Community based Ranking in Peer-to-Peer Networks

Christoph Tempich[1], Alexander Löser[2], and Jörg Heizmann[1]

[1] AIFB, University of Karlsruhe, 76128 Karlsruhe, Germany
{tempich,johi}@aifb.uni-karlsruhe.de
[2] CIS, University of Technology Berlin, Einsteinufer 17, 10587 Berlin, Germany
aloeser@cs.tu-berlin.de

Abstract. We address the problem of efficiently ranking the best peers w.r.t. a query with multiple, equally weighted predicates – conjunctive queries – in shortcut overlay networks. This problem occurs when routing queries in unstructured peer-to-peer networks, such as in peer-to-peer information retrieval applications. Requirements for this task include, e.g., full autonomy of peers as well as full control over own resources. Therefore prominent resource location and query routing schemes such as distributed hash tables can not be applied in this setting. In order to tackle these requirements we combine a new resource location and query routing approach that exploits social metaphors of topical experts and experts' experts with standard IR measures for ranking peers based on collection-wide information. The approach has been fully tested in simulation runs for skewed data distributions and network churn and has been partially implemented in the Bibster system [3].

1 Introduction

Finding relevant information from a heterogeneous and distributed set of information resources is a longstanding problem in computer science. Requirements for this task include, for example full autonomy of peers as well as full control over own resources. For this reason prominent resource location and query routing solutions such as distributed hash tables can not be applied in this setting.

Studies of social networks show that the challenge of finding relevant information may be reduced to asking the 'right' people. 'The right people' generally are the ones who either have the desired piece of information and can directly provide the relevant content or the ones who can recommend 'the right people'. Milgram's [20] and Kleinberg's [17] experiments illustrated that people with only local knowledge of the network (i.e. their immediate acquaintances) are quite successful in constructing acquaintance chains of short length leading to 'small world' networks. We observe that such mechanisms in social networks work although

- people may not always be available to respond to requests
- people may shift their interests and attention
- people may not have exactly the 'right' knowledge w.r.t. a particular information request, but only knowledge which is *semantically close* to it.

[3] http://bibster.semanticweb.org

This means that the real-world social networks, as opposed to theoretical network models, are *highly dynamic* w.r.t. peer availability and topic expertise. Therefore real world peer-to-peer scenarios need a concept of *semantic similarity* in order to efficiently answer the information needs of the peers, i.e. to realistically determine 'the right person'. Starting from requirements of semantic search in the setting of distributed, autonomous information sources, in which documents are annotated using Thesaurus-like or Description Logic-like ontologies, we have applied these observations and conceived the INGA algorithm. In this novel peer-to-peer algorithm each peer plays the role of a person in a social network. The novel design principle of INGA lies in the dynamic adaptation of the network topology, adaptation which is driven by the history of successful or semantically similar queries. We realize this in that we bound the local shortcut indices storing semantically labeled shortcuts, and a dynamic shortcut selection strategy. It forwards queries to a community of peers that are likely to best answer them. Shortcuts connect peers that share similar interests and thus spontaneously form semantic communities. Based on local shortcut indexes we select the k-best peers for each query to forward it depending exclusively on collection wide information (CWI).

Contributions and Paper Organisation: Former techniques used gossiping to disseminate a summary of the documents to update local indexes of a peer, an approach with severe scalability short comings. To avoid such drawbacks we create local shortcut indices by observing query and answers. We present new shortcut creation and deletion strategies that cluster peers within semantic communities. Rather than disseminating a query across the network we route queries within such communities to only the best matching peers. Our peer selection is bases on an IR-ranking strategy [9]. We modify this strategy so that it is able to select the top-k peers from a local shortcut index for conjunctive queries. Our extensive simulations using ontologies for Description Logics and thesaurus information show that our approach selects peers and forwards queries with high efficiency even in a highly dynamic setting and with bounded indices.

Our paper is organized as follows: In the next section we review related work, mainly information retrieval techniques and work on shortcut networks. We describe the social metaphors, our infrastructure to maintain the index and our query and results model in Section 3. Section 4 shows the index structure and update strategy for each type of shortcut. Section 5 presents our dynamic routing model. Section 6 describes our simulation methodology and the results of our simulations. Section 7 summarizes the results and points to future work.

2 Related work

Our approach combines information retrieval (IR) techniques with central indices and work on shortcut networks. We provide a brief overview of both areas in order to situate our contribution. Prior research on distributed information retrieval and meta search engines has addressed the ranking of data sources and the reconciliation of search results from different sources. GlOSS [13] and CORI [4] are among the most prominent distributed IR systems, but neither of them aimed at very large-scale, highly dynamic, self-organizing P2P environments, which were not an issue at the time these systems were developed.

Top-k queries [10] delivering a well-defined set of k best answers according to a user-provided, probably weighted compensation function, have shown their broad applicability, e.g. in content-based retrieval in multimedia collections or digital libraries. Algorithms for top-k retrieval [19, 11] in databases generally try to minimize the number of database objects that have to be accessed before being able to return a correct result set of the k best matching objects. Previous work in distributed top-k retrieval, however does not cover all the issues of high dynamics and autonomy which are unavoidable in a P2P network. Feasible information retrieval solutions need to cope with the following problems:

- Ranked Retrieval Model: In this setting the ranked retrieval models should be adopted in such a way that the models avoid flooding the entire network when selecting the top-k peers for a query with a conjunction of multiple predicates.
- Network Churn: Local index structures should provide all necessary information about what documents are available and should adapt to changing local documents of remote peers or peers joining or leaving the network.
- Collection-Wide Information: Queries should be answered only using collection-wide information, e.g. stored in local shortcut indices. Without constantly disseminating this information a peer should ideally restrict collecting information it is interested in in its index.

