# ADAPTIVE RDF GRAPH REPLICATION FOR MOBILE SEMANTIC WEB APPLICATIONS*

**Bernhard Schandl, Stefan Zander**
University of Vienna
Department of Distributed and Multimedia Systems
{bernhard.schandl,stefan.zander}@univie.ac.at

**ABSTRACT**

An increasing number of applications are based on Semantic Web technologies and the amount of information available on the Web in the form of RDF is continuously growing. The adaption of the Semantic Web for Personal Information Management and the increasing desire for mobility is often accompanied by situations where no network connectivity is available and hence access to remote data is limited. Such situations could be obviated when mobile devices are able to operate on offline data replicas and synchronize changes when connectivity is re-established. In this paper we present a framework allowing for adaptive RDF graph replication and synchronization on mobile devices. It extends the architecture of typical Semantic Web applications with components that analyze various information sources of semantic applications (including ontologies, queries, and expressed user interest) and use them for selecting parts of RDF data bases, which are then made available offline using a proxy SPARQL endpoint on a mobile device. Thus, we provide access to Semantic Web data without the need for permanent network connectivity.

Keywords: Semantic Web, replication, synchronization, mobile systems

## 1 INTRODUCTION

The original design of the World Wide Web is *document-centric*: digital information resources are published on servers and can be retrieved by using Uniform Resource Locators (URLs). Such documents are mainly HTML pages with embedded media like images, which are connected by hyperlinks. While there exist a large number of static documents (i.e., documents that reside on a server and are delivered to clients as-is), large amounts of data are embedded in the so-called *hidden web*, which consists of virtual documents that are created on request time using data that is stored in other systems, e.g. relational data bases. In most cases, these data are exposed via query forms and are available to clients also in the form of semi-structured HTML documents.

If the consumer of such data is not a human (through the usage of a Web browser) but a machine, it is required to re-extract the raw data from the HTML representation, being optimized for human consumption, which is usually an expensive and error-prone task [8]. It is the goal of the *Semantic Web* [2] to eliminate this source of potential errors by providing the technical infrastructure to directly publish machine-interpretable information on the Web, thus making it *data-centric*. The Semantic Web builds upon the Web infrastructure [17] and extends it with a meta format for information representation (Resource Description Framework [16]) and languages that allow publishers to semantically describe their data (e.g., RDF Schema [5] and Web Ontology Language [11]). This technology stack has been complemented by the activities of the Linked Open Data initiative, which demonstrate how to publish and interlink data sets using Semantic Web technologies [3] and hence creating a world-wide distributed database.

Recently, the application of Semantic Web technologies to the problem of Personal Information Management (PIM) has gained lots of interest, most notably in the form of the *Semantic Desktop* [22], which has been investigated in the course of a number of projects (e.g., [14, 18, 23]). With the increasing proliferation of mobile devices like smart phones or netbooks, issues of Personal Information Management are no longer restricted to desktop machines. In mobile scenarios, users frequently face the problem that data are not available because of several reasons: first, there may be no physical network connectivity (e.g., because of the lack of mobile network coverage), and secondly, security restrictions may apply (e.g., a VPN connection to the company network cannot be established). In such situations it is desirable to make relevant data available on the mobile device so that applications can operate offline, and to synchronize changes back

* This paper is an extended version of [24].

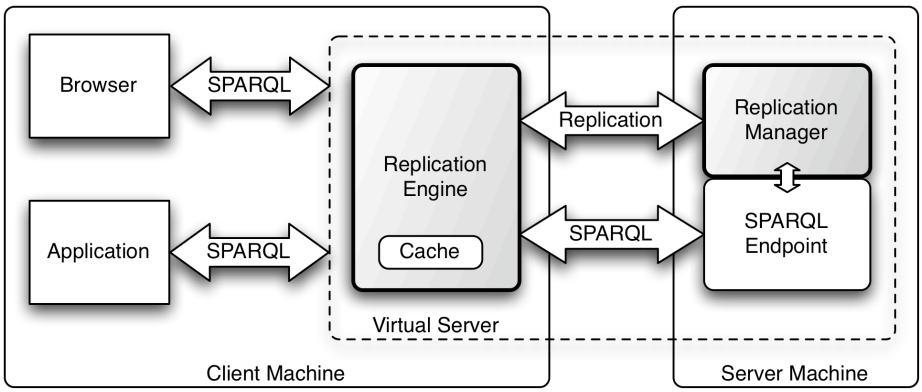


**Figure 2:** Proposed Architecture Extension by an Intermediate SPARQL Proxy

to the base data set when connectivity is recovered. However, because of the still limited storage and computing power of mobile devices, it is advisable to carefully select the information to replicate; ideally in an automatic, transparent, and adaptive manner.

