ISSN 2319-2720
Volume 7, No.2, April – June 2018
International Journal of Computing, Communications and Networking
Available Online at http://www.warse.org/ijccn/static/pdf/file/ijccn38722018.pdf
https://doi.org/10.30534/ijccn/2018/38722018

FILE SYNCHRONIZATION : TOWARDS AN EFFICIENT FILE SYNCHRONIZATION BETWEEN DIGITAL SAFES

Prof.Jagadeesh B N[1], Usha A[2], Shriraksha B Rao[3] , Sheethal Gowda K[4], Sahana H S[5]
[1]Assistant professor EWIT, India, jagadeesh.nagaraj.nl@gmail.com
[2]Student, EWIT, India, ushagowda10@gmail.com
[3]Student, EWIT, India, shreeraksharao05@gmail.com
[4]Student, EWIT, India, sheethalgowdak13@gmail.com
[5]Student, EWIT, India, 14sahanahs@gmail.com

ABSTRACT

The paper's goal is to develop a secure framework that ensures file synchronization with high quality and minimal resource consumption. As a first step towards this goal, we propose the SyncDS protocol with its associated architecture. The synchronization protocol efficiency raises through the choice of the used networking protocol as well as the strategy of changes detection between two versions of file systems located in different devices. Our experiment results show that adopting the Hierarchical Hash Tree to detect the changes between two file systems and adopting the. WebSocket protocol for the data exchanges improve the efficiency of the synchronization protocol.

Key words: File synchronization, Digital Safe, HTML5 Local

Storage API, WebSocket, Hierarchical Hash Tree, Web services.

1. INTRODUCTION

A standardized Cloud storage solution with the probative value is introduced by the Safe Box As A Service (SBaaS) . It is a standardized architecture that provides a secure environment to store sensitive documents. The conception of this safe follows the AFNOR specifications . It is a standard that offers the best possible security, integrity and quality to preserve user's data. It is characterized by its probative value as a proof of storage is preserved in a third trusted party. Introducing the synchronization in a Digital Safe platfo rm implies the definition of a Client Digital Safe. In our platform, the client safe is based on HTML5 APIs. The major interest of using HTML5 is to avoid proprietary solutions and to adopt standardized one. It offers also the transparency to the end user

and guaranties the mobility and portability while preserving an efficient traffic exchange. In this paper, we highlight the major need of defining a standardized architecture for file synchronization with efficiency considerations. First, we focus

on the synchronization between the Client Digital Safe and the

Cloud Digital Safe. Second, we address the efficiency in this context by the choice of the used networking protocol as well as the strategy of changes detection between two versions of file systems.

With the invasion of used smart objects, the cross-device data management remains the concern of multiple research and industrial works. Various devices, owned by the same user or different one, rely on synchronization protocols in order to maintain data consistency. Several Cloud storage solutions are

proposed to handle this issue. However, the commercialized products are usually based on proprietary solutions. They obviously lack transparency for the users how their data are managed in the client and Cloud side. These closed solutions depend heavily on the used machine.

2.NETWORK PROTOCOLS FOR FILE SYNCHRONIZATION

Various networking protocols and architectures are used to ensure data synchronization including both the notification and the data transfer. In fact, a part of the infrastructure sends messages to the concerned connected clients. It notifies them that changes have occurred and the data transfer for synchronization should start. For example, Dropbox uses the HTTP for the data transfer and the long HTTP polling for the notification. Regarding Google Chrome synchronization, the data transfer is also based on HTTP and the notification exploits an existing XMPP-based Google Talk server. REST API is also used to ensure data replication in multiple solutions such as CouchDB . Indeed, RESTful applications are mainly based on HTTP protocol.

Choosing the best protocol for the data exchange is very important to the synchronization protocol efficiency. In fact, it is crucial to reduce the amount and the size of messages

exchanged between the both end points. In this paper, we address the introduction of the WebSocket API and protocol in the architecture of synchronization.

3. EFFICIENT SYNCHRONIZATION PROTOCOL REQUIREMENTS

In the context of file synchronization, the protocol has to provide four main key properties that should be respected in order to guarantee an efficient bandwidth, the fastest data exchange and an effective computation.

- **Property 1**: Low computation of the data and metadata at the client side. This property adds the scalability and enables simultaneous synchronization.

- **Property 2:** Efficient change detection between the file system versions of the client and the server. The goal is to reduce the time of matching and therefore, the time of whole synchronization.

- **Property 3:** Reducing the number of synchronized files by finding the maximum number of matched content.

- **Property 4:** Reducing the messages between the client and the server.

