Technical University of Catalonia

Master thesis

A Transparent, Reputation-Based Architecture for Semantic Web Annotation

Author:
Laia Subirats

Advisor:
Jordi Forné
Co-advisor:
David
Rebollo-Monedero

Internet Security Group
http://isg.upc.es
Department of Telematics Engineering
Barcelona July 29, 2009

```
<rdf:RDF
xmlns="http://web.resource.org/cc/"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns">
<Work rdf:about="">
   <dc:title>A transparent, reputation-based architecture for semantic
   web annotation</dc:title>
   <dc:date>2009</dc:date>
   <dc:creator>
      <Agent><dc:title>Laia SUBIRATS</dc:title></Agent>
   </dc:creator>
   <dc:rights>
      <Agent><dc:title>Laia SUBIRATS</dc:title></Agent>
   </dc:rights>
   <dc:type rdf:resource="http://purl.org/dc/dcmitype/Text"/>
   <license rdf:resource=
   "http://creativecommons.org/licenses/by-nc-nd/3.0/es"/>
</Work>
<License rdf:about="http://creativecommons.org/licenses/by-nc-nd/3.0/es">
   <permits rdf:resource="http://web.resource.org/cc/Reproduction"/>
   <permits rdf:resource="http://web.resource.org/cc/Distribution"/>
   <requires rdf:resource="http://web.resource.org/cc/Notice"/>
   <requires rdf:resource="http://web.resource.org/cc/Attribution"/>
   <prohibits rdf:resource="http://web.resource.org/cc/CommercialUse"/>
</License>
</rdf:RDF>
```


BibTeX document entry:

```
@MastersThesis{LaiaSubirats:ITACA,
  author={Subirats, Laia},
  title={A transparent, reputation-based architecture for semantic web
  annotation}
  school={Escola T\‘{e}cnica Superior d’Enginyeria de Telecomunicaci{\’o}
  de Barcelona, Universitat Polit\‘{e} de Catalunya},
  year=2009,
  month=sep,
  address={Jordi Girona, 1-3, E-08034 Barcelona; Spain}
}
```

L. Subirats, “A transparent, reputation-based architecture for semantic web annotation”, Master’s thesis, Escola Tècnica Superior d’Enginyeria de Telecomunicació de Barcelona, Universitat Politècnica de Catalunya, Jordi Girona, 1-3, E-08034 Barcelona; Spain, Sep. 2009.

"I have frequently been questioned, especially by women, of how I could reconcile family life with a scientific career. Well, it has not been easy."

Maria Sklodowska

Acknowledgments

I would like to thank especially Jordi Forné and David Rebollo for all their support and invaluable comments on this Master Thesis. I would also like to thank all members of the department, the web technologies lab department of Carlos III university and the semantic department of Lleida university for their kindness and ideas on this project.

I am also very grateful to "la Caixa" Foundation and the Google Anita Borg scholarship who gave me financial support. I will never forget "la Caixa" scholars' meetings or the Google retreat at Zurich, where I could meet other telecommunication, computer science and mathematics student girls from Europe, Asia and Africa.

Regarding my office mates, thanks Jose for introducing me to the world of tea and for helping me with LaTeX, Elisabeth for her body classes knowledge, Aida and Victoria for sharing with me the belly dance lessons, Joan for reminding me of my Artesa de Segre roots, Rafa for his jokes and conversation, the Joomla! team, and all nice people I have met during this master research period. I would also like to mention the debate team for giving me the opportunity to enjoy this quick, but nice, oratory experience. All of you have contributed to make my first year at UPC a wonderful and unforgettable one.

I do not want to forget Esther, Maria, Rafa, Jara, "les nenes", Laia and many other people for all the happy memories we shared. I am very grateful to my parents and family for their love and patience during my studies. Finally, special thanks to David for sharing our lives, both the bad moments and the happy ones, which have been many more.

Contents

List of Figures

List of Tables

Summary

New forms of conceiving the web such as web 2.0 and the semantic web have emerged for numerous purposes ranging from professional activities to leisure. The semantic web is based on associating concepts with web pages, rather than only identifying hyperlinks and repeated literals. ITACA is a project whose aim is to add semantic annotations to web pages, where semantic annotations are Wikipedia URLs. Therefore, users can write, read and vote on semantic annotations of a webpage. Semantic annotations of a webpage are ranked according to users' votes. Building upon the ITACA project, we propose a *transparent, reputation-based* architecture. With this proposal, semantic annotations are stored in the users' local machines instead of web servers, so that web servers transparency is preserved. To achieve transparency, an indexing server is added to the architecture to locate semantic annotations. Moreover, users are grouped into reputation domains, providing accurate semantic annotation ranking when retrieving annotations of a web page. Cache copies of semantic annotations in annotation servers are done to improve efficiency of the algorithm, reducing the number of sent messages.

Keywords: Semantic web, semantic annotation, collaborative systems, reputation.

Section 1

Introduction

Since Tim Berners Lee invented the World Wide Web (WWW) at the European organization of nuclear research (CERN), new forms of conceiving the web, such as web 2.0 and the semantic web have emerged. Web 2.0 refers to what is commonly known as social networks. Nowadays there are many social networks used for numerous purposes such as professional activities, leisure, image or video sharing and many commercial initiatives. The semantic web, also known as web 3.0, is based on associating concepts with web pages, rather than identifying hyperlinks and repeated literals.

Associating concepts with web pages allows semantic interoperability in web applications. The semantic web also permits applications to easily decrease interaction with the user, obtain the semantic distance between web pages, and process pages whose content is nowadays only understandable by humans. Regarding security, we note that each layer of the semantic web architecture defined by the World Wide Web Consortium (W3C), which can be observed in Fig. 1.1, can be secured [2, 3].

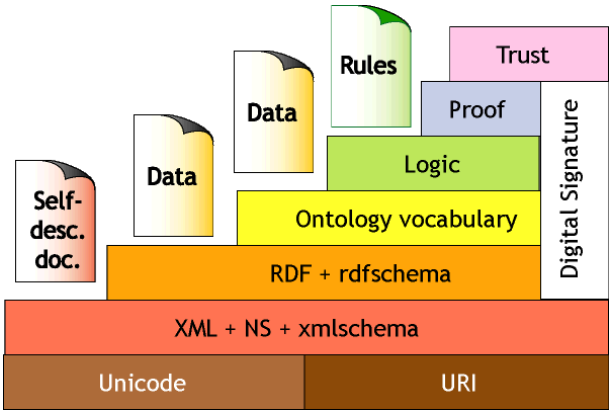


Figure 1.1: Semantic web architecture (Source: World Wide Web Consortium http://www.w3.org).

In this paper, we present a novel approach for a transparent, reputation-based architecture of a particular semantic web project which is called inquiry-

based, with a trustiness model support, semantic annotations and communities formation assistance (ITACA) [4, 5, 6]. In ITACA, peers create semantic annotations to web pages, where semantic annotations are Wikipedia URLs. Users can perform three kinds of actions in ITACA: writing, reading and voting on semantic annotations of a webpage. Semantic annotations are ranked with a voting system, so when a user wishes to read semantic annotations of a web page, the most voted semantic annotations will be displayed.

However, one desirable goal that this architecture does not seem to attain is seamless integration with existing web services. Moreover, user reputation is not considered and this can allow users to make inappropriate semantic annotations. Our contribution improves the existing ITACA architecture by transforming it into a transparent, reputation-based system. Furthermore, DBpedia [7, 8] URLs are proposed to replace Wikipedia URLs, because DBpedia extracts semantic relationships from Wikipedia concepts and more accurate semantic annotations can be obtained. Concepts introduced in the current proposal are summarized in Fig. 1.2.

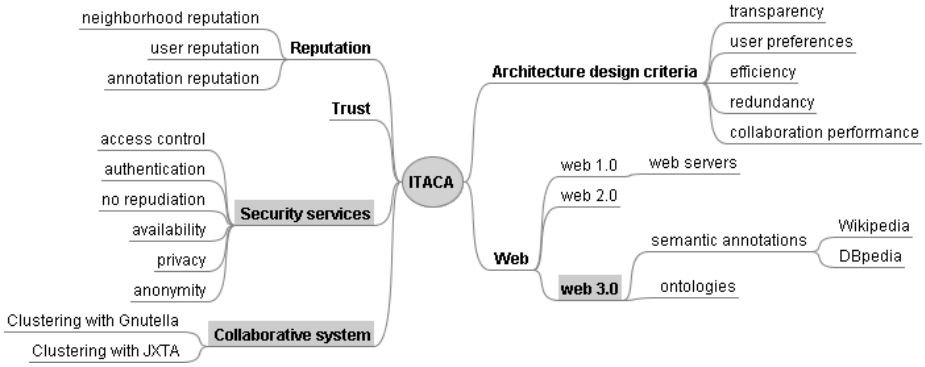


Figure 1.2: Mind map of the current proposal

The rest of the paper is structured as follows. In Sec. 2, the ITACA architecture is introduced, along with some of its problems. In Sec. 3, our proposed architecture is defined, including the block description in Sec. 3.1, functional description in Sec. 3.2, two reputation approaches in Sec. 3.3 and discussion about the defined architecture in Sec. 4 regarding clustering, security and data lifetime. Experimental results are reported in Sec. 5. Sec. 6 reviews different initiatives similar in spirit to ITACA. Finally, conclusions are drawn and future lines of research are suggested in Sec. 7 and Sec. 8 respectively.

Section 2

The ITACA architecture

Our proposal is a review of the architecture presented in [4, 5, 9], where peers create Wikipedia annotations to web pages. Semantic annotations are simply Wikipedia URLs. For instance, the web page `http://www.google.com/anitaborg-emena` can be tagged with Wikipedia concepts such as `http://en.wikipedia.org/wiki/Anita_Borg`.

This Wikipedia URL, together with some metadata information, would be a semantic annotation of the aforementioned URL. In the current ITACA approach, in the first step peers search for a webpage using a search engine, or provide a URL themselves. Afterwards, peers select several Wikipedia concepts to associate with the web page being annotated. After that, a semantic annotation, which consists in associating a Wikipedia URL to a webpage and some metadata information, is created, as can be seen in Fig. 2.1. Finally, the semantic annotation is stored in the web server so that when the URL in question is retrieved, semantic annotations can be retrieved as well.

This functionality is implemented in the ITACA architecture, which is depicted in Fig. 2.2. This architecture defines also a voting system where peers vote on the semantic annotations made by other users. Semantic annotation reputation is computed as the average of user votes of this annotation. When a user requests the semantic annotations of a URL, the list of annotations of the URL is ranked by annotation reputation.

However, the main limitations of this architecture are:

- Web server **transparency** is not preserved, as web servers have to store semantic annotations and perform new functionality.
- **User reputation** is not taken into account, which can lead to some users making inappropriate semantic annotations, as they will not be penalized. Furthermore, it is not possible to weight user votes by user reputation.
- Since both votes and semantic annotations are stored in web servers, **privacy** is not preserved, due to the fact that users have to trust every web server they annotate.
- There is neither **authentication** nor **access control** of resources.
- The model is lacking in **advanced functionalities**, such as enabling users to specify their language, trusted domains, preferences when reading se-

mantic annotations, etc. Therefore, the same annotations ranking is used for all users.

