Searching Dynamic Communities with Personal Indexes

Alexander Löser[1], Christoph Tempich[3], Bastian Quilitz[1], Wolf-Tilo Balke[2], Steffen Staab[4], and Wolfgang Nejdl[2]

[1] CIS,University of Technology Berlin, Einsteinufer 17, 10587 Berlin, Germany
aloeser, baqui@cs.tu-berlin.de
[2] L3S, University of Hannover, 30167 Hannover,Germany
balke,nejdl@l3s.de
[3] AIFB, University of Karlsruhe 76128 Karlsruhe,Germany
tempich@aifb.uni-karlsruhe.de
[4] ISWeb, University of Koblenz Landau 56016 Koblenz, Germany
staab@uni-koblenz.de

Abstract. Often the challenge of finding relevant information is reduced to find the 'right' people who will answer our question. In this paper we present innovative algorithms called INGA (Interest-based Node Grouping Algorithms) which integrate personal routing indices into semantic query processing to boost performance. Similar to social networks peers in INGA cooperate to efficiently route queries for documents along adaptive shortcut-based overlays using only local, but semantically well chosen information. We propose active and passive shortcut creation strategies for index building and a novel algorithm to select the most promising content providers depending on each peer index with respect to the individual query. We quantify the benefit of our indexing strategy by extensive performance experiments in the SWAP simulation infrastructure. While obtaining high recall values compared to other state-of-the-art algorithms, we show that INGA improves recall and reduces the number of messages significantly.

1 Introduction

Finding relevant information from a heterogeneous set of information resources is a longstanding problem in computing. In everyday life we observe that there are successful strategies for finding relevant information in a social network of people. Studies of social networks show that the challenge of finding relevant information may be reduced to find 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 [15] and Kleinberg's [12] experiments illustrated that people with only local knowledge of the network (i.e. their immediate acquaintances) were quite successful at constructing acquaintance chains of short length, leading to 'small world' networks. In such a network, a query is forwarded along that outgoing link which takes it 'closest' to the destination. 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, but only knowledge which is *semantically close*.

I.e., the real-world social network is *highly dynamic* with regard to availability of peers and with regard to expertise about topics and it needs *semantic similarity* in order to determine ‘the right person’.

Inspired by these observations and focussed by the requirements of semantic search in the setting of distributed autonomous information sources, we have conceived INGA a novel peer-to-peer algorithm where each peer plays the role of a person in a social network. In INGA , facts are stored and managed locally on each peer constituting the ‘topical knowledge’ of the peer. A peer responds to a query be providing an answer matching the query or by forwarding the query to what he deems to be the most appropriate peers. 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 (in decreasing order of utility):

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

Seen 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. Seen from a global perspective, each of the four roles results in a network layer of peers that is independent from the other layers.

1.1 Related Work

The first approaches for efficient indexing in P2P architectures were central indices, that have to transmit either meta data about the available content to central indexing peers, like e.g. GlOSS [9] or *Napster*. One of today’s main technique for indexing P2P systems are so-called distributed hash tables (DHTs),(e.g. [1] or see [4] for a survey) that without need of a central index allows to route queries with certain keys to particular peers containing the desired data. But to provide this functionality all new content in the

network has to be published at the node for the respective key, if new data on a peer arrives or a new peer joins the network. And in case that a peer leaves the network the information about its content has to be unpublished. Recent research in [14] shows that due to the publishing/unpublishing overhead, DHTs lack efficiency when highly replicated items are requested and in practical settings perform even worse than flooding approaches degrading further if network churn is introduced.