4. EXISTING SYSTEMS AND THEIR DRAWBACKS

The different proposed change detection algorithms can be classified into three main approaches.

Operation Approach:

This approach is based on recording the different operations performed on the node A and sending them to the remote node B. These operations are sent to the node B to update its replica

Drawback: Storing the operation is a consumption of the storage memory.

Changes Approach:

The main idea behind this approach is to save a copy of the last synchronized file system version After a series of modifications and when the user becomes online, the old copy and the new version are compared. Actions are therefore detected and sent to the remote node B .

Drawback: It needs additional storage space for archiving the file system. In addition there is a lack of automatic detection since it must be triggered periodically.

Differencing Approach:

The node A sends an abstract of its files and directories to the remote node. This abstract includes some metadata of files. In the case of dropbox , the metadata contains the object path, object type (files or directories) and the hashs of 4Mb file**'**s blocks. In the case of Rsync , it sends the hash of file blocks, and Taper sends the Hierarchical Hash Tree, the hash of file**'**s chunks and the hash of file**'**s blocks. The node B, therefore, compares its abstract with the one received from A to detect the changes. It asks A then to send back missing blocks of files. **Drawback:** This approach cannot guarantee an automatic synchronization and sending periodically the abstract leads to an efficient bandwidth usage.

5. PROPOSED SYSTEM

The goal of our framework is to ensure the data synchronization between a Local Digital Safe and a Cloud Digital Safe while considering the probative value. As a first step, we need to define the architecture as well as the messages exchanged between its different entities. The architecture is divided into

Client Storage Layer:

The novelty in the architecture is the non-proprietary characteristic. The data are stored securely in a Local Digital Safe. This Safe is based on the HTML5 Local Storage API with additional security considerations. In fact, we add into the existent APIs the confidentiality by encrypting the data locally, the data integrity and metadata integrity. These security enhancements are the subject of previous work, and more details can be found in .

Application Layer:

This part includes the web application with the different used APIs. We introduce two main modules. The first one, Digital Safe, implements the AFNOR specifications to manage locally the data stored using the File System API. The second module, synchronization, is used to detect the user**'**s operations applied on the Local Digital Safe and to record them. It manages then the exchanged messages between the Local Digital Safe and the Cloud Digital Safe following the SyncDS protocol.

Synchronization Control Layer :

To go beyond the commercialized solutions and to have the best performances, we choose the WebSocket protocol to ensure the bidirectional communication between the local and the remote Digital Safe. The synchronization management server handles the different synchronization requests and responses as well as the conflict resolution. It also notifies the devices concerned by the modification to propagate the changes. Our proposed algorithm, based on HHT, is introduced at this level.

Server Storage Layer:

As introduced in , the Cloud Digital Safe is a standardized architecture that provides a secure environment for storing sensitive document. This environment fully fits both the user requirements and Cloud security challenges. This Cloud Digital Safe is composed of three main components: Metadata server, storage servers and Proof Manager.

Advantages of the proposed system:

- Less on storage space
- Ensures automatic synchronization

6. ARCHITECTURE

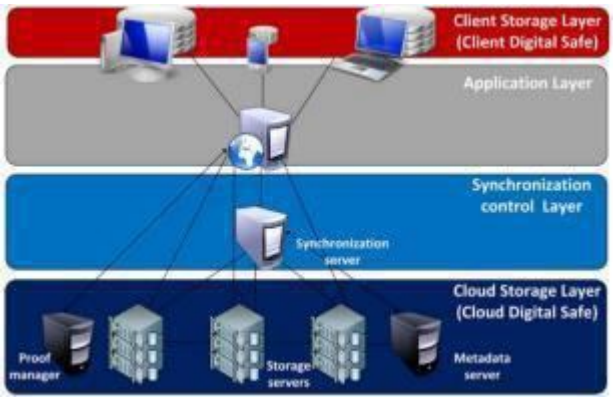

Fig 6.1:Digital Safe Synchronization Architecture

7. SYNCDS SYNCHRONIZATION PROTOCOL:

Overview on synchronization steps

After the architecture definition, we need to itemize the synchronization protocol. SyncDS ensures data synchronization in a standardized Digital Safe context.

Offline phase: when the user goes offline, the Client Digital Safe stores in a log file the different operations performed locally. This strategy is the unique which can be used even if it needs storage capacity. In fact, it guarantees the non-proprietary characteristic and avoid the use of solutions which depend on the used operating system. These operations are detected through the enhanced HTML5 fileSystem APIs specification and the Application cache API. They are stored then using the HTML5 WebStorage API.