In the context of peer-to-peer networks only very few authors have explored retrieval algorithms which are based on collection-wide information for ranking peers in volatile networks. PlanetP [9] concentrates on peer-to-peer communities in unstructured networks with sizes up to ten thousand peers. They introduce two data structures for searching and ranking which create a replicated global index using gossiping algorithms. Each peer maintains an inverted index of its documents and spreads the term-to-peer index across the network. Inspired by the simple TFxIDF metric and based on the replicated index a simple ranking algorithm using the inverse peer frequency is implemented. However, by using gossiping to disseminate Bloom filters the system's scalability is severely limited. In section 5 we show how to apply the ranking strategy to our local shortcut indices to reduce the communication overhead for the dissemination of a peer's summary .

Local index information was first introduced by [7] to improve the efficiency of Gnutella routing indices. This indexing strategy locally stores information about specific queries and about peers which were successfully queried in the past. [22] first considers the semantics of the query to exploit interest-based locality in a static network. They use shortcuts that are generated after each successful query and are used to further requests, hence they are comparable to content provider shortcuts, which we introduce next. Their search strategy differs from ours, since they only follow a shortcut if it exactly matches a query, else they use a flooding approach. To update the index they use a LRU strategy, while we utilize also semantic similarity. In a similar way, [3] uses a local routing index for content provider shortcuts for the specific scenario of top k retrieval in P2P networks. Local indices are maintained in a static super-peer network. Their index policy considers temporal locality, each index entry has a certain time to live after which the shortcut has to be re-established for the next query on that topic. However, the emphasis in the approach is on appropriate topologies for overlay

networks. The paper develops efficient routing methods among static super-peers in a hypercube topology, while we consider dynamic networks without super-peers.

3 System Architecture

Our peer selection strategies described in section 4 can be integrated into any unstructured P2P network system. However, for evaluation purposes we used the SWAP infrastructure [15], which provides all standard peer-to-peer functionality such as information sharing, searching and publishing of resources.

3.1 Social Metaphors

In INGA, facts are stored and managed locally on each peer, thus constituting the 'topical knowledge' of the peer. A peer responds to a query by providing an answer matching the query or by forwarding it to whom it deems to be the most appropriate peers for this task. For the purpose of determining the most appropriate peers, each peer maintains a *personal semantic shortcut index*. The index is created and maintained in our highly dynamic setting in a lazy manner, i.e. by analyzing the queries that are initiated by users of the peer-to-peer network and that happen to pass through the peer. The personal semantic shortcut index maintained at each peer reflects that a peer may play the following four different roles for the other peers in the network (roles are listed in decreasing order of utility):

- The best peers to query are always those that already have answered it or a semantically similar one in the past successfully. We call such peers *content providers*.
- If no content providers are known, peers that have *issued semantically similar queries* in the past are queried. The assumption is that this peer has been successful in getting matching answers and now we can directly learn from it about suitable content providers. We call such peers *recommenders*.
- If we do not know either of the aforementioned items we query peers that have established a good social network to other ones over a variety of general domains. Such peers form a *bootstrapping network*.
- If we fail to discover any of the above we return to the default layer of neighboring peers. To avoid over-fitting to peers already known we occasionally select random peers for a query. We call this the *default network*.

From a local perspective, each peer maintains in its index information about some peers, about what roles these peers play for which topic and how useful they were in the past. From a global perspective, each of the four roles results in a network layer of peers that is independent from the other layers.

3.2 Building Blocks

We assume that each peer provides a unique peer identifier (PID). Similar to file sharing networks each peer may publish all resources from its *local content database*, so that

other peers can discover them by their requests (this also applies to resources downloaded from other peers). Information is wrapped as RDF statements and stored in an RDF repository[3]. Additionally to local meta data (e.g. *Robert Meersman isOrganizerOf ODBASE2005*) each resource is assigned one or more topics (such as *ODBASE2005 isTypeOf OTMConference*) and hierarchical information about the topics is stored *(OTM subTopicOf Conference)*. The topics a peer stores resources for are subsequently referred to as the peers own topics. Note, that our algorithm does not require a shared topic hierarchy, though this restriction provides certain advantages. In particular the ranking of shortcuts depends partly on the availability of a shared hierarchy. Therefore we use in all our experiments a shared hierarchy. For successful queries (own queries or those of other peers), which returned at least one match, the *shortcut management* extracts information about answering and forwarding peers to create, update or remove shortcuts in the *local shortcut index*. The *routing logic* selects 'most suitable' peers to forward a query to, for all own queries or queries forwarded from remote peers. The selection depends on the knowledge a peer has already acquired for the specific query and the similarity between the query and locally stored shortcuts.

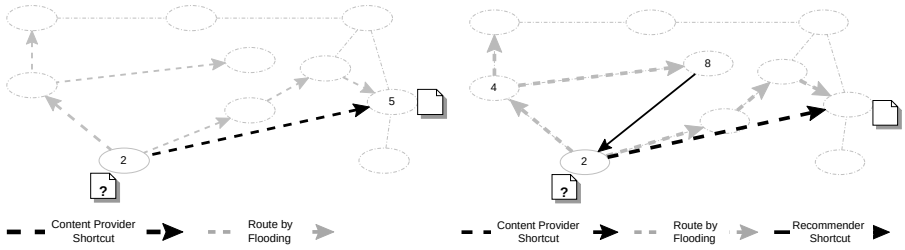
3.3 Query and Result Messages

We use a simple query message model which is similar to the structure of a Gnutella query message[12]. Each query message is a quadruple: $QM(q, b, mp, qid)$ where q is a SERQL query. In this paper we present ranking techniques for queries with multiple predicates such as: *Select all resources that belong to the topic semantic web and to the topic p2p*. In the Semantic Web context we formalize this query using common topic hierarchies, such as the Open Directory: *Find any resource with the topics /computer/web/semanticweb* $\wedge$ */computer/distributed/p2p* . More generally: q is a set of n predicates $q = (p_1...p_n)$. where each predicate is defined in a common ontology, b is the bootstrapping capability of the querying peer to allow the creation of bootstrapping shortcuts, mp the message path for each query message containing the unique PIDs of all peers, which have already received the query, to avoid duplicated query messages, and qid a unique query ID to ensure that a peer does not respond to a query it has already answered. Unique query IDs in INGA are computed by using a random number generator that has sufficiently high probability of generating unique numbers. A result message is a tuple: $RM(r, mp, qid)$ where r represents the answer to the query. We just consider results which exactly match the query. Besides, the message path mp is copied to the answer message to allow the creation of recommender and content provider shortcuts.

