

# pRoute: Peer Selection using shared term similarity matrices

Ronny Siebes, Spyros Kotoulas

<sup>1</sup>Department of Computer Science, Vrije Universiteit Amsterdam, The Netherlands  
{ronny,kotoula}@cs.vu.nl

**Abstract.** Peer-to-Peer systems have proven to be an effective way of sharing data. The focus of this paper is on distributed search based on Peer-to-Peer technology. In this paper we present the pRoute system where peers advertise a short description of the content that they share, namely a set of terms. Peers remember the advertisements of related peers and thereby form a semantic overlay by which we mean that peers with similar content are grouped together. Peers calculate the similarity between their content descriptions by a term similarity function which, in the ideal case, is identical for all peers. In simulation experiments we compare the performance of different advertisement- and forwarding policies with respect to precision, recall and the number of messages. The results indicate precision and recall increase when the policies take semantics into account, without an increase of the number of advertisement- and query messages.

## Keywords

Peer-to-Peer Systems, Semantic Overlay Networks, Latent Semantic Indexing, Similarity Matrices

## 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 therefore 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 associate 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.

A big advantage of centralized search engines is that the number of messages needed in the query process often is only 2 and the number of hops is only 1, guaranteeing efficient bandwidth usage and quick response times. In Peer-to-Peer systems, the number of messages and hops mainly depends on how quickly the relevant peers are found. Needless to say that much of the current research on P2P-based document searching is focussed on reducing the number of messages and hops to generate an attractive alternative to the centralized approach especially when privacy, undisclosed content and censorship play a role.

In our short literature study we try to give an impression of the different distributed search approaches and indicate respectively their strengths and weaknesses. One of the approaches described in this overview is peer selection based on semantic overlay networks (SONs), which will also be the way we organize our P2P network. More precisely, we adopt the model of expertise-based peer selection using a shared data-model as is described in our previous work [11]. In this model, peers (automatically) summarize their content in so-called *expertise-descriptions*, being a set of terms provided a shared data-model. These descriptions are spread to some other peers in the network via *advertisement messages*. In this way, peers become aware of the expertise of other peers, enabling them to route queries only to those peers whose expertise is semantically close related to the content of the queries.

By this paper we present our pRoute system and hope to contribute in the following ways: first we propose a method to automatically generate a similarity matrix that can be used as an instantiation of the shared term similarity function needed by our expertise-based routing approach. Second, we compare different query and advertisement policies via simulation results based on large peer-sets and realistic data-sets. The results should be interpreted as guidelines for those who want to build a system that uses expertise-based peer selection for routing query messages. Important to note is that the routing method can be easily combined with other techniques like using Super-peers or Distributed Hash Tables.

The structure of this paper is as follows: first we discuss related work in section 2. Section 3 describes expertise-based routing model and the different query and advertisement policies. Section 4 proposes a method for automatically making a term similarity matrix which can be used as an instantiation of the shared term similarity function. Section 5 is about our experimental setup where the data set, the research questions and the simulation are described. Section 6 shows the results of our experiments and finally, section 7 interprets the results and indicates our future work.

## 2 Related work on distributed search

In this related work section, we organize the different systems found in the literature by methods that do not exclude each-other, meaning that a system can combine two or more methods together. Due to our focus on reducing messages and hops and increasing recall, we judge these systems on criteria concerning efficiency (network band-width, memory and cpu usages) and robustness (fault-tolerance, network dynamics).

*Broadcasting* Although a very simple technique, broadcasting has already proven its usefulness in small networks and in larger P2P file-sharing systems like Gnutella<sup>1</sup> [12]. The idea is that peers keep forwarding a query to their neighbors until a sufficient number of answers has been found or until the maximum number of forwards (hops) has been reached. This approach is not very scalable, because a query can result in many messages which consumes much network capacity. Thus, finding the expert that can answer a given query results in a blind search, where it is possible that even if the data is somewhere in the network, it will not be found due to the maximum number of hops. The big advantage of broadcasting approaches is that they have very low maintenance costs and dependency, meaning that almost no messages are needed to keep the network alive and it is very robust to frequent peer drops and joins (network dynamics). In case where broadcasting really is needed, Hypercup [19] guarantees that only  $N - 1$  messages are needed to reach all peers and  $O(\log N)$  hops, where  $N$  is the number of peers in the network. Moreover, they show how their scheme can be made even more efficient by using a global semantic network to determine the organization of peers in the graph topology. Namely, when peers describe their content in terms of this shared data-structure, peers are able to cluster themselves with similar peers. However, this approach based on a structured hypercube overlay has more maintenance overhead and is therefore also more sensitive to network dynamics than traditional broadcasting approaches.

*Central registries* A very popular approach is to have one central registry where systems can store their content descriptions or where the registry itself searches the network for the content descriptions. A well known example is Napster<sup>2</sup>, which has one large repository which combines filenames with peers that offer those files for downloading. Such a repository can be seen as a kind of yellow pages, where each member in the network can look up the person or system that fulfills its needs. In small organizations, such an approach could work very well because the network is small and stable, so that the registry does not have to do much query processing and updates. In larger networks the approach is not very robust and has the same disadvantages as completely centralized approaches: undisclosed content, lack of privacy and censor possibilities.

---

<sup>1</sup> The Gnutella protocol specification v4.0. <http://dss.clip2.com/GnutellaProtocol04.pdf/>

<sup>2</sup> Napster. <http://www.napster.com/about.us.html>, 2002

*Brokering* The Multi-Agent community proposed the concept of 'broker agents' such as in InfoSleuth [3]. These brokers semantically match information needs (specified in terms of some shared data-structure, e.g. an ontology) with currently available resources which are found by the broker or registered by the providing agents themselves. In InfoSleuth, agents advertise their services to the broker via the KQML [10] language. Broker agents respond to an agent's request for service with information about the other agents that have previously advertised relevant services. The literature on broker agents has a clear focus on finding services. Therefore, it is not surprising that the brokering approach is very popular in the literature on finding web-services which are semantically described [15]. One thing where the literature is not clear about is on how scalable and robust this approach is. In a network where millions of agents offer their services, one broker agent probably will not be enough and will have the same problems as with a central registry.

*Super Peers/nodes* An approach that looks very similar to brokering but with a different goal in mind, comes from the P2P research community. The technique, that seems to work well for file sharing, is to make use of the different capacities of the nodes in the P2P network. The fact, that some peers have more processing power, memory or network bandwidth than other peers is used to give them a harder job in the network. For example, KaZaa [13] let peers voluntarily be a super-peer where they have big routing tables (kind of yellow pages) where information is stored about the content of other peers. This approach is thus a hybrid solution between real P2P (all peers playing the same role) and centralized systems. Although better than broadcasting in a network without super-nodes, it remains broadcasting and therefore can be improved by techniques that do more efficient routing described in the next paragraphs.

*Distributed Hash Tables* A technique from the P2P research community makes use of Distributed Hash Tables (DHT) [17,18,21,1]. DHT's are based on the idea to route content (or a pointer to the content) to the peer whose identifier lies closest to the unique identifier of the content. This technique assumes that all peers share the same 'hash' function to assign a unique (mostly 128 bit) identifier to content, which could be anything like documents, music, URL's or words. The characteristic of this technique is that it allows to route content and queries in  $\log(N)$  steps (where  $N$  is the number of peers in the network) to the right peers. A disadvantage of most DHT approaches is that they have high maintenance costs, due to continuous changes in the overlay network as a result of peers joining and leaving. P-Grid [1] is a Peer-to-Peer search system based on a virtual distributed search tree, similarly structured as standard distributed hash tables however with an unstructured way of building the DHT overlay. Namely, they use randomized algorithms for constructing the access structure, updating the data and performing search. In this way probabilistic estimates can be given for the success of search requests, and search is more robust than the previously described DHT approaches against failures of nodes. A disadvantage of all DHT approaches is that content which is not hashed, cannot be found,

which is especially a problem with full-text searching. Namely, in a document sharing case, one could roughly do three things (1) the file itself is hashed to a unique key with the disadvantage that the user has to know this key too, in order to find the file, or (2) the title of the document is hashed. This is also a problem because one typing error would result in a completely different hash key. (3) All the words in the document are hashed and the document or the location of the document is stored at the peers whose id's are closest to the hash keys of the words. Although now someone is able to find the documents that contain the keywords, the procedure of distributing the hash keys is not efficient because all these keys have to be distributed to the right peers in the network. Another disadvantage of DHT-based approaches is that load-balancing is not an emergent property of the topology. Due to the fact that content and queries follow a powerlaw distribution [5], some peers (responsible for popular keys) are much more loaded than other peers that accidentally are responsible for less popular keys. Imagine that one peer is responsible for the hashed key of 'Britney Spears', and that this peer joined the network with a slow 56K modem! Therefore, active load balancing policies have to be developed on top of DHT, [6] which is not needed for broadcast-based and expertise-based (described in next paragraph) alike approaches. Also, a pure DHT-based approach is less robust than broadcast-based and expertise-based approaches, because normally only one peer is responsible for one key, and if that peer does not respond on queries (for example behind a firewall), no content can be found that is hashed to that key.