- **Wikipedia is a semi-formal ontology**, and as such no structured semantic information can be obtained between different Wikipedia concepts; consequently, Wikipedia concepts are not part of structured information.

A transparent and secure architecture is defined in Sec. 3 in order to solve these issues.

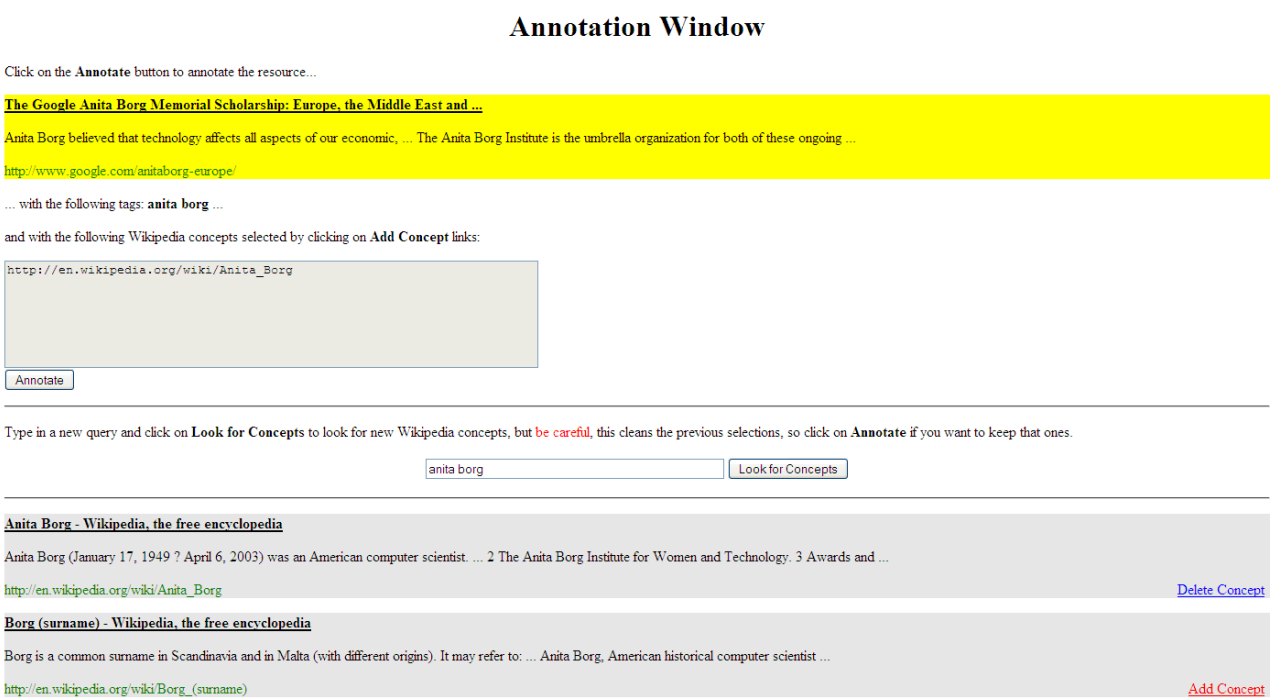


Figure 2.1: Example of adding a semantic annotation to a URL.

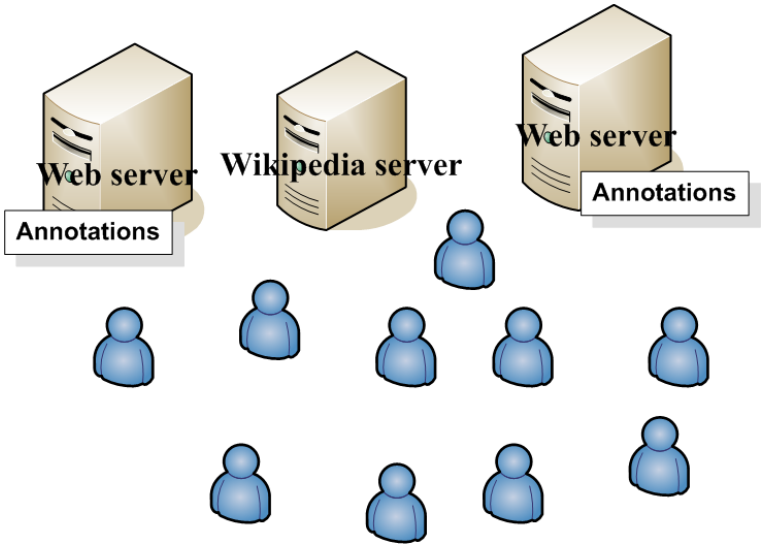


Figure 2.2: ITACA architecture.

Section 3

Definition of the proposed architecture

This section starts with a block description of the protocol (Sec. 3.1), followed by a functional description (Sec. 3.2). Finally, we provide two approaches to computing semantic annotation reputation: the democratic approach, introduced in Sec. 3.3.1; and the credibility-based approach, explained in Sec. 3.3.2.

3.1 Block description

Our proposal consists in grouping users into trust domains, building a collaborative semantic annotation system of web resources. Unlike ITACA, where all semantic annotations were stored in web servers, in our work peers are grouped into communities, and semantic annotations are shared among members of a community. Therefore, transparency of web servers is preserved in this architecture. Our approach assumes that users in a community trust their annotation server with regard to semantic information, votes and preferences. This architecture is formed by entities of the ITACA architecture, such as **web servers** and **users**, and new entities such as annotation servers (which form reputation domains), the indexing server and **DBpedia severs**, as depicted in Fig. 3.1.

There is seamless integration with existing web servers and DBpedia servers, as they neither extend their functionality nor store new variables. Regarding **annotation servers**, they split users into neighborhoods and they reply to requests of their community, the indexing server or other annotation servers. Additionally, they maintain neighborhood reputation by expelling users who do not reach the required reputation of the neighborhood. They also keep cache copies of both semantic annotations and user private information for efficiency. Users must trust the annotation server of their community. Finally, the **indexing server** stores annotation server IDs ($anServIDs$) of the annotation servers which contain semantic annotations of a determined URL.

When defining the protocol to provide the requirements of this architecture, these main design criteria have been followed: transparency, ability of users to decide their policies to read semantic annotations and create their own annotation ranking, anonymity (users reputation is only known by their annotation server) and efficiency (use of caching). Furthermore, DBpedia [7, 8] is suggested

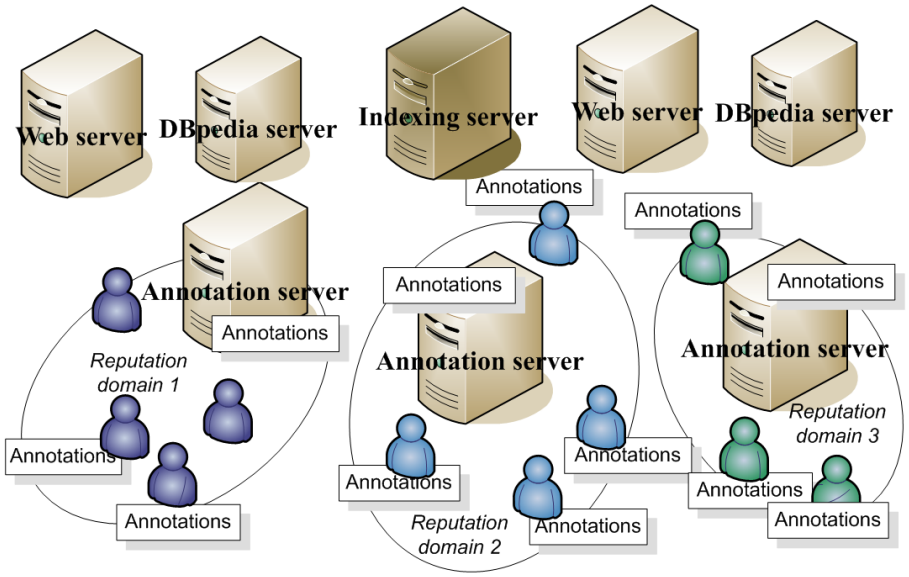


Figure 3.1: Architecture proposal.

as a replacement to Wikipedia, because DBpedia extracts structured information from Wikipedia and semantic relationships between semantic annotations can be obtained. If we follow the example in Fig. 2.1, we can see the correspondent DBpedia entry, shown in Fig. 3.2. The DBpedia page follows "friend of a friend" (FOAF) Vocabulary Specification 0.91 [10], describing relationships among concepts with resource description framework (RDF) [11] and, consequently, enabling easier automatic processing of pages.

3.2 Functional description

The three main actions of the architecture are writing, reading and voting on a semantic annotation.

When **writing a semantic annotation**, the first action is to browse a webpage and then search for a DBpedia concept. After that, the user communicates the semantic annotation he intends to perform to the annotation server he previously registered to. Then, the control is handed over to the annotation server. If that DBpedia concept is already associated to the URL, an exception will be raised. If not, the annotation server stores the annotation and writes the location tuple formed by its ID, the annotation server ID and the URL ($anID$,$anServID$,URL) in the indexing server. Therefore, when users want to browse that page, after asking the indexing server, they know which annotation servers store annotations of that URL. Then, the indexing server confirms that the annotation server ID has been written properly. Afterwards, the user writes his annotation in his local machine. The process of writing an annotation is detailed in Fig. 3.3. Note that the tuple ($anID$,$anServID$,URL) is stored in the indexing server in order to locate semantic annotations in an annotation server domain. With a view to avoiding bottlenecks in the indexing server, several

actions can be carried out, such as:

- Replicating indexing server information in several indexing servers, all of them containing the same information. For simplicity and consistency problems among indexing servers, this option is not considered.
- Copying indexing server information in annotation servers' cache. Sec. 5 evaluates how cache copies help to avoid bottlenecks and improve scalability of the architecture.
- Writing locating tuples in the indexing server, after having several of them (event-driven) or after a timeout (time-driven). This action entails not writing the tuple in every annotation. For simplicity, writing several locating tuples at the same time is left for future studies.

When **reading a semantic annotation** the user tells his annotation server he wishes to browse a determined URL. Afterwards, the indexing server will be in charge of selecting the annotation servers involved with this URL through its location tuples. Location tuples are formed by identifiers of both of annotation servers and annotations and URLs $(anID, anServID, URL)$. Afterwards, the annotation server asks other annotation servers for semantic annotations of this webpage. Finally, the annotation server returns to the user a list of all semantic annotations of this webpage. The process of reading a semantic annotation is detailed in Fig. 3.4. If a user is thrown out of the reputation domain, all his information stored in the annotation server is deleted. Cache memories are used in annotation servers to store their users' information and to communicate with the indexing server. Furthermore, annotations are stored both at users' local machines and in the cache memory of annotation servers. This is done in order to improve the algorithm efficiency and scalability. There is more information about data lifetime in cache and scalability improvement with cache copies in Sec. 4.3 and Sec. 5 respectively.

