REMINDIN': Semantic Query Routing in Peer-to-Peer Networks based on Social Metaphors

Christoph Tempich
Institute AIFB, University of Karlsruhe
D-76128 Karlsruhe, Germany
tempich@aifb.uni-karlsruhe.de

Steffen Staab
Institute AIFB, University of Karlsruhe
D-76128 Karlsruhe, Germany
staab@aifb.uni-karlsruhe.de

Adrian Wranik*
Institute AIFB, University of Karlsruhe
D-76128 Karlsruhe, Germany
A.Wranik@isi.fraunhofer.de

ABSTRACT

In peer-to-peer networks, finding the appropriate answer for an information request, such as the answer to a query for RDF(S) data, depends on selecting the right peer in the network. We here investigate how social metaphors can be exploited effectively and efficiently to solve this task. To this end, we define a method for query routing, REMINDIN', that lets *(i)* peers observe which queries are successfully answered by other peers, *(ii)*, memorizes this observation, and, *(iii)*, subsequently uses this information in order to select peers to forward requests to.

REMINDIN' has been implemented for the SWAP peer-to-peer platform as well as for a simulation environment. We have used the simulation environment in order to investigate how successful variations of REMINDIN' are and how they compare to baseline strategies in terms of number of messages forwarded in the network and statements appropriately retrieved.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Abstracting methods*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Selection process*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Information networks*

General Terms

Algorithms, Experimentation, Performance

Keywords

Ontologies, Peer-to-Peer, Peer Selection, Query Routing

1. INTRODUCTION

In spite of the success of distributed systems like the World Wide Web, a large share of today's information available on computers is not made available to the outside, but it remains secluded on personal computers stored in files, emails and databases — information that we will call *PC data* in the following. In theory, peer-to-peer networks are ideally suited to facilitate PC data exchange between peers. In practice, however, there remain unsurmountable obstacles:

1. PC data constitutes an *open domain*. Though one can define some core schema, e.g. as has been done for learning object metadata (LOM[1]; [16]), the core schema needs to be extended frequently.

2. Peers do not know *where* to find information.

3. Deciding *what* information about other peers to maintain is difficult, because relevance of data is hard to assess and possibilities for duplication are limited.

For some of these individual problems solutions have been found: For instance, haystack has shown that PC data can be nicely managed via RDF as it supports a flexible semi-structured data model [22]. Current search engines show how to find information. Current applications, such as TAP [15] show how to handle text as well as semi-structured data. Then, full text search indices can be maintained via centralized indices or through P2P exchange of indices [8]. Also, for fixed schemata algorithms exist that allow the finding of relevant information with only local knowledge [12].

Together, however, the requirements just given above overstretch the possibilities that current peer-to-peer systems offer. **Our contribution** described here is a peer-to-peer system that easily accommodates various semantic descriptions, that organizes itself in a way such that local knowledge is sufficient to localize data sources and that maintains its knowledge in a non-obtrusive manner based on what is observed as answers by other peers.

In brief, what we have conceived is a query routing capability that mimicks what a person is doing in a social network:

- she retains meta-information about what other peers know;
- she might not even ask the others about their knowledge, but observe it from communication;
- she does not have a fixed schema, but easily builds up new schematic or taxonomic knowledge structure;
- she then decides to ask one or a few peers based on how she estimates their coverage and reliability of information about particular topics.

*Research for this work was performed while A. Wranik was a master's student at AIFB.

WWW2004, May 17–22, 2004, New York, NY USA.
ACM xxx.xxx.

[1]http://kmr.nada.kth.se/el/ims/md-lomrdf.html

To this end, we have implemented a general P2P platform, the **SWAP platform** (Semantic Web And Peer-to-peer) and for this platform we have developed an original algorithm, **REMINDIN' (Routing Enabled by Memorizing INformation about Distributed INformation)**, that

1. Selects (at most) two peers from a set of known peers based on a given triple query, hence avoids network flooding;
2. Forwards the query; and
3. Assesses and retains knowledge about which peer has answered which queries successfully.

In contrast to, e.g., [2, 20], this is a lazy learning approach [3] that does not advertise peer capabilities upfront, but that estimates it from observation — the main advantage being that a dynamic semantic topology is made possible by adapting to user queries.

We evaluate the algorithm on a simulation platform with a structure that is aligned to the structure of the original system. Thereby, we evaluate the hypotheses that

1. REMINDIN' is advantageous for effective query routing to estimate capabilities from observation of queries. In particular, this effect is achieved as meta-information is accumulated over time;
2. REMINDIN' can accommodate for changes when the typical information being available and queried changes;
3. REMINDIN's use of background knowledge further improves effectiveness.

We conclude with a survey of related work and an embedding of our work into some overall objectives for self-organizing information systems.

2. SCENARIOS

The EU IST project SWAP (Semantic Web And Peer-to-peer) features two case studies. Both build on top of the SWAP platform exploiting its principal features as outlined in Section 3.

The IBIT case study is about sharing of databases and documents between cooperating tourism organization on the Baleares, a group of Spanish islands in the Mediterranean. Some of the features of the case study include that the definition of a unique global schema or ontology is not possible, that their topics are under drift and that the knowledge management support to be provided by the peer-to-peer must deal with this flexibility.

In SWAP Bibliography case study, we will explore the sharing of Bibtex information between peers of researchers. Bibtex will be locally harvested from files and stored in the SWAP *local node repository*. Then one may search on the own peer as well as in the network in order to retrieve the appropriate bibliographic data. This scenario is particularly interesting for further investigation, because *(i)* it is small enough to be realistic and successful; *(ii)* bibtex data have a stable interesting core, but also greatly varying additional fields as each user may define his own bibtex entries; *(iii)* bibtex data can never be fully captured in a centralized repository, because one repository such as DBLP can only reflect a small set of topics (e.g., databases and AI, but not organizational issues of knowledge management).

To make such case studies realistic it is necessary to effectively and efficiently locate the appropriate peer that can answer particular questions. In the following, we first survey the SWAP platform, then we describe the algorithm we have conceived to solve the localization problem, viz. REMINDIN'.

Table 1: Data structures and parameters

Data structures	
O	Local node repository (ontology)
Q	Query
MO	Meta data object for a specific peer and resource
Configuration Parameter	
$p_{max} \in \mathbb{N}^+$	The maximum number of peers selected for query forwarding
$h_{max} \in \mathbb{N}^+$	The maximum number of hops a query is allowed to travel (the horizon of the query)
$\mu \in \mathbb{R}^+ \wedge \mu \geq 1$	The mean of a distribution
$\sigma \in \mathbb{R}^+ \wedge \sigma \geq 1$	The standard deviation of a distribution
$randomContribution \in \{0..1\}$	A proportion of peers which are selected randomly instead of by the algorithm
$tc \in \{0..1\}$	Parameter for contribution of the overall confidence value to the overall rating
Parameters observed during runtime	
$P.OC$	Overall confidence into a peer
$MO.RC$	Confidence into knowledge of a specific peer about a specific resource
$selectedPeers_Q$	A set of peers to forward query Q to

3. SWAP PLATFORM

The SWAP platform is built on JXTA. It features (among others) a set of modules for information extraction from the local peer, for information storage and for query routing.

