Backup Synchronization on Different Operating Systems

[1]Boshra T. Haghighi, [2]Azlan Iqbal
College of Information Technology, University Tenaga Nasional
Jalan IKRAM-UNITEN, 43000 Kajang, Selangor Darul Ehsan, Malaysia
E-mail: [1]boshra@uniten.edu.my, [2]azlan@uniten.edu.my

Abstract – **The processes of backing up and synchronizing computer data are common. Backups may be retained in multiple computers or storage devices to protect data from risks such as hard disk failure, computer theft or even accidental deletion. There are many software applications which are designed for synchronizing backups but few of them are able to work across different operating systems and platforms. The undertaken policies and methods in these applications make the common synchronizing process time-consuming. Moreover it is not easy to manage large-sized backups across relatively slow networks like the Internet. In this research we present a systematic model which is able to synchronize large sized backups, in terms of mirroring the name and quantity of the files, across different operating systems. Time and cost efficiency are two main factors in this study. We present a meta-analysis and review of the literature. Several software applications with similar capabilities are reviewed in which their strengths and weaknesses have been analyzed and synthesized to achieve the objectives of the research. Computer data and the associated backups in both source and target locations will be checked for synchronization. However, the model does not check for file modifications because this may affect the speed of the sync-ing process. This study presents a meta-analysis of selected literature and also develops a synchronization application for large-sized backups which work across different operating systems.**

Keywords - Synchronization; backup; operating system; systematic model; synchronization application

I. INTRODUCTION

The process of archiving computer data is referred to as ‘backing up’ which helps users to restore their corrupted data or to recover data from an earlier time. Users may use multiple computers or different storage media to store their backups, such as external hard drives. Once the archived data is changed or new data is created, backups have to be updated as well. The file synchronization process is commonly used to ensure all the files are updated in the backup’s replica(s). Some challenges such as speed, cost, conflict detection, and compatibility lead to different synchronizing products which are competing in the market. One of the challenges involved in the concept of backup synchronization is Operating System(s) (OS) incompatibility. Backups on multiple computers with different OS can be updated by synchronizer applications (see section II). In this research we aim to review and compare synchronizer applications which are compatible with different OS and propose similar systematic synchronizing methods with respect to the reviewed constraints. The proposed model is designed for synchronizing large-sized backups of scanned research articles across a desktop PC (Windows 7) and an iPad (iOS). The model focuses on mirroring the duplicates of the backups across different OS with the focus on the speed and cost efficiency.

II. LITERATURE REVIEW

There are some open source file synchronizer applications which synchronize the files by a renaming or a moving process, such as Unison[1], DirSync Pro[2], LuckyBackup,[3] rsync[4]. One of the common advantages of these applications is their compatibility with different OSs. These applications are able to detect the renamed/moved files and synchronize the source and target location(s) accordingly. One of the notable differences between these applications is the speed of the synchronizing process for files with large sizes. The speed of synchronizing process depends on many different factors such as setup time, network bandwidth, and policies employed to replicate either the entire file(s) or portions of the file(s). Some of these applications have been briefly reviewed in the immediate sections (A to D) as follows:

A. Unison

Unison is an open source file synchronization application which is designed for Windows and Unix-like OS such as Linux, Mac OS X and Solaris under GNU public license [2]. It synchronizes all the files which are stored on different hosts (or different disks at the same host) according to the modified replica. It works across any pair of computers (even with different OS) which is connected to the Internet through TCP/IP protocol. Unison is also flexible with the network bandwidth and slow links as it uses compression protocols to synchronize the large backups after small updates through slow connections. Updates in all the replicas in different directories of storage media will be transferred; however conflicting updates will be identified as well [2, 3]. Unison also has some feedback in the relevant forums or websites such as Ubuntu Geek which claim insufficient speed, unreliability and incompatibility problems. Moreover this open source application may not be available on smartphones (e.g.