On connection phase: It is a two way synchronization that includes two steps. In the first step, the client posts changes performed when it was offline. This step is based on the operation approach. In fact, the log file is sent to the server. The server then applies on his version the changes as listed in the log file. In the second step, the server reveals changes performed on his side when the user was offline. These changes can be performed by the same user in a different device or by another user who shares a part of the file system. In this step, the differencing approach is adopted.The Client Safe sends an abstract of its file system to the server. The server then compares the received abstract with his version, detects changes to send them back to the user. We propose a new algorithm based on HHT to detect the changes. This algorithm will be detailed later. In case of multiple devices connected at the same time, the problem of conflict resolution is raised and many techniques are already proposed to solve it.

Online phase: It is a two way synchronization. Changes made on the client side are sent to the server and changes, made by different devices and synchronized to the server, are sent to the user. This step is based on the operation approach. The novelty and originality of our protocol are the introduction of the WebSocket to send data from the client to the server. It is also adopted by the server to send notifications and data to the client.

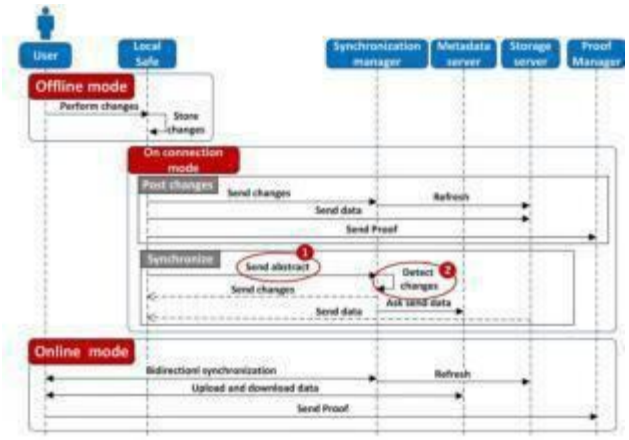

Fig 7.1:Overview on the Synchronization Protocol

8.ALGORITHM

1: **procedure** DETECT MATCHING(T1,T2)
2: **Input** T1, T2: tree
3: **Output** T1,T2: tree
4: **Traverse** T1 in a level order and top-down
5: **let** x be the current node
6: **if** y T2/ match(x,y)=true **then**
7: **Delete** (subtree(x), T1);
8: **Delete** (subtree(y), T2);
9: **if** Name(x) _ = Name(y) **then**
10: **Rename**(x,Name(y))
11: **end if**
12: **if** y T2/ Hash(x)=hash(y) and Parent(x)=Parent(y) **then**
13: **Mark** (y, x);
14:
15: **end if**
16: **end if**
17: **End for**-traverse
18: **if** y T2/ Mark(y)=True **then**
19: **Copy** (x, Name(y));
20: **Delete** (subtree(y), T2);
21: **end if**
22:
23:**end procedure**

9. IMPLEMENTATION AND PROOF OF CONCEPT

As a proof of concept and a proof of the protocol efficiency, we focus mainly on three parts of the synchronization architecture.

- The Chromium browser: We focus, in this paper, on the Filesystem API. We add the encryption of files content in th client side. We add also the data integrity with the verification of the file hash and the metadata integrity by encrypting the file

name when stored in the indexed database by the chromium browser.

- Synchronization in the Web application: A HTML5 web application is developed based on the enhanced File System, the basic Webstorage, the Application Cache and WebSocket APIs. At this level, the abstract of the file system is build following the HHT structure. This application allows the user to manage his Digital Safe, store the operations when he is offline and to ensure the synchronization of his Digital Safe content following the SyncDS protocol.

- Synchronization in the WebSocket Server: To introduce the WebSocket server, we use the web server Apache with its extension mod pywebsocket. We add in the server side the detection of changes by comparing two abstracts (the one sent by the Client Digital Safe and the other extracted from the Cloud Digital Safe). The comparison of both abstracts is implemented using the Java language and more specifically using the TreeModel interface.

