Approximating Query Completeness by Predicting the Number of Answers in DHT-based Web Applications[*]

Marcel Karnstedt,
Kai-Uwe Sattler, Michael Haß
TU Ilmenau, Germany
{first.last}@tu-ilmenau.de

Manfred Hauswirth,
Brahmananda Sapkota
DERI, NUI Galway, Ireland
{first.last}@deri.org

Roman Schmidt
EPFL, Switzerland
{first.last}@epfl.ch

ABSTRACT

Due to the rapid development of the Web, applications based on the P2P paradigm gain more and more interest. Recently, such systems start to evolve to adopt standard database functionalities in terms of complex query processing support. This goes far beyond simple key lookups, as provided by standard DHT systems, which makes estimating the completeness of query answers a crucial challenge. In this paper, we discuss the semantics of completeness for complex queries in P2P database systems and propose methods based on the notion of routing graphs for estimating the number of expected query answers. Further, we discuss probabilistic guarantees for the estimated values and evaluate the proposed methods through an implemented system.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed Databases; H.3.3 [**Information Search and Retrieval**]: Search process

General Terms

Algorithms, Management, Performance

1. INTRODUCTION

Many new applications on the Web are based on the idea of collecting and combining large public data sets and services. In such public data management applications, the information, its structure and its semantics in many cases are the result of the collaborative effort of the participants. Examples of such applications are social networks, e.g., friend-of-a-friend networks, distributed recommender systems, distributed directory and index services, and sharing of sensor data. These applications typically require the indexing and management of data distributed over a large number of independent data stores, which is a typical scenario targeted by overlay networks. The most efficient family of overlay networks, distributed hash tables (DHT), so far have only been applicable to a certain degree in these scenarios, as support for managing and querying structured data in DHTs still is limited. P2P data management is inherently open world: While processing a query, peers can fail, leave or join the network, or simply send no or a delayed answer [5]. Though this can be mitigated by replication and delay-tolerant query techniques, there is no guarantee that all answers which potentially exist can be returned.

Thus, we argue that estimating the completeness of query answers is a key aspect of reliable query processing in P2P databases. Often, an approximated, but prompt estimation is satisfying for the user. We achieve this by estimating the number of query answers without explicitly analyzing the actual content of each answer. Note that we are not trying to guarantee completeness, neither we want to improve the functionality of the underlying DHT. Instead we only get what is available at query time ("in situ querying") but are able to assess the completeness of this result. Furthermore, we aim at an incremental and online refinement of the estimation. The main contribution of our work is the development of lightweight techniques for providing reliable information about the result completeness of complex database-like queries in a dynamic and unreliable P2P environment. This leverages modern Web applications without the need for precomputed data summaries or detailed global information.

2. COMPLETENESS IN P2P SYSTEMS

The notion of completeness has been discussed in the literature mainly in the context of data quality, e.g., in [10, 12] where completeness is typically understood as the ratio of answer set size to the total amount of known data. However, this definition does not apply in our context as it requires the knowledge of the total amount of data in the system and relies on the closed world assumption. To come to a meaningful definition which is applicable in our context we have to distinguish between data availability and completeness of query answering. We introduce the following refined definitions.

Data availability denotes the classical notion of completeness, i.e., what amount of data w.r.t. the real world is stored in a system. Availability can be seen (more or less) as a static aspect if we assume that good replication tech-

[*]The work presented in this paper was (partly) carried out in the frameworks of the EU project TripCom (FP6-IST-4-027324-STP), the Líon project supported by the Science Foundation Ireland under Grant No. SFI/02/CE1/I131 and the EPFL Center for Global Computing and supported by the Swiss National Funding Agency OFES as part of the European project NEPOMUK No FP6-027705.

WIDM'08, October 30, 2008, Napa Valley, California, USA.
Copyright 2008 ACM 978-1-60558-260-3/08/10 ...$5.00.

niques are used to deal with churn (peer failures or peers joining/leaving) [2].

Completeness of query answers, in contrast, denotes the ratio of the amount of data of the answer $\mathcal{A}$ to a query Q and the amount of answers $\overline{\mathcal{A}}$ we would get if all peers participating at query time t would respond: $C(Q)[t] = \frac{|\mathcal{A}(Q)[t]|}{|\overline{\mathcal{A}}(Q)[t]|}$. Note, that this also holds true for any sub-query of Q, i.e., for each intermediate operator the completeness of its result can be estimated.

Our notion of completeness deals with the dynamic aspects inherent in P2P systems and addresses the problem of churn during query processing. Note, that replication helps to solve the problem of "guaranteeing" that data is available over time but may not help if peers which are expected to process portions of a certain query fail or leave during processing [2].

Figure 1 shows this problem in a real-world situation from a Planet-Lab experiment: it shows the fraction of received results and expected results versus time for three queries each initiated 15 times.

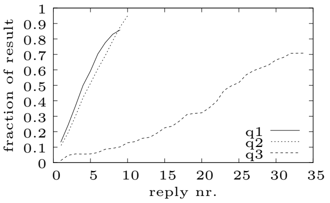


Figure 1: Result quality in Planet-Lab

Depending on the complexity of the query, the average result size is significantly below the expected size. This is normal in such systems, but cannot be recognized nor handled accordingly without an accurate completeness estimation.

