University of Freiburg
Department of Computer Science
Computer Networks and Telematics

SS 2009

Seminar Paper

F. Heine et. al.: Processing Complex RDF Queries over p2p Networks

Johannes Schwenk

July 24, 2009

Prof. Dr. Christian Schindelhauer

Contents

1 Introduction and Overview **2**
1.1 Goal & Motivation . . . 3
1.2 RDF/S . . . 3
1.3 Distributed Hash Tables based Peer-to-Peer Networks . . . 3

2 Presented Approach **4**
2.1 Scenario and Node Architecture . . . 4
2.2 Query Evaluation . . . 6
2.2.1 Determination of Candidate Sets . . . 7
2.2.2 Refinement . . . 8
2.2.3 Final Evaluation . . . 8
2.2.4 Top k Rated Queries . . . 9
2.3 Results . . . 10

3 Notes **12**

List of Figures **16**

1 Introduction and Overview

This seminar paper covers the papers "Processing complex RDF queries over P2P networks"[**?**], "Scalable P2P based RDF Querying"[**?**] and "Top k RDF Query Evaluation in Structured P2P Networks"[**?**].

The first two papers mainly cover the same topic, where the second paper is a restatement of the first one with optimizations and some corrections. In the third paper further optimizations are introduced.

In this section I will give an overview on queries in peer-to-peer networks (p2p networks) and summarize the motivation of the authors. Further more I will give a short introduction on the topics of the Resource Description Framework (RDF) and RDF Schema (RDFS), Distributet Hash Tables (DHTs) and p2p networks. The second section will explain the architecture and the algorithm proposed by the authors in detail. The last section will discuss advantages and disadvantages of the approach.

1.1 Goal & Motivation

Todays internet contains information in abundance, most of which is only meaningful to humans. For automated agents, it is almost impossible to process the information, let alone to infer facts on its basis. To give software agents the possibility to navigate and act autonomously within the internet, it should contain machine readable meta data about the resources it provides.

As Tim Berners Lee envisioned, automated agents could some day organize a persons timetable, booking flights, making appointments and finding experts on a certain task. Given the growing connectivity of devices of all sorts, an easy and intelligent way of communication and interaction could make life easier.

The Resource Description Framework and RDF Schema (RDF/S) is are an established way of storing meta data linked to Uniform Resource Identifiers (URIs) or Extensible Markup Language (XML) literals. RDF/S is an official recommandation of the The World Wide Web Consortium (W3C). Still missing, is an efficient way of processing this meta data and to reason about it. The main problem is to keep the balance between network overhead and the size of the result sets. The authors therefore propose a distributed RDF storage based on a DHT based p2p network and develop an algorithm that reduces network traffic for queries, while still maintining good scalability.

1.2 RDF/S

Resource Description Framework and RDF Schema are key components in what is the W3C's proposal of a Semantic Web. The RDF is a family of W3C specifications that describe resources idententified by URI as triples of subject, predicate and object. A collection of such statements represents a labeled, directed multi-graph. RDFS is a language that can be used to build a RDF vocabulary also known as an *ontology* which allows to reason about the properties of a given domain.

1.3 Distributed Hash Tables based Peer-to-Peer Networks

Distributet Hash Tables consist of a collection of seperate connected storages that allow to retrieve and store pieces of information by key via a hash function. Peer-to-peer networks are decentralized networks, that distribute work loads, computational tasks and or store data between their peers. A DHT based p2p network stores data in a distributed fashion and allows for retrievel of stored information by quering the network via the DHT's hash function. Every node in the network is assigned a subset of the

networks available information in that way. Ideally all nodes should have equally many parts of information stored, which in reality can be difficult to accomplish, since there might be to few information and thus not all nodes get to store anything. On the other hand, the hash function could be suboptiomal in the sense of equal distribution.