In this paper we present the *MobiSem Replication and Versioning* (MRV) framework that aims to provide this functionality. Its architecture consists of a number of middleware components that selectively replicate data from an RDF data base to a (mobile) client. This selection is done by considering, on the one hand, automatically derived metrics about the data set and its usage, and, on the other hand, manually defined rules that allow the user to specify subsets of the data to be replicated. On the mobile device, replicated data are wrapped by a SPARQL endpoint to be transparently used by applications.

## 2 MOBILE RDF REPLICATION AND SYNCHRONIZATION ARCHITECTURE

In Fig. 1 the typical architecture of RDF-based applications is depicted. Such applications usually consist of two main components:

- A *SPARQL endpoint*, which wraps an RDF dataset and hides its implementation details from a client. The data may actually be stored in a relational database, in the file system, in memory, or it may be accessible via a network protocol. The endpoint implementation accepts SPARQL query strings, executes them on the actual RDF data, and returns the results in the correct target format.
- An *application* or *browser*, which accesses RDF data by issuing SPARQL queries to the endpoint, and interprets the results[1]. Just as it is the case with applications that build upon relational databases, all details of generating results and processing updates are delegated to the SPARQL endpoint. The only defined interface between the application and the data set is the SPARQL language and its transport protocol [9].

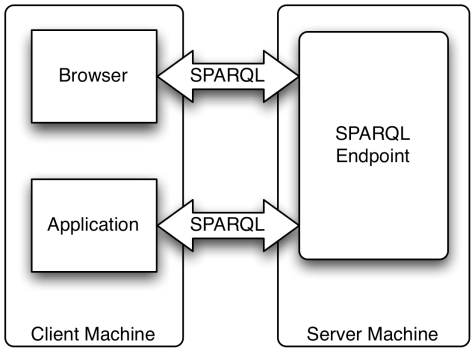


**Figure 1:** Typical Architecture of Semantic Web-based Applications

Naturally, our proposed replication and synchronization mechanisms are beneficial only in situations where these components are distributed over different physical machines and the network link between them is potentially unstable (e.g., when the SPARQL endpoint resides on a company server, while the application is executed on an employee's mobile device).

To introduce a replication and synchronization layer into such a semantic application, it is not necessary to modify any of the existing system components. Instead, we introduce two new components that serve as mediator layer between the client application and the SPARQL endpoint. We denote these components the client-side *replication engine* and the server-side *replication manager*. This extended system architecture is depicted in Fig. 2 and described in the following.

**Replication Engine**. The replication engine is instantiated on the client machine and acts as a transparent proxy for applications. The only change

[1] We assume that update functionality will be included into SPARQL in the near future; the current effort towards this direction has been subsumed by a corresponding W3C member submission, cf. `http://www.w3.org/Submission/2008/SUBM-SPARQL-Update-20080715/`.

