COSC480 Project
Interim Report

Calum O'Hare
Supervisor David Eyers

1 Project goal

The aim of this project is to develop a file synchronisation tool. Similar to Dropbox (and others) its main function should be to keep data synchronised between multiple devices. What makes it different however is it should:

- Be decentralised. It will not necessarily need to be run in "the cloud", there should be no centralised server, just many cooperating client nodes.
- Allow file synchronisation between multiple clients—not just point-to-point between two clients. Clients may be running different operating systems. Clients may run on different networks, with different costs of access (including being disconnected from the Internet at times).
- Allow for fine-grained user control for the majority of the program's functions, *e.g.*, how often, and what, to replicate within different sets of files. 'What' could be filename, filetype, filesize *etc.*
- Show statistics about which files are being replicated, efficiency, cost (bandwidth, disk space). These statistics could also possible lead to a heuristic for when to sync a given file.

2 Background

There are already many services out there that synchronize your files. Dropbox, Google Drive, Microsoft SkyDrive, Apple iCloud which all offer cloud

based solutions for automatically synchronizing your files. The problems with these services is privacy and availability. Storing your data with a third party gives them access to your documents. If you are a commercial venture with sensitive information this might be concerning. You also can not guarantee that you will always be able to access your data, if the company who owns your data goes bankrupt or decides to shutdown their service you could lose all of your data with little or no warning.

The are other possible approaches to replicating files across multiple computers. For example you could use version control systems like git, subversion, mercurial, cvs. The potential problem here is that they are centralised, they rely on a central server should that server fail the replication will break. The same can be said for cloud based solutions, although these may be distributed on the back end a DDOS attack can still cut off or slow down the replication. Using version control for this purpose creates a bottleneck on the central server. It also means that replication is not automatic and has to be manually invoked.

Example use case

I like to keep all of the data on my laptop backed up to an external hard drive. The data on my computer that I wish to back up falls in to three main categories: documents, music, and movies. Documents are mostly scripts and programs that I am writing for University or work projects. Documents also include reports for assessment. These documents change very frequently and are very important to me. Often these are small files (but not always). My music collection changes relatively infrequently, files are around ≈5MB and I like to have a relatively current backup of this collection. My movie collection contains fairly large files but I don't need it to be backed up very often as it doesn't change very much and I don't care if I loose a couple of DVDs. Files that I work on at University would be very useful to have on my laptop at home. Files I work on at work mostly stay at work but occasionally I might want to bring something home to work on. The other device I always have with me and may be on one of any given (Wi-Fi or 3G) network at a certain time is my smartphone. I would like to have photos taken on this backed up to either (or both) my laptop and external hard drive.

Some of the files I move around are of sensitive or personal nature and I would prefer not to store them with a third party vendor. I also have different synchronisation requirements for different types of data. An effective file

synchronisation tool would be of great use to me personally.

3 Achievements to date

3.1 Virtual Machines, Node networks

For testing my program I needed to have a network of computers which can be linked together in different arrangements easily. I decided to use virtual machines for this job since it means I do not need to have set amount of physical machines and that I can manipulate the links between them easily and create new machines very easily.

I have used Oracle's VirtualBox for this job. I chose VirtualBox because of its easy to use command line interface. I have several scripts which call the vbmoxmange command to set up the internal network connections between machines and then start up the machine itself. This makes switching between network configurations very easy as I can just run a different script depending on what network topology I would like to test.

I have decided to use some basic topolgies to test my program to start with as shown below.

– Insert network graphs –

3.2 Python

I have chosen to use python to implement my program. Python appealed to me because it supports many different platforms (Windows, Linux, OS X). This is useful because it means I will (hopefully) encounter fewer compatibility problems when running my program across different operating systems in the future.

3.3 User control

One of the main goals of my project is to allow the user to have a large amount of control over how the programs behaves. I currently have the program reading from a configuration files which allows the user to specify which directories they want to watch and where those directories should be synchronised to.

I chose to use directories as my granularity for replication as opposed to files because keeping track of a big list of files may become unwieldy and because I replicate directories recursively I can replicate large amounts of data without a cluttered configuration file.

3.4 Inotify

3.5 Point-to-Point synchronisation

After some preliminary analysis of the available file synchronisation tools I have found a tool called Unison to be a promising starting base for this project. Unison is an open source file synchronisation tool, it supports efficient (*i.e.,* it attempts to only send changes between file versions) file synchronisation between two directories (including sub folders) between two machines (or the same machine).

I decided to run some tests on using this program and the network I had set up to determine whether this would make a good base for my program or not.

I looked at three methods of files synchronisation across different networks. Naive copying, using rsync, an application for efficiently copying files in one direction by looking at the differences in the files, and unison described above.

– Insert graphs and diagrams here –

As you can see rsync and unison performed significantly better than the naive copy method (as I expected). After the initial file transfer subsequent edits to the file meant much less data had to be transmitted over the network which meant the node graph became up to date much more quickly.

4 Future work

4.1 Full node graph replication

4.2 Mobile nodes

4.3 More user control

4.4 Feedback

5 Objectives

Primary

- Preliminary analysis
  - Research into which tools would be best to use to build the system. What is the closest thing out there to what I'm trying to build and how does it work *etc.*
- Program that keeps two directories in sync on the computer(s)
  - First simple step in building the program it just needs to replicate files between two directories.
- Sync multiple directories across multiple nodes
  - The program should be able to sync files between many nodes of a graph where the nodes are devices with files and the edges of the graph are network links that may or may not be up at any given time. The program may need to work out an efficient strategy for propagating changes throughout the entire graph.
- Implement fine-grained control (settings)
  - Provide a way for the user to control what is being replicated and when. Also give them useful information on these choice such as the cost of doing the replication.
- Implement statistics about replication

- Create a user front end that shows interesting statistics about what replication is happening and when. In a clear and easy to understand way.

6 Future Work