IEEE International Conference on Complex Systems, ICCS'12
http://ieeexplore.ieee.org/document/6458550/
DOI: 10.1109/ICoCS.2012.6458550

# B-Set: a synchronization method for distributed semantic stores

Hafed Zarzour *
Department of Computer Science
Mohamed Cherif Messaadia University
41000, Souk-Ahras, Algeria
hafed.zarzour@gmail.com

Mokhtar Sellami
LABGED, Department of Computer Science
Badji Mokhtar University
23000, Annaba, Algeria.
mokhtar.sellami@nasr-dz.org

*Abstract*— **Nowadays, there are increasing interests in developing methods for synchronizing distributed triple-stores by ensuring eventual data consistency in distributed architecture. The most well-known of them have been designed to serve as a common replicated data type (CRDT), where all concurrent operations commute independently of the centralized control. In this context, CRDT has been proposed for semantic stores, such as SWOOKI, C-Set and SU-Set. However none of the exiting synchronization solutions mention how to ensure Causality, Consistency and Intention preservation criteria of CCI model. This paper proposes B-Set, a new CRDT for the synchronization of semantic stores. B-Set is designed not only to ensure convergence of triples replicas but also to preserve user's intentions integrated in distributed architecture. The sets of operations are also defined in order to allow concurrent editing of the same shared triple-stores.**

***Keywords-component: Collaborative editing system, Synchronization, Semantic Stores, CRDT, Consistency.***

## I. INTRODUCTION

The Resource Description Format (RDF) [1,2,3] is a data model used to represent information about World Wide Web resources as a "graph": a set of individual objects, along with a set of connections among those objects. RDF enables us to make assertions and describe resources with triples (subject, predicate, object) that can be viewed as "the subject, verb and object of an elementary sentence". SPARQL/UPDATE [4] reuses a syntax of the SPARQL query language for RDF and defines updating operations of RDF data to update triples from of the target graph. In that role, RDF and SPARQL/UPDATE are one of the pillars of the Semantic Web. The Semantic Web community has developed a set of standards for expressing schemas that enable the creation, exchange and updating of information among communities of different backgrounds.

Collaborative editing systems (CES) of semantic repositories allow two or more participants to work on a shared semantic store regardless of their physical location. There have been many systems developed to support triples-store collaborative editing in various environments, ranging from the ones designed for centralized architecture to the ones designed for distributed architecture. Although the existing CES for semantic stores can support simple editing in centralized architectures, they have shortcomings with regards to satisfying the requirements of concurrent updating for distributed architectures.

A CES for semantic stores is considered as correct and sound if it preserves the CCI model [5] that means Causality, Consistency, and Intention preservation defined as follows:

- Causality: the execution order of all operations is performed in the same way on each copy.
- Convergence: when the system is idle, all copies are identical.
- Intention: The expected effect of a delete and insert operation must be observed on all copies.

Recently, Commutative Replicated Data Types [6] (CRDT) is a promising new category of approaches used to construct operation-based optimistic replication [7] mechanisms. CRDT is designed to ensure eventual consistency of replicated data without complex concurrency control. It has been successfully applied to scalable collaborative editing of textual document, semi-structured data types, tree structure, set type but not yet on semantic stores that take into account the CCI model.

In this paper, we propose B-Set, a new CRDT for synchronizing semantic stores having a set structure. B-Set is designed not only to ensure convergence of triples replicas but also to ensure CCI model criteria integrated in distributed architecture. The sets of operations are also defined in order to allow a concurrent editing of the same shared triples-sores. B-Set integrated with SPARQL/UPDATE allows to DBedia [8] to transit from the extraction of structure information of Wikipedia to the scalable collaborative editing of Wikipedia.

The paper proceeds as follows. Section 2 presents backgrounds and related works. Section 3 details our proposition model for the synchronization of triple-stores based on set structure. Section 4 discusses our approach. Section 5 concludes the paper and points to future works.