[1] Unison File Synchronizer, University of Pennsylvania - Department of Computer and Information Science, Accessed June 2013, Available from http://www.cis.upenn.edu/~bcpierce/unison/lists.html
[2] DirSync Pro, Directory Synchronize Pro, Accessed June 2013, Available from http://www.dirsyncpro.org/
[3] LuckyBackup, Accessed June 2013, Available from http://luckybackup.sourceforge.net/index.html.
[4] Rsync, Accessed June 2013, Available from http://rsync.samba.org/.

iPhone) and tablet PCs (e.g. iPad) and needs some technical savvy and advanced planning for installation and file synchronization [4]. Unison hashes each file and check for any changes which is time consuming for large collection of files. Hence this application cannot be the best option for synchronizing two directories with large files to check for file names and modified dates.

B. rsync

One of the synchronizer applications with open source code under GNU Public License is rsync. rsync is designed for Windows and Unix-Like OS to synchronize files and backups. This application uses the 'rsync algorithm' to get the remote files for synchronization and minimizing the transferred data. Users can manipulate the code to suit their requirements. It can work through different platforms and be bundled into different applications. rsync is flexible in low bandwidth or slow link connections and supports multiple transport protocols as well as secured data transports. It speeds up the process of transferring large files in which only small changes have been made by sending only the updated portions. However some disadvantages have been identified for rsync such as difficulty to manage large-sized files and also an inefficient managing process for sparse files [5]. Moreover rsync is unable to handle undo and detect conflicting processes. For example if the user deletes one local file mistakenly, he/she cannot recover the deleted file from remote rsync backups [6].

C. DirSync Pro

DirSync Pro is another example of open source (written in Java) application which is designed for synchronizing unidirectional and bidirectional directories for Windows, Linux, Mac OS X and Java-based OS. Multiple synchronization policies can be set up and executed through its graphical user interface (GUI). This application provides the possibility of synchronizing a couple of directories with their contents by detecting any changes in the source. However DirSync Pro is not able to transfer data over a network and it is designed for synchronizing files in a single machine[5] [7]. Since we are usually advised to keep the backups on a separate machine, which is physically located elsewhere (in case of data lost threats such as fire and theft), it is important that the synchronizer applications can transfer data over a network.

D. LuckyBackup

Yet another example of an open source synchronizer application is LuckyBackup which is designed for Linux but it also has versions which work in Windows [8]. It is written in C++, based on rsyc. This application is able to recover lost backups in both source and target locations but it only checks for changes in the source. Also, it restores all the backups even after few changes have been applied; however restoring a part of a backup requires advanced planning [8].

E. Cloud-based Backups

Cloud-based backup services upload a copy of the backup files in a third party server in order to protect them against backup's disaster such as hard disk failure, computer theft or even accidental deletion. Cloud backup services provide a secure and safe way of backing up which also can be automated. Data will be stored on a remote server and it is easy to access them from anywhere. Hardware and software maintenance processes are not required and most of the plans are cheap. However cloud based services usually need high speed internet channels and large backups sizes are common. Most Wi-Fi packages tend to have monthly usage limits and regularly backing up gigabytes of data can take a lot of time and cost extra money. Restoring data with regards to the network bandwidth and speed can be slow. Also the safety of the third part server and private retention of data cannot be guaranteed [9, 10].

III. METHODOLOGY

In this research, we have chosen to do a meta-analysis of the relevant literature. To design the systematic synchronizing model, this methodology enables us to focus on gathered data and to contrast different studies. Aggregate data (AD) which is gathered through the literature reflects the strengths, weaknesses and relationships among the results. Synthesizing the AD from each study helps us identify the pattern and achieve its objectives.

IV. FINDINGS AND DISSCUSION

In the literature review section synchronizer applications and backup services were reviewed. Table 1 shows the synthesizing of AD and a comparison of advantages and disadvantages. Time-efficiency is one of the main challenges in the mentioned applications. The undertaken strategies and policies in the reviewed applications make the synchronizing process time-consuming. Since the check-up process for any updates, requires considerable time and also a suitable Internet channel, majority of them have insufficient speed for synchronizing the large sized backups. Moreover not all of them are able to work across the different OS since some of them are designed for single machine or are unable to transfer data through a network.

