Querying Structured Data in an Unstructured P2P System

Verena Kantere
School of Electr. and Comp. Engineering
National Technical University of Athens
vkante@dbnet.ece.ntua.gr

Dimitrios Tsoumakos
Department of Computer Science
University of Maryland, College Park
dtsouma@cs.umd.edu

Nick Roussopoulos
Department of Computer Science
University of Maryland, College Park
nick@cs.umd.edu

ABSTRACT

Peer-to-Peer networking has become a major research topic over the last few years. Sharing of structured data in such decentralized environments is a challenging problem, especially in the absence of a global schema. The standard practice of answering a query that is consecutively rewritten along the propagation path often results in significant loss of information. In this paper, we present an adaptive and bandwidth-efficient solution to the problem in the context of an unstructured, purely decentralized system. Our method allows peers to individually choose which rewritten version of a query to answer and discover information-rich sources left hidden otherwise. Utilizing normal query traffic only, we describe how efficient query routing and clustering of peers can be used to produce high quality answers. Simulation results show that our technique is both effective and bandwidth-efficient in a variety of workloads and network sizes.

Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval; C.2.4 [Computer-Communication Networks]: Distributed Systems—Distributed Databases

General Terms

Algorithms

Keywords

Peer-to-Peer, Query Reformulation, Structured Data

1. INTRODUCTION

In the last years there has been a growing interest in the Peer-to-Peer (hence P2P) paradigm, at first as a new trend for network applications and later for general decentralized applications of data sharing. The P2P paradigm dictates a fully-distributed, cooperative network design, where nodes collectively form a system without any supervision.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
WIDM'04, November 12–13, 2004, Washington, DC, USA.
Copyright 2004 ACM 1-58113-978-0/04/0011 ...$5.00.

Today, the most popular P2P applications operate on *unstructured* networks, with peers joining and leaving the system in an ad-hoc fashion, while maintaining only local knowledge. While structured overlays (e.g., [1,2]) provide with efficient lookup operations, in many realistic scenarios the topology cannot be controlled and thus they cannot be used (e.g., dynamic ad-hoc networks or existing large-scale unstructured overlays).

In contrast with data integration architectures, P2P data sharing systems do not assume a mediated schema to which all sources of the system should conform. In such a system, where peers share (semi)structured data, each is an autonomous source that has a local schema. Sources store and manage their data locally, revealing only part of their schemas to the rest of the peers. In a pure P2P system, peers perform local data management and coordination with their acquaintees, i.e. their one-hop neighbors in the overlay. During the acquaintance procedure between two peers, they exchange information about part of their local schema and create a mediating mapping semi-automatically [3]. The establishment of an acquaintance implies an agreement for the performance of data coordination between the acquaintees based on the respective schema mapping. However, peers do not have to conform to any kind of data or schema transformation in order to establish acquaintances with other peers, and, hence, participate in the system.

In a large-scale structureless P2P data management system as described above, joining peers become acquainted to the first randomly available nodes and not to the most useful ones, i.e. the peers that best meet their need for information. Therefore, they have to direct queries not only to their neighbors, but also to a greater part of the system. As a consequence of the lack of global schema, peers express and answer queries on their local schema. Furthermore, the lack of global knowledge deprives peers from the ability to direct their queries to appropriate remote nodes.

The straightforward procedure for query processing in an unstructured P2P data management system consists of the propagation of the query on paths of bounded depth in the overlay. At each routing step, the query is rewritten to the schema of its new host based on the respective acquaintance mappings. A query may have to be rewritten several times from peer to peer till it reaches nodes that are able to answer it sufficiently in terms of quality but also quantity. It is obvious that the successive rewritings decrease monotonically the information held by a query and, thus, also reduce monotonically the possibility of accurate query answering. Moreover, it is the case that peers may not be able to sufficiently answer received queries not because their local schema does not match the initial query adequately, but because the incoming rewritten version has been gradually reduced. Therefore, the performance of the query processing procedure is degraded during the rewritings on intermediate peers.

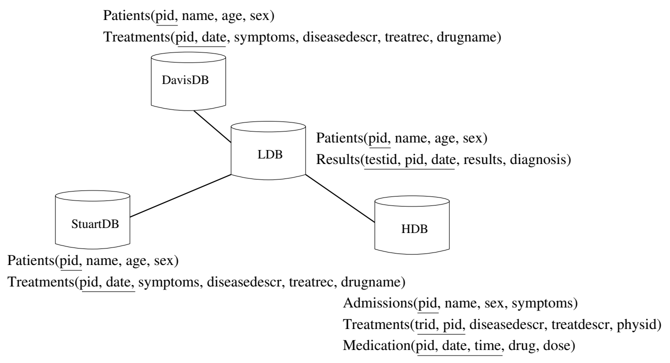
Patients(pid, name, age, sex)
Treatments(pid, date, symptoms, diseasedescr, treatrec, drugname)
DavisDB
LDB
Patients(pid, name, age, sex)
Results(testid, pid, date, results, diagnosis)
StuartDB
HDB
Patients(pid, name, age, sex)
Treatments(pid, date, symptoms, diseasedescr, treatrec, drugname)
Admissions(pid, name, sex, symptoms)
Treatments(trid, pid, diseasedescr, treatdescr, physid)
Medication(pid, date, time, drug, dose)

Figure 1: Part of a P2P system with peer-databases from the health environment

In this paper, we propose a methodology that aims at increasing the accuracy of query answering in the absence of a global schema in unstructured P2P data management systems. Our method proposes a first solution to the problem of query degradation by evading query rewriting at peers poor in relevant information. We propose an adaptive, bandwidth-efficient scheme that performs a gradual clustering between peers with common interests in terms of information. This is solely performed by exploiting normally posed queries and their replies in a decentralized manner. We then proceed to describe the query routing and clustering mechanisms as well as the local metadata maintenance used to achieve these goals. Finally, we present experimental results for a variety of environments and workloads which show that our method manages to improve the quality of query results using very few messages without any kind of global knowledge.