4 Building and Maintenance of the Index

Each peer is connected to a set of other peers in the network via uni-directional shortcuts. Hence, each peer can locally select all other peers it wants to be linked to. Following the social metaphors in section 1, we generally distinguish between the following types of shortcuts:

[3] `http://www.openrdf.org/`



(a) Content provider shortcut creation (b) Recommender shortcut creation

Fig. 1. Topic specific shortcut creation

4.1 Content Provider and Recommender Shortcuts

Content Provider Layer: The design of the content provider shortcut overlay departs from existing work as published by [22, 23] and exploits the simple, yet powerful principle of interest-based locality. When a peer joins the system, it may not have any information about the interest of other peers. It first attempts to receive answers for its queries by exploiting lower layers of the INGA peer network, e.g. by flooding. The lookup returns a set of peers that store documents for the topic of the query. These peers are potential candidates to be added to the content provider shortcut list. Each time the querying peer receives an answer from a remote peer, content provider shortcuts *sc* for each query topic to new remote peers are added to the list in the form: *sc(*$topic_i$*, pid, query hits,'c', update)*, where $topic_i$ is one of the topics taken from the query message and *pid* is the unique identifier of the answering peer. *Query hits* total the number of returned statements from a remote peer for a particular topic. *'c'* is the type of content provider shortcuts and *update* is the time, when the shortcut was created or the last time, when the shortcut was used successfully. Subsequent queries of the local peer or of a remote peer are matched against the topic column of the content provider shortcut list. If a peer cannot find suitable shortcuts in the list, it issues a lookup through lower layers, and repeats the process for adding new shortcuts. For example consider Figure 1(a) and Table 1: Peer 2 discovers shortcuts for the conjunctive query $/computer/web/semanticweb \wedge /computer/distributed/p2p$ by flooding the default network with a maximum number of three hops (TTL) and creates content provider shortcuts to peer 5.

PID	Topic	Query Hits	Type	Update
5	/computer/web/semanticweb	300	C	2005:31:05:16:37:34
5	/computer/distributed/p2p	300	C	2005:31:05:16:37:34

Table 1. Content Provider Creation

Recommender Layer: To foster the learning process of recommender shortcuts, especially for new peers in the network, we consider the incoming queries that are routed through a peer. Similar to a content provider shortcut a recommender shortcut *sc(*$topic_i$*,pid,query hits maxsim,rp, update)* is created for each topic in the query. The *PID* of a shortcut is extracted from the query message as the PID of the querying peer. Since we will not get any information about the number of results retrieved for the query, we set the number of query hits to 1. Finally, *r* indicates the type of shortcut for passive recommender shortcut and *update* is the time, when the shortcut was created or the last time, when the shortcut was used successfully. For example consider again Figure 1(b). Peer 2 issues the query $/computer/web/semanticweb \wedge /computer/distributed/p2p$. Peer 8 creates a shortcut to peer 2 since this query was routed through peer 8 as shown in table 2.

PID	Topic	Query Hits	Type	Update
2	/computer/web/semanticweb	1	R	2005:31:05:16:37:34
2	/computer/distributed/p2p	1	R	2005:31:05:16:37:34

Table 2. Recommender Shortcut Creation

Content Provider and Recommender Index: We assume that each peer may only store a limited amount of shortcuts, hence it only knows a limited set of topic specific neighbors it can route a query to. If the local index size is reached a peer has to decide which shortcut should be deleted from the index. For each shortcut in the index we compute a rank based on the following types of localities:

Semantic locality We measure the maximum semantic similarity $maxsim$ between the topic of a shortcut and the topics represented by the local content of a peer according to equation 1. Hence, we retain a shortcut about topic t to a remote peer, if t is close to our own interests. We define the similarity function $sim : t_1 \times t_2 \rightarrow [0; 1]$ between two terms in the same topic hierarchy, according to [18]:

$$sim_{Topic}(t_1, t_2) = \begin{cases} e^{-\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} & \text{if } t_1 \neq t_2 \\ 1 & \text{otherwise} \end{cases} \quad (1)$$

where l is the length of the shortest path between t_1 and t_2 in the graph spanned by the sub-topic relation and h is the minimum level in the topic hierarchy of either t_1 or t_2. α and β are parameters scaling the contribution of shortest path length l and depth h, respectively. Based on the benchmark data set given in [18], we chose $\alpha = 0.2$ and $\beta = 0.6$ as optimal values.

LRU locality To adapt content and interests changes we use the LRU replacement policy [2]. Shortcuts that have been used recently receive a higher rank. Each local shortcut is marked with a timestamp indicating when it was created. The timestamp will be updated, if the shortcut will be used successfully by the local peer. There is thus an 'oldest' and 'latest' shortcut. The value $update \in [0..1]$ is normalized by

difference between the shortcut's timestamp and the 'oldest' time stamp divided by the difference between the 'latest' and the 'oldest'.