*Semantic Overlay Networks* Peers that keep pointers to other peers which have similar content as themselves form a Semantic Overlay Network (SON). Edutella [16] 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 this provides complex query facilities, there is 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 of the queries based on the semantic overlay, but by applying the underlying DHT approach for mapping terms to peers.

From here on, due to our focus, we only consider 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 topic vector where each element in the vector contains the relevance for that given peer for the respective content. pSearch [22] is such an example where document indices are distributed through the P2P network based on document semantics generated by Latent Semantic Indexing (LSI) [9]. 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 [23]. The assumption is that queries posted by (the users of) peers are semantically closely related to the content of the peer itself. This results in a high probability that the neighbors of the peer (the peers in the cluster of that peer) have answers to the query. 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. This results in the fact that much data has to be shared between peers for determining closeness. In [20], peers cluster passively by remembering peers that previously provided them with satisfying answers using a Gnutella overlay. This means that for each peer, clustering is achieved after an initial 'warm-up' period, which could take a while. In terms of performance, they achieve a workload reduction by a factor of 3 to 7 in comparison to Gnutella. The system described in [7] uses a similarity measure based on keyword overlap between the set of terms in a query and the set of terms that describe the content of a peer. Unfortunately, in the paper there is no information about the clustering overhead in the number of additional messages to form the overlay, which makes it difficult to compare with other approaches. Like most papers on semantic overlays, this paper does assume a peer availability of 100%, which is unrealistic.

In contrast to the previous clustering approaches, the last SON approach that we discuss here lets peers describe their content in a shared set of terms. Mostly these terms are organized in a topic network or hierarchy making it possible to determine the semantic similarity between terms. Thus a peer has a set of topics which describe its expertise, similar to the model described in our previous work [11]. A peer knows about other peer's expertise topics by analyzing answers or advertisement messages. In this way peers form clusters of semantically related expertise descriptions. Given a query, a 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. The advantages of this approach are threefold:

- *Peer autonomy* Each peer can, in principle, have its own distance measure, peer selection mechanism and 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 create its own matrix that it could share with other peers. The quality of the routing process only depends on the overlap of the matrices, which means that the bigger the overlap, the better the routing performance.

- *Automatic load balancing* When some content is queried very often, most systems will also have copies of that content on their machines, due to downloading the results of the queries. This means that the semantic cluster on that content will contain many peers. In this way, load balancing is an emergent property of the network when popular content is distributed over many peers.
- *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. For example, imagine that a peer has some documents containing the word 'hound', but the shared data-structure only contains the term 'dog', then two things can be done (1) extend the shared data-structure with the word 'hound' so that peers are able to query and describe their expertise with that term. (2) the functions that extracts the expertise description and abstract the queries should be intelligent enough to see that 'dog' is a good replacement for 'hound'. Note that in this case the original query still contains 'hound', but the routing mechanism uses the shared term 'dog' to route it to the peer that registered itself on that term. The problem with the first solution is that it will lead eventually to very large data-structures. To reduce this problem it is important to keep unimportant terms (e.g. stop-words) out the data-structure. The problem with the second solution is the strong dependency on the quality of the extraction and abstraction algorithms.

In the next section we introduce our SON approach where routing is also based on matching queries on content-descriptions described in terms that occur in a shared data-structure.

### **3 Expertise-Based Peer Selection based on a Shared Similarity Matrix**

In this section we explain our expertise-based selection method by showing the individual building blocks of the system and describing the different advertisement and query policies.

### 3.1 Elements

*Neighbor set* Peers joining a network are assumed to know an initial random set of other peers called its initial neighbor set. More formally, each peer initially has a bootstrap neighbor list in a small, fixed-sized cache of entries (with typical value 5, 10, or 20). A cache entry contains the network address (i.e., IP-address and port) of another peer in the overlay. This set could, for example, be downloaded from a web-site, a gate-way peer or come with the download of the implementation. This set of peers is used by the advertisement policy that sends the expertise descriptions (discussed later).

*Expertise Description* Each peer has an expertise extraction function

$$\epsilon : docs_p \mapsto \mathcal{T}_p \quad (1)$$

which maps the content of the documents of the peer  $docs_p$  into a set of terms  $\mathcal{T}_p$ . These terms describe the content of the peer’s documents. This function could be based on an automatic NLP algorithm or be performed by a user that selects a set of terms which classifies the documents. To improve the semantic overlay, in our simulations we always let peers describe their expertise in terms occurring in a shared term similarity matrix on certain domain, which will be introduced in on of the next paragraphs. The assumption of a globally shared similarity matrix is clearly a limitation. We expect that our approach still works with only partial overlapping matrices, but we do not explore this in our experiments. So this is future work.

*Query Abstraction* For simplicity reasons we assume that a query posted by a user is just a set of words, thus without boolean operators or other constructs except an implicit AND between the words. An answer on the query is therefore correct when all the words are in each of the result documents. The words in the query are mapped by a mapping function

$$\pi : query \mapsto \mathcal{T}_q \quad (2)$$

into a set of terms  $\mathcal{T}_q \subset 2^{\text{STRING}}$ . These terms could exactly match the terms in the users query, but also be stemmed terms and with stop words removed. In our simulations we instantiate  $\pi$  by mapping queries into a set of terms occurring in a shared similarity matrix. This means that  $\mathcal{T}_q \subseteq \mathcal{T}_{dom}$ .

*Term Similarity Matrix* A term similarity matrix is a matrix  $\mathcal{S}_{dom} : \mathcal{T}_{dom} \times \mathcal{T}_{dom}$  about a certain domain  $dom$  which contains semantic distances  $[0, 1]$  between the set of (popular) terms  $\mathcal{T}_{dom}$  in  $dom$ . In this way a peer is able to look up the semantic similarity  $sim : (t_1, t_2)_{dom} \mapsto [0, 1]$  between two terms  $t_1$  and  $t_2$ . These similarity values are used by the similarity function which will be described in the next paragraph. A user could download the domain matrices of interest (e.g. computer science or biology) from a web-site or develop its own although it is recommended to re-use similarity matrices as much as possible (will be

discussed later). There are numerous ways to determine the semantic distance between terms. One way is to do it manually namely by letting a group of domain experts giving the distances. Another way is to use an ontology [11] or another kind of semantic graph where the minimal path distance between topics is an indication of similarity. Our approach, which we will discuss later, is letting statistical algorithms automatically analyze a representative set of documents and determine the most descriptive terms and the similarity between them.

*Similarity Function* According to the expertise-based selection method, each peer has complete autonomy to choose its own similarity measure between two expertise descriptions or between an expertise description and a query:

$$\sigma : (\mathcal{T}_e \cup \mathcal{T}_q, \mathcal{T}_e) \mapsto [0, 1] \quad (3)$$

where  $\mathcal{T}_e$  consists of sets of terms from expertise descriptions and  $\mathcal{T}_{eq}$  is the union of sets of expertise terms and abstracted query terms. In our simulations we assume that all peers in the network share documents about the same domain *dom* and therefore use the same term similarity matrix to determine the semantic similarity between terms. As said, future work should answer on what would be the effect of peers having different distance matrices from different domains and variations of them within the same domain. We instantiate  $\sigma$  in the following way:

$$\sigma(\bar{t}_{eq}, \bar{t}_e) = \frac{1}{|\bar{t}_{eq}|} \sum_{t_i \in \bar{t}_{eq}} \max_{t_j \in \bar{t}_{eq}, \bar{t}_e} (sim(t_i t_j)_{dom}) \quad (4)$$

where  $\bar{t}_e \in \mathcal{T}_e$  and  $\bar{t}_{eq} \in \mathcal{T}_e \cup \mathcal{T}_q$ . This formula takes the sum of the maxima of term similarities and divides it by the length of the expertise description  $\bar{t}_e$ . Note that this function is asymmetric,  $\sigma(\bar{t}_{eq}, \bar{t}_e) \neq \sigma(\bar{t}_e, \bar{t}_{eq})$ , because the elements in  $\bar{t}_{eq}$  are all relevant in the match algorithm. The elements in  $\bar{t}_e$  are only needed to determine how related  $\bar{t}_{eq}$  is to them. For example, imagine a query [**ferrari**, **mercedes**] and an expertise description [**car**, **environment**]. The similarity matrix probably has values like: (**ferrari**, **car**)=0.8, (**mercedes**, **car**)=0.9, (**ferrari**, **environment**)=0.3, (**mercedes**, **environment**)=0.3, (**car**, **environment**)=0.8, (**ferrari**, **mercedes**)=0.8. In this case,  $\sigma(\bar{t}_{eq}, \bar{t}_e) = \frac{1}{2} \times (0.8 + 0.9) = 0.85$ . This value is high because for both the 'mercedes' and 'ferrari', the expertise of 'car' is relevant (the elements in the expertise descriptions are connected via a logical OR). If the query would be **car**, **environment** and the expertise description **ferrari**, **mercedes**, the value is  $\frac{1}{2} \times (0.9 + 0.3) = 0.6$  which is much lower. This is reasonable because 'environment' is also part of the query (please recall that we assume that terms in the query are connected by a logical AND) and has only a low relevance to the terms 'ferrari' and 'mercedes' and therefore should lower the value.