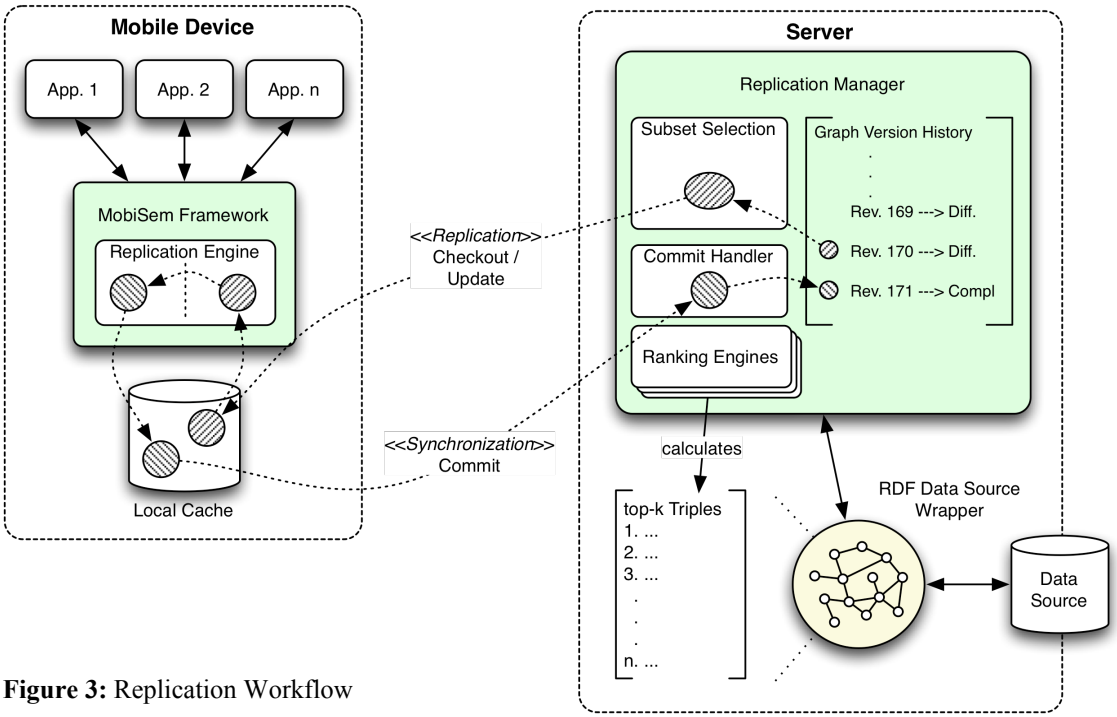


**Figure 3:** Replication Workflow

to applications is a configuration modification: applications must be re-configured to query the local SPARQL endpoint instead of the original remote endpoint.

The replication engine is a fully-functional SPARQL endpoint that is able to process queries and return the results to the application. It is configured to establish a connection to the original SPARQL endpoint, as well as to a corresponding replication manager. It has two operation modes, *online* and *offline* mode. In online mode all queries are directly passed to the original (remote) SPARQL endpoint, and results from the endpoint are forwarded to the application where the request originated. In offline mode the replication engine answers queries from its local cache, which holds a subset of the original data set. The virtual endpoint is hence enabled to return at least partial results for application queries, which is a significant improvement compared to situations where no data can be retrieved at all.

Updates are processed in a similar manner: in online mode they are forwarded to both the local cache and the original data base, while in offline mode changes are recorded on the mobile device for subsequent synchronization between the cached copy and the original data set.

**Replication Manager**. The task of the replication manager is to compute a ranking for the selective replication; i.e., it determines which subset of the data is to be replicated on the client. To accomplish this it needs access to the whole RDF data set, which can in general be achieved through the SPARQL endpoint. In order to yield better performance, it may however be necessary to integrate these two components more tightly, as SPARQL can not be used to notify the manager about data updates. The degree of such an integration is subject of further research.

**Replication Control Protocol**. The replication manager and the replication engine exchange information about the current status of the original endpoint and the client's cache via a replication control protocol, which is also used to coordinate the execution of data replication tasks. Possible reasons for initiating a new data replication task include the execution of a SPARQL query or a data update on the client machine. The replication control protocol should ensure a maximum of offline data availability in the engine's cache at any time. This strategy is preferred over manual synchronization on demand because it also holds when the network connection is unexpectedly interrupted. Additionally, it enables the client to disconnect at any time, instead of requiring it to start a tedious synchronization procedure before a planned disconnect.

## 3 REPLICATION AND SYNCHRONIZA–TION WORKFLOW

Figure 3 shows the workflow of the MobiSem Replication and Versioning framework. Basically, data are processed within a cycle between the replication manager residing on the server and the replication engine running on the client.

The MRV framework operates on data expressed in RDF; however not all data are available in this form. In this case the system can employ *wrappers*

that convert data from other formats into RDF, either on-the-fly or by buffering them in a designated triple store. Such converters are available[2] for a wide variety of data sources, ranging from relational data bases (e.g., D2R [4]) to OAI-PMH [15]. Once such a conversion has been taken place, a Subversion[3]-like management of changes is applied to the named graphs contained therein. The versioning framework treats named graphs as the subject of versioning; each modification to a named graph creates a new revision. Changesets are, depending on their size, persisted either as diff or as full graph.

