# Mobile Co-Authoring of Linked Data in the Cloud

Moulay Driss Mechaoui[1(✉)], Nadir Guetmi[2(✉)], and Abdessamad Imine[3(✉)]

[1] University of Sciences and Technology Oran 'Mohamed Boudiaf' USTO-MB, Oran, Algeria
moulaydriss.mechaoui@univ-usto.dz
[2] LIAS/ISAE-ENSMA, Poitiers University, Chasseneuil, France
nadir.guetmi@ensma.fr
[3] Université de Lorraine and INRIA-LORIA Grand Est, Nancy, France
abdessamad.imine@loria.fr

**Abstract.** The powerful evolution of hardware, software and data connectivity of mobile devices (such as smartphones and tablets) stimulates people to publish and share their personal data (like social network information or sensor readings) independently of spatial and temporal constraints. To do this, the development of an efficient semantic web collaborative editor for mobile devices is needed. However, collaboratively editing a shared semantic web document in real-time through ad-hoc peer-to-peer mobile networks requires increasing amounts of computation, data storage and network communication. In this paper, we propose a new cloud service-based model that allows a real-time co-authoring of Linked-Data (LD) as a RDF (Resource Description Framework) graph using mobile devices. Our model is built upon two layers: (i) cloning engine that enables users to clone their mobile devices in the cloud to delegate the overload of collaborative tasks and provides peer-to-peer networks where users can create ad-hoc groups; (ii) Collaborative engine that allows updating freely and concurrently a shared RDF graph in peer-to-peer fashion without requiring a central server. This work represents a step forward toward a practical and flexible co-authoring environment for LD.

**Keywords:** Linked data · RDF · Mobile collaboration · Mobile cloud computing

## 1 Introduction

The massive development of structured data on semantic web and their growing utilization require high availability. Thus, the replication of semantic web data enables services and applications to run independently of network connection quality and central server availability. Recently, and due to the rapid development of mobile devices (e.g. smartphones and tablets), users can download semantic data from central servers and process/query it locally on their mobile

© Springer International Publishing Switzerland 2015
T. Morzy et al. (Eds): ADBIS 2015, CCIS 539, pp. 371–381, 2015.
DOI: 10.1007/978-3-319-23201-0_38

devices. This replication leads to the distribution of the computation among a large number of mobile devices, and accordingly, a great level of scalability can be achieved [13]. In addition, users can process semantic data (e.g. their personal data like social network information or sensor readings) even in off-line. Several mobile applications such as DBpedia Mobile [2], HDTourist [9] and RDF On the Go [13] that allow using RDF (Resource Description Framework) documents in mobile devices. However, the mobile user is *passive* in such applications since she/he stores RDF graphs and interrogates them using locally SPARQL queries. But, if the mobile user goes *active* (i.e. she/he modifies the local semantic data): what about updating and synchronizing copies of RDF graphs?

Updating and synchronizing semantic web data is a serious problem that has been raised by Berners-Lee in [17]. Linked Data (LD) enables semantic web data to be created in online mode and to be accessible to a large public; it allows replacing collections of offline RDF data [17]. The goal of LD is to enable people to share structured data on the web as easily as they can share documents today. It uses RDF technology that (i) relies on HTTP URIs to denote things; (ii) provides useful information about a thing at that things URI; and (iii) includes in that information other URIs of LD. Tabulator [3] is a LD browser, designed to provide the ability to navigate the web of linked things. In [17], Berners-Lee et al. raise some interesting challenges when adding collaborative co-authoring mode in Tabulator. This mode consists in collaboratively editing the LD which is represented by a RDF graph.

Using mobile environments for supporting collaborative editing of LD is not without problems. Indeed, mobile devices are resource-poor despite their continuous development (less secure, unstable connectivity, and constrained energy). Consequently, computation on mobile devices will always involve a compromise [16]. For example, peer-to-peer (P2P) collaborative editing using mobile devices is often costly since it requires important energy consumption in order to synchronize multiple copies of shared data to preserve consistency and handle the group scalability (i.e. with join/leave events). Furthermore, ensuring a continuous collaboration with frequent disconnections is impossible.

To attenuate the resource limitation of mobile devices, one simple solution is to harness cloud computing, that is an emerged model based on virtualization for efficient and flexible use of hardware assets and software services over a network. It extends the mobile device resources by offloading execution from the mobile to the cloud where a *clone* (or *virtual machine*) of the mobile is running. Cloud computing allows building Virtual Private Networks (VPN) such as peer-to-peer where a mobile device can be continuously connected to other mobiles to achieve a common task.