Community locality We measure how close a shortcut leads us to a document. Content provider shortcuts, marked with a c, provide a one hop distance, therefore we set $type = 1$. Recommender shortcuts, marked with a r require at least two hops to reach a peer with relevant documents. In this case we set $type = 0.5$.

We weight the localities using a weighted moving average and compute the index relevance according to equation 2.

$$relevance = \frac{a * maxsim + b * type + c * update}{a + b + c} \tag{2}$$

Shortcuts with the highest relevance are ranked at the top of the index, while peers with a lower relevance are deleted from the index.

4.2 Bootstrapping Shortcuts

Bootstrapping shortcuts link to peers that have established many shortcuts for different query topics to many remote peers. We determine the bootstrapping capability by analyzing the in-degree and out-degree of a peer. We use the out-degree as a measure of how successful a peer discovers other peers by querying. To weigh the out-degree we measure the amount of distinct sources a peer receives queries from. We use the in-degree as a measure, that such a peer may share prestigious shortcuts with a high availability. By routing a query along bootstrapping shortcuts, we foster the probability to find a matching shortcut for a query and avoid the drawbacks of having to select peers randomly, e.g., by flooding.

Discovery and Update: Each incoming query that is stored in our index includes the bootstrapping information of the querying peer. While a peer is online it continuously updates its content/recommender index based on incoming queries and stores additional bootstrapping shortcuts in the form *sc(pid, bo)*, where *pid* is the PID of the querying peer and *bo* its bootstrapping capability. Once an initial set of bootstrapping nodes is found, a peer may route its queries to the nodes with the highest *bo* value. The *bo* value is calculated using equation 3

$$Bo = (1 + |outdegree|) \times (1 + |indegree|) \tag{3}$$

where *out-degree* is the number of distinct remote peers one knows. To compute an approximation of the *in-degree* without any central server we count the number of distinct peers that send a query via one's peer. To do this from the message path of indexed recommender shortcuts we examine the pen-ultimate peers. The number of distinct pen-ultimate peers denotes one's in-degree. To avoid zero values we limited the minimum for both values to 1.

4.3 Default Network Shortcuts

When a new peer enters the network, it has not yet stored any specific shortcuts in its index. Default network shortcuts connect each peer p to a set of other peers (p's neighbors) chosen at random, as in typical Gnutella-like networks (e.g., using rendezvous techniques).

5 Dynamic Shortcut Selection

The basic principle of shortcuts consists of dynamically adapting the topology of the P2P network so that the peers that share common interests spontaneously form well-connected semantic communities. It has been shown that users are generally interested in only some specific types of content[8]. Therefore being part of a community that shares common interests is likely to increase search efficiency and query success rate. To build semantic communities, for each query INGA executes the following steps:
Across the network: Selecting top-k peers: Whenever a peer receives a query message, it first extracts meta-information about the querying peer and updates its index if needed. Then our forwarding strategy is invoked to select a set of *k* peers which appear most promising to answer the query successfully. Finally, the original query message is forwarded to these *k* peers.
Across the network: Answering Queries: When a peer receives a query, it will try to answer the query using local content. We only return non-empty, exact results and send them directly to the querying peer. If the maximum number of hops is not yet reached, the query is forwarded to a set of peers selected as above.
Locally: Receiving Results: A querying peer analyzes the message path of result item and the number of results to create or update local content provider and recommender shortcuts.

Community based top-k rank: A naive approach would route a query only to a peer that matches *all* predicates of the query using a simple exact match paradigm. Too specific query predicates under the exact match paradigm often lead to empty result sets. Therefore the notion of best matches and relative importance of predicates can be a good alternative to satisfy a user's information needs independently of the individual peer instances. To rank peers we use a measure called the *inverse peer frequency* (IPF), based on TFxIDF, a popular method for assigning term weights. This measure was first introduced by [9] in the PlanetP system to rank peers efficiently for multiple query terms and local indices that are built via gossiping bloom filter summaries. However to create local indices we use social metaphors and interest based locality. We calculate the rank R for a peer p for a query q using formula 4, where N represents the number of distinct peers in one's shortcut index, N_i represents the number of peers providing documents for topic t and q_i^p the query hits q per topic t of each peer N_i.

$$R_p(q) = \sum_{i=1}^{t} q_i^p * \log(1 + \frac{N}{N_i}) \tag{4}$$

Intuitively, this scheme gives peers that contain all terms in a query the highest ranking.

Dynamic peer selection algorithm: The task of the INGA shortcut selection algorithm *Dynamic* is to determine best matching candidates to which a query should be forwarded. We rely on forwarding strategies, depending on the local knowledge for the topic of the query a peer has acquired yet in its index:

- We only forward a query via its *k best matching* shortcuts.
- We try to select content and recommender shortcuts before selecting bootstrapping and default network shortcuts.
- To avoid overfitting queries are also randomly forwarded to selected remote peers.

Algorithm 1 Dynamic

Require: **Query** q, **int** k,
Ensure: $TTL_q < maxTTL$
1: $s \leftarrow$ *TopIPF(q,Content/RecommenderShortcuts,*(k))
2: **if** $(|s| < k)$ **then**
3: $s \leftarrow s$ + *TopBoot(BootstrappingShortcuts,*$(k - |s|)$)
4: **end if**
5: $s \leftarrow$ *RandomFill(s,defaultNetworkShortcuts,f,k)*
6: **Return** s.