## II. RELATED WORK

Centralized semantic stores systems such as RDFStore [9], Jena [10, 9] and Sesame [11] have been implemented to support the storing, sharing and querying of triple stores. These systems are very fast and can scale up to many millions of triples. However, they have the same limitations as other centralized methods, such as impossibility to enable

collaborative editing for maintaining semantic stores and absence of mechanism that supports concurrent operation.

RDFPeers [12] is one of the first efforts for structured peer-to-peer RDF stores. The key idea is to use a MAAN overlay [13] to index a triple three times, once based on the subject, another based on the predicate, and a final based on the object. However, it lacks the ability for supporting collaborative update operations on replicas.

RDFSync [14] is an algorithm for synchronizing a semantic data. Semantic data is defined as RDF graphs where each RDF graph is decomposed unequivocally into minimal subsets of triples and canonically represented by ordered lists of the identifiers. To ensure the synchronization, the difference is performed between the source and the target of the ordered list. However, it is not explicitly specified what happens in the case of concurrent updates on copies.

RDFGrowth [15] proposes a semantic data sharing environment where each peer can only update the shared data or read them. Concurrent operations are integrated by merge algorithms. However, the anatomy of RDFGrowth allows sharing of data but not collaborating.

Edutella [16] presents P2P platform for semantic data based on metadata. Its mechanism focuses on querying RDF metadata stored in distributed RDF stores. A replication service is proposed as complements local storage by replicating in additional peers to achieve metadata persistence / availability and workload balancing while maintaining metadata integrity and consistency. However, they do not mention how to replicate and synchronize metadata.

To managing replica consistency in the context of collaborative editing systems, many algorithms have been proposed focused on Operation Transformation (OT) [17] such as GOTO [18], GOT [5], SOCT2 [19], SOCT4 [20], MOT2 [21]. However, there are no transformation functions for semantic data are available particularly for set structure.

CRDT [6] is a new framework where all concurrent updating operations must commute to ensure convergence. Initially, the algorithm has been successfully applied to different data representations types in scalable collaborative editing for linear data type (text document) [22], tree document structure data type [5] and XML data representation [23].

Recently, many CRDT are proposed to support collaborative editing of semantic stores having set structure.

In [24] authors define many CRDTs having a set structure, Grow Only Set (G-Set), Last Writer Wins Set (LWW-element-Set) and Observed Remove Set (OR-Set). In a G-Set, there is only an insertion operation where each element can be inserted and not deleted from the set. The reconciliation Principe is based on simple set union, since union is commutative. In a LWW-element-Set, A timestamp is attached to each element. If an element is not already exists, a local operation updates its timestamp and adds it to the set and cannot be scalable. In an Observed Remove Set (OR-Set) each element is associated to a set of unique tag. A local add creates a tag for the element and a local remove deletes all the tag of the element. However, G-Set ignores the intention of remove operations, LWW-element-Set is not enable to scale since it uses the tombstone mechanism and OR-Set requires transparent mechanism of unique tag generation between different sites.

Based on OR-Set, SU-Set [25] presents a CRDT for RDF graphs that can take into account the SPARQL 1.1 Upadate specification and can ensures consistency of the data. The main idea of this model is to serve as base for an element CRDT that could be implemented in an existing RDF engine. Since OR-Set considers only insertion and deletion of single elements, it is not possible to apply OR-Set directly to SPARQL Update. Therefore, SU-Set modifies the operations to send the relevant set of triples to affect one by one, but that could flood the network with traffic considering the potential size of an RDF-Graph. However, SU-Set relies on causal delivery of the underlying network, which is challenging and can pose problems in highly dynamic platforms.

C-Set [26] is a data structure defined as CRDT for sets that can be integrated within a semantic store in order to provide P2P synchronization of autonomous semantic store. The main idea of C-set is to assign a counter to each triple of set for tracking how many times a triple t has been added or removed. To this end, four operations are defined on this set. The delete operation del() can performed locally and sends remote delete operation rdel() that is executed remotely. The ins() is an insert operation executed locally. It sends remote insert operation rins() that is executed remotely. However, they do not mention how to ensure the causality and preserve the intention of operations. Although c-set has been designed to ensure consistency, it violates the operations intentions especially when it comes to mutually execute remote delete operations on the same triples that locally have already been removed several times then reinserted.