For estimating completeness of query answers in P2P systems we can basically distinguish two main approaches: estimating (1) on *data level* and (2) on *peer level*. The data level approach is based on data summaries representing the distribution of data of all or only a subset of peers. Using this information, the received answers can be compared to the expected answers to estimate the ratio between these two sets. In contrast, the peer level approach does not count the received data items in the answer but the responding peers. Based on this and on information about the structure of the DHT, the numbers of the responding peers and the expected number of responding peers are compared. The underlying assumption is that data is balanced among the peers and that a linear correlation between the number of failed peers and the resulting miss of data from the expected answers exists. This assumption holds in DHTs providing load balancing features such as P-Grid [1]. For peer level estimation the above definition of completeness has to be modified accordingly: $C(Q)[t] = \frac{|\mathcal{P}(Q)[t]|}{|\overline{\mathcal{P}}(Q)[t]|}$ where $\mathcal{P}$ and $\overline{\mathcal{P}}$ denote the set of all peers responding to query Q and the set of expected peers responding to Q resp.

In this paper, we focus on this peer level approach for the following reason: because there is no need to maintain summary information about data beside the routing information which is needed anyway, the peer level approach is much cheaper and better suited for large-scale and dynamic networks.

3. RELATED WORK

Existing works on completeness estimation exists both in relational and distributed database systems, e.g., [10]. Most of these works apply completeness estimation from a different point of view than we do. The aimed application scenario is usually located in the area of information integration and focuses on data quality. We focus on completeness as the fraction of received results and expected results.

A recent system dealing with the problem of completeness estimation in structured overlays is Seaweed [11], which facilitates to estimate query completeness directly on data level. For this, Seaweed uses data summaries representing the data distribution of neighboring peers, which are regularly updated using a heartbeat mechanism. We assume a wider variety of different query types and query processing strategies, exceeding the idea of broadcast and spanning trees. Our method for completeness estimation is designed specifically to be generic in order to support other modern processing strategies, such as similarity and skyline search. The idea behind Seaweed is to provide delay-aware querying, which means data summaries and availability models of currently unavailable nodes are used in order to predict query completeness and response times. In contrast to this, we use the DHT to handle unavailable and new nodes, usually by managing replicated data objects and corresponding routing techniques. We primarily use completeness estimation to measure the currently received portion of a query result in the context of slow peers. Unavailable peer data is handled in a best-effort manner, where completeness estimation helps to rate the significance of partial results. The idea of regularly maintained data summaries is a very interesting possibility to extend our approach.

Concerning P2P query processing, the most related work is PIER [6]. The authors propose a similar database-like style of query processing as we do (see Section 4). However, until now, PIER provides no mechanisms for estimating query completeness, but the authors mention this as future work. A couple of distributed triple stores, e.g., [3], are based on a similar data model, preferably used to manage and query RDF data. Despite that they provide sophisticated query processing capabilities as well, query processing is based on a rather different approach and completeness estimation is not supported.

4. BASIC QUERY PROCESSING

We implemented our approach for estimating query completeness in the UniStore system [8]. This section summarizes the fundamentals of query processing in this system required to understand the subsequent sections which present our approach. We used the implementation in UniStore as a proof-of-concept, but would like to point out that our approach is generally applicable to DHT-based query systems.

The philosophy of UniStore is similar to that of PIER [6]. Both systems aim at supporting structured queries on structured data from multiple users, managed in a structured overlay, i.e., a DHT. overlay). Queries are transformed into according query plans and processed relying only on routing functionality provided by the DHT. In contrast to PIER, UniStore targets very heterogeneous environments, typical for the Web. UniStore uses a triple-based storage model, where each triple (o, a, v) includes a (system-generated) object ID o (OID for short), an attribute name a (schema level) and a value v for this attribute (instance level). The benefits of this storage model are its flexibility and it is self-descriptive and easily extensible. All triples are indexed multiple times in the DHT to support various functionalities: For instance, we build an index on the OIDs to support efficient tuple reconstruction, an index on the concatena-

Operator	Description
ξ_{A}	extracts all tuples with attribute `A`
ω_{B}	materializes attribute `B` for each input tuple, if exists (join on OID)
$\bowtie_{A=B}$	equal join on $A = B$ ($\sim$: similarity join)
α	aggregation operator
op^{RQ}	physical operator *op* based on contacting all responsible peers using a range query (DOR)
op^{ParOID}	physical operator *op* based on parallel direct lookups for input OIDs (FNR)

Table 1: Used operators

tion of a and v for prefix and range-queries over attributes, a *qgram* index to support string similarity, etc. UniStore also supports special operators which are helpful in heterogeneous environments such as similarity queries, ranking queries, skyline queries, etc.

UniStore's query language supports the triple model very similar to RDF query languages. For processing queries, the query engine chooses among different processing strategies by applying different access paths (different indexes build on top of the DHT) Multiple instances of a query plan (called sub-plans) are shipped where relevant data is expected to be (using the DHT's hash functions). At these peers the plans are processed, physical operators are successively replaced by (partial) result data and the plan is forwarded again. Finally, a reply (which can be empty) is sent to the query initiator for every finished sub-plan (i.e., the last plan operator has been processed). Thus, in order to estimate completeness on peer level, we have to determine the number of plans generated during the processing of one query.

We use a graph-based notation for representing query plans, where each node represents a plan operator symbolized by a Greek letter. We will use the following abbreviations and symbols: DOR (Dynamic Overlay Routing), FNR (Fixed Number Routing), R (estimated number of final replies), Ŗ (actual number of final replies), r (number of currently received replies), and L (maximal routing level of a routing graph). Table 1 lists the operators used in the following sections.