### 3.2 Messages

There are three kinds of messages in our system needed to fulfill the functionalities of our algorithms:

- **Advertisement messages** These messages each contain a message identifier to prevent that a peer forwards the same message more than once, an expertise description, a list of peers that forwarded the message and the creator of the message. The purpose of making advertisements is that peers share expertise in order to increase the probability that peers in the network know the right peers to forward a query to. This should reduce the number of (forwarded) messages needed to satisfy a query. We have to find the optimum between two sides: (1) There is no advertising at all, which results in the fact that peers only know about others expertise if they received an answer on a query. This often results in many forwarded query messages. (2) Peers initialize and forward advertisements to all peers they know, which results in the fact that peers have a rich overview of knowledge in the network and almost no query forwarding is needed, but many (forwarded) advertisement messages. In our experiments we try to find some indication on what is a good balance between the two, but we think that a good trade-off mainly depends on how static the content is in the network.
- **Query messages** These messages each contain a message identifier, a query, the query abstraction, the origin of the message and a list of peers that forwarded the message.
- **Query result messages** These messages each contain the identifier of the original query message, the creator of the message and the query result (or pointers to it).

### 3.3 Policies

In this subsection, we describe the policies that concern how queries and advertisements are handled in our pRoute system. These policies are needed to fulfill the basic requirements of our SON based P2P system: distribution and acceptance of expertise descriptions, distribution of queries and dealing with joining and leaving nodes. For each policy we give instantiations that we tested in our simulations. More precisely, for each policy we provide one instantiation based on a random selection method and one that uses the shared similarity matrix in the selection process. This allows us to see when and in which degree using the shared similarity matrix is beneficial. For all policies it holds that when a message is sent to a peer that is off-line, the sender will select a new peer from the neighbor list. This means that in a very dynamic system and/or a network where not many peers are on-line, a significant number of messages will be lost. We record this in our statistics and just count them as sent messages.

*Query forwarding policy* Besides that a peer tries to answer a query, peers also select peers to forward the query via the query forwarding policy. This policy is

applied when a peer receives a forwarded query, or when a query is initialized by the peers user. We simulated the following instantiations:

- $QF1_{(h,n)}$ : Queries are maximally forwarded  $h$  hops, each step to maximally  $n$  random neighbors. We call this respectively the forwarding *depth* and forwarding *breadth*
- $QF2_{(h,n)}$ : Queries are maximally forwarded  $h$  hops, each step to maximally  $n$  neighbors where peers whose expertise descriptions are semantically closest to the terms in the query.

*Advertisement interval policy* This policy regulates at which moment the peer automatically starts to advertise its automatically generated expertise description. For all policies hold that when a peer (re)joins the network is advertises its expertise. More on this issue in the paragraph on the join/leave network policy.

- $AI1_{(r)}$  : Advertising is periodically triggered after  $r$  expertise description updates.
- $AI2_{(s)}$  : Advertising is only triggered when the similarity (see formula 4) between the expertise description from the last advertisement round and its new expertise description is lower than a certain threshold  $s$  (i.e. when the content of a peer has been significantly altered since the previous advertisement).

*Advertisement distribution policy* When the previous policy decides to advertise or when a peer receives an advertisement that it perhaps wants to forward, the advertisement distribution policy determines to whom to send the advertisements. This policy thus both applies to peers that want to advertise their own expertise and to peers that want to forward expertise descriptions to other peers.

- $AD1_{(h,n)}$  : In the first hop, advertisements are sent to all neighbors and from the second hop, advertisements are distributed to a random subset of  $n$  neighbors until a maximum of  $h$  hops.
- $AD2_{(h,n,t)}$  : In the first hop, advertisements are sent to those peers of which the expertise description has a similarity to the query above a threshold  $t$ . For two hops and more, the advertisements are sent to the  $n$  peers of which the expertise descriptions are semantically closest to the expertise description until a maximum of  $h$  hops.

*Advertisement acceptance policy* When a peer receives an advertisement it can decide to keep it or to ignore it. In the situation when the maximum number of  $n$  advertisements that a peer wants to store is not reached, we decide either to store every advertisement or the semantically close ones until all free advertisement slots are filled. After that, it can be the case that a new advertisement replaces an advertisement that exists in the receivers storage.

We tested the following two instantiations:

- $AA1_{(n)}$ : Only those advertisements are stored of which the expertise descriptions are semantically closest to the receivers’ own expertise description. This means that when the new advertisements description is closer than the worst description in the storage, it replaces this description.
- $AA2_{(n)}$ : A received advertisement randomly replaces one of the stored advertisements.

*Join/leave network policy* In our experiments we only use one constant network join and leave policy:

- *Joining*: when a peer joins the network we assume that it has an initial (random) list of gateway-peers, or has the list of peers that it had before it left the network. When the peer successfully joined the network it starts the advertising initialization policy.
- *Leaving*: when a peer leaves the network, by a network failure or intentionally initialized by the user, we do not assume any policy: the peer is just unreachable. We implemented the policy that when a peer sends a message to an off-line peer, it will recognize this and remove the peer from its neighbor list.

Now that we have introduced all elements of the pRoute system, we show a method to build a term similarity matrix which is one of these elements.

## 4 Method for making a Term Similarity Matrix

This section describes a method for automatically building a term similarity matrix.

Literal matching schemes suffer from synonyms and noise in documents. Latent Semantic Indexing (LSI) overcomes these problems by using statistically derived concepts instead of terms for retrieval. LSI uses Singular Value Decomposition (SVD) [9] to transform a high-dimensional document vector into a lower-dimensional semantic vector, by projecting the former into a semantic subspace. In this section we adopt a method using SVD to make a term similarity matrix on a certain domain which can be shared among a group of peers. This method also is used to calculate the shared matrix for our experiments.

Our method contains five steps:

1. **Get representative document set.** For text collections spanning many contexts (e.g., an encyclopedia), the number of terms is often much greater than the number of documents:  $t \gg d$ . However, in our case where the Internet is the document collection, the situation is very likely to be reversed: there are much more documents than terms. Even if we divide the set of documents in popular domains, it is likely that there are much more documents than items. In that case we only need a representative subset of the documents to get most terms used in that domain.

2. **Extract the keywords for each document.** Use a keyword extraction algorithm like those which are developed in the Natural Language Processing (NLP) research domain [8], to extract for each document a set of keywords which are the terms that describe the content of the documents. The size of the set depends on the trade-off between the size of the term-by-document matrix and the possibility to classify the assumed queries into one or more of the terms. Many keywords will lead to a big term-to-document matrix, for which it becomes computationally very expensive to calculate the term-to-term similarity matrix. However if there are many terms, the algorithm of abstracting a query into these terms will be very easy because many of the strings in the query (after stemming) match the terms in the matrix. Based on our experiences with the SVD tool 'SVDLIBC'<sup>3</sup>, we can say that for 100.000 documents containing together 200.000 terms it is still doable to do the calculations as described in step four of our method (AMD dual Athlon 3000+ 2G RAM roughly taking 1 hour)
3. **Create a term-by-document matrix.** Create a database containing the total  $d$  document sets described by  $t$  terms (the union of all document keyword sets) is represented as a  $t \times d$  *term-by-document matrix*  $\mathcal{A}$ . The  $d$  vectors representing the  $d$  document descriptions form the columns of the matrix. Thus, the matrix element  $a_{ij}$  is the weighted frequency at which term  $i$  occurs in document description  $j$ .
4. **Calculate the SVD of the term-by-document matrix.** Each document vector in the term-by-document matrix can be placed as a point in a large multi-dimensional *term space*. The Singular value decomposition (SVD) algorithm can be used to reduce the term space to a smaller number of dimensions. In doing so, terms that are semantically similar will get squeezed together, and will no longer be completely distinct. We re-use the description of [22] to describe the SVD algorithm. Let  $\mathcal{A}$  be a  $t \times d$  matrix as described in the previous step, whose entry  $a_{ij}$  indicates the importance of term  $i$  in document  $j$ . Suppose the rank of  $\mathcal{A}$  is  $r$ . SVD decomposes  $\mathcal{A}$  into the product of three matrices,  $\mathcal{A} = \mathcal{U}\Sigma\mathcal{V}^T$ , where  $\mathcal{U} = (u_1, \dots, u_r)$  is a  $t \times r$  matrix,  $\Sigma = \text{diag}(\alpha_1, \dots, \alpha_r)$  is an  $r \times r$  diagonal matrix, and  $\mathcal{V} = (v_1, \dots, v_r)$  is a  $d \times r$  matrix.  $\alpha_i$ 's are  $\mathcal{A}$ 's singular values,  $\alpha_1 \geq \alpha_2 \geq \dots \geq \alpha_r$ . Now the 'trick' of LSI is to reduce the dimensionality of the matrix by simply removing the singular values that have a low value. The result of this reduction is term clustering based on similarity, which we discuss in the next step. [4] give a good overview how such fundamental mathematical concepts from linear algebra can be used to manage and index large text collections.
5. **Use the SVD to generate the term-by-term similarity matrix** Before we explain how the SVD can be used for generating a term-by-term similarity matrix from  $\mathcal{A}$ , we first explain how it already can be done by simply comparing the term vectors in  $\mathcal{A}$ . The term-to-term comparison is

