

Semantic Methods for P2P Query Routing

Alexander Löser¹, Steffen Staab², and Christoph Tempich³

¹ CIS, University of Technology Berlin, Einsteinufer 17, 10587 Berlin, Germany
aloeser@cs.tu-berlin.de

² ISWeb, University of Koblenz Landau 56016 Koblenz, Germany
staab@uni-koblenz.de

³ AIFB, University of Karlsruhe 76128 Karlsruhe, Germany
tempich@aifb.uni-karlsruhe.de

Abstract. Knowledge sharing in a virtual organization requires a knowledge life cycle including knowledge provisioning, terminology alignment, determination of resource location, query routing, and query answering. In this talk we focus on the issue of determining a relevant resource in a completely decentralized setting such as necessitated by peer-to-peer knowledge management in virtual organizations. Requirements for this task include, e.g., full autonomy of peers as well as full control over own resources and therefore preclude prominent resource location and query routing schemes such as distributed hash tables. In order to tackle given requirements we use a resource location and query routing approach that exploits social metaphors of topical experts and experts' experts as well as semantic similarity of queries and information sources. The approach has been fully tested in simulation runs and partially implemented in the system Bibster (<http://bibster.semanticweb.org>).

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 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 [11] and Kleinbergs [8] 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 out going 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 by 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.

Contributions and Paper Organisation. In this paper, we propose an improved shortcut selection strategy able to identify and semantical 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. To further boost performance and enhance recall in a dynamic setting we introduce in INGA 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 while it displays small world characteristics.

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 System Architecture

Our peer selection strategies described in section 3 are implemented independent on top of any unstructured P2P network. For evaluation purposes, though we use the SWAP infrastructure [5]. We recall that it provides all standard peer-to-peer functionality such as information sharing, searching and publishing of resources.

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 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 ¹. Additionally to local meta data (*MKlusch isOrganizerOf CIA2005*) each resource is assigned a topic (*MATES2005 isTypeOf AgentConference*) and hierarchical information about the topics is stored (*AgentConference 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 it is advantageous for it. 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*. 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.

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: $QM(q, b, mp, qid)$ where q is a SERQL query (cf. footnote 1). We support any SERQL queries, however for routing purposes only the topic information is used. From a query for all *AgentConferences* organized by *MKlusch*, only *AgentConference* 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

¹ <http://www.openrdf.org/>

and content provider shortcuts. We generate simplified queries such as `getdata(s,p,o)` with `s`, `p`, `o` being either concrete URIs or (for `o` only) literals. Furthermore, instead of a general RDFS ontology, we assume that we have topic hierarchies, which exploit the transitivity of RDF(S) *subclassOf*.

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 $sim : q \times sc \rightarrow [0; 1]$ between a query q and a shortcut sc , which are both given by query terms in the same topic hierarchy, as according to [9] as :

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

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 [9], we chose $\alpha = 0.2$ and $\beta = 0.6$ as optimal values.

3 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:

3.1 Content Provider and Recommender Shortcuts

Content Provider Layer. The design of the content provider shortcut overlay departs from existing work as published by [13,14] 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 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. 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 an example 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.

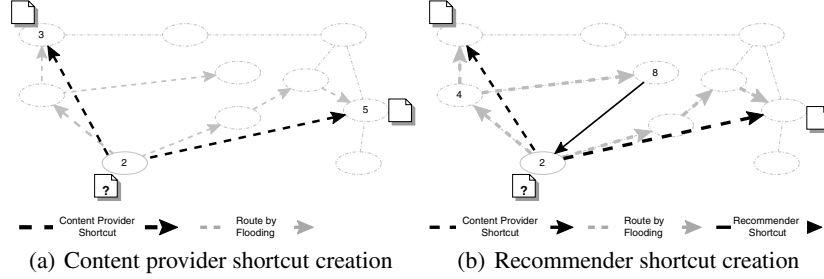


Fig. 1. Topic specific shortcut creation

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 ones peer. A recommender shortcut $sc(topic, pid, query\ hits\ maxsim, rp, update)$ 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 PID of the querying peer. Since we will get no 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 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. For an example 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.

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 [1]. 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 normalized with 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.

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

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

3.2 Bootstrapping Shortcuts

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 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 it's 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 it's bo value using equation 3

$$Bo = (1 + |outdegree|) \times (1 + |indegree|) \quad (3)$$

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.

3.3 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).

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. To form such semantic communities, for each query

INGA is executed in several steps executed locally and across the network, we already described the steps in [14,10]. 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 it’s *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.