2 Presented Approach

The presented approach describes a p2p network which stores all RDF/S knwoledge in a DHT realized using FreePastry[1]. I will give a short overview on the scenario and then elaborate how queries are completed, which is accomplished in two phases: by first collecting candidate sets and then local evaluation of the query. Refinement reduces the candidate sets efficiently. *Lookahead strategies*, *caching* and *Bloom filters* reduce the network owerhead. Finally the authors show the possibility to increase scalability further with *Top k rated queries*.

2.1 Scenario and Node Architecture

The authors assume to have n nodes participating in a p2p network. Every node in the p2p network has a so called *local knowledge* (LK) or local storage, which stores all RDF triples locally availabe to the node plus those which have been the result of inference. The RDF-Schema knowledge of every node is kept in a *schema knowledge* (SK) or schema storage. This schema knowledge does not have to be the same for every node in the network, though it is assumed that there is a small ontology that is available to at least a small subset of nodes from the beginning, serving as common starting ground for queries and *inference*. Figure 1 shows an overview of the architecture of this scenario.

A query is a directed RDF graph where parts of the labels and URI references are replaced by variables, this is also known as the *query graph*. The union of all local knowledge across the p2p network plus all triples that are results of the appliance of RDFS entailment rules[2] is called the directed *model graph*.

The implementation supports a subset of two RDFS entailment rules, rdfs7 and rdfs9 for inference. The first one defines inheritance of properties, the second one defines

[1] http://www.freepastry.org/

[2] RDF Semantics W3C Recommendation 10 February 2004 - http://www.w3.org/TR/2004/REC-rdf-mt-20040210/

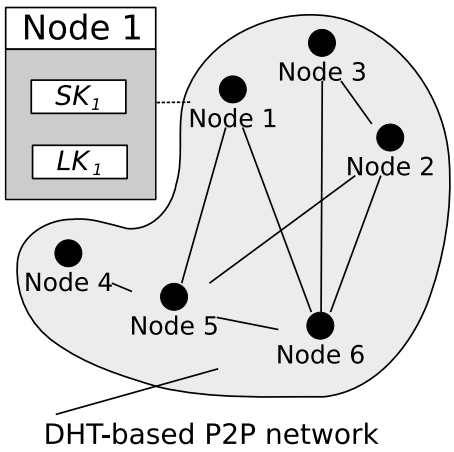

Figure 1: Architecture (adapted from [**?**])

subclassing. Rdfs5 and rdfs11, which introduce the predicates subClassOf and subPropertyOf, are supported implicitly.

The following definitions are beeing made:

- L set of labels
- B set of blank node labels
- V set of variables

The model graph can be defined as a set of triples as follows

$$T_M \subseteq (L \cup B) \times L \times (L \cup B) \tag{1}$$

The query graph then is defined as

$$T_Q \subseteq (L \cup V) \times (L \cup V) \times (L \cup V) \tag{2}$$

It can be assumed that blank node labels are unique. To ensure that, every blank label can be thought as having received a node identifier, thus making it unique in the model graph. Further more the model graph is assumed WLOG to be connected. The query graph has at least one triple with at least one labelled element, serving as a starting point for the evaluation.

Figure 2 gives an overview on the node architecture. Note that all knowledge that is obtained via the p2p network or is a result of inference, is stored in a container seperate from the knowledge that may be supplied locally to the client - e.g. as RDF/S files.