---

<sup>3</sup> SVDLIBC: <http://tedlab.mit.edu/~dr/SVDLIBC/> version 1.34 (2004)

carried out by computing the cosines of the angles  $\omega_{ij}$  between all pairs of term vectors  $i$  and  $j$ :

$$\cos \omega_{ij} = \frac{(e_i^T \mathcal{A})(\mathcal{A}^T e_j)}{\|\mathcal{A}^T e_i\| \|\mathcal{A}^T e_j\|} \quad (5)$$

where  $i$  and  $j$  is the number of terms in  $\mathcal{A}$  and where  $e_l$  denotes the  $l$ th canonical vector of dimension  $t$  (the  $l$ th column of the  $t \times t$  identity matrix). The cosines can now be listed in a  $t \times t$  symmetric identity matrix  $\mathcal{S}$ , where  $\mathcal{S}_{ij} = \cos \omega_{ij}$ . The entry  $\mathcal{S}_{ij}$  reveals how closely term  $i$  is associated with term  $j$ . If the entry is near 1, the term vectors for the two terms are nearly parallel meaning that the terms are closely correlated. The problem with this approach is that in our case  $\mathcal{A}$  is sparse because on average each document description only contains a small subset of the terms. This means that many entries in  $\mathcal{S}$  will be 0. Recall that our goal of the similarity matrix is to find terms that are related to other terms, which means that it would be better that for many terms a set of terms can be found which are at least not 0 but still reflect a correct similarity value. This would allow the advertisement- and query forwarding mechanisms to route messages to peers on which the expertise descriptions are close to the content of the received advertisement or query. This is where SVD becomes a good candidate, because reducing the rank of the matrix uncovers the associations among terms in a large collection of terms [9]. Because it falls beyond the scope of this paper to go into detail on LSI, we only show how this term matrix based on the reduced rank SVD is constructed. By the reduced-rank method,  $\mathcal{A}_k = \mathcal{U}_k \Sigma_k \mathcal{V}_k^T$  replaces the original term-by-document matrix  $\mathcal{A}$ , where  $k$  is the reduced number of singular values (and therefore the rank of the new matrix  $\mathcal{A}_k$ ). Recall that the columns of  $\mathcal{U}_k$  forms a basis for the column space of  $\mathcal{A}_k$ , and so those columns could be used in place of the columns of  $\mathcal{A}_k$  for query matching. In the same way, the rows of  $\mathcal{V}_k$  are a basis for the row space of  $\mathcal{A}_k$  and so can replace the rows of  $\mathcal{A}_k$ . Thus, in a reduced-rank approximation, the term-by-term can be constructed by calculating for each pair  $i, j$  the cosine [4] :

$$\cos w_{ij} = \frac{(e_i^T \mathcal{U}_k \Sigma_k \mathcal{V}_k^T)(e_j^T \mathcal{U}_k \Sigma_k \mathcal{V}_k^T)}{\|\mathcal{V}_k \Sigma_k \mathcal{U}_k^T e_i\|_2 \|\mathcal{V}_k \Sigma_k \mathcal{U}_k^T e_j\|_2} = \frac{(e_i^T \mathcal{U}_k \Sigma_k)(\Sigma_k \mathcal{U}_k^T e_j)}{\|\Sigma_k \mathcal{U}_k^T e_i\|_2 \|\Sigma_k \mathcal{U}_k^T e_j\|_2} \quad (6)$$

for  $i = 1, \dots, t$  and  $j = 1, \dots, d$ . Defining  $b_j = \Sigma_k \mathcal{U}_k^T e_j$ , we have

$$\cos w_{ij} = \frac{b_i^T b_j}{\|b_i\|_2 \|b_j\|_2} \quad (7)$$

for  $i = 1, \dots, t$  and  $j = 1, \dots, d$ .

In this section we proposed a method to automatically build a term similarity matrix, which is an important element in the pRoute system. In the next section we will show our experimental setup and introduce our simulation platform by which we test the characteristics of our approach.

## 5 Experimental Setup

We designed and implemented a simulation platform to test the performance of the different policies. As we will see, the semantics-based policies always outperform their random-based counterparts, although it will turn out to be situation dependent which combination of policies is the optimal one. For example, when the content of peers in the network remains fairly static, it is worthwhile to 'invest' in the visibility of peers by sending relatively many advertisement messages resulting in a reduction of the number of query forwards due to better peer selection. Also the recall rate that a user likes to see will turn out to be an important factor in selecting the policy.

### 5.1 Scenario and the data-set

Although the characteristics of our system are probably independent from the domain, we base our experiments on a realistic scenario where we can also easily get the needed data-set. Namely, we simulate a scenario in which our P2P system runs in a scientific domain where researchers share their publications with other researchers via a P2P network. This means that in our simulation researchers are represented by peers that share the publications of those researchers and publications that are interesting for them. We do this by letting the peers also include a (small) subset of the documents that are returned for queries posted by those peers. Each time when a peer adds a document of its user or includes a set of documents by the query process, the expertise description of that peer will be updated. The expertise descriptions will contain the most important terms of the researcher it represents, which reflects the content of the documents it shares. In this way we get a realistic overlap of document ownership and also a realistic growth of the document set over time. The queries that peers post are based on the content of those peers which supports the assumption that in our research scenario, the interests and expertise of users are closely related. We make the queries by selecting some terms which are in the document descriptions of that peer.

Our data set contains a crawled set of research articles from the computer-science domain sorted per author. The crawling procedure was first to extract the authors and titles from the DBLP resource<sup>4</sup>, a popular database containing bibliographic items, and use CiteSeer<sup>5</sup> to fetch the PDF document. We used the unix tool PDF2Text to extract the textual version of the PDF documents. Only when the tool was able to extract the text, the entry (author + text) is added to our dataset.

*Peer set, document description set* The document set contains 101,133 scientific documents in the computer science domain, namely a subset of those cited in the DBLP-database. The document crawl procedure is based on a breadth-first

---

<sup>4</sup> DBLP: <http://dblp.uni-trier.de/>

<sup>5</sup> CiteSeer: <http://citeseer.ist.psu.edu/>

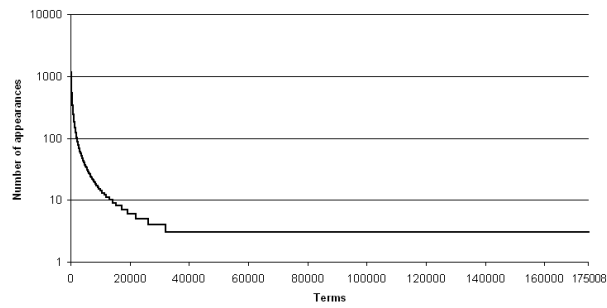
search over co-authors, starting by one author. More precisely, we started by crawling and storing all documents and co-authors from the author 'Maarten van Steen'. Next, for all the co-authors we crawled their documents and their co-authors and removed the doubles. This procedure is continued until a given maximum that we put on 82.054 peers. The characteristics of our data-set are as follows: Each document in our data-set has on average 2.71 authors (see Figure 3), and each author has on average 3.22 documents (see Figure 4). We used an NLP tool called 'TextToOnto' [14] to extract for each document a set of descriptive terms (around 10 terms per document, see Figure 2), which resulted in 175,008 unique terms.

*Shared similarity matrix* In our experiments we assume that peers all share the same term similarity matrix created via the methodology of the previous section. We used a random subset of the 101K document descriptions from the previous paragraph as a source for creating this similarity matrix. The number of documents that is selected clearly needs to cover most of the shared terms from the 101K documents. We define a term as a *shared term*, when it is shared by at least three peers (an arbitrary number). This resulted in a set of 25K shared terms (see Figure 1). Now that we have this shared term set, we have to find out how much we can reduce the document set before the number of shared terms decreases below an acceptable ratio. Also here we choose an arbitrary threshold of 90%, which means that we stop reducing the set when the reduced shared term set covers less than 90% of the original shared term set. In our case we could randomly reduce the document set by 90% which means that in our case we only needed 10K documents to get 92% of the shared terms. This reflects the realistic situation that the initial set of shared terms is created based on a small initial data-set. Now for each of the 10K documents a term vector is created of 25K elements, with value 1 at places where the term occurs in the document and value 0 (most of them) where the term does not occur. We use the method as described in the previous chapter to calculate the Term  $\times$  Term matrix, based on these 10K document vectors.

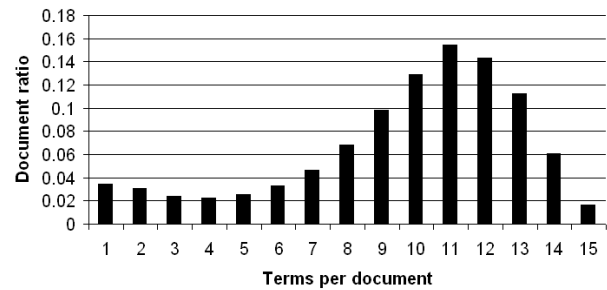
*Expertise descriptions* As said in previous sections, an expertise description describes the 'expertise' of a peer by a set of terms from the shared distance matrix. The expertise descriptions in our experiments are made by taking the union of all document descriptions of the peers and filter out the terms which are not in the distance matrix.

*Query abstractions* We create the set of query abstractions for a peer  $p$  by taking random sets of 3-5 terms from randomly chosen document descriptions  $d_{desc}$  owned by the peer, where the terms should also occur in the term matrix.

