Efficient User Data Synchronization

Stančin, Sara; Đorđević, Nemanja; and Tomažič, Sašo

Abstract

User data synchronization feature of computing user devices presents an essential role in the vision of pervasive networks. Typical user environment includes several devices - a work desktop computer, home laptop, and mobile device for example. Through each of these devices a user can access different data records. The aim of synchronization is to enable user access to a particular data record through a device other than the one on which the record was whether created whether last modified. Users can read, change, or delete a specific data record by accessing any of its available replications. Such independent access can embrace any type of file or web content in applications like email, file content distribution, and personal information manager.

The synchronization environment can include numerous devices with different processing, memory and connection capabilities. Further on, devices involved operate in two possible modes, the online and the offline. As long as a device is not connected to a particular network new data record modifications are not consistent with corresponding data records on other devices in that network. As soon as a device connects to a network, the synchronization process needs to determine and reconcile possible differences between documents replicated on multiple devices. Due to multi device involvement in data updating, different synchronization conflicts may occur. Synchronization protocols in force implement different, more or less complex mechanisms which dictate system behaviour if synchronization conflicts arise. The efficiency of these mechanisms varies.

A new paradigm for achieving efficient user data synchronization is presented. Documents are associated with globally unique identification marks while document replication validity is achieved using timestamp values. The proposed paradigm enables reliable synchronization of user data, while the synchronization process remains simple and efficient from the computational point of view. The synchronization procedure according to the new paradigm is efficient even in a dynamic environment which involves numerous devices.

Manuscript received May, 11, 2009.

Sara Stančin is with the University of Lubljana, Slovenia
Sašo Tomažič is with the University of Ljubljana, Slovenia
Nemanja Đorđević is with the University of Belgrade, Serbia

KEYWORDS

User data synchronization, Data reconciliation, Unique global document identification

1. INTRODUCTION

1.1. Fundamentals of user data synchronization

Users access their data through different computing devices like mobile telephones, laptops, desktops, and handheld computers. Each single user device stores a specific user data subset. Depending on the used device, users can therefore access different subsets of data. Recently, more and more of these devices, belonging to the same user, communicate and collaborate among themselves in such information delivery. Access to user data that has previously been enabled through one particular user device is being enabled through other user devices as well. Users can read, change, or delete a specific data record by accessing any of its available replications. Such independent access can embrace any type of file or web content in applications like email, file content distribution, and personal information manager.

Multiple copies of data records that represent the same data on different devices must be the same in content. This leads us to the problem of user data record synchronization. The synchronization process needs to determine the differences between user data replicated on multiple devices and thereafter reconcile these differences as illustrated in Figure 1.1.