The terminology for operations also follows Subversion standards. During a *checkout* a client first replicates data from the server. The client may send context information describing its current situation to the server, which then performs a ranking of the triples contained in the requested graph (cf. Section 4). The top-*k* triples (where *k* depends on the storage and processing capacity of the requesting client) are taken from the ranked list and are sent to the client, which buffers them in its local triple store.

Now the client is able to work with, and update its local data replica independent of any network connection; i.e., it is enabled to operate in offline mode. In this phase access to the replicated RDF graph is provided by the client-side replication engine, which also tracks modifications to the graph. After reconnection the replication engine switches back to online mode and initiates the *synchronization process*. It contacts the server and merges all changes from the server to its local replica (following Subversion, this process is denoted as *update*). Conflicts that may occur during the update process are always resolved on the client side; if needed, user input can be requested.

After all conflicts have been resolved, the client's changes can be written back to the server, which is called *commit*. During the commit process a new graph revision is created on the server, and the changes are written to the graph history. From now on the new graph revision is used whenever clients issue checkout or update requests.

Processing user-related contextual data provided by the replication engine is another important task and serves as the basis for the selection of RDF subgraphs according to the user's current activities and intentions. We introduce some of these selection strategies in the following chapter.

## 4 SELECTION OF RDF REPLICA SETS

It is not practicable to replicate entire data sets under the restrictions of mobile devices imposed by technical and user-related context. To provide a tradeoff in such situations, we have developed algorithms for selective replication of RDF subgraphs. The goal of these algorithms is to provide a *subjective interest ranking* of RDF triples, where we take into account structural and semantic characteristics of the dataset, as well as user preferences and context information. In the following we describe some of these input parameters in more detail.

1. *Graph Structure and Metrics*. RDF is based on a graph model; therefore, various metrics and analysis algorithms can be applied to it (e.g., degrees of graph nodes). We are currently investigating the applicability of these metrics for deriving conclusions on the relevance of graph elements for offline replication. Such metrics, however, do not take into account the semantics of the RDF model and ontologies [27], which is addressed by the following two information sources, ontology structure and queries.
2. *Ontology Structure and Metrics*. Ontologies are used to express shared conceptualizations between communicating partners. In our work we focus on the Web Ontology Language (OWL) [11], which is one of the standard languages for ontology modeling on the Semantic Web. OWL ontologies consist of three types of elements: classes, individuals, and properties. Their structure as well as the semantics of the relationships between them is expressed using different OWL language constructs, e.g., `subClassOf` or `equivalentProperty`. From the analysis of these expressions we are able to infer information about the importance of instance data that adheres to these ontologies, and to detect redundant data that does not need to be replicated on the client.
3. *Queries*. As described in Section 2, applications usually access RDF data through issuing SPARQL queries. Hence, the structure of these queries as well as the vocabularies used therein are indicators which data are relevant for an application. To exploit this information we will analyze the syntactic and semantic structure of queries (with the help of ontologies, as described before) and draw conclusions regarding the importance of the data sets that these queries are applied to.
4. *User Context*. Context and context-awareness play a critical role in interactive information systems [10,12]. Recent research in this area reveals that the prevailing system-centric view of context-awareness should be replaced by a user-centric view [25]. Intelligent and adaptive RDF subgraph selection must therefore elaborate on the user's tasks and information needs on a semantic level to provide appropriate and

[2] A list of RDF converters is maintained by the World Wide Web Consortium at `http://esw.w3.org/topic/ConverterToRdf`.
[3] Subversion: `http://subversion.tigris.org`

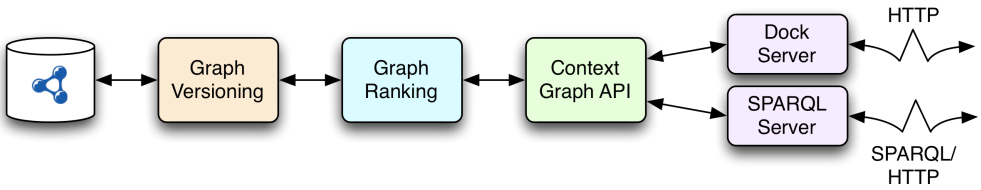


**Figure 4:** MRV Server Processing Pipeline

valuable data. For instance, based on upcoming appointments or events in the user's calendar, the replication engine could infer on the data probably needed. We investigate further approaches on how to utilize user behavior and contextual information to enhance the quality of the data retrieval process.

