RDF Data Based Storage and Retrieval In Structured P2P Networks

Imen Filali, Fabrice Huet, Laurent Pellegrino
INRIA Sophia-Antipolis, Université de Nice Sophia-Antipolis, CNRS - I3S
2004, Route des Lucioles, BP 93, F-06902 Sophia-Antipolis Cedex, France
{last.first@sophia.inria.fr}

***Abstract*—Since its appearance, the Peer-to-Peer communication model has gained a significant interest due to its ability to build large scale distributed applications, such as file sharing, distributed computing. In spite of the diversity of such applications, the crucial issue is the data storage and retrieval mechanisms: how the data should be stored to be efficiently retrieved. Some P2P approaches, such as structured P2P systems, focus on a logical organization, sometimes supported by network topology, in order to efficiently store data items. These approaches are usually suitable for exact searches but do not support complex queries. Unstructured P2P overlays, on the other hand, can handle arbitrary complex queries at the cost of a higher overhead. Others solutions are focused not only on the network topology but also the semantic of stored items, moving from plain data storage to a predefined model of data description such as the Resource Description Framework (RDF).**

In this paper, we propose a fully distributed P2P architecture that combines a hierarchical P2P model with an RDF data storage model to come up with an efficient infrastructure for a reliable RDF data storage and retrieval. While the upper layer indexes the semantic nodes in chord like overlay network, the lower layer, a CAN-based overlay, exploits the RDF triple format to efficiently store triples in a fully distributed 3-dimensional space. The proposed architecture takes into account the dynamicity of the P2P network and supports the management of both simple and complex queries. [1]

***Index Terms*—Peer-to-Peer Overlays, CAN, Chord, RDF, Semantic Data Storage**

I. Introduction

The Peer-to-Peer (P2P) communication model has been asserted as a key solution to build robust and scalable distributed systems. While unstructured P2P systems like Gnutella [1] are not scalable because they generally rely on flooding search based approaches, structured P2P networks such as CAN [2], Chord [3], Pastry [4], Tapestry [5] have been shown, through many empirical studies, to be an efficient network topology model for data storage and retrieval in fully decentralized environment. However, all these DHTs approaches have flat topologies, without any hierarchical routing model. Even though they are considered as scalable systems with guaranteed logarithmic search complexity, the hierarchical P2P approaches can improve this complexity [6] [7]. Researches on P2P networks have focused not only on the network architecture but also on the semantic of the stored data, moving from simple keywords based data storage to more sophisticated RDF (Resource Description Framework) [8] based data format. In their simplest form, RDF based data items are described as triples <*subject, predicate, object*> where the subject identifies the resource to describe, the predicate presents the specific property in the statement and finally the object is the property value of the predicate. The RDF data model is flexible, i.e., it is simple to express any data in such format. Hence, a need has arisen to efficiently support storing and querying of RDF data.

Taking advantage of both technologies, a fully distributed P2P semantic infrastructure is proposed where P2P technology and RDF data model are working together to come up with a scalable P2P infrastructure. The proposed architecture provides complex and extendable descriptions of resources and allows the management of atomic, conjunctive and range-based queries. The contributions of this paper are:

- The design of a fully decentralized and hierarchical P2P infrastructure relying on Chord [3] and CAN [2] overlay networks
- A scalable storage infrastructure that allows the distributed storage of RDF triples.
- A distributed query processing and optimization based on SPARQL-like query language.

This paper is organized as follows: Section II briefly surveys related approaches. Section III presents the proposed solution of RDF data storage in fully decentralized peer-to-peer system and details the data indexation and query processing mechanisms. Section IV concludes the paper and outline future directions.

II. Related Work

Many P2P solutions have been proposed to ensure a distributed storage for RDF data. Some of them are built on top of super-peer-based infrastructure such as Edutella [9] which is a distributed RDF repository. In this approach, the network is composed of a set of super peers nodes. Each super peer is connected to a number of leaf nodes. Super-peers nodes manage local RDF repositories and are responsible for queries processing. This approach is not scalable for two main reasons. First, the super peers nodes are a single point of failure. Second, it uses the flooding-like search mechanism to route queries between super-peers.