While the visualization of keys and objects in the same name space used in structured overlays provides an elegant clean solution to routing within logarithmical bounds it comes at the significant cost of destroying the locality of the content: Content at a user's desktop is co-located with other relevant items, structured overlays destroy this locality meaning that enhanced opportunities for browsing and pre-fetching are lost [11]. Unstructured networks, such as Gnutella, keep this locality, since a query is forwarded to randomly picked neighbors. To bound the number of hops it can travel, each query is tagged with a maximum number of hops (TTL). In addition Gnutella employs a duplicate detection mechanism, so that peers do not forward queries that they have already previously forwarded. To improve the efficiency of Gnutella routing indices local index information are first introduced by [8]. This indexing strategy locally stores information about specific queries and what peers were successfully queried in the past. Edutella [16] combines a super-peer network with routing indices and an efficient broadcast. While its routing approach is efficient, especially when churn is high its performance suffers from (de-)registering complex semantics in the distributed indices. [18] 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. However their search strategy differs from ours, since they only follow a shortcut if it matches exact with a query, else they use a flooding approach. To update the index they use a LRU strategy. Similar, [5] 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 reestablished for the next query on that topic. REMINDIN [19] used a routing table storing content provider shortcuts and a relaxation based routing strategy. The approach was only designed for a static setting without any index size limitation, an assumption that is not realistic.

1.2 Contributions and Paper organisation

In this paper, we propose an improved shortcut selection strategy able to identify and group peers with similar interests efficiently in a dynamic setting. To our best knowledge, this is the first approach simulating volatile shortcut networks without any static peers. To adapt to the dynamics of the networks and to bound the local index we present an index update policy combining temporal, semantic and community locality. We show, that by indexing shortcuts linking only to a small fraction of peers we perform like an 'unlimited' index. To further boost performance and enhance recall in a dynamic setting we introduce in INGA two additional types of overlays, namely recommender and bootstrapping overlays. We have built a network simulator and conducted

extensive experiments under realistic conditions. Results show that INGA outperforms other state-of-the-art approaches significantly.

We describe the infrastructure to maintain the index and the semantic similarity function to select peers in section 2. Section 3 shows the index structure and update strategy for each type of shortcut. Section 4 presents our dynamic routing model. Section 5 describes our simulation methodology and the results of our simulations.

2 Basic Building Blocks of an INGA Peer

Our peer selection strategies described in section 3 are implementation independent. For evaluation purposes, though we use the SWAP infrastructure [10]. We recall that it provides all standard peer-to-peer functionality such as information sharing, searching and publishing of resources. Specifically it comprises the following main building blocks:

- The *network component's* task is to provide core network functionality, such as maintaining network connections to other peers and to provide a unique peer identifer (PID).
- Similar to file sharing networks each peer may publish all resources from its *local content database*, so other peers can discover them by its requests (this also applies to resources downloaded from other peers). All information is wrapped as RDF statements and stored in an RDF repository [5]. Additionally to local meta data (*MMusen isOrganizerOf ISWC2005*) each resource is assigned a topic *(ISWC2005 isTypeOf SemanticWebConference)* and hierarchical information about the topics is stored *(SemanticWebConference 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, although it is advantageous for it (*cf.* 2).
- For successful queries (own queries or those of other peers), which returned at least one match, the *shortcut management component* extracts information about answering and forwarding peers to create, update or remove shortcuts in the *local shortcut index*. Contrary to related approaches, such as DHTs, INGA peers only index 'egoistically', i.e. shortcuts on topics they requested themselves.
- 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.

For simplicity throughout this paper we will assume peers not to be malicious (i.e. they do not intentionally return false shortcuts); strategies for identifying malicious peers in overlay networks are e.g. given in [6].

Query and Result Messages. We use a simple query message model which is similar to the structure of a Gnutella query message. Each query message is a quadruple:

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

$QM(q, b, mp, qid)$ where q is a SERQL query (*cf.* footnote 5). We support any SERQL queries, however for routing purposes only the topic information is used. From a query for all *SemanticWebConference*s organized by *MMusen*, only *SemanticWebConference* is utilized for routing. 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.