5. *Explicit User Interest*. The end of the Semantic Web information chain is the human user. In every situation, the user should have the possibility to overrule or supplement automatically replicated datasets. This selection may be carried out on various levels, e.g., using elements from an ontology, using range definitions for attribute values, or even (on the lowest level) the selection of single triples out of the graph. Depending on the user's experience, sophisticated user interfaces are required for this task, especially in cases where the amount of data exceeds certain sizes.

From the analysis of these data it may be possible to derive information that is relevant not only to replication and synchronization, but also for other aspects of the stored data: for instance, the analysis algorithms might reveal that certain parts of a data set are never queried. In this case, it could be advisable to move these parts from the live data store into a long-term archive. On the other hand, analysis of data graphs may evidence that sub-graphs are disconnected, therefore semantic relations between resources are missing. If such a graph is generated from an external data source, this may indicate a potential error in the mapping or in the transformation algorithm.

## 5 IMPLEMENTATION

As a starting point for a reference implementation we have conducted a survey on existing mobile Semantic Web frameworks. We have analyzed two XML parsers for mobile environments, *NanoXML for J2ME*[4] and *kXML*[5], as well as two mobile RDF frameworks, *Mobile RDF*[6] and *µJena*[7]. Our survey revealed that µJena is the most advanced framework providing ontology and inference support, although its API is currently in prototypical status and only allows for processing RDF data serialized in N-Triples format[8]. However, none of the evaluated frameworks supports queries on RDF data via SPARQL or other query languages. A serialization mechanism between RDF and the internal storage mechanisms used by certain mobile devices for storing data permanently could also not be found. Such mechanisms are however needed since many mobile platforms do not use a file system for storing application data, but provide platform-specific storage systems, such as the Record Management System (RMS) in case of J2ME MIDP[9] applications.

**Client.** We have developed an initial version of our framework on the Google Android platform[10] since the underlying operating system and application model provide substantial advantages compared to other mobile operating system architectures. Android itself is an environment for running Java applications on the *Dalvik Virtual Machine*[11] which is especially optimized for mobile environments. It includes *SQLite*, a lightweight and powerful relational database engine, and makes use of some advanced software design patters such as the Model-View-Controller (MVC) pattern to separate application logic from user interface design and underlying data models. Android provides access to the core system operating functions through standard APIs as well as a complete multitasking environment where each application is executed within its own thread, thus providing the possibility to implement background services, like a synchronization process that is automatically activated when the mobile device has online connectivity to its home network (e.g., by automatically establishing a VPN connection within a public wireless local area network).

**Server**. The replication manager is able to process contextual information, such as the number of triples expected by the replication engine, the user's current location, or information about the

[4] NanoXML: `http://sourceforge.net/projects/nanoxml-j2me`
[5] kXML: `http://kxml.sourceforge.net`
[6] Mobile RDF: `http://www.hedenus.de/rdf`
[7] µJena: `http://poseidon.elet.polimi.it/ca/?page_id=59`
[8] N-Triples Syntax for RDF: `http://www.w3.org/TR/rdf-testcases/#ntriples`
[9] Mobile Information Device Profile (MIDP): `http://java.sun.com/products/midp`
[10] Google Android Platform: `http://code.google.com/android`
[11] Dalvik Virtual Machine: `http://www.dalvikvm.com`

serialization formats the client is able to process. Based on this information it selects a subset of the RDF data set and transmits it to the client. An RDF abstraction layer has been introduced in the replication manager (cf. Section 4) so that its implementation is independent from the underlying RDF store. The client locally caches the data and hence makes it available to applications, and changes made to this cache are subsequently forwarded to the replication manager.

We have implemented a processing pipeline (cf. Figure 4) that allows to customize the ranking applied to RDF triples. Each triple in the graph is assigned an absolute rank $r \in \mathbb{R}$, which can be normalized to $r_n$, $-1 \leq r_n \leq 1$. The ranked graph runs through a pipeline of ranking modules, each of which may modify the rank for every triple. A number of ranking modules have been developed so far; a detailed description of their underlying algorithms is out of the scope of this work.

