

Toward widely deployable Semantic Web P2P: tools, definitions and the RDFGrowth algorithm

Giovanni Tummarello¹, Christian Morbidoni¹, Joakim Petersson², Francesco Piazza¹, Mauro Mazzieri¹, Paolo Puliti¹

¹Dipartimento di Elettronica, Intelligenza Artificiale e Telecomunicazioni
Università Politecnica delle Marche,
Via Brezze Bianche – 60131 Ancona (ITALY)
{g.tummarello, c.morbidoni, upf,
p.puliti}@deit.univpm.it
<http://semanticweb.deit.univpm.it>

² KTH, Royal Institute of Technology,
Stockholm (SWEDEN)
jocke@kth.se
<http://www.kth.se>

Abstract. There are a number of reasons why user scalability is inherently difficult for Semantic Web P2P scenarios. Approaches distributing queries or data can easily allow any user to consume arbitrary amounts of common resources in terms of computational burden and network traffic respectively. In this paper we illustrate these shortcomings and introduce RDFGrowth, an incremental RDF annotation exchange model and algorithm in an effort to overcome them. RDFGrowth is based on growing the local peer database using only basic direct P2P queries. As these require only minimal and bounded remote computational resources and as information is highly replicated, the algorithm opens the way to real world “napster like” P2P SW applications. In describing RDFGrowth, we furthermore provide formal definitions of general applicability when RDF sub-graphs are exchanged or signed.

1. Introduction

P2P models have long been considered very promising technologies for the Semantic Web [1][2]. There are, however, a number of difficult theoretical and practical challenges, see section 2, which have hindered widespread usage. A number of frameworks have been proposed targeted at specific scenarios (see section 2.2) but so far none have addresses directly the requirements of large scale, “napster like” metadata exchange P2P applications. This scenario, for example, is clearly not possible if browsing and querying is based on invoking arbitrary remote query executions.

We detail our model in section 3: in the P2P network each peer has a local RDF database and, using the presented algorithm (RDFGrowth), increases its internal knowledge by discovering and importing it from the others. This procedure, described in 4, causes a minimal external computational burden that is in any way independent of the user browsing and querying activity. With respect to this, this approach offers a

greatly increased user scalability opening the way to real world, application ideas based on RDFGrowth are highlighted in section 5.

2. Semantic web and P2P: challenges and related works

While there are multiple challenges in P2P semantic web system (e.g. Ontology integrations), we focus here on CPU and Network traffic scalability issues in general RDF utilization.

2.1.1. Computational burden

On the WWW, the interaction is based on HTTP requests/replies that in the great majority of the cases will be of limited impact on the server (e.g serving a file). This means that, disregarding anomalous cases, both the computational resources and network traffic required by a HTTP request are bounded.

On the contrary, “requests” on the semantic web are naturally expressed in query languages and, given the graph nature of RDF structured information, the complexity of execution is not bounded a priori as it is a function of the query type as well as the quantity and the structure of the data. In other words, whoever would decide to offer the ability to answer “arbitrary questions” on a SW P2P, would easily open himself to “denial of service” situations even in the ideal, good faith usage.

2.1.2. Network traffic

In [3], drawing inspiration from research in classic relational databases, a heuristic is proposed to optimize the query execution plan taking into consideration both the computational and data transfer costs. In case of queries involving the RDF equivalent of “joins” over distributed data sources, however, large dataset transmission is usually unavoidable unless sources provide natively the required “join” capabilities. Providing such capability (as in a mediator) outside the original sources is essentially equivalent to copying the whole dataset. Finally, it is to be noticed that the classic distributed DBMS techniques are just of limited applicability. While in typical DB scenarios there is a concept of “table” aggregating statements about a certain class of resource (e.g. a table responsible for statements about all the users), RDF annotations are distributed in nature as statements about a given URI can be made in different locations on the web and could all be useful when answering a semantic query, with heavy consequences on network traffic.

2.2. Existing P2P Semantic Web frameworks and related works

Considering the above complexities, alternative approaches have been proposed in literature to address the specific needs of different distributed RDF use cases.

2.2.1. Centralized repository plus crawlers

One approach could be to have a central repository where crawlers deliver RDF annotations [4][5]. While this would require large infrastructure, it would certainly not allow arbitrary complexity queries and there could be considerable refresh times be-

tween successive updates. Additionally, when information spread and searching is concerned, centralized systems are to be questioned about their ability to guarantee unbiased treatment of information.