3.1 Information and Meta-information

Information at each peer is stored in a *local node repository*. In our implementation these are RDF(S) statements in Sesame [4]. We here recollect that each RDF(S) statement consists of a subject, a predicate and an object. A statement may describe either data, e.g. (TBL isAuthorOf WeavingTheWeb)[2], or conceptual information, e.g. (University subClassOf Organization).

The SWAP storage model[3] (cf. [11]) does not just capture some statement like the two examples just given, but it also provides meta-information about these statements in order to memorize where the statement came from and how much resource-specific confidence and overall confidence is put into these statements and peers, respectively. The SWAP model for meta-information consists of two RDFS classes, namely Swabbi and Peer. For these classes, several properties are defined to provide the basis for the social metaphors outlined above and specified further below. Their corresponding data structures are summarized in Table 1.

Swabbi (MO): Swabbi objects are used to capture meta-information about statements and resources. They comprise the following properties:

- hasPeer ($MO.peer$) : This property is used to track which peer this Swabbi object is associated with.
- Resource-specific Confidence ($MO.RC$): This confidence value indicates how knowledgeable a peer is about a specific resource on a scale from 0 to 1. High confidence is expressed

[2]For sake of simplicity we here do not spell out namespaces, though they are very useful and indeed exploited in the SWAP platform.

[3]`http://swap.semanticweb.org/2003/01/swap-peer#`

by values near 1 and low or no confidence is expressed by values near or equaling 0.

Thus, the Swabbi object can capture for each rdf:Resource how much confidence one assigns to the remote peer concerning matters of the particular resource. The link between rdf:Resource and Swabbi is given by property hasSwabbi. The reader may note that when we speak about "confidence assigned to a remote peer concerning a particular statement", this is equivalent to the longer formulation "confidence assigned to a remote peer concerning the subject of a particular statement". Thus, we gather confidence meta-information around resources as anchors and avoid inefficient reification.

Peer (P): For each statement we have to memorize which peer it originated from. Information about a peer, e.g. its name, is specified by instances of the class Peer. The Swabbi links via hasPeer to Peer. In particular, each peer also memorizes and updates how much he confides overall into the other one:

- Overall Confidence ($P.OC$): Some peers may be more knowledgeable than others. This peer attribute is used to measure the overall confidence on a scale from 0 to 1, with 1 indicating that the remote peer is knowledgable and 0 indicating the opposite. Knowledgeable peers are the ones that provide a lot of information in general.

3.2 Querying for Data

We here consider two querying modes of the SWAP platform. First, we have a general query language, SeRQL [5], which was conceived by Broekstra et al. and which combines advantages of languages such as RQL [10], RDQL [17] and QEL [13]. Second, to reduce complexity of the simulation and get a better experimental grip at what peers do ask in the simulation environment, we have restricted the general SeRQL queries to the parts that it consists of, viz. queries for triples. Comparably to Tap [15], we query by

$$\text{getData(s,p,o)} \tag{1}$$

With s, p, o being either concrete URIs or (for o only) literals. In addition, s, p, o may be a wildcard '*' with the intuitive meaning that any URI or literal would match here. For instance, $getData(*, uri2, *)$ would match triples like $(uri{-}bill, uri2, uri{-}hillary)$ and $(uri - ronald, uri2, uri - nancy)$.

This is a reasonable simplification, as all SeRQL queries are eventually compiled to sets of such triple queries and since even such a simple querying mechanism allows comprehensive information requests.

4. ALGORITHM

4.1 The Social Metaphors

Peer-to-peer system are computer networks. The decentralized governing principles of peer-to-peer networks resemble social networks to a large extent. As mentioned before, a core task in such a network is finding the right peer among the multitude of possible addressees such that this peer returns a good answer to a given question. To do this effectively and efficiently, REMINDIN' builds on social metaphors of how such a human network works: We observe that a human who searches for answers to a question may exploit the following assumptions[4]:

1. A question is asked to the person who one assumes that he best answers the question.[5]
2. One perceives a person as knowledgeable in a certain domain if he/she knew answers to our previous questions.
3. A general assumption is that if a person is well informed about a specific domain, he/she will probably be well informed about a similar, e.g. the next more general, topic, too.
4. To quite some extent, people are more or less knowledgeable independently of the domain.
5. The profoundness of knowledge that one perceives in other persons is not measured on an absolute scale. Rather, it is often considered to be relative to one's own knowledge.

REMINDIN' builds on the metaphors of peer-to-peer networks being like a human social network and adopts the above mentioned assumptions in an algorithmic manner.

REMINDIN' consists of three major phases realizing these assumptions. *Peer selection* of REMINDIN' is based on assumptions (1) and (2). *Query relaxation* of REMINDIN' weakens the conditions that must be met such that we select a peer (assumption (3)). *Statement evaluation* modifies our estimation of the general profoundness of a peer's knowledge (4) as well as its topic specific profoundness (5). These phases are embedded in the following into the overall, high level network protocol.

4.2 Protocol Scenario

REMINDIN' consists of several steps executed *locally* and *across the network* when *forwarding* as well as *answering queries* and when *receiving responses*. Assuming the user of a peer issues a query to the peer network, the query is evaluated:

Locally against the local node repository. Its answers are presented.

Across the network: Forwarding. Simultaneously, *peer selection* (algorithm 1) is invoked to select a set of peers which appear more promising than the others to answer the given query. If it cannot select any peers for the given query, the query relaxation (algorithm 3) will be used to broaden the query until either all peers can be selected or eventually all known peers are returned. The original query is then send to a subset of the selected peers according to their *strength*. The message containing the query has a unique id and stores the id's of visited peers (message path) to avoid cycles.

Across the network: Answering Queries. When a peer receives a query, it will try to answer it and it will store an instance of Peer in its local repository referencing the querying peer. A meta object is created for each resource the query was about. The answer is returned directly to the querying peer. We just return an answer if it is not empty. However, for the peers selected by the querying peer the peer rating algorithm is invoked even if they have no answers. If the number of maximum hops is not reached yet, the query will be forwarded to a selected set of peers — using the same peer selection described before.

Receiving Responses. On the arrival of answers at the querying peer, relevant answers are selected with the *statement selection* algorithm and included into the repository. The answering peer and the included statements are rated according to the *statement evaluation* (algorithm 4).