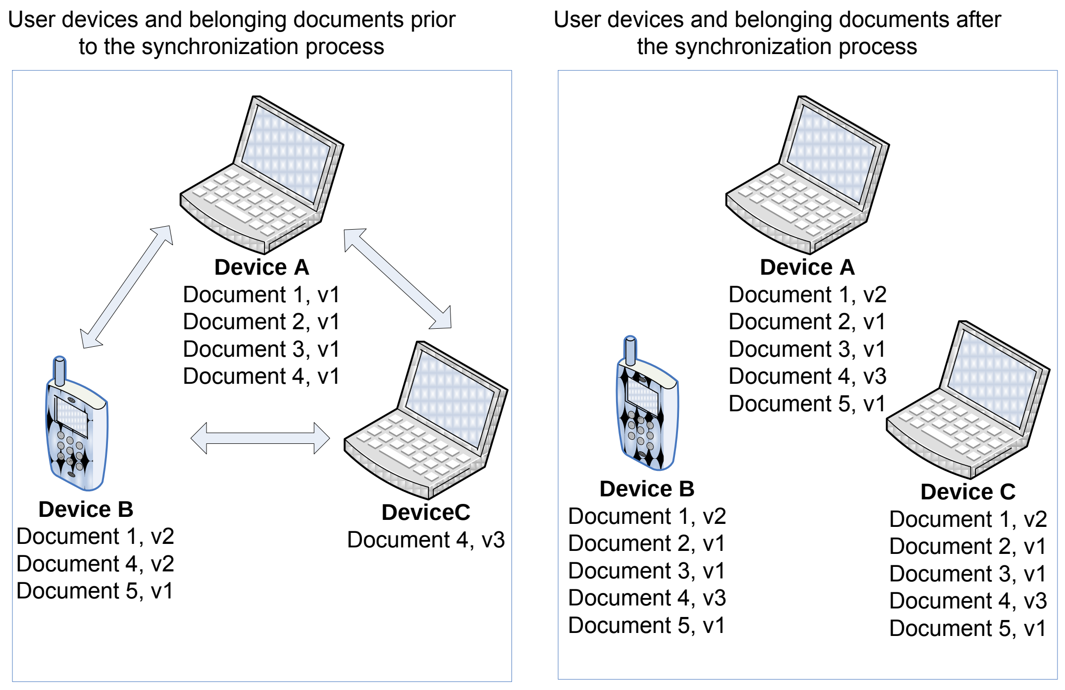

Figure 1.1: User data synchronization aim and purpose. Before the synchronization process each devices stores a different subset of user documents while after the synchronization process all documents in the user network are stored on all of the involved devices and have equal content.

The synchronization environment usually includes numerous devices like powerful desktop machines (computationally very fast and efficient personal computers equipped with sufficient memory and relatively fast network connections) as well as mobile devices with limited capabilities and resources. Limited processing capabilities and memory resources of user devices can obstruct extensive synchronization computation and data replication. Moreover, due to numerous devices involved, which work in both, the offline and the online mode, different synchronization conflicts may occur as shown in Figure 1.2. These problems lead to data inconsistencies and can make the realization of a reliable data synchronization protocol a complex problem to solve. Synchronization protocols in force implement different, more or less complex mechanisms which dictate system behavior if synchronization conflicts arise. The efficiency of these mechanisms and procedures varies. It is not unusual that after synchronization is achieved users notice different faults and unexpected results. Synchronization faults embrace problems like data loss, data record duplication, and invalid data record content.

1.2. Article orientation

Our aim is to present a new paradigm that embraces possible user data synchronization improvements. First we give a presentation of the synchronization process. We give special attention to principles which determine how data reconciliation is achieved and how conflicts are resolved. Further on we present the introduction of unique global document identification for reliable user data synchronization. Different replications of a particular document have all the same unique global identification mark. During the synchronization process, valid replications are determined according to their timestamp value. Using timestamps and global identification at the same time, simple and exclusive reconciliation procedures are enabled. The presented solution is efficient and reliable even in the environment which involves numerous user devices.

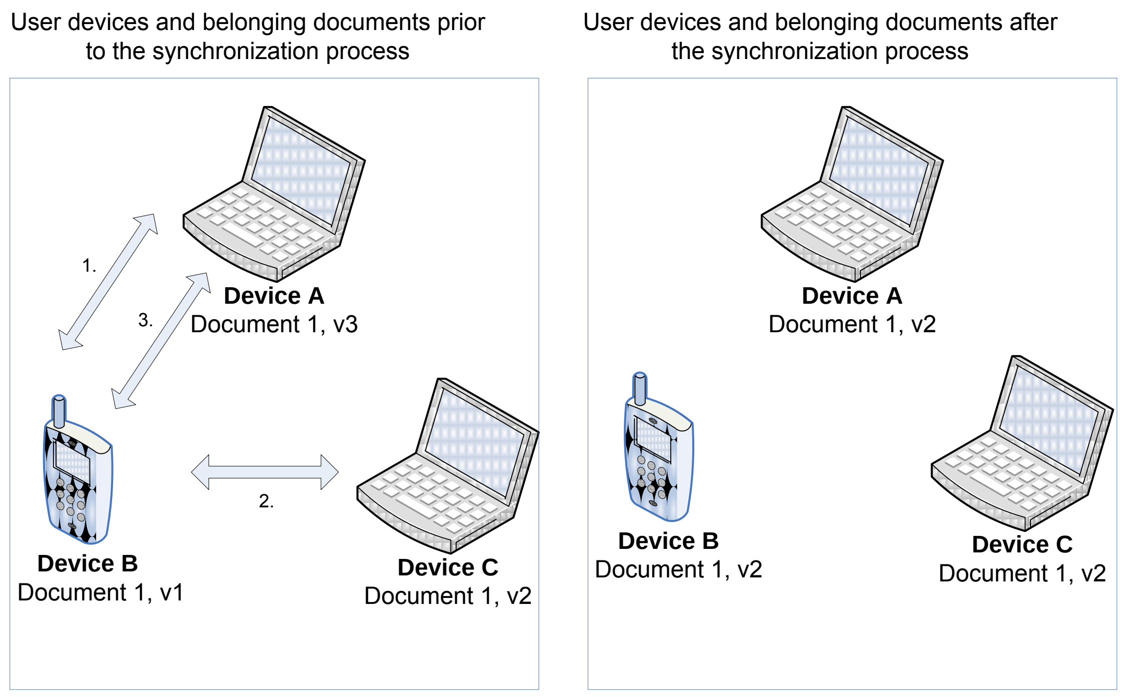
User devices and belonging documents prior to the synchronization process
1.
3.
Device A
Document 1, v3
2.
Device B
Document 1, v1
Device C
Document 1, v2
User devices and belonging documents after the synchronization process
Device A
Document 1, v2
Device B
Document 1, v2
Device C
Document 1, v2

