MultiRep - Asynchronous Multi-Device Consistency

João Ribeiro, João Barreto and Paulo Ferreira
INESC ID / Instituto Superior Técnico / Technical University of Lisbon
joao.f.ribeiro@ist.utl.pt, joao.barreto@ist.utl.pt, paulo.ferreira@inesc-id.pt

ABSTRACT

Nowadays, people increasingly use multiple devices to manage and share information anywhere anytime. Users are increasingly spreading large sets of files and folders among several devices. Since users cannot know at one device which files/folders are stored by other devices, data management across multiple devices has become a very difficult task. For instance, when one user needs an object that is not stored on the device being used, he/she needs to manually explore the entire object collection, which is spread across multiple devices. Additionally, different versions of files are created on multiple devices raising a consistency problem. Most current solutions ensure data management and consistency across devices through central servers or Internet services. Since portable devices have intermittent network connection or no connection at all to access these services, it is essential to take advantage of proximity between devices to synchronize data among them. This paper introduces MultiRep, a single-user file synchronizer middleware that provides the user with information about the location of files and folders stored on multiple devices. MultiRep is a totally decentralized system based on optimistic replication. It ensures eventual consistency among multiple devices through pairwise interactions, reporting all relevant information about conflicts to the users.

Keywords: Conflict Resolution, Eventual Consistency, File Synchronizer, Metadata Management

1. INTRODUCTION

In the last few years, people own more and more multiple devices to store, manage and share information. At the same time, mobile devices (e.g. PDAs, smartphones, laptops) are becoming more computationally powerful, with greater memory and networking support capacity. Due to this fact, people increasingly use them for entertainment and to perform work that in the past was only possible with a desktop PC.

Due to the increasing number of personal devices, user's data is scattered throughout several devices. This data easily becomes disorganized because there is no way to let users know which data is stored by each device. For instance, imagine that one user takes several photos during a trip. Then, he/she copies different sets of photos to different devices. How can he/she find one specific photo? How can he/she know which photos exists? And where are they? Currently, to answer these questions, users need to manually explore the entire data collection, which is spread across multiple devices. Additionally, consistency problems begin to arise when one user modifies the same files concurrently on multiple devices.

Many systems ensure data management and consistency among multiple devices by storing all files/folders on a central server or online storage web service. These approaches have important limitations. For instance, some devices (e.g. digital camera) may not have network access to a central server or Internet service. Additionally, when traveling long distances, communication with a local device is often easier, more efficient, and more cost effective than synchronization over long-distance links. Also, scalability and availability issues arise when we are using systems based on a central server. For instance, if the number of user's devices increases too much, the central server becomes a bottleneck. Moreover, when the central server becomes unavailable, devices cannot synchronize with each other.

Currently, there are many systems allowing users to synchronize files/folders between multiple devices. These systems typically can be divided into two groups: 1) online file synchronizers, such as Dropbox [1], which use cloud computing to store and share data among devices; 2) offline file synchronizers, such as GoodSync [2] and Microsoft's Briefcase [3], which allow offline synchronization of files between different devices without requiring access to Internet services. In the first group, the file synchronization mechanism only works using an Internet connection. Due to this fact, these systems do not take advantage of the proximity of devices to share data among them. Also, user's data is stored in an unfamiliar environment not controlled by the user, thus raising privacy issues. The second group has several drawbacks related to data management and network flexibility. For instance, Microsoft's Briefcase does not allow file synchronization between all devices in the system that share the same files and folders. Also, like other systems, it does not provide any information regarding the location of files and folders stored on several devices. Thus, when a user needs an object that is not stored on the device being used, he/she needs to manually explore the entire object collection which is spread across multiple devices.