Algorithm 1. Dynamic

Require: Query q , int k , int t_{Greedy}

Ensure: $TTL_q < maxTTL$

- 1: $s \leftarrow TopGreedy(q, Content/RecommenderShortcuts, (k, t_{Greedy}))$
 - 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* we select k peers from content or recommender shortcuts in subroutine 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} . Subroutine *TopGreedy* browses through the index of all content or recommender shortcuts and identifies the most similar matching shortcuts for a query above t_{greedy} . If two shortcuts have the same similarity, we choose the shortcut with the higher query hits value. The subroutine carefully selects the top- k peers for a query by avoiding different shortcuts with overlapping peers step. If found less than k shortcuts we select the top bootstrapping shortcuts (step 3) in subroutine *TopBoot*. It works similar to *TopGreedy*, 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 the up remaining shortcuts randomly from the default network and return the set of selected shortcuts. 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 some randomly chosen peers to the selected set of k peers to avoid overfitting of the selection process as known from simulated annealing techniques. The *Dynamic* algorithm terminates if the query has reached its maximum number of hops.

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 have 2 ; 204 have 3 ; . . . ; 44 have 6; . . . ;14 have 10 ; 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; 128 three ; ... one editor 20 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 choose a uniform query distribution instead of a ZIPF-distribution, which is typically observed in file sharing networks [12]. 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 networks². 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 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 [12]: 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. To simulate changing interests, after 15 queries, equal to ca. 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.
- **Clustering coefficient** represents the compactness of the network. It captures how many of a node's neighbors are connected to each other. We define the clustering coefficient as

$$C = \frac{1}{|V|} \sum_{v \in V} \frac{|E(F_v)|}{k_v * (k_v - 1)} \quad (4)$$

² We used the Colt library <http://nicewww.cern.ch/~hoschek/colt/>

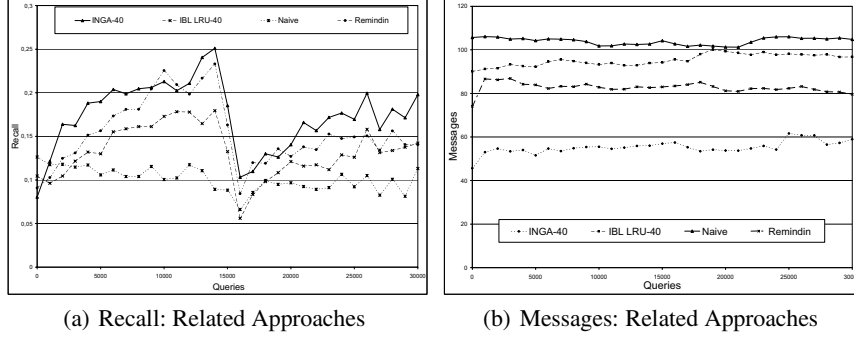


Fig. 2. Comparison Recall and Message: Dynamic Network 1024 Peers, 6 Hops, $k=2$

where V denotes the set of peers in the network, k_v denotes the maximum number of shortcuts for a peer v , Γ_v the direct neighbors of a peer and $E(\Gamma_v)$ represents a function that counts the number of links in Γ_v .

- **Average path length** A short average path length denotes a highly directed information flow between two peers in the network. Given two arbitrary selected peers $v_1, v_2 \in V$ and $d_{min}(v_1, v_2)$ the minimum path length between v_1 and v_2 , we define the average path length as

$$d = \frac{1}{\binom{|V|}{2}} \sum_{v_1 \neq v_2} d_{min}(v_1, v_2) \quad (5)$$

INGA outperforms in terms of Messages. As a baseline we compare INGA against the strategy of [13](IBL), REMINDIN [14] (all with an index size of 40 entries) and of Gnutella (Naive). Figure 2(a) shows the recall in contrast to 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 achieve 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 2(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.

Tradeoff Between Clustering and Recall. Small-world graphs are defined by comparison with random graphs with the same number of nodes and edges: first, a small-world displays a small average path length, similar to a random graph; second, a small-world has a significantly larger clustering coefficient than a random graph of the same size [6]. To measure the small world characteristics for different index settings we conducted experiments where we only consider the similarity locality ($a = 10, b = 0, c = 0$), only community locality (with $a = 0, b = 10, c = 0$), only LRU-locality ($a = 0, b = 0, c = 10$) and an 'optimal' combination ($a = 3, b = 6, c = 1$). We discover that the