We here briefly survey the just mentioned algorithms, before we go into more details in Sections 4.3 to 4.5.

[4]We do not claim that these observations of social networks are in any way exhaustive or without exceptions.

[5]'Best' in our current terms only means that he has the most knowledge. In future versions one may consider properties like latency, costs, etc.

Peer selection (algorithm 1): As motivated in Section 4.1, peer selection is based on observations of remote peers' knowledge. Statements from the local node repository that match the query constitute the basis yielding meta-information about where they came from. Thus, these statements help to identify the set of most knowledgeable peers.

Often, this procedure alone does not result in a sufficient number of peers to forward the query to. Then, the query relaxation algorithm is applied to the query.

Based on the resulting set of statements and peers, we combine the $P.OC$ value into each peer as well as the $MO.RC$ values, which may vary for each statement and peer, in order to derive a ranking according to algorithm 2. The result of algorithm 1 is an ordered set of peers to forward a query to (see Section 4.3 for details).

Query relaxation (algorithm 3): As just outlined, a query to the local node repository may not directly match any of its statements. Following observation (3), REMINDIN' relaxes the given query subsequently targeting peers with similar knowledge (see Section 4.4 for details).

Statement selection: Often the answer of a query contains more information than one wants to retain in the local node repository. Then, the user must either manually determine which information to store or the system must provide an automatic mechanism. Currently SWAP supports only manual statement selection. For evaluation of REMINDIN'in our simulation, we have not retained any statement of any answer at all in order to test REMINDIN'with the worst-case assumption.

Update overall ($P.OC$) and resource-specific ($MO.RC$) confidence values (algorithm 4): The $P.OC$ and $MO.RC$ values a peer assigns to remote peers and its associated statements are updated separately on the basis of the received answers. The number of statements returned is measured against the statements matching the original query already known by the querying peer. This measure is combined with the existing ratings in order to adjust the $P.OC$ and $MO.RC$ values according to algorithm 4 (see Section 4.5 for details).

4.3 Peer selection algorithm

Evaluating a query against the local node repository returns a set of statements matching the query. For each statement we retrieve its meta data, viz. a set of meta objects which comprise resource-specific confidence values for each peer's knowledge about the particular statement and overall confidence values for each peer.

The remote peers (local peer is omitted) are sorted according to their *strength*. Up to p_{max} best rated peers are returned as targets for the query. If REMINDIN' was not able to select suitable remote peers until then, REMINDIN' would relax the query and repeat the procedure. Algorithm 1 formalizes this procedure. It is called with the ontology O of the peer considering the query for forwarding, a query queue $\mathcal{Q}$ containing only the given query Q as an and an empty queue. Algorithm 1 uses Algorithm 2 as subroutine. Table 2 summarizes all auxiliary functions and procedures.

4.4 Query relaxation algorithm

Relaxation of a query can be achieved in a number of ways. We exploit the following considerations, which are also summarized in Table 3.

Table 2: Auxiliary Functions and Procedures

Algorithm 1	
performQuery(O, Q) $\rightarrow \mathcal{S}$	Evaluate getData(Q) on local node repository O returning statements $\mathcal{S}$
retrieveMetadata(O, S, P) $\rightarrow MO$	Retrieves from the local repository the meta data object for the subject of the specific statement S and peer P; if the subject does not have a Swabbi attached, retrieveMetadata queries the object for hasSwabbi
retrieveAllMetadata(O, S) $\rightarrow \mathcal{MO}$	Generalizes retrieveMetadata to all peers and returns sets of metadata, too
rankPeers(*selectedPeers*) $\rightarrow$ *rankedPeers*	Algorithm 2 rates each peer P that is found on the queue *selectedPeers* in a pairing with confidence values and returns the ranked queue *rankedPeers*
Algorithm 3	
determineState(Q) $\rightarrow$ *state*	Returns the current state of the query according to table 3
newQuery$(O, Q,$ *state*$)$ $\rightarrow \mathcal{Q}$	Returns queue of relaxed queries
Algorithm 4	
update(μ, σ, v) $\rightarrow [0, 1]$	$v \in \mathbb{R}^+, u \in [0..1]$ Update map a given value v onto a value between 0 and 1. The actual outcome depends on the function used in update
invUpdate(μ, σ, v) $\rightarrow iu$	$v \in [0, 1], iu \in \mathbb{R}^+$ invUpdate reverses the mapping done by Update. The mapping is a means to keep the overall/resource-specific confidence value between 0..1

1. A peer might be knowledgeable about a particular query, if one knows he had statements about the same subject-object combination, but with other predicates (states 1-3).

2. A peer might be knowledgeable about a subject-predicate combination, if one knows he had statements about the same subject alone, but maybe with other predicates (state 4).

3. A peer might be knowledgeable about an object, if one knows he had statements about the object, but where the object appeared in the subject position (state 5).

4. A peer might be knowledgeable about a property, if one knows he had statements about the superproperty (state 6).

5. A peer might be knowledgeable about a subject, if either

   (a) the subject is a class and one knows the peer was knowledgeable about the superclass (state 7a), or

   (b) the subject is an instance and one knows the peer was knowledgeable about a class the subject is an immediate instance of (state 7b).

6. A query for everything or the $\mathcal{ROOT}$ concepts or properties (i.e. in RDF these are rdf:resource, rdfs:class, rdfs:property, rdf:type, etc.) cannot be relaxed further.

Algorithm 3 implements these considerations. It exploits Table 3 in order to derive a(n often single-element) set of relaxed queries. One may note in particular: *(i)*, multiple relax queries exist when one asks for superproperties of a given property or immediate types

Algorithm 1 Peer Selection: peerSelection($O, \mathcal{Q}, A$)

Require: LocalNodeRepository O, Queue of Queries $\mathcal{Q}$, Queue of peers A

1: $\mathcal{Q}_{relaxed} := \emptyset$
2: **Queue** *selectedPeers*:= $\emptyset$
3: **for all** $Q \in \mathcal{Q}$ **do**
4: S_Q :=performQuery(O, Q)
5: **for all** $S \in \mathcal{S}_Q$ **do**
6: $\mathcal{MO}$:=retrieveAllMetadata(O, S)
7: **for all** $MO \in \mathcal{MO}$ **do**
8: *selectedPeers*.push($(MO.peer, MO.RC)$)
9: **end for**
10: **end for**
11: $\mathcal{Q}_{relaxed} := \mathcal{Q}_{relaxed}$+relaxQuery($O, Q$)
12: **end for**
13: $A :=$ A.append(rankPeers(*selectedPeers*))
14: **if** |A| $< p_{max}$ **then**
15: A :=peerSelection(O, $\mathcal{Q}_{relaxed}$, A)
16: **end if**
17: **return** A