- *RDF Base Vocabulary Ranker.* RDF provides a set of core terms, which are often crucial for correct data processing (e.g., `rdf:type`, which is needed to determine the classes a resource belongs to). This ranker increases the rank of all terms that are in the RDF core vocabulary.
- *Annotation Property Ranker.* A number of RDFS and OWL terms are annotation properties, i.e., they describe aspects of resources that may be less important for search and retrieval (e.g., `owl:versionInfo` or `rdfs:isDefinedBy`). This ranker decreases the rank of triples that contain these terms on the property position.
- *Ontology Ranker.* Based on the assumption that classes and properties from formalized ontologies are more important than terms that are freely chosen, this component increases the rank of triples whose predicate and object conform to a given formalized ontology.
- *Resource Ranker.* This component ranks triples using an algorithm similar to PageRank [6] where the rank of a resource (and the triples it participates in) depends on the rank of resources that refer to this resource.
- *Information Gain Ranker.* This component ranks triples based on the information entropy of each property found in the graph.
- *SPARQL Ranker.* This component analyzes SPARQL [21] queries that have been issued against the data set, and increases the rank of those triples that are required to correctly and completely evaluate the query.

An initial analysis of the results of these rankers indicates that there exists no generically applicable strategy for ranking triples, since for this task we must always consider the requirements of the concrete application scenario. However, with our architecture and a number of basis ranker components we intend to provide a generic framework that can be customized and extended to fit concrete needs.

**Preliminary Results.** In a tentatively conducted evaluation we analyzed the performance of processing RDF on a Google Android G1 device, which was the first physically available Android device on the market. It represents the latest generation of mobile devices in terms of technical capabilities such as processing power, memory capacity, screen size and resolution, etc. The G1 is equipped with a 32-bit Qualcomm MSM7201A RISC CPU running at a nominal clock speed of 528 MHz[12]. All tests were performed with the standard memory capacity of 192 MB.

For a first performance evaluation, we measured the processing times needed for building and maintaining RDF documents of different sizes. Every test has been repeated ten times for each RDF document, all containing a different amount of triples. First results show that processing times were rather low (<200ms) when reading and parsing small RDF documents containing a few hundred triples. Accessing specific RDF elements such as properties, subjects, or entire statements within an RDF document also performs reasonably well: accessing a specific statement in an RDF document containing 129 triples took around 50ms in average. However, the overall performance of the system significantly decreased when larger RDF documents consisting of several hundred triples have been processed. According to our observations, a doubling in the amount of triples causes approximately 4 times longer processing times. The processing times we have observed are currently far from being acceptable in real-world scenarios where replication tasks are commonly performed with much larger documents. Especially for the efficient provision of larger RDF data sets, the efficiency of the underlying RDF processing framework needs to be significantly improved.

First investigations revealed that the performance drops have been caused by the internal data structures used by the μJena framework, which need to be adapted to the Android environment and operating system specifics. Provisional adaptation efforts reduced processing times by 50%, whereas a slight increase in memory consumption was observed. In future work, we will concentrate on tailoring the μJena framework towards the operating specifics of the Android platform so that building and maintaining realistic amount of triples occurs within reasonable amounts of time.

[12] Due to battery saving reasons, the G1's CPU runs at a clock speed of only 350 MHz.

## 6 RELATED WORK

Although RDF databases are gaining industry strength in terms of performance and memory efficiency, mechanisms for synchronization and offline replication can hardly be found. To the best of our knowledge, many of today's state-of-the-art triple stores, such as Jena[13], Sesame[14], and Redland[15], do not include support for (selective) offline replication.

Most of the systems mentioned before can be configured to make use of a relational data base to store RDF data. For this, they employ mapping algorithms in order to represent RDF graphs as relations. One could make use of a RDBMS's replication and synchronization facilities; however, this has two drawbacks: (*1*) they do not consider the special aspects of RDF and semantic graphs, including ontologies, and (*2*) performing selective replication is difficult unless the developer analyzes the exact mapping algorithms for the target system. Usually, those systems do not provide facilities to elaborate on the meaningfulness and semantics of RDF data sets. Larger-scale database systems like OpenLink Virtuoso [13], Oracle [1], and OpenAnzo[16] do not solely focus on RDF but may serve as a data integration point for different sources, including RDF. While these systems often provide support for replication and synchronization, they are not designed to be deployed on mobile devices.