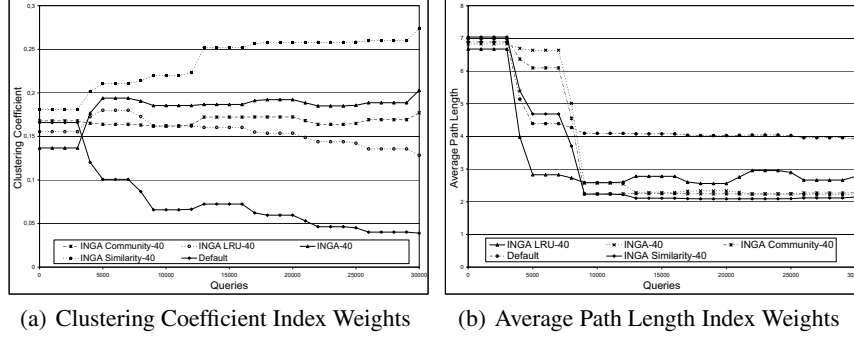


Fig. 3. Index Behavior: Dynamic Network 1024 Peers, 6 Hops, $k=2$

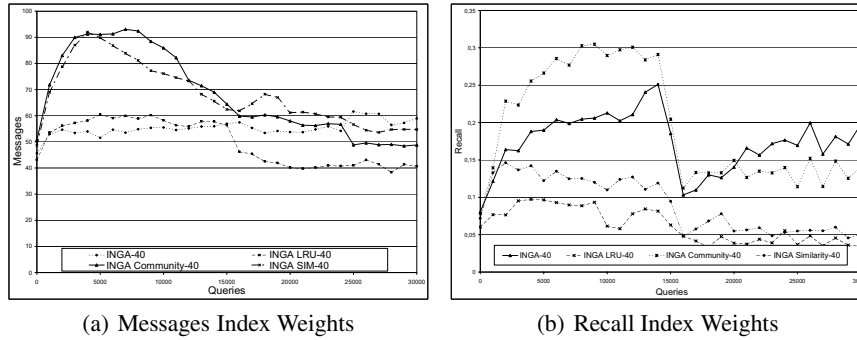


Fig. 4. Index Behavior: Dynamic Network 1024 Peers, 6 Hops, $k=2$

INGA data-sharing graph displays small-world properties. Figure 3(b) shows, that all index settings reduce the average path length in contrast to the default network. Due to the peer dynamics the default network selects the most stable peers with a path length of four hops while all index settings reduce the path length to 2 hops. However, *INGA LRU-40* stabilizes less than the other approaches. The clustering coefficient increases for most of our index configurations. Only *INGA LRU-40* decreases after a slight increase the clustering coefficient. Hence a LRU strategy alone is not able to create a highly clustered network. However, a high clustering coefficient does not correlate with a high recall: Figure 3(a) shows that the high clustering coefficient of *INGA SIM-40* outperforms while Figure 4(b) shows that the highest recall is achieved through the optimal setting *INGA 40*. Since clustering in the network focusses queries to a small set of peers storing similar shortcuts it reduces the number of randomly discovered peers as well. However, such randomness is crucial in a highly dynamic setting to achieve a high document recall. We therefore recommend the *INGA SIM-40* setting especially for expert finder applications, that use a more static setting and that are optimized towards knowing the right peers in contrast to a high document recall. For applications that prefer a high document recall, we recommend the setting of *INGA 40*.

6 Related Work

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 [4] or *Napster*. One of today's main technique for indexing P2P systems are so-called distributed hash tables (DHTs)(see [2] for a survey) that without need of a central index allow to route queries with certain keys to particular peers containing the desired data. While the visualization of keys and objects in the same name space used in structured overlays provides a 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 [7]. 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 [3]. This indexing strategy locally stores information about specific queries and what peers were successfully queried in the past. [13] 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 exact matches a query, else they use a flooding approach. To update the index they use a LRU strategy. REMINDIN [14] 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 assumptions that is not realistic.

7 Summary and Outlook

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 that show typical small world characteristics, e.g. a high clustering coefficient and a low average path length. The clustering of peers within semantical communities drastically improves the overall performance of our algorithm even in a highly volatile setting. In extensive simulations with different index strategies we have shown an trade-off between recall and clustering: Especially in volatile networks leads 'over clustering' to a local optimum which reduces the recall for query. We hope that our findings will find their way into future semantic query routing applications and will help them gaining the ability to deliver high quality search results efficiently.

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. A. V. Aho, P. J. Denning, and J. D. Ullman. Principles of optimal page replacement. *J. ACM*, 18(1):80–93, 1971.
2. S. Androutsellis-Theotokis and D. Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, 36(4):335–371, 2004.
3. A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *International Conference on Distributed Computing Systems*, july 2002.
4. 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.
5. 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.
6. A. Iamnitchi, M. Ripeanu, and I. Foster. Small-World File-Sharing Communities. In *23th. IEEE InfoCom HongKong*, 2004.
7. 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.
8. J. Kleinberg. Navigation in a small world. *Nature*, 406, 2000.
9. Y. Li, Z. Bandar, and D. McLean. An Approach for measuring semantic similarity between words using semantic multiple information sources. In *IEEE Transactions on Knowledge and Data Engineering*, volume 15, 2003.
10. A. Löser, C. Tempich, B. Quilitz, W.-T. Balke, S. Staab, and W. Nejdl. Searching dynamic communities with personal indexes. Technical report, University of Karlsruhe, Institute AIFB, 2005.
11. S. Milgram. The small world problem. *Psychology Today*, 67(1), 1967.
12. S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. *Multimedia Systems*, 9(2), 2003.
13. K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient Content Location Using Interest Based Locality in Peer-to-Peer System. In *Infocom. IEEE*, 2003.
14. 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.