The main goal of this work is to design and build a system called MultiRep that takes advantage of devices proximity to synchronize data among them, ensuring data consistency in a decentralized fashion. Furthermore, MultiRep allows users to see at any device anytime which files and folders are stored by other devices even if they are turned off or inaccessible. In order to achieve this, MultiRep stores on each device the metadata of all devices in the system. This approach faces three challenges: 1) How can we ensure that devices with low memory capacity are able to keep a copy of all metadata in

M-MPAC'2011, December 12th, 2011, Lisbon, Portugal.

the system? 2) How is the metadata of a device distributed among other devices? 3) How to ensure eventual consistency of metadata spread across several devices? The first challenge is the most critical one. In fact, MultiRep does not guarantee that the smallest device in the system is able to store metadata of all other devices. Nevertheless, MultiRep employs efforts to minimize this problem. First of all, it separates metadata from content. Thus, only metadata of files and folders are propagated between devices, instead of propagating the whole content of files. Moreover, MultiRep provides users with an option to disable the synchronization of metadata stored by other devices. MultiRep addresses the second challenge by synchronizing metadata of all devices every time users start the synchronization of files and folders between any pair of devices. For instance, imagine that one user synchronizes files/folders between two devices *D1* and *D2*. Next, he/she synchronizes the same files/folders between *D1* and other device *D3*. Since all metadata is synchronized between *D1* and *D3*, the device *D3* now knows which files and folders are stored by *D2*. The last challenge is solved by employing version tracking mechanisms based on version vectors[5] to the metadata of each file/folder.

Using MultiRep is very simple. First of all, a user creates a special folder, called *briefcase*, as he would create a normal folder. Then, he copies all files to be replicated into the *briefcase*. Finally, he copies the *briefcase* folder to other devices in which he wants to keep files synchronized. When a given *briefcase* is copied between two devices, MultiRep creates a *synchronization pair* between those devices. A *synchronization pair* is an association between two devices that are able to synchronize one *briefcase*. Figure 1 illustrates several *synchronization pairs* created in MultiRep at different times between devices closest to the user.

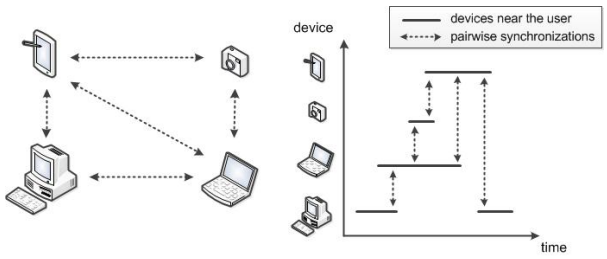

Figure 1: A typical set of devices and communication links. On the right, horizontal bars represent devices near the user and the vertical dotted lines represent the synchronization pairs created at a given time between devices.

The evaluation of MultiRep system shows that the memory footprint of MultiRep prototype is very reasonable, even when storing metadata of three briefcases with 4096 folders and 16384 files. Additionally, results show that the metadata synchronization of one *briefcase* has no impact on the speed of data synchronization in comparison with other systems such as Microsoft's Briefcase.

The rest of this paper is organized as follows. Section 2 presents the MultiRep's architecture. Section 3 describes how conflicts are handled by the system. Section 4 presents the obtained results of the evaluation performed to the implemented solution. Finally, Section 5 describes the related work.

2. ARCHITECTURE

This section presents the architecture of the proposed file synchronizer, named MultiRep. This system is a peer-to-peer middleware that allows a single-user to keep files and folders replicated and consistent among multiple devices. It is based on optimistic replication [6] using mechanisms based on version vectors [5] to track changes to files and folders on several devices. With these mechanisms, MultiRep detects conflicts and determines the set of updates to be exchanged during synchronization sessions. MultiRep allows users to see at any device which files and folders are kept by other devices. To achieve this, it does not need to store the entire data collection on each device. Instead, it stores on each device the metadata of all other devices.

2.1 Layers and Modules

The MultiRep's architecture illustrated in Figure 2 is divided into 4 main layers: GUI Layer, Consistency Layer, Monitorization Layer and Network Layer. Each layer is composed of several modules.

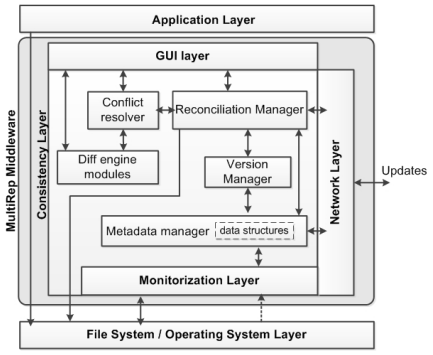

Figure 2: MultiRep's architecture.

