pNear: combining Content Clustering and Distributed Hash Tables

Ronny Siebes

Vrije Universiteit Amsterdam, The Netherlands
`ronny@cs.vu.nl`

Abstract. Full-text search is a challenging problem in Peer-to-Peer (P2P) systems. Currently two promising directions to solve this problem are (1) distributed indexes like hash-tables (DHTs) and (2) semantic overlay networks (SONs) which can be divided into systems that cluster peers with similar content based on term overlap and systems that map both the content and queries on a shared semantic data structure. In this paper we present the pNear system that combines DHTs with clustering via term overlap and show that we are able to tackle some important disadvantages that hold for the individual approaches. We evaluate our approach via simulations based on a large and realistic data-set that we have constructed for this purpose, and which will be useful for similar experiments by others.

1 Introduction

Undisclosed content, lack of privacy and the possibility to censor data, are seen as important disadvantages of the centralized approach of today's popular search engines. Firstly, in such a centralized approach, the owner of the server has complete control over which content and in which order the content is presented to the user. The drawback of this approach is that authorities could force the search engine not to show some content that they do not like. Secondly, much data on the web is dynamically generated via databases and therefor are very difficult to crawl by search engines. Often this is by intention of the provider because it wants a unique access point for users to find its data, guaranteeing user traffic to the web-site and/or keeping full control over the data. Thirdly, also privacy is an important issue that is in principle not guaranteed by centralized search engines. Namely, search engines easily can connect IP-addresses with queries and can make a profile of the users behind it. Peer-to-Peer systems, where nobody is in control, are in principal much more difficult to be used for tracing the behavior of users. These three issues are important reasons for doing research on P2P-based search engines.

In this paper we propose a Peer-to-Peer system named pNear, where peers describe their content by a set of terms. pNear combines distributed indexing based on Distributed Hash Tables (DHT) together with clustering peers with similar content descriptions without depending on a shared data-structure like in pSearch [11] and Bibster [5] or a complete distributed index like in GridVine [2]. We show that only a small part of the terms in the content descriptions need to be indexed via DHT to still guarantee a

good recall and precision, even when a peer is not clustered due to an empty expertise description or posts a query that is not related to its expertise. The next section provides a brief overview of existing work on DHTs and semantic overlays. Section 3 shows our model that combines methods that are described in section 2. Section 4 presents an empirical validation via simulation experiments. Section 5 concludes the work.

2 DHTs and Semantic Overlays

2.1 DHT

Distributed hash tables (DHTs) are currently seen as an important building block for peer-to-peer systems for storing and allocating content in a completely decentralized way [1, 8, 10, 9]. This allows each node to function independently and collectively form the complete efficient search system without any central coordination. The general idea of DHTs is that each item shared on the network is hashed to a unique key, and that this key together with the content (or a pointer to it) are efficiently routed to the unique peer responsible for that key. In this way, each peer is responsible for storing the content (or a pointer to it) that is associated with the key. In principle, all DHT based systems provide two functionalities: $store(key, object)$ storing an $object$ identified by its key, and $search(key)$ which returns the $object$ (when it exists) from the peer whose network identifier is numerically closest to the key. The current systems based on DHTs provide these efficient key lookup and storage algorithms needing only $O(log(N))$ messages per search and storage, where N is the number of peers in the network. There are also some disadvantages with standard DHT approaches: firstly, there are the administration costs needed to maintain the network overlay during content updates and at peer joins, leaves and failures. This results in much maintenance traffic over the network when there are many updates or when peers frequently join and leave the network. Secondly, when the data distribution is extremely skewed, for example in document distributions on the web that follow a Zipf-distribution, additional load-balancing algorithms on top of DHT are needed to equally distribute data over the network to prevent node bottlenecks. Thirdly, in standard DHT approaches each key is mapped to one peer, which means that when this peer for whatever reason does not respond, the content cannot be found on the network.

2.2 Semantic Overlays

Peers that keep pointers to other peers which have similar content as themselves form a Semantic Overlay Network (SON). Edutella [7] is a schema based network where peers describe their functionality (i.e. services) and share this with other peers. In this way, peers know about the capabilities of other peers and only route a query to those peers that are probably able to handle it. Although, Edutella provides complex query facilities, it has still no sophisticated means for semantic clustering of peers and their broadcasting does not scale well. Gridvine [2] uses the semantic overlay for managing and mapping data and meta-data schemas, on top of a physical layer consisting of a structured peer-to-peer overlay network, namely P-Grid, for efficient routing of messages. In essence,

the efficiency of the search algorithm is caused not by smart forwarding queries based on the semantic overlay, but by applying the underlying DHT approach for mapping terms to peers.