2.2.2. P2P Query distribution

In [6] RDF queries are broadcasted for execution through the entire network and results are collected when they are sent back. In [7] super peers and schema based routing are added, with the effect of limiting the number of queries sent to the final servers. In [8], a further optimization is considered, with the concept of “publish subscribe”, thus providing an alternative to reissuing complete queries to obtain new results. In cases where there is a limited number of powerful query servers and a limited number of disciplined users (e.g. interlibrary query, as in the EDUTELLA project), these P2P models are probably ideal. Their characteristics, however, make them less suitable in other scenarios. Query results are in fact limited to the union of those obtainable in the query execution space of the single servers, thus resembling more the union of traditional DB results rather than queries across a distributed information graph. Secondly, the approach has limited scalability since each end “information source” is individually going to sustain a load that is proportional to the number of users. In case the query syntax is not fixed a priori, the computational burden issue as described in section 2.1.1 fully applies. When subscription and schema routing models are used, “access point” servers are furthermore expected to verify a potentially large number of queries each time a new information is published or a query needs routing.

2.2.3. P2P RDF storage

Distributed RDF storage is considered in the RDFPeers framework [9]. By making use of distributed indexes on the Subject, Predicate and Object of each stored RDF triple, efficient retrieval can be obtained. Simple queries, e.g. in the form of selecting all triples with a given subject, are shown to be solved using a number of network hops in the order of $\log(N)$ where N is the number of peers participating. Unlike [7] however, due to its indexing structure, the system does not make use of schema knowledge neither for storing or query execution. As a result of this, trying to answer queries that make use of ontologies or even simple RDF schemas (as in second generation RDF query languages [10]) could require massive amount of data (potentially all the graph) to be moved from the network to the individual querying peer for local execution.

While the system guarantees that the computational requirements for each remote invocation is minimal, the number of remote invocation per peer in a scenario where each peer is querying at a constant rate remains a function of the number of participants, thus indicating a limit for usage scalability. Furthermore, each request will be answered with the list of all the triples with the given subject, predicate or object and this generates not only a lot of requests for peers storing popular RDF nodes but also each answer could be large in size due to the same node appearing in many statements.

Overall, this approach seems ideal for graph size scalability and information seek time, while relatively good overall scalability is obtained under the assumption that peer queries will be limited to simple or very infrequent complex ones.

3. Incremental P2P scenario, characteristics and definitions

3.1.1. Semantic Web P2P User Communities: scenario and features

In this paper we introduce a scenario of a P2P network where each peer has a local RDF database and is interested in growing its internal knowledge by discovering and importing it from others peers in a group in a sustainable way (see P1, and P2 in the next section). As all the searching and browsing is based on the local database, this scenario is of particular practical interest, main reasons being:

- 1) The local computational power is almost fully available for queries originated locally. In this sense, the complexities of the queries that a user can decide to perform doesn't need to be bound a priori, as opposed to the remote query execution scenarios.
- 2) Local filtering policies. When information is received the users have the ability to locally “tag” it with a trust rating to be then used at browse time. Information that does not reach a selected trust level would simply not contribute at all to the browsing experience.
- 3) Browsing activity does not directly cause network traffic. A single peer bandwidth limitations will be shown to cause minor bottlenecks in the overall growth performance. Browsing or inserting new information off-line or over dial up lines is possible.

3.1.2. Requirements and guidelines

The analysis of the scalability limitations of the existing approaches suggested the following design principles in addressing the P2P communities scenario previously highlighted:

- P1) The system can't assume any reliability. Individual peers (e.g. A very slow or poorly networked one) should never be bottlenecks to the operativity of the others. No peer should have a particular importance in the system. Furthermore the system should behave reasonably also in “high churn rate” conditions, that is when peers frequently decide to join and leave.
- P2) Each peer must be requested to answer only queries which generate a bounded computational and network load sufficiently low for end user machines. This rule should apply uniformly in all the states of the system.
- P3) The system should reward social cooperative behaviours rather than relying on pure enforcing of the protocol [11].
- P4) Users tend to cluster around specific aspects of knowledge (communities) [11]. The algorithm should therefore allow the user to restrict the attention to the topics he is interested in, with the consequent advantages in terms of efficiency of information exchange.

The following definitions naturally introduce the actual algorithm description.