Query plans and operators used in this work are chosen in order to illustrate the proposed approach as much intuitive as possible, and to capture the three different general processing strategies proposed in related works: sequential (peers to finish an operator are contacted in sequence), intra-operator parallel (all peers needed to finish one operator are contacted in parallel), and inter-operator parallel (allows for processing branches in parallel, e.g., both input sides of a join).

Query processing in P2P networks depends on the underlying topographic structure and the routing principles of the concrete system. For equality queries, the query originator only has to expect no or one reply from the peer holding matching data. For more complex query types, e.g., range queries and similarity queries, the query initiator is usually unaware of the number of peers involved in resolving its query and will receive an *a priori* unknown number of query replies. It is therefore not possible to determine when the result set is final.

5. APPROACH

This section provides a general view on the proposed scheme for completeness estimation using a small example. In [7] we present a more high-level description.

5.1 Routing Methods

We observed that the different processing strategies can be classified by only two relevant routing methods. The crucial difference is whether a peer starting a routing knows about how many peers will be contacted (i.e., the out-degree of the corresponding routing point is known a priori) or not. Following, we distinguish:

1. Dynamic Overlay Routing (DOR): The out-degree is not known a priori. Usually, this is done by addressing a certain key space and let the overlay decide how to forward the separate plans to all peers in this key space. In most cases, this means that the routing of plans is independent of already determined data. We apply this concept by issuing range queries in order to extract or materialize certain attributes. The number of peers in a queried range is not known, though the range itself is.

2. Fixed Number Routing (FNR): This class of routing methods covers all processing strategies where the peer starting a routing knows the number of generated sub-plans. In most cases, the routing of plans is dependent of intermediate input data. As an example, imagine a nested-loop like processing of a join using an appropriate attribute-value index.

In the following, we will base our explanations on these two classes of routing methods, because the processing strategies implemented in our system perfectly match them. Special cases, like a mixture of both approaches (e.g., issuing a fixed number of range queries depending on the input data) or simplified routing methods (e.g., if the number of peers contacted in one processing step is already known at query planning time) integrate easily in the proposed framework. For example, an operator processed sequentially corresponds to an FNR with a fixed out-degree of 1 at each routing point.

5.2 General Idea

The idea is to build a *routing graph* that represents the peers and connections a query travels during processing. Each node in the graph, a *routing point*, represents one peer involved in query processing. Actually, the graph is a tree. The number of leafs in the tree is Ŗ, the number of replies we need to estimate. Each operator is processed on one *routing level*, which corresponds to the according level in the routing graph. A peer may represent multiple vertices of a routing graph because it may be contacted several times for processing different parts of q. As a consequence, the query routing graph is a topology overlaid on the topology of the overlay (ring, tree, etc.). The graph also differs from its query plan q as the processing of one operator may span multiple routing points and at one routing point multiple operators may be processed. The latter is true when multiple operators can be processed locally on one peer without routing.

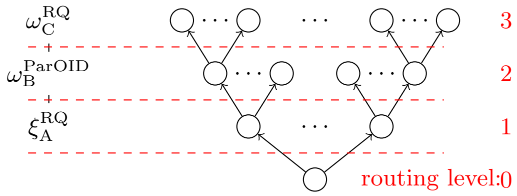

$\omega_{\mathrm{C}}^{\mathrm{RQ}}$
$\omega_{\mathrm{B}}^{\mathrm{ParOID}}$
$\xi_{\mathrm{A}}^{\mathrm{RQ}}$
3
2
1
routing level:0

Figure 2: Example query

Imagine the query plan in Figure 2. $\xi_{\mathrm{A}}^{\mathrm{RQ}}$ extracts all data items for attribute `A` using a range query to contact all peers responsible for a part of A. Thus, first routing method is DOR. $\omega_{\mathrm{B}}^{\mathrm{ParOID}}$ and $\omega_{\mathrm{C}}^{\mathrm{RQ}}$ materialize attributes B and C as additional information for each input data item. For any reasons, e.g., cost-based decisions, $\omega_{\mathrm{B}}^{\mathrm{ParOID}}$ uses direct lookups for candidate items' IDs and $\omega_{\mathrm{C}}^{\mathrm{RQ}}$ is range-based again. This means, second routing method is FNR, third is DOR. If the query would contain a select filter on A or B, this would be processed at the peers contacted for extracting/materializing A/B. Thus, the routing graph would look the same.

At time t the following knowledge is available at the query initiator: the ranges r_{A} and r_{C} of attributes A and C, subsets K_A and K_C of all paths from peers responsible for a part of r_{A} and r_{C}, a subset D_1 of all peers on level 1 with their fanout d, $|D_1| = |K_A|$. Naturally, we assume $t \geq t_0$, where t_0 is the time of the first reply. Before t_0, completeness is always 0. From r_{A} and K_A we can estimate a minimal number p_1 of peers on level 1, as shown in [9]. Thus, the following holds:

$$p_2 = \sum_{d \in D_1} d + (p_1 - |D_1|) \quad .$$

This corresponds to the minimal number of replies to be expected, because we assume an out-degree of 1 for each peer on level 1 we do have no exact knowledge of. In analogy to p_1, we estimate d_2 (the fanout of each routing point on level 2) from r_{C} and K_C. Finally, the number of expected replies at time t is

$$R = p_3 = d_2 \cdot p_2 \quad .$$

Here you can see why assuming static query planning eases completeness estimation and improves its quality: we can assume the fanout of d_2 for each peer from level 2, even for those we did not receive information from until now.