The main layers and modules are described below:

- Monitorization Layer - This layer is responsible for reporting modifications performed by users and applications in the file system to the Consistency Layer.
- Consistency Layer - This layer ensures the eventual consistency of files and folders stored inside *briefcases*. To achieve this, it is composed of the following modules:
  - Reconciliation Manager - module that implements the synchronization phase. It detects all conflicts that occur during this phase.
  - Metadata Manager - module responsible for storing and managing all metadata required for the synchronization phase. It also maintains the metadata of all *briefcases* stored in devices previously synchronized.
  - Version Manager - module that implements the version vector mechanism for tracking changes to data on multiple devices.
  - Conflict Resolver - module responsible for the conflict resolution phase.
  - Diff Engine Modules - this module is used to detect the differences between the contents of files in conflict.
- Network Layer - This Layer allows communication between multiple instances of MultiRep running on different devices.

2.2 Data Structures

The main data structures required for the synchronization phase are organized as follows:

- File and folder structures - objects that store metadata of files and folders stored inside *briefcases*. Each of these structures has an associated list of version vectors for tracking changes to data, as well as a global unique identifier for recognizing a given file/folder within a *briefcase* regardless its name. Folder structures contain a list of file structures.
- Directory tree structure - a tree representing the hierarchical structure of files and folders inside *briefcases*.

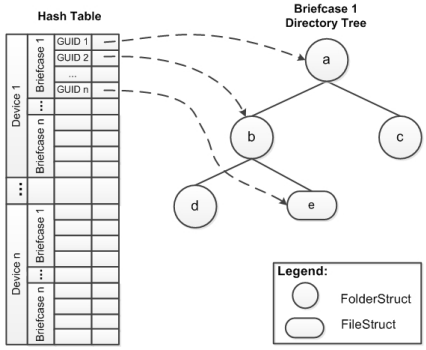

Figure 3: Hash directory tree structure.

Directory trees are used to compare which files and folders are new, renamed or deleted during synchronization. These trees are composed of file and folder structures.

- Device structure - used to store relevant information about devices, such as device's name, device's type and device's network information. It also maintains metadata of all *briefcases* stored by the device. Each device structure has a global unique identifier.
- Hash directory tree structure - a data structure that provides other modules with directly access to any file/folder structure stored inside any briefcase and any device (Figure 3). Additionally, it is used to detect when a file/folder was created inside a given briefcase during the synchronization phase.

2.3 Version Tracking

A version vector [5] is a vector of n entries, where n is the number of devices holding the replica of data object associated with the vector. Each entry holds an integer counter. Intuitively, the version vector's counter of each data object counts the number of updates performed by each device to that object. In MultiRep, the two types of data objects are files and folders. Each file and folder has two version vectors: 1) Global Version Vector (GVV) - used to track all updates performed to files/folders; 2) Rename Version Vector (RVV) - used to track updates performed only to the name of files/folders.
Before describing update rules, it is important to present some definitions:

Definition 1 One vector *vv1 dominates vv2* if all entries related to every device in *vv1* are greater or equal than corresponding entries in *vv2*.
Definition 2 One version vector *vv1* is *compatible* with *vv2*, if *vv1 dominates vv2* or *vv2 dominates vv1*.
Definition 3 The *merge* operation returns a version vector which has for each entry the maximum value of the corresponding entry in the input version vectors.

Update rules define how version vectors are updated according to the operations performed to files/folders stored inside a given briefcase. In MultiRep, there are 4 operations that can be performed to files/folders: 1) Create - create a file/folder inside a given folder; 2) Delete - remove a file/folder from a given folder; 3) Rename - change file/folder's name; 4) Write/Modify - write a file means change its contents (we use the term write to emphasize the difference of modifying a file and modifying a folder). A folder is modified when a file or folder inside it is created/renamed/deleted. The following update rules are applied by each device independently.

Update Rules for GVV:

R1. Initially all counters are zero - for all $i \in GVV$ do $GVVi \leftarrow 0$;

Folders update rules

R2. Every time a device Di creates/renames/deletes a folder Fo, the counter i on GVV of Fo is incremented;

R3. Every time the counter i on GVV of Fo is incremented, the counter i on GVV of Fo's parent folder is also incremented;

Files update rules

R4. Every time a device Di creates/renames/deletes/write a file Fl, the counter i on GVV of Fl is also incremented;

R5. Every time the counter i on GVV of Fl is incremented, the counter i on GVV associated to the folder that contains Fl is incremented.