3.2. Defining swappable subspaces of interest in the RDF knowledge: MSG and RDFN definitions and properties

Goal of the algorithm is to allow a local database to incrementally grow its knowledge by exchanges with other peers. Given a ground RDF graph, it is always possible to transfer all the semantic information contained within it to another peer transferring each triple independently. This is however not the case when blank nodes are present. As blank nodes are not addressable from outside a graph, they only have meaning when tied to all the statements surrounding them, i.e. all the statements where they are the subject or the object. With this regard, the smallest “self contained” chunk of information involving a statement with a blank node must necessarily contain all the statements that use the same blank node. This is not a sufficient condition, however. A minimal graph that has sufficient conditions to allow incremental knowledge exchanged is described in definition 3 below.

Definition 1. An RDF statement *involves* a name if it has that name as subject or object.

Definition 2. An RDF graph *involves* a name, if any of its statements involves that name.

Definition 3. Given an RDF statement s , the *Minimum Self-contained Graph* (MSG) containing that statement, written $MSG(s)$, is the set of RDF statements comprised of the following:

1. The statement in question;
2. Recursively, for all the blank nodes involved by statements included in the description so far, the MSG of all the statements involving such blank nodes;

Although the definition recursively build the MSG from a starting statement, we now show that the choice of the starting statement is arbitrary and this leads to an unique decomposition of the RDF graph.

Proposition 1. The MSG of a ground statement is the statement itself.

Theorem 1. If s and t are distinct statements and t belong to $MSG(s)$, then $MSG(t) = MSG(s)$.

Proof. If t belong to $MSG(s)$, then, by the recursive definition, all statements in $MSG(t)$ belong to $MSG(s)$, so $MSG(t)$ is a subset of $MSG(s)$. We will now show that $MSG(s)$ is a subset of $MSG(t)$, thus proving the theorem.

If t is a ground statement, $MSG(t) = t \neq s$, so t is not a ground statement. If s involves one of the blank nodes of t , then s belong to $MSG(t)$ and $MSG(s)$ is a subset of $MSG(t)$. Recursively, if s involves one of the blank nodes of $MSG(t)$, $MSG(s)$ is a subset of $MSG(t)$. If s does not involves any of the blank nodes of $MSG(t)$, then $MSG(s)$ and $MSG(t)$ must be disjoint, which is against the original hypothesis.

Theorem 2. Each statement belong to one and only one MSG.

Proof. Is it straightforward to see that a statement belongs at least to a MSG, as the definition gives also an algorithm to build it.

Lets suppose that a statement s belongs both to $MSG(t)$ and $MSG(u)$, where t and u are distinct statements. Then $MSG(s) = MSG(t)$ and $MSG(s) = MSG(u)$, so $MSG(t) = MSG(u)$, i.e. they are the actually the same MSG.

Corollary 1. A graph can be univocally decomposed in MSGs. Merging these will restore the original graph.

This is a consequence of the fact that all the black nodes, in the MSG definition, are “properly surrounded” by actual URIs (or literals). As a consequence, a graph can be properly reconstructed between 2 peers by transferring and merging one or more MSG at a time.

Definition 4. The RDFN of a resource is the graph composed by all the MSGs involving the resource itself.

Given these definitions, it can be shown that RDFN of a resource is univocally determined and that a whole graph can be transferred by moving the RDFN of all the involved URIs. Example MSGs and RDFN involving a resource are illustrated in image 1. Although more extended, this definition is similar to the Concise Bound Description as used in the URIQUA semantic web agent model [12].

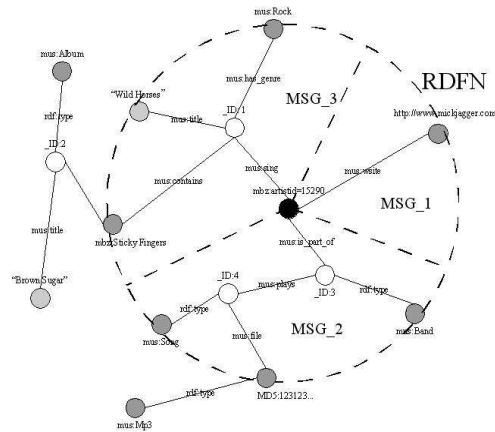


Image 1 A graphical illustration of the MSGs composing the RDFN involving a URI. Statements outside the circle are not included in the RDFN. Darker nodes are ground (URI) while lighter ones are bnodes

3.3. Defining a “subsets of interest” in a RDFGraph

According to P4 design principle, we define a way for peers to select which rdf knowledge they are interested to learn about.