In general, the number of leafs can be described by a recursive formula:

$$p_0 = 1, \quad p_l = \sum_{k=1}^{p_{l-1}} d_{(l-1),k} \quad (l = 1, \ldots, L), \quad \text{Ŗ} = p_L$$

where p_l is the number of vertices at level l, and $d_{l,k}$ is the fanout of the kth vertex at level l.

Algorithm 1 Basic algorithm: new-reply()

```
update-overlay();
update-RG();
$r = r + 1$;
$R = 1$;
for all routings $\rho \in RG \rightarrow$ get-levels() do
    $R = R + \rho \rightarrow$ estimate($R$);
end for
return $\frac{r}{R}$;
```

Based on the notion of routing graphs, the basic idea of estimating query completeness works as shown in Algorithm 1. This procedure is called every time a new reply arrives at the initiator. After updating necessary information in the estimated overlay structure and the routing graph RG (one for each initiated query), the fraction of received results r and estimated final replies R is returned. Thus, a currently estimated completeness (on peer level) is determined in an online fashion. In contrast to, for instance, Seaweed, we do not estimate this completeness directly on data level. Note that also aggregation queries return multiple replies, as the computed aggregation values are refined in an online fashion.

5.3 Routing Trace

But why is it adequate to differ only between two routing methods, when there is such a wide variety of possible processing strategies and indexes to be used? The point is the amount of information we can extract from a single reply received. Into each reply, we integrate information about the way the sub-plan took when traveling through the network during processing. As a consequence, this *routing trace* allows to identify the one path from the query initiator to a leaf of the routing graph that corresponds to this reply. Intuitively, for estimating the number of replies a query results in, the quality of this estimation depends on the kind of information about each routing point we can extract. For instance, if we know the out-degree d of a routing point before starting that routing, we can include this information in each of the d resulting routing traces – and thus, we know this information at the query initiating side as soon as the first of the corresponding sub-plans is replied.

5.4 Overhead

The proposed method comes along with a very low overhead. In terms of messages, there is no overhead at all, because any information needed is included into query plan messages and reply messages. The bandwidth consumption resulting from applying completeness estimation is almost negligible. Query plans contain only small additional information, which is the routing trace and a sketch of the overlay structure as it is known at the query initiator. The size of the routing trace depends on the size of the routing graph (more exact, on the maximal routing level L). Each entry of the routing trace has a constant size. The overlay structure grows with the number of peers participating in the system. In UniStore, we have $|p|$B for each path p known to the current peer. Thus, the bandwidth overhead for each (sub-)plan generated at routing point rp is in bits:

$$B_{rp} = \sum_{\text{known paths } p} |p| + B_m \quad \in O(N) \quad ,$$

where B_m refers to the size of the MiMe sent from this routing point. Note that information of size B_m is always added to a query plan, regardless if MiMes are sent separately or not. Thus, bandwidth consumption for one query neglecting MiMes is

$$\sum_{rp \in RG} d(rp) \cdot B_{rp} \quad \in O(N \cdot N^{L-1}) = O(N^L) \quad .$$

To make the completeness estimation work, we only have to keep small information for each routing level. Further, we have to detect replicated replies, which can occur in replicating overlays, and rely on the DHT to provide estimation techniques for every DOR method used.

6. GUARANTEES

Completeness estimation as introduced up to this point is a nice tool, but it only gains really significance and importance if we provide guarantees, more generally, meaningful quality measures. This section deals with this important aspect. First, we have to consider which kind of measurements are conceivable and meaningful in order to weight the accuracy of the query completeness predicted. We distinguish between three different kinds of guarantees:

1. A general guarantee for the accuracy that says: "If completeness is predicted as $c\%$, this is true with a probability of $x\%$".

2. A guaranteed boundary: “If completeness is predicted as c%, this is above/below the actual completeness with a probability of x%”.
3. A confidence interval: “If completeness is predicted as c%, the actual completeness is between $(c-y)$% and $(c+z)$% (with a probability of x%)”.

Which guarantee should be preferred will depend on the specific application.

We always predict an expected number of replies which is guaranteed to be below the actual number of replies. This results in a predicted completeness $\frac{r}{R}$ which is guaranteed to be above the actual query completeness. This and the fact that iff completeness is predicted to be 100% the actual completeness is 100% can be proved straight-forward. The idea is to apply an induction over the number of routing levels in the routing graph. We omit the full details here. Note that this guarantee holds for peer and data level, because if all potential replies are received all result data is received as well.

Providing a completeness which is always above the actual one is more than many systems can provide. But, of course, we are interested in a general accuracy and in a guaranteed lower bound. We investigate three different estimation techniques: estimate *(i)* a *minimal* number of replies, as introduced before, *(ii)* an *average* number of replies, and *(iii)* a *maximal* number of replies. Unfortunately, the unpredictable nature of the underlying systems prevents from providing exact guarantees for the latter two methods. After briefly introducing them, we will discuss how to provide probabilistic guarantees instead.

