Journal of Network and Computer Applications 46 (2014) 124–138

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

An RDF-based P2P overlay network supporting range and wildcard queries

Ing-Chau Chang [a], Eric Jui-Lin Lu [b,*], Shiuan-Yin Huang [b], Yi-Hui Chen [c,d,**]

[a] Department of Computer Science and Information Engineering, National Changhua University of Education, Changhua, Taiwan, ROC
[b] Department of Management Information Systems, National Chung Hsing University, Taichung, Taiwan, ROC
[c] Department of Applied Informatics and Multimedia, Asia University, Taichung, Taiwan, ROC
[d] Department of Medical Research, China Medical University Hospital, China Medical University, Taichung, Taiwan

ARTICLE INFO

Article history:
Received 8 April 2013
Received in revised form
19 May 2014
Accepted 10 August 2014
Available online 27 August 2014

Keywords:
Distributed Hash Table
DHT
Peer-to-Peer overlay network
RDF
Wildcard

ABSTRACT

In recent years, a new branch of the P2P research called the semantic-based system has been emerged. The semantic-based P2P overlay network uses metadata to describe resources so that complex queries can be implemented. Although systems such as RDFPeers and RDF-Chord support complex queries including the range query, none of these systems in the literature supports the wildcard query. In this paper, we propose a RDF-based P2P overlay network, called RDFChord-W, which supports all atomic queries, conjunctive and disjunctive queries, the range query, and the wildcard query. In RDFChord-W, nodes are arranged into multiple layers where each layer is organized as a Chord-like ring. Resources are described by their RDF triples. Hence, the indices of resources can be generated by hashing their RDF triples and then distributed to corresponding nodes. Based on our survey in the P2P literature, RDFChord-W is the first RDF-based P2P overlay network that supports the wildcard query. Although RDFChord-W supports various types of queries, its performance is much superior to traditional P2P overlay networks such as RDFPeers and Squid through intensive experiments, which are executed by the well-known simulator PeerSim.

1. Introduction

Due to the tremendous growth of Internet usage, more and more resources are shared in Peer-to-Peer (P2P) systems. Among various types of network architectures, structured P2P systems such as Chord (Stoica et al., 2003), Pastry (Rowstron and Druschel, 2001), and Tapestry (Bloom, 1970) became very popular because of their efficiencies and the guaranteed-to-find property. In general, a structured P2P system generates indices of resources by hashing their names or keywords with a Distributed Hash Table (DHT), and then saves indices in the corresponding nodes. When receiving a query, a node in a structured P2P system hashes the query to generate a key, and the node storing the index that matches the key is obtained. Hence, this kind of structured P2P system is also called the DHT-based system.

According to the design of DHT, the query time complexity of most structured P2P systems is $O(\log n)$ where n is the total number of nodes. Because traditional DHT-based P2P systems only support exact-match queries, they are not suitable for complex queries (Sagan, 1994). For example, for a query Q("author", "Eric"), resources described with keywords "author" and "Eric" can be found, while resources described with keywords "author" and "Eric Lu" cannot. As a result, many DHT-based P2P systems have been proposed to support complex queries such as the range query (Cai et al., 2004a; Li et al., 2009; Schmidt and Parashar, 2008) and wildcard query (Ahmed and Boutaba, 2007, 2009; Joung and Yang, 2009; Schmidt and Parashar, 2008).

In recent years, a new branch of P2P systems called the semantic-based system has emerged. In the semantic-based system, metadata are used for describing resources. With the richness of metadata, queries can be more complex and flexible. Semantic-based systems can be classified into two categories according to their objectives. The main objective of systems such as Peer-to-Peer Semantic Link Network (P2PSLN) (Zhuge et al., 2005), R-Chord (Liu and Zhuge, 2006), GridVine (Cudre-Mauroux et al., 2007), Atlas (Kaoudi et al., 2010), and Two-Level Chord (Yeh et al., 2010) in the first category is to enrich queries by utilizing and deriving the semantics of metadata. For example, in R-Chord, relevant resources can be described by using semantic links such as *Equal-to*. Thus, when resource A is retrieved by a query, resource B will be obtained if A is Equal-to B. The main objective of systems such as RDFPeers (Cai et al., 2004b) and RDF-Chord (Lu et al., 2012)

* Corresponding author. Department of Management Information Systems, National Chung Hsing University, Taichung, Taiwan R.O.C. Tel.: +886 4 22840864; fax: +886 4 22857173.
** Corresponding author.
E-mail addresses: jllu@nchu.edu.tw (E.-L. Lu), chenyh@asia.edu.tw (Y.-H. Chen).

http://dx.doi.org/10.1016/j.jnca.2014.08.005
1084-8045/

in the second category is to integrate semantic queries with complex queries such as the range query and provide efficient query lookups. Generally, systems in this category adopt the DHT-based architecture and use the Resource Description Framework (RDF) (2013) to annotate resources. Our research in this paper falls into this category.

RDF is one of the popular languages for representing metadata developed by World Wide Web Consortium (W3C). An RDF document is composed of a set of RDF triples. Each triple t is represented as (*subject*, *predicate*, *object*), where *subject* denotes a resource, *predicate* defines a specific property of the resource, and *object* describes the actual value of the predicate. A resource can be described as one or more RDF triples. For example, if the unit price of an url is 100 and the publication year of the url is 2013, it can be described by RDF triples such as (url, price, 100) and (url, year, 2013). In RDFPeers, indices are generated by hashing each field of a RDF triple with a locality-preserving hash function. Hence, RDFPeers supports RDF atomic query, disjunctive query, conjunctive multi-predicate query, and range query. Because disjunctive-and-range query is accomplished by issuing multiple atomic queries, this results in extra overhead. In order to provide more efficient range queries by fully utilizing the relationships between any two fields of a RDF triple, nine RDF keys are designed and proposed in our preliminary work, RDF-Chord (Lu et al., 2012). Although the design of RDF keys in RDF-Chord makes execution speeds of range queries at least 100 times faster than those of RDFPeers, the maintenance cost of RDF-Chord is higher than that of RDFPeers because RDF-Chord has to maintain routing information for three ring-sets.