Given R the set of resources identifiable with any URI schema, if we define a G selector operator $G: R \rightarrow \{True, False\}$ and call R_G the induced set

$R_G = \{r \in R : G(r) = True\}$, the overall subspace of interest ($RDF_{interest}$) that a single peer can obtain is then given by:

$$RDF_{interest} \stackrel{def.}{=} U_{\forall RDFdb} \left(U_{r \in R_G} (RDFN(r, RDFdb)) \right)$$

Where by $RDFdb$ we mean a repository of RDF knowledge that participates in the exchange. As a result of the growth algorithm (see 4.5), given that the peers agree on the G operator, the $RDFN$ of each URI in R_G will converge across the peers. In this equilibrium state the $RDF_{interest}$ is therefore a sole function of the G operator.

4. RDFGrowth: algorithm, issues and implementation

4.1. Implementing the G “resource selector” function (GUEDs)

We implement the above mentioned G function, extracting from the local database a set of resources matching a specific topic, with an operator called “Group URI Exposing Definition” (*GUED*).

GUEDs, in a sense, define what the “goods” are in the specified “information market” and can be considered similar to what happens in existing Internet discussion mediums such as the “Group FAQ” often posted in specific Usenet discussion groups to specify what topic are appropriate and what are not.

Example of a such operator for an exchange network interested in “Rap Music” would be “Select all the songs with “rap” genre , select all the albums containing these songs, all the musicians who performed them, and all concerts where they were played”. It is clear how this, in practice, could be simply implemented as the aggregation of results obtained from a prespecified set of queries in one of the semantic web query languages. As the GUED operator will be run very seldom (usually when a peer joins a group) there are no strict requirements on the computational cost of calculating it. As a result it might be interesting to consider more complex rule sets as GUEDs, maybe expressed in higher level language.

As the GUED is itself a resource, in the P2P group it will be indicated with its own, agreed upon, URI. We will see in 4.5 how a GUED-node, is used for the purposes of the algorithm.

4.2. Implementing the RDFN

The current implementation covers directly the definition of $RDFN$ as previously given. For the purpose of evaluating the scalability of the overall system, it is important to analyze the computational cost of a remote $RDFN$ request (from here called R3). Simply, it can be noticed that in the most common situation, when no blank nodes are present, this function can be mapped to a simple relational database query on a unique indexable field. When blank nodes or reifications are present the algorithm explores them but will stop at the first non-blank node. The cost associated with the operation, in terms of simple queries, is therefore linear with the number of blank

nodes or reifications attached to the original URI. On top of the low computational cost it is also straightforward to consider employing a local R3 cache.

Given that the RDFN description of a resource is the only thing a peer can ask another in the described algorithm (see 4.5 and 4.4), we comply with the P2 design rule.

4.3. RDFN “oracle signatures”

A “signature” heuristic is used to avoid interrogating all the group peers about their specific RDFN about a given URI.

We define a RDFN signature as a concise information derived from the information known by the peer about a URI. Purpose of the RDFN signature is to enable an “oracle heuristic” to select among a list of signatures the one corresponding to the peer to whom it is optimal to have an exchange with, if such exchange exists, or will allow it to stop exchanging otherwise.

In the above definition, “optimal”, has to refer to a particular aspect one decides to optimize (e.g. Minimal number of overall exchanges, vs fastest personal knowledge growth) but the necessary property of a signature is to effectively determine a stop condition. As there might be more signature “kinds”, ideally the oracle will consider them all before suggesting to perform a R3. We here define *hit* when, after a R3, the retrieved information is not found be a subset of the local RDFN and define *miss* the opposite situation. *Hit ratio* is also immediately defined as the average number of signature hits over R3s.

In our implementation, one of the signatures is a MD5 hash of a canonicalized RDFN (a procedure similar to [13]) and is therefore capable to avoid that peers having identical RDFN will initiate calls to each other. Adding a local signature memory furthermore prevents that peers initiate calls to others who have information corresponding to older states of the local db. Clearly, the hit ratio can be further increased by adding more signatures for example by indicating the date of the production of the most current annotation or the number of statement in the RDFN itself. While the oracle heuristic (given the MD5 signature alone) does in fact only guarantee that the hit ratio will be strictly above 0, this condition alone can be shown to be sufficient for the convergence of the $RDF_{interest}$ across the peers. In quasistatic exchange conditions, when new information is inserted with an average frequency significantly lower than the average time of news propagation across all the peers, this simple MD5 signature alone is sufficient for a hit ratio of approximately 1.