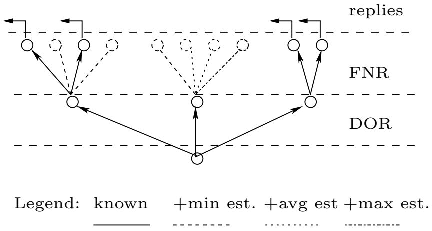
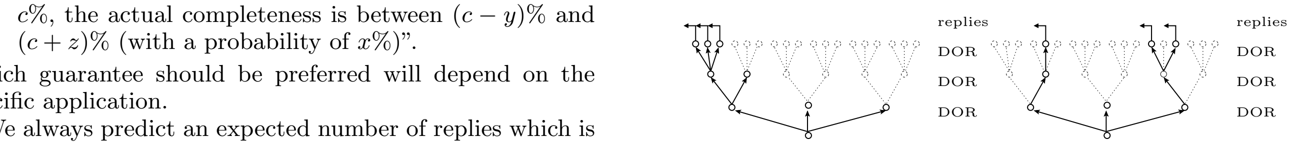


Figure 3: FNR example for min, max and average

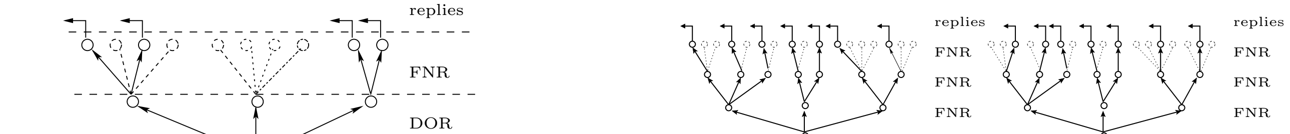
Both approaches are based on the already received information about the routing graph. The idea is best illustrated using the small example graph in Figure 3. The middle routing point from level 1 is known, but not its out-degree in the FNR level. From the other two subtrees, we already received 4 replies. By this, we know that there will be 2 more replies for sure (we know the out-degrees of both routing points) plus at least 1 from the “missing” subtree. Thus, with the minimal method we estimate 7 replies. As its name suggests, the average method calculates the average of out-degrees on the routing level in question, which is 3. Thus, 9 replies are estimated. Similar, the maximal method uses the maximum of out-degrees and estimates 10 replies. This methodology is applied for each routing point with an unknown out-degree. DOR estimations are based on a predicted structure of the overlay. As UniStore uses P-Grid, a corresponding tree structure is maintained at every peer. Here, average and maximal path lengths from that tree are used in order to predict the shape of the actual P-Grid tree. The actual number of replies may still be higher than predicted with the maximal method. In early states we also tried a method based on maximal values learned during time and from different setups. We soon observed that this method is always far below the maximal method introduced above – and as we will show in Section 7 this one itself is below the actual completeness in most cases and times. Thus, we focus on the three methods introduced so far.



(a) left-bound (b) random

Figure 4: Minimal routing trees for complete DOR

Next, we can relax the conditions needed in order to achieve a completeness estimation guaranteed by 100%. In fact, we already reach this point as soon as all needed information about the routing graph is achieved. For a better understanding, we introduce the notion of *minimal routing graphs.* For DOR, we need to know about the queried range (known when planning the query) and the exact number of peers in that range. This is achieved if one of all identical subqueries is completely answered, all the same from which part of the routing graph. This can also be true if no subtree is completely known at all, depending on the paths of replying peers. If all paths of a range are known, the exact number of responsible peers is known. In the FNR case, much more information is needed in order to achieve an exact completeness. The out-degree of each routing point must be known. Thus, at least one result from each routing point on level $L-1$ must be received. This results in complete knowledge of the routing graph below level L.



(a) left-bound (b) random

Figure 5: Minimal routing trees for complete FNR

Figures 4 and 5 illustrate this. All routing points and connections drawn solid are needed at minimum in order to achieve the 100% guarantee. We picture two out of many possibilities. In the first one the required information is located “as much left-oriented as possible”, the second one is random.

The notion of minimal routing graphs helps to understand the possible errors we make during completeness estimation. If any of the solid parts in Figures 4 and 5 is missing, we include such an error. The depth of the routing graph is known as long as we expect homogeneous subtrees, because at least one reply is already received. Thus, the only errors we can make are to estimate a wrong number of peers in a range when applying DOR, and estimating a wrong out-degree of a routing point when applying FNR. In order to provide a probabilistic guarantee, we have to weight these errors for each point in time where we estimate the query completeness c as $0 < c < 100$.

On each routing level, we assume the out-degrees of routing points, the path lengths of nodes in the overlay tree respectively, to follow a specific distribution model. In principle, any distribution model can be applied, even histograms for describing arbitrary distributions can be used. Problematic is the complexity of the following calculations. [13] gives suggestions and instructions for some of the most popular models. We refer to the Poisson distribution for out-degrees and path lengths, because it fits the actual distributions particularly good and calculations based on this model are very easy and efficient to implement.