1.1 Motivating Example

Envision a P2P system where the participating peers are databases of private doctors of various specialties, diagnostic laboratories and databases of hospitals. Figure 1 depicts a small part of this system, where nodes are: DavisDB - the database of the private doctor Dr. Davis, LDB - the database of a diagnostic laboratory, StuartDB - the database of the private doctor Dr. Stuart and HDB - the database of a department of a hospital. On top of each database sits a P2P layer, which is responsible for all data exchange between this peer with its acquaintees. The P2P layer is also responsible for the creation and maintenance of mappings of local schemas during the establishment of acquaintances towards the line of [3]. The schemas of the databases are shown in Figure 1. Moreover, each peer owns a query rewriting and a query-schema matching mechanism. Thus, when a query expressed on the local schema of a peer is propagated to one of its acquaintees, it can be rewritten to the latter's local schema as follows: the peer that has to perform the rewriting uses the mapping that exists between its own schema and the schema of the respective acquaintee and employs the query-schema matching mechanism to deduce which matches are necessary for the rewriting of the query. Then, it employs the rewriting mechanism and the selected matches in order to express the query on the local schema. Note that the rewritten query will carry only the part of information of the original version that is present in the schema mapping between the two acquaintees. The methodologies of peer-schema matching and query rewriting are out of the scope of this work. Other works like [3], [4], [5] are dealing with these issues.

Suppose that Dr. Davis would like to search the P2P system for information about cases of patients that suffered from X. He would like to know what are the characteristics of these patients, i.e., sex and age, the symptoms that led to the diagnosis of X and what was their treatment. Thus, he poses the following query in the P2P system:
Qorig:

```
SELECT  age,sex,symptoms,treatrec,drugname
FROM    Patients,Treatments
WHERE   Patients.pid = Treatments.pid
        and Treatments.diseasedescr = "X"
```

DavisDB propagates Q_{orig} to its single acquaintee, LDB. In order for the latter to answer the above query, it rewrites it to each own schema:
Qorig_LDB:

```
SELECT  age,sex
FROM    Patients,Results
WHERE   Patients.pid = Results.pid and
        Results.diagnosis = "X"
```

In the above rewriting we have assumed that Treatments.diseasedescr is mapped to Results.diagnosis. LDB sends the above query to its acquaintees: StuartDB and HDB, that rewrite it to their own schemas as follows:
QLDB_StuartDB:

```
SELECT  age,sex
FROM    Patients,Treatments
WHERE   Patients.pid = Treatments.pid and
        Treatments.diseasedescr = "X"
```

QLDB_HDB:

```
SELECT  sex
FROM    Admissions,Treatments
WHERE   Admissions.pid =Treatments.pid and
        Treatments.diseasedescr = "X"
```

In the above rewritings we have assumed that Results.diagnosis is mapped to Treatments.diseasedescr. However, if we could rewrite the original query to peers StuartDB and HDB, they would have to answer the following queries, respectively.
Qorig_StuartDB:

```
SELECT  age,sex,symptoms,treatrec,drugname
FROM    Patients,Treatments
WHERE   Patients.pid = Treatments.pid and
        Treatments.diseasedescr = "X"
```

Qorig_HDB:

```
SELECT  sex,symptoms,treatdescr,drug
FROM    Admissions,Treatments,Medication
WHERE   Admissions.pid = Treatments.pid and
        Treatments.pid = Medication.pid and
        Treatments.diseasedescr = "X"
```

In the general case of a P2P database system such as the one described above, it is not possible to thoroughly rewrite a query posed on a peer-local schema when it is propagated to another peer, because of the incomplete matching of the two peer-schemas. A non-complete query rewriting is acceptable in our context, even though in a centralized or distributed database system it is not. The high autonomy of the involved peer-databases leads to an acceptable loose consistency level concerning data management, different from the strict consistency imposed on traditional database systems. Despite this fact, it is desirable to limit the inconsistencies during this process as much as possible. As far as query answering is concerned, this means that our goal is to achieve a query rewriting that will lead to the retrieval of the most accurate and complete answers possible.

It is obvious that the rewritings of the original query in StuartDB and HDB maintain more information than the respective rewritings of the query coming from LDB. This extra information is lost in the rewritings of the LDB query version, not because of malfunction of the rewriting mechanism in any of the peers in the propagation path, but because of the poor schema mapping between DavisDB and LDB. Thus, with the approach of query rewriting in every step of the propagation procedure, DavisDB fails not only to retrieve sufficiently accurate answers to the initiated query, but also to spot more convenient candidates for immediate neighbors in the P2P system.

2. ALGORITHM DESCRIPTION

2.1 Methodology

Driven by applications similar to the aforementioned example, we have developed a methodology that combines query rewriting together with automatic schema matching that overcomes obstacles of poor matching in query propagation paths. Our goal is for peers to gradually discover remote nodes affluent in pertinent information that are "hidden" behind peers with very dissimilar data and poor rewriting mechanisms. Moreover, the proposed methodology enables peers to evaluate these promising nodes over multiple queries and decide whether they can benefit from them in terms of relevant information. In such a case, peers can ask to get acquainted with these remote peers. The result is a gradual reformulation of the P2P system towards a structure where acquaintances are between peers with similar schemas and data.