A further immediate optimization is possible. After a miss, by definition there would be an an hit if the other side was to issue a R3. When implemented, this “R3 induction” strategy alone would make it that *each* R3 initiated would benefit at least one of the peers, with obvious positive effects on the overall convergence speed.

4.4. Knowledge Exchange Network Interface

The “Knowledge Exchange Layer” (KEL) is the generic API on top of which we build the knowledge growth algorithm. We define an “Entrypoint” (EP) as a concise and optimized structure composed of: a record containing a URI, a network peer address

and the relative RDFN oracle signatures. A KEL implementation (driver) can be built on top of any network facility by supporting the following schematized API:

- Publish (EP) : Takes a URI and publish it along with its “oracle signatures” as evaluated by the publishing peer
- EP[]=Lookup(URI) : Gets a list of EPs each indicating a peer that has previously Published that URI. The signatures will be immediately available attached to the list of EPs.
- RDFN=GetRDFN(EP [,MYEP]) Returns the RDFN associated with the EP. This usually corresponds to a direct P2P call to the peer that published the EP. Optionally, the local EP can be passed along to support the “R3 induction” optimization.
- [optional] Join/Leave(group-name) Optional but recommended. Acts as a partition of the space of the “publish” and “lookup” API calls. Intuitively this would map to join an “interest” group as in P4 design principle.
- [optional] Broadcast(EP) Notifies the participating peers of an EP that we think its highly worth for them to import. This is the case when a peer is certain it possesses more annotations than the commonly known $RDF_{interest}$. Note that function, when not natively available, can be effectively simulated using techniques as described in 4.6.1.

Interestingly, due to the minimal API required, KEL drivers can be implemented on top of a large variety of mediums. Currently available drivers include a local network emulation, a Jabber based client/server model and a one based on the ad hoc developed P2P API “JaSiMPA”¹ are available. A fully distributed JXTA driver is under development. Future drivers might be developed to work over Email, Mailing Lists, News-Groups, Existing P2P networks, Web Publishing and possibly also over Freenet² (thus introducing an interesting scenario of fully anonymized metadata publication and retrieval).

4.5. Main algorithm

The synchronize(URI) procedure

The principal component of the algorithm is the “synchronize” procedure. Taking a URI as a parameter, the peer calls LOOKUP to receive a set of remote EPs. It removes the ones with the same signature as that calculated about the URI on the local DB and will call a heuristic *Hrdfn* which will suggest, using the signatures provided, the best remote EP to get information from. If a valid reply is received when requesting the RDFN from a remote EP, the peer will import the data into the local database. To keep the local EP and our “public state” updated, the signature is then recalculated and the EP republished.

¹ <http://www.dbin.org/JaSiMPA>

² <http://www.freenetproject.org>

Before republishing the EP, the peer checks if it is in possession of information not otherwise known in the group, that is, if the newly calculated signature is not among those of the received EPs. This is usually the case when the peer synchronizes a URI about which new information was inserted locally.

If this is the case, it will attempt to “broadcast” or, if not available, issue a “newsflash” procedure before reinserting. The newsflash procedure is explained in 4.6.1 and avoids being the only peer exposing a new piece of information in a crowded group.

If, at the earlier stage, the GETRDFN had failed, the peer would have removed the corresponding EP from the set and proceeded in the loop.

As a result of this procedure, at the end of the transient period, the local RDFN about a URI will converge to the one of the other peers that also chose to publish and synchronize.

The main loop

The algorithm starts with an indication of the GUED to use (main() parameter). In the main cycle, an RDF graph is created or merged with the previous where the GUED-Node is connected directly with the GUED-matching URIs. This graph, by definition, is fully contained in the RDFN of the GUED URI and will be used to communicate to the other peers the existence of a new URI belonging to the GUED itself.