Update Rules for RVV:

R6. Initially all counters are zero - for all $i \in RVV$ do $RVVi \leftarrow 0$;

R7. Every time a device Di renames a file/folder Fl/Fo, the counter i on RVV of Fl/Fo is incremented.

As a result of applying the above rules we have the following invariant - global version vectors of folders and files never *dominate* the global version vector of their parents' folders.

During synchronization, a *merge* operation is applied between all version vectors of files/folders being synchronized for timestamping their new version.

2.4 The Synchronization Phase

MultiRep's synchronization is always performed between two devices: the local device (in which the whole synchronization process takes place) and the remote device. In general, the device that starts synchronization is the one that performs all steps required for its conclusion. This is not true, however, if a mobile device starts briefcase synchronization with a desktop PC. In this case, the device that will perform the whole synchronization is the last one.

MultiRep's synchronization is divided into three main phases: version vector synchronization, metadata synchronization and briefcase synchronization. Only the last two phases can be manually triggered by the users. Every time users start the briefcase synchronization, MultiRep automatically performs synchronization of all metadata kept by two target devices. This is done to accomplish that all metadata kept by one device is globally replicated among all devices in the system. Therefore, one device *D1* can know which metadata is stored by another device *D3* as a result of performing briefcase synchronization with another device *D2*. The previous synchronization phases are described bellow.

Version Vector Synchronization: The version vector synchronization uses the update rules defined in Section 2.3 to synchronize files/folders version vectors stored by local and remote device. This phase is performed during metadata and briefcase synchronization.

Metadata Synchronization: Metadata synchronization provides other modules with two options for synchronize metadata: 1) synchronize a given briefcase's metadata; 2) synchronize all metadata. In the first one, only metadata of briefcases selected by the user are synchronized with the remote device. In the second one, metadata of briefcases physically held by the local device are synchronized, as well as all metadata of other briefcases that are only known by the remote device. The last phase supports the metadata-everywhere approach, ensuring eventual consistency of all metadata globally replicated between devices being synchronized. Metadata synchronization is divided into two phases:

1) Briefcase pairs synchronization - responsible for synchronizing all briefcase synchronization pairs created between devices when copying same briefcase through multiple devices;

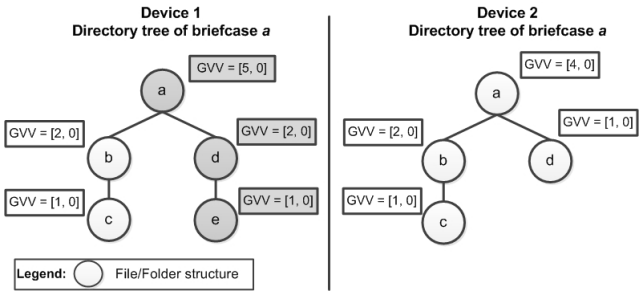

Figure 4: Example of two directory trees associated to briefcase *a* stored in device 1 and 2. In this case, device 1 is responsible for the whole synchronization phase. Grey nodes are the nodes explored by the tree exploration procedure. White nodes are ignored during this procedure.

2) Directory tree synchronization - responsible for synchronizing directory trees of briefcases. During this phase, devices only exchange file/folder structures of directory trees that have been modified since the previous synchronization session between these two devices. To achieve this, MultiRep explores the two directory trees of briefcases being synchronized comparing Global Version Vectors (GVV) of each file/folder structure stored in local and remote directory tree. As we can see in Figure 4, directory tree synchronization only explores nodes that have been modified since the last synchronization session between these trees. If MultiRep finds a node that was not modified, it does not explore the subtree associated to that node. One node is modified when its GVV is not equal to the corresponding GVV of another node stored in another directory tree.

Briefcase Synchronization.

The briefcase synchronization is responsible for synchronizing data and metadata between two selected briefcases. It uses metadata synchronization to update directory trees of the two target briefcases. After updating metadata of briefcases, the **data synchronization** begins.