In step 1 of algorithm *Dynamic* method *TopIPF* browses through the index of all content and recommender shortcuts and identifies the k peers with the highest $R_p(q)$. If less then k peers are found we select peers from the top bootstrapping shortcuts (step 3) in subroutine *TopBoot*. It works similarly to *TopIPF*, but selects the peers with highest bootstrapping capability from the index. It also avoids overlapping peers within the set of selected shortcuts. Finally, in subroutine *RandomFill* we fill up the remaining peers randomly from the default network and return the set of selected peers. The algorithm's task is twofold: if the other subroutines fail to discover k peers for a query, it fills up remaining peers until k is reached. The second task of the algorithm is to contribute with some randomly chosen peers to the selected set of k peers to avoid overfitting of the selection process as known from simulated annealing techniques [16]. The *Dynamic* algorithm terminates if the query has reached its maximum number of hops.

6 Experimental Evaluation

Our approach is partially implemented in the Bibster system, which was evaluated in a real world case study [14]. During the case study a maximum of 50 researchers were simultaneously online. This number is too small to evaluate the scalability of a peer-to-peer routing algorithm. Although we have collected information about the content and the queries the peers have shared and submitted, again the amount of available information was too small to test our algorithm for a larger number of peers. Therefore, we have opted for simulating a peer-to-peer network with more than 1.000 peers and to generate

data sets considering different types of ontologies and content distributions.[4]In particular, we have created two synthetic data sets to evaluate our approach. The data sets are distributed among the peers. From the data set we further generated candidate queries which can be submitted by the peers. Before we present the results of our evaluation the distribution parameters to generate the data sets are described.

6.1 Simulation setup

Content: We have generated two ontologies with instances. The generation of the ontologies is based on the observation in [24] and [5]. In [24] it was observed that ontologies publicly available[5] follow broadly three different design characteristics while the distinguishing factors are the number of subclasses of a class, the number of restrictions and the number of properties. The three types of ontologies can be described as (1) Thesaurus-like, (2) Database-like and (3) Description Logic-like structures. The first one defines a large hierarchy with many classes and few restrictions and properties per class. Database-like ontologies are characterized by defining a medium number of primitive classes and equal number of restrictions per class. Finally, Description Logic-like ontologies have a high number of restrictions and properties per class and only a small number of primitive classes.

	Thesaurus-like		Description logic-like	
	No.	Distribution	No.	Distribution
$NoOfClasses$	1.000		100	
$NoSubClasses$	$\overline{\ln(1000)} = 7$	Zipf(1.1)	$\overline{\ln(100)} = 5$	Zipf(1.1)
$TotalNoProperties$	357		213	
$NoProperties$	0–5	Zipf(2.5)	0–7	Zipf(0.9)
$NoOfInstances$	200.000	Zipf(1)	200.000	Zipf(1)
$TotalNoPropertyInstances$	35.700	Zipf(1)	21.300	Zipf(1)

Table 3. Parameter setting for ontology types

We have created two synthetic ontologies representing the two extreme types of ontologies namely a Thesaurus-like and a Description Logic-like ontology. The parameters to create the ontologies were set according to the observations made in [24]. Unfortunately, in [24] no information about instance data is available. Therefore, we use the content distribution model of [5] to generate instances of the ontology. The distribution of content in our Bibster case study is comparable to the generated content distribution.

For the Thesaurus-like ontology we generated $NoOfClasses = 1.000$ classes[6]. Each class was assigned a popularity following a Zipf distribution with parameter set to

[4] According to our knowledge this is the first approach to create a synthetic data set for semantics based peer-to-peer systems. To create the data set we combined results from different analysis regarding distribution parameters for ontologies, content, content distribution and queries. The data set is available online at `http://ontoware.org/projects/swapsim/`.

[5] The analysis was based on the ontologies available in the DAML ontology library.

[6] Our algorithm was first conceived for topics organized in a topic hierarchy. In order to apply it to the generated ontologies we interpret classes as topics and instances as documents.

1. The popularity index is later used to generate the ranges of properties and instances. The classes were then arranged in a sub-class of hierarchy. The maximum depth of the hierarchy was set to $MaxDepth = \ln(NoOfClasses)$. Each class could have a maximum of $NoSubClasses = \ln(NoOfClasses)$ and a minimum of $NoSubClasses = 0$. The number of sub-classes for each class follows a Zipf distribution with parameter set to 1.1. The sub-classes are chosen randomly from the classes not already modelled in the hierarchy.

Similar to the hierarchy generation each class can have a maximum of $NoProperties = 5$ and a minimum of $NoProperties = 0$ number of properties. The number of properties per class, or the number of properties a class is domain for, follows a Zipf distribution with a skew parameter set to 1.1. The ranges of the properties were selected from all classes according to their popularity. In total, the ontology contains $TotalNoProperties = 357$ properties.

We have instantiated the classes with $NoOfInstances = 200.000$ instances taking into account the popularity of the classes. We have further instantiated the properties with $TotalNoPropertyInstances = 100 * TotalNoProperties$ properties. Equally to the instantiation of the classes a property carries a popularity rank following a Zipf distribution with parameter set to 1. After selecting a property according to its popularity, we instantiated it using the instances of its domain and range. We selected the instances according to a Zipf distribution with parameter set to 1.

The parameters used for the Thesaurus-like and Description Logic-like ontologies are summarized in Table 3.

Content Distribution: In order to distribute the content over the peers we adopt the model presented in [8]. Generally, users are interested in a subset of the content available in a peer-to-peer network. Furthermore, they are only interested in a limited number of topics (e.g., databases, Description Logic). Moreover, they are more interested in some classes while less in others. A peer can be interested in and have content for $ClassesOfInterest = \ln NoOfClasses * 2$ classes. The number of classes a peer is interested in is chosen randomly from a uniform distribution. The classes it is interested in are selected randomly from the available classes taking into account their popularity. As observed in [1] not all users share the same amount of data. Most of them do not contribute anything to the available knowledge while a small proportion makes two thirds of the knowledge available. Based on the study in [1] we assigned the following storage capacity to the peers in the network: 70% of the peers do not share any instances (free-riders); 20% share 100 instances or less; 7% share between 101 and 1000 instances; finally, 3% of the peers share between 1001 and 2000 instances (actual storage capacities are chosen uniformly at random). A peer sharing an instance knows all its properties and the type of the range instance.