Semantic Similarity Function. In case the peers in the network share a common topic hierarchy our routing algorithm uses not only exact index hits, but also exploits the semantic similarity between a query and an indexed shortcut. We define the similarity function between a query q and a shortcut sc, which are both given by query terms in the same topic hierarchy as $sim : q \times sc \rightarrow [0; 1]$. Such similarity metrics are often domain specific and depend on the query semantics. In our implementation we use a similarity metric for topic hierarchies proposed by [13] (but of course any other suitable similarity can be used):

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

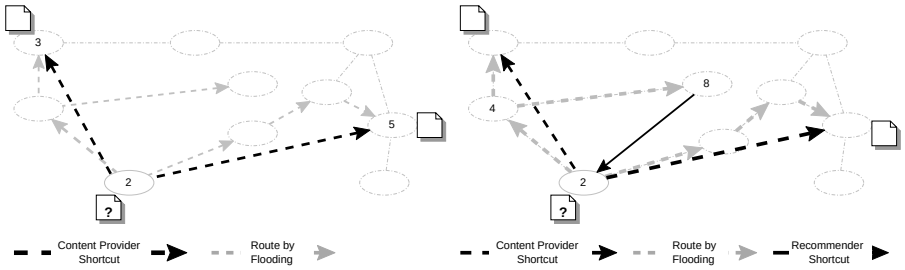
where l is the length of the shortest path between q and sc in the graph spanned by the sub topic relation and h is the minimal level in the topic hierarchy of either q or sc. α and β are parameters scaling the contribution of shortest path length l and depth h, respectively. Based on the benchmark data set given in [13], we chose $\alpha = 0.2$ and $\beta = 0.6$ as optimal values.

3 Building and Maintaining 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. However, due to limited local resources and each peer's specific interests, peers only maintain a bounded index of shortcuts . The decision of replacing a shortcut from the index, i.e. promoting new peers as shortcut acquaintances, depends on the history of the responses to previous requests issued by each peer. We now will propose index building and update strategies for each shortcut type that efficiently limit the index size to only the most useful shortcuts for each local peer. Following the social metaphors in section 1, we generally distinguish between four types of shortcuts.

3.1 Content Provider Layer

The design of the content provider shortcut overlay departs from existing work as published by [18], [19], or [7] and exploits the simple, yet powerful principle of interest-



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

Fig. 1. Topic specific shortcut creation

based locality. That means if a content provider peer has a particular piece of content that a peer is interested in, it can be considered very likely that the content provider will also have other interesting items for that peer.

Discovery and Creation. 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 to new remote peers are added to the list in the form: *sc(topic, pid, query hits,'c', update)*, where *topic* is the query terms taken from the query message, *pid* is the unique identifier of the answering peer, *query hits* is the number of returned statements, *'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 successful. The content provider shortcut list will grow with each submitted query until the maximum number of content provider peers is reached. 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. Consider figure 1(a). Peer 2 discovers shortcuts for the topic */Education/UML* by flooding the default network with a maximum number of hops (TTL) of three hops and creates two content provider shortcuts to peer 3 and peer 5.

3.2 Recommender Layer

Very active peers issue many successful queries and produce many shortcuts. If a remote peer issues many queries that are similar to one's own interests, it will be beneficial to establish links to this peer. The reason is that, if a remote peer has established a shortcut to an interesting content provider, it is likely that this peer will issue other queries on related topics that one will be interested in, too. Such recommender shortcuts thus represent a new kind of links in the semantic overlay structure. If a peer can not directly

determine a content provider peer for a given query, it can always forward the query to the best matching recommender.

Creation by controlled listening. To foster the learning process of recommender shortcuts, especially for new peers in the network, we consider the incoming queries that are routed through ones peer. A recommender shortcut *sc(topic,pid,query hits maxsim,r, update)* is created, where *topic* is the set of query terms from the query message. The *pid* for a respective shortcut is extracted from the query message as the ID of the querying peer. Since we will get no information about the number of results retrieved for the query, we set the number of $queryhits$ to 1. Finally *r* indicates the type of the 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. Consider again Figure 1(b). Peer 2 issues the query /Top/Education/UML. Peer 8 creates a shortcut to peer 2 since this query was routed through peer 8.

3.3 Content Provider and Recommender Index

We assume that each peer may only store a limited amount of shortcuts, hence 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.

