings of the* 13*th International Conference on World Wide Web*, 2004, pp. 650-657. http://dx.doi.org/10.1145/988672.988760

[33] Cam4Home Deliverable D4.32, "Community Network and Services".

[34] A. Mesbah and A. van Deursen, "A Component and Push-Based Architectural Style for AJAX Applications," *Journal of Systems and Software*, Vol. 81, pp. 2194-2209.

[35] ICEfaces, "An Open Source Ajax Framework that Enables Java EE Application Developers to Create and Deploy Server-Based Rich Internet Application (RIA) Using the Java Language" http://www.icefaces.org/main/home/.

[36] M. Castro, P. Druschel and A. M. Kermarrec and A. Rowstron, "SCRIBE: A Large-Scale and Decentralised Application-Level Multicast Infrastructure," *IEEE Journal on Selected Areas in Communication* (*JSAC*), Vol. 20, No. 8, October 2002.