Storing and Querying RDF Data in Atlas*

Zoi Kaoudi, Iris Miliaraki, Matoula Magiridou, Antonios Papadakis-Pesaresi and Manolis Koubarakis

Dept. of Informatics and Telecommunications
National and Kapodistrian University of Athens
Panepistimioupolis, Ilissia
Athens 15784 Greece
{zoi,iris,matoula,apapadak,koubarak}@di.uoa.gr

Keywords

RDF, query processing, Semantic Web, peer-to-peer networks, DHT, scalability.

1. INTRODUCTION

In recent years, more and more resources are semantically annotated, thus generating huge amounts of RDF metadata. Current centralized RDF repositories lack the required scalability and fault tolerance to deal with this emerging situation. Therefore, the need for a scalable system that will be able to scale to millions of RDF triples is becoming prevalent. Distributed hash tables (DHTs) is a recent P2P technology that has been proposed for the scalable and fault-tolerant storage and querying of RDF data [2, 1]. Since annotation is by itself a distributed process it ties very well with the model of work imposed by P2P systems.

In this demo paper, we present Atlas, a P2P system for the distributed storage and retrieval of RDF data. Atlas is built on top of the distributed hash table Bamboo[1] and supports pull and push querying scenarios. It inherits all the nice features of Bamboo (openness, scalability, fault-tolerance, resistance to high churn rates) [10] and extends Bamboo's protocols for storing and querying RDF data. In the OntoGrid project, Atlas is used to implement the metadata service of S-OGSA, a new architecture for the Semantic Grid [3].

2. ATLAS ARCHITECTURE

Nodes in an Atlas network are organized in an identifier ring using the Bamboo DHT protocol. Nodes can enter RDF data into the network and pose RQL [6] queries. In the typical Atlas application that we envision, RDF data will be used to describe resources owned by network nodes (e.g., ontologies or services). Atlas supports two querying scenarios: *one-time* querying and *publish/subscribe*. Each time a node poses an one-time query, the network nodes cooperate to find RDF data that form the answer to the query. In the publish/subscribe scenario, a node subscribes with a continuous query. A continuous query is indexed somewhere in the network and each time matching RDF data is published, Atlas nodes cooperate to notify the subscriber. A high level view of the Atlas architecture is shown in Figure 1.

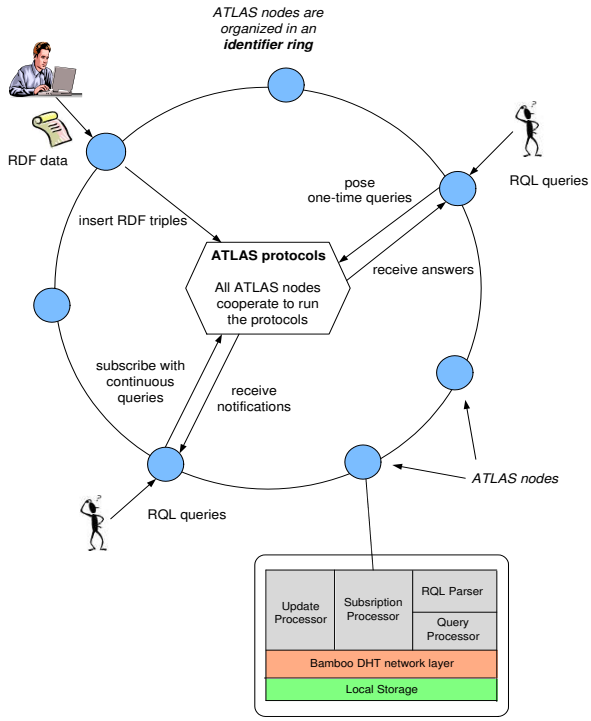

Figure 1: Atlas architecture

In the following, we describe the architecture of each node, which is similar to the architecture of [2] (see Figure 1). We distinguish between six components in an Atlas node:

*This work is partially funded by FP6/IST project OntoGrid. The work was performed while the authors were with the Dept. of Electronic and Computer Engineering, Technical University of Crete.

[1] `http://bamboo-dht.org/`

Demos and Posters of the *3rd European Semantic Web Conference (ESWC 2006)*, Budva, Montenegro, 11th - 14th June, 2006

the *update processor*, the *subscription processor*, the *query processor*, the *RQL parser*, the *local storage* and the *Bamboo DHT network layer*, which is responsible for routing and handling the network messages. We will describe briefly the role and functionality of each component.