The action of **voting on a semantic annotation** is rather simple if the annotation is stored in the same annotation server as the user who wants to vote. In this situation, the user informs the annotation server of the annotation ID and the vote value. Afterwards, the annotation server recomputes the annotations' reputation values. Finally, an acknowledgment is sent to the user to confirm that the action has been done correctly. However, when the annotation is in a different annotation server from that of the user who wants to vote, the annotation server should ask the indexing server about the annotation server containing that annotation. Then, annotation reputation will be recomputed. Fig. 3.5 shows the sequence diagram of voting on a semantic annotation. Notice that, because of the use of cache memories, the annotation server may already know in which annotation server the semantic annotation is. Consequently, some of the queries of the sequence diagrams may be omitted.

3.3 Reputation approaches

Annotation reputation makes the ranking of semantic annotations simpler, as annotations can be easily ordered by reputation. However, in a transparent and decentralized scenario, user reputation enables annotation servers to impose a

reputation threshold on users of their domain. In addition, neighborhood reputation, which is computed as the average of the user reputations of a domain, makes the user reputation solution much more scalable. To achieve the best of both approaches, annotation reputation, user reputation and neighborhood reputation have been included in the architecture. User reputation is computed as the average of annotation reputation and neighborhood reputation is computed as the average of user reputation.

As for annotation reputation, we propose two different approaches to compute it: the democratic approach and the credibility-based approach. In the first one, all votes have the same weight. The second approach is more complex, as concepts of witness reputation error and trust come into play, and votes are weighted by user reputation.

Table 3.1: Variables stored in the indexing server, annotation servers and users.

Location	Variable	Access control
Indexing server	Tuples to locate annotations: $anServID$s (annotation server ID), $anID$s (annotation IDs) and URLs.	Public.
Annotation server	Domain information: $anServID$ and $neRep$ (neighborhood reputation).	Public.
	Users' information: uID (user ID), $uRep$ (user reputation) and $warn$ (warnings).	uID is public, and other variables are private.
	Cache: users' semantic annotations and preferences.	Private.
User	uID (user ID).	Public.
	User preferences: $repThres$ (reputation threshold) and $minVotes$ (minimum number of votes).	Private (but their annotation server have access).
	Semantic annotations.	Private (but their annotation server have access).

Common variables used in both approaches are summarized in Table 3.1. User variables are stored in their personal information model ontology (PIMO) profile [6]. However, user identity can be removed from the ITACA architecture [12] in order to preserve anonymity of the semantic annotation (see Sec. 4.2). Semantic annotations are formed by $anID$, $DBpediaID$, URL, $time$, rep, $numVotes$; which are the variables of annotation ID, DBpedia ID, URL, timestamp, reputation and number of votes received respectively. Furthermore, in order to avoid bottlenecks and improve scalability, annotation servers maintain cache copies of semantic annotations and user information. Cache policies will be studied in further detail in Sec. 5.

3.3.1 Democratic approach

There are three different algorithms in the democratic approach, namely: annotation reputation computation, preservation of reputation of a domain and the

algorithm to retrieve semantic annotations of a URL. They are detailed in Algorithm 1, Algorithm 2 and Algorithm 3 respectively. It is worth remarking that although annotation reputation computation algorithm is triggered by a vote on a semantic annotation, user reputation and neighborhood reputation computation algorithms are triggered by a timeout specified by annotation servers. Furthermore, user reputation and neighborhood reputation are computed as the average of annotation reputations and the average of user reputations, respectively.

Algorithm 1 Annotation reputation update. The algorithm is triggered when someone votes on a semantic annotation.

Require: updateRep (rep, $voteValue$, α)
rep: annotation reputation.
$voteValue$: value of the vote on a semantic annotation which has triggered the algorithm.
α: weight between 0 and 1 given to the last vote by the annotation server. Usually α is closer to one as early votes will count more. This is a way to reward early voters because it is more difficult to vote when there are no references.
Ensure: rep is the updated annotation reputation.
1: $rep \leftarrow \alpha * rep + (1 - \alpha) * voteValue$
2: **return** rep

Algorithm 2 Preservation of neighborhood reputation. The algorithm is triggered by a timeout specified by annotation servers.

Require: preservRep ($\{uIDs\}$, $uID.uRep$, $uID.warn$, β, γ)
$\{uIDs\}$: set of users who form the domain.
$uID.uRep$: user reputation of users of the domain.
$uID.warn$: number of warnings of users of the domain.
β: reputation threshold of the domain.
γ: maximum number of warnings before users are expelled from the domain.
Ensure: $\{usersBlackList\}$ and $\{uIDs\}$ are the set of blocked users who had not fulfill reputation requirements of the domain and users in the domain respectively.
1: **for all** $\{uIDs\}$ **do**
2: **if** $uID.uRep < \beta$ **then**
3: **if** $uID.warn > \gamma$ **then**
4: $\{uIDs\} \leftarrow \{uIDs\}$ - uID
5: $\{usersBlackList\} \leftarrow \{usersBlackList\} \cup uID$
6: **end if**
7: $uID.warn \leftarrow uID.warn + 1$
8: **end if**
9: **end for**
10: **return** $\{usersBlackList\}$,$\{uIDs\}$

3.3.2 Credibility-based approach

Variables needed in the credibility-based approach are specified in Table 3.1 and Table 3.2. The system should motivate and reward users that vote and annotate, so that when users write or vote on a semantic annotation, their *participation* increases. The higher the *participation*, the faster their petition to read, write or vote on annotations will be processed by the annotation server. Moreover, it should be advantageous to have a good reputation, to prevent newcomers from reentering the system.

Table 3.2: New variables added to the architecture in the credibility-based approach in annotation servers and users.

Location	**Variable**	**Accesscontrol**
Annotation server	Users' information: uID and $wiRepError$ (witness reputation error). Cache copies of new users' preferences	uID is public, and other variables are private.
User	User preferences: $wiRepErrorThres$ (witness reputation error threshold), $maxShare$ (maximum depth of the trusting graph), trusting tuples (rt, p) (uID has voted rt's annotations with average p), *participation* and *language*.	Private (but their annotation server has access).

As for the algorithms, the **witness reputation error computation** algorithm is detailed in Algorithm 4. Therefore, there are some variations on the **preservation of the neighborhood reputation** in Algorithm 2, because in addition, if user's witness reputation error is above a threshold, the user can be expelled from the neighborhood.

With regard to the **trusting tuple computation**, it is computed in a similar way as the reputation computation in Algorithm 1, but having p as the rep variable and having rt as the uID of the person you vote. Consequently, rep is the average reputation given to a user uID by the community; while p is the average reputation given to a user rt by the uID who voted him. In [13], trust is described by several parameters: target, representation, method and computation. In this approach, the target of this system is users (we remark that, in the previous scenario, trust was centered on annotations). The representation is log files with past trust information. The method used is past interactions. Regarding the computation, it will be triggered when someone votes someone else with a value, and then the user who voted trusts the other user with a certain probability.

In the credibility-based approach, the condition to **retrieve annotations** includes not only annotations higher than a certain threshold, but also annotations of users rt who have in the trusting tuple a p value higher than the threshold. Therefore, both the community reputation and the reputation given by the user to the writer of the annotation are considered.

About: Anita Borg

An Entity in Data Space: dbpedia.org

Anita Borg (January 17, 1949 – April 6, 2003) was an American computer scientist.

Property	Value
dbpprop:abstract	▪ Anita Borg (January 17, 1949 – April 6, 2003) was an American computer scientist. She was born Anita Borg Naffz in Chicago, Mukilteo, Washington. (en) ▪ Anita Borg Naffz (* 17. Januar 1949 in Chicago; † 6. April 2003 in Kalifornien) war eine US-amerikanische Informatikerin und Frau Cyberfeminismus. (de) ▪ Anita Borg foi uma cientista estadunidense. (pt)
dbpprop:birth	▪ 1949-01-17 (xsd:date)
dbpprop:birthPlace	▪ dbpedia:Chicago
dbpprop:death	▪ 2003-04-06 (xsd:date)
dbpprop:deathPlace	▪ dbpedia:California
dbpprop:hasPhotoCollection	▪ http://www4.wiwiss.fu-berlin.de/flickrwrappr/photos/Anita_Borg
dbpprop:reference	▪ http://www.anitaborg.org/ ▪ http://www.gracehopper.org/ ▪ http://www.systers.org/
dc:description	▪ Informatikerin und Frauenrechtlerin (de)
rdf:type	▪ foaf:Person ▪ http://dbpedia.org/class/yago/FellowsOfTheAssociationForComputingMachinery ▪ http://dbpedia.org/class/yago/WomenComputerScientists
rdfs:comment	▪ Anita Borg (January 17, 1949 – April 6, 2003) was an American computer scientist. (en) ▪ Anita Borg Naffz (* 17. Januar 1949 in Chicago; □ 6. (de) ▪ Anita Borg foi uma cientista estadunidense. (pt)
rdfs:label	▪ Anita Borg (en) ▪ Anita Borg (de) ▪ Anita Borg (pt)
owl:sameAs	▪ fbase:Anita Borg
skos:subject	▪ dbpedia:Category:2003_deaths ▪ dbpedia:Category:Fellows_of_the_Association_for_Computing_Machinery ▪ dbpedia:Category:1949_births ▪ dbpedia:Category:Women_computer_scientists ▪ dbpedia:Category:Deaths_from_brain_cancer
foaf:givenname	▪ Anita (de)
foaf:name	▪ Anita Borg (de)
foaf:page	▪ http://en.wikipedia.org/wiki/Anita_Borg
foaf:surname	▪ Borg (de)
is dbpedia-owl:award of	▪ dbpedia:Leah_Jamieson
is owl:sameAs of	▪ yago:Anita Borg

Browse using: OpenLink Data Explorer | Zitgist Data Viewer | Marbles | DISCO | Tabulator Raw Data in: N3 | RDF/XML About

Figure 3.2: DBpedia page example.