[1]The presented work is funded by the EU FP7 NESSI strategic integrated project SOA4All; http://www.soa4all.eu

By using DHTs (Distributed Hash Tables), other systems, such as RDFPeers [10], address the scalability issue in the previous approach. RDFPeers is distributed repository built on top of Multi-Attribute Addressable Network (MAAN) [11]. Each triple is indexed three times by hashing its subject, its predicate and its object. This approach supports the processing of atomic triple patterns as well as conjunctive patterns limited to the same variable in the subject (e.g., $(?s, p_1, o_1) \wedge (?s, p_2, o_2)$ which indicates any subject with the given predicates and objects). The query processing algorithm intersects the candidate sets for the subject variable by routing them through the peers that hold the matches to each pattern. The approach proposed in [12] is similar to [10] in the sense that each triple is also stored three times. P-Grid [13] is a binary tree structured P2P overlay in which each $p \in \mathcal{P}$ is associated with a leaf node of the binary tree. Each leaf corresponds to a binary string $\pi \in \Pi$ such as Π is the entire key partition. Keys are generated using an order preserving hash function. Each peer is responsible for storing keys that fall under its current key space ($key \in \pi(p)$). Every peer's position is determined by its path. Peer's path indicates the subset of the tree's overall information that it is responsible for. Peers also maintain references to others peers in the binary tree. Queries are resolved by prefix matching. Thus, if a peer receives a query on key k that can not be locally resolved, it forwards the query to a peer, among its references, that prefixes k at most. Regarding the fault tolerance and query load balancing, multiple peers can be associated with the same key partition. GridVine [14] is build on top of P-Grid and uses the semantic overlay for managing and mapping data and metadata schemas on top of a physical layer. GridVine reuses two primitives of P-Grid: *insert(key,value)* and *retrieve(key)* for respectively data storage and retrieval. Triples are associated with three keys based on their subjects, objects and predicates. A lookup operation is performed by hashing the constant term(s) of the triple pattern. Once the key space is discovered, the query will be forwarded to peers responsible for that key space.

III. System Architecture

As already mentioned, we aim to provide a scalable distributed infrastructure for storing and retrieving of semantic data. In this regard, a natural solution to support these features is to exploit the DHT based peer-to-peer approaches. Since the proposed P2P architecture is relayed on CAN and Chord structured overlays networks, we first introduce these two overlays as well as the RDF data model. Then, we detail the P2P semantic space infrastructure and different algorithms for both data indexation and querying.

A. Background

CAN. In [2], Ratnasamy *et al.* describe a structured P2P network based on a d-dimensional Cartesian coordinate space. This Cartesian space is dynamically partitioned among all peers in the system such that each node "owns" a zone in the overall space. This virtual coordinate space is used to store $(key, value)$ pairs. For instance, to store the $(key, value)$ pair, the key k is deterministically mapped onto a point p in the coordinate space. The $(key, value)$ pair is then stored at the node that owns the zone which the point p lies in. The lookup process of a value v associated to a key k is achieved by applying the same deterministic function on k in order to map it into p. The query then will be routed through intermediate nodes in the overlay until it reaches the peer's zone containing p. When a peer joins the CAN overlay, it picks a random point $p\prime$ belonging to the Cartesian coordinate space and a *JOIN_QUERY* will be routed to the zone that contains that point. A zone will then be allocated to the new peer by splitting the current peer's zone in half: keeping half for the current peer owner and allocate the other one to the new peer. On the average, each node keeps information about $O(d)$ neighbors and a routing path involves $O(dN^{1/d})$ overlay hops such as d is the space dimensionality and N is the number of nodes in the network.

Chord. In [3], peers are organized in a logical ring topology. Each Chord node has an identifier id that represents its position in a circular identifier space of size N. A node keeps $m = log(N)$ routing entries, called fingers. For $1 \leq i \leq m$, a $finger[i]$ of a node n contains the address of a node whose identifier equals to $n + 2^{i-1}$. During a *retrieve(key)* operation, a peer forwards a query to the finger with the largest id that precedes the *key* value. The process is repeated from peer to another until the peer preceding the *key* is reached, which is the "closest" peer to the *key*.

When a new peer p joins the network, a set of keys previously assigned to p's successor will be assigned to p. Inversely, when p leaves the network, all keys assigned to it will be reassigned to its successor. These are the only changes in key assignments that have to occur in order to maintain a load balancing between peers.