From this point, due to our focus, we only look at systems where the goal is an efficient search mechanism based on routing queries to peers that are semantically closest to the content of the query.

One way to do this is that the content of a peer is classified into a shared term vector where each element in the vector contains the relevance for that given peer for the respective content. pSearch [11] is such an example where document indices are distributed through the P2P network based on document semantics generated by Latent Semantic Indexing (LSI) [3]. The search cost (in terms of different nodes searched and data transmitted) for a given query is thereby reduced, since the indices of semantically related documents are likely to be co-located in the network. In pSearch each peer has the responsibility for a range for each element in the semantic vector, e.g. ($[0.2 - 0.4], [0.1 - 0.3]$). Now all vectors that fall in that range are routed to that peer, meaning that, following the example vector, the vector $[0.2333, 0.1939]$ would be routed to this peer and $[0.1322, 0.1939]$ not. One disadvantage of pSearch is that new documents in the network are 'folded' into the existing semantic vector, which means that when there are new terms in the documents that are not in the existing vector, they will not be used in the routing process. This means that when the content in the network changes frequently, also the computationally expensive LSI method has to be applied very often.

Another approach is based on random walk clustering, where peers with similar content are going to know each-other by doing random visits [12]. The problem of this approach in the domain of full-text searches, is what information a peer has to tell to another peer so that they are able to determine if they are related or not. When there is no shared data-structure (like a fixed set of terms) in which they can describe their content, the whole content has to be shared. As a result, much data has to be shared between peers for determining closeness. To reduce this problem, solutions have been proposed in which peers describe their content in a (much smaller) set of terms that are shared by all peers in the network. Mostly these terms are organized in a topic hierarchy which allows a shared distance metric to determine the semantic similarity between terms by looking for example to the path distances in the graph. A peer knows about the expertise of other peer's by analyzing answers or advertisement messages where the shared terms have to be extracted from or are given explicitly [4]. In this way peers know peers that are semantically related to its own expertise descriptions. Given a query, the shared distance metric allows to forward queries (described by a shared set of terms) to neighbors of which their expertise description is semantically closely related to the query which is much more efficient than sending the query to random peers.

The specific advantages of Semantic Overlays are threefold:

– *Peer autonomy* Each peer can, in principle, have its own distance metric, peer selection mechanism and/or advertisement policy. This allows peers, for example to keep their neighbor list or similarity metric secret. Also peers can decide at any time to change their visibility on the network by sending advertisement messages. Also a peer can choose a shared distance matrix or topic hierarchy or create its own data-structures that it could share with other peers. The quality of the routing

process only depends on how effective the clustering process is and how effective individual peers are in determining a subset of its neighboring peers which have a good chance of answering the query.

– *Automatic load balancing* When some content is very popular, the semantic cluster on that content will contain many peers. In this way, load balancing is an emergent property of the network.

– *Robustness/fault tolerance* When peers leave the network or do not respond to a query, the only consequence is that they probably will not be asked a next time until they send new advertisement messages or are recommended by other peers. In contrast, most DHT approaches have to move routing tables to other peers in order to restore the overlay.

The problem of most SON approaches is that they either rely on a shared data-structure (distance matrix, topic-hierarchy or term-vector) in which the content has to be described and/or rely on the assumption that the queries of a peer are related to its own content and that a peer also has content that allows it to cluster itself into the overlay. These assumptions are not always realistic. In the next section we propose an approach that combines DHT with SON which benefits from the advantages of both approaches but where the combination levers out the individual disadvantages.

3 pNear: combining Expertise Clustering and Distributed Indexes

In this section we informally describe our approach of combining methods that we described in the previous section. At the end of this section we argue how our system resolves the problems of the individual drawbacks of the methods that our system depends on.

