Publish/Subscribe Functionalities for Future Digital Libraries using Structured Overlay Networks

Christos Tryfonopoulos Stratos Idreos Manolis Koubarakis

Dept. of Electronic & Computer Engineering
Technical University of Crete, 73100 Chania, Crete, Greece
{trifon,sidraios,manolis}@intelligence.tuc.gr

1 Introduction

We are interested in the problem of *distributed resource sharing* in future digital libraries (DLs). We adopt a pure P2P architecture (illustrated in Figure 1), but our ideas can be easily modified to work in the case of hierarchical P2P networks, as in [3]. *Information providers* (DLs) and *information consumers* (users) are both represented by peers participating in a peer-to-peer (P2P) overlay network. There are two kinds of basic functionality that we expect this architecture to offer: *information retrieval (IR)* and *publish/subscribe (pub/sub)*. In an IR scenario a user poses a *query* (e.g., "I am interested in papers on bio-informatics") and the system returns information about matching resources. In a pub/sub scenario (also known as *information filtering (IF)* or *selective dissemination of information (SDI)*) a user posts a *subscription* (or *profile* or *continuous query*) to the system to receive notifications whenever certain events of interest take place (e.g., when a paper on bio-informatics becomes available).

In this extended abstract we concentrate on the latter kind of functionality (pub/sub) and sketch how to provide it by extending the distributed hash table Chord [4]. *Distributed Hash Tables (DHTs)* are the second generation *structured* P2P overlay networks devised as a remedy for the known limitations of earlier P2P networks such as Napster and Gnutella. We present a set of protocols, collectively called *DHTrie*, that extend the Chord protocols with pub/sub functionality.

We assume that resources are annotated using a well-understood attribute-value model called $\mathcal{AWPS}$ in [2]. Thus publications and subscriptions will also be expressed in $\mathcal{AWPS}$. $\mathcal{AWPS}$ is based on *named attributes* with value *free text* interpreted under the Boolean and vector space (VSM) models. The query language of $\mathcal{AWPS}$ allows Boolean combinations of comparisons A *op* v, where A is an attribute, v is a text value and *op* is one of the operators "equals", "contains" or "similar" ("equals" and "contains" are Boolean operators and "similar" is interpreted using the VSM or LSI model). The following is an example of a publication in $\mathcal{AWPS}$:

$$\{\ (AUTHOR, \text{“}John\ Smith\text{”}),\ (TITLE, \text{“}Information\ dissemination\ in\ P2P\ ...\text{”}),\ (ABSTRACT, \text{“}In\ this\ paper\ we\ show\ that\ ...\text{”})\ \}$$

The following is an example of a query:

$$(AUTHOR = \text{“}John\ Smith\text{”})\ \wedge\ (TITLE \sqsupseteq P2P \wedge (information \prec_{[0,0]} alert)) \wedge (ABSTRACT \sim_{0.7} \text{“}P2P\ architectures\ have\ been...\text{”})$$

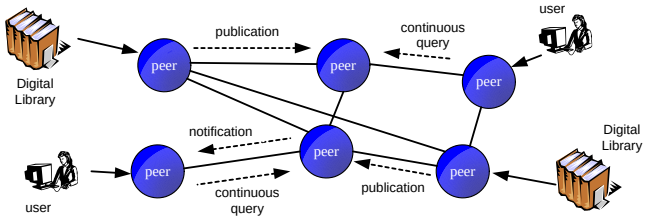

Fig. 1. Distributed resource sharing in future DLs

This query requests resources that have *John Smith* as their author, and their title contains the word *P2P* and a word pattern where the word *information* is immediately followed by the word *alert*. Additionally, the resources should have an abstract similar to the text value *"P2P architectures have been ..."* with similarity greater than 0.7. The contributions of this abstract are the following.

The research presented in this abstract is a continuation of our previous work on the DL information alert architecture DIAS [2] and the system P2P-DIET [1]. The main contribution of our current work is that our protocols are extensions of DHTs and achieve much better scalability, robustness, fault-tolerance and load balancing.

The rest of this extended abstract is as follows. Section 2 gives some details of the DHTrie protocols and Section 3 discusses very briefly our experimental evaluation of DHTrie.

2 The DHTrie Protocols

We implement pub/sub functionality by a set of protocols called the *DHTrie protocols* (from the words DHT and trie). The DHTrie protocols use *two levels of indexing* to store queries submitted by clients. The first level corresponds to the partitioning of the global query index to different nodes using DHTs as the underlying infrastructure. Each node is responsible for a fraction of the submitted user queries through a mapping of attribute values to node identifiers. The DHT infrastructure is used to define the mapping scheme and also manages the routing of messages between different nodes. We use an extension of the Chord DHT [4] to implement our network. The set of protocols that regulate node interactions are described in the next sections.