Query Set: In order to test our algorithm we have generated two different types of queries. All queries ask for instances satisfying a varying number of constraints. The first query type instantiates the blueprint $(instance; isTypeOf; class) \wedge (instance; property; instance2) \wedge (instance2; isTypeOf; class2)$. We thus query for all $instance$s of a certain $class$ with the constraint that the instance has one particular $property$ pointing to another instance. The range of the $property$ determines the type of $instance2$.

The second query type extends the first by adding a constraint on the properties and instantiates the blueprint $(instance; isTypeOf; class) \wedge (instance; property; instance2) \wedge (instance2;\ isTypeOf; class2)\ \wedge\ (instance; property2; instance3) \wedge\ (instance3; isTypeOf; class3)$. We have summarized the number of distinct queries for the two ontology types in table 4. We have only generated queries to which at least one answer exists.

Query Type	Description	Ontology Type	
		Thesaurus-like	Description logic-like
QT1	1 class and 1 property	2194	8493
QT2	1 class and 2 properties	1087	8502

Table 4. Number of queries for the ontology types

Queries are chosen randomly from all available queries. We choose a uniform query distribution instead of a ZIPF-distribution, which is typically observed in file sharing networks [21]. This simulates the worst case scenario, in which we do not take advantage of frequent queries for popular topics.

Gnutella style network: The simulation is initialized with a network topology which resembles the small world properties of file sharing networks[7]. We simulated 1024 peers. In the simulation, peers were chosen randomly and they were given a randomly selected query to question the remote peers in the network. The peers decide on the basis of their local shortcut which remote peers to send the query to. Each peer uses INGA to select up to $pmax = 2$ peers to send the query to. Each query was forwarded until the maximum number of hops $hmax = 6$ was reached.

Volatile network and interest shifts: We implemented the dynamic network model observed for Gnutella networks of [21]: 60% of the peers have a availability of less then 20%, while 20% of the peers are available between 20 and 60% and 20 % are available more then 60%. Hence, only a small fraction of peers is available more than half of the simulation time, while the majority of the peers is only online for a fraction of the simulation time. Users' interest may change over time, e.g. to account for different search goals. To simulate changing interests, after 15 queries, equal to approx. 15.000 queries over all peers, each peer queries a complete different set of topics.

Evaluation Measures: We measure the search efficiency using the following metrics:

- **Recall** is a standard measure in information retrieval. In our setting, it describes the proportion between all relevant documents in peer network and the retrieved ones.
- **Messages** represent the required search costs per query that can be used to indirectly justify the system scalability.
- **Message Gain** To compare the recall per message we introduce the message gain, i.e. the proportion of messages with respect to the achieved recall. We define the

[7] We used the Colt library `http://nicewww.cern.ch/~hoschek/colt/`

message gain formally as

$$messagegain = \frac{Recall}{|Messages|} \tag{5}$$

6.2 Evaluation results

We have evaluated our approach with several parameter settings. To analyze the influencing variables. If not stated otherwise all simulations were performed using the Thesaurus-like ontology with the second Query Type. We used the Bootstrapping layer and the *TfxIPF* metric to combine the query hits for the shortcuts. Each peer can maintain a shortcut index with 40 shortcuts.

Shortcut indices enable efficient peer selection: It was already shown in [23] that shortcuts enable efficient query routing in unstructured peer-to-peer networks for single class queries. In figure 2(a) we have plotted the results comparing the recall for different selection strategies. We refer to the ranking based on equation 4 as *TfxIPF*. After a warm up phase of 2.000 queries, or approximately two queries per peer, we constantly reach around 75% of the available content. Not all peers are always online, thus we have plotted the maximum available content as *Online Available* in the graph. We compare our selection strategy to other available combination functions known in the database community and the *naive* approach. The naive approach represents the baseline for all algorithms likewise. In this approach we use only the default layer to select peers as in Gnutella. From the known peers on the default layer, we randomly select two remote peers to send and forward a query to.

The selection function described in [6] uses an equation similar to 6 to combine query hits in a distributed document retrieval scenario. We refer to this strategy as *Multiply*.

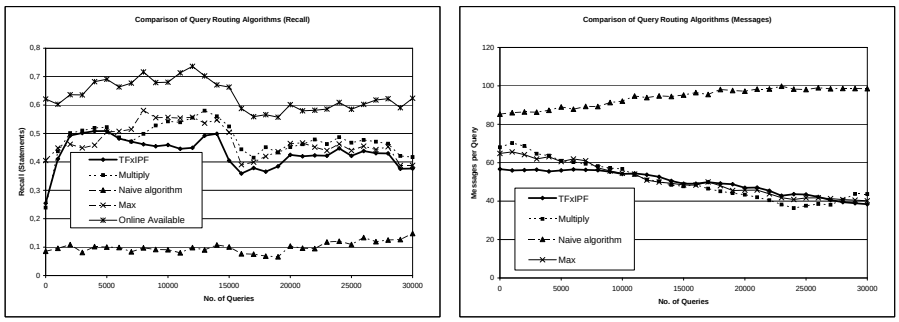
$$R_p(q) = \prod_{i=1}^{t} q_i^p \tag{6}$$

However, besides the query terms contained in the queries [6] also uses the most frequent words contained in the retrieved documents to create the index. This is not possible in our application. Furthermore, the total number of queries and total number of query hits are considered in the ranking function. These are optimizations which we have not considered so far, in order to be able to compare the different combination functions.

