# RDFGrowth, a P2P annotation exchange algorithm for scalable Semantic Web applications

Giovanni Tummarello[1], Christian Morbidoni[1], Joakim Petersson[2], Francesco Piazza[1], Paolo Puliti[1]

[1]Dipartimento di Elettronica, Intelligenza Artificiale e Telecomunicazioni
Università Politecnica delle Marche,
Via Brecce 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.** We present RDFGrowth, an algorithm that addresses a specific yet important scenario: large scale, end user targeted, metadata exchange P2P applications. In this scenario, peers perform browsing and querying of semantic web statements on a local database without directly generating network traffic or remote query execution. The database grows by learning from other peers in the P2P group using only a minimal amount of direct queries that are guaranteed to be executable with a low, predictable computational cost. Although full RDF graphs could be treated, the design allows a peer to learn only about resources considered interesting by a specific "community" and makes it possible to tag the received information according to individual trust rules. Inspired by well known viral distributed information techniques, the algorithm is in agreement with the RDF semantics and is specifically suited for the properties of Distributed Hash Table P2P networks. A few assessments about the applicability of RDFGrowth to real semantic web applications are also given.

## 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 addressed directly the problem of large scale, end user targeted metadata exchange P2P applications. This scenario has a few fundamental requirements that we take as starting point for the work in this paper. As an example, browsing and querying "the semantic web" shouldn't be a direct burden for others peers.

We detail our setting 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. Some real world application ideas based on RDFGrowth are highlighted in section 4.

# 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 between 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 cases where there is a limited number of powerful query servers and a limited number of disciplined users (e.g. interlibrary query) the P2P approach investigated in the Edutella [6] seems ideal. In this system, RDF queries are broadcasted for execution trough the entire network and results are collected when they are sent back. Although a successive version of the system [7] has introduced super peers and schema based routing to limit the number of queries sent to the final servers, the approach seems limited to the proposed applications. 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 query server 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 expressed in section  fully applies.

#### 2.2.3. P2P RDF storage

Distributed RDF storage is considered in the RDFPeers framework [8]. 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 [9]) 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. The knowledge growth algorithm

#### 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.

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.
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.

This knowledge base "growth only" scenario is rooted in the monotonic nature of the RDF semantics [10]. To obtain more information about a resource can't in fact "hurt" since, by definition, previously inferred statements will still hold when new data is presented. It will be of course possible, in the real world applications, to rely on non monotonic rules and "context" information (e.g. Digital signatures) on the local database for several purposes (e.g. inference, smart visualization) but it should remain a local peer decision to do so with no consequences on the shared knowledge.

As an example of non-monotonic reasoning useful when applied to a local database, especially in such heterogeneous P2P scenario, we can imagine a "Semantic spam filter" which, e.g., would only visualize the statements coming from "the three most trusted sources currently known".

#### 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. Over a loosely coupled distributed system communication may fail, peers might disappear at any time or refuse to answer remote request. 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 subspaces of interest in the RDF knowledge

Goal of the algorithm is to allow a local database to grow its knowledge with statements found by exchanges with other peers. 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:

$$R_G = \{r \in R : G(r) = True\}$$

If we then define a "RDF neighbours" (RDFN) function shared across the peers:

$$RDFN : R \, x \, RDF \rightarrow RDF$$

which given a URI and a database containing RDF statements, will select a subset of the RDF knowledge "about" a URI. The overall subspace of interest ($RDF_{interest}$) that a single peer can obtain is then given by:

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

Where for RDFdb we mean a repository of RDF knowledge that participates in the exchange. As a result of the growth algorithm (see 3.7), the RDFN function results will begin to converge and become identical across peers. In this equilibrium state the $RDF_{interest}$ is therefore a sole function of the G operator.

### 3.3. 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 3.7 how a GUED-node, is used for the purposes of the algorithm.

## 3.4. Defining RDFN, the annotation "directly" concerning a specific resource

Given a URI denoting some resource, we define as "RDF neighbourhood" (RDFN) of that resource the set of RDF statements, explicitly asserted and/or inferred, comprised of the following:

1. All statements in which subject or object denotes the resource in question; and
2. Recursively, for all statements included in the description thus far, for all anonymous node objects, all statements where subject or object denotes anonymous resource in question; and
3. Recursively, for all statements included in the description thus far, for all reifications of each statement, the "RDF neighbourhood" of each reification.