**Contribution.** In this paper, we present a model aimed at the combination of mobile and cloud environments where the management of real-time collaborative editing service plays central role. Our objective is to provide computer support for modifying concurrently shared RDF documents by dispersed mobile users. To improve availability of data, each user has a local copy of the shared RDF documents. The collaboration is performed as follows: the updates of each user

are locally executed in non-blocking manner and then are propagated to other users in order to be executed on other copies. Our model is a two levels system. The first level provides self-protocol to create clones of mobiles, manage users groups and recover failed clones in the cloud. The second level provides the group collaboration mechanisms in real-time, without any role assigned to the server. To illustrate our proposed model, we give the following use case:

*Assisting Conference Attendees.* Suppose after attending the ADBIS 2015 conference, a group of participants want to make a tour in Poitiers city. They are already provided with mobile application based on our proposed model helping them to visit the city using a map. One of the participants creates a collaborative group in the cloud and downloads the RDF document with relevant information about Poitiers city from DBPedia[1]. The other members join the created group and share the Poitiers RDF document. During the journey one of them realizes that, instead of the restaurant appearing on his map, there is a bookshop. Hence, she/he corrects/updates the description of this localization in his map, and then he sends it to his clone in the cloud in order to be broadcast to other group members. At the meanwhile, one participant was disconnected when sending the update information. In this case, her/his clone will notify the update information after her/his re-connection to the group as if she/he did not quit it. Moreover, when visiting museums and restaurants, participants can share in real-time their thoughts by writing their reviews in the local RDF document.

**Outline.** The remainder of this paper is organized as follows: our model description is given in Section 2. Section 3 presents the cloning engine. In Section 4, we describe our collaborative editing protocols for consistency maintenance of shared RDF graph. We discuss the related work in Section 5 and conclude in Section 6.

## 2 Model Description

We propose a new collaboration model for manipulating semantic web data, regardless of spatial and temporal constraints, where mobile users can edit collaboratively shared RDF documents in peer-to-peer mode. The advantages of our model are (i) the availability of semantic web data anytime and anywhere, and (ii) the optimal use of mobile devices resources. In fact, the collaboration and communication tasks are seamlessly turned on the cloud.

As illustrated in Figure 1, our model consists of two levels. The first level provides self-protocol to create clones of mobiles, manages users groups and recover failed clones in the cloud. It is based on a cloning engine protocol that (i) instantiates clones for mobile devices, (ii) builds virtual peer-to-peer networks across collaborative groups, (iii) manages seamlessly the join and the leave of clones inside the groups, and (iv) creates a new instance of a clone when a failure appears.

[1] http://fr.dbpedia.org/page/Poitiers

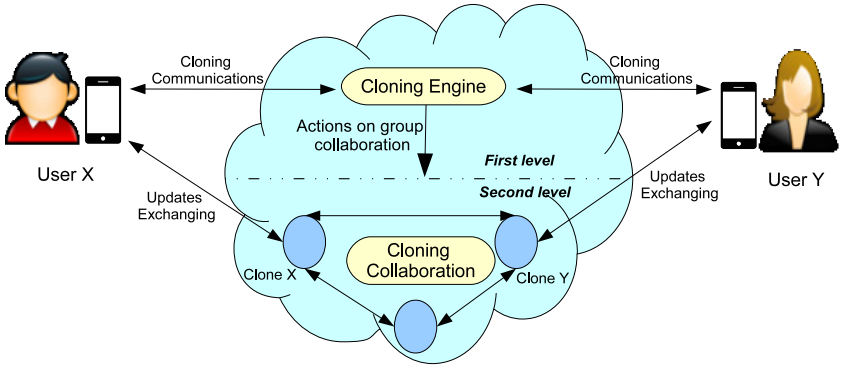


**Fig. 1.** Our model presentation.

The second level provides the group collaboration mechanisms in real-time, without any role assigned to the server. It provides a pure *peer-to-peer virtual private network* platform where users can form ad-hoc groups based on their clones to achieve a common objective. This platform is equipped with mechanisms to transparently manage the user departure, the arrival of new users joining the group and the collaboration spots. Our mobile co-authoring level allows users to cooperate as follows: each user has a local copy of the shared RDF document in her/his mobile and another copy of the same document in the clone; the user's updates are locally executed in mobile device and then are sent to its clone in order to be executed on other mobile devices (via their clones). It should be noted that concurrent execution of updates may lead to document inconsistency. Therefore, to maintain consistency, two synchronization protocols are proposed for clone-to-clone and mobile-to-clone interactions, respectively.