Additionally, we have evaluated equation 7 to select remote peers. The *Max* evaluation function selects the peer which has delivered the most statements for one of the query terms contained in the query.

$$R_p(q) = max(q_i^p) \tag{7}$$

Intriguingly, all three evaluation functions result in almost the same recall values. This is due to the high proportion of content delivered by a relatively small number of peers



(a) Recall (b) Messages

Fig. 2. Comparison Related Approaches: Dynamic Network 1024 Peers, 6 Hops, k=2

in typical peer-to-peer systems. We thus can conclude that shortcuts deliver an efficient yet simple way to boost peer selection quality.

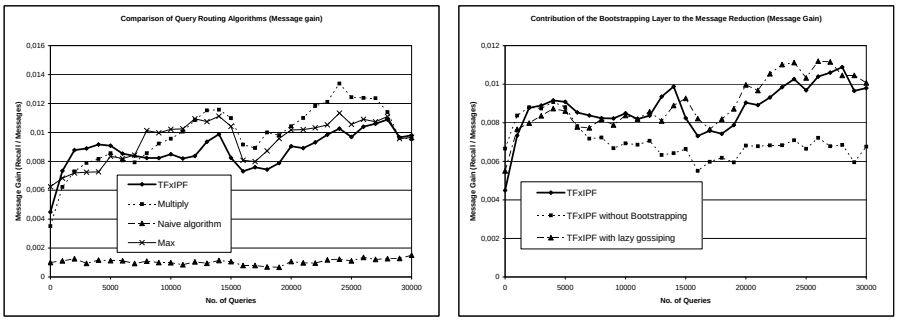
So far we have only considered the recall to evaluate our approach. In figure 2(b) we have plotted the number of messages produced to achieve this recall. The number of messages produced by the *Naive* approach slightly increases over time. Due to the high network churn the peers have to discover new remote peers since the available ones assigned in the setup phase are offline. Thus, the chances to send a query to a remote peer which has not received the query yet increase over time.

In contrast to the observation made for the *Naive* approaches the number of messages produced based on the shortcut selection decreases significantly. The number of messages decreases because different remote peers forward a query to the same peers. Each peer treats a query only once, thus double postings decrease the number of messages used. In figure 3(a) we have combined the two measurers for recall and messages and plotted the message gain. As expected the message gain increases because the recall stays at the same level and the number of messages decreases.

Bootstrapping decreases messages: In figure 3(b) we compare the performance of our algorithm with and without the bootstrapping layer. The bootstrapping metric favors remote peers which are well-connected and receive a high number of queries. In case a peer has no information about any of the classes used in the query it will forward the query to the best bootstrapping peers or - in case bootstrapping is not used - to randomly selected peers found on the network layer. As the bootstrapping capabilities of the peers are disseminated across the network, more peers select the same remote peers and thus the number of messages decreases. The recall is almost not affected by this decrease and thus the message gain increases.

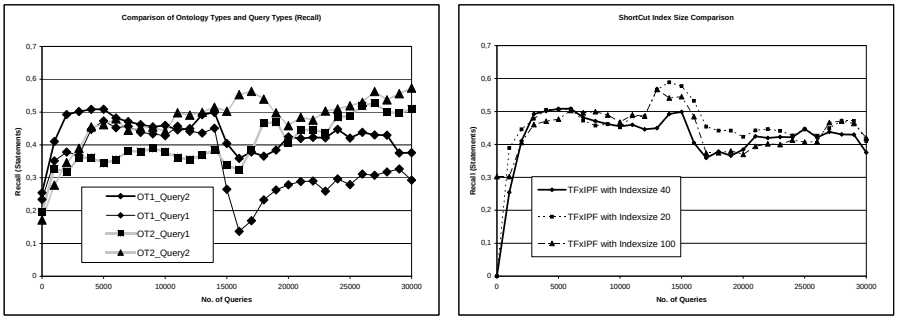
Gossiping does not increase the message gain: Gossiping is an established method to boost peer selection performance. We have implemented a lazy gossiping algorithm for comparison reasons. Each time a peer does not receive an answer to a query it sends around a discovery message to its neighbors on the network layer. The discovery message returns all classes a remote peer has content for. The result is treated like an incoming query, thus we create a number of recommender shortcuts for the returned

classes. We observe in figure 3(b) that gossiping is not advantageous in comparison to our standard selection algorithm.



(a) Related Approaches (b) Layer Contribution

Fig. 3. Comparison of Message Gain: Dynamic Network 1024 Peers, 6 Hops, k=2



(a) Ontology Types (b) Index Size

Fig. 4. Influencing variables: Dynamic Network 1024 Peers, 6 Hops, k=2

Performance of INGA is independent from the underlying ontology type: In figure 4(a) we compare the recall of INGA in case of changing Ontology Types and varying queries. We observe that the recall increases quickly using our approach independent of the underlying ontology and query type. However, when the queries change after 15.000 queries the recall for Query Type 1 and the Thesaurus-like ontology decrease strongly. We conclude that the high number of properties for the Description-logic style ontologies facilitates the quick emergence of an efficient shortcut network.

Performance of INGA is robust against varying index sizes: In figure 4(b) we have compared the recall of INGA with a varying index size. The size of the index determines the required resource allocation for routing purposes at the local peer. We have varied the index size starting from 20 shortcuts, to 40 and finally 100 shortcuts per peer. The examination of the results reveals, that INGA is very robust against the change of the index size.

7 Summary and Outlook

In this paper we present the first semantic query routing algorithm for a completely distributed, dynamic and unstructured peer-to-peer system. Our routing algorithm is based on the creation of local shortcuts and the maintenance of a small shortcut index. The shortcuts memorize the number of instances a remote peer has answered for a conjunctive query. While the local maintenance of the shortcut index allows for complete control over the usage of the shortcuts, no summary of the peers' content must be disseminated to other peers. In our extensive evaluations we show that our shortcut indices are efficient for peer selection in dynamic peer-to-peer networks.