LRU locality. To adapt to changes in the content and interests we use a LRU replacement policy [2]. Shortcuts that have been used recently receive a higher rank. Each local shortcut is marked with a time stamp when it was created. The time stamp will be updated, if the shortcut will be used successful by the local peer. There is thus an 'oldest' and 'latest' shortcut. The value $update \in [0..1]$ is calculated as difference between the shortcuts time stamp 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, we set $type = 1$. Recommender shortcuts, marked with a r require at least two hops to reach a peer with relevant documents, we set $type = 0.5$.

We weight the localities 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.

3.4 Bootstrapping Layer

Bootstrapping shortcuts link to peers that have established many shortcuts for different query topics to a lot of 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 weight 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 continually updates its bootstrapping index based on incoming queries and stores 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. One calculates its bo value using equation 3

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

where *out-degree* is the number of distinct remote peers one's 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 scrutinize 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 one.

3.5 Default Network Layer

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).

4 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. Therefore being part of a community that shares common interests is likely to increase search efficiency and success rate. To optimize the overall message traffic we will now propose a dynamic shortcut selection strategy, where each peer selects only a certain number *k* of most promising shortcuts for query forwarding. Then we will evaluate our approach against related approaches.

4.1 Overview

INGA consists of several steps executed locally and across the network when recommending peers for a query and retrieving or returning results. Consider a query posed to the P2P network. Necessary steps are:

Across the network:Recommending. Whenever a peer receives a query message, it first extracts meta-information about the querying peer and updates its bootstrapping and recommender 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 with local content. We only return non-empty, exact results and route 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. On the arrival of result items a querying peer analyzes the message path and the respective number of results to create or update local content provider and recommender shortcuts.

4.2 Selecting best matching shortcuts

The task of the INGA shortcut selection algorithm 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 and accommodate a little volatility (especially in the form of new joining peers), queries are also randomly forwarded to some peers.

The following algorithm shows the basic peer selection procedure: The algorithm works

Algorithm 1 Dynamic

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

as follows: in step 1 we select *k* peers from content or recommender shortcuts that match the topic of the query with the highest similarity. To avoid forwarding queries along

shortcuts with only low similarity we introduce a minimum similarity threshold t_{greedy}. If found less then *k* shortcuts we select the top bootstrapping shortcuts (step 3). Finally we fill the up remaining shortcuts randomly from the default network and return the set of selected shortcuts. The algorithm terminates if the query has reached its maximum number of hops. We will now show all subroutines for shortcut selection in more details. Algorithm *TopGreedy* allows for selecting the top peers above a similarity threshold. The algorithm browses trough the index of all topic dependent shortcuts (step 3) and identifies the most similar matching shortcuts for a query (step 4) above t_{greedy} (step 5). If two shortcuts have the same similarity, we choose the shortcut with the higher query hits value. The algorithm carefully selects the top-k peers for a query by avoiding different shortcuts with overlapping peers step (7-8).[6] The *TopBoot* Algorithm (omitted

Algorithm 2 TopGreedy

Require: **Query** *q*, **Set** *QueryDependentShortcuts*, **int** *k*, **int** t_{greedy}
1: *topShortcuts* $\leftarrow \{\}$
2: *s_tmp* $\leftarrow$ *QueryDependentShortcuts*
3: **while** (*s_tmp is* not empty) $\wedge$ ($k > 0$) **do**
4: *Next* $\leftarrow maxSim_{Topic}(q, s\_tmp)$
5: **if** sim_{Topic} (*q*,*Next*) $> t_{greedy}$ **then**
6: *s_tmp* $\leftarrow$ *s_tmp* $-$ (*Next*)
7: **if** (*Next* routes not to a peer in *topShortcuts*) **then**
8: *topShortcuts* $\leftarrow$ *topShortcuts* + *Next*
9: $k \leftarrow k - 1$
10: **end if**
11: **else**
12: break
13: **end if**
14: **end while**
15: **Return** *topShortcuts*

here due to space restrictions) works similar to the *TopGreedy* Algorithm, but selects the best peers with highest known bootstrapping capability. It also avoids overlapping peers within the set of selected shortcuts. The task of algorithm *RandomFill* is twofold: if the other subroutines fail to discover k peers for a query, it fills up remaining peers until k is reached (step 12-14). The second task of the algorithm is to contribute some randomly chosen peers to the selected set of k peers to avoid overfitting of the selection process as known from simulated annealing techniques. In step 2 the algorithm determines if new peers should be added to the already selected set, or if peers have to be exchanged. Depending on the probability f in step 6-7 the algorithm exchanges already selected peers with randomly chosen ones.