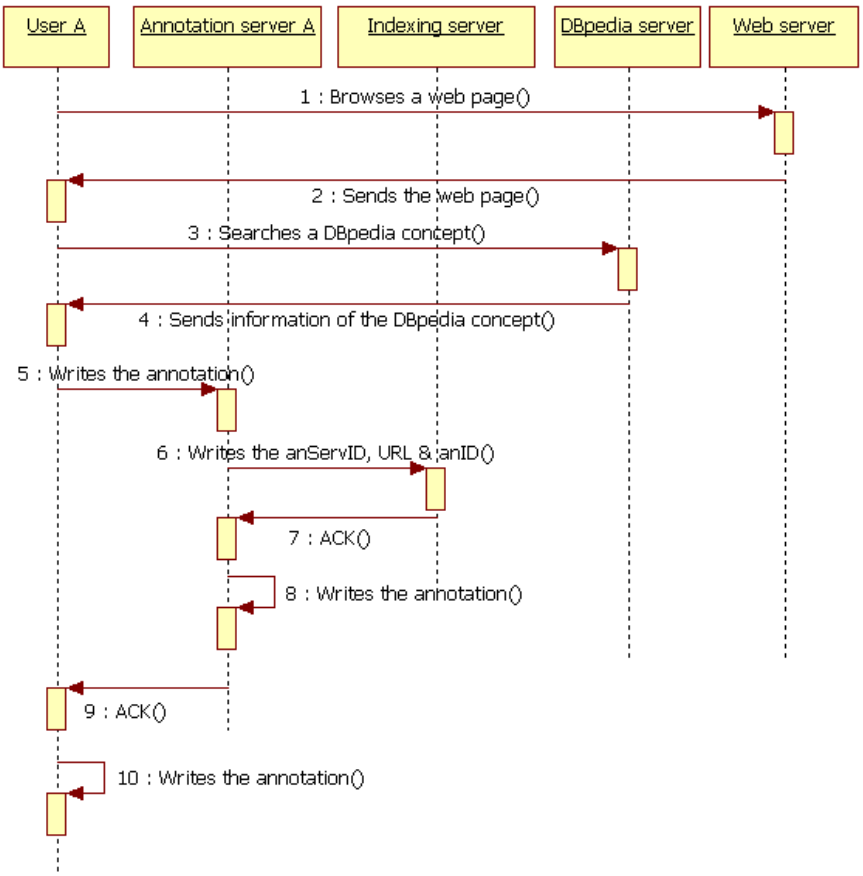


Figure 3.3: Sequence diagram of writing a semantic annotation.

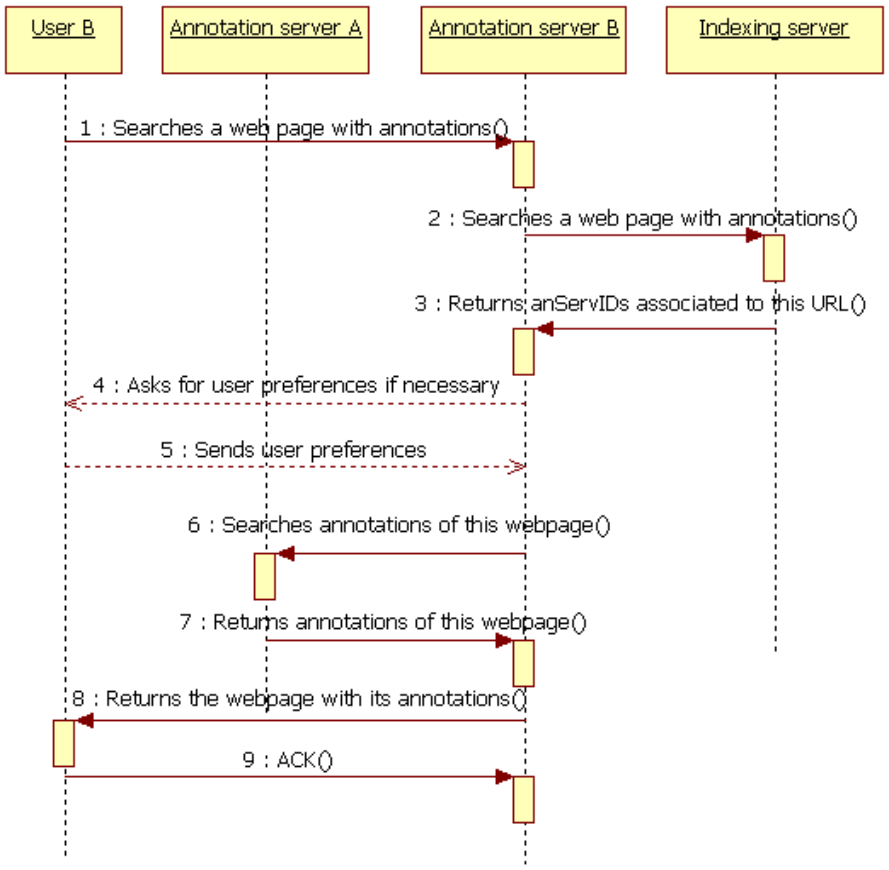


Figure 3.4: Sequence diagram of reading a semantic annotation.

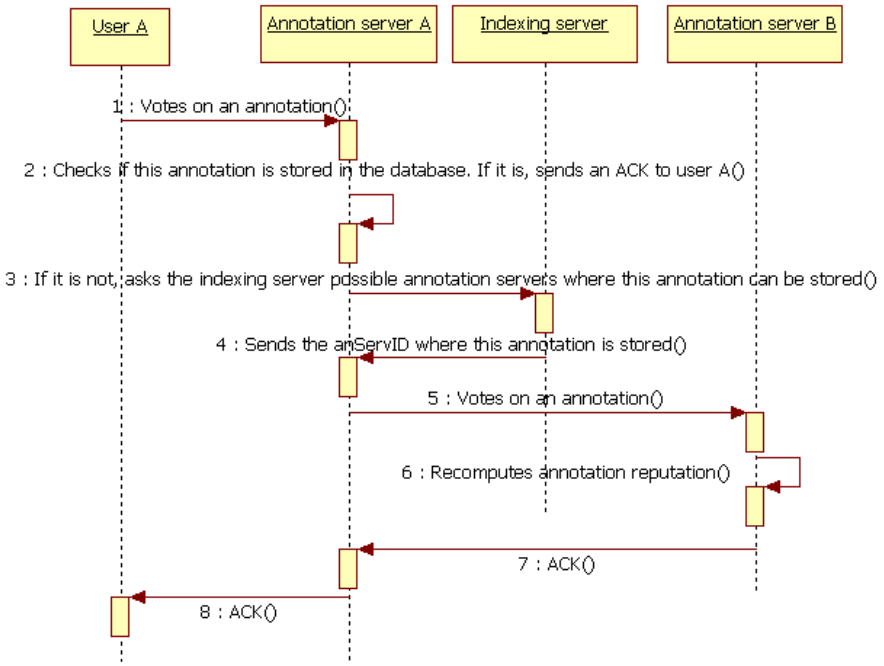


Figure 3.5: Sequence diagram of voting on a semantic annotation.

Algorithm 3 Obtaining semantic annotations of a URL considering user reputation threshold and minimum of votes on an annotation. The algorithm is triggered when someone reads a semantic annotation.

Require: obtainingAn (URL, $\{anServIDs\}$, $anServID.neRep$, $anServID.\{anIDs\}$, $anID.URL$, $anID.rep$, $anID.numVotes$, $repThres$, $minVotes$)
URL: web page where annotations will be retrieved from.
$\{anServIDs\}$: list of $anServIDs$ which have annotations of the URL.
$anServID.neRep$: neighborhood reputation of a domain.
$anServID.\{anIDs\}$: annotations stored in a domain.
$anID.URL$: URL associated to the annotation.
$anID.rep$: annotation reputation.
$anID.numVotes$: number of received votes of an annotation.
$repThres$: annotation reputation threshold required by a user to consider an annotation (it is part of user preferences).
$minVotes$: minimum number of votes required by a user to consider an annotation (it is part of user preferences).
Ensure: $\{DBpediaIDs\}$ is the set of semantic annotations of the URL.

```
1: for all {anIDs} do
2:   if anID.URL == URL then
3:     if anID.rep > repThres&anID.numVotes > minVotes then
4:       DBpediaIDs ← DBpediaIDs ∪ DBpediaID
5:     end if
6:   end if
7: end for
8: for all {anServIDs} do
9:   if anServID.neRep > repThres then
10:    for all {anIDs} do
11:      if anID.URL == URL then
12:        if anID.rep > repThres&anID.numVotes > minVotes then
13:          DBpediaIDs ← DBpediaIDs ∪ DBpediaID
14:        end if
15:      end if
16:    end for
17:  end if
18: end for
19: return DBpediaIDs
```

Algorithm 4 Computation of witness reputation error. The algorithm is triggered by a timeout specified by the annotation server.

Require: wiRepErrorComp (uID, $uID.\{rts\}$, $rt.p, rt.rep$, $dMaxShare$)
$\{uID\}$: ID of the user for whom reputation is being computed.
$uID.\{rts\}$: list of writers of the annotations voted by the user.
$rt.p$: the user voted rt's annotations with an average value p.
$rt.rep$: rt's reputation. In the algorithm there is a summation of the differences between the reputation given by the user and the reputation given by the community.
$dMaxShare$: maximum depth of the trusting graph.

Ensure: $wiRepError$ is the witness error reputation of the user.

1: **if** $dMaxShare == 0$ **then**
2: **return** $wiRepError = 0$
3: **end if**
4: $wiRepError \leftarrow dMaxShare * wiRepError + \sum_{\forall rt} |rt.p - rt.rep| +$ wiRepErrorComp (rt,$rt.\{rts\}$,$rt.rt.p, rt.rt.rep$,$dMaxShare - 1$)
5: **return** $wiRepError$

Section 4

Discussion

In this section, we study clustering methods in Sec. 4.1, security in Sec. 4.2 and data lifetime in cache in Sec. 4.3.

4.1 Clustering

It is important to define how users are assigned to annotation servers. Since Napster was created, many P2P clustering methods have appeared, such as Gnutella and Juxtapose (JXTA). In Gnutella, communities are created according to their queries and the nodes themselves perform access control tasks [14], while in JXTA some users of the P2P network are able to create explicit community groups [15]. Regarding annotation servers, they will store semantic annotations, user preferences, and will act on behalf of the user sending messages to the web server and the DBpedia server. It would be a good idea to use **JXTA** while giving annotation servers the possibility to specify their community in an explicit way, although for automatic clustering Gnutella would be more appropriate. Communities might be specified with the goal of maximizing the number of common web pages among users of a community, so that the algorithm's efficiency and scalability are improved. Furthermore, it must be noted that annotation servers reject users if they do not fulfill reputation or witness reputation error neighborhood requirements.

Notice that when every user has an associated annotation server, we are left with the case of a P2P network. On the other hand, when all users belong to a single annotation server, the architecture becomes a centralized one. Consequently, the whole spectrum of **intermediate situations** between a centralized and a distributed architecture can be analyzed this way.

This architecture has several benefits, such as solving bottlenecks, preservation of anonymity and encouragement of good users' behavior. Regarding **anonymity**, user identity is not used at all by the ranking algorithm, and user reputation is only known by its annotation server. **Bottlenecks** are avoided through cache copies in the indexing server. As for **encouraging good behavior**, reputation gives an incentive to making proper annotations, and likewise witness reputation error promotes fair voting. If a user reputation or witness reputation error are not above thresholds β and γ specified by its domain, the user will be expelled from the domain.

4.2 Security

Several security services are taken into account in this architecture, such as authentication, anonymity, availability and access control.

- Regarding **authentication and anonymity**, in the ITACA approach defined in Sec. 2, every user signs its annotations, leading to the following disadvantages:
  - Integration of key management with existing systems: from a practical viewpoint, most users do not seem to have digital certificates, needed to verify a private key signature; consequently, they are not able to sign messages.
  - Repeated authentication of the reputation value: when the reputation value changes, users should provide their annotation and sign a message again. This can be inefficient if reputation values are frequently modified.
  - Privacy: As users sign their messages, anonymity is lost.