In order to provide a lower bound of query completeness, below we describe a formula for determining the probability that the estimated completeness is below the actual. We expect all occurring random variables to be independent from each other. Let N_l denote the random variable that estimates the number of routing points on routing level l, and n_l the actual number. Looking only on the maximal routing level, the probability $P(N_L = n_L)$ under the condition that n_{L-1} routing points exist on level $L-1$ is

$$P(N_L = n_L | N_{L-1} = n_{L-1}) = \sum_{S_{L-1}} \prod_{i=1}^{n_{L-1}} P(D_i = d_i) \quad ,$$

where S_{L-1} is the set of all combinations $\{d_1, \ldots, d_{n_{L-1}}\}$ such that $d_1 + \cdots + d_{n_{L-1}} = n_L$. D_i describes the out-degree of an arbitrary routing point on level $L-1$. This formula computes the convolution of the n_{L-1} random variables. As mentioned above, using Poisson distributions

$$P(D_i = d) = \frac{\lambda_i^d}{d!} e^{-\lambda_i}$$

we can easily calculate this convolution using

$$P(D = d) = \frac{(\sum_i \lambda_i)^d}{d!} e^{-(\sum_i \lambda_i)}$$

Known out-degrees are simply excluded from this summation. This works directly for distributions describing the observed out-degrees on FNR routing levels. A distribution describing the path lengths in the overlay must be mapped to a distribution of out-degrees accordingly. Currently, we do this using a repository of trees observed over time, but are investigating other approaches. Using the cumulative distribution, the guarantee is determined by

$$P(\text{est. compl.} \leq \text{actual compl.}) = P(\text{\c{R}} \leq R) = P(D \leq R) \quad .$$

Including lower levels results in a recursive formula. Let K_l refer to the set of out-degrees already known on level l, k_l to the sum of all out-degrees in K_l. Then,

$$\begin{aligned} &P(N_L \leq n_L) = \\ &\sum_{n_{L-1}=k_{L-1}}^{u_{L-1}} P(N_L \leq n_L | N_{L-1} = n_{L-1}) \\ &\cdot P(N_{L-1} = n_{L-1}) \quad . \end{aligned}$$

if we use $u_{L-1} := k_{L-1} + n_L - k_L - (k_{L-1} - |K_L|)$ for the maximal number of nodes that is possible on level $L-1$ (respecting known out-degrees and the estimated number of nodes on level L). This gets

$$\begin{aligned} &P(N_L \leq n_L) = \\ &\sum_{n_{L-1}=k_{L-1}}^{u_{L-1}} P(N_L \leq n_L | N_{L-1} = n_{L-1}) \cdot \\ &\sum_{n_{L-2}=k_{L-2}}^{u_{L-2}} P(N_{L-1} = n_{L-1} | N_{L-2} = n_{L-2}) \cdot \cdots \cdot \\ &\sum_{n_1=k_l}^{u_1} P(N_2 = n_2 | N_1 = n_1) \cdot P(N_1 = n_1) \quad . \end{aligned}$$

If levels of the routing graph are completely known, corresponding probabilities resolve to either 1.0 or 0.0 and these parts could be extracted from the calculation. This is what we did with level 0 in the above formula. If individual parts of one level are unknown, this requires computing the convolution in the according routing level. With Poisson distributions and the information gathered for completeness estimation this probabilistic guarantee can be calculated with ease. An according line $P = P(\text{\c{R}} \leq R)$ is added between line 7 and 8 in Algorithm 1, and the determined probability P is returned together with the estimated completeness. In early query states, the significance of the calculations will be very poor. With proceeding time this improves quickly. Evaluating different distribution models and the significance of applying them is part of our ongoing work.

As an optimization technique, we introduce *Milestone Messages* (MiMes). The idea is to send information from each routing point in a separate message directly back to the initiator while forwarded sub-plans are still processed in the network. They only include information that is available in the routing trace of a reply as well. But, due to slow or failing peers, they help to improve the accuracy of the estimation in early states. MiMes are sent irregularly when a certain part of the query plan is processed. They are not essential for applying the proposed completeness estimation, but increase estimation quality represented by better probabilistic guarantees. The number of generated MiMes increases with the number of peers involved in processing a single plan. Thus, it increases with the number of peers in the system. The bandwidth overhead can be regarded as a constant factor. Due to space restrictions, we omit further details here.

7. EVALUATION

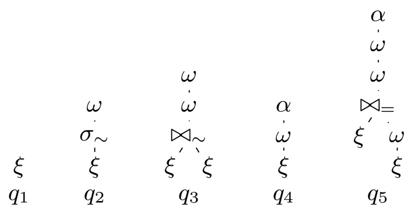
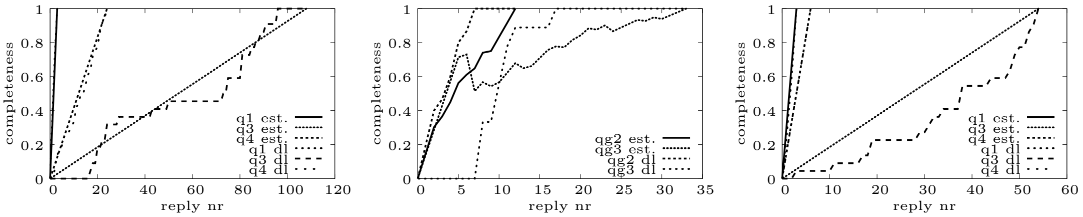


Figure 6: Shape of evaluation queries

Setup We evaluated the proposed approach running an extended set of experiments on a local environment running up to 74 independent instances. Further, we repeated these tests on Planet-Lab [4], a world-wide consortium created especially for running large-scale distributed experiments. Planet-Lab is specifically dynamic and unreliable. At the time of our experiments, we could allocate only very few resources, so that we could only run a couple of our queries in a meaningful sense. In each run we built a P-Grid network with standard parameters from scratch. We based our experiments on a mixture of triple data from DBPedia[1], geographical data in relational format from Mondial[2], and a small set of ontology data. This data is taken from a realistic scenario combining geographical data from both sets. After a certain waiting time for establishing a suitable overlay trie, we initiated the queries described below. Applying UniStore's vertical data scheme and building two different indexes, this resulted in a total of about 16,000 index entries. The set of all generated keys shows a skewed heavy-tail distribution (power-law like). We ran the tests on networks of