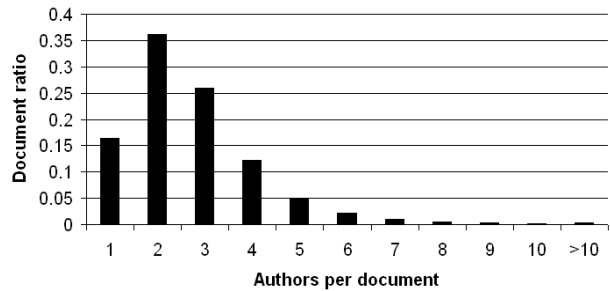
We believe that the creation of this data-set in itself is a contribution of this paper. The data-set is available from the authors on request.



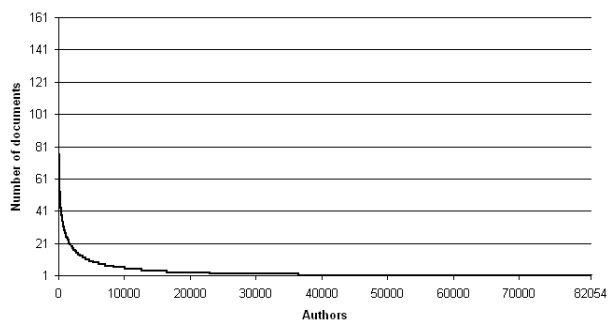
**Fig. 1. Term popularity.** This figure shows the distribution of the number of documents per term. There are a few terms that occur in many term vectors, and many terms only occur in three term vectors. Most terms (175008-32718=142290) have three matching vectors (we fixed this minimum to this value and discarded the terms that have less matching vectors).



**Fig. 2. Terms per document.** This figure shows the distribution of the number of terms per document(vector). Most documents have around 9-13 terms in its vector. We fixed the maximum number of terms to 15.



**Fig. 3. Authors per document.** This figure shows the distribution of the number of authors per document. Most documents have 1-4 authors.



**Fig. 4. Documents per author.** This figure shows the distribution of the number of documents per author. There are only a few authors that have more than 10 documents (in our data-set).

## 5.2 Simulation platform and Scenario Generator

We wrote a simulation platform to examine the different characteristics of our pRoute system. The platform is composed out of two elements: the 'scenario generator' and the 'scenario executor'. The generator is used to create a sequence of events to be used as input for the scenario executor. Having a different program to create the scenario yields several benefits compared to having a single program for both creating a scenario and applying it as input to our scenario executor:

- Firstly, comparing the output of the Peer-to-Peer system is more straightforward when exactly the same sequence of external events is used as input.
- Secondly, it yields performance gains when simulating several configurations of the Peer-to-Peer system.
- Finally, its output, the scenario, is in XML format, making the platform reusable and allowing user modifications to the scenario.

Each scenario script contains a large set of events from three different types, query events, document inclusion events and 'leave/join network' events, which normally would all be initiated by the users in a real network. A document inclusion event lets a peer include one of its own documents (i.e. of which the user is the author of that document) into its existing set of documents that it already shares on the network. A query event has the effect that a peer sends a query on the network which has to be logged for our statistics. But also it means that when documents are returned as an answer on the query, some of them will be included into the shared document set. A 'leave/join network' event has the effect that a peer temporally disconnects from the network. A system parameter in the scenario executor determines the time that the peer stays off-line. This means that after a certain amount of time, the peer goes back on-line. The scenario generator is described by algorithm 1, which we briefly describe as follows: In step 1, a set of document descriptions is generated from the complete

set of documents that possibly will be shared in the network. These descriptions are sets of terms that occur in the shared term matrix. Step 2 combines authors (represented as peer identifiers) with their documents. Step 3 initializes an empty set of scenario events. From step 4 to step 27, the set is iteratively filled with three different types of events until a maximum number of events (which is a system parameter, given in step 4) is reached or when the set of documents that a peer may add to the network is empty. The type of event that is added per iteration is determined by the *determineEventType()* function (step 8), which is a probabilistic function that decides if the next event that will be added to the script is a query event, a document inclusion event or a leave/join network event.

---

**Algorithm 1** Scenario script generator

---

```

1: Let  $D_{desc} := describe(D, S)$  be the set of document descriptions generated from
   the complete set documents  $D$  by terms occurring in the shared term similarity
   matrix  $S$ 
2: Let  $C := D_{desc} \times ID$  the set of tuples that combines each document description  $d \in
   D_{desc}$  with the peers (identified by peer identifiers  $p \in ID$ ) that have a document
   that matches the description
3: Let  $E := \emptyset$  be a list of scenario events
4: Let TotalNrOfEvents be the total number of events that the generated scenario
   script will contain.
5: while  $E.size < TotalNrOfEvents$  AND  $C \neq \emptyset$  do
6:   //select a random tuple from  $C$ 
7:    $[p, d] := getRandomTuple(C)$ 
8:    $eventType := determineEventType()$ 
9:   if  $eventType == events.query\_event$  then
10:    //get a random set of terms  $q$  from  $d$ 
11:     $q := getRandomSubset(d)$ 
12:    //create a query event for peer  $p$  and query  $q$   $E$ 
13:     $queryEvent := createQueryEvent(p, q)$ 
14:    //add the query event to  $E$ 
15:     $E := E.add(queryEvent)$ 
16:   if  $eventType == events.inclusion\_event$  then
17:    //create an inclusion event for peer  $p$  and document  $d$ 
18:     $inclusionEvent := createInclusionEvent(p, d)$ 
19:    //add the doc inclusion event to  $E$ 
20:     $E := E.add(inclusionEvent)$ 
21:    //remove the tuple from the  $C$ 
22:     $C := C \setminus [p, d]$ 
23:   if  $eventType == events.leave\_join\_network\_event$  then
24:    //create a leave/join network event for peer  $p$ 
25:     $leaveJoinEvent := createLeaveJoinEvent(p)$ 
26:    //add the leave/join network event to  $E$ 
27:     $E := E.add(leaveEvent)$ 
28: return  $E$ 

```

---

The scenario executor parses a scenario script which is generated by the scenario generator. Mechanisms that implement various policies can be easily added without modification to existing code. The simulator runs in a single process. Furthermore, it is capable of handling up to 100,000 Peers and hundreds of thousands of documents. Performance has been an important issue on the development of the simulator. We do not assume a reliable message delivery, which in our case is limited to the situations where peers send messages to peers that are not online. In such a case, besides that the message is counted by the statistics, the sender also knows that the message was not received. In that case the peer, to which the message was sent, is removed from the neighbor list. We assume the following simplified ordering of messages: a message only can be forwarded for the  $n_{th}$  time if all previous messages have been forwarded for the  $(n - 1)_{st}$  time. This assumption is made because it simplifies our simulator and considerably improves its execution time. We do not expect any significant changes in the results when the message delivery is more parallel.’

### 5.3 Evaluation criteria

We evaluate the different policies and the influence of other parameters by indicating user satisfaction (measured as document description recall), system efficiency and robustness. Please recall that each document and query is described by a set of terms from the shared term similarity matrix. Our precision and recall measures are based on these descriptions and not on the actual queries and documents. We expect that this is not a problem because the most important stemmed terms are in the 160K terms from the similarity matrix and many queries will contain (after stemming and removing the stop-words) these terms. Our evaluation is reflected by the following criteria:

– **Document description precision**

$$Precision_{Doc} = \frac{|D_{relevant} \cap D_{returned}|}{|D_{returned}|} \quad (8)$$

indicates how many of the returned document descriptions are relevant, with  $D_{relevant}$  being the total set of matching document descriptions in the network and  $D_{returned}$  being the set of returned document descriptions for queries. A document description is seen as relevant when all the terms in the query description occur in the document description. We determine the set of relevant documents  $D_{relevant}$  by evaluating the query against a centralized database which contains the complete set of document descriptions. In our model we work with exact queries, therefore only correctly matched documents are returned. The precision will therefore always be one, since  $D_{returned} \subseteq D_{relevant}$ :

$$Precision_{Doc} = \frac{|D_{returned}|}{|D_{returned}|} = 1 \quad (9)$$

and will therefore not be an evaluation criterion.

- **Document description recall**  $Doc_{rec}$

$$\frac{|P_{returned} \cap P_{relevant}|}{|P_{relevant}|} = \frac{|P_{returned}|}{|P_{relevant}|} \quad (10)$$

The recall on the document level states how many ( $|P_{returned}|$ ) of the total amount of relevant documents for the queries ( $|P_{relevant}|$ ) are returned and can be seen as an indication for user satisfaction. As we will see, the optimal policies differ for distinct recall levels. Therefore, recall level can also be seen as a requirement.

- **Peer precision**  $Peer_{prec}$  The peer precision is defined as:

$$\frac{|P_{relevant} \cap P_{queried}|}{|P_{queried}|} = \frac{|P_{relevant}|}{|P_{queried}|} \quad (11)$$

, which is the percentage of queried peers during the query process  $\mathbb{P}_{queried}$  which are 'correctly' queried  $\mathbb{P}_{relevant}$ , i.e. where at least one document of the peer was correctly matched. This measure can be seen as an indication for the efficiency of the search process.