  Due to the foregoing limitations, it seems more appropriate that it be the annotation servers, rather than the users, who sign the messages. However, in our proposal, the authentication scheme is dependent on the action performed, whether it is writing, reading or voting an annotation:
  - Writing an annotation: Users must specify their uID and their annotation server must authenticate the domain to the indexing server. For the sequence diagram of writing an annotation see Fig. 3.3.
  - Reading an annotation: Users do not have to specify their uID, but their annotation server specifies the domain that issued the request to read an annotation. In that case, users use a pseudonym. Consequently, there will be a list of pseudonym and user pairs in the annotation server, reflecting this anonymous reading. For the sequence diagram of writing an annotation see Fig. 3.4.
  - Voting an annotation: Users must specify their uID and their annotation server must authenticate the domain to the indexing server. For the sequence diagram of writing an annotation see Fig. 3.5.
- Cache copies increase the **availability** of both resources and entities. Caching and the associated duplication of information improves not only availability but also scalability. However, with the use of caching the question arises of what the best tradeoff between availability and required storage capacity is. The mechanisms to improve availability of resources and entities are:
  - Resource availability: if a user is not available in the system, and his annotation server has a cache copy of the annotation, the annotation will still be available to the community.
  - Availability of the indexing server and annotation servers: if there is a decline in the number of sent messages among entities, the indexing server and annotation servers will be more readily available,

thus helping to avoid bottlenecks. Annotation servers' availability can be improved with cache copies of annotations of its domain, or even cache copies of annotations of other domains. As for the indexing server, cache copies of location tuples can be time-driven or event-driven; the same question can be asked for writing location tuples in the indexing server and writing votes on annotations of other domains. With time-driven cache copies, location tuples are periodically copied to annotation servers. On the other hand, with event-driven cache copying, location tuples can be copied in annotation servers when writing, reading or voting on an annotation.

Availability is studied in further detail in Sec. 5.

- **Access control** of variables of different entities is specified in Table 3.1.

We proceed to study several possible attacks on the architecture. The source of these attacks can be roughly classified into malicious peers, malicious collectives or malicious spies:

- In the credibility-based approach, **malicious peers** can make a number appropriate annotations, with a view to building a good reputation. Then, attackers proceed to make a high number of bad annotations in a short period of time. As their reputation has not been updated yet (user reputation is updated usually once per day), and as they have a good reputation, their opinions will be trusted by the community. This attack could easily be solved by annotation servers, if we ensure user reputation computation is not only time-driven but also event-driven (for example, every δ new annotations from users, their user reputation must be recomputed). Other problems arise from improper recommendations or malicious peers manipulating the reputation of other nodes (*slandering*), so that lower reputations are reported. Some studies have already appeared tackling this issue [16, 17, 1]. In the credibility-based approach, improper recommendations are detected by means of the witness reputation error in Algorithm 4. When malicious peers exceed the allowed witness reputation error, they are expelled from the neighborhood.

- Reputation or trust outside the community can prove to be quite dangerous. A **malicious collective** forming a domain might falsely pretend to have a high reputation (i.e. to have highly reputed users). In other words, annotation servers can be malicious and pretend to have a reputed community. Malicious annotation servers can also notify the indexing server that a user has written an annotation when he has not. This could be detected if every annotation server computed neighborhood reputation of other domains. For instance, suppose that $currentServID$ makes his own estimate of the neighborhood reputation of $anServID$, as detailed in Algorithm 5. If the neighborhood reputation computed by $currentServID$ differs by more than ϵ[1] from that provided by $anServID$, then $anServID$ might be a malicious collective. Consequently, $anServID$ would be added to $currentServID$'s black list, and neither $anServID$'s resources, votes nor petitions to read annotations will be considered by $currentServID$.

[1] $\epsilon \in \mathbb{R}$ is a threshold specified by every annotation server.

Algorithm 5 Update of the $anServID$'s reputation computed by $currentServID$. The algorithm is triggered by the vote of a user of $currentServID$.

Require: myNeRepUpdate ($vote$, $anServID$, $anServID.myNeRep$, $anServID.elements$)
$vote$: a user of $currentServID$ votes on an annotation stored in $anServID$ with value $vote$.
$anServID$: ID of the annotation server where the annotation is stored.
$myNeRep$: average of votes on annotations stored in $anServID$ voted by $currentServID$'s users.
$elements$: number of annotations stored in $anServID$ voted by $currentServID$'s users.

Ensure: $myNeRep$ is the updated $anServID$'s reputation computed by $currentServID$.

1: $anServID.elements \leftarrow anServID.elements + 1$
2: $myNeRep \leftarrow \frac{myNeRep+vote}{anServID.elements}$
3: **return** $myNeRep$

- **Malicious spies** write correct annotations of the most popular web sites, and likewise vote rightfully on the most popular semantic annotations, but behave maliciously on the less voted annotations or less annotated web pages. This attack is difficult to detect, as users sometimes act maliciously and sometimes do not, and thus their behaviour can remain unnoticed. A deeper study on the detection and prevention of these attacks is left for further work.

4.3 Data lifetime in cache

Data lifetime in cache can refer both to the indexing server variables or to the annotation servers' variables. Data lifetime in cache of the **indexing server** is **permanent**, the reason being that once the tuple to locate an annotation is erased, the annotation will no longer be available to other domains.

As for data lifetime in cache of the **annotation server**, it can refer to users' preferences, users' information, the tuples to locate annotations and semantic annotations. The list of uIDs could be saved in order to prevent malicious peers from reentering the system. However, the following information could be deleted:

- User preferences and information of expelled users.
- Unread annotations[2]. This can be applied to all policies shown in Table 5.1. Another option, which applies only to the third policy, would be to delete annotations created in different domains which have not been read. In this case, the user who created the annotation does not belong to the annotation server's domain.

[2] A counter to compute the number of times that an annotation has been read could be a simple mechanism to discover unread annotations.

- Locating tuples of annotation servers which have not been used. This can be applied to both the second and third policies.
- Unread annotations originated from different domains, together with their correspondent locating tuple of the annotation server. This can be applied only to the third policy.

Even though the deletion of unread annotations would decrease storage capacity requirements while incurring in a very small increase in the number of messages, neither semantic annotations nor location tuple deletions have been considered in simulations, in order to keep the architecture simple.

Section 5

Evaluation of the architecture

We state our assumptions and definitions concerning simulations in Sec. 5.1 and then proceed to discuss the simulations themselves in Sec. 5.2.

5.1 Definition of cache policies and storage requirements

Regarding the architecture defined in Fig. 3.1, bottlenecks in the indexing server are avoided by storing the tuple to locate annotations $(anServID, URL, anID)$ in the annotation servers' cache. In order to reduce bottlenecks in the indexing server, reduce the number of sent messages in the architecture and improve scalability, three different policies are shown in Table 5.1. These three policies are evaluated in Sec. 5.2

Table 5.1: Cache copies in annotation servers' cache when writing (Fig. 3.3), reading (Fig. 3.4) and / or voting (Fig. 3.5) on semantic annotations.

Policy	**Writing semantic annotations**	**Reading semantic annotations**	**Voting semantic annotations**
1	Domain's annotations		
		User preferences	
2	Domain's annotations		
		User preferences	
	Tuple to locate annotations		
3	All domains' annotations		
		User preferences	
	Tuple to locate annotations		

In order to simulate the improvement on efficiency with caching, the storage requirements of a semantic annotation, user preferences, and the tuple to store annotations have been computed in Table 5.2 assuming that the average URL, IDs and timestamp size are 63.4 bytes [18], 8 bytes and 8 bytes [19], respectively.

Table 5.2: Storage requirements of variables stored in users' local machines, annotation servers and the indexing server in the democratic approach.

Location	Variable	Size
Indexing server	$anServID$, URL, $anID$	79.4 bytes
Annotation server	Domain information: $anServID$ and $neRep$	16 bytes
	Users' information: uID, $uRep$ and $warn$	24 bytes
User	uID	8 bytes
	Semantic annotations: $anID$, $DBpediaID$, URL, $timestamp$, $reputation$, $numVotes$	158.8 bytes
	User preferences: $repThres$, $minVote$	16 bytes

5.2 Simulations

Fig. 5.1 has been obtained using Matlab 7.6.0 2008 and Eclipse IDE for Java EE Developers, version 1.2.0.20090621-0820.

The **evolution in time of the number of messages and bytes transferred** has been studied in different policies and scenarios. Taking into account the three policies described in Table 5.1 and three scenarios described in Table 5.3, the evolution of messages and bytes can bee seen in Fig. 5.1.

Table 5.3: Simulation assumptions in three different scenarios.

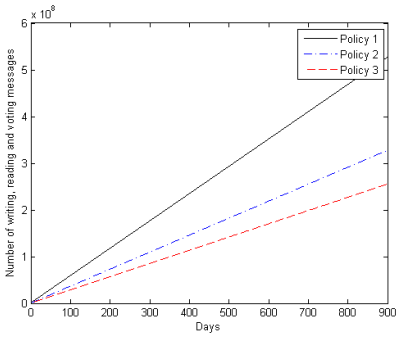
Variable	1st scenario	2nd scenario	3rd scenario
Writing frequency	2 per day		4 per day
Reading frequency	4 per day		
Voting frequency	1 per day		4 per day
Increment of the frequency in writing, reading and voting annotations	0.01		0.1
Number of domains	20	10000	10
Number of users per domain (random number between an interval)	$[100, 1000]$		$[1000000, 2000000]$
Increment of users per domain (random number between an interval)	$[1, 100]$		$[1, 1000]$
Number of read annotations	Proportional to the simulation time		

In the first scenario (Fig. 5.1a), the number of sent messages decreases considerably by applying the second policy. This decrease is somewhat higher with the third policy. However, regarding Fig. 5.1b, the required storage capacity of the third policy is considerably higher. Therefore, the **second policy** exhibits a **better tradeoff** for this architecture and the described scenario.

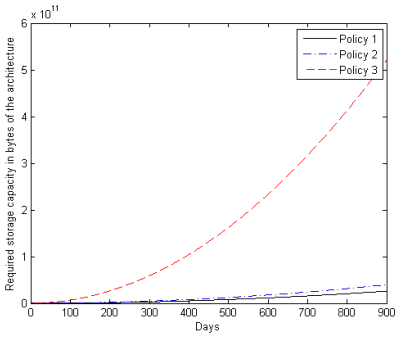
In the second scenario, the **number of domains** increases. A proper selec-

tion of the policy applied becomes crucial for preserving architecture scalability, as depicted in Fig. 5.1c and Fig. 5.1d.

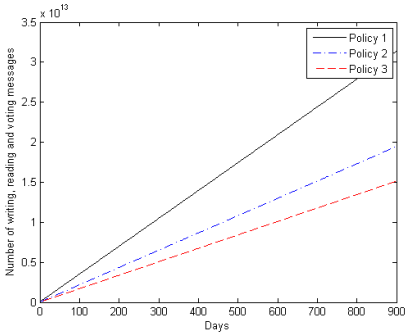
In the third scenario, the **frequency** of writes and votes on annotations increases, as well as the **number of users** per domain. The number of sent messages for the second and third policies become more similar, as can be seen in Fig. 5.1e. In Fig. 5.1f, the number of users in the domain also affects considerably the required storage capacity.