The second level of our indexing mechanism is managed locally by each node and is used for indexing the user queries the node is responsible for. In this level, each node uses a hash table to index all the atomic queries contained in a query by using their attribute name as the key. For each atomic Boolean query the hash table points to a *trie*-like structure that exploits *common words* and a hash table that indexes text values in equalities as in [6]. Additionally for atomic VSM queries an inverted index for the most "significant" query words is used as in [7].

In this abstract we will focus on the first level of indexing and present the subscription, publication and notification protocols that regulate node interactions. Protocols for query updating and removal are omitted due to space. The local

indexing algorithms we use and their experimental evaluation are thoroughly discussed in [6, 7].

2.1 The Subscription Protocol

Let us assume that a node P wants to submit a query q of the form:

$$\bigwedge_{i=1}^{m} A_i = s_i \ \wedge \ \bigwedge_{i=m+1}^{n} A_i \sqsupseteq wp_i \ \wedge \ \bigwedge_{i=n+1}^{k} A_i \sim_{a_i} s_i$$

To do so, P randomly selects a single word w contained in any of the text values $s_1, \ldots, s_m, s_{n+1}, \ldots, s_k$ or word patterns $wp_{m+1}, \ldots, wp_n$ and computes $H(w)$ to obtain the identifier of the node that will be responsible for query q. Then P creates message FwdQuery$(id(P), IP(P), qid(q), q)$, where $qid(q)$ is a unique query identifier assigned to q by P and $IP(P)$ is the IP address of P. This message is then forwarded in $O(logN)$ steps to the node with identifier $H(w)$ using the routing infrastructure of the DHT. Notice that $id(P)$ and $IP(P)$ need to be sent to the node that will store Q to facilitate notification delivery (see Section 2.3).

When a node P' receives a message FwdQuery containing q, it stores q using the second level of our indexing mechanism. P' uses a hash table to index all the atomic queries of q, using as key the attributes $A_1, \ldots, A_k$. To index each atomic query, three different data structures are also used: (i) a hash table for text values $s_1, \ldots, s_m$, (ii) a trie-like structure that exploits common words in word patterns $wp_{m+1}, \ldots, wp_n$, and (iii) an inverted index for the most "significant" words in text values $s_{n+1}, \ldots, s_k$. P' utilises these data structures at filtering time to find quickly all queries q that match an incoming publication p. This is done using an algorithm that combines algorithms BestFitTrie [6] and SQI [7].

2.2 The Publication Protocol

When a node P wants to publish a resource, it first constructs a publication $p = \{(A_1, s_1), (A_2, s_2), \ldots, (A_n, s_n)\}$ (the resource description). Let $D_1, \ldots, D_n$ be the sets of *distinct* words in $s_1, \ldots, s_n$. Then publication p is sent to *all* nodes with identifiers in the list $L = \{H(w_j) : \ w_j \in D_1 \cup \cdots \cup D_n\}$. The subscription protocol guarantees that L is a superset of the set of identifiers responsible for queries that match p.

The propagation of publication p in the DHT proceeds as follows. P removes duplicates from L and sorts it in ascending order clockwise starting from $id(P)$. This way we obtain less identifiers than the distinct words in $D_1 \cup \cdots \cup D_n$, since a node may be responsible for more than one words contained in the document. Having obtained L, P creates a message FwdResource$(id(P), pid(p), p, L)$, where $pid(p)$ is a unique metadata identifier assigned to p by P, and sends it to node with identifier equal to $head(L)$ (the first element of L). This forwarding is done by the following *recursive* method: message FwdResource is sent to a node P', where $id(P')$ is the greatest identifier contained in the finger table of P, for which $id(P') \leq head(L)$ holds.

Upon reception of a message FwdResource by a node P, $head(L)$ is checked. If $id(P) < head(L)$ then P just forwards the message as described in the previous paragraph. If $id(P) \geq head(L)$ then P makes a copy of the message, since this means that P is one of the intended recipients contained in list L (in other

words P is responsible for key $head(L)$). Subsequently the publication part of this message is matched with the node's local query database using the algorithm mentioned in Section 2.1 and the appropriate subscribers are notified (see Section 2.3). Additionally list L is modified to L' in the following way. P deletes all elements of L that are smaller than $id(P)$ starting from $head(L)$, since all these elements have P as their intended recipient. In the new list L' that results from these deletions we have that $id(P) < head(L')$. This happens because in the general case L may contain more than one node identifiers that are managed by P (these identifiers are all located in ascending order at the beginning of L). Finally, P forwards the message to node with identifier $head(L')$.