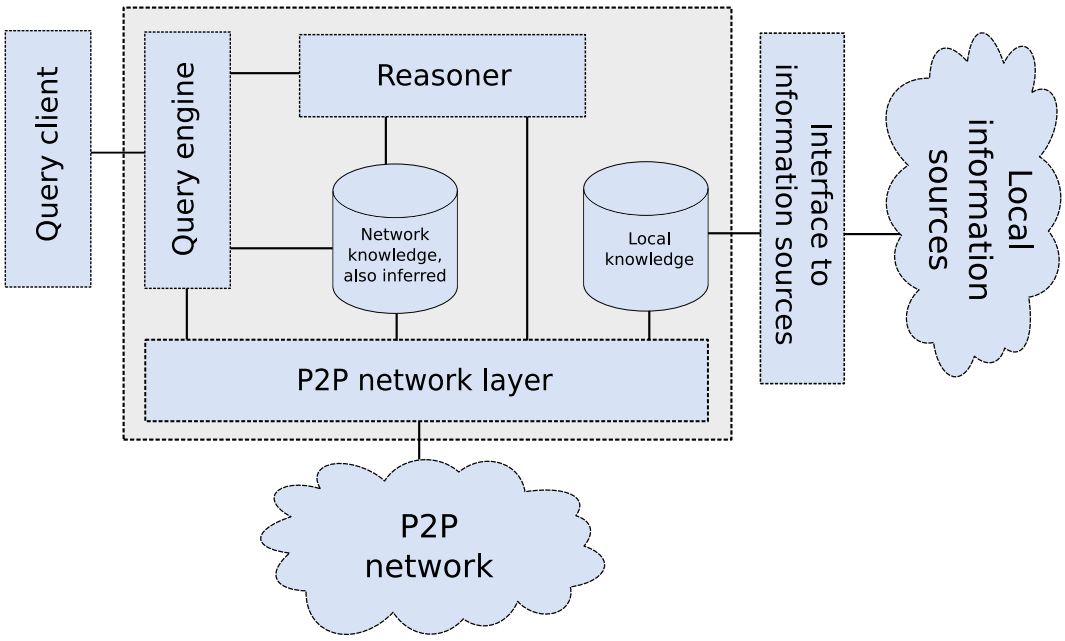

Figure 2: Node architecture (adapted from [**?**])

Reasoning takes place in respect to the network and inferred knowledge only. Triples generated during evaluation are also sent to their responsible nodes via the DHT.

The whole model graph is stored in the p2p network using the DHT. Every triple gets stored three times, once each by subject, object and predicate. To retrieve a set of triples at least one part has to be known. The retrievel can be done by using the three functions getBySubject, getByPredicate and getByObject. Additionally there are the three statistical functions cntBySubject, cntByPredicate and cntByObject, which only return the count of the result set instead of the result set itself.

2.2 Query Evaluation

Query evaluation takes place in two phases: *determination of the candidate sets* and the *evaluation phase* where the retrieved candidates are matched against the query graph. In the first phase a refinement procedure takes care of reducing candidate sets and backtracking reduces the number of combinations to test for in the evaluation phase.

2.2.1 Determination of Candidate Sets

This phase determines those parts of the model graph, which are relevant for the query, it's doing this while avoiding unnecessary network traffic. Different strategies are presented to achieve this goal.

An important notion in respect to reducing network overhead is the *specification grade (sg)* of a triple, which determines the number of lookup operations needed to retrieve its candidate set. The specification grade of a literal is defined as beeing 1 and the sg of a variable as beeing the current number of candidates for this variable. The sg of a triple then is defined as the minimum of the sg of its elements:

$$sg1(\langle s, p, o\rangle) = min(sg1(s), sg1(p), sg1(o)) \tag{3}$$

To further reduce the network load, there are some *enhancements* to the sg function:

The first enhancement is a lookahead for the sg function. The definition for the sg function applied to a variable, changes to beeing defined as the return value of the cntBy[...] functions defined above. This results in further lookups during the determination of the candidate sets, but reduces the overall message size. This is leading to a new definition of the sg function:

$$sg2(\langle s, p, o\rangle) = min(sgs(s), sgp(p), sgo(o)) \tag{4}$$

Where sgs, sgp and sgo are returning the value of the cntBy[...] functions.

A third definition of the sg function ($sg3$) is beeing introduced by the second paper [**?**]. It has caching included, so multiple calls of $sg3$ don't result in identical lookups. The caching reduces the number of lookups dramatically.