Figure 2.2: User data synchronization error due to conflict occurrence. In the presented example three devices synchronize among themselves, where Devices A and C replications of Document 1 have been changed since last synchronization was performed. These changes occurred is such time sequence, that Document 1 v3 is the newest version and should therefore be propagated to other devices included in the network (Devices B and C). The result of the first synchronization is unambiguous as only one replication (Document 1 v3) is changed and established to be valid. As no conflicts arise, Devices B version of Document 1 is replaced with Devices A version of the same document. During the second synchronization process however, the two cooperating devices, Devices C and B can not establish which version of Document 1 is to be treated valid as both of the replications have been changed since the last synchronization between these devices. In the presented case, such synchronization mode was chosen, that in the case of conflicts, Devices B replications are replaced by replications on Device C. If now Devices B and A should again reconcile their documents sets, a synchronization error would occur as devices A replication content is replaced with the invalid content originated on Device C.

2. USER DATA SYNCHRONIZATION

Synchronization protocols dictate the rules for devices communication, user data distribution and reconciliation, and conflict resolution. Implemented rules affect protocol performances - network scalability, time, memory and computation efficiency, their openness and wideness of applicability. We focus of synchronization mechanisms that enable change detection and reconciliation and conflict resolution.

2.1. Detection of data record differences

Synchronization protocols relay on special mechanisms for establishing which data records have been changed since the last synchronization. These include status flags, timestamps and algebraic set reconciliation. Maintaining change logs on both the server and the client side is also possible but is rarely sufficient. When no change detection mechanisms are considered all applications data records get overridden by records from another application data set. Having in mind possible extensiveness of data set, the realization of such a concept is usually inefficient from the bandwidth usage, latency and energy consumption point of view.

Supporting status flags is the most

common mechanism that different protocols implement. These flags are used to indicate if a special data record has been modified, inserted or deleted since the last synchronization has been accomplished. Usage of single status flags considers maintenance of one flag for each record in an application's data set. In general, upon a synchronization request all data records whose status flags have been changed get transmitted from the client to the server device. The server then establishes what the proper synchronization result is and sends its corrections back to the client. While computation, memory, data transmission load and central point of failure are shown as strengths of the single status flag network size is considered to be a problem. This particular usage of status flags limits the synchronization process in a way that a device can synchronize with a device with which it synchronized last.

Usage of multiple status flags per record supports multi device synchronization. For each data record as many flags are maintained as is the number of devices in the user network with which that device synchronizes – a flag per record with reference to each device. Devices can therefore synchronize with different devices each time but the synchronization procedure still does not scale with network size due to time and memory problems of maintaining a vast number of flags.

Change detection can also be achieved using timestamp record denotation. Synchronization is then achieved by transferring all data records marked with time later than the time when previous synchronization was accomplished. Some argue the efficiency of timestamps and the protocols that implement this mechanism as they [1] "poorly adapt to dynamic situations where devices are frequently added or removed from the network".