For two versions of a query Q, Q_{v1} and Q_{v2} we define a similarity measure M_{sim} as follows:

$M_{sim}(Q_{v1}, Q_{v2}) : dom(Q_{v1}) \cup dom(Q_{v2}) \rightarrow \Re$

$M_{sim}(Q_{v1}, Q_{v2}) = \frac{|dom(Q_{v1}) \cap dom(Q_{v2})|}{|dom(Q_{v1})|}$

Where Q_{vi} for $i = 1,2$ is either the original version of Q, Q_{orig}, or a rewritten version of it, Q_{rewr}. Also, $dom(Q_{vi})$ is the domain of Q_{vi}, meaning the schema elements together with the relationships among them that it involves. The implementation of M_{sim} depends on the *individual understanding* of the above measure definition of each peer. In consequence, it depends on the schema matching methodology used by each peer but also on their preference for consistent or numerous answers: a peer may need to retrieve answers from the P2P system that are very consistent with its local schema; however, most of the peers that will be asked to send relevant information will have very dissimilar schemas to the schema of the first peer. Thus, peers often have to choose between quality and quantity, a choice that can be denoted and controlled with the implementation of M_{sim}. For our motivating example, we introduce a simplistic metric of schema similarity suitable for queries without wildcards (like stars and variables) and WHERE clauses only with arithmetic operators, such as Q_{orig}:

$M_{sim}(Q_{v1}, Q_{v2}) = \frac{|\{\cup J \in Q_{v1} / \exists sat(J) \in Q_{v2}\}|}{|\{\cup J \in Q_{v1}\}|} \cdot \frac{|\{\cup A \in Q_{v1} / \exists match(A) \in Q_{v2}\}|}{|\{\cup A \in Q_{v1}\}|}$

Where:

-J is a join of Q_{v1}, $sat(J)$ is the satisfaction of J in Q_{v2}, i.e. $sat(J)$ is either a relation or a join of relations in Q_{v2} that can be matched with J,

-A is an attribute of the Q_{v1} (beyond the join attributes), and $match(A)$ is a matching of A to attribute(s) of Q_{v2}.

The above similarity metric computes the fraction of mapped attributes of the SELECT and WHERE clause of the source query Q_{v1} to the target query Q_{v2}, weighted by the fraction of joins in Q_{v1} that can be matched in Q_{v2}. For the motivating example, in all cases all the joins can be mapped to the target schema. Thus:

$M_{sim}(Q_{orig}, Q_{orig\_LDB}) = 2/5 = 0.4$
$M_{sim}(Q_{LDB}, Q_{LDB\_StuartDB}) = 2/2 = 1.0$
$M_{sim}(Q_{LDB}, Q_{LDB\_HDB}) = 1/2 = 0.5$
$M_{sim}(Q_{orig}, Q_{orig\_StuartDB}) = 5/5 = 1.0$
$M_{sim}(Q_{orig}, Q_{orig\_HDB}) = 4/5 = 0.8$

In order to compute the similarity of the original query Q_{orig} and the doubly rewritten versions $Q_{LDB\_StuartDB}$ and Q_{LDB_HDB}, we multiply the similarity values of the successive respective rewritings:

$M_{sim}(Q_{orig}, Q_{LDB\_StuartDB}) = M_{sim}(Q_{orig}, Q_{orig\_LDB}) \cdot$
$M_{sim}(Q_{LDB}, Q_{LDB\_StuartDB}) = 0.4 \cdot 1.0 = 0.4$,
$M_{sim}(Q_{orig}, Q_{LDB\_HDB}) = M_{sim}(Q_{orig}, Q_{orig\_LDB}) \cdot$
$M_{sim}(Q_{LDB}, Q_{LDB\_HDB}) = 0.4 \cdot 0.5 = 0.2$.

Thus:

$M_{sim}(Q_{orig}, Q_{LDB\_StuartDB}) \ll M_{sim}(Q_{orig}, Q_{orig\_StuartDB})$ and
$M_{sim}(Q_{orig}, Q_{LDB\_HDB}) \ll M_{sim}(Q_{orig}, Q_{orig\_HDB})$.

The comparison of the similarity values of a source query to two target versions of it reveals which of the latter captures the information encapsulated in the source query in a more complete way. The above comparisons show that the straightforward rewriting of Q_{orig} in StuartDB and HDB is far better than the rewriting of the respective rewritten version they receive from LDB, Q_{orig_LDB}.

Lets assume that HDB performs a conservative matching and rewriting of Q_{orig}. The produced query is not the complete rewriting Q_{orig_HDB} presented in section 1.1 but:

Q'orig_HDB:

```
SELECT  sex,symptoms
FROM    Admissions,Treatments
WHERE   Admissions.pid = Treatments.pid
        and Treatments.diseasedescr = "X"
```

However, it is obvious that it captures more information than the query produced from rewriting the already rewritten query on LDB: $M_{sim}(Q_{orig}, Q'_{orig\_HDB}) = 2/5 = 0.4 > M_{sim}(Q_{orig}, Q_{LDB\_HDB})$. Hence, we can infer that even a poor automatic query-schema matching may produce a better rewriting than the “safe” successive query reformulation on all nodes of the propagation path.

Beyond M_{sim}, each peer-database has a confidence measure θ that characterizes the overall ability of the peer to be able to “guess” the correct rewritten version of a query for which it has no schema information. Based on this confidence, a peer decides whether to answer the original query for which it has no schema information or the already rewritten query on a schema for which it has mappings. Thus, each peer propagates to its acquaintees neither the initial query nor the rewritten one, but *both* as a pair and the receiving node decides which one to answer according to the confidence measure. Generally, a confidence parameter depends on internal features of a peer:
a) the peer's estimation about its schema matching ability, and
b) the structure of the query: specifically, the amount of information given by the structure of the query for the schema on which it is expressed, so that there can be a certain degree of guarantee that a respective rewriting will be accurate.