Data synchronization receives the two updated directory trees of briefcases, merging all modifications performed to them in both devices. To do this, it is divided into two phases: folder reconciliation and file reconciliation. The first one only synchronizes modifications performed to folders, unlike the second that only synchronizes modifications performed to files. In the first phase, all nodes containing modified folder structures are explored. Folder reconciliation detects which folders structures were modified by comparing their global version vectors. One folder is modified when its global version vector is not equal to the corresponding version vector of other folder. If both version vectors are equal, i.e., there is no modification to that folder, the folder reconciliation stops the tree exploration on that node. Otherwise, three steps are performed in the following order:

- resolve folder deletions - verifies if a certain folder was deleted by a given device by analyzing which folder is marked as deleted in the corresponding folder structure;
- resolve folder renames - verifies if a certain folder was renamed by a given device by comparing the Rename Version Vector, as well as the name of the folders;
- resolve folder creations - detects if a certain folder was created by a given device by verifying which folders structures are stored by both directory trees;

File reconciliation begins after folder reconciliation. The actions performed in this process are similar to the ones employed during the previous phase. File reconciliation handles files deletions, renames, creation and modifications by this specific order. As it happens in folder reconciliation, all conflicts detected during this phase are displayed to the user. Then, he/she can choose to resolve them or postpone their resolution. If he/she chooses to resolve conflicts, the synchronization process successfully ends, otherwise it is aborted.

3. CONFLICT RESOLUTION

During synchronization, several types of conflicts can be detected between files/folders of two devices being synchronized, as well as between all other devices in the system. All conflicts between devices being synchronized can be resolved during synchronization, unlike conflicts between other devices that can only be resolved when users synchronize directly with devices in conflict. The conflict detection with devices that are not participating in the synchronization process is supported by the metadata-everywhere approach.

Conflicts between two folders/files are detected using version vectors. One file/folder *f1* is in conflict with other file/folder *f2*, if the Global Version Vector (GVV) of *f1* is not *compatible* with GVV of *f2*.

The following conflicts are detected by MultiRep and must be manually resolved by the user:

- Creation - create two files at the same path with the same name on different devices. In this case, Global Version Vectors of the two files are not *compatible*;
- Rename/Rename - change the name of one file/folder to different names on different devices. In this case, Global Version Vectors and Rename Version Vectors of the two files/folders are not *compatible*;
- Delete/Rename - delete a file/folder on one device and change its name on other device. In this case, Global Version Vectors of the two files/folders are not *compatible*;
- Delete/Modification - delete a file/folder on one device and modify it on other device. In this case, Global Version Vectors of the two files/folders are not *compatible*;
- Modification - change contents of the same file on different devices. In this case, Global Version Vectors of the two files/folders are not *compatible*.

When these conflicts occur, users can choose to resolve them or postpone their resolution. Several options are provided for conflict resolution, such as: keep one of the files/folders in conflict; or view the content of each file side by side.

MultiRep automatically detects and resolves other concurrent modifications to files/folders such as:

- Create two folders with the same path and same name on different devices - MultiRep merges the contents stored by the two folders on both devices;
- Change the name of one file/folder to the same name on different devices - MultiRep does not detect this case as a conflict and therefore does not warn the user;
- Deletes the same file/folder on different devices - MultiRep does not detect this case as a conflict and therefore does not warn the user;
- Create/rename/delete/modify different files/folders stored inside the same folder on different devices - MultiRep updates those files/folders on both devices;
- Change content of a file on one device and renames it on other device - MultiRep renames the file on the first device and updates its content on the second device.

When two files/folders in conflict are resolved, a new version of those files/folders are created in the two devices being synchronized. The new version covers all previous conflicting versions. It is calculated by applying a *merge* procedure to both version vectors of conflicting files/folders. Due to this fact, MultiRep ensures a deterministic conflict resolution, i.e. a conflict resolution performed in the same way between any pair of devices.