The last enhancement of the sg function is the addition of *Bloom filters* also introduced in the second paper [**?**]. If a lookup for the triple $\langle x, v_2, v_3\rangle$ is performed for x via $getBySubject(x)$ or $cntBySubject(x)$ the Bloom filters for the candidate sets of v_2 and v_3 are sent also. The queried node will then send only those triples back, that are detected via the Bloom filter as beeing present in the already known candidate sets of v_2 and v_3 plus as small number of *false positives* resulting from the probablistic nature of Bloom filters. The modified versions of $getBy[...]$ functions can be combined with all

former definitions of the sg function. The modified versions of the $cntBy[\ldots]$ functions result in a new definition of sg called $sg4$. The caching function of $sg3$ is also used for $sg4$. This however means, that at a later time better estimates are beeing lost, when candidate sets for more variables are available.

As Bloom filters can be applied to all the previous definitions of sg functions, there are in total eight versions of the sg function and the suffix $/b$ denotes the versions with Bloom filters.

The determination of candidate sets is done in a loop over all triples with undefined candidate set. In each iteration the triple with the smallest sg is choosen and the candidates retrieved from the network. In the case of $sg3, sg4$ caching takes effect if the the sg of the current triple has been calculated before. The cache is valid for one evaluation. The $/b$ versions of the sg functions use modified $getBy[\ldots]$ functions which take optionally the Bloom filters of known candidate sets of the other variables in the triple as up to two optional parameters. The queried node(s) then send back only those triples which are a match in the passed Bloom filter(s).

2.2.2 Refinement

The refinement process is a rather important part of the presented algorithm. It runs completely local and therefore does not increase the network load. There are two ways of refinement:

The variables candidate set is compared to the candidate sets of each triple where this variable occurs. If a variable candidate is not ocurring in the triple candidate set it will be removed.

The other way around any candidiates for a triple which have some value not within the mathing variable's candidates will be removed.

If after those two steps one or more empty candidate sets in either the candidate set of the triple t or the candidate set of variable v occur, the refinement function returns an error. Otherwise the refinement procedure restarts from the top until in the last iteration no more variables get changed and no triples are left to process.

2.2.3 Final Evaluation

Candidate sets for all triples and variables have been retrieved and are now locally available to the quering node. Because the evaluation could result in doing exponentionally

many tests, since every combination of triples has to be tested, the authors employ a backtracking algorithm. For every triple in the query graph check if the variable assignments contradict previous assignments. If not, store the new assignments in a map and recursively call the *evaluate* function with this new map and the query graph (without the previously testet triple). When the algorithm halts, the variable assignments in the map are a match.

2.2.4 Top *k* Rated Queries

In the last paper [**?**] the authors assume that in most cases it suffices to deliver only the Top k results of a query according to a sorting attribute, since the quering user might not be interestet in an exhaustive search, but in a limited number of best results. To cope with large candidate sets, the nodes are instructed to deliver them *in order*, so the *next best* rated triple can be chosen to request more candidates, in case the last request did not deliver enough candidates to get k matches.

To reduce the amount of information sent over the network, the authors employ lookahead caches to be prepared in case of backtracking beeing ncessary. For each lookup by one element of a triple, there remain two components which can either be bound or unbound variables, or a fixed URI literal. Thus there are six different kinds of caches to be constructed:

- fixed/fixed
- fixed/bound
- bound/bound
- fixed/unbound
- bound/unbound
- unbound/unbound

All retrieval operations for triples are managed by the *varoius caches*. In case of a cache *hit*, the cache can confirm the existance of the requested triple. In case of a *miss*, the cache will retrieve the *needed information* from the network. Also the cache will *retrieve additional candidates*, in anticipation of them beeing needed later on. Since the queried peer for scalability reasons does not store the state of the query between requests, the cache has to send this information along with its request.