Algorithm 2 Peer Rating: rankPeers(*selectedPeers*)

Require: Queue of pairs *selectedPeers*

1: Set $\mathcal{P} := \{P | \exists RC : (P, RC) \in$ *selectedPeers*$\} = \{P_1 \dots P_n\}$
2: **for all** $P \in \mathcal{P}$ **do**
3: $\mathcal{RC} := \{RC | (P, RC) \in$ *selectedPeers*$\}$
4: **for all** $RC \in \mathcal{RC}$ **do**
5: *strength*$(P) :=$ *tc*$\cdot P.OC + (1 -$ *tc*$) \cdot \frac{1}{|\mathcal{RC}|} \sum_{RC \in \mathcal{RC}} RC$
6: **end for**
7: **end for**
8: Queue *rankedPeers* $:= (P_1, \dots, P_{|\mathcal{P}|})$, where *strength*$(P_1) \geq \dots \geq P_{|\mathcal{P}|}$
9: **RETURN** *rankedPeers*

of a subject; *(ii)*, implicitly this relaxation is recursively applied in Algorithm 1; and, *(iii)*, there remain other options for query relaxation; first tests revealed that the ones above give quite good results for later-on selecting an appropriate peer to send the original query to. In general, however, further exploration and optimization of the current design choices are desirable.

Table 3: Query relaxation order

State	Query	Relaxed Query
1	(s, p, o)	$(s, *, o)$
2	$(s, p, *)$	$(s, *, *)$
3	$(*, p, o)$	$(*, *, o)$
4	$(s, *, o)$	$(s, *, *)$
5	$(*, *, o)$	$(o, *, *)$
6	$(*, p, *)$	$(*, super(p), *)$
7a	$(s, *, *)$	$(super(s), *, *)$
7b	$(s, *, *)$	$(class(s), *, *)$
8	$(*, *, *) \vee (\mathcal{ROOT}, *, *) \vee (*, \mathcal{ROOT}, *)$	

4.5 Update resource-specific and overall confidence values

The algorithm updateValues updates the overall confidence values one memorizes about a peer P ($P.OC$), and it updates the resource-specific confidence values $MO.RC$ one memorizes about a pair of a peer P and a resource. The algorithm consists of three major parts. First, it quantifies what the local peer knows in O about the original query Q in the measure $localAnswer$ and it quantifies what the remote peer P knows about the same query Q in the measure $remoteAnswer$ (lines 1 to 3). Second it compares the ratio $\frac{remoteAnswer}{localAnswer}$ with different numbers that depend on the mean μ and standard deviation σ of *all* statements with regard to the 'average' query (lines 6,10,14,18). Thereby, μ and σ have been found by counting results of observed queries and assuming a Gaussian distribution. Correspondingly, in the third step, one increases or decreases or does not touch the $P.OC$ values (lines 9,13,17,21,24,25) and the $MO.RC$ values attached to subjects (or objects, if subjects are not found in O) of statements (lines 8,12,16,20,26-33). The size of the modifications depend in particular on the size of the result set as compared to the local result.

Algorithm 3 Query Relaxation: relaxQuery(O, Q)

Require: O, Q
$state :=$ determineState(Q)
$\mathcal{Q} :=$ newQuery($O, Q, state$)
return $\mathcal{Q}$

An interesting special case happens, when a remote peer has been asked directly by the local peer, but when it has not returned an answer.[6] REMINDIN' then assumes after a certain time that the queried peer has no answer at all and correspondingly, the $P.OC$ and $MO.RC$ values are downgraded. Thus, even if the remote peer is very knowledgeable, but unwilling to answer or overfreight with queries, the remote peer will be considered as a less worthy candidate for querying — hence, a simple form of load balancing will be achieved, too.

5. EVALUATION SETTING

Though our plans for the SWAP bibliography case study will involve the participation of several dozens and up to 100 researchers (cf. Section 2), even this number will be too small to actually evaluate REMINDIN'. In addition, it will be difficult to investigate crucial parameters of REMINDIN' without jeopardizing the running of the overall network. Hence, we opted for evaluating REMINDIN' by simulating a peer-to-peer network with plausible datasets of statements and query routing by REMINDIN'.

To this end, we here discuss the data source on which we have based the local node repositories of the individual peers, viz. the DMOZ open directory and its RDF dump[7] (Section 5.1). We briefly discuss the assignment of statements to peers — mostly modeling human editors of DMOZ as peers (Section 5.2). We describe the queries that are generated and sent around (Section 5.3), the initial configuration of the peer-to-peer network (Section 5.4), and the evaluation measures (Section 5.5) we consider subsequently to assess REMINDIN'.

5.1 Data source characteristics

DMOZ, the open directory project, manually categorizes Web pages of general interest into a topic hierarchy. For each topic one or several editors are responsible to define subtopics or related topics and to contribute links to outside Web pages to the topic pages of DMOZ. For this semi-structured data source, there exists an RDF dump comprising a small schema and many instances. The main classes of the DMOZ data set are Topic, Alias, and ExternalPage:

[6] This is not observed if the remote peer has been asked indirectly.
[7] http://rdf.dmoz.org/

Algorithm 4 Update overall/resource-specific confidence: updateValues$(O, P, Q, \mathcal{S}, \mu, \sigma)$

Require: $O, P, Q, \mathcal{S}, \mu, \sigma$
1: $localAnswer := |\text{performQuery}(O, Q)| + 1$
2: $remoteAnswer := |\mathcal{S}| + 1$
3: $cover := \frac{remoteAnswer}{localAnswer}$
4: $COVER := \frac{\sigma}{\mu} + 1$
5: // Compute changes to resource/overall confidence
6: **if** $cover > COVER^2$ **then**
7: // Large increase of resource/overall confidence
8: $updateValue_{RC} := COVER * cover$
9: $updateValue_{OC} := COVER$
10: **else if** $cover > COVER$ **then**
11: // Moderate increase of resource/overall confidence
12: $updateValue_{RC} := cover$
13: $updateValue_{OC} := 1$
14: **else if** $cover > \frac{1}{COVER}$ **then**
15: // Do nothing, no definitive conclusion for this range!
16: $updateValue_{RC} := 0$
17: $updateValue_{OC} := 0$
18: **else**
19: // Decrease of resource/overall confidence
20: $updateValue_{RC} := -cover^{-1}$
21: $updateValue_{OC} := -1$
22: **end if**
23: // Update resource and overall confidence values for peer P
24: $P.OC :=$
25: update(invUpdate$(P.OC) + updateValue_{OC}$)
26: **for all** $S \in \mathcal{S}$ **do**
27: **if** $S \in O$ **then**
28: $MO :=$retrieveMetadata(O, S, P)
29: $MO.RC :=$
30: update(invUpdate$(\mu, \sigma, MO.RC)+$
31: $updateValue_{RC}$)
32: **end if**
33: **end for**

Topic The resource Topic has properties for link[8], containing a reference to an ExternalPage and to an editor, viz. an editor of the topic. The properties related , symbolic and narrow describe relations to other topics and aliases.