[6] Due to limited space details have been omitted.

Algorithm 3 RandomFill

Require: **Set** *preSelected*, **Set** *defaultNetWorkShortcuts*, **int** *f*, **int** *k*

Set *postSelected* $\leftarrow \{\}$
if ($\frac{k-|preSelected|}{k} < f$) **then**
  while (*preSelected is* not empty) **do**
    Next $\leftarrow$ *next(preSelected)*
    preSelected $\leftarrow$ *preSelected* $-$ (*Next*)
    if (*rand*(0,1) $> f$) **then**
      postSelected $\leftarrow$ *postSelected* + *Next*
    end if
  end while
end if
$k \leftarrow\ k - |postselected|$
while $k > 0$ **do**
  postSelected $\leftarrow$ *postSelected* + *next(defaultNetworkShortcuts*)
  $k \leftarrow k - 1$
end while
Return *postSelected*

5 Experimental Evaluation

Open Directory (DMOZ) as real world data set. We base our simulation framework on a data set of the open directory *DMOZ.org*, since it consists of realistic data about the content distribution among persons within a large community. For the topic distribution we select the 1657 topics in the first three levels of the DMOZ hierarchy that have one or more editors assigned to them. We represent one editor by one peer and assume that peers that are interested in a topic also store resources for this topic. We observed that editors are distributed with a heavily tailored Zipf popularity over the topics: 755 topics have 1 editor; 333 topics have 2 editors; 204 topics have 3 editors; . . . ; 44 topics have 6 editors; . . . ;14 topics have 10 editors ;1 topic has 32 editors. Furthermore some editors are interested in more than one topic. Again we observed a heavily tailored Zipf distribution: 991 editors only have one topic; 295 two topics; 128 three topics; ... one editor has 18 topics; one editor 20 topics and one editor has 22 topics.

Query Distribution. Queries are generated in the experiments by instantiating the blueprint $(*; isTypeOf; topic)$, with topics arbitrarily chosen from the set of topics that had at least one document. We generated 30000 queries, uniformly distributed over the 1657 different topics. We distribute the queries uniformly over the peers, hence each peer may issue a query to any topic and each topic is requested with the same probability. We choose a uniform query distribution instead of a ZIPF-distribution, which is typically observed in file sharing networks [17]. This simulates the worst case scenario, where we do not take advantage of often repeated 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.

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

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 short cut 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 maximal number of hops $hmax = 6$ was reached.

Volatile network and interest shifts. We implemented the dynamic network model observed for Gnutella networks of [17] in our simulation: 60% of the peers have a availability of less than 20%, while 20% of the peers are available between 20 and 60% and 20 % are available more than 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 a fraction of the simulation time. Users' interest may change over time, e.g. to account for different search goals [3]. To simulate the effect of changing interests in the network, after 15 queries, equal to ca. 15.000 queries over all peers, each peer will ask for a completely different set of topics.

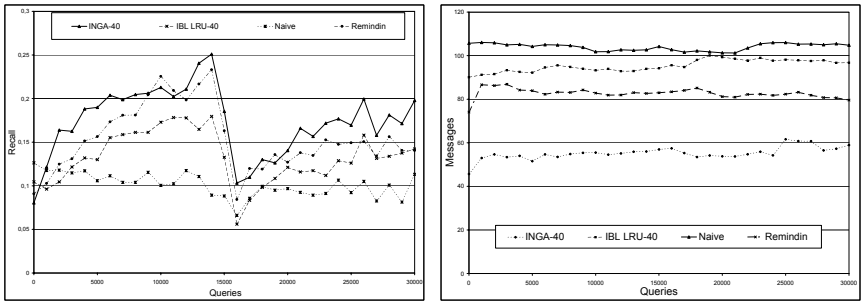
Evaluation Measures. We measure the search efficiency using the following metrics:

- **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** compares the recall per message, hence the proportion of messages with respect to to the achieved recall.