In the sequel, both levels of our model (see Figure 1) will described in Sections 3 and 4, respectively.

## 3 Cloning Engine

Our cloning engine involves a set of mobile users and a set of clones (or virtual machines) in such a way each mobile user owns her/his clone in the cloud. The clone has its own machine features (e.g. CPU frequency, memory size) and a virtual image to be set up (e.g. softwares like the operating system and synchronization protocols).

A set of web services is deployed to manage user groups and create clones for mobiles in the cloud. To deal with server failure and/or migration to a new cloud provider, clones are equipped with a pre-configuration to get a fast redeployment. In this case, the web services play a crucial role to ensure the continuity of clone-mobile connection (and *a fortiori* the continuity of collaboration between mobile users). The cloning engine is therefore responsible of the smooth running of clones; its main role is to:

**Instantiate Clones for Mobile Devices:** A new clone is built for a mobile user when she/he registers for the first time. The cloning process consists of creating a new virtual machine (e.g. x86 Android) as well as its configuration and autostart. This process is presented as follows:

- *Saving clone parameters*: It saves information related to the user and its clone (e.g. username, password, email, new clone identifier and IP address). During this step, the user can create a new group or join an existing one.
- *Clone startup and initialization*: A new virtual machine (e.g. Android x86[2]) is created for each mobile device. When the clone starts (i.e. the clone boot), it gets its IP address from a host server and calls the web service to receive the required data for any collaboration, such as clone identifier, group identifier, multicast address of the group and the IP address of the mobile. This data is important for the communication and collaboration of mobile-to-clone and clone-to-clone.

**Build P2P VPNs Across Collaborative Groups:** it consists of (a) assigning a new identifier and a multicast address to a new group, (b) building a new VPN for clones to communicate between them in the group, and (c) managing seamlessly the join and leave of clones inside the groups.

**Recover Clone State After Failure:** It supports many failure situations (clone, host server and network failure) and allows any failed clone to restore its consistent state and re-join its collaborative group. For instance, in the case of the host server failure, the clone calls automatically the web service using its initial settings to provide the new IP address of the new host server after migrating.

## 4 Collaborative Engine

Our collaborative engines manages all concurrent access for editing collaboratively a shared RDF graph and preserving the consistency of all copies. In this section, we describe how to map an RDF graph into a sequence data structure and we present our concurrency control procedures.

### 4.1 Mapping RDF to a Sequence

On the web, any information published about resources of LD is represented using the RDF. Each expression in RDF is a collection of triples, a triple consists of a *subject*, a *predicate* (also called property) and an *object*. The subject of a triple is the URI describing resource. The object can either be a simple literal value (e.g.,a string, a number) or the URI of another resource. The predicate indicates what kind of relation exists between subject and object. The predicate is a URI too. A set of such triples is called an RDF graph. This can be illustrated

[2] http://www.android-x86.org/

by a node and directed-arc diagram, in which each triple is represented as a node-arc-node link.

Since a set is implemented by means of a list, then we can use operations such as insert and delete to edit a shared list. Thus, we can reuse the state-of-the-art of collaborative editing systems [10]. For instance, the following group of statements "there is a Person" identified by http://www.w3.org/People/EM/contact#me (this example is taken from [8]):

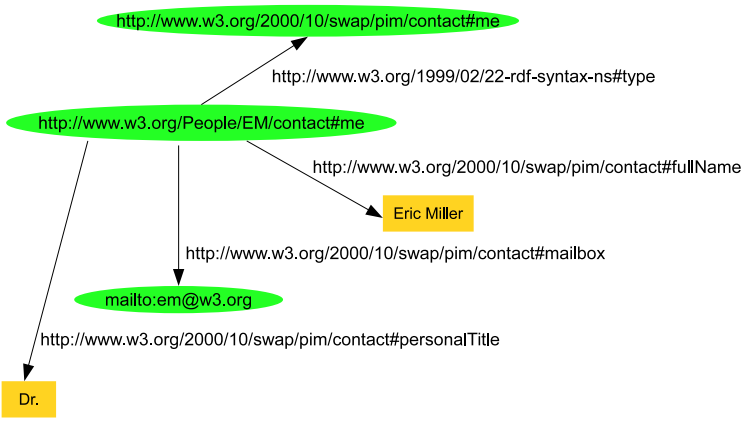


**Fig. 2.** An RDF Graph

- http://www.w3.org/People/EM/contact#me has a name whose value is Eric Miller
- http://www.w3.org/People/EM/contact#me has a title whose value is Dr
- http://www.w3.org/People/EM/contact#me has a email address whose value is em@w3.org