Therefore, θ expresses the guarantee for correctness and completeness of a query rewriting based on automatic schema matching. Thus, θ is a function of the evaluated query Q and the structure of the function depends on the peer p: $\theta = \theta_p(Q)$. In our example, according to the proposed query propagation technique, LDB sends to StuartDB and HDB the pair (Q_{orig_LDB}, Q_{orig}) and the latter decide which one to answer according to the value of the respective similarity measure weighted by the respective parameter θ. We in-

troduce this estimation metric as the *coverage* of the target query Q_{v2} on the source query Q_{v1}:
$Cov_p(Q_{v2}, Q_{v1}) = M_{sim_p}(Q_{v1}, Q_{v2}) \cdot \theta_p(Q_{v1})$

Generally, the initial value of θ should be low. The peers of the example are not highly confident for their query-schema matching ability. However, query rewriting based on acquaintance mappings is always complete and correct. Thus:
$\theta_{StuartDB}(Q_{orig}) = \theta_{HDB}(Q_{orig}) = 0.5$ and
$\theta_{LDB}(Q_{orig}) = \theta_{StuartDB}(Q_{LDB}) = \theta_{HDB}(Q_{LDB}) = 1.0$.
The coverages of the rewritten queries versus the original are:
$Cov_{LDB}(Q_{orig\_LDB}, Q_{orig}) = 0.4 \cdot 1.0 = 0.4$,
$Cov_{StuartDB}(Q_{orig\_StuartDB}, Q_{orig}) = 1.0 \cdot 0.5 = 0.5$,
$Cov_{HDB}(Q'_{orig\_HDB}, Q_{orig}) = 0.4 \cdot 0.5 = 0.2$,
$Cov_{StuartDB}(Q_{LDB\_StuartDB}, Q_{orig}) = Cov_{LDB}(Q_{orig\_LDB}, Q_{orig}) \cdot$
$Cov_{StuartDB}(Q_{LDB\_Stuart}, Q_{orig\_LDB}) = 0.4 \cdot 1.0 \cdot 1.0 = 0.4$,
$Cov_{HDB}(Q_{LDB\_HDB}, Q_{orig}) = Cov_{LDB}(Q_{orig\_LDB}, Q_{orig}) \cdot$
$Cov_{HDB}(Q_{LDB\_HDB}, Q_{orig\_LDB}) = 0.4 \cdot 0.5 \cdot 1.0 = 0.2$.
Thus:
$Cov_{StuartDB}(Q_{orig\_StuartDB}, Q_{orig}) > Cov_{StuartDB}(Q_{LDB\_StuartDB}, Q_{orig})$, and
$Cov_{HDB}(Q'_{orig\_HDB}, Q_{orig}) = Cov_{HDB}(Q_{LDB\_HDB}, Q_{orig})$.

Based on the computed coverages, StuartDB decides to answer the rewriting of the original query Q_{orig_Stuart}, but HDB, because of the tie of the calculated coverages, decides to answer Q_{LDB_HDB} and not the rewriting Q'_{orig_HDB}. Each node that answers a query sends back a packet with the original query Q, the successively rewritten version Q_{sr}, the automatic matched-rewritten version Q_{ar} and the resulted tuples, Res with the indication of which one of the two rewritings was finally answered:
$Ans(Q, (Q_{ar}, Q_{sr}), Res(Q_x))$, $x = ar$ or sr.
StuartDB and HDB send back to DavisDB the following replies:
$Ans(Q_{orig}, (Q_{orig\_StuartDB}, Q_{LDB\_StuartDB}), Res(Q_{orig\_StuartDB}))$,
$Ans(Q_{orig}, (Q_{orig\_HDB}, Q_{LDB\_HDB}), Res(Q_{LDB\_HDB}))$.

A node that receives an answer evaluates the result and the two rewritings of the query according to the following function:
$Ev(Ans) = Cov(Q_x, Q) \cdot max(1, w_Q \cdot |Res(Q_x)|), x = ar$ or sr,
where the multiplier $max(1, w_Q \cdot |Res(Q_x)|)$ gives the option to either take into consideration the number of returned tuples weighted by w_Q, or not.
Also: $Cov(Q_x, Q) = M_{sim}(Q, Q_x) \cdot \theta_{local}(Q)$, where $\theta_{local}(Q)$ is a special θ parameter that represents, on behalf of the peer that initiates Q: a) the estimation of how easily other peers can correctly and completely rewrite Q, and b) the peer's requirement for accurate/complete answers to Q. Thus, high values of $\theta_{local}(Q)$ show that the query-initiating peer estimates that Q is easy to rewrite and answers that are not fully accurate are welcome, whereas low values indicate the opposite. Consequently, the value of $\theta_{local}(Q)$ will be high if we want to boost the $\theta(Q)$ values on other peer-databases and encourage them to risk more during the automatic query-schema matching procedure, or $\theta_{local}(Q)$ will be low otherwise. It is straightforward that for acquaintees or successively rewritten queries: θ_{local}=θ_{succ}=1.0. We remind that M_{sim} is implemented differently in each peer and the similarity evaluation of two queries depends on which one of them is mapped to the other, i.e., for which one the matching mechanism is aware of the respective schema. Thus, in general it is $M_{sim}(Q_x, Q) \neq M_{sim}(Q, Q_x)$ and $M_{sim_i}(Q_x, Q) \neq M_{sim_j}(Q_x, Q)$ where $i \neq j$ for peers i, j. For simplicity, we assume the same implementation of M_{sim} on all peers of the example and that the similarity outcome is independent of the database on which the query matching is performed.