A different approach for selective distribution and replication of RDF data is the Peer-to-Peer (P2P) paradigm, where multiple equal systems exchange data over a network. Such systems (e.g., Edutella [19] and RDFPeers [7]) provide valuable knowledge about efficient distribution and exchange of RDF data, but do not focus on selective replication. Tumarrello et al. [26] describe an algorithm for selective exchange of RDF data, based on P2P systems. We aim to extend the results presented by them and apply them to non-P2P environments.

The Open Mobile Alliance (OMA) provides the SyncML framework for data synchronization [20], which allows data of different kinds (including contacts, calendars, and e-mail messages) to be synchronized between devices. The framework also specifies a number of bindings to protocols that are commonly used in the context of mobile devices, as well as limited means to express device context information, e.g., the available memory or the supported databases. Since this framework does not consider a generic data format like RDF, we will analyze potential synergies and links between our approach and the OMA activities.

[13] Jena Semantic Web Framework: `http://jena.sourceforge.net`
[14] Sesame Framework: `http://www.openrdf.org`
[15] Redland RDF Libraries: `http://librdf.org`
[16] OpenAnzo: `http://www.openanzo.org`

## 7 CONCLUSIONS

In this paper we have outlined the MobiSem Replication and Versioning framework for selective replication of RDF data sets to mobile devices. The goal of this framework is to provide access to RDF data sets in situations where there is no network connectivity available and hence communication with remote data sources is impossible. Our proposed architecture extends current Semantic Web applications with intermediate components that handle SPARQL queries transparently, either by forwarding them to the actual data store if connectivity is up, or by answering them from a locally cached partial replica of the data set on the mobile device, if there is no connectivity. We have discussed our proposed architecture and its associated workflow, as well as an initial reference implementation together with preliminary evaluation results.

We have already implemented a number of ranking components that select RDF triples based on various criteria (including graph metrics and knowledge from formal ontologies), and we are currently in the process of evaluating and extending their underlying algorithms so that they can be applied to real-world scenarios and application requirements. In parallel we are working on further improvements of the client-side part of our framework; namely the optimization of processing and storing larger RDF graphs on mobile devices.

*Acknowledgements.* Parts of this work have been funded by FIT-IT grants 812513 and 815133 from Austrian Federal Ministry of Transport, Innovation, and Technology.

## REFERENCES

[1] Omar Alonso, Sandeepan Banerjee, and Mark Drake. GIO: *A Semantic Web Application Using the Iinformation Grid Framework*. In Proceedings of the 15th international conference on World Wide Web, pages 857–858, New York, NY, USA, 2006. ACM.

[2] T. Berners-Lee, J. Hendler, and O. Lassila. *The Semantic Web*. Scientific American Magazine, 284(5):34–43, 2001.

[3] Chris Bizer, Richard Cyganiak, and Tom Heath. *How to Publish Linked Data on the Web*, 2007. Available at http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedData-Tutorial/, retrieved 02-Dec-2008.

[4] Chris Bizer and Andy Seaborne. D2RQ - *Treating Non-RDF Databases as Virtual RDF Graphs*. Poster at the 3rd International Semantic Web Conference (ISWC2004), 2004.

[5] Dan Brickley and R.V. Guha. *RDF Vocabulary*

*Description Language 1.0: RDF Schema* (W3C Recommendation 10 Februar 2004). World Wide Web Consortium, 2004.

[6] Sergey Brin and Larry Page. *The Anatomy of a Large-Scale Hypertextual Web Search Engine*. In Seventh International World-Wide Web Conference (WWW 1998), 1998.

[7] Min Cai and Martin Frank. *RDFPeers: A Scalable Distributed RDF Repository Based on a Structured Peer-to-peer Network*. In WWW '04: Proceedings of the 13th international conference on World Wide Web, pages 650–657, New York, NY, USA, 2004. ACM Press.

[8] Chia-Hui Chang, Mohammed Kayed, Moheb Ramzy Girgis, and Khaled F. Shaalan. *A Survey of Web Information Extraction Systems*. IEEE Transactions on Knowledge and Data Engineering, 18(10):1411–1428, 2006.

[9] Kendall Grant Clark, Lee Feigenbaum, and Elias Torres. *SPARQL Protocol for RDF* (W3C Recommendation 15 January 2008). World Wide Web Consortium, 2008.

[10] Joëlle Coutaz, James L. Crowley, Simon Dobson, and David Garlan. *Context is Key*. Commun. ACM, 48(3):49–53, 2005.