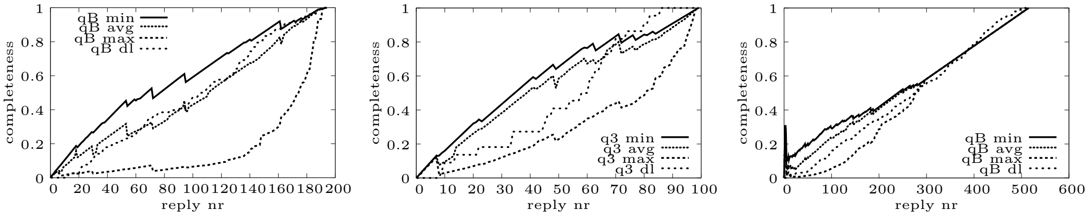
[1] http://dbepdia.org

[2] http://www.dbis.informatik.uni-goettingen.de/Mondial/



(a) Only DOR routings (b) Only FNR routings (c) DOR with sequential parts

Figure 7: Accuracy for separated routing methods



(a) q_B (b) q_3 (c) q_B on 50 peers

Figure 8: Minimal, maximal and average estimation

size 25, 50 and 74. In Planet-Lab we were able to include about 400 nodes in each run.

Queries The used query mix was chosen in order to be representative for both routing methods introduced and to span from simple to rather complex queries. We divided them into classes of queries, which we refer using $q1, q2, ..., qA, qB$ etc. Figure 6 shows the general shapes of 5 of these classes. They were issued using different access paths. In addition, we ran two complex queries q_A and q_B involving ontology data and combining it with data from DBPedia and Mondial. qg_1, qg_2 and qg_3 represent queries containing only FNR routings.

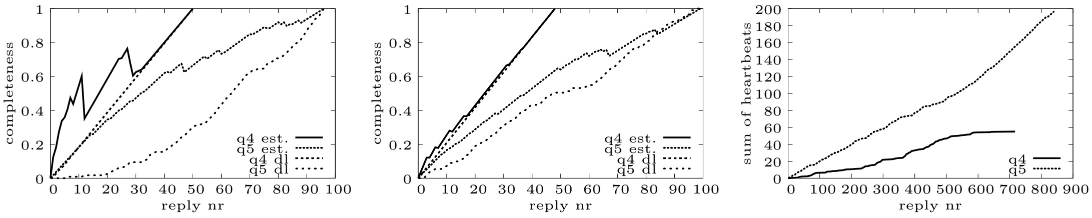
The aim of the evaluation is to show that the proposed method works correctly. Further, we will highlight the benefits of using MiMes for achieving a higher accuracy in early query states and compare the three different estimation techniques min, max and avg. From all experiments, we selected the results that are particularly suited for this. In all of the following figures we plot the number of received replies on the x axis. Thus, actual completeness for each query is the straight line from $(0,)$ to $(\c{R}, \c{R})$. We omit this line in order to improve readability. Rather, we plot completeness on data level (denoted by *dl*) to additionally show the correlation between both.

We begin with evaluating the accuracy of completeness estimation separately for the supported routing methods. As already shown in [9], we expect DOR routings to be estimated very accurate. A more challenging task is to satisfyingly estimates FNR routings, because more information from the routing graph is needed for that. Figure 7 shows representative results by plotting the minimal CE calculated with each received reply. As expected, DOR routings (7(a)) are estimated very accurately with first replies. This is due to the fact that sufficient information about the overlay structure can be collected by one sub-plan traveling through the network. Depending on the number of FNR parts, estimation gets inaccurate in early query states. If only FNR is applied (7(b)), there is a clear gap between estimated and actual query completeness. This gap can also be observed with less FNR parts, e.g., when integrating sequential operators (7(c)). All three figures show that completeness on data level is satisfyingly good approximated by peer level, albeit the drift between both increases with rising query complexity.

Next, we want to investigate the impact of the introduced average and maximal estimation methods. As DOR routings seem to be a by far lower challenge than FNR routings, we focus on queries containing at least one FNR routing. Figure 8 shows corresponding results for selected queries. In 8(a)-8(c)we show that there can be significant differences between all three techniques. As expected, the minimal number estimation is always above the actual completeness, whereas the maximal one is mostly located below. To our pleasure, completeness on data level is mostly very well approximated by both, minimal and average completeness. The maximal method is rather pessimistic, resulting in a predicted completeness far below the actual one – but this also means, it provides what it was invented for: a guaranteed lower bound. The observed probabilistic guarantee approaches 90% for the maximal estimation rather quickly, whereas the average one usually balances between 50%-80%. As expected, the proposed guarantee is around 2%-5% for the minimal estimation technique.