4. EVALUATION

In this section, we present the results collected from the evaluation of MultiRep. MultiRep was implemented using Microsoft Visual Studio 2010, C# 4.0 and the 4.0 .Net

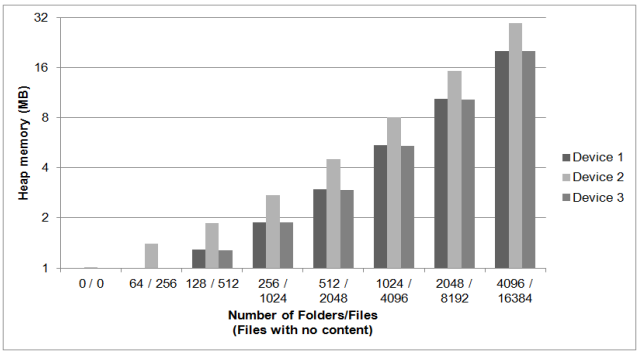

Figure 5: Synchronization of one briefcase between all devices. In this case, the option for synchronizing all metadata was disabled.

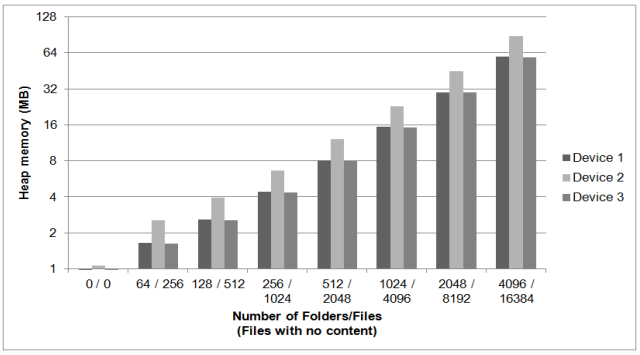

Figure 6: Synchronization of one briefcase between all devices. In this case, the option for synchronizing all metadata was enabled.

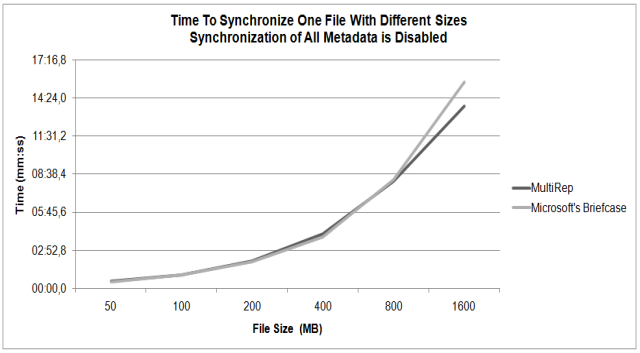

Figure 7: Comparison between Microsoft's Briefcase and MultiRep w.r.t. the speed of synchronization.

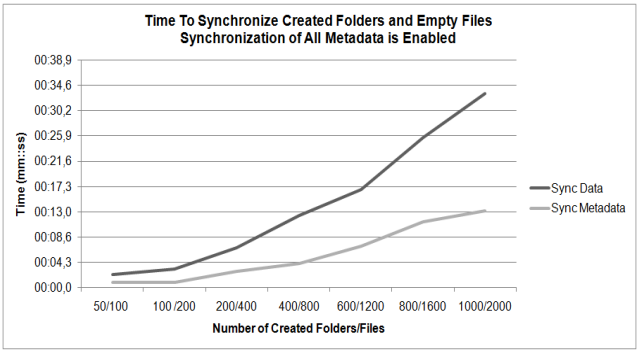

Figure 8: Comparison between speed of data synchronization and metadata synchronization.

Framework. The solution was tested under Windows Vista and Windows 7. The same work load was used to evaluate and compare MultiRep with Microsoft's Briefcase.

4.1 Methodology

To evaluate the system, three dedicated devices were required (see Table 1). As we can see in Figure 9, the communication between these devices was performed by using temporary wireless ad-hoc networks. Every time it was required to synchronize briefcases between two given devices, a 54Mbps Wi-fi ad-hoc network was created between those devices. The followings steps describe in more detail the

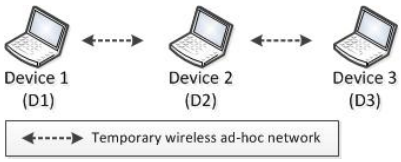

Figure 9: Test environment

	Device 1	Device 2	Device 3
CPU	Intel Core Duo T2250 @ 1.73 GHz	Intel Core i7 740QM @ 1.73 GHz	Intel Dual CPU T3200 @ 2.0 GHz
Memory	2559 MB	4021 MB	2048 MB
OS	Windows 7 x86	Windows 7 x64	Windows Vista x86