This definition is similar to that of Concise Bounded Resource Description (CBRD) given in [12], but is extended to comprise also statements where the resource appears as object. The need for such extended definition is clear when considering that human communications, even when structured and expressed with well thought ontologies, is inherently not making a distinction between active and passive statements. As an example:

author → composed → song
song → composed by → author

are both reasonable structured statements with the same overall semantics and should, for the purpose of the considered P2P system, be equally threaded.
While the RDFN seems suited for the current application, future studies might also involve a even broader definition including statements where the selected resource appears as "predicate" and recursively so.

For the purpose of evaluating the scalability of the overall system, it is of great interest to analyze the computational cost of retrieving a resource RDFN. 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 very simple to create a local cache of URI→RDFN.

Given that the RDFN description of a resource is the only thing a peer can ask another in the described algorithm (see 3.7 and 3.6), we comply with the P2 design rule.

## 3.5. 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 that represents the information known by the peer about a URI that will 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 important part is that a signature will effectively determine a stop condition so that a convergence state is determined.

In our implementation, for example, 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.

Thanks to the mutual importing of information, after a full exchange two peers will have identical signatures. From this can be shown that, under normal condition, the simple RDFN hash signature is sufficient for the algorithm to converge the $\text{RDF}_{\text{interest}}$ in the peers.

It is clear how the convergence speed can be optimized by using more advanced signatures techniques. Adding a local memory signature or using combinatory techniques, for example, often prevents that peers initiate calls to others who have information corresponding to older states of the local db. Further RDFN signature aiding heuristics might include indicating the date of the production of the most current information in the RDFN or the size of the RDFN itself.

## 3.6. 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)
  Publish will take a URI and publish it along with its “oracle signatures” as evaluated by the publishing peer
- Lookup(URI)
  The peer performing a Lookup on a specified URI will get 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.
- GetRDFN(EP)
  This will return the RDFN associated with the EP. This usually corresponds to a direct P2P call to the peer that published the EP.
- [optional] Join/Leave(group-name)
  This optional but recommended function 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)
  This optional function will notify 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 $\text{RDF}_{\text{interest}}$. To be noticed that this function, when not natively available,

can be simulated with very positive results using techniques as described in 3.8.1.

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

## 3.7. Main algorithm

### The syncronize(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.

Before republishing the EP, the peer checks if it is in possessions 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 3.8.1 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] http://www.freenetproject.org

```
MAIN(GUED)
        WHILE is ACTIVE
          MERGE_GUED_RDFGRAPH(GUED.URI,GETMATCHINGURI(GUED))
          SYNCHRONIZE(GUED.URI)
          URIs ← GETMATCHINGURI(GUED)
          URIs ← RANDOMIZE(URIs)
          FOR URI IN URIs DO
           SYNCHRONIZE(URI)
          ENDFOR
        ENDWHILE

SYNCHRONIZE(URI)
   REPS ← LOOKUP(URI)
   REPS ← REPS - { rep | rep ∈ REPS,
                          SIGNATURE(rep) = SIGNATURE(URI,LOCALDB) }
        WHILE REPS ≠ { 0 }
            REP ← Hrdfn (REPS)
            RDFData ← GETRDFN(REP)
            IF RDFDATA ≠ {0}
              IMPORTRDF(RDFDATA)
              EP ← RECALCULATESIGNATURES(EP)
              GOTO END
            ELSE
              REPS ← REPS – {REP}
            ENDIF
        ENDWHILE
  END:IF SIGNATURE(EP) ∉ SIGNATURES(REPS)
          IF BROADCAST_AVAILABLE
             BROADCAST(EP)
          ELSE
             NEWSFLASH(EP)
          ENDIF
        ENDIF
        PUBLISH(ENTRYPOINT)
```

### 3.8. Optimizations and specific issues

A considerable number of optimizations and scalability solutions have been considered and implemented. For space limitations, however, we will not discuss the details in this article except for those addressed in 3.8.1. Addressed issues not discussed here include: minimizing the average delay in discovering newly inserted information, maximizing the growth rate of the system therefore minimizing the exchanges using better heuristic signatures and intelligently reducing the overall number of published EP and splitting too large RDFN into different URI entries so that the signature based news discovery maintains efficiency.

#### 3.8.1. Newsflashing

As the transport layer (KEL) can have many widely different implementations, a number of optimizations deal with peculiar behaviors in presence, i.e., of delays, network cost 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 Dl 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 Dl = 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 Dl 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 Rmax.