The algorithm evaluates the query on the fly as more and more triples are getting requested. In each recursive step one candidate (t) after the other will be retreived from the cache (resulting in more triples getting cached as a lookahead). The retrieval is

Query	Triples	Variables	Matches
1	11	8	1417
2	11	11	1417
3	14	10	9879
4	14	13	9879

Table 1: Test data (from [**?**])

Query	time	Sesame time
1	< 1 sec	5 sec
2	< 1 sec	7 sec
3	2 sec	47 sec
4	3 sec	59 sec

Table 2: Local query evaluation time (from [**?**])

done with respect to the already known variable bindings. The variable bindings and the candidate set of t are updated and the evaluation function is called recursively.

The function terminates when: the number of matches exceeds the limit k, there are no more candidates, a complete match has been found or contradictions have been found.

Besides returning the number of found matches, the function returns the found variable bindings in a set of tuples $(variable, value)$.

2.3 Results

The authors claim, that their tests with simulated nodes confirm a performance 20 times as high as a comparable setup, based on the Sesame[3] toolkit. For this they compared only the local evaluation phase using an in-memoy representation. The model graph consisted of roughly 350.000 triples on which several queries have been evaluated (see table 1). For comparing results see table 2.

Forwarding a message through the network has a complexity of $O(logn)$ where n is the number of nodes. Also large result sets are not desirable as they lead to congestion and slow speeds. Figure 3 shows the number of lookups needed to resolve ten different queries on two different model graphs. The queries are sorted by variable count from left to right. Here the *sg* functions version 1-3 are compared. Figure 4 shows the total

[3] `http://www.open-rdf.org` - version 1.1.3 was used by the authors

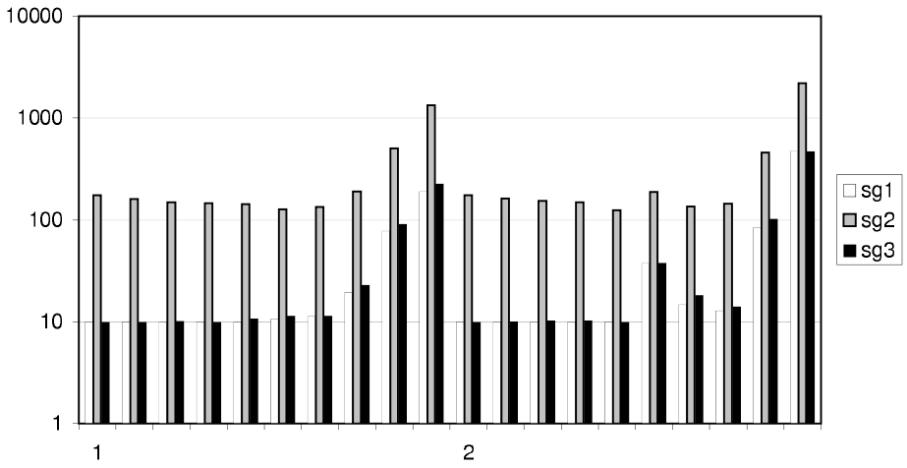

Figure 3: Number of lookup operations (from [**?**])

number of triples transfered during each query evaluation. Version *sg*3 has an optimal tradeoff between number of lookups and message size.

For comparison of the eight different *sg* versions of the **second paper** a RDF base of around 60.000 triples and 100 queries were used. The results are shown in table 3.

To examine the effect of the measured values for each version of the *sg* function on the overall system performance, the authors deployed a p2p network to their cluster system ARMENIUS with varying node count from one to 64 nodes in stepsizes of a power of two. After the distribution phase, each node in the network sent out constantly the 100

Variant	Message Size	Lookups	Filters
sg1	461062	1043	0
sg1/b	71138	1043	1144
sg2	20716	12395	0
sg2/b	16700	12395	1065
sg3	11324	3003	0
sg3/b	7308	3003	1065
sg4	98373	2450	1643
sg4/b	6602	2447	2702

Table 3: Comparison of the *sg* versions (from [**?**])