An RDF graph can be serialized into a sequence of triples and considered as a text where each line corresponds to a simple triple of subject, predicate and object. For example, the first statement shown in Figure 2 would be written as a text line (subject, predicate, object):

```
http://www.w3.org/People/EM/contact\#me
   http://www.w3.org/2000/10/swap/pim/contact\#fullName "Eric Miller"
```

Since we consider an RDF graph as a sequence, then each triple is addressed simply by a position within the sequence. Therefore, we assume that the sequence of triples can be modified by the following primitive operations: $Ins(p, T)$ which adds triple $T$ at position $p$ and $Del(p, T)$ which deletes triple $T$ at position $p$. Updating a triple (*e.g.*, by modifying the predicate URI) can be expressed by a sequence of delete (by removing the old triple) and insert (by adding the new one) operations.

### 4.2 Synchronization Protocols

We inspired from Optic [10] that allows managing collaborative editing works in P2P environments. We have designed a new protocol for coordinating the same works in the cloud. Our protocol supports an unconstrained editing work. Using optimistic replication scheme (shared RDF documents are replicated at mobiles and their clones), it provides simultaneous access to shared RDF documents. Using operational transformation approach for maintaining consistency, reconciliation of divergent copies is done automatically in decentralized fashion (i.e., without the necessity of central synchronization). Moreover, it supports dynamic groups where users can leave and join at any time. There are two layers in our synchronization protocol: the first layer ensures synchronization between clones and the second one consists in synchronizing the mobile with its clone. Figure 3 presents a general view of our proposal model.

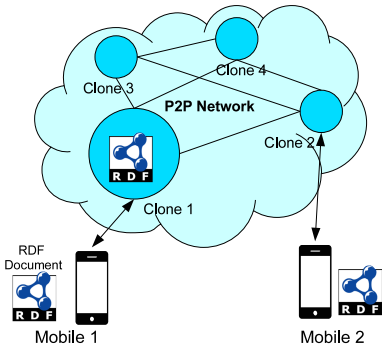


**Fig. 3.** Scenario of collaboration.

It runs in split mode: the light computing task is executed on mobile device (as front-end) and the heavy computing task is performed on the clone (as back-end) in order to delegate the maximum consuming-resources tasks (such as operational transformation and communication) to the cloud.

**Clone-Clone Synchronization.** Each clone has a local copy of RDF graph and a unique identity. It generates operations sequentially and stores these operations in a buffer called a *log*. When a clone receives a remote operation $O$, the integration of this operation proceeds by the following steps:

1. from the local log it determines the sequence *seq* of operations that are concurrent to $O$;
2. it calls the transformation component in order to get operation $O'$ that is the transformation of $O$ according to *seq*;
3. it executes $O'$ on the current state;
4. it adds $O'$ to the local log.

*Illustrative Example.* Given two clones, Clone 1 and Clone 2 editing a shared RDF graph as described in Figure 4. Initially, each Clone has an empty copy. Two local insertion operations $O_1$ and $O_2$ have been executed by Clone 1. Concurrently, Clone 2 has executed another insertion operation $O_3$. The added triples $T_1$, $T_2$ and $T_3$ are respectively as follows (where `UR1` is http://www.w3.org/People/EM/contact#me and UR2 is mailto:em@w3.org):

```
<UR1> <http://www.w3.org/2000/10/swap/pim/contact#fullName> "Eric Miller".
<UR1> <http://www.w3.org/2000/10/swap/pim/contact#mailbox> <UR2>.
<UR1>  <http://www.w3.org/2000/10/swap/pim/contact#personalTitle> "Dr".
```

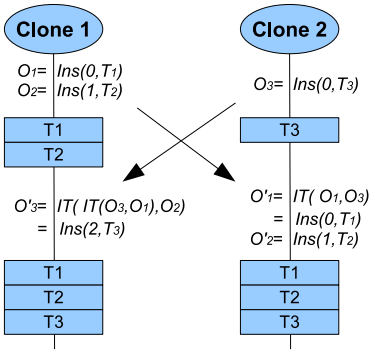


**Fig. 4.** Scenario of collaboration.

There is a dependency relation between operations $O_1$ and $O_2$ in such a way $O_1$ must be executed before $O_2$ in all clones. This is due to the fact that their added triples are adjacent (positions 0 and 1) and created by the same clone (from more details see [10]). This dependency relation is minimal in the sense that when $O_2$ is broadcast to all clones it holds only the identity of $O_1$ as it depends on directly. As illustrated in Figure 4, the execution order is as follows.