Considering large-sized backups, which have multiple replicas in different computers with different OS, the synchronizing process for simple check-up (such as name and quantity of the files) can be designed based on a new systematic model. This model focuses on large-sized backups on different platforms that require a simple synchronizing process. Hence the time-efficiency and compatibility factors would be the main concerns of this model.

[5] In order to sync files over a network DirSync Pro has to use NFS or SSH file systems.

Table 1. Advantages and disadvantages of the reviewed applications

	Unison	**rsync**	**DirSync Pro**	**LuckyBackup**	**Cloude-based Backup Services**
Advantages	• Supports different OS and platforms • Check for updates in both local files and replicas • Flexible with internet channels and connection speeds • Confilicting updates will be identified • Free	• Supports different OS and platforms • Delta encoding approach to minimize trasffered data • Flexible with internet channels and connection speeds • Free	• Supports different OS and platforms • Support GUI • Support multiple sync policies • Free	• Supports different OS and platforms • Support GUI • Able to recover backups for bosh local and remote replicas • Free	• Copy backups in a third party server • Automated backups • Accessible from anywhere • Hardware/software maintenance not reuired
Disadvantages	• Insufficient speed due to hashing process for each single file to check for updates • Unreliability • May not be available on iPhones & iPads • Time consuming for large backups for name and modified dates checkups	• Difficulty to manage large sized backups • Insufficient management for sparse files • Unable to undo a process	• Updates will be checked in the source and replicas will be synced accordingly • Unable to trasffer data over network • Desinged for sync process in single machine	• Updates will be checked in the source and replicas will be synced accordingly • Restoring part of backups required advance planning	• High internet channels required • Time consuming for restoring the backups with large size • Safety of third part server cannot be guaranteed • Private retention of data cannot be guaranteed

A. Proposed Model:

1. Sort the available files alphabetically, from A to Z in both source and target locations manually.

2. Separate files by file name (e.g. category initial letter start with A, category initial letter start with B and etc.) in both source and target locations.

3. Take the first file of the first category and insert it into a 'checking list'. This list keeps the name and size of the file being compared in each loop.

4. Compare the file in action from the 'checking list' against all other files from the same category in the target location. The model also considers similar file names to prevent duplication. For example, if a file from the source is named "File Name A" and a file in the target location is named "File Name –A", the model checks that these two files may be the same (Explained in step 8 to 12 of the example).

5. If more than one similar file (with the exact same name or similar name as explained in step 4) are found in the same category of the target location, their names and addresses are logged. To determine if the file is same, the size of the file will be checked. If the sizes are matched or not, in both situations it will ask the user's confirmation at the end of the process.

6. All compared file names will be logged into a 'checked list' so they are not checked again.

7. The next file is taken into the 'checking list' and steps 3 to 6 will be repeated again until all files from the category have been checked. Upon completing the comparison process in one category, the next one comes in action until all files from the source location are compared against the replicas in target locations.

8. The same process from steps 1 to 7 are repeated, but source and target's list are switched. So in the first round the source is compared against the target, and in the second round, the target is compared against the source. This step is designed to mirror the replicas in the source and target location(s) by checking for the changes in both local files and the replicas.

9. The model will display a log to the user if duplicate files with exact (or similar) name are found. It then asks users confirmation to check if the files are the same.

Figure 1 shows the visual representation of the proposed model:

B. Example:

As an example of the model's progress, consider the following two sets of data which are going to be synced:

Table 2. Example 1

No.	Source	Size	Target	Size
1.	"Glick, J. (2007). AI in the News (column) Vol 28. No 1.pdf"	200kb	"Glick, J. (2007). AI in the News (column) Vol 28. No 1.pdf"	200kb
2.	"Glick, J. (2007). AI in the News (column) Vol 28. No 2.pdf"	140kb	"Glick, J. (2007). AI in the News (column) Vol 28. No 3.pdf"	500kb
3.	"Glick, J. (2007). AI in the News (column) Vol 28. No 4.pdf"	500kb		