Table 1: Hardware and software specifications of the three devices used in MultiRep's evaluation.

evaluation procedure to measure the memory footprint of MultiRep: 1) Create one briefcase on device *D1*, copy it to device *D2* and then copy it from *D2* to *D3*; 2) Create N files and M folders inside *briefcase* of device *D2*, where N and M are parameters of our experiments; 3) Synchronize *D2* with *D3*; 4) Finally, synchronize *D1* with *D2* and *D1* with *D3*. To measure the speed of MultiRep's synchronization the same steps were performed, but with only two devices *D1* and *D2*.

4.1.1 Workload Description

To obtain the memory footprint of MultiRep, an increasing number of folders and files were created inside one briefcase. In each step of the evaluation procedure, the number of folders were multiplied by two. In this case, each folder stored 4 files. Empty files were created since the metadata of each file does not depend on its content.
To measure the speed of the synchronization process, an increasing number of folders and files were created inside one *briefcase*. Each folder stored 2 files. All files have the same size. Initially, 50 folders and 100 files were created. In this case, the work load has a total size of 50MB. So, the medium size of each file is 512KB since each file has identical size. The size of each file depends on the number of files created and the total size of work load. The work load was multiplied by two in each step of the evaluation procedure. In order to obtain correct results, all evaluation procedures used to measure the memory usage and speed synchronization were repeated five times for each test. With the collected results an average was calculated. The average was used to create the graphs displayed in the following sections.

4.2 Memory usage

The memory usage is the most important evaluation parameter of MultiRep since each device stores all metadata of other devices. The obtained results show that the memory usage by the application is proportional to the amount of metadata stored by devices. These results can be seen by comparing Figures 5 and 6. In Figure 5 the synchronization of all metadata was disabled and so each device only stores metadata of its briefcase. On the other hand, in Figure 6, the synchronization of all metadata was enabled and

thus each device stores the directory tree of all other devices corresponding to the briefcase being synchronized. As we can see, when enabled the synchronization of all metadata the amount of memory used by application almost triples. Nevertheless, the memory footprint of MultiRep is reasonable even when storing metadata of three briefcases each one with 4096 folders and 16384 files. For this number of folders/files, device 2 based on a 64-bit processor uses more 29MB of memory than devices 1 and 3 that have inside a 32-bit processor. This happens because all registers and pointers in the x64 architecture were expanded from 4 bytes to 8 bytes.

4.3 Performance

W.r.t the MultiRep's performance, the results showed that the metadata synchronization of one briefcase has no impact on the speed of data synchronization in comparison with other systems such as Microsoft's Briefcase (see Figure 7). In fact, MultiRep is more fast than Microsoft's Briefcase. For instance, in case of transferring 5000 folders and 10000 files, Microsoft's Briefcase takes more 3 minutes than MultiRep. Moreover, as we can see in Figure 8, the speed of metadata synchronization directly affects the speed of data synchronization. For instance, in case of synchronizing 1000 folders and 2000 empty files, data synchronization takes 39% of the time on metadata synchronization phase when the option to synchronize all metadata is enabled.

5. RELATED WORK

This section describes several solutions that already address the issue of replicating content throughout several devices.

Eyo [7] is the system most closely related to MultiRep. It is a single-user metadata-everywhere storage system based on the idea that user's data should be managed transparently by the user from any of his devices. Like MultiRep, Eyo aims to provide on each device a global consistent view of all user's data objects distributed among devices. Unlike MultiRep, Eyo allows users to change metadata of all devices at any device. In this case, an overlay network is used to send frequent metadata updates. When compared to MultiRep, Eyo has important limitations. For instance, in Eyo, applications must use the Eyo's storage API to manage their data. Due to this fact, legacy applications must be modified in order to use Eyo. Also, the conflict resolution procedure is the sole responsibility of applications. Moreover, Eyo uses an overlay network to manage all inter-device communication unlike MultiRep that does not require any.