At Clone 1, $O_3$ is considered as concurrent. It is then transformed against $O_1$ and $O_2$. The sequence [ $O_1$ = Ins(0,$T_1$), $O_2$ = Ins(1,$T_2$), $O'_3$ = Ins(2,$T_3$)] is executed and logged in Clone 1, where $O'_3$ results from transforming $O_3$ to include the effect of operations $O_1$ and $O_2$ (i.e. $O'_3$ = IT(IT($O_3$,$O_1$),$O_2$)=Ins(2,$T_3$) using transformation function $IT$ given in [10]).

At Clone 2, $O_1$ and $O_2$ are concurrent with respect to $O_3$. They must be transformed before to be executed after $O_3$ according to their dependency relation. Thus, the following sequence is executed and logged in Clone 2: $O_3$ = Ins(0,$T_3$) and $O'_1$ = IT($O_1$,$O_3$) = $O_1$ and $O'_2$ = $O_2$.

**Clone-Mobile Synchronization.** Mobile and its clone are two entities that are physically separated. Therefore, a delay of applying the same operations on both sides with the same state is possible. This may lead to document inconsistency.

To overcome this problem, we propose a solution based on distributed mutual exclusion between the mobile and its clone. The shared RDF document will be considered as a critical section (CS) when the mobile tries to commit/synchronize w.r.t its clone. Only one of them will have the exclusive right to access in synchronizing mode to its document copy. This distributed mutual exclusion protocol is achieved by the exchange of messages (i.e., token). Initially, the mobile device has the right to be the first to commit/synchronize with its clone. Whatever where the exclusive access right is, the mobile device and its clone can edit independently their local copies. The clone continues to receive remote operations from other clones to integrate them later on the local state. At the meanwhile, the mobile user can work on his copy in unconstrained way. But, once she/he decides to synchronize with its clone, all local editing operations are sent to its clone and the exclusive access right is released to enable the clone to start the synchronization with the mobile device.

## 5 Related work

In this section, we review the most important works done in the context of our proposal based on the main facets: (i) P2P Semantic Web replication, (ii) Mobile Semantic Data Replication and (iii) Cloud Collaborative Editing.

*P2P Semantic Web Replication.* Except C-Set [1] that is a CRDT (Commutative Replicated Data Types) designed to be integrated within a semantic store in order to provide P2P synchronisation of autonomous semantic store, all previous work on replication in semantic P2P systems such Delta [17], RDFGrowth [18] and RDFPeers [4] are focused on sharing RDF resources. However, C-Set is not suitable for mobile devices since it incurs some overhead and does not consider directly a set as a list (or a sequence).

*Mobile Semantic Data Replication.* The replication of semantic data (as RDF) in mobile devices allows services and applications to operate independently of the network connection quality. Currently, several mobile applications make use of semantic stored in mobile devices [2,5,13,15]. However, mobile devices cannot use natively semantic datasets due to their limited resources. In [19] authors propose a framework based on contextual information and user description to selectively replicate data from external datasets in order to reduce the amounts of replicated data in the mobile device. This idea is interesting for our model since it allows updating just a subgraph of the shared RDF graph.

*Cloud Collaborative Editing.* To overcome resource constraints on mobile devices, several approaches have been proposed to offload parts of resource-intensive tasks to the cloud. Since execution in the cloud is considerably faster than the one on mobile devices and mobile-cloud offloading mechanisms delegate heavy mobile computation to the cloud. Therefore, it extends the battery life and speeds up the application [7,8,14]. CloneDoc [12] enables collaboration for mobile devices that are cloned in the cloud in order to alleviate the burden of collaborative editing works on mobile devices. CloneDoc is implemented upon C2C platform [11].

It is based on Operational Transformation (OT) approach and a single server to enforce a continuous and global order to avoid the divergence of client's document view from the server. Unfortunately, a server failure could stop the collaboration between mobile devices. Moreover, CloneDoc needs additional treatment of OT on the mobile side to ensure convergence between the mobile and its clone. However, this will cause supplementary energy consumption.