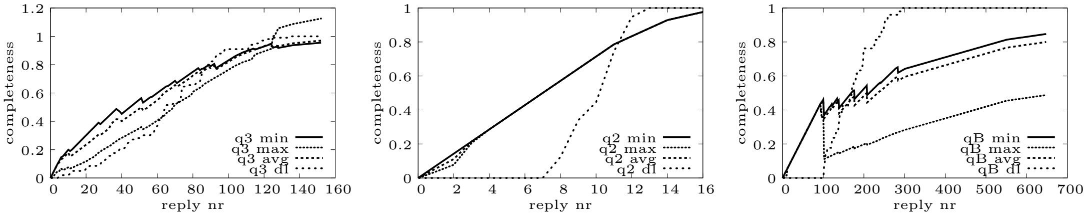
All the results presented up to here were gathered using MiMe support. We believe in the small overhead worth for achieving a higher quality of estimation. This is approved by the results shown in Figure 9. We chose to picture a query with high degree of parallelism, because MiMes become particularly advantageous in this case. The differences between 9(a) and 9(b) reveal that without heartbeats corrections to the estimated value are bigger and occur more often. The overhead paid for this increases linearly with the number of received replies (9(c).

Finally, we show that scalability in terms of network size is really no issue for the achieved accuracy. In Figure 10 we picture the completeness plots for three selected queries. All three estimation techniques behave analog to the local setup. As seen before, completeness on peer level is estimated very exactly. This matches the data level very well in most cases, but obviously not in all situations. An irritating point is that our maximal completeness estimation reaches a value higher than 1.0, which should not happen by implementation. This can only be due to high inconsistencies in the overlay structure, caused by the painfully slow nature of Planet-Lab. Currently, we are investigating this issue in detail.



(a) High parallelism, no MiMes (b) High parallelism, MiMes (c) Afforded overhead

Figure 9: Impact of MiMes



(a) q_3 (b) q_2 (c) q_B

Figure 10: Queries on Planet-Lab

Summarizing, we were able to show the accuracy of the proposed method for estimating query completeness. By introducing three different techniques, we are also able to determine guaranteed lower and upper bounds. The resulting curves show either a straight line or a lightning-like shape, getting the more escalating the higher the degree of parallelism (and data-dependence) gets. The curves produced by Seaweed show a Z-shape instead. This is due to the fact that Seaweed aims at providing delay-aware querying while focusing on aggregate queries. Downtime of unavailable peers is predicted and results are gathered when this time passes. In contrast, UniStore follows a best-effort approach, which is also reflected in the dynamic and online character of its completeness estimation.

8. CONCLUSION

In this paper, we have proposed an approach for addressing one of the main problems in P2P database systems which is completeness estimation for query answering. We have defined completeness in the context of structured P2P systems and motivated its need for both user and system requirements. For completeness estimation, we distinguished two basic classes of routing methods underlying the query processing strategies. Based on this, we have described the estimation of completeness by observing the progress of query execution at peer (routing) level. Furthermore, we have discussed the accuracy of the estimations in terms of probabilistic guarantees and introduced milestone messages for tracking the query progress. We have implemented the overall approach as part of our P-Grid-based UniStore system where it is exploited both for giving feedback to the user as well as for supporting blocking operators such as aggregations or skyline operators efficiently. The results of our large-scale experimental evaluation show the suitability of the approach as well as the validity of the estimations.

9. REFERENCES

[1] K. Aberer, M. Hauswirth, M. Punceva, and R. Schmidt. Improving Data Access in P2P Systems. *IEEE Internet Computing*, 6(1), 2002.

[2] R. Bhagwan, D. Moore, S. Savage, and G. M. Voelker. Replication Strategies for Highly Available Peer-to-Peer Storage. In *Future directions in Distributed Computing*, pages 153–158, 2002.

[3] M. Cai and M. Frank. RDFPeers: a scalable distributed RDF repository based on a structured peer-to-peer network. In *WWW*, pages 650–657, 2004.

[4] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. PlanetLab: An Overlay Testbed for Broad-Coverage Services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12, 2003.

[5] S. D. Gribble, A. Y. Halevy, Z. G. Ives, M. Rodrig, and D. Suciu. What Can Database Do for Peer-to-Peer? In *WebDB 2001*, pages 31–36, 2001.

[6] R. Huebsch, J. M. Hellerstein, N. Lanham, B. Thau Loo, S. Shenker, and I. Stoica. Querying the Internet with PIER. In *VLDB*, pages 321–332, 2003.

[7] M. Karnstedt, K. Sattler, M. Haß, M. Hauswirth, B. Sapkota, and R. Schmidt. Estimating the Number of Answers with Guarantees for Structured Queries in P2P Databases (Poster). In *CIKM*, 2008. to appear.

[8] M. Karnstedt, K. Sattler, M. Richtarsky, J. Müller, M. Hauswirth, R. Schmidt, and R. John. UniStore: Querying a DHT-based Universal Storage. In *ICDE'07, Demonstr. Program*, pages 1503–1504, 2007.

[9] M. Karnstedt, K. Sattler, and R. Schmidt. Completeness Estimation of Range Queries in Structured Overlays. In *P2P*, pages 71–78, 2007.

[10] A. Motro. Completeness Information and Its Application to Query Processing. In *VLDB*, pages 170–178, 1986.

[11] D. Narayanan, A. Donnelly, R. Mortier, and A. Rowstron. Delay aware querying with seaweed. In *VLDB'06*, pages 727–738, 2006.

[12] F. Naumann, J.-C. Freytag, and U. Leser. Completeness of integrated information sources. *Information Systems*, 29(7):583–615, 2004.

[13] M. Theobald, G. Weikum, and R. Schenkel. Top-k query evaluation with probabilistic guarantees. In *VLDB'04*, pages 648–659, 2004.