We have introduced a novel metric to characterize well-connected peers, viz. bootstrapping peers. They maintain a high number of shortcuts and have many incoming connections. We provide evidence that bootstrapping is a good metric to reduce the number of messages. We have compared different metrics to combine local shortcuts and shown that the right index strategy and shortcut creation strategy are more important for efficient routing than right combination metrics. A small index size is sufficient to guarantee a high performance. Intriguingly, our comparisons show that routing based on shortcuts is as efficient as gossiping algorithms.

Our evaluations are based on the first available synthetic data set to setup the knowledge base of the peers. We were able to demonstrate that our approach is equally suitable for Description Logic-like ontologies as well as Thesaurus-like ontologies.

With the full support of conjunctive queries our algorithms are perfectly suitable for application around the social semantic desktop and other applications in which local control about available information is important. Current research in keyword based distributed document retrieval suggests that our shortcut update strategies can be further improved in the future. As more instantiated ontologies become available an evaluation of our approach with a real data set is desirable. An interesting additional problem is the generalization of our approach for a network with individual semantics on each peer. Peers within the same community may share their facts and possibly agree on a common set of semantics. Such a community search engine would enable flexible and efficient wide area knowledge sharing applications without the maintenance of central indexing servers or a static semantic structure.

Acknowledgement Research reported in this paper has been partially financed by EU in the IST project SEKT (IST-2003-506826). Alexander Löser was generously funded by the German Research Society, Berlin-Brandenburg School in Distributed Information Systems (DFG grant GRK 316/3).

References

1. E. Adar and B. Huberman. Free riding on gnutella. *First Monday*, 5(10), Oktober 2000.
2. A. V. Aho, P. J. Denning, and J. D. Ullman. Principles of optimal page replacement. *J. ACM*, 18(1):80–93, 1971.
3. W.-T. Balke, W. Nejdl, W. Siberski, and U. Thaden. Progressive distributed top-k retrieval in peer-to-peer networks. In *21st International Conference on Data Engineering (ICDE)*, Tokyo, Japan, 2005.

4. J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 21–28, New York, NY, USA, 1995. ACM Press.
5. V. Cholvi, P. Felber, and E. Biersack. Efficient search in unstructured peer-to-peer networks. *European Transactions on Telecommunications: Special Issue on P2P Networking and P2P Services*, 15(6):535–548, November 2004.
6. B. Cooper. Guiding queries to information sources with InfoBeacons. In *ACM/IFIP/USENIX 5th International Middleware Conference*, Toronto, 2004.
7. A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *International Conference on Distributed Computing Systems*, july 2002.
8. A. Crespo and H. Garcia-Molina. Semantic Overlay Networks for P2P Systems. Technical report, Computer Science Department, Stanford University, 2002.
9. F. M. Cuenca-Acuna, C. Peery, R. P. Martin, and T. D. Nguyen. PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities. In *Intern. Symp. on High-Performance Distributed Computing*, Seattle, USA, 2003.
10. R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *Symposium on Principles of Database Systems*, 2001.
11. U. Güntzer, W.-T. Balke, and W. Kießling. Optimizing multi-feature queries for image databases. In *Intern. Conf. on Very Large Databases*, Cairo, Egypt, 2000.
12. The gnutella developer forum, http://rfc-gnutella.sourceforge.net, 2004.
13. L. Gravano and H. García-Molina. Generalizing GlOSS to vector-space databases and broker hierarchies. In *International Conference on Very Large Databases, VLDB*, pages 78–89, 1995.
14. P. Haase, J. Broekstra, M. Ehrig, M. Menken, P. Mika, M. Plechawski, P. Pyszlak, B. Schnizler, R. Siebes, S. Staab, and C. Tempich. Bibster - a semantics-based bibliographic peer-to-peer system. In S. A. McIlraith, D. Plexousakis, and F. van Harmelen, editors, *Proceedings of the Third International Semantic Web Conference, Hiroshima, Japan, 2004*, volume 3298 of *LNCS*, pages 122–136. Springer, NOV 2004.
15. P. Haase et al. Bibster - a semantics-based bibliographic peer-to-peer system. In *Proc. of the 3rd International Semantic Web Conference, Japan*. Springer, 2004.
16. S. Kirkpatrick, J. C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. pages 606–615, 1987.
17. J. Kleinberg. Navigation in a small world. *Nature*, 406, 2000.
18. Y. Li, Z. Bandar, and D. McLean. An Approach for messuring semantic similarity between words using semantic multiple information sources. In *IEEE Transactions on Knowledge and Data Engineering*, volume 15, 2003.
19. A. Marian, N. Bruno, and L. Gravano. Evaluating top-k queries over web-accessible databases. *ACM Trans. Database Syst.*, 29(2):319–362, 2004.
20. S. Milgram. The small world problem. *Psychology Today*, 67(1), 1967.
21. S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. *Multimedia Systems*, 9(2), 2003.
22. K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient Content Location Using Interest Based Locality in Peer-to-Peer System. In *Infocom*. IEEE, 2003.
23. C. Tempich, S. Staab, and A. Wranik. REMINDIN:Semantic Query Routing in Peer-to-Peer Networks based on Social Metaphers. In *Proceedings of the 13th WWW Conference New York*. ACM, 2004.
24. C. Tempich and R. Volz. Towards a benchmark for Semantic Web reasoners - an analysis of the DAML ontology library. In Y. Sure, editor, *Proceedings of Evaluation of Ontology-based Tools (EON2003) at 2nd International Semantic Web Conference (ISWC 2003)*, pages 4–15, OCT 2003.