SWOOKI [27] is P2P semantic wiki that couples two domains: a semantic wikis domain and P2P wikis domain where users can add a semantic annotation in wiki pages. The users of SWOOKI collaborate for writing semantic annotations by editing wiki pages. This system is structured on distributed nodes where each node corresponds to a hosting server that contains replicated pages of semantic wiki. The semantic data are stored in RDF repositories. To edit the triple data, add and remove operations are used for inserting and deleting a RDF triple respectively. The insert operation is interpreted by the fact to increment the counter element that is associated to each triple. When the occurrence of a given triple is equal to zero, it will be permanently removed from a semantic replica. However, when a delete operation is invoked, this solution can fail in ensuring the consistency condition between peers as illustrated in these sequences: [ins(t),del(t),del(t),ins(t)]= {} and [ins(t),del(t),ins(t), del(t)]={t}. This gives a counter-example for triples-stores and implies an inconsistency in both peers in such a way that the first site with a triple and the second with an empty repository.

## III. PROPOSITION

In this section, we define a new CRDT for semantic stores, called B-Set. We define also the set of operations that modify the semantic stores and their effect.

In a collaborative editing system of replicated semantic stores, each user owns a replica of the shared semantic data and each user is responsible for processing all the changes to the replica that it owns. Each participant performs changes to his/her local replica, and then notifies other participants about the updating by broadcasting updates operations through a network. The semantic store that appears at one user's site must be consistent with all other stores that appear at other participants' sites. At the end of collaborative session, all participants see exactly the same triples, thus, all replicated stores that have been executed the same operations set in different order are identical.

### *A. Data model*

In our context, a semantic store includes a set of RDF statements where each RDF statement is represented as the triple (subject, predicate, object) which signifies that the relationship denoted as predicate holds between the concepts denoted as subject and object, where predicate and object are resources or strings. Each triple is associated to boolean variable that can take two values true or false. This boolean variable is used to preserve user's intentions and ensure consistency between different sites that execute concurrent updating operations.

The model of our semantic store, denoted by Ss, can be formalized as a set of a pair such as:

Ss=( opType ,Content ), where opType is used to describe the type of the executed operation. It can have tow values L or R. whilst L means that the operation is locally generated and performed, R means that the operation is remotely executed and has been generated on another site. Content is a couple of <T, B>, where T is a set of all triples added or deleted by users and B is a boolean variable that indicates a state of each triple which corresponds to its visibility. The visibility represents if the triple is visible or not. A triple is never really removed it is just noted as invisible. opType and B are used together to ensure eventual consistency by commutative operations.

The figure 1 shows an example of data structure of B-Set. There are eight triples extracted from ICCS'12 website site where only the fourth and the penultimate triples have been masked.

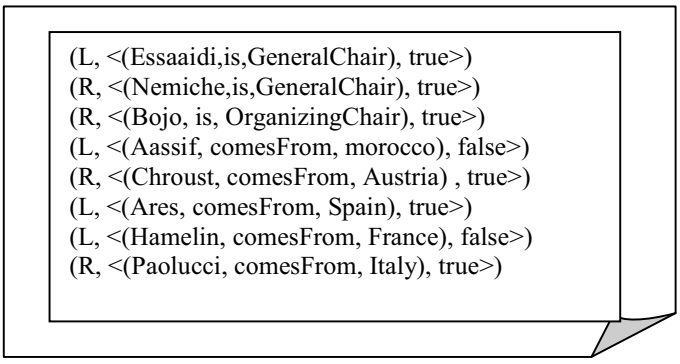


Figure 1. Example of B-Set data structure

### *B. B-Set rules*

On collaborative editing systems participants can modify the data by performing editing operations, such as insert and delete. An update is considered as a delete of old value followed by an insert of a new value.