- **Average number of messages per query**  $Q_{msg}$  The average number of messages used to resolve a query. It is an indication of the efficiency of the system.
- **Average number of advertisement messages per peer per experiment**  $A_{msg}$  The advertisement initialization policy decides if the peer should (re)advertise its expertise. Normally the advertisement procedure starts when the expertise description of a peer changes, for example caused by a significant change of content that a peer stores or when the peer goes back online. When a peer decides to advertise its expertise, its advertisement distribution policy decides to which peers the advertisements should be sent to. The average number of advertisement messages per peer during one simulation experiment is based on the multiplication of the total number of advertisement initializations and the number of messages per initialization. This number can be seen as the construction costs and maintenance costs of the semantic overlay for that experiment.

#### 5.4 A short note on the simulation method

When we finished our simulation platform and started different combinations of policies to get an idea of the behavior, we discovered that the results of the experiments are not only determined by the individual policies but also strongly depends on the combination of different policies. For example, an advertisement forward policy  $A1$  combined with a query forward policy  $Q1$  could give very good results like also advertisement forward policy  $A2$  combined with query forward policy  $Q2$ , but the combination of  $A1$  and  $Q2$  could give very bad results. This means that the way to combine policies is as important as the choice of the individual policies themselves. Due to the fact that simulating all different possibilities together with all different parameters for each individual policy would

lead to an unacceptable amount (millions) of experiments, we choose an iterative local search selection procedure. Namely, first we generated a pool of hundreds of experiments instantiated with some possible good combinations of policies and their parameters. These choices are made based on intuition and with the goal in mind of showing the influence of using semantics in the whole process. After that, we generated a new pool of experiments, by taking the best performing combinations of the previous step and make some changes in their settings to see how dependent the result was on the individual parameters. In this way we cannot guarantee that we found the best combination of policies together with their optimal settings, however at least can show a variety of experiments showing the difference between policies using semantics in the advertisement and querying process and those base on random behavior. In Table 1 the default settings are shown for the different experiments. Sometimes we deviate from these settings, and if so, we mention these alternated settings.

description	default value
total number of peers in the system	27,351
average peer availability ( $\alpha$ )	40%
average nr of reconnects per peer ( $\beta$ )	75

**Table 1.** Default parameters in experiments

Now that we have shown how our simulation platform, data-set and evaluation criteria look like, we can present the results of our experiments in the next section.

## 6 Results

*Random-based strategies* Table 2 shows how the system performs when it uses random query- and advertisement strategies and can be seen as our baseline settings.

- Advertisements are needed to make peers visible in the network. We only tested the settings where the number of advertisement hops are two or three. We skipped the one-hop experiments because we can already know that these will perform bad. Namely, each peer knows in the beginning a set of random pointers (peers) to which it can send its advertisements. If the receiving peer does not forward the advertisements, the topology always will be the initial topology which gets outdated after a while. Which means that a new set of pointers has to be found (e.g. downloaded from a web-site) somewhere, which is an undesirable option due to the centralized characteristics of it. When we compare the recall from experiments 1,2,3 with experiments 4,5,6 (where we increase the advertising depth), we can see a 20% drop in the recall.

This is because the visibility of the peers that send the latest advertisements becomes too strong because their advertisements will be sent to a unacceptable large number of peers. This results in less visibility of other peers that sent previously advertisements. For example in experiment 6, there are 1092 messages sent per advertisement initialization. Given that on average there are 20000 peers online, the advertisement storages of the individual peers will be updated too often. Therefore, only the last advertisements sent in the network will be stored, resulting in a smaller fraction of the peers being known. Summarizing: be careful with the number of advertisement messages. When there are too much forwards of the same advertisement, not only will the network traffic be increased, it will also result in too many existing advertisements being replaced by it, thus making peers either too visible or too little visible.

- A second observation is that, besides the recall, the precision increases when there are more query hops. This is because when there are more query hops, more results will be returned. Due to the fact that in our simulation some of the query results are added to the document of the querying peer, it will lead to more copies of the documents which automatically increases the chance that a peer contains an answer for a query. We deliberately keep this inclusion rate small to keep this effect into reasonable boundaries.

*Random-based query routing combined with semantic-based advertisements* Table 3 shows the results of combining the random-based query routing strategy with the semantic-based advertisement strategies.

- The results of experiment 7 to 11 show that replacing random strategies by their semantic strategies increases the performance, however the gain differs per choice. For example, it seems that it is better to choose a semantic advertisement acceptance policy instead of a semantic advertisement distribution policy (compare experiment 8 with 9). Choosing both seems to be the best solution (experiment 10). Experiment 11, compared with experiment 10, shows that doing semantic advertisement initialization reduces the number of advertisements but also the recall and precision. These results indicate that replacing random policies by their semantic counterparts has a positive effect.
- Experiment 11, 12 and 13 show the influence of different query hops on the number of query messages and the recall. They indicate that in order to double the recall, the number of query messages need to be more than tripled.
- Experiment 11, 14 and 15 indicate that there is a positive effect of the number of advertisements on the recall and precision. This confirms the positive effect of clustering peers with the same expertise: the recall and precision of this clustering is higher because in our system peers normally ask questions related to their expertise and therefore can benefit from the fact that neighbors have already a good chance to answer the query.

Nr	Simulation setting			$P_{prec}$	$Doc_{rec}$	$Q_{msg}$	$A_{msg}$
	$QF1$	$AI1$	$ADI$				
1	$QF1_{(3,3)}$	$AI1_{(1)}$	$ADI_{(2,5)}$	0.0076	0.0093	44	214
2	$QF1_{(4,3)}$	$AI1_{(1)}$	$ADI_{(2,5)}$	0.0082	0.0326	133	211
3	$QF1_{(5,3)}$	$AI1_{(1)}$	$ADI_{(2,5)}$	0.0085	0.1013	384	214
4	$QF1_{(3,3)}$	$AI1_{(1)}$	$ADI_{(3,5)}$	0.0070	0.0086	42	1121
5	$QF1_{(4,3)}$	$AI1_{(1)}$	$ADI_{(3,5)}$	0.0072	0.0281	124	1099
6	$QF1_{(5,3)}$	$AI1_{(1)}$	$ADI_{(3,5)}$	0.0078	0.0835	357	1092

**Table 2. Random query routing with random advertising.** For the meaning of the simulation settings, please look at Subsection 3.3

- Experiment 16 to 22 show the influence of the number of neighbors that a peer knows. The results indicate that this factor is important for the performance of the system. Namely a too small number of neighbors makes that a peer cannot remember a sufficient set of the relevant peers. However the number can also be too large. Namely this makes that a peer stores relatively too many irrelevant peer advertisements, which increases the chance that, due to the random query forwarding process, peers are selected that have no expertise on the query (experiment 22).

*Semantic query routing combined with random-based advertisement strategies*

Table 4 shows the results of combining the semantic-based routing strategy with the random-based advertisement strategy. The results indicate that this combination outperforms the previous two combinations. Due to this random-based advertising, there is no clustering of expertise, meaning that each peer only knows a random set of peers where a new advertisement replaces randomly an old advertisement. The drop in recall for experiments with a high number of hops (compare experiment 26 with 27) is likely to have the same reason as given at the experiments that combines random advertisement and query strategies. Namely, peers that recently advertised are unacceptably oppressing the other peers that have advertised before due to the enormous amount of copies of the same advertisement caused by the large number of forwards. As before, this shows that a careful balancing of parameters is required.

*Semantic query routing, semantic advertisements* Table 5 shows the results of combining semantic query policies with semantic advertisement policies. The results indicate that this combination outperforms all the previous three combinations.

- Experiments 28 to 33 show the effect of changing the random policies from the advertisement process by their semantic counterparts. Result 33 indicates that by replacing all of the random policies by semantic ones, a good recall can be achieved (42%) together with a low number of advertisement and query messages.
- Experiments 33 and 34 show that changing the similarity threshold in the advertisement initialization policy increases the recall but also the number of advertisement messages.
- Experiments 35 and 36 show the effect of playing with the number of query hops. The difference between two and three hops (experiment 32) results in this case in a recall increase from 27% to 42%. By doubling the number of messages again by increasing the number of hops from three to four (experiment 36), the recall increases to almost 50%.
- The effect of increasing the number of advertisements is shown in result 37 and 38. Although there is a positive effect (compare result 37 with 32), it seems that the topology gets saturated at a certain point because the difference in recall between result 32 and 38 is almost the same although the number of advertisements almost doubled.