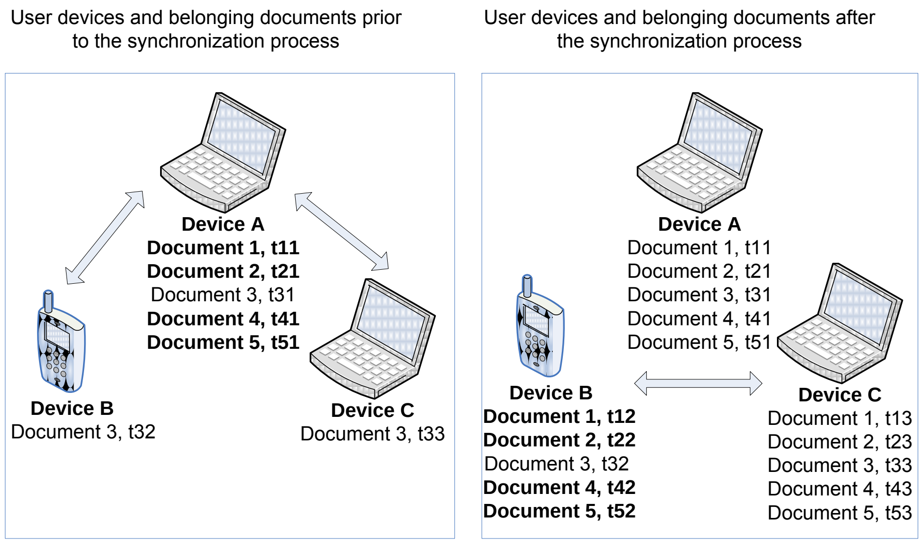

Figure 2.1: Inefficient communication cost when synchronizing user data differences established according to the timestamp and status flags mechanism. In the presented example, Devices B and C one after another synchronize with Device A. The sequence of these two synchronizations is not important. Documents 1, 2, 4, and 5 are transmitted from Device A to the other two devices and so all three store documents which are equal in content. Let us now consider the scenario where Device B synchronizes with Device C. If timestamps are used, Documents 1, 2, 3, and 4, stored on Device B are marked with a timestamp value later then the last synchronization process between the two devices. If multiple status flags are used, these documents are also marked as changed. In either way Documents 1, 2, 4, and 5 would be again transmitted, regardless of the fact that there are no differences between the content of these two sets of documents.

Synchronization schemes can be improved using more sophisticated computational methods [3]. Algebraic set reconciliation considers data to be represented by sets. When a pair of devices exchanges information, the devices must reconcile their respective data sets without the a priori knowledge of which data records need to be transmitted. The gain of algebraic set reconciliation algorithms typically manifests when the number of differences to be reconciled is far smaller than the size of the reconciling databases. Intensive computation provides a replacement for huge amounts of data transmitted over the network. Algebraic set reconciliation protocols do not need to maintain any state information about other devices in the network, but they are computationally intensive.

2.2. Conflict resolution

Specific nodes in a network have to be able to efficiently identify data record changes and propagate updates to other devices present in the network. The problems arise when a specific data record has been inconsistently changed on both cooperating sides, the one acting as a client and on the one acting as a server. These situations represent synchronization conflicts. Resolving conflict depends on the server side synchronization algorithm. A synchronization algorithm my simply decide which changes to be treated as the valid ones: client changes always win, server changes always wins or, in the example of timestamps, latest change always wins. Some synchronization algorithms are less exclusive and take into consideration merging data records and data duplication.

3. New Synchronization Paradigm Proposition

3.1. New paradigm basis

We propose a new paradigm for achieving efficient user data synchronization from both, the time and the processing consumption point of view. Efficient synchronization is achieved by associating documents with globally unique identification marks and with document replications differentiation referring to their timestamp values according to the following:

- In the network of user devices, only one specific document exists at a specific moment of time. A unique global document identification (GDID) value is associated with each document. Different devices obtain different replications of a particular document and so multiple replications exist. Each of these replications has the same identification mark and when synchronized, these replications contain the same user content.

- For GDID determination we consider the usage of cryptographic hash functions. The determination of the GDID value is extremely important for keeping data sets in a consistent state. Each user document must have a different GDID value. The GDID value should be a value with characteristics of a random value, long enough so we can presume that the possibility of a collision would be contentedly low. A cryptographic hash system that works with one billion 160-bit hashes has an approximate $10^{\wedge(-32)}$ collision probability.

- Each replication of a document has its own timestamp mark. Same timestamp values of two documents replications imply that they are synchronized. If the values are different, the replications are different in content and must be reconciled during the next synchronization.

- The timestamp value also enables conflict resolution. If during the synchronization process, two involved replications of a particular document (recognized by the same GDID) have different timestamp values, one that has the bigger timestamp value, we can also say - the "younger" replication - is considered to be the valid replication of that particular document. The synchronization protocol must then provide for the according modification of the invalid replication.