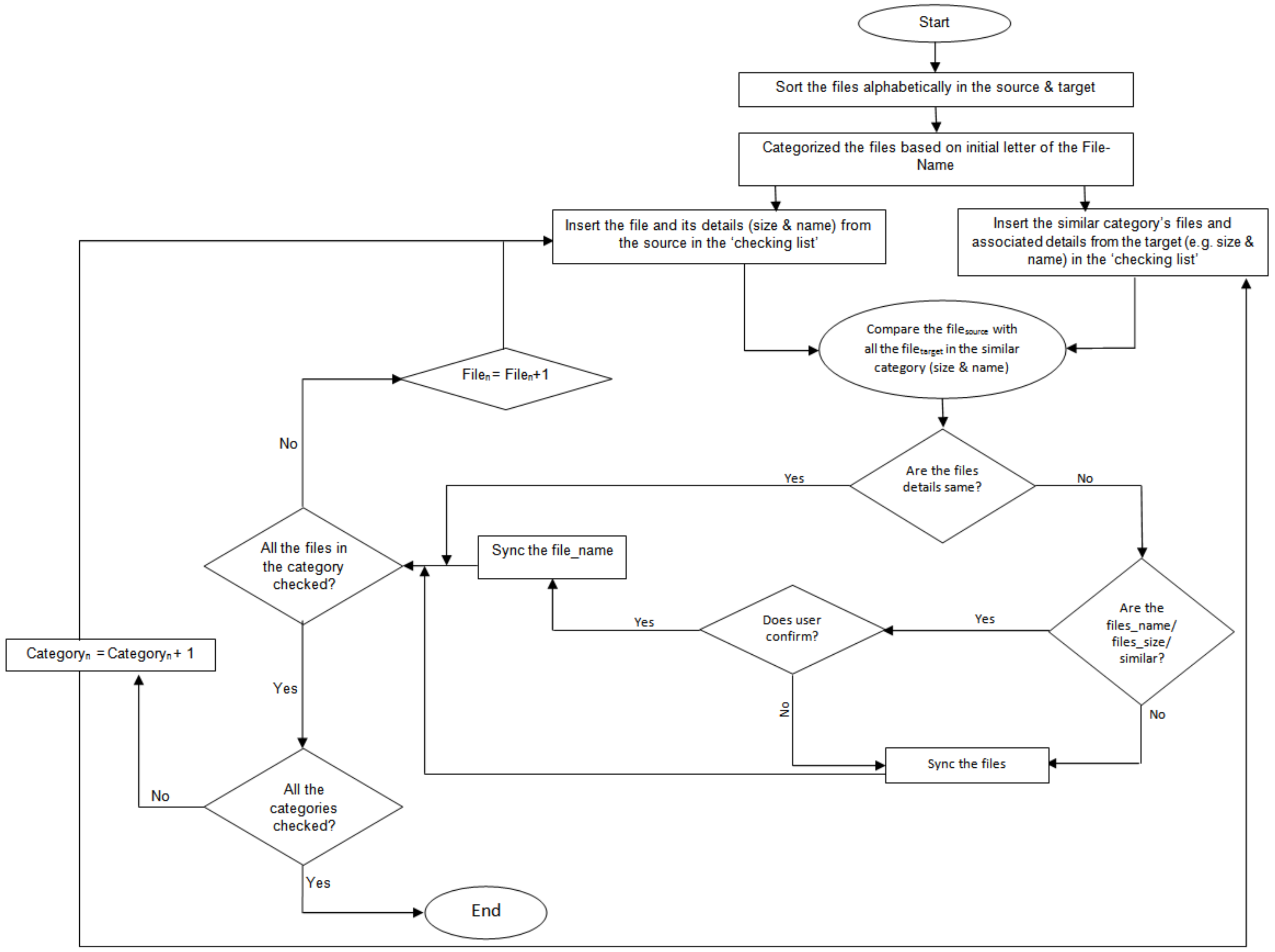
Start
Sort the files alphabetically in the source & target
Categorized the files based on initial letter of the File-Name
Insert the file and its details (size & name) from the source in the 'checking list'
Insert the similar category's files and associated details from the target (e.g. size & name) in the 'checking list'
Compare the file_source with all the file_target in the similar category (size & name)
Are the files details same?
Yes
No
File_n = File_n+1
No
All the files in the category checked?
Sync the file_name
Yes
Does user confirm?
Yes
Are the files_name/ files_size/ similar?
No
No
Sync the files
Category_n = Category_n + 1
Yes
No
All the categories checked?
Yes
End