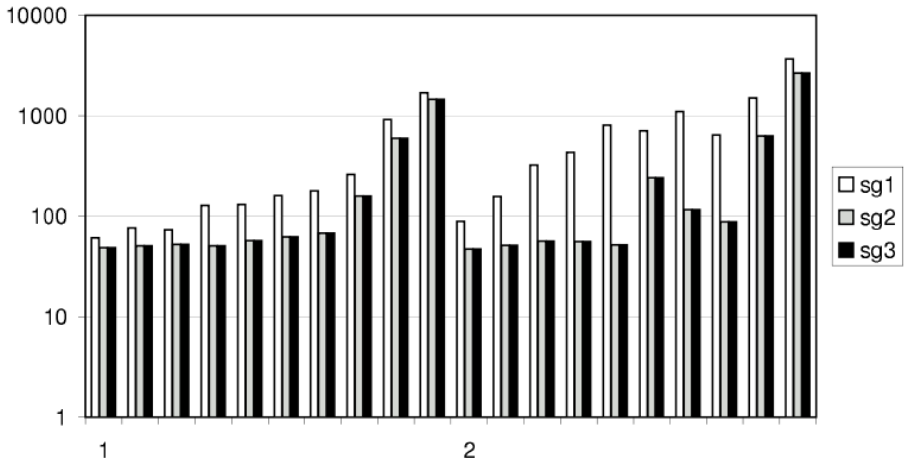

Figure 4: Message size (from [**?**])

different test queries. The measured results can be seen in figure 5. The results show $sg3/b$ as having the advantage over the other versions.

Using Top k rated queries further increases the scalability as can be seen in figures 6 and 7.