Alias The resource Alias has properties for Title and Target. Target is a relation to another Topic.

ExternalPage has the properties Title and Description

The data source has some interesting properties rendering it adequate for our evaluation purposes: (1) There are many relations other than taxonomic ones between the topics — in contrast to many other datasets. (2) Each topic has at least one editor, many have several — implying a natural way to assign statements to peers. (3) Topics are 'populated' with many links.

The DMOZ hierarchy is available in pure RDF only. To enhance its semantic description, we have converted it to RDFS. We converted the topics to instances of `rdfs:class`. narrow and narrow1 were converted into `rdfs:subClassOf`. The properties link were interpreted as `rdf:type` of the topics they belong to. For instance, `http://www.w3.org/People/Berners-Lee/` is then an instance of `http://dmoz.org/Computers/Internet/History/People/Berners-Lee,_Tim/`.

[8]Actually, DMOZ distinguishes important and less important links by categorization into link, link1, link2. We have merged these three again into one property. The analogous case is true for symbolic, etc.

Table 4: Survey of DMOZ Open Directory structure realized as RDF schema and instances

DMOZ	RDF(S) entity	Example	Number
Schema			
Topic	Class	Top/Arts	1657
Property	Property	symbolic, related	16
Property domain			
Topic	a class	Top/Arts	1657
Property range			
Topic, Alias	a class	Top/Arts	1657
Instances			
link	rdf:type	"`http://www.w3.org/People/Berners-Lee/`" rdf:type "Top/Computers/Internet/"	45347
symbolic, related	symbolic, related	"Top/Arts/" related "Top/Business/Arts_and_Entertainment"	3520
narrow, narrow1	subClassOf	"Top/Arts/Movies" subClassOf "Top/Arts"	1952
editor	Peer		1100

Table 5: Major statistical parameters

Property	Mean	Standard deviation
No. of topics / editor	4	5,4
No. of links / topic	27	35,2
No. of links / editor	53	69,1
No. of queries	1657	
Expected no. of answers / query / peer	13	23,1

In order to handle the sheer size of the DMOZ hierarchy, we included only the first three levels of the hierarchy in our experiments. The properties of the remaining data source are summarized in table 4.

5.2 Distribution of Statements

All of the 1657 topics in the first three levels of the DMOZ hierarchy have one or more editors assigned to them. Everybody can become an editor of a category in DMOZ. DMOZ encourages users to "choose a topic you know something about and join". Hence, we assume that editors who edit a topic (which became classes in RDF(S)) also store links they have assigned to a topic locally. Since, a topic can have more than one editor, not all of the links need to come from one editor alone. Finally, editors may also add links to other topics. Thus, they are probably also informed about the related topic but to a lesser extent.

These assumptions have led us to the following distribution of instances in our simulation. We represent one editor by one peer, thus we have 1100 peers. Assuming that an editor is not the source of all instances within 'his' topic ('his' class) we choose randomly 70% of the direct instances within 'his' class and assigned them to the peer's local node repository. In addition, we considered all classes directly related to 'his' class (via subClassOf, via related, etc.) and we randomly assigned 12% of the direct instances of these directly related classes to the peer's local node repository. Thus, all

peers were assigned their local node repository.

To give a further impression of the resulting information distribution, we list the number of topics which have i editors. 755 topics have 1 editor; 333 topics have 2 editors; 204 topics have 3 editors; ...; 44 topics have 6 editors; ...;14 topics have 10 editors ;1 topic has 32 editors.

5.3 Generation of queries in experiment

Queries are generated in the experiments by instantiating the blueprint $(*, < rdf : type >, topic)$, with $topics$ arbitrarily chosen from the set of topics that had at least one instance. Thus, generated queries retrieved all instances of a topic — considering also the transitivity of the subclassOf-relationship to subtopics. I.e. we generated 1657 different queries.

We had two different scenarios for evaluating the effectiveness of REMINDIN'. In the first set of scenarios, we continuously choose from the 1657 queries and evaluated right away the results. In the second set of scenarios, we partitioned the set of 1657 queries into two sets of equal size. There were two phases. First, there was a 'learning phase' where the peer network was confronted with the first set 828 queries. Then, there was an explicit 'test phase', in which one could observe how the peer network would re-adjust to the second set of queries.

5.4 Initial configuration of peer-to-peer network simulation

The simulation is initialized with a network topology which resembles the JXTA network. 10 peers connect to 1 rendezvous peer randomly and the rendezvous peers connect to the central JXTA peer[9]. Hence, in the beginning the only remote peer visible to the peers is its rendezvous peer. During the simulation any peer can become visible to any other peer in the network if it knows its unique identifier. The identifiers are propagated with the queries[10]. This assumption is valid, since we focus on the *semantic routing of queries* rather than the technical routing. In the simulation, peers were chosen randomly and they were given a randomly selected query to question the remote peers in the network. The peers decided on the basis of their local node repository which remote peers to send the query to. Each peer used REMINDIN' to select up to $p_{max} = 2$ peers to send the query to. As a baseline we compared REMINDIN' to a naive algorithm. In this case a peer selected randomly up to p_{max} peers to send the query to. A peer that received a query tried to answer the query. Each query stores the path that it is forwarded along and if a peer had appeared in this path, it was deselected. In some evaluation scenarios, we have integrated a *randomContribution*. The $randomContribution$ percentage of selected peers were randomly exchanged again randomly selected ones known by the querying peer. Each query was forwarded until the maximal number of hops (h_{max}) is reached. In our experiments, we have not considered the leaving or joining of nodes, so far.

5.5 Evaluation measures

There are many criteria to evaluate algorithms for peer-to-peer systems. In [9] we summarize many of them. For our evaluation we rely on two major measures.

Recall R. Recall is a standard measure in information retrieval. In our setting, it describes the proportion between all relevant statements in the peer network and the retrieved ones.

[9]The IP numbers of central JXTA and rendezvous peers can be downloaded from Suns servers in the initialization phase of a peer

[10]In our setting "being visible", "being known" and "being a possible direct addressee of a query" a synonymous to each other

Table 6: Standard parameter in evaluation

Parameter	Value
μ	20
σ	10
p_{max}	2
h_{max}	7
tc	0.1
$randomContribution$	0.0
$Peers$	1100

$$R = \frac{|retrieved|}{|relevant|}$$

We use recall to assess the *effectiveness* of REMINDIN', i.e. to measure to which extent one may retrieve statements from the peer-to-peer network based only on local knowledge about possibly relevant peers.