The operations defined on B-Set data structure are:

- L-Ins(t): is a local insert operation generated and executed on the same site. It inserts a triple t.
- R-Ins(t): is a remote insert operation generated executed on another site. It inserts a triple t.
- L-Del(t) : is a local delete operation generated and executed on the same site. It deletes a triple t.
- R-Del(t) : is a remote delete operation generated and executed on another site. It deletes a triple t.

Initially, L-Ins(t) and L-Del(t) operations are executed locally, then lunch R-Ins(t) and R-Del(t) respectively to execute remotely.

In order to maintain semantic stores consistency, every operation generated and executed on a site must be executed on all other semantic stores as well. This requires every generated L-Ins(t) and L-Del(t) to be broadcast to the other sites; after reception on a site the R-Ins(t) and R-Del(t) are executed on the local replica of the semantic store.

To guarantee that all concurrent users operations commute, we define a set of rules in all possible combination cases. These rules describe the algorithms comportments of remote and local operations executed on a given site.

Let us consider the following axioms:

- A triple t is locally removed if opType=L and B=false, and it is remotely removed if opType=R and B=false.
- A triple t is locally added if opType=L and B=true, and it is remotely added if opType=R and B=true.

We define also S: B-Set, S is a set of type B-Set where B-Set is a set of triples written as (opType, <T, B>).

Figures 2 and 3 show local insert and delete operations while figures 4 and 5 show remote insert and delete operations respectively.

```
L-Ins(t):
        If (t is locally added) then DoN(); //Do
        Nothing;
        else If (t is remotely added) then
        (S\(<R, <t, true>)) U (<L, <t, true>);
        else If (t is locally removed) then
         (S\(<L, <t, false>)) U (<L, <t, true>);
        else If (t is remotely removed) then
         (S\(<R, <t, false >)) U (<L, <t, true>);
        else (S U (<L, <t, true >));
Broadcast(R-Ins(t));
End.
```

Figure 2. A local insert operation L-Ins()

```
L-Del(t):
        If (t is locally added) then
        (S\(<L, <t, true>)) U (<L, <t, false >);
        else If (t is remotely added) then
        (S\(<R, <t, true>)) U (<L, <t, false>);
        else If (t is locally removed) then DoN();
        else If (t is remotely removed) then
        (S\(<R, <t, false >)) U (<L, <t, false>);
        else DoN();
Broadcast(R-Del(t));
End.
```

Figure 3. A local delete operation L-Del()

At the end of L-Ins() and L-Del functions, Broadcast(R-Ins()) and Broadcast(R-Del()) are invoked respectively in order to propagate and execute the operations on all remote sites.

```
R-Ins(t):
        If (t is locally added) then
        (S\(<L, <t, true >)) U (<R, <t, false>);
        else If (t is remotely added) then DoN();
        else If (t is locally removed) then
        (S\(<L, <t, false >)) U (<R, <t, false>);
        else If (t is remotely removed) then
        (S\(<L, <t, false >)) U (<R, <t, true>);
        else (S U (<R, <t, true >));
End.
```

Figure 4. A remote insert operation R-Ins()

```
R-Del(t):
        If (t is locally added) then
        (S\(<L, <t,true>)) U (<R, <t,false>);
        else If (t is remotely added) then
        (S\(<R, <t,true >)) U (<R, <t,false>);
        else If (t is locally removed) then
         (S\(<L, <t,false >)) U (<R, <t,false>);
        else If (t is remotely removed) then
        DoN();
        else DoN();
End.
```

Figure 5. A remote delete operation R-Del()

DoN() function in the case of L-Del(t) and R-Del(t) means that the delete operation hasn't effect if this triple hasn't been inserted yet.