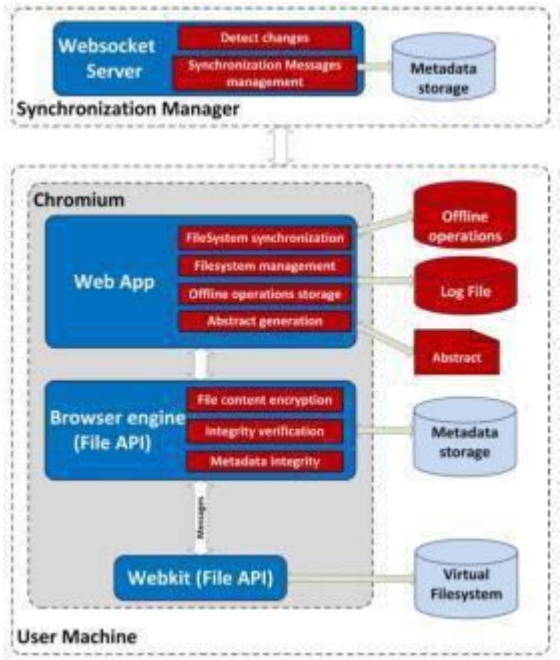

Fig 9.1:Implementation of the synchronization architecture and Protocol

10. CONCLUSION

In front of the various owned devices and the need of synchronizing the data between them, ensuring an efficient file synchronization protocol is crucial. In this paper, we propose an architecture and a protocol that ensure file synchronization a probative value Cloud. Two keynote novelties of the SyncDS protocol are highlighted: first, the integration of the Hierarchical Hash Tree into the metadata abstract and second, the non-proprietary characteristics with the adoption of HTML5 APIs. Our experimental results show that using the

new proposed framework, reduces the time of change detection and therefore, reduces the time of file synchronization across devices .

The conflict resolution is raised in our architecture in case of multiple connections at the same time. As future work, we will deal with the interference of the conflict resolution strategies with the execution of our protocol.

REFERENCES

[1] M. Msahli and A. Serhrouchni, **"**Sbaas: Safe box as a service,**"** in 9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing, Nov 2013.

[2] **"**Afnor groups.**"** [Online]. Available: http://www.afnor.org/en

[3] M. D. N. Jain and R. Tewari., **"**Taper: Tiered approach for eliminating redundancy in replica synchronization,**"** in . In Proc. of the USENIX Conference on File And Storage Systems, 2005.

[4] S. Agarwal, D. Starobinski, and A. Trachtenberg, **"**On the scalability of data synchronization protocols for pdas and mobile devices,**"** IEEE Network, Jul 2002.

[5] B. Xianqiang, X. Nong, S. Weisong, L. Fang, M. Huajian, and Z. Hang, **"**Syncviews: Toward consistent user views in cloud-based file synchronization services,**"** in Sixth AnnualChinagrid Conference (ChinaGrid),, Aug 2011.

[6] H. Yan, U. Irmak, and T. Suel, **"**Algorithms for low-latency remote file synchronization,**"** in The 27th Conference on Computer Communications. IEEE INFOCOM 2008, April 2008.

[7] C. Liang, L. Hu, Z. Lei, and J. Wang, **"**Synccs: A cloud storage based file synchronization approach,**"** Jul 2014.

[8] Benjamin, C.Pierce, and J. Vouillon, **"**What**'**s in unison? a formal specification and reference implementation of a file synchronizer,**"** in Tech. rep. MS-CIS-03-36, Department of Computer and Information Science, University of Pennsylvania, 2004.

[9] I. Drago, M. Mellia, M. M. Munafo, A. Sperotto, R. Sadre, and A. Pras, **"**Inside dropbox: Understanding personal cloud storage services,**"** in Proceedings of the 2012 ACM

Conference on Internet Measurement Conference, ser. IMC '12, 2012.

[10] A. Tridgell and P. Mackerras, "The rsync algorithm. technical report trcs-96-05, department of computer science," in The Australian National University, Canberra, Australia, 1996.

[11] S. S. Chawathe, A. Rajaraman, H. Garcia-Molina, and J. Widom, "Change detection in hierarchically structured information," in Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD '96, 1996.

[12] S. S. Chawathe and H. Garcia-Molina, "Meaningful change detection in structured data," in Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD '97, 1997.

[13] R. Al-Ekram, A. Adma, and O. Baysal, "diffx: An algorithm to detect changes in multi-version xml documents," in Proceedings of the 2005 Conference of the Centre for Advanced Studies on Collaborative Research, ser. CASCON '05, 2005.

[14] N. Jain, M. Dahlin, and R. Tewari, "Taper: Tiered approach for eliminating redundancy in replica synchronization," in In Proc. of the USENIX Conference on File And Storage Systems, 2005.

[15] J. C. Anderson, J. Lehnardt, and N. Slater, "Couchdb the definitive guide." [Online]. Available: http://guide.couchdb.org/

[16] M. jemel and A. serhrouchni, "Security assurance of local data storedby html5 web application," in The 10th International conference on Information Assurance and Security, Okinawa, Japan, Nov 2014.

[17] Y. Minsky, A. Trachtenberg, and R. Zippel, "Set reconciliation with nearly optimal communication complexity," Information Theory, IEEE Transactions on, Sept 2003.