Network load. This figure can be measured with different subparameters. *Messages per query* traces to what extent the network is being flooded by one query. The number of *average hops* can indicate how goal-oriented a query is routed and how fast a answer may be returned. We use the network load, and messages per query in particular, to assess the *efficiency* of our approach.

6. RESULTS

Our simulations show that REMINDIN' reaches a significant higher recall than the naive baseline. In particular, peer selection is improved by query relaxation and some random selection of peers. Before we present the final evaluation results, we here summarize the major hypotheses we wanted to investigate.

6.1 Hypotheses

1. The proposed algorithm provides better results in terms of recall than the naive algorithm.
2. The network load needed to reach a specific recall decreases over time, such as measured in terms of messages per query and number of hops.
3. The peers adapt quickly to new requirements expressed by new queries.
4. Using our query relaxation mechanism is better than considering just the original query to select peers.
5. The parameters of the algorithm have an effect on the effectiveness of the peer selection.
6. Some randomness contributing to peer selection helps to escape over fitting.

6.2 Evaluation

In Table 2 we define the different parameters of the algorithm. In case we did not state otherwise, they were set to the values given in Table 6. In the naive approach the peer has used the same parameters as REMINDIN' — except that all the peers were chosen randomly. Points in all the graphs represent averages for 1000 queries.

Hypothesis 1: Figure 1 summarizes the comparison between REMINDIN' and the naive approach. In this scenario we used 20.000 queries and 50% of all queries. The naive approach produced an average of 600 messages per query and had a constant recall of approximately 20%. The recall of REMINDIN' without random contribution increases steadily over time and reaches a recall of 50% after all queries. Note that 20.000 queries in total result in just

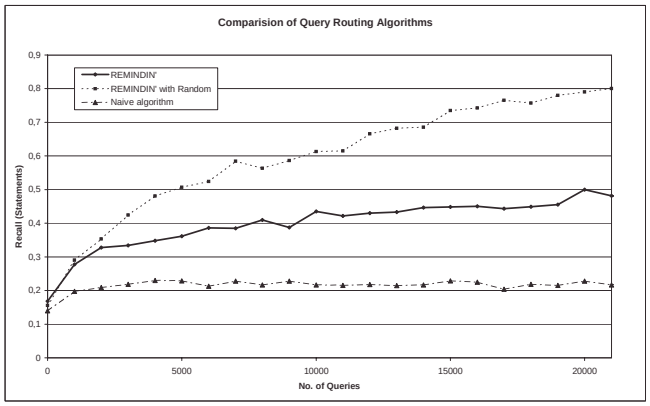

Figure 1: The proposed algorithm provides better results in terms of recall than the baseline.

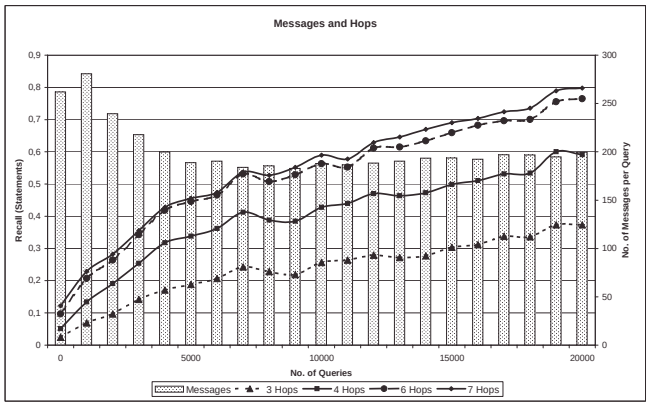

Figure 2: The number of messages decreases over time (with standard parameter setting). The recall increases with time for all numbers h_{max} from 3 to 7. The contribution of additional recall is highest for h_{max} between 3 and 4.

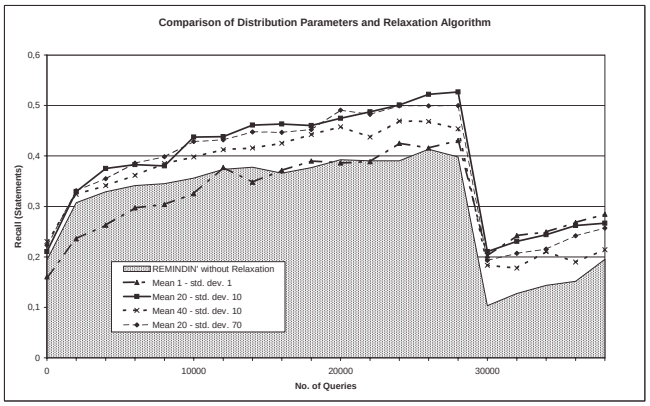

Figure 3: Using relaxation to broaden the set of selected peers is especially useful when previously unknown queries are introduced (here: after 30,000 queries). The parameters of the distribution affect the effectiveness of the peer selection.

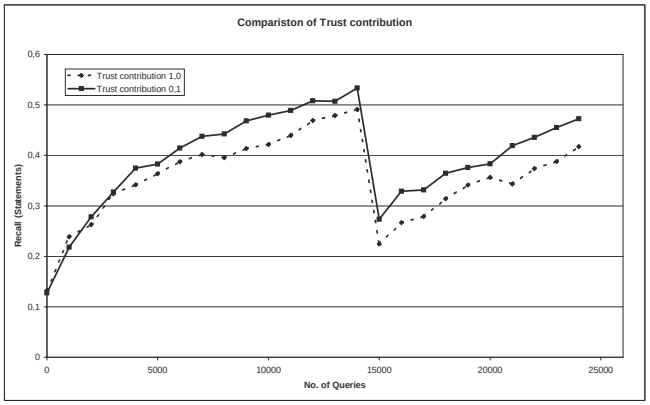

Figure 4: Peer selection based on overall confidence only is less effective mixed accounts of overall and resource confidence.

about 18 queries per peer, a fairly low number. REMINDIN' with a little random contribution to the peer selection produces even better results. After 20.000 queries it reaches a recall of almost 80%.

Hypothesis 2: Figure 2 illustrates a simulation run for REMINDIN' with a random contribution of 20%. Note that we plot the number of queries on the left hand side against the average recall and on the right hand side against the number of messages per query. Figure 2 depicts the number of messages needed to reach the recall, in average 200. Hence REMINDIN' increases the recall by 300% while requiring only one third of the messages per query as compared to the baseline of approx. 600.

We observe that the recall contribution decreases with the number of hops. The recall increases by a large margin from three hops to four hops accounting for 25% of the total recall, while the recall increases only 7% from six to seven hops. This suggests that one could decrease the number of h_{max} from seven to five without significantly changing the overall recall, but producing less network load.