In pNear, besides sharing content, peers play two additional roles, namely that of *expertise cache* and of *expertise register*. The word 'expertise' should be read as a set of terms that describe the content from a peer together with its network identifier. Expertise registers are distributed indexes that map terms to peers that registered themselves as experts on these terms. The registering process is as follows: when a peer joins the network and/or has new content, it summarizes its content that it shares by a set of terms that we name *expertise descriptions*. After that, a small random set of terms from the expertise descriptions are selected for registering where each term from this set is hashed to a unique key that serves as the identifier of the *register message* that has to be routed, via DHT algorithms, to the peer (register) that is responsible for the key. These register messages each contain the term that was hashed, the sending peer and its expertise description. In this way a register which is responsible for a given term t contains a set of peers with their expertise descriptions that registered themselves on the term t. The process of storing and retrieving content mapped to keys is very efficient, because it is based on the DHT algorithms [pastry, chord, can] that only need $O(\log(N))$ messages (where N is the number of peers in the network) to retrieve or store a key and the mapped content. The registering process is not only meant for distributing expertise descriptions to registers so that they can be used in the query process, but also to allow the registering peers to know other peers that registered themselves at the register on

the same terms. Namely, when a register message is processed by a register, it returns a *register result message* containing pointers to peers that are similar together with their rankings based on a shared similarity measure on the expertise descriptions. Now that the registering peer has some pointers to related peers, the clustering process starts. Namely, the peer sends an *advertisement message* to some of the peers (which were returned by the register) containing the senders expertise description. In this way the receiver gets to know the sender and when its expertise is semantically similar, it stores the peer in its *expertise cache*. Expertise caches therefor will contain pointers to peers that have similar content to the peer that maintains the cache. It also looks in its own expertise cache for peers that are semantically close to the sender of the advertisement message. If some peers are found, they are put together with their rankings in an *advertisement result message* and sent back to the initiator of the advertisement message. In this way also the sender gets to know even more peers that have similar content to which it can repeat the advertising process. This process stops till a given maximum number of *advertisement rounds*.

The query process is started when a user initiates a query on the peer that represents that user on the network. First, the peer abstracts the query, e.g. via stemming and removing stop-words, into a set of terms that is used to compare the similarity between the query and expertise descriptions that it and others will encounter in the query distribution process. Next, the peer looks in its local expertise cache for peers of which their expertise descriptions are semantically close to the terms in the query abstraction. When there are enough peers these are selected to send the *query messages* to containing the query itself and the query abstraction. The receiving peer tries to answer the query and looks in its own expertise cache if it knows some peers that are semantically close to the query abstraction and sends both the eventual answer and the eventual set of pointers to related peers back via an *query result message*. When the initiator of the query receives the messages it (or the user) decides to continue the query process or if it is satisfied with the number of answers. It could be that the peer that initiated the query has no, or not enough, semantically related peers in its expertise cache to send the query to. This could for example happen when the peer does not share any content on the network, resulting in no expertise description, or when the user posts a query that is completely unrelated to the content (and therefor the peers in the cache) that a peer shares. When this happens, the expertise registers come into place. Namely, first the terms in the query abstractions are hashed to unique keys that will serve as identifiers of the *register consult messages* that the peer will send to the registers. A register consult message contains the term that was hashed, the senders identifier and the complete query abstraction. The routing process of these messages is identical to the expertise registering process.

Now that we informally described our method we have to show how it circumvents the drawbacks of the individual methods where it is based on. We do this by showing results of simulation experiments in the following section combined with the following argumentation:

The drawbacks of pure DHT again are first its expensive maintenance, where there is a linear increase of costs with the number of terms that are stored in the network. The results of the next section have to show that we can reduce the number of terms to only

a fraction of the original set. The experiments also need to show that the maintenance costs of the semantic overlay are much less than the maintenance of the pure DHT overlay. Secondly in a pure DHT approach only one peer is responsible for a key space. When this peer contains a very popular key, the peer may become a bottleneck if it has not enough bandwidth or processing power to deal with the huge number of requests. Although in our system there is still one register for each hashed term, the clustering of related peers allows that, given the assumption that interest and expertise are related, even when the register does not respond, the neighbors of the querying peer are able to answer it and/or know some good candidates. Note that the assumption only is needed in this case where a register does not exist or respond. Thirdly the skewed distribution of content and queries result that in pure DHT some peers have to store much more keys and deal with much more queries than other peers. Results need to show that our method reduces this problem because the method allows peers to have a small fixed register and cache size and still give good results. The main drawbacks of semantic overlays are that either they need a shared semantic data structure which is expensive to update and can be very large or it completely depends on the assumption that peers have an expertise description and that the expertise of a peer and its interests (the queries) are closely related. Our method does not depend on a shared semantic data-structure and in our experiments we let queries originate from peers that are not part of the semantic overlay and which also have no expertise description from itself.