For the motivating example, we assume that the number of tuples of the results does not influence the evaluation (i.e., $w_Q = 0$ for all the answers). Also, we assume that DavisDB estimates that Q_{orig} is easy to be "decrypted", i.e. the elements of Q_{orig} are reasonably named and it has a simple structure. Also, the user of DavisDB prefers (up to a point) to sacrifice quality for quantity, i.e. prefers to receive answers that are not quite accurate than not to receive enough of them. Based on this reasoning, the selected value for $\theta_{local}(Q_{orig})$ is 0.8. Thus:
$Cov_{DavisDB}(Q_{orig\_LDB}, Q_{orig}) = M_{sim}(Q_{orig}, Q_{orig\_LDB}) \cdot$
$\theta_{succ}(Q_{orig}) = 0.4 \cdot 1.0 = 0.4$
$Cov_{DavisDB}(Q_{orig\_StuartDB}, Q_{orig}) = M_{sim}(Q_{orig}, Q_{orig\_StuartDB}) \cdot$
$\theta_{local}(Q_{orig}) = 1.0 \cdot 0.8 = 0.8$
$Cov_{DavisDB}(Q'_{orig\_HDB}, Q_{orig}) = M_{sim}(Q_{orig}, Q'_{orig\_HDB}) \cdot$
$\theta_{local}(Q_{orig}) = 0.4 \cdot 0.8 = 0.32$
$Cov_{DavisDB}(Q_{LDB\_HDB}, Q_{orig}) = M_{sim}(Q_{orig}, Q_{LDB\_HDB}) \cdot$
$\theta_{succ}(Q_{orig}) = 0.2 \cdot 1.0 = 0.2$
Hence:
$Cov_{DavisDB}(Q_{orig\_StuartDB}, Q_{orig}) > Cov_{HDB}(Q_{orig\_StuartDB}, Q_{orig})$ and $Cov_{DavisDB}(Q'_{orig\_HDB}, Q_{orig}) > Cov_{HDB}(Q'_{orig\_HDB}, Q_{orig})$.

DavisDB sends back to StuartDB and HDB the respective $Cov_{DavisDB}$ as computed above, together with the estimation of $\theta_{local}(Q_{orig})$. Assume now that StuartDB and HDB again host Q_{orig} initiated by DavisDB. Taking the previous $\theta_{local}(Q_{orig})$ estimation into account, they increase the $\theta(Q_{orig})$ value and loosen up the matching mechanism in order to discover more matchings between Q_{orig} and their schema.

StuartDB increases $\theta_{StuartDB}(Q_{orig})$ to 0.8 and computes the new coverage value since the respective M_{sim} is already 1.0:
$Cov_{StuartDB}(Q_{orig\_StuartDB}, Q_{orig}) = 1.0 \cdot 0.8 = 0.8$ and sends it back to DavisDB together with new results. At this point:
$Cov_{DavisDB}(Q_{orig\_StuartDB}, Q_{orig}) = Cov_{HDB}(Q_{orig\_StuartDB}, Q_{orig})$.
Thus, StuartDB and DavisDB gradually come to an agreement of their estimation about StuartDB's ability of answering Q_{orig}.

The HDB database, encouraged by DavisDB, loosens the matching procedure and produces the new rewritten query:
Q"orig_HDB:

```
SELECT  sex,symptoms,treatdescr
FROM    Admissions,Treatments
WHERE   Admissions.pid = Treatments.pid
        and Treatments.diseasedescr = "X"
```

HDB achieved to match 'treatdescr' to 'treatrec' but failed again to match 'drugname' to 'drug'. It decides to increase $\theta_{HDB}(Q_{orig})$ to 0.7. The new coverage is:
$Cov_{StuartDB}(Q''_{orig}, Q_{orig\_HDB}) = (3/5) \cdot 0.7 = 0.6 \cdot 0.7 = 0.42$.
The new evaluation on DavisDB produces:
$Cov_{DavisDB}(Q''_{orig\_HDB}, Q_{orig}) = 0.6 \cdot 0.8 = 0.48$. Again:
$Cov_{DavisDB}(Q''_{orig\_HDB}, Q_{orig}) > Cov_{HDB}(Q''_{orig\_HDB}, Q_{orig})$.

If Q_{orig} is hosted for a third time in HDB, the latter will decide to risk even more on the query-schema matching procedure and also increase $\theta_{HDB}(Q_{orig})$ to 0.8. Thus, the third rewritten version will be Q_{orig_HDB} presented in section 1.1 in which HDB achieved to discover all possible matchings. For Q_{orig_HDB} DavisDB and HDB come to an agreement:
$Cov_{DavisDB}(Q_{orig\_HDB}, Q_{orig}) = 0.64 = Cov_{HDB}(Q_{orig\_HDB}, Q_{orig})$

Note that peers that perform automatic query-schema matching for a query Q are not allowed to set the $\theta(Q)$ to a value higher than the respective $\theta_{local}(Q)$, which they receive as feedback from the query-initiating peer. In this way the $\theta_{local}(Q)$ feedback value controls the boosting or declining of $\theta(Q)$ and the encouragement or discouragement of Q-matching on answering peers. Also, the increment step of a θ value influences the number of recursions of the procedure. Hence, a low such step gives the opportunity to the peer to get more potentially good evaluations from the requester and ameliorate its matching procedure gradually and to a finer degree.

For example, if HDB decides to set $\theta_{HDB} = \theta_{local}$ when receiving the first feedback from DavisDB, then it agrees to the latter's estimation right away, losing the chance to refine the matching for Q_{orig}. Accordingly, the described methodology achieves two goals: a) the gradual training of peers in automatic schema matching for queries initiated by specific peers and b) the discovery of nodes concealed in the network that are convenient for acquaintees using only the queries posed in the P2P system. Therefore, this methodology can be used in order to gradually cluster the P2P network so that peers with common interests are close enough or acquainted. Generally, a peer evaluates the ability of another peer to fulfill its information needs with the following function:

$$Eval(Nid) = \sum_j w_j \frac{\sum_{i=0}^{k(Q_j)} Ev_{ji}(Q_j, Q_{j\_rewr}, Res(Q_{j\_rewr}))}{k(Q_j)}, \sum_j w_j = 1$$

The above function is the weighted average of j queries posed on peer Nid k times (depending on the query). In our motivating example DavisDB may decide to become acquainted with StuartDB and/or HDB, since it has discovered that the latter can satisfactorily fulfill its need for information.

2.2 The feasibility context of automatic schema matching

Schema matching is a fundamental issue in the database field, from database integration and warehousing to the newly proposed P2P data management systems. As discussed in [6], most approaches to this problem are semi-automatic, in that they assume human tuning of parameters and final refinement of the results. This is also the case in some recent P2P data management approaches (e.g., [7]).