Intriguingly, the overall recall increases over time while the number of messages per query remains about the same. The number of messages comes down when the peers have built up their model, since they agree then who is knowledgable about which topic.

Hypothesis 3: We cannot confirm hypothesis three, at least not for the basic algorithm without random contribution. As figure 3 demonstrates for different parameters μ and σ, the recall degrades to 20% when we introduce new queries after 30.000 queries (cf. the second scenario for generating queries in Section 5.3). However, we note that the introduction of completely new queries rarely happens in real world applications.

Then, Figure 4 is more promising. Here, peer selection was randomized to 20%. The result was that the recall retains 27% and the adaptation occurs rather quickly. It seems that the random contribution helps to avoid overfitting!

Hypothesis 4: Figure 3 nicely exemplifies the effect of the query relaxation algorithm. In the beginning the peer selection without relaxation works almost as good as with relaxation. When new queries arise, REMINDIN' with relaxation performs significantly better than without.

Hypothesis 5: Figure 3 contrasts the effects of different parameters and query relaxation algorithm on the peer selection. The hypothesis is confirmed. However, the consequence of the parameters are less severe than the application of the query relaxation algorithm. In particular the simulation runs suggest, that within certain ranges they do no harm. As expected a setting with *COVER* not matching the actual distribution (13;533 $\Rightarrow$ *COVER* $= 2.7$) as in case (40; 100 $\Rightarrow$ *COVER* $= 1.25$) rates peers with minor knowledge to good and thus hinders the identification of the real champions.

Figure 4 concentrates on parameter *tc*, the contribution of the overall confidence value to the overall peer rating during peer selection. This figure is of particular interest, because one could argue to store only the information if a peer is knowledgeable or not. As we can see from the diagram the inclusion of the confidence rating increases the achievable recall and adapts better to new queries. This test was performed with a 20% random contribution. Nevertheless, it is still interesting to note that overall confidence alone does not fare too bad, either.

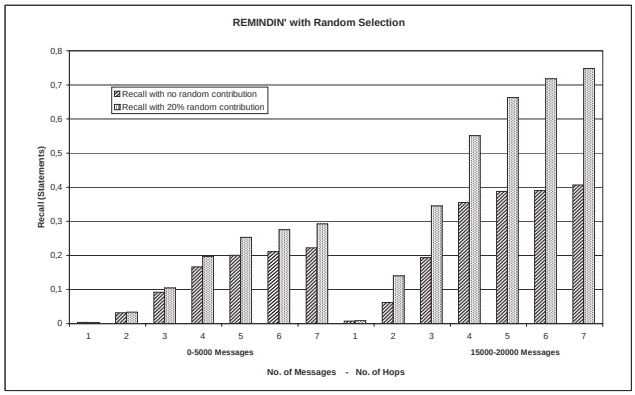

Figure 5: Selecting some of the peers (20%) randomly enhances recall.

Hypothesis 6: Most of our hypotheses were supported with most strength when we combined our algorithm with a proportion of randomly selected peers. To motivate this result we want to recall an observation from human interaction. It happens sometimes that we meet a previously unknown person and she provides us with a yet new view on the world or on a certain topics. Figure 5 analyzes the observation in more detail. We put side by side the average recall with and without random contribution (20%), which we have averaged over the first and last 5.000 messages. It is obvious that the achievable recall of REMINDIN' without random contribution reaches a certain level and does not increase further. Note that in average just 87 messages per peer were needed to get to this recall. The difference is yet more obvious in the case of the last 5.000 messages then the first 5.000. However, with the introduction of randomness the recall can be enhanced substantially!

7. RELATED WORK

We consider three areas of research related to our work. The first is the general research in peer-to-peer systems. The second area deals with semantic peer-to-peer systems and the third area of related work is chosen with respect to our query relaxation algorithm.

General research in peer-to-peer system concentrates either on efficient topologies for these networks or on distribution of documents. The *small-world-effect* (cf. [2]) is one example how those topologies can be exploited to establish a connection between two nodes efficiently. In contrast to our work the content of a node is advertised to all neighbors, and thus needs updates when a nodes content changes. The algorithm ensures that a given query is forwarded to a node with the most neighbors. There are a number of other P2P research system which are related to the question of how to allocate documents within a peer-to-peer network. They mostly require equally shaped keys for nodes and their contents [24] [26], thus once a key for the searched content has been created, the address and thus the root to the target peer can be easily found. One problem with this system is, that it generates a substantial overhead when nodes join and leave the network. In [1] an algorithm is proposed which replicates documents on different peers in a way that joining and leaving produces less overhead, but efficient structured search for documents is still possible. In contrast to our work they examine how a known resource can be found with least possible messages. They do not provide a solution how a relevant resource can be found in the first place. This is the question examined by the second group of related works.

EDUTELLA [13] is a peer-to-peer system based on the JXTA platform, which offers very similar base functionality as the SWAP system. In [20] they propose a query routing mechanism based on super peers. Peers which have topics in common are arranged in a hypercube topology. This topology guarantees that each node is queried exactly once for each query. Our algorithm is not based on an explicit topology, thus it does not generate any overhead to establish it. Our simulations illustrate that we need much less queries than the number of peers to reach most knowledgable peers. Furthermore most information is in the reach of four to five hops which is advantageous in terms of expected response time. They do not provide any test on the performance of their algorithm. Within the proposed hypercube topology all peers are equal, while we can distinguish between more or less knowledgeable peers.

In [7] a Semantic Overlay Network (SON) is introduced. Resources are clustered into a topic hierarchy and peers subsequently join a SON. The SONs a peer joins depend on the clustering result of the local knowledge. In contrast to our work a peer actively joins peers which have assigned themselves to a certain SON. We establish connections to remote peers based on the queries. While in our case resources are organized in a graph they use a hierarchy. Hence, they cannot exploit other relations than hierarchical ones to find other promising peers. They do not weight the connections.

In [27] an algorithm is presented which concentrates on load balancing between the peers. While they are interested in querying peers equally often we concentrate on the selection of the most knowledgeable peers. However, our algorithm adapts in a self organizing way to peer overloading, while they use leading nodes to calculate different measures and reassign categories depending on results of their algorithm.

Regarding our query relaxation algorithm, a lot of research has been done in the field of query relaxation in the context of cooperative databases (*cf.* [6, 19]). In those contexts queries are relaxed according to a similarity function in case a given query does not result in answers when posted to a database. In [23] the similarity of two concepts is determined by the length of the shortest path between them, which is interesting for non hierarchical taxonomies with multiple concept inheritance. We exploit the relations of the knowledge structure to find relevant peers rather than map a query between different schema. To our knowledge this approach has not yet been applied to peer-to-peer networks.