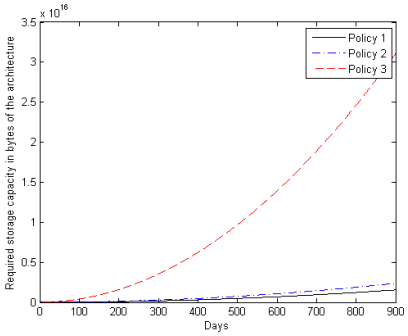
(a) Evolution in time of number of sent messages considering first scenario of Table 5.3.



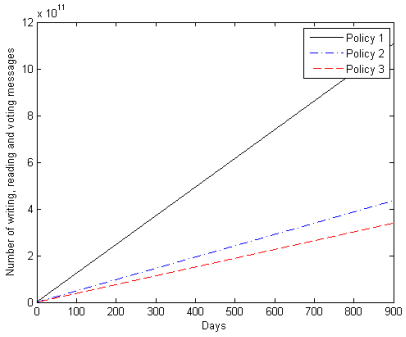
(b) Evolution in time of required bytes considering first scenario of Table 5.3.



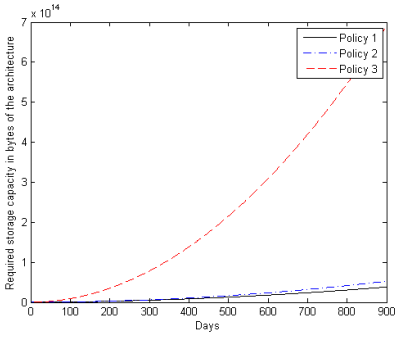
(c) Evolution in time of sent messages when the number of domains increases. Parameters are detailed in second scenario of Table 5.3.



(d) Evolution in time of the number of bytes when the number of domains increase. Parameters are detailed in second scenario of Table 5.3.



(e) Evolution in time of sent messages when the number of users increases. Parameters are detailed in third scenario of Table 5.3.



(f) Evolution in time of the number of bytes when the number of users increases. Parameters detailed in third scenario of Table 5.3.

Figure 5.1: Evolution of messages sent and storage capacity requirements of the architecture, according to three different scenarios.

Section 6

Related work

There are different types of P2P applications, such as file sharing, distributed processing and instant messaging [20]. ITACA is the case of a P2P application that attempts to be secure using concepts such as trust, reputation, access control, authentication and privacy.

Trust [21] becomes much more complicated when dealing with decentralized networks, which were first analyzed by [22]. The problem of trust in P2P networks is that the availability and reliability of a node are not guaranteed [13]. In decentralized networks, monitoring and log files may be needed [23, 24] and some considerations must be taken into account to handle trust in the semantic web [25].

User **reputation** can be computed in different ways (e.g. considering users behavior, users capability, etc.), and in different types of P2P networks, such as Gnutella or Juxtapose (JXTA) [26]. Scalability can be improved using the concept of neighborhood reputation and witness reputation [27].

Regarding **access control** in a semi-decentralized solution, usually there is a central node storing certificates. However, very demanding software and hardware requirements are needed in implementations of completely decentralized architectures, in which path discovery can be a daunting task [28, 29, 30].

As for **authentication** schemes, we can talk about trusted third parties (TTPs). Users trust a TTP and TTPs cooperate between them, increasing the scalability of a P2P network, and reputation information can be shared among these servers [31, 32, 33].

Finally, there is the issue of architecture **privacy**. In semantic networks, it is normal for users to set up their privacy preferences, which include the user with whom information is shared, the type of relationship with the user, and the maximum depth of the privacy graph [34].

Some initiatives have appeared that resemble ITACA, such as reputation assessment for trust establishment among web services (**RATEWeb**) [35]. There are several algorithms to compute web service reputation. It must be noted that in that article, members of the community do not evaluate their annotations, but services offered by other web service providers, based on their perceived quality of service (QoS). In their analysis, it is really interesting to investigate the concepts of **web service** reputation and user credibility, and ontologies are used in order to form communities. However, our democratic approach explained in Sec. 3.3.1 includes the concept of neighborhood reputation, the centralized

authority of the indexing server, and each person's vote has the same weight in the reputation computation.

Another initiative is the **Quatro** project [36]. The Quatro project is a European project where web pages can carry useful and **trustworthy metadata** to make the web a safer place, where users can read the metadata before accessing the website. However, it has an important difference with respect to ITACA: trust in Quatro is generated by labeling authorities, who grant their reliability, whereas in ITACA trust is structured in a collaborative system through user votes, reputation systems, trust systems and witness systems. In the same philosophy as Quatro, [20] presents a reliable polling algorithm which checks if the resources offered are malicious before starting the download of a resource.

There are other initiatives such as infrastructures for **semantic gossiping** where peers have their semantic schemas and, as time goes on, peers should decide whether to implement other people's schemata or adapt their own schema. In this scenario, it would be interesting to see if some **particular schemas dominate the network** [37]. Furthermore, some research has been carried out regarding decreasing resources of secure P2P file-sharing protocols [38]. There are also systems engineered to discover resource-sharing communities, considering the community is a resource described by the metadata specified in an XML Schema description [39].

Another example of a P2P resource sharing initiative is the personalized trust model with reputation and risk evaluation (PET) [40]. That paper differs from others in that it considers the **risk value**, which allows us to consider short-time reputation, thus eliminating the need to wait for a long time to know the reputation of a person. In that project, when calculating the reputation, more weight is supposed to be given to the most recent values. However, we do not consider the risk value with an explicit parameter, as has been done in [40].

There are also studies about how reputation models increase the number of **successful recommendations** as the number of messages increases. In [41], global reputation is a mixture of local reputation and recommendation reputation, and evaluations are made in a Gnutella-like network.

A **visualization of analysis framework for reputation systems**, proposed in [1], can be of great use to summarize differences among reputation systems. Other reputation systems, ITACA and the current proposal, are compared in Table 6.1. The following values have been given to ITACA, defined in Sec. 2, and the current proposal:

- **Formulation**
  - Source of information: manual in both ITACA and the current proposal. Votes are obtained from users ratings, after having written an annotation on a webpage.
  - Information type: both reputation systems include positive, neutral and negative events as an annotation can be voted with 3, 2 or 1 points. Consequently, the reputation metric is discrete in both reputation systems.
  - Temporal aspects: in both systems, reputation does not change over time. However, in the current proposal, earlier votes have more weight on reputation computation, as showed in Algorithm 1.

- Reputation metric: in both systems, annotations' reputation is determined by votes, and consequently, it is deterministic. Reputation is a $\mathbb{R}$ number in the interval $[1, 3]$.

- **Calculation**
  - Distribution: in ITACA, the calculation structure is distributed, as web servers compute annotation reputation of their web pages. In the current proposal, the same distribution is followed for annotation reputation. However, new concepts of reputation are introduced, such as user and neighborhood reputation, which are computed by annotation servers. As in the current proposal there can be a wide range of intermediate situations between a centralized and a distributed architecture (see Sec. 4.1), the distribution of the current proposal will be defined as centralized / distributed.
  - Determinism: in both systems, the calculation is deterministic, because there are deterministic calculations for global reputation values.
  - Efficiency: in both systems, the efficiency of the calculation is $O(1)$ as the algorithm's complexity of the average reputation value is independent of the number of votes. Annotation reputation of ITACA can be computed as $updatedAnRep = \frac{anRep+newValue}{numElements+1}$ and annotation reputation of the current proposal is detailed in Algorithm 1.

- **Dissemination**
  - Distribution: in ITACA, the dissemination of reputation is done by web servers, and hence is distributed. However, in the current proposal, user and neighborhood reputation are distributed by annotation servers and the indexing server. Consequently, the distribution of the current proposal is distributed / centralized.
  - Determinism: the dissemination is deterministic in both systems because their communication mechanisms include distribution hierarchies instead of epidemic-based techniques.
  - Data lifetime in cache: permanent in both cases. For more information regarding data lifetime in cache of the current proposal see Sec. 4.3.
  - Redundancy: there is no redundancy in ITACA, as annotation reputation is only stored in web servers. In the current proposal, there is partial redundancy as user reputations are computed through annotation reputations and stored in annotation servers. In the same way, neighborhood reputation also contributes to this redundancy in reputation.

Table 6.1: Reputation systems comparison (based on [1]).

Reputation systems	**Formulation**				**Calculation**			**Dissemination**			
	Source of Information	Information Type	Termporal Aspects	Reputation Metric	Distribution	Determinism	Efficiency	Distribution	Determinism	Data lifetime in cache	Redundancy
Beth [42]	M, AI	P,N/D	Y	D/C	D	D	$O(a^n)$	D	NA^{ϕ}	NA^{ϕ}	NA^{ϕ}
Zimmermann [43]	M, AI	P/D,C	N	D/D,C	D	D	$O(n^2)$	D	D	P	N
eBay [44]	M	P,N/D	N	D/D	C	D	$O(n)$	C	D	P	N
Yu [45]	AD,AI	P,N/C	N	D/C	D	D	$O(n)$	D	D	T	N
P-GRID [46]	M	N/B	N	D/D	D	P	$O(\log n)$	D	D	P	P
CORE [47]	AD,AI	P,N/C	Y	D/C	D	D	$O(n)$	D	D	T	N
X-Rep [20]	M	P,N/B	N	P/D	D	P	NA^{ϕ}	D	P	P	N
Eigentrust [38]	M	P,N/B,D	N	D/C	D	P	$O(n)$	D	D	T	F
Lee [48]	M	P,N/B,C	Y	D/C	D	D	$O(n)$	D	D	T	P
Trustme [49]	NA^{θ}	NA^{θ}	NA^{θ}	NA^{θ}	D	D	$O(n)$	D	D	P	F
Xiong [50]	M	N/B	Y	D/B	D	D	$O(n)$	D	D	T	N
Buchegger [51]	M,AI	P/C	Y	D/B, C	D	D	$O(1)$	D	D	T	P
Feldman [52]	M	P,N/D,C	Y	D/C	D	P	$O(\log n)$	D	D	P	N
Guha [53]	M	P,N/C	Y	D/B, C	C	P	$O(n^2)$	C	D	T	N
Marti [54]	M	P/C	Y	D/C	D	D	$O(n)$	D	D	T	N
ARA [55]	AD	P/C	Y	D/C	D	D	$O(n)$	D	NA^{ϕ}	T	P
Scrivener [56]	AD	P,N/D	N	D/D	D	D	$O(1)$	D	D	P	P
Song [57]	M,AI	P/C	Y	P/C	D	P	$O(n)$	D	NA^{ϕ}	NA^{ϕ}	NA^{ϕ}
Trustguard [58]	NA^{θ}	P,N/C	Y	D/C	NA^{θ}	NA^{θ}	$O(\log n)$	NA^{ϕ}	NA^{ϕ}	NA^{ϕ}	NA^{ϕ}
Credence [59]	M	P,N/D	Y	D/C	D	D	NA^{ϕ}	D	P	P	P
PowerTrust [60]	M	P,N/D	N	D/C	D	P	$O(n)$	D	D	T	F
P2PRep [61]	M, AD	P/C	Y	P/C	D	P	$O(n)$	D	D	T	P
Li [62]	M, AD	P,N/C	Y	P/C	D	P	$O(n)$	D	D	T	N
ITACA [4]	**M**	**P,N/D**	**N**	**D/D**	**D**	**D**	$O(1)$	**D**	**D**	**P**	**N**
Current proposal	**M**	**P,N/D**	**N**	**D/D**	**C/D**	**D**	$O(1)$	**C/D**	**D**	**P**	**P**