4 Experimental setup and simulation results

In this section we show how we evaluate our approach and the results by describing the data set, determining the evaluation criteria and showing the results of our simulations.

4.1 Data set

We built our data set in the following way:

- **Query set** Q We used SearchSpy[1] to crawl a set of real user queries. SearchSpy offers the possibility to filter out queries that are 'family unfriendly', which more or less means that the queries on porn are removed from the set. We chose to use this filter. The crawling process resulted in a set of 28.606 unique user queries Q with an average length of 2.88, a minimum of 1 and a maximum of 15 terms per query.
- **Pointers to web-pages** G For each query $q \in Q$ we used Google[2] to find at least 1 and at most 100 web-pages in the English language that match q and put the URL's in a set G. When a query has not a matching web-page, the query is ignored which happened in 0.06% of the queries. This resulted in a set of 28.589 queries that have on average 84 URL pointers per query.

---

[1] http://www.infospace.com/info.xcite/searchspy
[2] http://www.google.com

– **Terms extractions from web-pages** T For each $g \in G$ we crawled the textual content of the web-page that belongs to the corresponding URL. For each crawled web page we used a natural language processing tool, called TextToOnto [6], which extracted the terms that occur at least three times (indication that term is important) in the documents, removed the stop words and stemmed them. Documents of which no terms could be extracted are removed from the data set which happened in 40% of the cases. The crawling process resulted in a set of terms per document. The average number of terms is 127, the minimum is 1 and the maximum is 2184.

– **Simulation Queries** Q_s Instead of using Q, we choose to select random subsets of minimally i and maximally j (both parameters in our simulation) terms from T that will function as queries that peers send during the simulation experiments. The reason for creating this artificial set is twofold. Firstly it gives us more flexibility to indirectly play with the number of matching peers that are in the network, namely queries that contain many terms normally have less answers than short queries. Secondly, the goal of this paper is to test how effective our approach is in finding matching expertise descriptions for queries that are subset of those descriptions and not how able the system is to match real user queries with expertise descriptions which is a complete different issue and more suitable for research in natural language processing. The distribution of the number of matching peers per query is shown in Figure 1, which is the same for all simulation experiments. As can be seen, most of the queries have only a few matching peers, which makes the chance that a peer by coincidence finds the matching peers very small. The average number of peers that match a query is 34, but note due to the exponential curve most queries have a smaller number of matches and some queries have many matching peers. Future work is to play with other distributions.
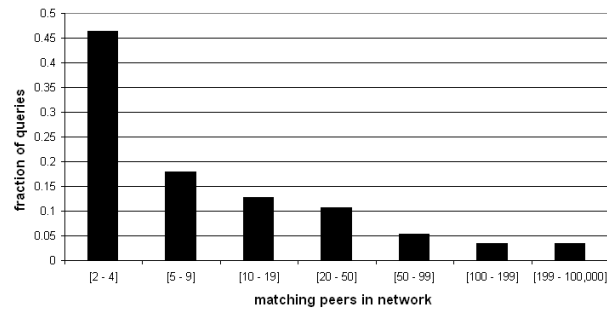

Fig. 1. The distribution of the number of matching peers per query used in all simulations.

In effect, this data-set allows us to simulate real queries on real web-pages as they appear on a real search (centralized) search engine, and measure how these realistic queries on realistic data-sets perform in a distributed P2P setting instead. We believe

that the creation of this data-set in itself is a contribution of this paper. The data-set is available from the author on request.

4.2 Evaluation criteria

We explore the characteristics of our system by defining a set of evaluation criteria of which we think are an accurate measure of user satisfaction and bandwidth usage.