RDF Data Model. Resource Description Framework [8] is recognized as a standard for storing and exchanging information on the Semantic Web. All resources have unique identifiers presented as *Uniform Resource Identifiers (URIs)*. These resources are described through properties and property values. Consequently, RDF data model can represent simple statements about resources, their properties and values. It defines the so-called *RDF* statements which consist of a subject, a predicate, and an object: *triple=<subject,predicate,object>*. These statements can be created by different users and widely distributed on the Web. With the increasing use of RDF statements, there is a need to support efficient retrieval of RDF data in large scale distributed environment.

B. A Semantic Space P2P infrastructure

1) Semantic Spaces: Semantic Spaces are considered as a new type of communication platform that has recently gained a significant interest in the middleware community, as a response to the raising challenges of data sharing and service coordination in large scale distributed and highly dynamic Web environment.

A Semantic Space exposes a set of operations for the publication and querying of RDF data, via SPARQL-like [15]

query language. It also offers subscription to published events in form of queries or triple patterns. Furthermore, a set of management operations aiming at the creation and deletion of spaces is also provided. Information stored in the space are represented in RDF data model.

Architecturally speaking, the Semantic Space infrastructure is build on top of peer-to-peer overlays. It exploits peer-to-peer communication model to distribute the RDF triples that are published and stored in the infrastructure.

2) CAN overlays for RDF storage: A Semantic Space is implemented using a 3-dimensional CAN overlay with lexicographical ordering. The three dimensions of each CAN coordinate space represent respectively the subject, the predicate and the object of the stored RDF triples. Thus, a triple directly represents a point in the CAN space without the use of hash functions. This preserves common prefix between elements of different triple and gives a form of clustering. Elements with a common prefix will be localized close to each other. If necessary, the number of dimensions can be increased to handle meta-information for each RDF triple.

Figure 1 depicts a CAN overlay where axis represent the RDF triples elements. As an example, the figures shows the indexation of triple *(CAN,creator,ratnasmy)*. The peer managing the zone where the point with coordinates *(CAN,creator,ratnasmy)* falls is responsible for the storage of that triple.

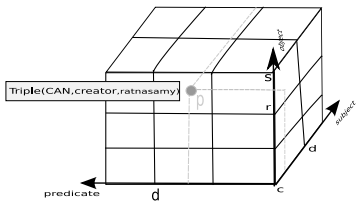

Fig. 1. Three-dimensional CAN space. Axis present the subject, the predicate and the object of RDF triples.

3) Chord overlay for Spaces organization: Using a structured overlay to implement a Space can limit the scalability. As the number of peers increases, handling churn can have a high cost. We thus offer a two level design based on dinstinct CAN overlays accessible through a **unified** view, giving the impression of a single Space. At the lowest level, a set of individual Spaces are implemented using CAN overlays. At the higher level, they are all linked together using a Chord, as shown in Figure 2. Each node of the Chord overlay (e.g., $\{S_1, S_2, S_3, S_4\}$ in Figure 2) stores references to individual Semantic Spaces and maintains references to a set of peers that may belong to different CAN overlays. In others words, the individual spaces in Chord layer are disjoint, and the write operations are targeted at a particular space by means of the *SpaceURI* which is considered as the unique identifier of a Semantic Space maintained at the Chord level.

Comparing with the flat topology, the proposed semantic space infrastructure takes advantage of the P2P hierarchical in many aspects:

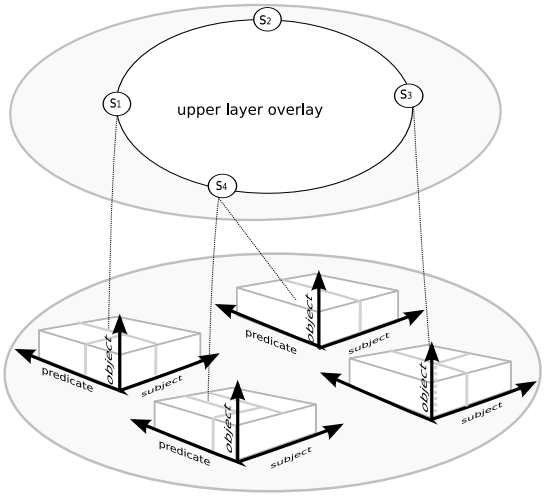

Fig. 2. Peer-To-Peer Semantic Space Infrastructure.The upper layer is Chord based overlay (semantic spaces). CAN axis represent the subject, the predicate and the object of RDF triples.