Nr	Simulation setting			$P_{prec}$	$Doc_{rec}$	$Q_{msg}$	$A_{msg}$
7	$QF1_{(4,3)}$	$AI2_{(0,8)}$	$ADI_{(3,5)}$	0.0059	0.0177	125	969
8	$QF1_{(4,3)}$	$AI1_{(1)}$	$AD2_{(3,5,0,2)}$	0.0054	0.0130	142	412
9	$QF1_{(4,3)}$	$AI1_{(1)}$	$ADI_{(3,5)}$	0.0245	0.1103	123	1021
10	$QF1_{(4,3)}$	$AI1_{(1)}$	$AD2_{(3,5,0,2)}$	0.0181	0.0809	136	464
11	$QF1_{(4,3)}$	$AI2_{(0,8)}$	$AD2_{(3,5,0,2)}$	0.0148	0.0647	137	422
12	$QF1_{(3,3)}$	$AI2_{(0,8)}$	$AD2_{(3,5,0,2)}$	0.0147	0.0309	47	423
13	$QF1_{(5,3)}$	$AI2_{(0,8)}$	$AD2_{(3,5,0,2)}$	0.0130	0.1299	391	403
14	$QF1_{(4,3)}$	$AI2_{(0,8)}$	$AD2_{(2,5,0,2)}$	0.0073	0.0192	126	55
15	$QF1_{(4,3)}$	$AI2_{(0,8)}$	$AD2_{(4,5,0,2)}$	0.0188	0.0904	132	807
16	$QF1_{(4,3)}$	$AI2_{(0,8)}$	$ADI_{(3,5)}$	0.0119	0.0249	51	41
17	$QF1_{(4,3)}$	$AI2_{(0,8)}$	$ADI_{(3,5)}$	0.0188	0.0580	73	87
18	$QF1_{(4,3)}$	$AI2_{(0,8)}$	$ADI_{(3,5)}$	0.0232	0.0795	86	121
19	$QF1_{(4,3)}$	$AI2_{(0,8)}$	$ADI_{(3,5)}$	0.0240	0.1137	111	309
20	$QF1_{(4,3)}$	$AI2_{(0,8)}$	$ADI_{(3,5)}$	0.0250	0.1257	115	413
21	$QF1_{(4,3)}$	$AI2_{(0,8)}$	$ADI_{(3,5)}$	0.0158	0.0887	118	577
22	$QF1_{(4,3)}$	$AI2_{(0,8)}$	$ADI_{(3,5)}$	0.0085	0.0395	139	373

Table 3. Random query routing with semantic-based advertising policies

Nr	Simulation setting			$P_{prec}$		$Q_{msg}$	$A_{msg}$	
				$Doc_{rec}$				
23	$QF2_{(3,3)}$	$AI_{(1)}$	$ADI_{(3,5)}$	$AAI_{(80)}$	0.0669	0.1213	40	1104
24	$QF2_{(4,3)}$	$AI_{(1)}$	$ADI_{(3,5)}$	$AAI_{(80)}$	0.0668	0.2299	105	1104
25	$QF2_{(5,3)}$	$AI_{(1)}$	$ADI_{(3,5)}$	$AAI_{(80)}$	0.0674	0.3454	211	1084
26	$QF2_{(4,3)}$	$AI_{(1)}$	$ADI_{(2,5)}$	$AAI_{(80)}$	0.0637	0.2724	113	213
27	$QF2_{(4,3)}$	$AI_{(1)}$	$ADI_{(4,5)}$	$AAI_{(80)}$	0.0665	0.1981	98	3486

**Table 4.** Semantic query routing with random-based advertising policies

Nr	Simulation setting			$P_{prec}$		$Q_{msg}$	$A_{msg}$	
					$D_{crec}$			
28	$QF2_{(4,3)}$	$AI2_{(0,8)}$	$ADI_{(3,5)}$	$AAI_{(80)}$	0.0594	0.2181	110	959
29	$QF2_{(4,3)}$	$AI1_{(1)}$	$AD2_{(3,5,0.2)}$	$AA1_{(80)}$	0.1052	0.4036	109	401
30	$QF2_{(4,3)}$	$AI1_{(1)}$	$ADI_{(3,5)}$	$AA2_{(80)}$	0.1792	0.3583	64	1017
31	$QF2_{(4,3)}$	$AI1_{(1)}$	$AD2_{(3,5,0.2)}$	$AA2_{(80)}$	0.1308	0.4383	98	459
32	$QF2_{(4,3)}$	$AI2_{(0,8)}$	$ADI_{(3,5)}$	$AA2_{(80)}$	0.1592	0.4003	78	890
33	$QF2_{(4,3)}$	$AI2_{(0,8)}$	$AD2_{(3,5,0.2)}$	$AA2_{(80)}$	0.1203	0.4215	100	398
34	$QF2_{(4,3)}$	$AI2_{(0,8)}$	$AD2_{(3,5,0)}$	$AA2_{(80)}$	0.1381	0.4660	95	573
35	$QF2_{(3,3)}$	$AI2_{(0,8)}$	$AD2_{(3,5,0.2)}$	$AA2_{(80)}$	0.1251	0.2744	40	410
36	$QF2_{(5,3)}$	$AI2_{(0,8)}$	$AD2_{(3,5,0.2)}$	$AA2_{(80)}$	0.1086	0.4984	198	401
37	$QF2_{(4,3)}$	$AI2_{(0,8)}$	$AD2_{(2,5,0.2)}$	$AA2_{(80)}$	0.0644	0.2548	107	60
38	$QF2_{(4,3)}$	$AI2_{(0,8)}$	$AD2_{(4,5,0.2)}$	$AA2_{(80)}$	0.1274	0.4383	95	761

Table 5. Semantic query routing with semantic-based advertising policies

*Influence of the number of neighbors* Table 6 shows the effect of varying the number of expertise descriptions (also called the number of neighbors) that a peer can store. As it can be seen, an increase improves the recall and precision, without an increase in the number of query messages although with an increase of the number of advertisements. This increase is because the number of neighbors selected at the first advertisement hop depends on the number of neighbors in the storage: the more neighbors, the bigger the chance that peers are above the similarity threshold. The trade-off between recall and the number of messages is of course dependent on the requirements of the user of the system. However, the results indicate that the positive effect on the recall by sending more advertisement messages in combination with a larger neighbor storage starts to diminish after 60 neighbors (compare experiment 42 with 43). Probably this result is dependent on the number of peers in the network, where the trade-off in larger networks lies at a higher number of peers.

*Influence of the network size* Table 7 shows the results for varying the network size (in the other tables, the network size is fixed at 27K peers, see Table 1). The results indicate the precision almost stays the same for different network sizes. That the recall drops a bit is because more peers means more content, which means that more peers needs to be visited to get the relevant results. In other words, given that the maximum number of query forwards stays the same during all the shown experiments, it can be derived that the recall should decrease. Experiment 49 shows a large decrease in recall and advertisement messages compared to experiment 44 to 48 which has the following cause: When there are not enough advertisement hops, the relevant experts for the advertiser (i.e. the peers that have similar expertise as the sender) are not reached at an advertisement phase. This results in the situation that both sides will not know each-other (the advertising peer and the related experts). This is because when the peers that are reached at the advertisement phase are not relevant they will not 'remember' the advertising peer and therefore the peer will be forgotten. Thus, more advertisements are needed to reach the relevant of the related peers, so that they will remember the advertising peers and will send advertisement messages to it too. As we can see, there is no linear decrease in recall and number of advertisement messages. The reason that there are less advertisements is that the initial random list of peers is the only set that a peer will know (because it receives no advertisements of peers due to the reason given before) together with the fact that peers are removed from the neighbor list if a messages is sent and the receiver is off-line, a peer's neighbor list will get smaller and smaller. Experiment 50 shows that one extra advertisement hop solves the problem. We did not perform experiments to find the 'drop-point' for these three hops, but probably it will lie in a network with more than 1M peers. Another solution that partly solves the problem of the enduring reduction of the neighbor list, which we did not test, is to have gateway peers that provide new pointers to random peers when a peer has not enough neighbors anymore.

Nr	Simulation setting			$P_{\text{err}_{\text{prec}}}$	$D_{\text{OC}_{\text{rec}}}$	$Q_{\text{msg}}$	$A_{\text{msg}}$	
39	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(3,5,0.2)}$	$AA2_{(10)}$	0.0359	0.1029	72	81
40	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(3,5,0.2)}$	$AA2_{(20)}$	0.0701	0.2400	90	161
41	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(3,5,0.2)}$	$AA2_{(40)}$	0.0995	0.3665	99	274
42	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(3,5,0.2)}$	$AA2_{(60)}$	0.1135	0.4077	99	350
43	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(3,5,0.2)}$	$AA2_{(160)}$	0.1256	0.4419	100	436
39	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(2,5,0.2)}$	$AA2_{(10)}$	0.0100	0.0189	53	10
40	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(2,5,0.2)}$	$AA2_{(20)}$	0.0235	0.0745	78	19
41	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(2,5,0.2)}$	$AA2_{(40)}$	0.0447	0.1648	97	37
42	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(2,5,0.2)}$	$AA2_{(60)}$	0.0560	0.2210	103	48
43	$QF2_{(4,3)}$	$AI2_{(0.8)}$	$AD2_{(2,5,0.2)}$	$AA2_{(160)}$	0.0709	0.2811	111	68