3 Notes

Acronyms

DHT Distributet Hash Table. 2, 3, 6

p2p network peer-to-peer network. 2, 3

RDF Resource Description Framework. 2–4

RDF/S Resource Description Framework and RDF Schema. 3

RDFS RDF Schema. 2

sg specification grade. 7

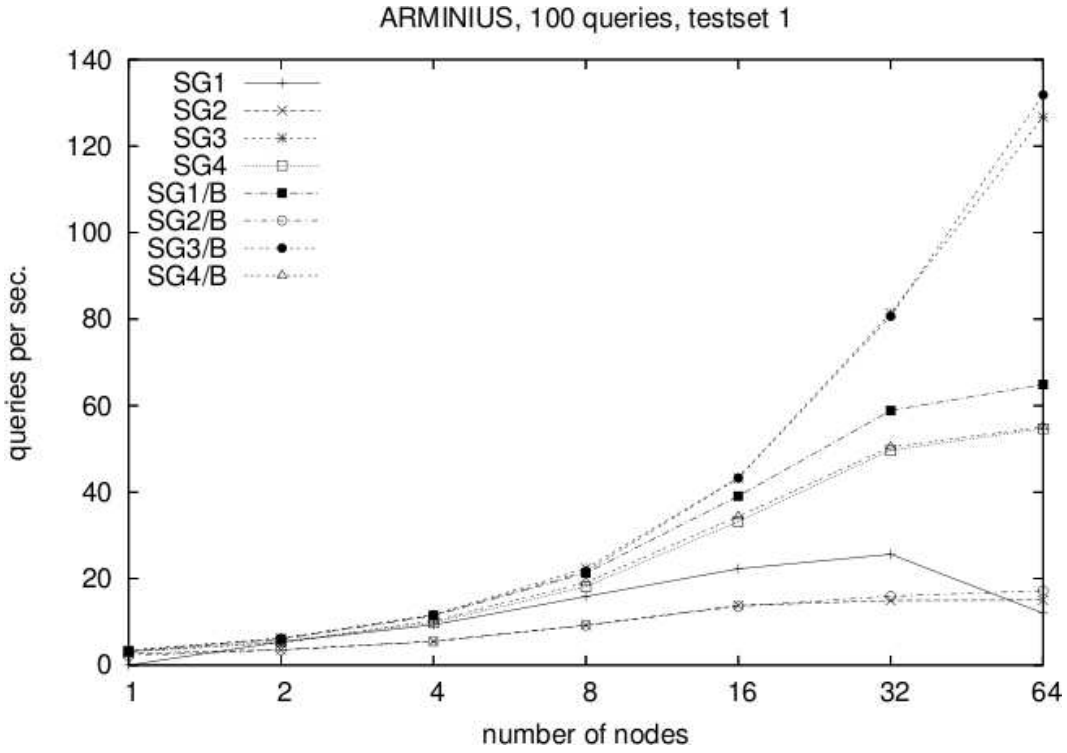

Figure 5: Results for prototype (from [**?**])

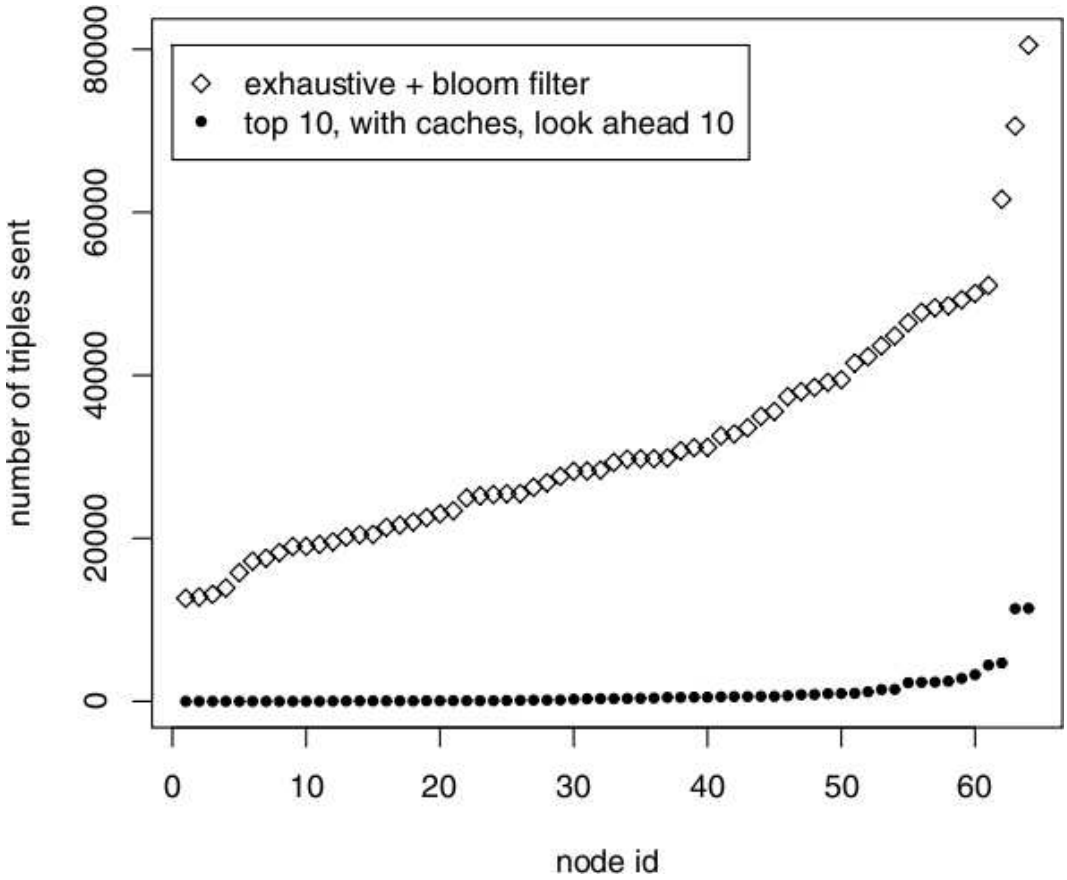

Figure 6: Empirical analysis: Total number of triples sent by peers to process exhaustive and Top 10 search (from [**?**])