Abbreviations:
Formulation: source of information is manual (M), automatic indirect (AI) and automatic direct (AD); information type is positive (P), negative (N) / binary (B), discrete (D) and continuous (C); temporal focus is strong emphasis (Y) and not a strong emphasis (N); and reputation metric is deterministic (D) and probabilistic (P) / binary (B), discrete (D) and continuous (C).
Calculation: distribution is centralized (C) and distributed (D); determinism is deterministic (D) and probabilistic; and efficiency measures the time complexity of calculating the reputation metric value for a single entity, n (number of nodes) and t (number of historical nodes).
Dissemination: distribution is centralized (C) and distributed (D); determinism is deterministic (D) and probabilistic (P); data lifetime in cache is transient (T) and permanent (P) and redundancy is full (F), partial (P) and none (N);
ϕ means that the property solely depends on the property of the underlying peer-to-peer network.
θ means that the property solely depends on the property of the underlying reputation system.

Section 7

Conclusion

Semantic annotations are undoubtedly essential in the new ways of conceiving the web, where **web resources** are not only accessible by literals but also by **concepts**. Adding concepts or semantic meaning to web pages is the aim of ITACA, which adds Wikipedia URLs to web pages. Through a collaborative voting system, each web page has its ranking of semantic annotations. Therefore, a user is able to write, read or vote on semantic annotations.

However, in the ITACA approach, all annotations are stored in **web servers**, which is not really convenient in terms of web server **transparency**. Therefore, in the architecture we propose, peers are grouped into communities and an indexing server is added to map URLs to users. Each community is managed by an annotation server, where votes of users who form the community are stored. Semantic annotations are written in the users' local machines and cached in the annotation servers when consulted.

To provide a more detailed semantic annotation ranking when retrieving semantic annotations of a webpage, a **reputation system** is created. Ranking is performed through semantic annotation reputations. The reputation of an annotation can be computed by two approaches: the **democratic approach**, where user votes have the same weight, and the **credibility-based approach**, where user votes have different weights depending on their reputation. User reputation is computed as the average of semantic annotation reputations, and neighborhood reputation is computed as the average of user reputations. This reputation model allows each domain to control its users' reputation and consequently, semantic annotation reputation. **DBpedia** [7, 8] is preferred to be used in place of Wikipedia, as DBpedia extracts structured information from Wikipedia and semantic relationships between semantic annotations can be obtained.

When discussing this architecture, several items have been studied, such as clustering, security and possible attacks on the architecture. **JXTA** is thought to be a better option as it gives annotation servers the possibility to specify their community in an explicit way, although for automatic clustering Gnutella would be more appropriate. Several security services are taken into account in this architecture, such as **authentication**, anonymity, availability and access control. Regarding authentication, it is more suitable that annotation servers digitally sign messages rather than relying on users doing it. Forcing users to sign messages has a number of drawbacks, such as difficult integration of **key**

management with existing systems, **repeated authentication of the reputation value** and **privacy**. Several attacks have been classified into malicious peers, malicious collectives and malicious spies, the last one being the most difficult to prevent, as sometimes users act maliciously and sometimes do not.

Three different policies have been evaluated, and the requirements of storage capacity and the number of sent messages through **cache copies** have been analyzed. In the first one, annotations are copied into their annotation server's cache when writing or voting on an annotation. The first policy has the largest number of sent messages and this can lead to bottlenecks and inefficient architectures.

The second policy is the one allowing a better **tradeoff** between the number of sent **messages** and the **storage capacity** requirements. Not only annotations, but also the tuple to locate them, are copied in annotation server's cache memory.

Finally, in the third policy, the annotation servers store not only annotations of their domain, but also annotations of other domains The tuple to locate an annotation is also copied in the annotation server's cache memory. Consequently, the third policy requires the highest storage capacity of all three.

Section 8

Future lines of research

Several improvements can be made on the **reputation** field, such as considering **future reputation** [63] and **qualitative voting**. Another improvement to the architecture would be associating a reputation level with each area of knowledge, and studying area reputation propagation. The propagation of other variables such as trust, considering a certain attenuation law could be also studied. Another area of research would be the use **data reliability indicators**, such as confidence intervals; along with a probabilistic model to improve the reputation algorithm. This would give more information about both annotation reputation and user reputation. Another possible model would add semantic annotations with similar meanings. This would prevent a URL from having two different annotations which share the same concept.

Regarding the **semantic distance** between concepts, it can be useful to apply it to **privacy** analysis [64]. Depending on the semantic fields that users are most interested in, they will have access to certain annotations. Semantic distance can be also applied to **clustering**. It would be interesting to define proximity metrics in order to analyze quantitatively the distance between two users' interests. This way, clustering methods and privacy would depend on how far apart the interests of two users are.

As for **cache copies**, another improvement would be to study a fourth policy consisting in only caching semantic annotations that are useful for the domain. Furthermore, studying the possibility of maintaining a **domain interest** list or the profile of other neighborhood reputation domains or areas, could lead to a considerable performance improvement. Therefore, the capacity requirements of annotation servers would be reduced. Another area to study is whether cache copies should be event-driven, **time-driven** or both. The same question can be asked for writing location tuples in the indexing server and votes on annotations in other domains.

Another possibility is studying **data lifetime in cache**. Semantic annotations in a transient (non-permanent) mode in annotation servers would improve the efficiency of the system. If annotations not frequently read were deleted from the annotation server, storage capacity would become available again without increasing the number of messages sent.

A great improvement to this architecture would be to **detect and avoid malicious spies or other possible attacks** on the system.

Glossary

ARA A robust audit to prevent free-riding in P2P networks, 34

CERN European center for nuclear research, 3
CORE A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks, 34

FOAF Friend of a friend, 10

ITACA Inquiry-based, with a trustiness model support, semantic annotations and communities formation assistance, 1

JXTA Juxtapose model with reputation and risk evaluation, 21

P-Grid Trust overlay networks for global reputation aggregation in P2P grid computing, 34
P2P Peer to peer, 21
PET Personalized trust, 32
PIMO Personal information model ontology, 12
PowerTrust A robust and scalable reputation system for trusted peer-to-peer computing, 34

QoS Quality of service, 31

RATEWeb Reputation assessment for trust establishment among web services, 31
RDF Resource description framework, 10

TrustMe Anonymous management of trust relationships in decentralized P2P systems, 34
TTPs Trusted third parties, 31

URL Uniform resource locator, 1

W3C World Wide Web Consortium, 3

WWW World Wide Web, 3

XML Extensible markup language, 32

Bibliography

[1] K. Hoffman, D. Zage, and C. Nita-Rotaru, "A survey of attack and defense techniques for reputation systems," Purdue University, Tech. Rep. 4, 2009.

[2] B. Thuraisingham, "Security issues for the semantic web," in *Proc. of the Int. Conf. on Computer Software and Applications (COMPSAC)*. IEEE Computer Soc., 2003, p. 632.

[3] ——, "Security standards for the semantic web," *Computer Standards & Interfaces*, vol. 27, no. 3, pp. 257–268, 2005.

[4] N. Fernández, J. M. Blázquez, J. Arias, and L. Sánchez, "A semantic web portal for semantic annotation and search," in *Proc. Semantic Web Engineering Applications (SWEA), colocated with the International Conf. on Knowledge-Based and Intelligent Inform. and Engineering Systems (KES)*, 2006.

[5] N. Fernández, J. B. del Toro, L. S. Fernández, and V. L. Centeno, "Exploiting Wikipedia in integrating semantic annotation with information retrieval," in *Advances in Web Intelligence and Data Mining*. Beer-Sheva, Israel: Springer-Verlang, 2006, pp. 61–70.

[6] N. Fernández, L. Sauermann, and A. B. L. Sánchez, "PIMO population and semantic annotation for the Gnowsis semantic desktop," in *Proc. of the Semantic Desktop and Social Semantic Collaboration Workshop (SemDesk)*. Proc. of the Int. Semantic Web Conf. (ISWC), 2006.

[7] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "DBpedia: A nucleus for a web of open data," in *Proc. Int. Semantic Web Conf.*, Busan, Korea, 2008, pp. 722–735.

[8] DBpediacommunity, "DBpedia webpage," 2007. [Online]. Available: http://wiki.dbpedia.org

[9] N. Fernández, L. S. Fernández, J. B. del Toro, and V. L. Centeno, "Exploiting user queries and web communities in semantic annotation," in *Proc. Int. Workshop on Knowledge Markup and Semantic Annotation (Semannot), colocated with the International Semantic Web Conf. (ISWC)*, 2005.

[10] D. Brickley and L. Miller, "Friend of a friend (FOAF) vocabulary specification 0.91," 2007. [Online]. Available: http://xmlns.com/foaf/spec

[11] World wide web consortium (W3C), "Resource description framework (RDF)," 2004. [Online]. Available: http://www.w3.org/RDF

[12] N. Fernández-García, L. Sánchez-Fernández, J. B. del Toro, and D. Larrabeiti, "An ontology-based P2P system for query-based semantic annotation sharing," in *Proc. Workshop on Ontologies in P2P Communities (OntoP2P) colocated with the European Semantic Web Conf. (ESWC)*, Heraklion, Crete, 2005.

[13] D. Artz and Y. Gil, "A survey of trust in computer science and the semantic web," *Web Semantics*, vol. 5, no. 2, pp. 58–71, 2007.

[14] M. Portmann, P. Sookavatana, S. Ardon, and A. Seneviratne, "The cost of peer discovery and searching in the gnutella peer-to-peer file sharing protocol," in *Proc. IEEE Int. Conf. on Networks*, 2001, pp. 263–268.

[15] S. Botros and S. Waterhouse, "Search in JXTA and other distributed networks," in *Proc. Int. Conf. on Peer-to-Peer Computing*, 2001, pp. 30–35.

[16] Y. Wang and J. Vassileva, "Trust and reputation model in peer-to-peer networks," in *Proc. Int. Conf. on Peer-to-Peer Computing (P2P)*. Washington, DC, USA: IEEE Computer Soc., 2003, p. 150.