However, there are cases where schema and query to schema matching can be solved automatically. If we restrict the domain of input of the automatic schema procedure in order to impound cases of queries leading to complicated or fuzzy rewritings, then it is viable to produce safe matchings within a small error tolerance. Of course, in a P2P database system, this error tolerance together with the matching and rewriting techniques used, depend on the individuality of each peer. In our motivating example, we consider queries with arithmetic operators (i.e., without nesting, aggregation, grouping, wildcards, etc). More complex queries demand more sophisticated matching procedures and, in our case, more refined M_{sim} functions.

Moreover, the similarity between source and target schemas observed in domain-specific applications is a step further towards automatic schema matching. This situation is possible in real life applications where peers may store their data in similar schemas because: a) they store the same kind of data, b) there are specific policies for designing databases of specific domains and c) there are popular database products used in various fields. In our example, private doctors in general and specialty doctors in particular have to store the same kind of information, which is not of a wide variety: i.e., they care about listing their patients, their medical histories, their patients' visits, their own diagnosis and their own prescriptions for their patients. Moreover, it could be the case that some of these specialists use the same commercial tool to store their information. Obviously, for peer-databases with similar schemas such as DavisDB and StuartDB, query rewriting can be done easily, even with speculations of the schema mapping.

Our approach incorporates the inherent difficulties of automatic schema matching. The θ parameters provide control of the limits of both schema and query structure for which peers perform automatic matching. For example, more complicated queries imply a more conservative θ value. Moreover, peers characterize and accept automatic matchings and rewritings according to their individual standards of query-match and query-answer quality. Finally, our proposal enables the enhancement of this process by having the query sources continually and automatically evaluating the outcome. Towards this line, the reinforcement or weakening of θ values enables a recursive training of the schema matching mechanisms based on feedback, just like supervised learning in neural networks.

2.3 Protocol Internals

In the following we describe our algorithm's internals, specifically the query routing scheme, the maintenance of local knowledge and the addition/deletion of acquaintances.

1) Routing: Our method utilizes informed walks with a TTL parameter in order to propagate queries to nodes in the overlay. The requester deploys k walkers, each following independent paths. A node forwards to the neighbor(s) whose schemas have the highest similarity value relative to the query in hand.

2) Local Knowledge: Each peer keeps schema information for all its one-hop neighbors in the overlay. Moreover, a peer q computes a value $\theta_q(Q)$ which represents the peer's ability to translate a query Q to its local schema. If Q comes from a neighbor, then $\theta_q(Q) = 1.0$. The θ_q values depend in part on the *reinforcement* mechanism we described: A requester p that receives an answer from a non-neighbor q can either reinforce or weaken θ_q. This depends on whether p decides that q's interpretation of the query is satisfying or not. Each peer keeps track of all recent transactions with other peers that have returned query results. Specifically, it keeps the *Eval* value averaged over the current number of results from each of those peers. The corresponding entries are updated whenever an answer is received. In case of limited storage/memory, *Eval* entries are replaced using the LRU policy. Assuming the created queries are equally important to the requester, we model the θ update operation in the following manner: If the requester is satisfied with the rewriting of a query (i.e., high *Eval* value), it signals for an increase $\theta_q \leftarrow \theta_q + |\eta|$, if $\theta_q < \theta_{p_{local}}$, otherwise $\theta_q \leftarrow \theta_{p_{local}}$. For a small $|\eta|$, many recursions are needed until the bound of $\theta_{p_{local}}$ is reached.

3) Adding/dropping acquaintances: A neighbor q is added whenever $\theta_q = \theta_{local}$, provided we have received at least T replies from q. We also define a maximum number of connections per peer MAXDEGREE, which forces a neighbor addition to be preceded by the dropping of the neighbor with the smallest schema similarity if that limit is reached. A link is dropped whenever the schema similarity between two peers is below $Min_{M_{sim}}$, provided the degrees of both the node and q are at least MINDEGREE. This ensures that peers do not get disconnected from the network.

Our protocol requires $O(\mathcal{A} + MAXDEGREE)$ space per node. $\mathcal{A}$ represents the maximum number of peers that return answers to a node's queries ($\mathcal{A} = O(k \cdot TTL)$ on average). The second factor represents the stored schema information for each neighbor.

3. PERFORMANCE EVALUATION

To evaluate the performance of our method, we use a message-level simulator written in C. We assume a *pure* P2P model, where all peers can make and forward requests. Each peer is only aware of its 1-hop neighbors in the overlay. By default, we randomly choose 100 nodes that play the role of the requesters, each making 200 queries to the system. We present results for *random* (default) and *power-law* graphs, utilizing the *BRITE* [8] and *Inet-3.0* [9] topology generators respectively. Our random graphs have 1,000 nodes, while power-law graphs have 3–10K nodes (both seem realistic values regarding our motivating application), with average node degrees around 4. Results are averaged over 20 graphs from each type and size, with several runs in each.

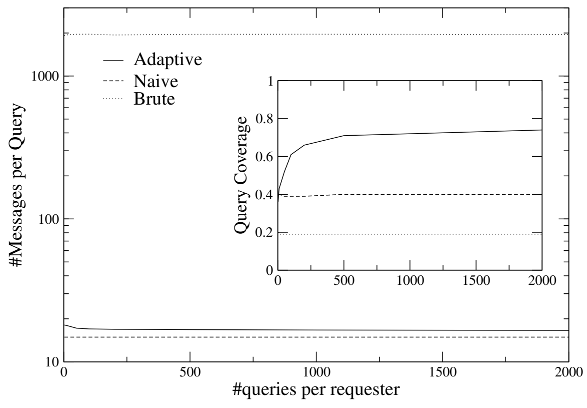

Figure 2: Query coverage and bandwidth consumption over variable number of queries per requester