In a classical set, there are two basic operations add(obj:object) and remove(obj:object), which are used for constructing new sets from given sets. For any set S, the first one corresponds to the union of the set S with an object obj ,and the second one corresponds to the difference of the set S and the abject obj. However, the traditional set data type with both operations can never provide a correct CRDT, because for the same object the addition and the removal do not commute such as illustrated in the figure 6.

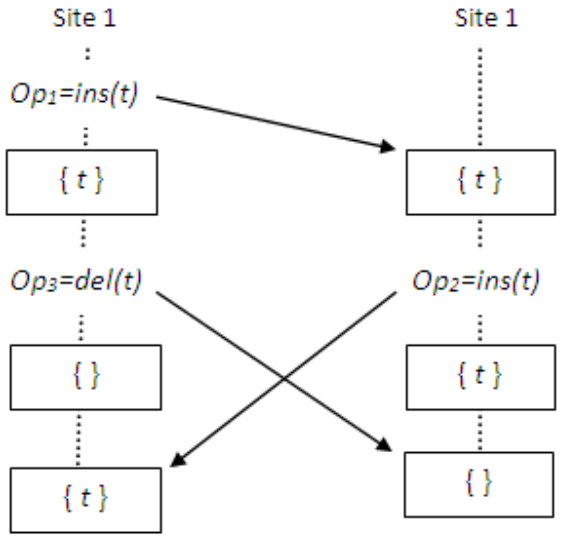


Figure 6. Classical set divergence

The figure 7 shows the execution of the example presented in figure 6 after integrating B-Set solution where both sites converge to the same semantic store.

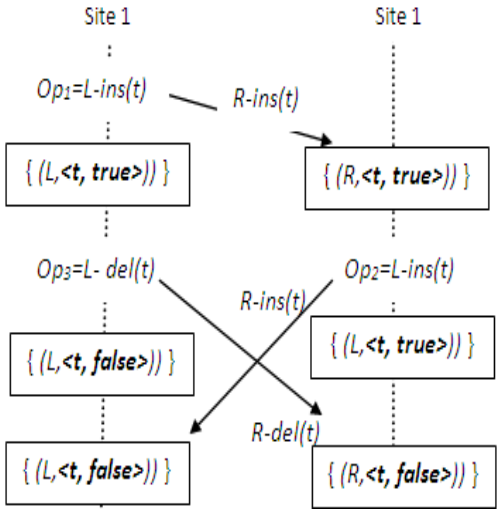


Figure 7. Calssical set divergence

The figure 8 shows another complex execution sample of B-Set. At beginning, all sites share the same semantic store that initially is empty. In site 2, the operations sequence are performed as [R-Ins(t), L-del(t), L-ins(t), R-del(t)]. The same operations are also executed on site 3 but in a different order [R-Ins(t), R-del(t), L-del(t), R-ins(t)] . Although both sequences present a real case of concurrent editing of semantics stores, they lead to identical results.

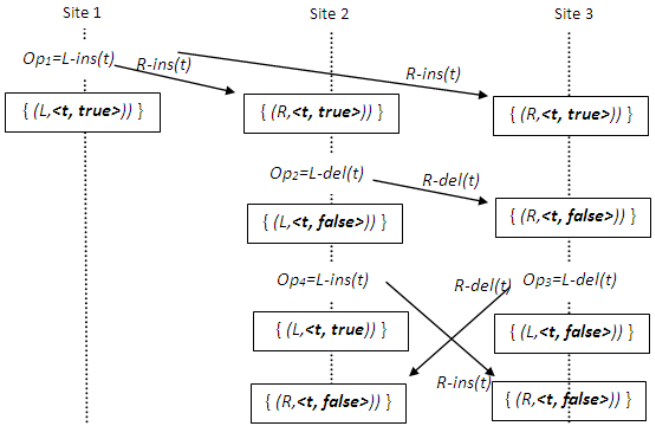


Figure 8. Another complex sample of concurrent editing with B-Set

## IV. DISCUSSION

B-Set is an original CRDT that has been designed for supporting the scalable collaborative editing of distributed semantic stores. Contrary to the other CRDT approaches based on set type, B-Set take into account the CCI model by the introduction of a new structure coupled with novel operations.