- **Peer recall per round** Rec_{round}

$$\frac{|P_{total\_relevant} \bigcap P_{queried\_relevant}|}{|P_{total\_relevant}|} = \frac{|P_{queried\_relevant}|}{|P_{total\_relevant}|}$$

  The peer recall per round states the cumulative fraction of the relevant peers in the network $P_{queried\_relevant}$ that are queried $P_{queried\_relevant}$ until the current query round $round$. This measure can be seen as an indication for user satisfaction. As we will see, the optimal policies differ for distinct recall levels which means that the optimal settings of the network depends on the recall requirements from the users.
- **Average number of messages per query** $\#Q_{msgs}$ The average number of messages used to resolve a query. It is an indication of the bandwidth usage of the system.
- **Average number of advertisement messages per register process** $\#A_{msgs}$ When a peer has registered its expertise at some registers, it uses the set of returned and known pointers to peers that are closely related to its own expertise for advertising its expertise and to find more related peers. When this peer decides to send advertisement messages to these peers, direct messages are used, because the peer knows the network address of the peers. This average number of advertisement messages together with the number of register messages can be seen as the costs of maintaining the semantic overlay.

4.3 System parameters

In this subsection we describe the parameters that we implemented in our simulation platform. To keep the number of experiments within reasonable proportions, we unfortunately cannot do all permutations of set of values for the different parameters. To keep our document readable, we fix some of the parameters with default values shown in Table 1. The default and static values of these parameters are found by performing random experiments and choose those that gave good results.

4.4 Results

In this subsection, we show four different sets of experiments in which we variate four parameters that had much influence on the results. For each set we show a figure that gives for 1 to 15 query rounds the recall and the average number of query messages ($\#Q_{msgs}$) and advertisement messages ($\#A_{msgs}$) used when the 15_{th} round is reached.

Description	Value
Total number of nodes in the system	100,000
Maximum number of expertise descriptions in a peer's cache	50
Maximum number of expertise descriptions in a peer's register	20
Maximum number of recommendations given by a register	20
Maximum number of recommendations given by a cache	30
Maximum number of advertisement rounds per advertisement initialization	5
Maximum number of neighbors selected per advertisement round	4
Minimum and maximum number of terms in query	1,6
Maximum number of neighbors selected per query round	7
Total number of queries per simulation experiment	10.000
Average number of terms to register selected from expertise description	3

Table 1. Static and default values for the parameters in the simulations

Number of terms selected for registering. Figure 2 shows the influence of the number of terms that a peer selects from its expertise description for the registering process. When 3 out of 127 terms (resulting in $3 \times 2log(100,000) = 50$ direct messages needed to store the object in the DHT overlay) are registered and on average 17.8 advertisement messages are sent, on average 86.4 query messages are needed to find around 31% of the on average 32 peers relevant peers in the network of 100K peers (Fig. 1). This means that this recall of 31% is reached by $50 + 17.8 + 91.7 \approx 160$ direct messages. For 20 terms and almost the same number of advertisement- and query messages, a recall of 43% is reached. It is important to look at the slope of the curves: the gain of recall per round gets smaller after each round where more registered terms means a more steep curve than one with less registered terms. Summarizing, these results show that with a fraction of the terms and a few advertisements, a good recall can be reached.
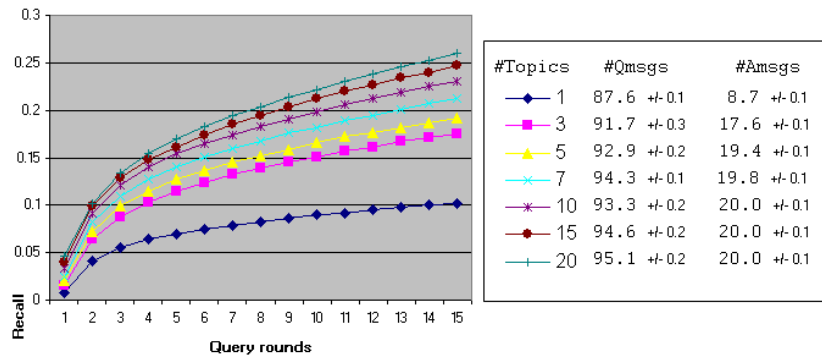

#Topics	#Qmsgs	#Amsgs
1	87.6 +/- 0.1	8.7 +/- 0.1
3	91.7 +/- 0.3	17.6 +/- 0.1
5	92.9 +/- 0.2	19.4 +/- 0.1
7	94.3 +/- 0.1	19.8 +/- 0.1
10	93.3 +/- 0.2	20.0 +/- 0.1
15	94.6 +/- 0.2	20.0 +/- 0.1
20	95.1 +/- 0.2	20.0 +/- 0.1

Fig. 2. Influence of varying the average number of terms that are registered per peer.