The peer then cycles by synchronizing its knowledge about the GUED URI and then continuing to synchronize on the URIs which, in the local database, are linked to it. The pseudo-code is listed below:

```

1  MAIN(GUED)
2      WHILE is ACTIVE
3          MERGE_GUED_RDFGRAPH(GUED.URI,GETMATCHINGURI(GUED))
4          SYNCHRONIZE(GUED.URI)
5          URIs ← GETMATCHINGURI(GUED)
6          URIs ← RANDOMIZE(URIs)
7          FOR URI IN URIs DO
8              SYNCHRONIZE(URI)
9          ENDFOR
10         ENDWHILE

11 SYNCHRONIZE(URI)
12     REPS ← LOOKUP(URI)
13     REPS ← REPS - { rep | rep ∈ REPS,
14                   SIGNATURE(rep) = SIGNATURE(URI,LOCALDB) }
15     WHILE REPS ≠ { 0 }
16         REP ← Hrdfn (REPS)
17         RDFData ← GETRDFN(REP)
18         IF RDFDATA ≠ {0}
19             IMPORTTRDF(RDFDATA)
20             EP ← RECALCULATESIGNATURES(EP)
21             GOTO END
22         ELSE
23             REPS ← REPS - {REP}
24         ENDIF
25     ENDWHILE
26     END:IF SIGNATURE(EP) ∉ SIGNATURES(REPS)
27     IF BROADCAST_AVAILABLE
28         BROADCAST(EP)
29     ELSE
30         NEWSFLASH(EP)
31     ENDIF
32     PUBLISH(ENTRYPOINT)

```

4.6. Optimizations and specific issues

Although a considerable number of optimizations and specific issues have been addressed, space limitations here prevent us from discussing them in this paper with the exception of those in 4.6.1. Issues not discussed here include: minimizing the average delay in discovering newly inserted information, maximizing the hit ratio using better heuristic signatures, reducing the overall number of published EP and splitting large RDFN into different EPs.

4.6.1. Newsflashing

As the transport layer (KEL) can have many widely different implementations, a number of optimizations deal with peculiar behaviors for example in presence of delays, network costs or bandwidth limitations. From here, let P be the number of peers in a group, U the number of different URIs in the common GUED and DI the delay before a new published information is available to those that call the lookup call. Let delays and other time units be measured in “timesteps”, the basic units in a discrete time simulation stepping on each call to the KEL API.

If we assume $DI = 0$, the detection of new information can be shown to take on average $U/2$ time steps, full group convergence to be reached after U time steps and the average load of direct requests on the peer that first published the new information to be in $O(\log(P))$. This sustainable load comes from the fact that the peers that learn the new information will republish it and immediately share the request load. When significantly high values of DI are present, however, the peer might have to answer all P peers in the system. It is therefore realistic to consider that a single peer will decide not to answer more than a certain amount of calls per unit of time. This limit is from here called R_{max} .

While there are a number of strategies to deal directly with R_{max} limitation, we illustrate an “epidemic newsflash” procedure, similar to that explored in [14], which effectively not only lower the number of requests on a single peer but also significantly lower the average time for the discovery of new information.

When recalculating the local URI signature, after synchronizing with the group, the peer will be able to determine if it is in possession of information not yet known to the group. This is usually the case if information was added locally.

If this is the case, the peer will select a few random peers from the list of those that were publishing the old signature and propagate the new information. In turn, they will repeat this procedure lowering time-to-live (TTL) counter.

As an additional stop condition, under certain condition, one might consider propagating the news if the TTL is greater than 0 and the news was not already locally known. Implications of these rules are discussed extensively in [14]. Interestingly, under ideal condition such as $DL=0$ and sufficiently high R_{Max} , new information can be proven to spread on average in time independent from U , that is, in $\log(P)$ steps.

In conclusion, the newsflash procedure can be seen as either a way to emulate a broadcast call, by putting a high TTL (e.g higher than $\log(P)$), or a simply a way to overcome the R_{max} limit, reaching a small but adequate number of peers before “going public”. In a actual network, the optimal parameters are the results of an optimization