To the best knowledge, there are no semantic-based P2P systems supporting the wildcard query. Therefore, to support the wildcard query and reduce high maintenance overhead, we will propose RDFChord-W in this paper. In RDFChord-W, we also design new locality-preserving hash functions and four RDF-W keys adapted from the concept of RDF keys. Hence, RDFChord-W not only supports all RDF atomic queries, conjunctive multi-predicate query, disjunctive query, and range query, but also supports three forms of wildcard queries, i.e., prefix-match query, suffix-match query, and pre/suffix-match query. Additionally, in RDFChord-W, the number of ring sets is reduced from three to one. The maintenance overhead, based on our experimental results, is in average about 20 and 40 hops lower than those of RDFPeers and Squid, respectively. Additionally, the cost of its range query is about 100 and 10 times faster than those of RDFPeers and Squid, respectively.

The rest of this paper is organized as follows. In Section 2, we briefly introduce the range query as well as the wildcard query, and then give an overview of existing systems that support these two kinds of queries. In Section 3, we first introduce the design of indices, and then describe the architecture, query resolving, node joining, and maintenance processes of RDFChord-W. Results of various simulations are presented and analyzed in Section 4. Finally, conclusions and possible future works are presented in Section 5.

2. Related works

2.1. Systems supporting the range query

Non-semantic-based P2P systems such as MAAN (Cai et al., 2004a), Squid (Schmidt and Parashar, 2008), and Armada (Li et al., 2009) are DHT-based systems supporting the range query. To support the range query, these systems utilize the locality-preserving hash function to hash keywords. A hash function H is a locality preserving hash function if it has the following property: $H(v_i) < H(v_j)$ *if* $v_i < v_j$, and if an interval $[v_i, v_j]$ is split into $[v_i, v_k]$ and $[v_k, v_j]$, the corresponding interval $[H(v_i), H(v_j)]$ can be split into $[H(v_i), H(v_k)]$ and $[H(v_k), H(v_j)]$ (Cai et al., 2004a). With such a feature, for a range query Q(2000–2500), the relationship $H(2000) < H(2500)$ holds and resources whose hashed keyword values falls between $H(2000)$ and $H(2500)$ will be retrieved.

Semantic-based systems that support range queries include RDFPeers and RDF-Chord. In RDFPeers, indices are generated by hashing each field of RDF triples with a locality-preserving hash function. Similar to non-semantic-based systems, for a range query Q(?*s*,?*p*, 2000–2500), resources whose hashed object values falls between $H(2000)$ and $H(2500)$ will be retrieved. If ?*p* is replaced by "*price*", the range query now becomes Q(?*s*, "*price*", 2000–2500). To process the query, these systems have to issue a query Q (?*s*,?*p*, 2000–2500) and then select only the resources whose *predicates* are "*price*" as queried results. It is noted that, in the query process of Q(?*s*,?*p*, 2000–2500), all nodes whose hashed *object* values falls between $H(2000)$ and $H(2500)$ will be retrieved. In other words, nodes that do not match Q(?*s*, "*price*", 2000–2500), e.g., (*s*, "*weight*", 2200) or (*s*, "*year*", 2010), will also be retrieved, and this results in extra overhead.

To resolve the above problem, each node in RDF-Chord is distributed to three ring sets, i.e., the *subject* ring set, the *predicate* ring set, and the *object* ring set, based on the RDF triples of the resources that the node has. Each ring set is split into multiple layers, where nodes on each layer are organized into a Chord-like ring. Layers in each ring set are linked by nodes called bridge peers (BP), where BPs are nodes that are relatively stable and with higher processing power or bandwidth. All BPs are organized into a Chord-like BP layer.

In order to take advantages of metadata and provide more efficient range query, RDF-Chord designed novel RDF keys for each type of RDF atomic query. Various types of the atomic query and their corresponding keys are shown in Table 1, where H_x is a locality-preserving hash function with 2^x hash space, and x is half of the length of index space m. With the design of RDF keys, to find resources whose *predicate* is "*price*" and the value of *object* falls between 2000 and 2500, only one query Q(?*s*, "*price*", 2000–2500) has to be issued, and the query will not be sent to nodes with resource such as (*s*, "*weight*", 2200) or (*s*, "*year*", 2010). Therefore, the range query overhead is significantly reduced.

Table 1
The RDF atomic queries and the corresponding keys.

Query type	Query semantic	Key
(*s*, ?*p*, ?*o*)	Find all the *p* and *o* of the triples which have matched *s*.	$k_s = H_x(s) \times 2^x$
(*s*, *p*, ?*o*)	Find all the *o* of the triples which have matched *s* and *p*.	$k_{sp} = H_x(s) \times 2^x + H_x(p)$
(*s*, ?*p*, *o*)	Find all the *p* of the triples which have matched *s* and *p*.	$k_{so} = H_x(s) \times 2^x + H_x(o)$
(?*s*, *p*, ?*o*)	Find all the *s* and *o* of the triples which have matched *p*.	$k_p = H_x(p) \times 2^x$
(?*s*, *p*, *o*)	Find all the *s* of the triples which have matched *p* and *o*.	$k_{po} = H_x(p) \times 2^x + H_x(o)$
(?*s*, ?*p*, *o*)	Find all the *o* of the triples which have matched *s* and *p*.	$k_o = H_x(o) \times 2^x$