- A document can be hard deleted or soft deleted. When a document has been soft deleted it implies that only the record content and not the record itself have been deleted. This kind of deletion is actually treated equally as data content modification. After deletion, the data record itself remains stored in device memory and it retains its GDID. The timestamp value belonging to that, content empty, replication is updated, at the same time it does not needlessly occupy valuable device memory space. For achieving efficient synchronization of data replications that should remain deleted, each deleted record can be marked with an additional element.

- After it is achieved that two documents replications are the same in content, they must have the same timestamp value. The timestamp value of the "older" replication is overridden by the "younger" replications value.

3.2. The synchronization process

The initial synchronization agreement includes the exchange of device capability information in order for devices to complete the client-server agreement. Each device maintains a list of timestamp values which represent the time of the last synchronization with each other user device in the network. Upon the initial synchronization agreement, either one either both devices, depending on the synchronization mode, create a list of data records. Data records are represented with their GDID and timestamp values. The list is created in such a way that a device compares records timestamp values with timestamp values of the last corresponding synchronization. Data records with timestamp values smaller, or "younger", then the corresponding timestamp value are considered to have been modified since the last synchronization and are inserted into the list. If the last synchronization timestamp is for some reason unavailable, these lists comprise GDID and last modification timestamp values of all records in a particular data set. The client device then sends this list to the server device. After receiving such a list, the server side makes a comparison of the received client list and its own list upon what it can establish replication validity. The server then requests that appropriate reconciliation is performed. The synchronization process can finish when timestamps are adjusted and all of the performed changes are reported to the server.

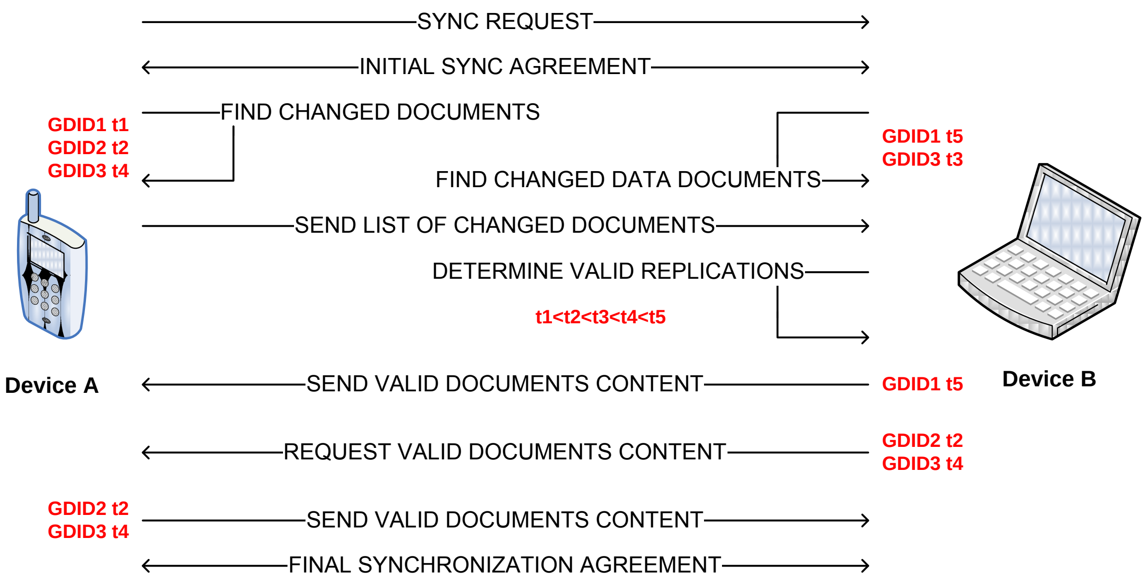
SYNC REQUEST
INITIAL SYNC AGREEMENT
FIND CHANGED DOCUMENTS
GDID1 t1
GDID2 t2
GDID3 t4
GDID1 t5
GDID3 t3
FIND CHANGED DATA DOCUMENTS
SEND LIST OF CHANGED DOCUMENTS
DETERMINE VALID REPLICATIONS
t1<t2<t3<t4<t5
Device A
SEND VALID DOCUMENTS CONTENT
GDID1 t5
Device B
REQUEST VALID DOCUMENTS CONTENT
GDID2 t2
GDID3 t4
GDID2 t2
GDID3 t4
SEND VALID DOCUMENTS CONTENT
FINAL SYNCHRONIZATION AGREEMENT

Figure 4.1: The synchronization process according to the new paradigm proposal. After Device A creates the list of changed documents, it sends it to Device B, which is in this case acting as server. After receiving such a list, Device B can compare Devices A list with its own and determined valid replications can be exchanged between the two devices.