## 6 Conclusion

In this paper, we have presented a step forward toward a practical linked data co-authoring system combining mobile and cloud environments. Our cloud-based service enables mobile users to benefit of a huge computing power and data storage of the cloud and manages an efficient and scalable real-time editing tasks. Our service is a two levels system: (i) a cloning engine which manages the complete life cycle of clones (creation, group management and failure recovery); (ii) a collaborative engine for co-authoring shared RDF graphs in fully decentralized way.
As future work, we plan to propose a distributed access control to protect and secure the shared linked data according to the optimistic access control model given in [6].

## References

1. Aslan, K., Molli, P., Skaf-Molli, H., Weiss, S.: C-set: a commutative replicated data type for semantic stores. In: Fourth International Workshop on REsource Discovery RED (2011)
2. Becker, C., Bizer, C.: Dbpedia mobile: a location-enabled linked data browser. In: LDOW (2008)
3. Berners-Lee, T., Hollenbach, J., Lu, K., Presbery, J., Pru d'ommeaux, E., Schraefel, M.: Tabulator redux: writing into the semantic web. In: LDOW (2008)
4. Cai, M., Frank, M.: Rdfpeers: a scalable distributed rdf repository based on a structured peer-to-peer network. In: Proceedings of the 13th International Conference on World Wide Web, pp. 182–197 (2004)
5. Cano, A., Dadzie, A., Hartmann, M.: Whos whoa linked data visualisation tool for mobile environments. In ISWC, pp. 451–455 (2011)
6. Cherif, A., Imine, A., Rusinowitch, M.: Practical access control management for distributed collaborative editors. Pervasive and Mobile Computing **15**, 62–86 (2014)
7. Chun, B.-G., Ihm, S., Maniatis, P., Naik, M., Patti, A.: Clonecloud: elastic execution between mobile device and cloud. In: EuroSys, pp. 301–314 (2011)
8. Cuervo, E., Balasubramanian, A., Cho, D.k., Wolman, A., Saroiu, S., Chandra, R., Bahl, P.: Maui: making smartphones last longer with code offload. In: MobiSys, pp. 49–62 (2010)
9. Hervalejo, E., Martínez-Prieto, M.A., Fernández, J.D., Corcho, Ó: Hdtourist: exploring urban data on android. In: ISWC (2014)
10. Imine, A.: Coordination model for real-time collaborative editors. In: Field, J., Vasconcelos, V.T. (eds.) COORDINATION 2009. LNCS, vol. 5521, pp. 225–246. Springer, Heidelberg (2009)

11. Kosta, S., Perta, V.C., Stefa, J., Hui, P., Mei, A.: Clone2clone (c2c): peer-to-peer networking of smartphones on the cloud. In: HotCloud (2013)
12. Kosta, S., Perta, V.C., Stefa, J., Hui, P., Mei, A.: Clonedoc: exploiting the cloud to leverage secure group collaboration mechanisms for smartphones. In: IEEE INFOCOM, vol. 13 (2013)
13. Le-Phuoc, D., Parreira, J.X., Reynolds, V., Hauswirth, M.: Rdf on the go: an rdf storage and query processor for mobile devices. In: ISWC (2010)
14. Liu, F., Shu, P., Jin, H., Ding, L., Yu, J., Niu, D., Li, B.: Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications. IEEE Wireless Commun. **20**(3), 1–10 (2013)
15. Ostuni, V.C., Gentile, G., Di Noia, T., Mirizzi, R., Romito, D., Di Sciascio, E.: Mobile movie recommendations with linked data. In: Cuzzocrea, A., Kittl, C., Simos, D.E., Weippl, E., Xu, L. (eds.) CD-ARES 2013. LNCS, vol. 8127, pp. 400–415. Springer, Heidelberg (2013)
16. Satyanarayanan, M., Bahl, P., Caceres, R., Davies, N.: The case for vm-based cloudlets in mobile computing. IEEE Pervasive Computing **8**(4), 14–23 (2009)
17. Tim, B.-L., Dan, C.: Delta: an ontology for the distribution of differences between rdf graphs (2004)
18. Tummarello, G., Morbidoni, C., Petersson, J., Puliti, P., Piazza, F.: Rdfgrowth, a p2p annotation exchange algorithm for scalable semantic web applications. In: P2PKM (2004)
19. Zander, S., Schandl, B.: A framework for context-driven rdf data replication on mobile devices. Semantic Web **3**, 131–155 (2012)