While there are a number of strategies to deal directly with Rmax 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 sufficently high RMax, 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 Rmax 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 problem considering the network status and the costs associated with each operation. Experimental results when applying this procedure are shown in the next section.

### 3.9. 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 1, 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 Dl, 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.

## 4. Applications

Upon such framework, a wealth of innovative applications can be imagined. In the same way as keeping a classic P2P file sharing has become a habit for many net users, P2P metadata information sharing software would steadily discover new information

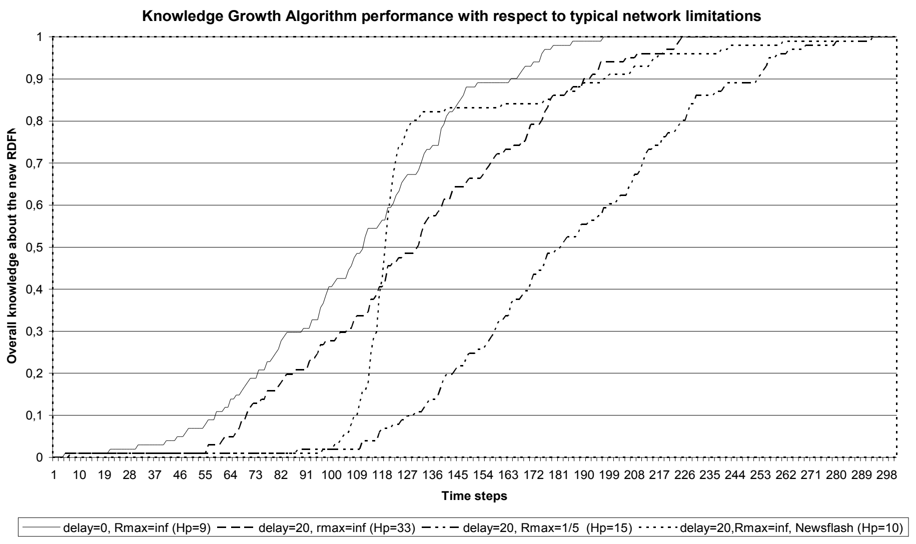

*Image 1 Knowledge Growth Performance with respect to typical network limitations. The 100 peer network with 100 URIs in the GUED, is monitored as one new URI is inserted. The effects of delay and network limitations are overcome by the epidemic "newsflash" procedure. The Hp value indicates the maximum number of direct calls that a single peer (usually the newcomer) had to answer.*

about topics in which the user expresses interest (e.g. baseball cards, Italian Opera). At the same time, applications based on such paradigm would be interesting and motivating to use since things that one would insert would be certain to, slowly but steadily and democratically, reach those that have expressed interest into 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 at an earlier time.

Under a usage point of view, it is to be noticed how point 1 in section 3.1.1 directly results in no inherent limitation on the complexity of the operativity, e.g. of the local browsing as well as of the queries. User with less powerful hardware would select simpler interfaces as opposed to richer ones performing more involved queries or inferences at each browsing step.

## 5. 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 read activitiy and computational burden is kept local. Given these conditions, we consider this approach scalable and useful to a number of applications.

Work is actively in progress both about the algorithm, e.g. about the optimizations mentioned in 3.8, and on the first actual P2P application embedding it. For these applications to be useful, “trust” and avoidance of “semantic spam” are issues that must be taken in early consideration. Current and future work is therefore concentrating primarily on modules supporting this aspect. While, in the current version of the algorithm, peers can only “tag” received information according to local trust rules, an advanced version is being developed to allow explicit P2P refusal of “unwanted” metadata with little impact on the overall convergence properties. Other work, in the DBIN project [15], deals with the integration in this SW P2P scenario of MPEG-7 multimedia metadata modules. A description of this integration is given in [16]. All the work presented here has been implemented in Java as Open Source, multi-platform, Free Software. We believe it is in fact fundamental to seek model usefulness assessment by feedback from actual user communities.

## 6. 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 QueryingDistributed 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" 2002 Proceedings of the Eleventh International World Wide Web Conference 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] Min Cai, Martin Frank , "RDFPeers: A Scalable Distributed RDF Repository based on A Structured Peer-to-Peer Network" 2004 13th International World Wide Web Conference WWW2004, New York
[9] Jeen Broekstra and Arjohn Kampman , "SeRQL: A Second Generation RDF Query Language" 2003 13-14 November 2003, Vrije Universiteit, Amsterdam
[10] RDF Semantics, W3C Recommendation , 2004
[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"
[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] "DBin project" http://dbin.org
[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