5.1 Comparing INGA with state-of-the-art approaches

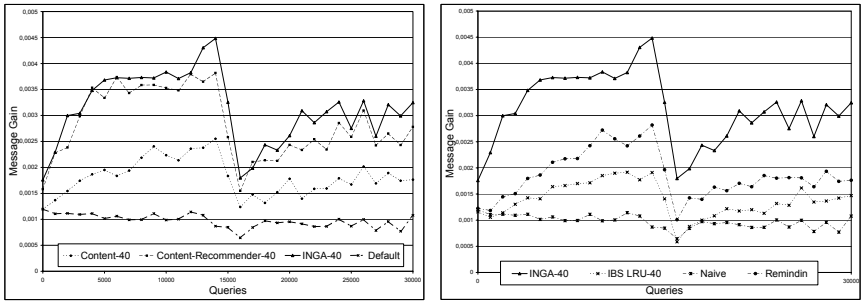
As a baseline we compare INGA with an index size of 40 entries against the interest based locality strategy (IBL) of [18] with an LRU strategy and an index size of 40 entries, the naive algorithm of Gnutella (Naive) and REMINDIN [19].

INGA outperforms in terms of messages and message gain. Figure 2(a) shows the recall in contrast of the maximum possible recall in a dynamic network. After only 15 queries INGA nearly doubles the recall of the naive approach and drastically outperforms *IBL*. Since INGA and REMINDIN use similar strategies for creating shortcuts both archive a similar recall. However, after introducing new topics in the network, INGA 's outperforms REMINDIN due to its optimized index for a dynamic network. Figure 3(b) shows the number of messages. Due to bootstrapping peers, that focus queries to a fraction of peers in the network, INGA outperforms and halves the messages in contrast to a naive approach. In contrast to REMINDIN INGA reduces the number of messages from about 85 to 58 messages. Figure 3(b) shows, that in terms of message gain INGA outperforms all approaches dramatically. Due to its improved indexing and shortcut selection strategy INGA nearly doubles the message gain of REMINDIN.



(a) Recall: Related Approaches (b) Messages: Related Approaches

Fig. 2. Comparison Recall and Message: Dynamic Network 1024 Peers, 6 Hops, k=2



(a) Message Gain: Layers (b) Message Gain: Related Approaches

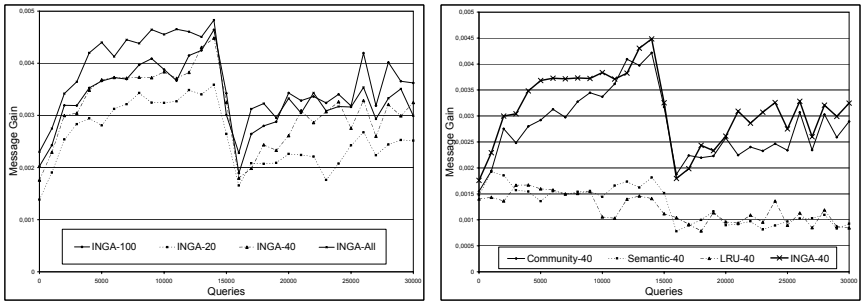
Fig. 3. Message Gain: Dynamic Network 1024 Peers, 6 Hops, k=2

Each layer contributes. Figure 3(a) shows the message gain of the different layers. Only using content provider shortcuts (Content-40) performs poorly, a combination of content and recommender shortcuts raise the message gain (Content Recommender-40) and finally the introduction of bootstrapping peers (INGA-40) additionally boosts INGA performance.

5.2 Setting optimal index size and weights

Limiting index size performs similar to an unlimited index. We conducted experiments with an unlimited index size and a maximum size of 100, 40, 20 shortcut entries. Figure 4(a) shows that an index size of 100 entries performs as good as an unlimited index while an index of 40 entries still is a reasonable tradeoff between size and routing efficiency.