- Individual spaces can be created and managed based on semantic properties of data (e.g., all data regarding a particular topic).
- The complexity of the Space infrastructure is hidden by the Chord layer, the only one visible by a user.
- The Chord layer is more stable since it presents a virtual organization of Semantic Spaces. In other words, the Chord node associated with a CAN overlay is not affected when an "ordinary" peer at the level of CAN layer joins or leaves the semantic space. In that case, only the CAN overlay has to deal with the peer join and failure and the update of peers' zones. We believe that the Chord layer maintenance cost is low since it is only responsible for Semantic Spaces indexation and all RDF triples are managed at the CAN level.

Figure 3 depicts a high level overview of the internal architecture of a peer. Both CAN and Chord peers share the same building blocks.

The user API enables a client to publish and access the information in the Semantic Space. This includes write, read, delete operations.

Each node that shares a given space has its own query engine that resolves user requests against the locally stored data. It takes care of reasoning and optimizing the query before being executed, as well as the query distribution across different peers in the Semantic Spaces. Each node of the P2P network is associated to a local RDF data store called OWLIM [16]. It provides basic and efficient reasoning facilities for efficient and effective retrieval of RDF data. The overlay layer composed of Chord and CAN process incoming data and route queries to relevant peers.

While the basic idea appears simple, there are many issues that have to be considered such as distributed data storage, overlay maintenance and processing. These points will be discussed in the remainder of this paper.

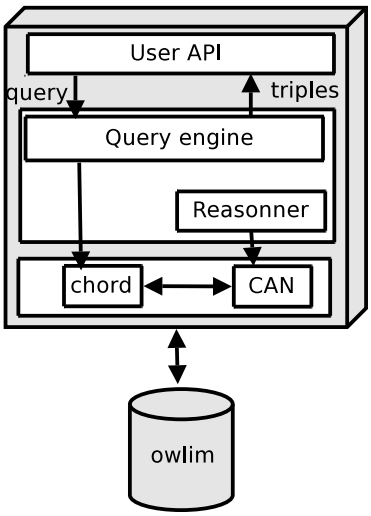

Fig. 3. Peer architecture

C. RDF Data Processing

1) Data Indexation: As mentioned earlier, a client will query a space at the Chord level. This operation will be performed through `write(SpaceURI uri, triple t)`. As a consequence, a write operation runs in two phases. First, the query is routed to the appropriate space at the Chord level by means of the `spaceURI`. Second, once the space is discovered, the triple`t` is routed towards the appropriate peer in the CAN overlay, i.e., it is routed to the peer that manage the zone where the point p indexed by $(t_{subject}, t_{predicate}, t_{object})$ falls.

2) Data processing: Query resolution algorithm takes into consideration the complexity of the query. As we maintain a lexicographic order on the CAN axis, the proposed architecture allows not only the resolution of simple atomic queries but also the conjunctive and disjunctive range queries. Unlike, write operation that can be only resolved against one particular space, read operations can either resolved against one semantic space or an undetermined set of spaces.

Atomic queries. are triples where the subject, the predicate and the object can either be variables or constant values. They are answered by first looking to the constant part(s) for the triple pattern. As an example, the query $q = (s_i, ?p, ?o)$ looks, for a given subject s_i, for all possible objects and predicates. To resolve this query pattern, the query will be routed on the *subject-axis* of the CAN overlay looking for the subject value s_i. Once the peer responsible for the specified value s_i is found, it forwards the query only to its neighbors that are most likely in the direction of peers storing the corresponding triples based on the peer zone's coordinates.

Conjunctive queries. are expressed as a conjunction of a set of atomic triple patterns (sub-queries), atomic triples will be processed first. Their results set would be returned and the intersection will be handled by the peer which has originally received the query.

Range queries. can be efficiently supported due to the preservation of lexicographic order at the CAN axis. As an example, we consider the following query q=($< s >< p >?o$ *FILTER* $(v_1 \leq ?o \leq v_2)$) with a given subject s and predicate p. It looks for a set of objects, given by the variable $?o$, such as $v_1 \leq o \leq v_2$. The routing process starts by finding the constant part(s) of the query. After that, it locates the lowest and the highest values by going over the object axis. If all results are found locally, they are returned to the requester. Otherwise, the query is forwarded to neighbors that may contain other potential results.