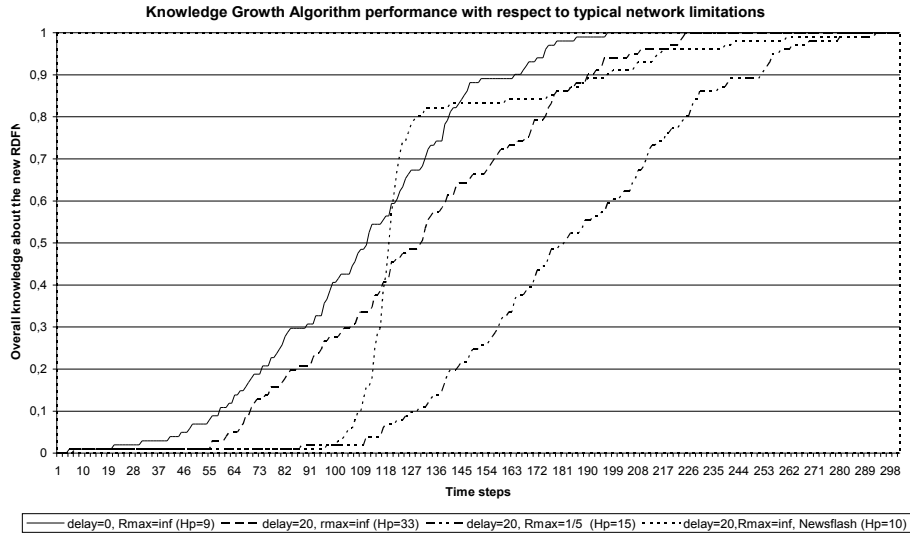


Image 2 Knowledge Growth Performance with respect to typical network limitations. The 100 peer network with 100 URIs in the GUED, is monitored as one new URI is inserted. The effects of delay and network limitations are overcome by the epidemic "newsflash" procedure. The Hp value indicates the maximum number of direct calls that a single peer (usually the newcomer) had to answer.

problem considering the network status and the costs associated with each operation. Experimental results when applying this procedure are shown in the next section.

4.7. Implementation and experimental results

When $DL=0$ and $Rmax=inf$, given that both the updated GUED node and the RDFN of the new node will be discovered in a uniformly distributed probabilistic time due to the randomize procedure in the main() loop, it can be shown that the resulting overall knowledge function is linear in the center with parabolic segments at the beginning and at the end.

In the simulation, a group of 100 peers is populates a group with a converged equilibrium state of 100 commonly known URIs. At time 0, a new peer knowing a new URI with a non empty RDFN is introduced in the network. The results in terms of overall knowledge about the new informations are shown in in image 2, 0 being no knowledge and 1 being full knowledge by all the peers. In the same figure, the other plots illustrate the behaviour in case of non zero DI, when a Rmax limit is introduced and when the auxiliary newsflash procedure is also employed. For completeness it is to be noticed that the newsflash implementation here plotted waits for complete GUED synchronization before spreading the RDFN. A more efficient implementation could merge the two steps.

5. Applicability

Such a “growth only” scenario matches the monotonic nature of the RDF semantics [15]. To obtain more information about a resource can’t in fact “hurt” since, by definition, previously inferred statements will still hold when new data becomes available. It is of course possible, in the real world applications, to rely on “context” information to later apply non monotonic rules on the local database, but it should remain a local peer decision to do so with no consequences on the shared knowledge. Digital signatures are an example of such context information which support several fundamental higher level nonmonotonic behaviours of the overall system.

The MSG definition and properties highlighted in section 3.2 provide a solid framework for signing statements to be spread. Given its properties, it is in fact possible to sign the MSG attaching the signature information to a single, arbitrary statement. The same signature would then, obviously, be valid independently from the RDFN to which the MSG then belongs when sent out in a R3. In fact, a RDFN taken from a remote peer is likely to be composed by several, interdependently signed MSGs and, by applying appropriate “trust chain” filters, it is possible for the user to only see those matching certain trust criteria.

At the same time, also based on the same MSG signing procedure, useful non “static” operativity can be obtained. By inserting into the MSG a statement providing it with an addressable name (e.g. An OWL inverse functional property), it would be possible to later revoke or modify it by spreading another appropriate statement signed by the same author. Complete discussion of these issues is impossible here due to space limitations.

6. Conclusions and future works

In this paper we presented RDFGrowth, an algorithm for semantic web P2P applications. RDFGrowth is targeted at a particular scenario where peers participate in “interest groups” to grow their internal knowledge about one or more “topics”. Being in a untrusted environment, they accept to answer only queries with minimal computational burden and provide no guarantee to answer a query. Furthermore, network traffic is not a direct function of the user's browsing and computational burden is kept local. Under a usage point of view, this directly results in no inherent limitation on the complexity of the operativity, e.g. of the local browsing as well as of the queries. Users with less powerful hardware would select simpler interfaces as opposed to richer ones performing more involved queries or inferences at each browsing step.