Combined weighting is ideal, community weight outperforms. To determine an optimal weighting of the parameter (a, b, c) of the index policy, we conducted experiments



(a) Size (b) Weights

Fig. 4. Index Behavior: Dynamic Network 1024 Peers, 6 Hops, k=2

where we only consider the similarity locality ($a = 10, b = 0, c = 0$) , where we only consider the community locality (with $a = 0, b = 10, c = 0$), where we only consider the LRU-Locality ($a = 0, b = 0, c = 10$) and an 'optimal' combination ($a = 3, b = 6, c = 1$). [18] proposes a LRU strategy to update the index. We found out that there are better strategies. Figure 4(b) shows a similarity and LRU strategy, both perform worse and are alone not capable to adopt to the dynamics of the network and the changing interests of each peer. The community locality raises the message gain, even after changing the interests of each peer, while the combined strategy performs best.

6 Summary and Outlook

To our best knowledge we presented the first semantic query routing algorithm in a fully decentralized setting without any super peers. The novel design principle of our approach lies in the dynamic adaptation of the network topology, driven by the history of successful or semantically similar queries. This is memorized by using bounded local shortcut indexes storing semantically labelled shortcuts and a dynamic shortcut selection strategy, which forwards queries to a community of peers that are likely to best answer queries. Shortcuts connect peers that share similar interests and thus spontaneously form semantic communities. The clustering of peers within semantically communities drastically improves the overall performance of our algorithm even in a highly volatile setting, while our index policy keeps the shortcuts to the 'right' peers, that provide facts to the core interests of a requesting peer.

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 its 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. K. Aberer, P. Cudre-Mauroux, M. Hauswirth, and T. van Pelt. GridVine: Building Internet-Scale Semantic Overlay Networks. In *3rd. International Semantic Web Conference (ISWC), Hiroshima, Japan*, 2004.
2. A. V. Aho, P. J. Denning, and J. D. Ullman. Principles of optimal page replacement. *J. ACM*, 18(1):80–93, 1971.
3. J. Allan. Incremental relevance feedback for information filtering. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 270–278, New York, NY, USA, 1996. ACM Press.
4. S. Androutsellis-Theotokis and D. Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, 36(4):335–371, 2004.
5. 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.
6. T. Condie, S. Kamvar, and H. Garcia-Molina. Adaptive Peer-to-Peer Topologies. In *Int. Conf. on Peer-to-Peer Computing (P2P), Zurich, Switzerland*, 2004.
7. B. Cooper. Guiding queries to information sources with InfoBeacons. In *ACM/IFIP/USENIX 5th International Middleware Conference*, Toronto, 2004.
8. A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *International Conference on Distributed Computing Systems*, july 2002.
9. 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.
10. 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.
11. P. J. Keleher, B. Bhattacharjee, and B. D. Silaghi. Are virtualized overlay networks too much of a good thing? In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 225–231. Springer-Verlag, 2002.
12. J. Kleinberg. Navigation in a small world. *Nature*, 406, 2000.
13. 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.
14. B. Loo, J. Hellerstein, R. Huebsch, S. Shenker, and I. Stoica. Enhancing p2p file-sharing with an internet-scale query processor. In *In Proc. of Int. Conf. on Very Large Databases (VLDB)*, Toronto, 2004.
15. S. Milgram. The small world problem. *Psychology Today*, 67(1), 1967.
16. W. Nejdl, M. Wolpers, W. Siberski, A. Löser, I. Bruckhorst, M. Schlosser, and C. Schmitz. Super-Peer-Based Routing and Clustering Strategies for RDF-Based Peer-To-Peer Networks. In *12th International World Wide Web Conference*, Budapest, Hungary, May 2003.
17. S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. *Multimedia Systems*, 9(2), 2003.
18. K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient Content Location Using Interest Based Locality in Peer-to-Peer System. In *Infocom*. IEEE, 2003.
19. 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.