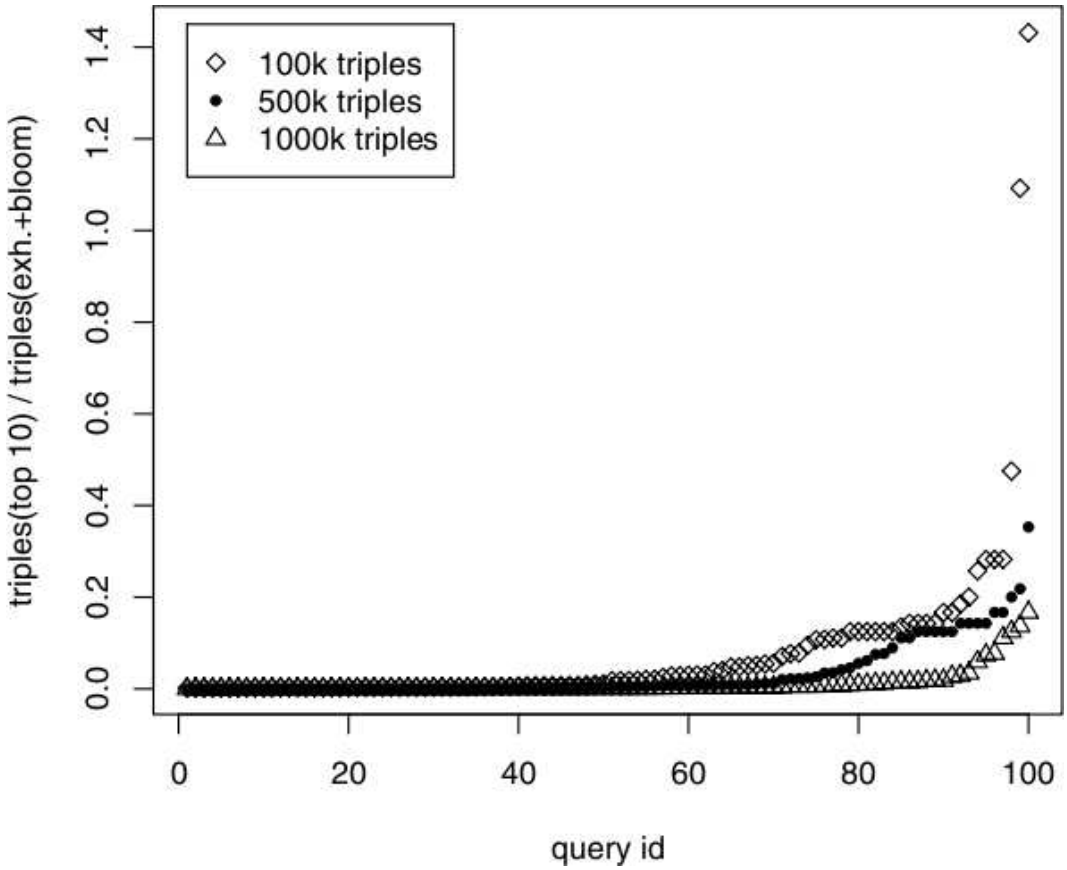

Figure 7: Empirical analysis: Ratio of triples sent by peers for Top 10 search divided by triples sent for exhaustive search (from [**?**])

URI Uniform Resource Identifier. 3, 9

W3C The World Wide Web Consortium. 3

XML Extensible Markup Language. 3

List of Figures

1 Architecture (adapted from [?]) 5
2 Node architecture (adapted from [?]) 6
3 Number of lookup operations (from [?]) 11
4 Message size (from [?]) 12
5 Results for prototype (from [?]) 13
6 Empirical analysis: Total number of triples sent by peers to process exhaustive and Top 10 search (from [?]) 14
7 Empirical analysis: Ratio of triples sent by peers for Top 10 search divided by triples sent for exhaustive search (from [?]) 15