It is possible to imagine a wealth of innovative applications using such a framework. In the same way as running a classic P2P file sharing has become a habit for many net users, P2P metadata information sharing software would steadily discover new information about topics about which the user expresses interest (e.g. baseball cards, Italian Opera). Users would be motivated to use applications based on this paradigm because the content and annotations that they produce would precisely reach those who had expressed interest in it. Furthermore, while on classic communication channels (e.g.

newsgroups, mailing lists) information can only reach those who chose to read that specific group in which the post was inserted, in this scenario this limitation is overcome. Annotations about a URI, in fact, are intrinsically and automatically bridged by peers that are attending different groups at the same time, or that have been in those groups previously.

Work is actively in progress both on the issues mentioned in 4.6 and on DBIN, the first actual P2P application embedding it [16]. For these applications to be practically usable, the issues of trust and avoidance of “semantic spam” need early consideration. Current and future work is therefore concentrating primarily on modules supporting this aspect using the techniques mentioned in section 5. A further version of RDF-Growth is also being developed to allow information refusal directly at R3 level with little impact on the overall convergence properties. Other work, also in the DBIN project, deals with the integration in this scenario of MPEG-7 multimedia metadata modules [17]. All the work presented here has been implemented in Java as Free Software and a downloadable application targeted at end user is expected to be released at the Semantic Web Challenge ISWC 2004. Actual application deployments are fundamental to this project as we believe that a meaningful assessment of the model usefulness can only be made by evaluating the feedback from the actual user communities.

7. Acknowledgments

Our gratitude goes to Johan Johansson and Michele Catasta for the support provided.

References

- [1] <http://p2p.semanticweb.org>
- [2] Madhan Arumugam, Amit Sheth, and I. Budak Arpinar , "Towards Peer-to-Peer Semantic Web: A Distributed Environment for Sharing Semantic Knowledge on the Web" WWW2002
- [3] Heiner Stuckenschmidt, Richard Vdovjak, Geert JanHouben, Jeen Broekstra , "Index Structures and Algorithms for Querying Distributed RDF Repositories" WWW2004, May 17-22, 2004, New York, New York, USA.
- [4] Kalvis Apsitis , "RDF Crawler"
- [5] A. Maedche, M. Ehrig, S. Handschuh, R. Volz, L. Stojanovic , "Ontology-Focused Crawling of Documents and Relational Metadata" WWW-2002, Hawaii
- [6] Wolfgang Nejdl, Boris Wolf , "EDUTELLA: A P2P Networking Infrastructure Based on RDF" 2002 WWW2002, Honolulu
- [7] Wolfgang Nejdl, Wolf Siberski, Martin Wolpers, Alexander Löser, Ingo Bruckhorst , "SuperPeer Based Routing and Clustering Strategies for RDF Based Peer-To-Peer Networks" 2003 Twelfth International WWW03 Conference, Budapest
- [8] Paul - Alexandru Chirita, Stratos Idreos, Manolis Koubarakis, and Wolfgang Nejdl , "Publish/Subscribe for RDF-based P2P Networks" 2004 ESWS, Heraklian, Greece
- [9] Min Cai, Martin Frank , "RDFPeers: A Scalable Distributed RDF Repository based on A Structured Peer-to-Peer Network" WWW2004, New York
- [10] Jeen Broekstra and Arjohn Kampman , "SeRQL: A Second Generation RDF Query Language" 2003 13-14 November 2003, Vrije Universiteit, Amsterdam
- [11] G. Paliouras, Ch. Papatheodorou, V. Karkaletsis, and C.D. Spyropoulos , "Learning communities of users on the Internet" 2001
- [12] URIQA The URI Query Agent Model, 2003
- [13] Jeremy Carroll, "Signing RDF Graphs" ISWC2003
- [14] Alan Demers, Dan Greene, Carl Houser, Wes Irish, John Larson , "Epidemic algorithms for replicated database maintenance" 1988
- [15] RDF Semantics, W3C Recommendation , 2004
- [16] G. Tummarello, C. Morbidoni, P. Puliti, A. F. Dragoni, F. Piazza , "From Multimedia to the Semantic Web using MPEG-7 and Computational Intelligence" 2004 WedelMusic 2004 , Barcellona
- [17] G. Tummarello, C.Morbidoni, J. Petersson, P.Puliti, F.Piazza , "P2P Multimedia Annotation and browsing based on Semantic Web and MPEG-7: An overview of the DBin Project" September 2004 Open MusicNetwork Workshop, Barcelona, Spain