3.3. The relation between the new synchronization paradigm and SyncML

For the proposed paradigm implementation purposes we suggest the usage of the SyncML standard. SyncML is an industry initiative which seeks to provide an open standard for data synchronization across different platforms. This standard assumes that each client device maintains status flags for each of its records with respect to every other device on the network. Therefore, multi device synchronization is supported. SyncML synchronization enables devices to synchronize with a different device each time. More extensive reports on SyncML can be found in [1], [3], [4], and [5]. The SyncML synchronization process can be extended by defining new data elements which enables timestamps and GDID inclusion. The proposed paradigm is SyncML supplementary and is not in confrontation with any of its goals. At the same time, including the proposed mechanisms, reported SyncML weaknesses regarding mapping tables and timestamps can be avoided.

The globally unique identification concept opposes to the identification concept, with which the SyncML initiative is familiar with. SyncML uses two identifiers, the LUID (*Local Unique ID*) and the GUID (*Globally Unique ID*). LUID identifiers are associated with client data records, while the GUID identifiers represent the corresponding data on the server side. The LUID is always assigned by the client device. As the LUID and GUID of a particular data record can be different, each server has to maintain special mapping tables – one for each of the involved clients. The maintenance of mapping tables is therefore of crucial importance for achieving efficient data synchronization. As the size of these mapping tables grows linearly with the number of data records, these tables can become significant in their extent. Moreover, different SyncML clients can assign different LUID identifiers for the same data record and different clients can assign same LUID identifier for different data records. Inadequate and faulty mapping table maintenance can consequently lead to user data inconsistency, duplication and loss of content.

The proposed usage of timestamps also opposes to the usage with which SyncML is familiar with. When more than two devices are synchronizing, the use of timestamps results in an inefficient communication cost. For example let us consider the synchronization environment which includes three devices: Device A, B, and C as illustrated in Figure 4.2. Further on, let us consider that since the last synchronization was performed, Documents 1, 2, 4, and 5 were created on device A. The problem arises if Devices B and C try to synchronize with each other after they have successively performed synchronization with device A. In such an example, modification records require transmission of eight differences, whereas, in fact, there are none. The proposed representation of document replications with GDID and timestamp values, this inefficiency is avoided.

5. CONCLUSION

The synchronization protocol must support reliable reconciliation of replicated user data which is a demanding task as from the computation as from the communication point of view considering that the synchronization environment can include devices with different capabilities and limited network resources. Synchronization protocols rely on special mechanisms for establishing which data records have been changed since the last synchronization. These include status flags, timestamps and algebraic set reconciliation. Supporting status flags is the most common mechanism that different protocols implement. While computation, memory, data transmission load and central point of failure are shown as strengths of the single status flag network size is considered to be a problem. This particular usage of status flags limits the synchronization process in a way that a device can reconcile its records with records stored on a device with which it synchronized last.