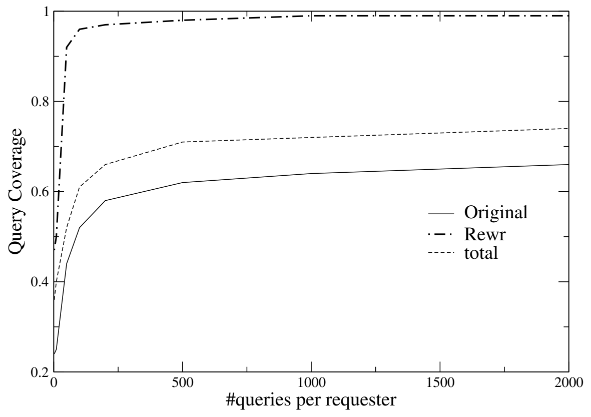

Figure 3: Query coverage for answers to the rewritten and original queries as the number of queries increases

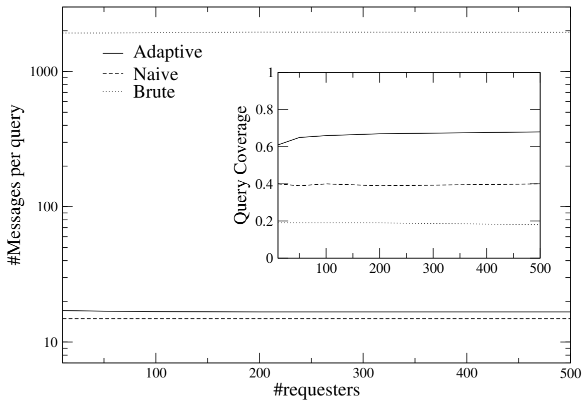

Figure 4: Query coverage and bandwidth consumption over variable number of requester nodes

Table 1: Performance under varying θ_{local} values

	Coverage	**Answers**	**d**
$\theta_{local} = 0.60$	0.69	2.8	5.5
$\theta_{local} = 0.70$	0.67	3.1	5.0
$\theta_{local} = 0.80$	0.42	3.3	4.3
$\theta_{local} = 0.90$	0.26	3.9	4.0

We assume a total number of 100 attributes, which represent fields of the tables that each peer shares and forms queries from. We utilize a uniform distribution to model both attribute placement and attribute selection during query formulation. By default, each peer obtains about 10 attributes and queries contain 3 attributes on average. Initially, all θ values for queries from non-neighbors are set to 0.5, while $w_Q = 0$. By default, we set $\theta_{local} = 0.7$ for all nodes, $\eta = 0.05$, $Min_{M_{Sim}} = 0.1$ and $T = 10$ queries. We allow at most 10 new extra links per node, while no links are dropped for nodes with less than 3 neighbors. Finally, we set $TTL = 7$ and the number of deployed walkers equal to 2.

We compare our method (named *Adaptive*) against two different schemes: The naive successive rewriting technique described in the introduction (*Naive*) that uses the same routing scheme as *Adaptive* and the brute-force method, where the requester performs a Gnutella-style flood [11], with each peer trying to answer the original version of the query (*Brute*). Our main comparison metrics are the average query coverage (in terms of the eventually returned attributes) and the average number of messages produced per request.

3.1 Simulation Results

In the first experiment (Figure 2) , we show the performance of our algorithm by varying the number of queries posed per requester node. Our method manages to return far more accurate results, answering 2 of the 3 queried attributes on average. The accuracy increases fast as more queries are created, since new acquaintances are made and neighbors with no contribution are dropped. Both *Naive* and *Brute* operate in a blind fashion, regardless of the workload, on the opposite ends of the spectrum: On one hand, *Brute* contacts all nodes within range; however answering the original query returns poor results. On the other hand, it may seem that always answering the rewritten query (like *Naive* does) yields acceptable results (average coverage about 0.4). This is not the case though: The method's relatively high average coverage is due to the fact that it gets an answer from less than one node on average, and that node is almost always a neighbor (average hop distance equal to 1.2). Conversely, using *Adaptive*, results are retrieved from 3–5 times as many nodes relative to the *Naive* approach.

Figure 3 shows how the accuracy of the received results for both the rewritten and the original query ranges in this experiment. Our method performs a partial restructuring around the query initiators, the results of which are shown in the coverage rates as more queries enter the system. It is interesting to notice that not only the original queries get answered with more precision, but this is also true for their rewritten versions. While only 20-30% of the results are answers to rewritten queries, our clustering mechanism also helps into bringing more information-rich nodes closer to requesters by making the forwarding more competitive.

Our method is almost as bandwidth-efficient as *Naive*, while it produces two orders of magnitude less messages than *Brute*. The few additional messages compared to *Naive* are due to the communication between sources and requesters during the θ reinforcement mechanism, as well as the message exchange when a new acquaintance is made. We observe that, regardless of the workload imposed by each of the requesters, no extra traffic is added to the network.

Next, we evaluate the scalability of our method by varying the number of requester nodes from 1% to 50% of the overlay and present the results in Figure 4. Our scheme proves very bandwidth-efficient in all runs, sending out only two extra messages compared to *Naive*, regardless of the number of requesters. Peers route messages and manage their neighborhoods in a completely distributed manner, so an increase in active peers puts no extra load on the network. In terms of the coverage of the returned results, *Adaptive* proves again more efficient and even succeeds in increasing query coverage with more requesters.

Table 1 presents how the behavior of our scheme is affected by a change in θ_{local} from 0.6–0.9 (given that the default $\theta = 0.5$). Our goal is to choose a value such that a high query coverage is achieved while avoiding unnecessary new acquaintances. We can observe that the smaller the θ_{local} values the more the average degree d increases, as acquaintances are added more frequently. While coverage and d decrease with high θ_{local} values, the number of distinct peers that answer the queries slightly increases. This is due to the fact that with smaller θ_{local} (therefore denser overlay), the "collisions" between walkers increase, thus reducing the number of discovered sources. Collisions, or duplicate messages, are messages regarding the same request that arrive at the same peer due to network cycles. Our choice $\theta_{local} = 0.7$ yields a good all-around performance. A similar effect is observed if we vary the average number of attributes maintained per peer: For smaller values, the queries are more accurate, but fewer results are returned since fewer nodes have similar schemas.