Table 6. Influence of number of neighbors

Nr	$N_{r_p}$	Simulation setting	$Peer_{prec}$	$Doc_{rec}$	$Q_{msg}$	$A_{msg}$
44	5K	$QF2_{(4,3)}$ $AI2_{(0,8)}$ $AD2_{(2,5,0.2)}$ $AA2_{(80)}$	0.0601	0.3895	99	36
45	10K	$QF2_{(4,3)}$ $AI2_{(0,8)}$ $AD2_{(2,5,0.2)}$ $AA2_{(80)}$	0.0576	0.3157	106	44
46	20K	$QF2_{(4,3)}$ $AI2_{(0,8)}$ $AD2_{(2,5,0.2)}$ $AA2_{(80)}$	0.0609	0.2433	105	55
47	27K	$QF2_{(4,3)}$ $AI2_{(0,8)}$ $AD2_{(2,5,0.2)}$ $AA2_{(80)}$	0.0586	0.2125	102	58
48	40K	$QF2_{(4,3)}$ $AI2_{(0,8)}$ $AD2_{(2,5,0.2)}$ $AA2_{(80)}$	0.0584	0.1682	96	61
49	80K	$QF2_{(4,3)}$ $AI2_{(0,8)}$ $AD2_{(2,5,0.2)}$ $AA2_{(80)}$	0.0265	0.0374	60	24
50	80K	$QF2_{(4,3)}$ $AI2_{(0,8)}$ $AD2_{(3,5,0.2)}$ $AA2_{(80)}$	0.1092	0.3497	107	538

Table 7. Influence of the maximum number of peers in the network for our best performing settings

*Influence of availability* Table 8 shows the results of varying the availability of peers for a pure random query- and advertisement policy and for the semantic-based policies (in the other tables, the availability is size is fixed at 40%, see 1). As can be seen, the performance is very dependent on the availability. When the availability is 3%, our semantic-based approach has the same (bad) performance as the random-based approach (compare experiment 51 with 56). However when the availability is 26%, the recall of the semantic-based approach performs already 10 times better than the random approach in terms of recall. It is likely that this difference will even be higher in larger networks because random networks will have a linear decrease in recall when the network grows linear and Table 7 shows that the semantic-based approach does not have the linear decrease.

## 7 Conclusions

In this paper we presented a Peer-to-Peer system where the nodes describe their content in terms which occur in a shared similarity matrix. The peers share their content descriptions with other peers where peers remember only those peers that are relevant (i.e. semantically close) to their own content. In this way, peers form a semantic overlay network. We have shown how the model can be applied in a bibliographic scenario based on a realistic data-set. We have shown a method to automatically make a term similarity matrix which can be shared among a group of peers. Simulation experiments that we performed with this bibliographic scenario where all peers share the same distance matrix show that it performs much better in recall and the reduction of the number of messages compared to a random approach. Our results are based on a large data-set and are comparable with the work of [11], except that our shared data-structure is richer and created automatically. We have shown that our system is scalable and robust to peer drops, although the availability of peers should not be too low. Future work has to solve the problem of dealing with very low availabilities, most likely with the introduction of a complementary protocol to renew neighbor caches.

## 8 Acknowledgements

We thank Frank van Harmelen for his very useful comments and suggestions. We thank Spyros Voulgaris for his help to crawl the dataset that we used for our experiments. The work was partially supported by the the European funded SWAP project (IST-2001-34103).

## References

1. K. Aberer. P-grid: A self-organizing access structure for p2p information systems. In *Proceedings of the Sixth International Conference on Cooperative Information*

Nr	$\alpha$	Simulation setting			$P_{peer\ prec}$		$Doc_{rec}$	$Q_{msg}$	$A_{msg}$
51	$\underline{3\%}$	$QF1_{(4,3)}$	$AI1_{(1)}$	$AD1_{(3,5)}$	$AA1_{(80)}$	0.0105	0.0083	12	32
52	$\underline{13\%}$	$QF1_{(4,3)}$	$AI1_{(1)}$	$AD1_{(3,5)}$	$AA1_{(80)}$	0.0080	0.0313	89	392
53	$\underline{26\%}$	$QF1_{(4,3)}$	$AI1_{(1)}$	$AD1_{(3,5)}$	$AA1_{(80)}$	0.0080	0.0331	116	757
54	$\underline{40\%}$	$QF1_{(4,3)}$	$AI1_{(1)}$	$AD1_{(3,5)}$	$AA1_{(80)}$	0.0072	0.0281	124	1099
55	$\underline{60\%}$	$QF1_{(4,3)}$	$AI1_{(1)}$	$AD1_{(3,5)}$	$AA1_{(80)}$	0.0072	0.0242	123	1442
56	$\underline{3\%}$	$QF2_{(4,3)}$	$AI2_{(0,8)}$	$AD2_{(3,5,0,2)}$	$AA2_{(80)}$	0.0169	0.0084	11	14
57	$\underline{13\%}$	$QF2_{(4,3)}$	$AI2_{(0,8)}$	$AD2_{(3,5,0,2)}$	$AA2_{(80)}$	0.0563	0.1705	72	105
58	$\underline{26\%}$	$QF2_{(4,3)}$	$AI2_{(0,8)}$	$AD2_{(3,5,0,2)}$	$AA2_{(80)}$	0.0933	0.3803	100	240
59	$\underline{40\%}$	$QF2_{(4,3)}$	$AI2_{(0,8)}$	$AD2_{(3,5,0,2)}$	$AA2_{(80)}$	0.1203	0.4215	100	398
60	$\underline{60\%}$	$QF2_{(4,3)}$	$AI2_{(0,8)}$	$AD2_{(3,5,0,2)}$	$AA2_{(80)}$	0.1483	0.4422	96	521

Table 8. Influence of peer availability

- Systems (CoopIS'01)*, volume 2172 of *Lecture Notes in Computer Science*, Trento, Italy, 2001. Springer Verlag.
2. K. Aberer, P. Cudré-Mauroux, M. Hauswirth, and T. V. Pelt. Gridvine: Building internet-scale semantic overlay networks. In *3rd International Semantic Web Conference (ISWC2004)*, pages 107–121, Hiroshima, Japan, 7-11 November 2004.
  3. R. J. Bayardo, Jr., W. Bohrer, R. Brice, A. Cichocki, J. Fowler, A. Helal, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, . . . Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh, and D. Woelk. InfoSleuth: Agent-based semantic integration of information in open and dynamic environments. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, volume 26,2, pages 195–206, New York, NY, USA, 1997. ACM Press.
  4. M. W. Berry, Z. Drmac, and E. R. Jessup. Matrices, vector spaces, and information retrieval. In *SIAM Review*, pages 335–362, 1999.
  5. L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: Evidence and implications. In *Proceedings of the IEEE INFOCOM conference*, pages 126–134, New York, USA, March 1999.
  6. J. Byers, J. Considine, and M. Mitzenmacher. Simple load balancing for distributed hash tables. Technical report, CS Department, Boston University, November 2002.
  7. E. Cohen, A. Fiat, and H. Kaplan. Associative search in peer to peer networks: Harnessing latent semantics. In *Proceedings of the IEEE INFOCOM conference*, San Francisco, CA, USA, march 2003.
  8. H. Cunningham, Y. Wilks, and R. Gaizauskas. New methods, current trends and software infrastructure for nlp. In *Proceedings of the Second Conference on New Methods in Language Processing*, pages 283–298, Ankara, Turkey, 1996.
  9. S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.
  10. T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an Agent Communication Language. In N. Adam, B. Bhargava, and Y. Yesha, editors, *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*, pages 456–463, Gaithersburg, MD, USA, 1994. ACM Press.
  11. P. Haase, R. Siebes, and F. van Harmelen. Peer selection in peer-to-peer networks with semantic topologies. In M. Bouzeghoub, editor, *Proceedings of the International Conference on Semantics in a Networked World (ICNSW'04)*, volume 3226 of *LNCS*, pages 108–125, Paris, France, June 2004. Springer Verlag.
  12. G. Kan. Gnutella. In A. Oram, editor, *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, pages 94–122. O'Reilly and Associates, 2001.
  13. N. Leibowitz, M. Ripeanu, and A. Wierzbicki. Deconstructing the kazaa network. In *3rd IEEE Workshop on Internet Applications (WIAPP'03)*, Santa Clara, CA, 2003.
  14. A. Maedche and S. Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2):72–79, 2001.
  15. S. McIlraith, T. Son, and H. Zeng. Semantic web services. In *IEEE Intelligent Systems (Special Issue on the Semantic Web)*, volume 16, pages 46–53, 2001.
  16. W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmer, and T. Risch. Edutella: A p2p networking infrastructure based on rdf. In *Proceedings of the 11th International World Wide Web Conference*, Honolulu, Hawaii, USA, May 2002.
  17. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM*, San Diego, CA, USA, august 2001. ACM Press.

18. 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.
19. M. T. Schlosser, M. Sintek, S. Decker, and W. Nejdl. Hypercup - hypercubes, ontologies, and efficient search on peer-to-peer networks. In *International Workshop on Agents and Peer-to-Peer Computing (AP2PC'02)*, pages 112–124, Bologna, Italy, July 2002.
20. K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *Proceedings of the IEEE INFOCOM conference*, San Fransisco, CA, USA, march 2003.
21. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of ACM SIGCOMM conference*, San Diego, CA, USA, august 2001. ACM Press.
22. C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. In *Proceedings of the ACM SIGCOMM Conference*, Karlsruhe, Germany, August 2003. ACM Press.
23. S. Voulgaris, A.-M. Kermarrec, L. Massoulie, and M. van Steen. Exploiting semantic proximity in peer-to-peer content searching. In *Proceedings of the 10th International Workshop on Future Trends in Distributed Computing Systems (FT-DCS'04)*, Suzhou, China, May 2004.