The causality criteria guarantees that every two operations generated on any site and ordered by a precedence relation will be executed in the same order on all other sites. In order to guarantee causality of the generated operations, there are many causal diffusions for scalable systems such as [28]. A scalable causally algorithm [29] can be used to verify the causal consistency by tracking and checking before the execution of each operation at any sites. However, B-Set has a very interesting characteristic that ensures eventual consistency without any requirement of causal delivery or receive.

A second interesting characteristic of B-Set is that the triples having false tag as visibility can be removed physically when the system is in idle state. This mechanism will facilitate to perform a garbage collection.

To prove that B-Set ensures eventual consistency, we must demonstrate that all pairs of operations commutes. Since a boolean variable is associated to each triple and the new value of this variable depends of the set of rules of basic conjunction and disjunction operation according to operation type, such as illustrated in the above algorithms. The proof of the commutativity of operations is straightforward. Thus, B-Set converges.

Since the effect of each generated operation is preserved in local or remote site by using of the field of opType combined with the visibility tag, the intention operation is respected. Compared to existing proposed techniques, our solution can support the CCI model by integrating a number of tags associated to each triple during a synchronization process.

## V. CONCLUSION

In this paper, we have established a new commutative replicated data type based on set structure in order to support synchronizing distributed semantic stores. As we have discussed above, the previous CRDTs consistency model shows ambiguities and deficiencies, which may not the best suitable one for describing the consistency issue in collaborative editing of semantic stores. Aimed at giving a more clear, suitable and complete consistency model, our model is designed not only to ensure convergence of triples replicas but also to ensure CCI model criteria integrated. We are currently examining open-source semantic stores' change logs and performing data mining to detect patterns in how users edit triple stores; additionally, we are examining the structure of thousands of triples extracted from DBpedia[8] to integrate B-Set as a synchronizing service. From these results, we plan to enhance our simulations with more authentic models of clients' synchronizing behavior and document structure. The comparative experiments are problems that we plan also to investigate in future research.

## REFERENCES

[1] T. Gruber, "Collective knowledge systems: Where the social web meets the semantic web", Journal of Web Semantics, Vol. 6, No 1, pp. 4-13, 2008.

[2] W3C. Resource Description Framework (RDF): Concepts and Abstract Syntax, Feb. 2004. http://www.w3.org/TR/rdf-concepts/.

[3] F. Manola and E. Miller. "RDF primer", W3C recommendation, February 2004.

[4] SPARQL 1.1 Update, http://www.w3.org/TR/sparql11-update/, accessed 05 January 2012.

[5] C. Sun, X. Jia, Y. Zhang, Y. Yang, and D. Chen, "Achieving Convergence, Causality Preservation, and Intention Preservation in

Real-Time Cooperative Editing Systems", ACM Transactions on Computer-Human Interaction, vol. 5, no. 1, pp. 63-108, 1998.

[6] N. M. Preguic, J.M. Marques, M. Shapiro, and M. Letia, "A commutative replicated data type for cooperative editing", Proc. International Conference On Distributed Computing Systems, IEEE Computer Society, pp. 395-403, 2009.

[7] Y. Saito and M. Shapiro. Optimistic replication. ACM Comput. Survey, vol. 37, no. 1, pp. 42-81, 2005.

[8] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann, "Dbpedia – a crystallization point for the web of data", Journal of Web Semantics, vol. 7, no. 3, pp. 154-165, 2009.

[9] RDFStore, 2006. Available: http://rdfstore.sourceforge.net/.

[10] M. Brian, "An Introduction to RDF and the Jena RDF API ", 2007.

[11] J. Broekstra, A. Kampman, and F. van Harmelen, 'Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema', In Proceedings of the 2nd International Semantic Web Conference, Sardinia, pp. 54–68, Springer, 2002.