Figure 1. Flow chart of the proposed model

1. Initially, the model will create a list such as the above table which includes the file sizes as well.

2. All files start with the letter "G", so there is only one category.

3. The first file named "Glick, J. (2007). AI in the News (column) Vol 28. No 1.pdf" from the source will be taken into the 'checking list'.

4. The file in action is compared against the list in the target location. Because the exact file name with the exact file size is found, it remains unchanged (it already exists in both sides).

5. The first file's name goes into 'checked list'.

6. The second file named "Glick, J. (2007). AI in the News (column) Vol 28. No 2.pdf" will be logged in the 'checking list' and compare against target's list.

7. A file with a same or similar name does not exist in the target location, so it will be copied and its name will be inserted into the 'checked list'. The updated lists would look like Table 3:

Table 3. Example 2

No.	Source	Size	Target	Size
1.	"Glick, J. (2007). AI in the News (column) Vol 28. No 1.pdf"	200kb	"Glick, J. (2007). AI in the News (column) Vol 28. No 1.pdf"	200kb
2.	"Glick, J. (2007). AI in the News (column) Vol 28. No 2.pdf"	140kb	"Glick, J. (2007). AI in the News (column) Vol 28. No 3.pdf"	500kb
3.	"Glick, J. (2007). AI in the News (column) Vol 28. No 4.pdf"	500kb	"Glick, J. (2007). AI in the News (column) Vol 28. No 2.pdf"	140kb

8. The third file named "Glick, J. (2007). AI in the News (column) Vol 28. No 4.pdf" is checked against the target list. To check similar file names, the number of characters is counted: 54.

9. The list at the target location will be checked. The model will find one file named "Glick, J. (2007). AI in the News (column) Vol 28. No 3.pdf", which has 54 characters as well, and they have the same size. Even though both files have 54 characters, one character is different in the names.

10. The similarity level will be calculated: 53 letters are similar out of 54 letters; the names are 98% similar. The name of the file is logged but nothing is copied or deleted yet.

11. All files from the source are compared against the target replicas.

12. Referring to the log list, there are two file names with 54 characters and having 53 same characters.

13. The two files are 98% similar. Since the similarity level is more than 90% (the user may change this indicator to lower or higher) the user is asked to confirm if the files are the same.

a. If they are the same, he will be asked to choose one of the names as the file name. Then, the file names for both files will be updated according to the user's preference.

b. If the user confirms they are not the same, a copy from "Glick, J. (2007). AI in the News (column) Vol 28. No 4.pdf" will be made into the target location and a copy from "Glick, J. (2007). AI in the News (column) Vol 28. No 3" will be copied into the source location.

Considering if the user selects option one, the list would be updated as Table 4:

Table 4. Example 3

No.	Source	Size	Target	Size
1.	"Glick, J. (2007). AI in the News (column) Vol 28. No 1.pdf"	200kb	"Glick, J. (2007). AI in the News (column) Vol 28. No 1.pdf"	200kb
2.	"Glick, J. (2007). AI in the News (column) Vol 28. No 2.pdf"	140kb	**"Glick, J. (2007). AI in the News (column) Vol 28. No 3.pdf" renamed to:** "Glick, J. (2007). AI in the News (column) Vol 28. No 4.pdf"	**500kb**
3.	"Glick, J. (2007). AI in the News (column) Vol 28. No 4.pdf"	500kb	"Glick, J. (2007). AI in the News (column) Vol 28. No 2.pdf"	140kb

If the user selects the second option, the list would be updated as Table 5:

Table 5. Example 4

No.	Source	Size	Target	Size
1.	"Glick, J. (2007). AI in the News (column) Vol 28. No 1.pdf"	200kb	"Glick, J. (2007). AI in the News (column) Vol 28. No 1.pdf"	200kb
2.	"Glick, J. (2007). AI in the News (column) Vol 28. No 2.pdf"	140kb	"Glick, J. (2007). AI in the News (column) Vol 28. No 3.pdf"	500kb
3.	"Glick, J. (2007). AI in the News (column) Vol 28. No 4.pdf"	500kb	"Glick, J. (2007). AI in the News (column) Vol 28. No 2.pdf"	140kb
	"Glick, J. (2007). AI in the News (column) Vol 28. No 3.pdf"	**500kb**	**"Glick, J. (2007). AI in the News (column) Vol 28. No 4.pdf"**	**500kb**

The process will be repeated once again, but in the second round the target location and the source location will be swapped. The model does not check for the modifications on the file's content and it is only suited for the research case study (large-sized archive of articles). This is to ensure that the process is completed as fast as possible. Also sorting the files based on the file size was not considered since it is highly possible to find different files of similar size. Comparing the files, which have similar file size but are totally different from each other in terms of the name, makes the progress longer and slower. The comparison will be based on file name and the item of file size will be considered as a supplementary option for further investigation about the file only if it is necessary.

V. Conclusions

In this research, we proposed a systematic model that synchronizes large sized backups of scanned research articles in a time and cost efficient manner. One of the notable weaknesses of the reviewed literature is the considerable duration to synchronize large-sized backups in order to mirror the files' names and numbers. Moreover there are very limited applications which work across different OS, in case the backups are retained in multiple computers with different OS. The similarities of the file's name can be detected and confirmed in order to prevent file duplication problems. The proposed model can be written in any programming language to work across different OS (or even performed 'manually'). However this model is unable to check for the file modifications since the respective function may jeopardize the speed of process. The needs to check for modified files have to be analyzed based on the type of backups and user's preference; otherwise, the efficiency of the common synchronizing process will be affected by unnecessary functions.

Future research may focus on a model which observes more of the user's requirements and preferences for the specific types of backup and synchronizing methods.

References

[1] S. Balasubramaniam, and B. C. Pierce, "What is a file synchronizer?", In Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '98), Full version available as Indiana University CSCI technical report #507, April 1998.

[2] B. C. Pierce, and J. Vouillon, "What's in Unison? A Formal Specification and Reference Implementation of a File Synchronizer", Technical Report MS-CIS-03-36, Dept. of Computer and Information Science, University of Pennsylvania, 2004.

[3] B. C. Pierce, "Unison: A file synchronizer and its specification", Invited talk at Theoretical Aspects of Computer Software (TACS), Sendai, Japan, 2001.

[4] S. Basu, "File synchronization with Unison", 2012. Available from: http://www.sbasu.com/blogs/2012/09/03/file-synchronization-with-unison/.

[5] C. Hare, "Keeping Data in Sync::rsync", 2005. Available from: http://anselmo.homeunix.net/SysAdmin-Journal/html/v13/i06/a3.htm.

[6] J. Z. Brockmeier, "Get to Know rsync", 2010. Available from http://www.linux.com/learn/tutorials/271175-get-to-know-rsync.

[7] B. Martin, "Bidirectional Filesystem Syncing – DirSync Pro vs. Unison", 2003. (Reviews). Available in http://archive09.linux.com/feature/154149.

[8] L. Avgeriou, "LuckBackup", 2012. Available in http://luckybackup.sourceforge.net/.

[9] ESYS. "Cloud Backup: Advantages and Disadvantages Compared with Traditional Methods", Technology, Research & Innovation, 2012. Available from: http://www.esys.co.uk/.

[10] S. Rhea, C. Wells, P. Eaton, D. Geels, B. Zhao, H. Weatherspoon, and J. Kubiatowicz, "Maintenance-Free Global Data Storage", IEEE Internet Computing, Vol 5, No 5, pp 40–49, 2001.

[11] A. Tridgell, "Efficient Algorithms for Sorting and Synchronization", Ph.D dissertation from The Australian National University,1999.