The number of recommendations from the cache for a query. Figure 3 shows the influence of the number of recommendations that a cache maximally returns for a query. The results indicate that the storage size is much more important than the number of recommendations. For example, 10-50 (max. 10 recommendations per query and max. 50 descriptions stored per cache) gives better results than 40-40. The slope of the curves indicate that for the caches with a storage capacity of 30 and more, an small increase of the number of rounds (or query neighbors per round which is not shown in this paper) would result in an significant increase in recall. The results confirm that it is no problem that there is a fixed and relative small cache size (also for the register).
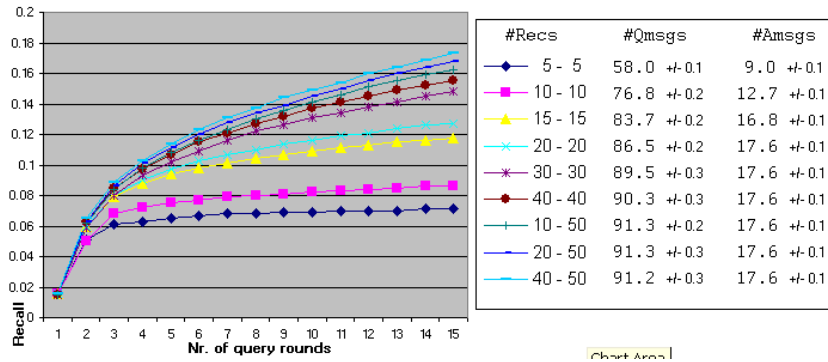

#Recs	#Qmsgs	#Amsgs
5 - 5	58.0 +/- 0.1	9.0 +/- 0.1
10 - 10	76.8 +/- 0.2	12.7 +/- 0.1
15 - 15	83.7 +/- 0.2	16.8 +/- 0.1
20 - 20	86.5 +/- 0.2	17.6 +/- 0.1
30 - 30	89.5 +/- 0.3	17.6 +/- 0.1
40 - 40	90.3 +/- 0.3	17.6 +/- 0.1
10 - 50	91.3 +/- 0.2	17.6 +/- 0.1
20 - 50	91.3 +/- 0.3	17.6 +/- 0.1
40 - 50	91.2 +/- 0.3	17.6 +/- 0.1

Fig. 3. Influence of varying the the maximum number of recommendations and the maximum storage size from a cache. The notation in the legend should be read as follows: e.g. 20-50 means max. 20 recommendations and max. 50 expertise descriptions in a peer's cache.

The number of advertisements. Figures 4 and 5 show the influence of the number of advertisements that are send on average per peer per advertisement initialization. The graphs indicate that both increasing the number of neighbors to advertise to and the number of advertisement rounds have a positive effect on the recall. The figures indicate that it is slightly better to have more rounds instead of more neighbors (for example compare $#neighbrs$ 4 from Fig. 4 with $#rounds$ 4 from Fig. 5, both needing around 17 advertisement messages), which makes sense because in the first case a peer gets quicker to the right clusters in the network. A disadvantage is that more rounds means also more time for an advertisement process to finish. The results confirm that only a relatively small number of advertisements are needed to build the semantic overlay.