The update processor is the component that stores an RDF document or a bunch of RDF data in the Atlas network. It takes as input an RDF document in RDF/XML or N3 format, it decomposes the document into triples and stores the triples in the network. Each triple is stored in the local storage of the appropriate node based on an identifier of the triple. We have used the triple indexing algorithm presented in [2]. The update processor is also responsible for updating or removing triples previously stored in the network. Updates in Atlas are not supported yet but will eventually follow the semantics of RUL [9].

The subscription processor is the component providing a publish/subscribe functionality in Atlas. This component is under development using the ideas presented in [8], which significantly extend the publish/subscribe algorithms of [2].

The query processor is the component responsible for evaluating the queries. The query processing algorithm of Atlas deals with *conjunctive triple-pattern* queries, an extension of the class of queries considered in [2]. The protocols followed for the evaluation of queries are described in [5].

The RQL parser is the component which takes as input RQL queries posed by the node. It parses the query and if it is correctly formed, it produces a conjunctive triple-pattern query and passes it to the query processor. Currently, Atlas supports data RQL queries expressed in RQL.

The local storage is the place where each node stores locally its $(key, value)$ pairs. In the Bamboo implementation, the Berkeley DB database [4] is used. Berkeley DB is an open source database library that provides a simple API for data access and management.

3. DEMONSTRATION SCENARIO

Let us now describe a complete demonstration scenario of our system. Initially, an Atlas network is created by a single node that bootstraps by itself. Then, other nodes join the network following the join protocol described in [5]. The application scenario demonstrated is based on the well-known problem of resource discovery [7]. Two actors are involved in this scenario, the resource providers and the resource consumers.

Resource providers want to publish their resources and create resource descriptions expressed using the RDF data model. They store them in the Atlas network using a store operation. Resource consumers want to discover resources that meet their needs. They pose appropriate RQL queries to search for resource information stored in Atlas.

4. CONCLUSION

We presented Atlas, a P2P system for the distributed storage and retrieval of RDF data. Atlas is implemented on top of the Bamboo DHT network and provides clients with the ability to store and query RDF data. In the future, we plan to expand the functionality of the Atlas system by supporting queries for RDFS and the update language RUL.

5. REFERENCES

[1] Karl Aberer, Philippe Cudre-Mauroux, Manfred Hauswirth, and Tim Van Pelt. GridVine: Building Internet-Scale Semantic Overlay Networks. In *Proceedings of the Thirteenth International World Wide Web Conference (WWW2004)*, New York, May 2004.

[2] Min Cai, Martin R. Frank, Baoshi Yan, and Robert M. MacGregor. A Subscribable Peer-to-Peer RDF Repository for Distributed Metadata Management. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 2(2):109–130, December 2004.

[3] Carole Goble, Ioannis Kotsiopoulos, Oscar Corcho, Paolo Missier, Pinar ALPER, and Sean Bechhofer. S-OGSA as a Reference Architecture for OntoGrid and for the Semantic Grid. In *GGF16 Semantic Grid Workshop*, February 2006.

[4] BerkeleyDB: http://sleepycat.com/.

[5] Zoi Kaoudi, Iris Miliaraki, Spiros Skiadopoulos, Matoula Magiridou, Erietta Liarou, Stratos Idreos, and Manolis Koubarakis. Specification and Design of Ontology Services and Semantic Grid Services on top of Self-organized P2P Networks. Deliverable D4.1, Ontogrid project, September 2005.

[6] Greg Karvounarakis, Sofia Alexaki, Vassilis Christophides, Dimitris Plexousakis, and Michel Scholl. RQL: A Declarative Query Language for RDF. In *Proceedings of the 11th International World Wide Web Conference*, May 2002.

[7] Manolis Koubarakis, Zoi Kaoudi, Iris Miliaraki, Matoula Magiridou, and Antonios Papadakis-Pesaresi. Semantic Grid Resource Discovery using DHTs in Atlas. In *GGF16 Semantic Grid Workshop*, February 2006.

[8] Erietta Liarou, Stratos Idreos, and Manolis Koubarakis. Publish-Subscribe with RDF Data over Large Structured Overlay Networks. In *Proceedings of the 3rd International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P 2005)*, Trondheim, Norway, 28-29 August.

[9] Matoula Magiridou, Stavros Sahtouris, Vassilis Christophides, and Manolis Koubarakis. RUL: A Declarative Update Language for RDF. In *Proceedings of the Fourth International Semantic Web Conference (ISWC2005)*, 2005.

[10] Sean Rhea, Dennis Geels, Timothy Roscoe, and John Kubiatowicz. Handling Churn in a DHT. In *USENIX Annual Technical Conference*, 2004.