D. Peer churn

As the data management operations, the `JOIN` operation exploits the multi-layer structure of the P2P network. As each Chord node can reference a set of CAN overlays, it stores references to a set of CAN peers. More specifically, when a peer p wants to join a specific Space, it asks an arbitrary Chord node. This information is maintained by a tracker which stores references to Chord nodes currently in the system. After finding the Chord node identified by *hash(spaceURI)*, the joining peer obtains references to CAN nodes and contact them to join the overlay. When a peer p leaves the CAN overlay, we have to ensure that its zone is taken over by its neighbors. To do so, each peer stores the last direction of its zone split operation. Before leaving, the peer p splits each zone on n sub-zones such as n is the number of neighbors on the side of the last split direction. Each sub-zone is assigned to one neighbor.

Note also that the Chord protocol itself has a stabilization algorithm periodically executed by each node to check if there is new Chord node joining or leaving the network. This results in an update of the finger tables and the successor and predecessor pointers.

IV. Conclusions and Future Works

In this paper, we have proposed an RDF-based P2P approach for RDF data storage and retrieval in fully distributed environment. We have also presented a set of algorithms in order to efficiently store and process complex RDF queries.

Currently, we are working on the implementation of this P2P architecture using ProActive middleware. We aim to evaluate different algorithms in a real benchmark data set using a simple and complex queries while taking into account dynamicity of the peer-to-peer network.

References

[1] "Gnutella," `http://www.gnutella2.com/gnutella2`.

[2] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A scalable content-addressable network," in *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '01)*, vol. 31, no. 4. ACM Press, October 2001, pp. 161–172.

[3] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '01)*. New York, NY, USA: ACM, 2001, pp. 149–160.

[4] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*. Springer-Verlag, 2001, pp. 329–350.

[5] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz, "Tapestry: a resilient global-scale overlay for service deployment," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 41–53, 2004.

[6] I. Martinez-Yelmo, R. Cuevas, C. Guerrero, and A. Mauthe, "Routing performance in a hierarchical dht-based overlay network," *Euromicro Conference on Parallel, Distributed, and Network-Based Processing*.

[7] L. Garcs-erice, E. W. Biersack, P. A. Felber, K. W. Ross, and G. Urvoy-keller, "Hierarchical peer-to-peer systems," in *Proceedings of ACM/IFIP International Conference on Parallel and Distributed Computing (Euro-Par)*, 2003, pp. 643–657.

[8] "Resource Description Framwork," http://www.w3.org/RDF/.

[9] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmer, and T. Risch, "Edutella: A P2P networking infrastructure based on RDF," in *Proceedings of the 11 International World Wide Web Conference*, Honolulu, USA, May 2002.

[10] M. Cai and M. R. Frank, "RDFPeers: a scalable distributed RDF repository based on a structured peer-to-peer network," in *WWW*, 2004, pp. 650–657.

[11] M. Cai, M. Frank, J. Chen, and P. Szekely, "MAAN: A multi-attribute addressable network for grid information services," in *Journal of Grid Computing*, vol. 2, 2003.

[12] R. Giuseppe, D. G. Federico, D. N. Pierluigi, S. Antonio, and D. M. J. Carlos, "A peer-to-peer architecture for distributed and reliable RDF storage," in *First International Conference on Networked Digital Technologies (NDT)*, July 2009, pp. 94–99.

[13] A. D. Z. D. M. H. M. P. Karl Aberer, Philippe Cudre-Mauroux and R. Schmidt, "P-grid: A self-organizing structured p2p system," 2003.

[14] K. Aberer, P. Cudr-Mauroux, M. Hauswirth, and T. V. Pelt, "GridVine: Building Internet-Scale Semantic Overlay Networks," in *International Semantic Web Conference*, 2004.

[15] E. Prud'hommeaux and A. Seaborne, "SPARQL Query Language for RDF," W3C Recommendation, January 2008, `http://www.w3.org/TR/rdf-sparql-query/`.

[16] "Owlim," `http://www.ontotext.com/owlim/`.