5 Conclusion

Distributed Hash Tables and content clustering are both two techniques to improve the efficiency of searching content in a Peer-to-Peer network. Both techniques have their own drawbacks which we discussed in Section 2. In this paper we presented the pNear
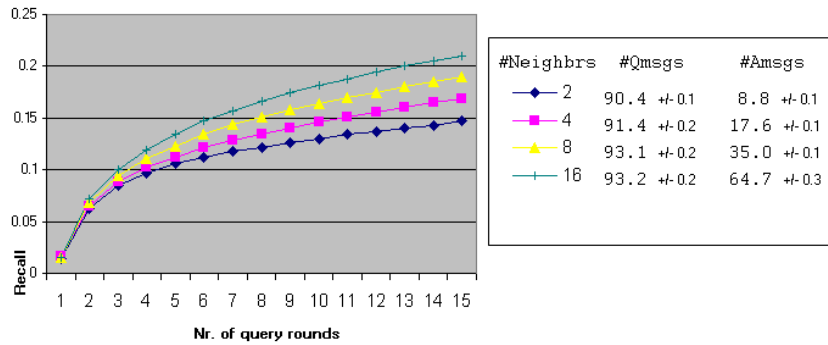
Fig. 4. Influence of varying the maximum number of neighbors selected per round in an advertisement process
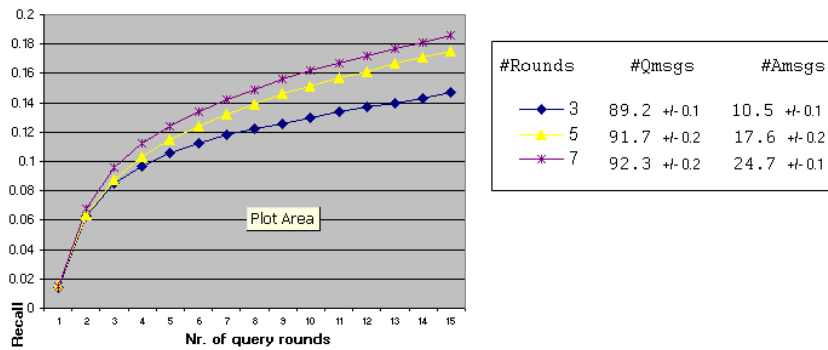

Fig. 5. Influence of varying the maximum number of advertisement rounds.

system that combines both techniques: it uses DHTs to add a functionality to each peer, making them a kind of 'yellow pages' where peers can register expertise descriptions, which are sets of terms that describe the content that those peers share. The DHT approach allows a peer to efficiently find registers that are responsible for the terms in the query. Due to the clustering of expertise, the returned peers by the register 'know' related peers by which the query has a good chance to be answered. In this way the rest of the potentially correct peers are found. Our simulation results indicate that the method finds a large fraction of the correct peers for a query originating from an unclustered peer, where only a small fraction of the terms from a peer's expertise description need to be hashed and stored at the registers. Also only a small number of advertisement messages is needed to build the semantic overlay.

6 Acknowledgments

References

1. Karl Aberer, Philippe Cudré-Mauroux, Anwitaman Datta, Zoran Despotovic, Manfred Hauswirth, Magdalena Punceva, and Roman Schmidt. P-grid: a self-organizing structured p2p system. *SIGMOD Rec.*, 32(3):29–33, 2003.
2. Karl Aberer, Philippe Cudré-Mauroux, Manfred Hauswirth, and Tim Van Pelt. Gridvine: Building internet-scale semantic overlay networks. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *International Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, pages 107–121. Springer, 2004.
3. Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.
4. P. Haase, R. Siebes, and F. van Harmelen. Peer selection in peer-to-peer networks with semantic topologies. In Mokrane Bouzeghoub, editor, *Proceedings of the International Conference on Semantics in a Networked World (ICNSW'04)*, volume 3226 of *LNCS*, pages 108–125, Paris, June 2004. Springer Verlag.
5. Peter Haase, Bjrn Schnizler, Jeen Broekstra, Marc Ehrig, Frank van Harmelen, Maarten Menken, Peter Mika, Michal Plechawski, Pawel Pyszlak, Ronny Siebes, Steffen Staab, and Christoph Tempich. Bibster - a semantics-based bibliographic peer-to-peer system. *Journal of Web Semantics*, 2(1):6, 2005.
6. Alexander Maedche and Steffen Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2):72–79, 2001.
7. Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjorn Naeve, Mikael Nilsson, Matthias Palmer, and Tore Risch. Edutella: A p2p networking infrastructure based on rdf. In *Proceedings of the 11th International World Wide Web Conference*, May 2002. schema based searching Presentation: http://www2002.org/presentations/nejdl.pdf.
8. Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM '01*, 2001.
9. A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, Heidelberg, Germany, November 2001.
10. Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of the ACM SIGCOMM '01*, 2001.
11. C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. Technical report, HP Labs, November 2002.
12. S. Voulgaris, A.-M. Kermarrec, L. Massoulie, and M. van Steen. Exploiting semantic proximity in peer-to-peer content searching. In *10th International Workshop on Future Trends in Distributed Computing Systems (FTDCS)*, Suzhou, China, may 2004.