2.3 The Notification Protocol

When a message FwdResource containing a publication p of a resource arrives at a node P, the queries matching p are found by utilising its local index structures and using the algorithms briefly described in Section 2.1.

Once all the matching queries have been retrieved from the database, P creates notification messages of the form QNotification($l(r)$) and contacts all the nodes that their queries where matched against p using their IP address associated with the query they submitted. If a node P' is not online when P tries to notify it about the published resource, the notification message is sent to the $successor(P')$. In this way P' will be notified the next time it logs on the network. The modifications to the join and leave protocols of Chord to achieve this functionality originally presented in the non-DHT system P2P-DIET [1] are omitted due to space considerations. To utilise the network in a more efficient way, notifications can also be batched and sent to the subscribers when traffic is expected to be low.

2.4 Frequency Cache

In this section we introduce an additional routing table that is maintained in each node. This table, called *frequency cache (FCache)*, is used to reduce the cost of publishing a resource by storing the IP addresses of the nodes responsible for frequent words contained in published documents. FCache is a hash table used to associate each word that appears in a published document with a node IP address. FCache uses a word w as a key, and each FCache entry is a data structure that holds an IP address. Thus, whenever P needs to contact another node P' that is responsible for queries containing w, it searches its FCache. If FCache contains an entry for w, P can directly contact P' using the IP stored in its FCache. If w is not contained in FCache, P uses the standard DHT lookup protocol to locate P' and stores contact information in FCache for further reference. Using FCache the cost of processing a published resource p is reduced to $O(v + (h - v) \log N)$, where v is the number of words of p contained in FCache.

FCache entries are populated as follows. Each time a resource p is published at a node P, P contacts the nodes responsible for storing queries with words contained in p, as we described in Section 2.2. After this process is over, P knows the contact information (namely the IP address) of those nodes, and stores it to FCache along with the word each node is responsible for. After that, for

each publication taking place at P, P maintains this routing information for the most frequent words contained in resources published to it. Notice that the construction and maintenance of FCache is based only on local information and that the only extra cost involved is FCache misses (which cost $O(logN)$ and the routing information discovered is also cached for further reference).

3 Brief Presentation of Experimental Results

We have evaluated DHTrie experimentally in a distributed digital library scenario with hundreds of thousands of nodes and millions of user profiles. For our experiments we used 10426 documents downloaded from CiteSeer and also used in [6]. The documents are research papers in the area of Neural Networks and we will refer to them as the NN corpus. Because no database of queries was available to us, our queries are synthetically generated by exploiting 2000 documents of the corpus. The remaining 8426 documents are used to generate publications.

Our experiments show that the DHTrie protocols are *scalable*: the number of messages it takes to publish a document and notify interested subscribers remains almost constant as the network grows. Moreover, the increase in message traffic shows little sensitivity to increase in document size. We demonstrate that simple data structures with only local information can make a big difference in a DHT environment: the routing table FCache manages to reduce network traffic by a factor of 4 in all the alternative methods we have studied.

Since probability distributions associated with publication and query elements are expected to be skewed in typical pub/sub scenarios, achieving a *balanced load* is an important problem. We have studied an important case of load balancing for DHTrie and present a new algorithm which is also applicable to the standard DHT look-up problem.

The details of our experiments appear in [5] and will be discussed in detail in our workshop presentation.

References

1. S. Idreos, M. Koubarakis, and C. Tryfonopoulos. P2P-DIET: An Extensible P2P Service that Unifies Ad-hoc and Continuous Querying in Super-Peer Networks. In *Proc. of SIGMOD*, 2004. Demo paper.
2. M. Koubarakis, T. Koutris, P. Raftopoulou, and C. Tryfonopoulos. Information Alert in Distributed Digital Libraries: The Models, Languages and Architecture of DIAS. In *Proc. of ECDL*, 2002.
3. J. Lu and J. Callan. Federated search of text-based digital libraries in hierarchical peer-to-peer networks. In *Proc. of ECIR*, 2005. To appear.
4. I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proc. of ACM SIGCOMM*, 2001.
5. C. Tryfonopoulos, S. Idreos, and M. Koubarakis. Publish/Subscribe Functionality in IR Environments using Structured Overlay Networks. Submitted to a conference.
6. C. Tryfonopoulos, M. Koubarakis, and Y. Drougas. Filtering Algorithms for Information Retrieval Models with Named Attributes and Proximity Operators. In *Proc. of ACM SIGIR*, 2004.
7. T.W. Yan and H. Garcia-Molina. The SIFT information dissemination system. *ACM TODS*, 24(4):529–565, 1999.