Finally, Table 2 presents results from comparing the three methods using 3 sets of power-law graphs, with $N = \{3K, 5K, 10K\}$ nodes and average degrees $d = \{3.7, 3.9, 4.4\}$ respectively. We notice that an increase in the number of nodes does not affect the performance of our method. The distinctiveness of the power-law topologies, where about 35% of the peers have degree one, forces fewer paths to be used compared to the random topologies. This explains the slight decrease in the average answer coverage (about 10%) compared to the random topologies. In contrast, the *Naive* scheme now discovers less than 0.7 sources per query, while *Brute* produces thousands of extra messages each time the size of the network increases.

Table 2: Comparison for power-law graphs of variable size

method	*N*	**Coverage**	**Messages**	**d**
Adaptive	3K	0.60	16.8	7.3
	5K	0.61	16.3	5.3
	10K	0.61	16.8	5.0
Naive	3K	0.40	8.9	3.7
	5K	0.41	9.7	3.9
	10K	0.40	9.8	4.4
Brute	3K	0.20	7026	3.7
	5K	0.19	11916	3.9
	10K	0.19	24938	4.4

4. RELATED WORK

In [12], the authors propose optimization techniques for query reformulation in P2P data management systems. They focus on minimizing the rewriting of a query and pruning the respective propagation path in order to avoid redundant reformulations. Additionally, it is indicated that pre-computation of the query reformulation path-tree proves to accelerate the reformulation procedure despite the disadvantage of the necessary maintenance of pre-computed mappings. Our approach is specifically designed for large-scale unstructured overlays. First, it evades reformulation at peers poor in query-relevant information by adaptively choosing the version of the query to be answered. Moreover, while the optimizations in [12] require central knowledge of the system structure, our scheme enables nodes to operate in a completely decentralized fashion, utilizing the standard lookup operations to refine their local knowledge.

PeerDB [7] facilitates relational data sharing without any schema knowledge. Query matching and rewriting is based on keywords (provided by the users). A two-step process is described: First all nodes within a TTL radius are contacted, returning prospective answer metadata. Then the user selects the ones that are relevant to the local query and the requester directly contacts the selected sources and asks for the results to the various rewritten versions of the query. Instead, our approach employs an automated technique based on a combination of successive query rewriting and query-schema matching, while it utilizes bandwidth-efficient walks instead of the costly flooding scheme.

The works in [4] and [3] deal with data exchange between peers. Ref. [4] presents a significant approach to the heterogeneity issue in P2P data management and proposes a language for schema mediation between peers. Also, the authors present an algorithm for query reformulation based on local-as-view as well as global-as-view query answering. In [3], the authors describe mechanisms for the declaration of data exchange policies on-the-fly based on ECA rules. They also propose a general architecture for peer-databases and elaborate on the establishment and abolishment of acquaintances between peers.

5. SUMMARY

This paper is a first approach to solve the query degradation problem in P2P data management systems in the absence of global schema information. By allowing peers to select the appropriate rewritten version of the query to answer, we discover remote peers on query propagation paths that are rich in interesting information but veiled by poor path predecessors. Using these discoveries, nodes seeking or holding similar information are gradually interconnected, increasing the quality of the returned results. Our solution is specifically suited for dynamic, unstructured environments, since it is adaptive, bandwidth-efficient and operates in a complete decentralized manner.

The next steps of the development of the presented process include theoretic as well as experimental work. Specifically, we plan to design an automatic query-schema matching process adapted to the particular needs of the described context. Additionally, we will elaborate on the structure of the θ parameter as a function of the characteristics of the schema matching mechanism of the peer and the structure of the query. Beyond these, we are going to further test the process in contexts similar to our motivating example.

ACKNOWLEDGMENTS

This material is based upon work supported in part by the U.S. Army Research Laboratory and the U.S. Army Research Office under contract/grant number DAAD19-01-1-0494

6. REFERENCES

[1] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content Addressable Network. In *SIGCOMM*, 2001.

[2] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *SIGCOMM*, 2001.

[3] V. Kantere, I. Kiringa, J. Mylopoulos, A. Kementsientidis, and M. Arenas. Coordinating P2P Databases Using ECA Rules. In *DBISP2P*, 2003.

[4] A. Halevy, Z. Ives, D. Suciu, and I. Tatarinov. Schema Mediation in Peer Data Management Systems. In *ICDE*, 2003.

[5] R.J. Miller A. Kementsietsidis, M. Arenas. Mapping Data in Peer-to-Peer Systems: Semantics and Algorithmic Issues. In *SIGMOD*, 2003.

[6] E. Rahm and P.Bernstein. A Survey of Approaches to Automatic Schema Matching. In *VLDB Journal*, 2001.

[7] B. Ooi, Y. Shu, K.L. Tan, and A.Y. Zhou. PeerDB: A P2P-based System for Distributed Data Sharing. In *ICDE*, 2003.

[8] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: An Approach to Universal Topology Generation. In *MASCOTS*, 2001.

[9] C. Jin, Q. Chen, and S. Jamin. Inet: Internet Topology Generator. Technical Report CSE-TR443-00, Department of EECS, University of Michigan, 2000.

[10] M. Ripeanu and Ian Foster. Mapping the gnutella network: Macroscopic properties of large-scale peer-to-peer systems. In *IPTPS*, 2002.

[11] Gnutella website: http://gnutella.wego.com.

[12] I. Tatarinov and A.Halevy. Efficient Query Reformulation in Peer-Data Management Systems. In *SIGMOD*, 2004.