[11] Mike Dean and Guus Schreiber. *OWL Web Ontology Language Reference* (W3C Recommendation 10 February 2004). World Wide Web Consortium, February 2004. Available at http://www.w3.org/TR/owl- ref/.

[12] Paul Dourish. *What We Talk About When We Talk About Context*. Personal Ubiquitous Comput., 8(1):19–30, 2004.

[13] Orri Erling and Ivan Mikhailov. *RDF Support in the Virtuoso DBMS*. In Sören Auer, Christian Bizer, Claudia Müller, and Anna V. Zhdanova, editors, CSSW, volume 113 of LNI, pages 59–68. GI, 2007.

[14] Tudor Groza, Siegfried Handschuh, Knud Moeller, Gunnar Grimnes, Leo Sauermann, Enrico Minack, Cedric Mesnage, Mehdi Jazayeri, Gerald Reif, and Rosa Gudjonsdottir. *The NEPOMUK Project — On the Way to the Social Semantic Desktop*. In Proceedings of I-Semantics'07, pages pp. 201–211. JUCS, 2007.

[15] Bernhard Haslhofer and Bernhard Schandl. *The OAI2LOD Server: Exposing OAI-PMH Metadata as Linked Data*. In International Workshop on Linked Data on the Web (LDOW2008), 2008.

[16] Patrick Hayes. *RDF Semantics* (W3C Recommendation 10 February 2004). World Wide Web Consortium, 2004.

[17] Ian Jacobs and Norman Walsh. *Architecture of the World Wide Web, Volume One* (W3C Recommendation 15 December 2004). World Wide Web Consortium, 2005. Available at http://www.w3.org/TR/webarch/.

[18] David R. Karger. Haystack: *Per-User Information Environments Based on Semistructured Data*. In Victor Kaptelinin and Mary Czerwinski, editors, Beyond the Desktop Metaphor, pages 49–100. Massachusetts Institute of Technology, 2007.

[19] Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjörn Naeve, Mikael Nilsson, Matthias Palmér, and Tore Risch. *EDUTELLA: A P2P Networking Infrastructure Based on RDF*. In WWW '02: Proceedings of the 11th international conference on World Wide Web, pages 604–615, New York, NY, USA, 2002. ACM Press.

[20] Open Mobile Alliance. *OMA Data Synchronization V1.2.1*, 2007. Available at http://www.openmobilealliance.org/Technical/release_program/ds_v12.aspx.

[21] Eric Prud'hommeaux and Andy Seaborne. *SPARQL Query Language for RDF* (W3C Recommendation 15 January 2008). World Wide Web Consortium, 2008.

[22] Leo Sauermann, Ansgar Bernardi, and Andreas Dengel. *Overview and Outlook on the Semantic Desktop*. In Proceedings of the 1st Semantic Desktop Workshop, volume 175, Galway, Ireland, November 2005. CEUR Workshop Proceedings.

[23] Bernhard Schandl. *SemDAV: A File Exchange Protocol for the Semantic Desktop*. In Proceedings of the Semantic Desktop and Social Semantic Collaboration Workshop, volume 202, Athens, GA, USA, November 2006. CEUR Workshop Proceedings.

[24] Bernhard Schandl and Stefan Zander. *A Framework for Adaptive RDF Graph Replication for Mobile Semantic Web Applications.* In Proceedings of the JointWorkshop on Advanced Technologies and Techniques for Enterprise Information Systems (Session on Managing Data with Mobile Devices), Milano, Italy, 2009.

[25] Hong-Siang Teo. *An Activity-driven Model for Context-awareness in Mobile Computing*. In MobileHCI '08: Proc. of the 10th international conference on Human computer interaction with mobile devices and services, pages 545–546, New York, NY, USA, 2008. ACM.

[26] Giovanni Tummarello, Christian Morbidoni, Joackin Petersson, Paolo Puliti, and Francesco Piazza. *RDFGrowth, a P2P Annotation Exchange Algorithm for Scalable Semantic Web Applications*. In Ilya Zaihrayeu and Matteo Bonifacio, editors, P2PKM, volume 108 of CEUR Workshop Proceedings. CEUR-WS.org, 2004.

[27] Denny Vrandecic and York Sure. *How to Design Better Ontology Metrics*. In Proceedings of the 4th European Semantic Web Conference (ESWC2007), 2007.