8. CONCLUSION

The principle of self-organization has been discussed for a long time as a paradigm for introducing order into complex systems without centralized control. In recent years one could see that this principle has found its entry into different types of engineering applications (cf., e.g., [25]) — in particular ones that involve the Web, such as identification of communities for better harvesting and classification of information [14] or ones that use self-organization in peer-to-peer networks [1]. In theory, the possibilities of self-organizing appear to be open-ended with ideas ranging up to social systems of human and machine agents that form networks with enormously effective communication structures — as one knows, e.g., from Milgram's experiment on six degree's of sep-

aration in 1967 [18]. Though the idea of transferring such communication principles from the original social networks to comparable technical networks like Peer-to-Peer networks has been ventilated for some time (cf. [21]), corresponding research has not taken a serious stance at it.[11] To this end, we have devised REMINDIN', a highly original algorithm to find peers in a semantic peer-to-peer network based on social metaphors. The algorithm comprises a peer selection algorithm based on confidence ratings, query relaxation and observation of useful responses given by other peers. The algorithm provides significantly better results than its naive counterpart. Our experiments with REMINDIN' have shown intriguing results: *(1)*, some randomness in peer selection helps escape overfitting and improves effectiveness of REMINDIN', *(2)* self-organized learning by the network reduces the network load over time, and, *(3)*, parameter settings play a role, but the behaviour of REMINDIN', is rather elastic to their setting.

Acknowledgements.

Research reported in this paper has been partially financed by EU in the IST project SWAP (IST-2001-34103); see `swap.semanticweb.org`. We thank all our colleagues in the SWAP project and at AIFB for fruitful discussions.

9. REFERENCES

[1] K. Aberer, P. Cudr-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, and R. Schmidt. P-Grid: a self-organizing structured p2p system. *ACM SIGMOD Record*, 32(3):29–33, 2003.

[2] Lada A. Adamic, Rajan M. Lukose, Amit R. Puniyani, and Bernardo A. Huberman. Search in power-law networks. *Physical Review E,* 64(46135), 2001.

[3] David W. Aha, editor. *Lazy Learning*. Kluwer, Dordrecht, 1997.

[4] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A generic architecture for storing and querying RDF and RDF schema. In *The Semantic Web - ISWC 2002*, volume 2342 of *LNCS*, pages 54–64. Springer, 2002.

[5] J. Broekstra. SeRQL: Sesame RDF query language. In M. Ehrig et. al., editor, *SWAP Deliverable 3.2 Method Design*, pages 55–68. 2003. `http://swap.semanticweb.org/public/Publications/swap-d3.2.pdf`.

[6] Wesley W. Chu, Hua Yang, Kuorong Chiang, Michael Minock, Gladys Chow, and Chris Larson. Cobase: a scalable and extensible cooperative information system. *J. Intell. Inf. Syst.*, 6(2-3):223–259, 1996.

[7] Arturo Crespo and Hector Garcia-Molina. Semantic Overlay Networks for P2P Systems. submitted for publication `http://www-db.stanford.edu/~crespo/publications/op2p.pdf`, 2002.

[8] A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, pages 23–32. IEEE Press, 2002.

[9] M. Ehrig et al. Towards evaluation of peer-to-peer-based distributed knowledge management systems. In L. van Elst et al., editors, *"Agent-Mediated Knowledge Management" International Symposium AMKM 2003 Stanford, CA, USA*, LNAI, pages 73–88. Springer, Berlin, 2003.

[11] We have discussed some very few noteworthy exceptions in Section 7.

[10] G. Karvounarakis et al. RQL: A declarative query language for rdf. In *Proc. of the 2002 WWW conference*, pages 592–603, Hawaii, USA, 2002.

[11] J. Broekstra et. al. A metadata model for semantics-based peer-to-peer systems. In K. Aberer et al., editor, *Semantics in Peer-toPeer and Grid Computing - SemPGRID, collocated with the 2003 WWW conference*, Budapest, May 20 2003.

[12] M. Schlosser et. al. A scalable and ontology-based P2P infrastructure for Semantic Web Services. In *P2P-2002 — Proceedings of the 2nd Int. Conf. on Peer-to-Peer Computing*, pages 104–111. IEEE Press, 2002.

[13] W. Nejdl et. al. EDUTELLA: A P2P networking infrastructure based on RDF. In *Proc. of the 2002 WWW conference*, pages 604–615, Hawaii, USA, May 2002.

[14] G.W. Flake, S. Lawrence, C.L. Giles, and F.M. Coetzee. Self-organization and identification of web communities. *IEEE Computer*, 35(3):66 –70, March 2002.

[15] R. Guha and Rob McCool. Tap: a semantic web platform. *Computer Networks*, 42(5):557–577, August 2003.

[16] A. Löser et al. Efficient data store discovery in a scientific P2P network. In N. Ashish and C. Goble, editors, *Proc. of the WS on Semantic Web Technologies for Searching and Retrieving Scientific Data*, CEUR WS 83, 2003. Colocated with the 2. ISWC-03 `http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-83/`.

[17] B. McBride. Jena: A semantic web toolkit. *IEEE Internet Computing*, 6(6):55–59, 2002.

[18] Stanlay Milgram. The small world problem. *Psychology Today*, 67(1), 1967.

[19] A. Motro. FLEX: A tolerant and cooperative user interface to databases. *IEEE Trans. on Knowledge and Data Engineering (TKDE)*, 2(2):231–246, Jun. 1990.

[20] W. Nejdl et al. Super-peer-based routing and clustering strategies for rdf-based peer-to-peer networks. In *Proc. of the 2003 WWW conference*, Budapest, Hungary, May 2003.

[21] Andy Oram, editor. *Peer-to-Peer. Harnessing the Power of Disruptive Technologies*. O'Reilly, 2001.

[22] Dennis Quan, David Huynh, and David R. Karger. Haystack: A platform for authoring end user semantic web applications. In *The SemanticWeb — ISWC 2003*, LNCS 2870, pages 738–753, Heidelberg, 2003. Springer-Verlag.

[23] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proc. of the 14th IJCAI*, pages 448–453, 1995.

[24] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. of the Int. Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, 2001.

[25] S. Staab, F. Heylighen, C. Gershenson, G. W. Flake, D. M. Pennock, D. C. Fain, D. De Roure, K. Aberer, W.-M. Shen, O. Dousse, and P. Thiran. Neurons, viscose fluids, freshwater polyp hydra — and self-organizing information systems. *IEEE Intelligent Systems*, 18(4):72–86, July-Aug. 2003.

[26] I. Stoica et al. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proc. of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.

[27] P. Triantafillou et al. Towards high performance peer-to-peer content and resource sharing systems. In *Proceedings of the 2003 CIDR Conference*, 2003.