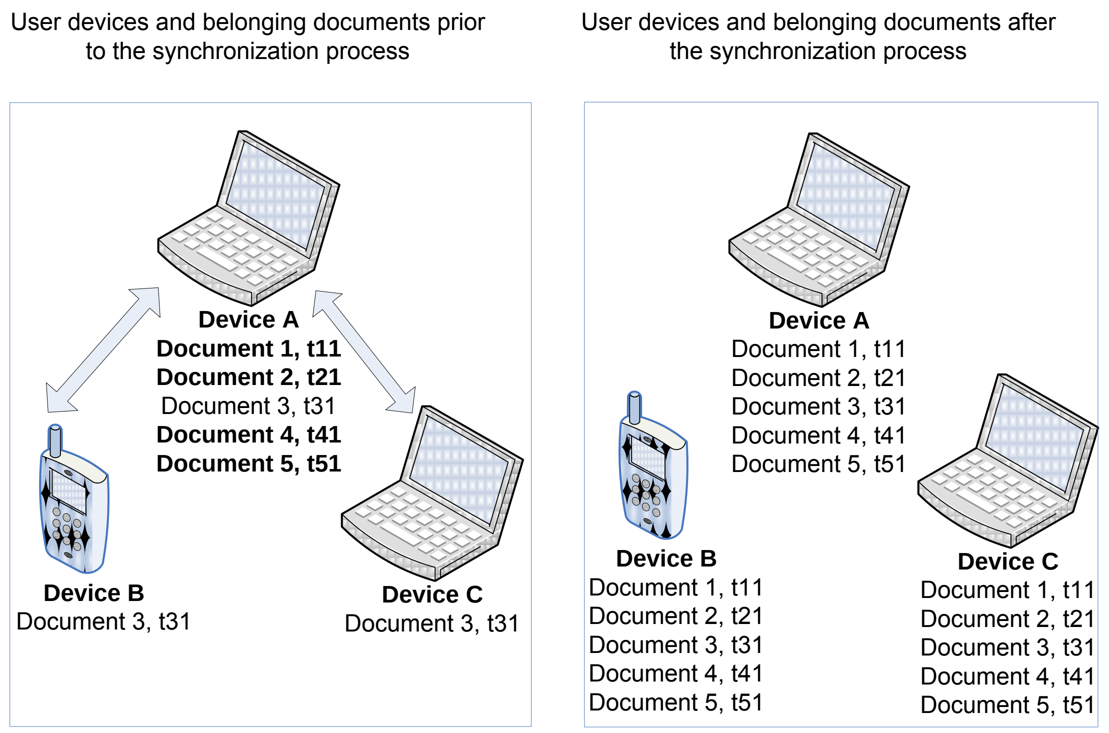

Figure 4.2: Efficiency of timestamp usage according to the proposed paradigm: after Devices B and C successively synchronize with Device A, all replications presenting the same document on different devices are marked with the same timestamp value. If now, Devices B and C would try to synchronize, no documents would be transmitted over the network as there are no differences between these two data sets and the problem of inefficient communication cost is avoided.

Usage of multiple status flags enables devices to synchronize with a different device each time but the synchronization procedure still does not scale with network size due to time and memory problems of maintaining a vast number of flags. Algebraic set reconciliation protocols do not need to maintain any state information about other devices in the network, but they are computationally intensive.

We presented a new synchronization paradigm which enables efficient synchronization even when connection bandwidth, user device processing capabilities and memory resources are very limited. According to the new paradigm, different replications of a particular document have all the same unique global identification mark. During the synchronization process, valid replications are determined according to their timestamp value, which presents the time of the last document modification. Using timestamps and global identification at the same time, the proposed paradigm enables simple and exclusive procedures for reconciliation of different replications of one document. For the proposed paradigm implementation purposes we suggest the usage of the open standard SyncML. Including the proposed GDID for document identification and timestamps for replication differentiation, reported SyncML weaknesses can be avoided. If the GDID is really unique, no mapping tables are required and the devices need not maintain modification information about each of its records with respect to each other device. Consequently, the memory problem is avoided and the protocol can be applicable to larger networks as well. At the same time, the protocol remains simple and efficient from the computation point of view.

REFERENCES

[1] Argawal, S. & Starobinski, A. & Trachtenberg, A. (2002) "On the Scalability of Data Synchronization Protocols for PDAs and Mobile Devices", *IEEE Network*, Vol.16, p22-28.

[2] Minski, Y. & Trachtenberg, A. & Zippel, R. (2003) "*Set Reconciliation with Nearly Optimal Communication Complexity*", *IEEE Transactions on Information Theory*, Vol. 49, Issue 9, p2213-2233.

[3] Starobinski, D. & Trachtenberg, A. & Argawal, S. (2003) "Efficient Data synchronization", *IEEE Transactions on Mobile Computing*, Vol. 2, No. 1, p40-51.

[4] Jönsson, A. & Novak, L. (2001) "SyncML-Getting the mobile Internet in sync", *Ericsson Review,* No.3

[5] Open Mobile Alliance, "SynML specifications", *http://www.openmobilealliance.org/tech/affiliates/syncml/syncmlindex.html.*

[6] Intellisync web resources *http://www.intellisync.com*

[7] Palm HotSync web resources http://www.palm.com/uk/en/software