[17] D. Donato, M. Paniccia, M. Selis, C. Castillo, G. Cortese, and S. Leonardi, "New metrics for reputation management in P2P networks," in *Proc. Int. Workshop on Adversarial Inform. retrieval on the web (AIRWeb)*. New York, USA: ACM, 2007, pp. 65–72.

[18] J.-L. Guillaume, M. Latapy, and L. Viennot, "Efficient and simple encodings for the web graph," in *Proc. Int. Conf. on Advances in Web-Age Inform. Management(WAIM)*. Springer-Verlag, 2002, pp. 328–337.

[19] K. Ylitalo and Y. Kortesniemi, "Privacy in distributed reputation management," in *Workshop of the Int. Conf. on Security and Privacy for Emerging Areas in Communication Networks*, 2005, pp. 63–71.

[20] E. Damiani, D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante, "A reputation-based approach for choosing reliable resources in peer-to-peer networks," in *Proc. ACM Conf. on Computer and communications security (CCS)*. New York, USA: ACM, 2002, pp. 207–216.

[21] A. Josang, *Trust and Reputation Systems*. Springer-Verlag, 2007.

[22] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in *Proc. IEEE Symp. on Security and Privacy (SP)*. IEEE Computer Soc., 1996, p. 164.

[23] D. Robertson, F. Giunchiglia, F. van Harmelen, M. Marchese, M. Sabou, M. Schorlemmer, N. Shadbolt, R. Siebes, C. Sierra, C. Walton, S. Dasmahapatra, D. Dupplaw, P. Lewis, M. Yatskevich, S. Kotoulas, and A. Perreau, "Open knowledge - coordinating knowledge sharing through peer-to-peer interaction," in *Proc. Int. Workshop of Languages, Methodologies and Development Tools for Multi-Agent Systems (LADS)*, vol. 5118, Durham, UK, 2008, pp. 1–18.

[24] A. C. Squicciarini, E. Bertino, E. Ferrari, and I. Ray, "Achieving privacy in trust negotiations with an ontology-based approach," *IEEE Trans. on Dependable and Secure Computing*, vol. 3, no. 1, pp. 13–30, 2006.

[25] J. Debenham and C. Sierra, "A map of trust between trading partners," in *Proc. Int. Conf. on Trust, Privacy and Security in Digital Business (TrustBus)*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 8–17.

[26] M. Gupta, P. Judge, and M. Ammar, "A reputation system for peer-to-peer networks," in *Proc. Int. Workshop on Network and operating systems support for digital audio and video (NOSSDAV)*. New York, USA: ACM, 2003, pp. 144–152.

[27] J. Sabater and C. Sierra, "Review on computational trust and reputation models," *Artificial Intelligence Review*, vol. 24, no. 1, pp. 33–60, 2005.

[28] B. Carminati and E. Ferrari, "Enforcing access control in web-based social networks," *ACM Trans. on Inform. and System Security*, 2008.

[29] E. Bertino, B. Carminati, and E. Ferrari, "Access control for XML documents and data," *Inform. Security Technical Report*, vol. 9, pp. 19–34, 2004.

[30] L. Qin and V. Atluri, "Concept-level access control for the semantic web," in *Proc. of the ACM Workshop on XML security (XMLSEC)*. New York, USA: ACM, 2003, pp. 94–103.

[31] M. Rodríguez-Pérez, O. Esparza, and J. Muñoz, "Analysis of peer-to-peer distributed reputation schemes," in *Proc. Int. Conf. on Collaborative Computing: Networking, Applications and Worksharing*, 2005, pp. 6 pp.–.

[32] M. Rodríguez-Pérez, O. Esparza, and J. L. Muñoz, "Surework: a super-peer reputation framework for P2P networks," in *Proc. ACM Symp. on Applied computing (SAC)*. New York, USA: ACM, 2008, pp. 2019–2023.

[33] E. Bertino, B. Carminati, E. Ferrari, B. Thuraisingham, and A. Gupta, "Selective and authentic third-party distribution of XML documents," *IEEE Trans. on Knowledge and Data Engineering*, vol. 16, no. 10, pp. 1263–1278, 2004.

[34] B. Carminati and E. Ferrari, "Privacy-aware collaborative access control in web-based social networks," in *Proc. 22nd annual IFIP WG 11.3 working Conf. on Data and Applications Security*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 81–96.

[35] A. Rezgui, B. Athman, and M. Zaki, *A Reputation-Based Approach to Preserving Privacy in Web Services*. Berlin, Heidelberg: Springer-Verlag, 2003, vol. 2819/2003.

[36] QuatroPartners, "The European project Quatro," 2006. [Online]. Available: http://www.quatro-project.org

[37] K. Aberer, P. Cudré-Mauroux, and M. Hauswirth, "A framework for semantic gossiping," *SIGMOD Record*, vol. 31, no. 4, pp. 48–53, 2002.

[38] S. D. Kamvar, M. T. Schlosser, and H. García-Molina, "The eigentrust algorithm for reputation management in P2P networks," in *Proc. Int. Conf. on World Wide Web (WWW)*. New York, USA: ACM, 2003, pp. 640–651.

[39] A. Mukherjee, B. Esfandiarl, and N. Arthorne, "U-P2P: a peer-to-peer system for description and discovery of resource-sharing communities," in *Proc. Int. Conf. on Distributed Computing Systems Workshops (ICDCS)*, 2002, pp. 701–705.

[40] Z. Liang and W. Shi, "PET: A personalized trust model with reputation and risk evaluation for P2P resource sharing," in *Proc. Hawaii Int. Conf. on System Sciences (HICSS)*, 2005, pp. 201b–201b.

[41] W. Wang, G. Zeng, and L. Yuan, "A semantic reputation mechanism in P2P semantic web," in *Proc. Asian Semantic Web Conf. (ASWC)*, 2006, pp. 682–688.

[42] T. Beth, M. Borcherding, and B. Klein, "Valuation of trust in open networks," in *Proc. European Symp. on Research in Computer Security (ESORICS)*, 1994, pp. 3–18.

[43] P. R. Zimmermann, *The official PGP user's guide*. Cambridge, MA, USA: MIT Press, 1995.

[44] I. Onur and K. Tomak, "Impact of ending rules in online auctions: the case of yahoo.com," *Decision Support Systems*, vol. 42, no. 3, pp. 1835–1842, 2006.

[45] B. Yu and M. P. Singh, "A social mechanism of reputation management in electronic communities," in *Proc. Int. Workshop on Cooperative Inform. Agents IV, The Future of Inform. Agents in Cyberspace (CIA)*. London, UK: Springer-Verlag, 2000, pp. 154–165.

[46] R. Zhou and K. Hwang, "Trust overlay networks for global reputation aggregation in P2P grid computing," in *Proc. IEEE Int. Parallel and Distributed Processing Symp. (IPDPS)*, 2006, pp. 10 pp.–.

[47] P. Michiardi and R. Molva, "CORE: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks," in *Proc. IFIP TC6/TC11 Sixth Joint Working Conf. on Communications and Multimedia Security*. Deventer, The Netherlands: Kluwer, B.V., 2002, pp. 107–121.

[48] S. Lee, R. Sherwood, and B. Bhattacharjee, "Cooperative peer groups in NICE," in *Proc. IEEE Computer and Communications Societies (INFOCOM)*, vol. 2, 2003, pp. 1272–1282.

[49] A. Singh and L. Liu, "Trustme: anonymous management of trust relationships in decentralized P2P systems," in *Proc. Int. Conf. on Peer-to-Peer Computing (P2P)*, 2003, pp. 142–149.

[50] L. Xiong and L. Liu, "A reputation-based trust model for peer-to-peer e-commerce communities," in *Proc. IEEE Int. Conf. on E-Commerce (CEC)*, 2003, pp. 275–284.

[51] S. Buchegger and J. Y. Le Boudec, "A robust reputation system for P2P and mobile ad-hoc networks," in *Proc. Workshop on the Economics of Peer-to-Peer Systems*, 2004.

[52] M. Feldman, K. Lai, I. Stoica, and J. Chuang, "Robust incentive techniques for peer-to-peer networks," in *Proc. ACM Conf. on Electronic commerce (EC)*. New York, USA: ACM, 2004, pp. 102–111.

[53] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins, "Propagation of trust and distrust," in *Proc. Int. Conf. on World Wide Web (WWW)*. New York, USA: ACM, 2004, pp. 403–412.

[54] S. Marti and H. Garcia-Molina, "Limited reputation sharing in P2P systems," in *Proc. ACM Conf. on Electronic commerce (EC)*. New York, USA: ACM, 2004, pp. 91–101.

[55] M. Ham and G. Agha, "ARA: A robust audit to prevent free-riding in P2P networks," in *Proc. IEEE Int. Conf. on Peer-to-Peer Computing (P2P)*. Washington, DC, USA: IEEE Computer Soc., 2005, pp. 125–132.

[56] A. Nandi, T.-W. J. Ngan, A. Singh, P. Druschel, and D. S. Wallach, "Scrivener: Providing incentives in cooperative content distribution systems," in *Proc. ACM/IFIP/USENIX Int. Conf. on Middleware (Middleware)*. New York, USA: Springer-Verlag New York, Inc., 2005, pp. 270–291.

[57] S. Song, K. Hwang, R. Zhou, and Y.-K. Kwok, "Trusted P2P transactions with fuzzy reputation aggregation," *IEEE Internet Computing*, vol. 9, no. 6, pp. 24–34, 2005.

[58] M. Srivatsa, L. Xiong, and L. Liu, "Trustguard: countering vulnerabilities in reputation management for decentralized overlay networks," in *Proc. Int. Conf. on World Wide Web (WWW)*. New York, USA: ACM, 2005, pp. 422–431.

[59] K. Walsh and E. G. Sirer, "Experience with an object reputation system for peer-to-peer filesharing," in *Proc. Conf. on Networked Systems Design and Implementation (NSDI)*. Berkeley, CA, USA: USENIX Association, 2006, pp. 1–1.

[60] R. Zhou and K. Hwang, "Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing," *IEEE Trans. Parallel Distributed Systems*, vol. 18, no. 4, pp. 460–473, 2007.

[61] R. Aringhieri, E. Damiani, S. D. C. Di Vimercati, S. Paraboschi, and P. Samarati, "Fuzzy techniques for trust and reputation management in anonymous peer-to-peer systems: Special topic section on soft approaches to information retrieval and information access on the web," *Journal of Amer. Soc. for Inform. Science*, vol. 57, no. 4, pp. 528–537, 2006.

[62] F. Li and J. Wu, "Mobility reduces uncertainty in MANETs," in *Proc. IEEE Int. Conf. on Computer Communications (INFOCOM)*, 2007, pp. 1946–1954.

[63] M. Mejía, N. Peña, J. L. Muñoz, and O. Esparza, "A review of trust modeling in ad hoc networks," *Internet Research*, vol. 19, pp. 88 – 104, 2009.

[64] D. Rebollo-Monedero, J. Forné, L. Subirats, A. Solanas, and A. Martínez-Ballesté, "A collaborative protocol for private retrieval of location-based information," in *Proc. Int. Conf. e-Society (IADIS)*, 2009.