[12] M. Cai and M. R. Frank, "Rdfpeers: a scalable distributed rdf repository based on a structured peer-to-peer network", In WWW, pp. 650-657, 2004.

[13] M. Cai, M. R. Frank, J. Chen, and P. A. Szekely, "Maan: A multi-attribute addressable network for grid information services", Journal of Grid Computing, vol. 2, no. 1, pp. 3-14, 2004.

[14] B. Quilitz, and U. Leser, "Querying Distributed RDF Data Sources with SPARQL', Proc. European Semantic Web Conference on The Semantic Web: Research and Applications", pp. 524-538, 2008

[15] G. Tummarello, C. Morbidoni, J. Petersson, P. Puliti, and F. Piazza, "Rdfgrowth, a p2p annotation exchange algorithm for scalable semantic web applications", In Proceedings of the MobiQuitous'04 Workshop on Peer-to-Peer Knowledge Management (P2PKM 2004), 2004.

[16] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmer, and T. Risch, "Edutella: a p2p networking infrastructure based on rdf", In 11th international conference on World Wide Web, pp. 604-615, 2002.

[17] C. A. Ellis, and S. J. Gibbs, "Concurrency control in groupware systems". Proc. ACM International Conference on Management of Data, pp. 399-407, 1989.

[18] C. Sun, and C. S. Ellis, "Operational transformation in real-time group editors: issues, algorithms, and achievements", Proc. ACM Conference on Computer Supported CooperativeWork, pp. 59-68, 1998.

[19] M. Suleiman, M. Cart, and J. Ferri "Concurrent operations in a distributed and mobile collaborative environment", Proc. International Conference on Data Engineering, IEEE Computer Society, pp. 36-45, 1998.

[20] N. Vidot, M. Cart, J. Ferri, and M. Suleiman, "Copies convergence in a distributed real-time collaborative environment", Proc. ACM Conference on Computer Supported Cooperative Work, pp. 171-180.2000.

[21] M. Cart, and J. Ferri, "Asynchronous reconciliation based on operational transformation for p2p collaborative environments", Proc. International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom, IEEE Computer Society, pp 127-138, 2007.

[22] S. Weiss, P. Urso, and P. Molli, "Logoot-Undo: Distributed Collaborative Editing System on P2P Networks", IEEE Transactions on Parallel and Distributed Systems, vol. 21, no. 8, pp.1162-1174, 2010.

[23] S. Martin, P. Urso, and S. Weiss, "Scalable XML Collaborative Editing with Undo", Proc. International Conference on Cooperative Information System, CoopIS, 2010.

[24] M. Shapiro, N. Preguica, C. Baquero, and M. Zawirski, "A comprehensive study of Convergent and Commutative Replicated Data Types", Research Report RR-7506, INRIA, January 2011.

[25] L. D. Ibáñez, H. Skaf-Molli, P. Molli, O. Corby, "Synchronizing semantic stores with commutative replicated data types", In Proceedings of the 21st international conference companion on World Wide Web, pp 1091-1096, 2012.

[26] K. Aslan, H. Skaf-Molli, P. Molli, and S. Weiss, "C-set : a commutative replicated data type for semantic stores". In RED: Fourth International Workshop on Resource Discovery, At the 8th Extended Semantic Web Conference (ESWC), pp. 123-130, 2011.

[27] H. Skaf-Molli, C. Rahhal, P. Molli, "Peer-to-peer Semantic Wikis", In Proceedings of International Conference on Database and Expert Systems Applications, DEXA, pp. 196-213, 2009.

[28] S. Kawanami, T. Nishimura, T. Enokido, M. Takizawa, "A Scalable Group Communication Protocol with Global Clock", AINA, pp. 625-630, 2005.

[29] W. Lloyd, M-J. Freedman, M Kaminsky, D-G. Andersen, "Don't settle for eventual: scalable causal consistency for wide-area storage with COPS", SOSP, pp. 401-416, 2011.