Roma [8] is a personal metadata service to locate current versions of personal files and ensure their availability across devices. In order to achieve this, unlike MultiRep, it uses a central metadata server to store information about files kept by all devices in the system. Due to this fact, Roma has some limitations. For instance, the central server must be always available to all devices despite intermittent network connectivity. To cover this problem, Roma argues that the metadata server should be a highly portable device that commonly is closest to the user. Nevertheless, we know that mobile devices have concerns w.r.t the battery life, which can be dramatically reduced by the several metadata updates and requests of other devices. Additionally, there is no guarantee that this portable device is always close to the user. Moreover, in order to use Roma, applications must be modified to communicate with the metadata server and to take advantage of metadata information stored by the metadata server. Finally, Roma does not specify any information regarding how conflicts are handled by the system.

Footloose [4] is a single-user replication system that aims to provide a user-centered data store that can share data and reconcile conflicts across multiple devices. Like MultiRep, Footloose is a peer-to-peer middleware that uses pairwise synchronization to enhance data consistency between devices in the system. Unlike MultiRep, Footloose does not provide any information to the user regarding the location of files and folders stored by multiple devices. In addition, applications have to be modified to use Footloose. Moreover, Footloose's conflict resolution is an entirely application-based process, i.e. applications must know how to resolve conflicts and in some cases must implement their own data structures to detect other conflicts.

Microsoft's Briefcase [3] is an offline file synchronizer that was introduced in Windows 95. With Briefcase, when users want to replicate files from a device to another, they just need to create a *briefcase* folder then drop target files into that *briefcase* and finally copy it to another device. Briefcase's file synchronization mechanism has several limitations. First, it does not propagate the creation of folders until they have a file inside. Second, it reports to the user too few information about conflicts detected during synchronization. Third, if a folder/file is renamed, Briefcase is not able to detect if it is still the same folder/file. In most cases, it splits the folder/file from the original, rendering it an orphan. Additionally, Briefcase has some limitations w.r.t. synchronization between multiple devices. For instance, imagine that a user copies a *briefcase* folder with the same contents from a device A to devices B and C. Devices B and C can synchronize files and folders with device A. However, they cannot synchronize such files/folders between them. Furthermore, Microsoft's Briefcase does not provide to the user any information about files and folders stored by other devices.

Online file synchronizers, such as Dropbox [1], allow users to share files/folders with others across the Internet. They typically use cloud storage services to store data and synchronize files/folders between multiple devices. Although these systems offer good flexibility in a scenario where one user has several devices connected to the network at the same time, they have important limitations. For instance, they depend strongly on the Internet connection to share updated resources without taking the advantage of physical device proximity. Moreover, user's data is stored in an unfamiliar environment not controlled by the user, thus raising privacy issues.

6. REFERENCES

[1] Dropbox: Secure backup, sync and sharing made easy. https://www.dropbox.com accessed on 29/10/2010.

[2] Goodsync: Backup or synchronize your files easy, fast, automatic and completely reliable. http://www.goodsync.com/support/manual accessed on 25/11/2010.

[3] Microsoft. How to use the briefcase feature in windows xp http://support.microsoft.com/kb/142574/ accessed on 24/11/2010.

[4] J. Paluska, D. Saff, T. Yeh, and K. Chen. Footloose: A case for physical eventual consistency and selective conflict resolution. *IEEE WMCSA*, 2003.

[5] D. Parker, G. Popek, G. Rudisin, A. Stoughton, B. Walker, E. Walton, J. Chow, D. Edwards, S. Kiser, and C. Kline. Detection of mutual inconsistency in distributed systems. *Transactions on Software Engineering*, 1983.

[6] Y. Saito and M. Shapiro. Optimistic replication. *ACM Computing Surveys (CSUR)*, 37(1):42–81, 2005.

[7] J. Strauss, C. Lesniewski-Laas, J. Paluska, B. Ford, R. Morris, and F. Kaashoek. Device transparency: a new model for mobile storage. *In: Proceedings of the SOSP Workshop on Hot Topics in Storage and File Systems (HotStorage)*, 2009.

[8] E. Swierk, E. Kiciman, N. Williams, T. Fukushima, H. Yoshiday, V. Laviano, and M. Baker. The roma personal metadata service. *MONET special issue of best papers from WMCSA 2000,*, 7(5), October 2002.