A fault tolerant, peer-to-peer replication network

Radu Potop

Otto Iovanici

Table of contents

Introduction

A peer-to-peer, commonly abbreviated to P2P, is any distributed network architecture composed of participants that make a portion of their resources (such as processing power, disk storage or network bandwidth) directly available to other network participants, without the need for central coordination instances (such as servers or stable hosts)[1]. Peers are both suppliers and consumers of resources, in contrast to the traditional client-server model where only servers supply, and clients consume.

Peer-to-peer systems often implement an Application Layer overlay network on top of the native or physical network topology. Such overlays are used for indexing and peer discovery. Content is typically exchanged directly over the underlying Internet Protocol (IP) network. Anonymous peer-to-peer systems are an exception, and implement extra routing layers to obscure the identity of the source or destination of queries.

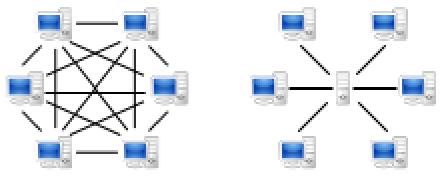

Fig. 1

P2P networks are typically used for connecting nodes via largely ad hoc connections. Sharing content files containing audio, video, data or anything in digital format is very common, and real time data, such as telephony traffic, is also passed using P2P technology[2].

Evolution

The system undergone three major steps of development.

Centralized node for synchronizing

It started similar to a centralized network where all computers were connected to one central node. Any peers had to authenticate and synchronize with the master. It was a static implementation where the central computer was holding the IP list of every peer as well as the content that had to be synchronized.

[1] Rüdiger Schollmeier, A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications, Proceedings of the First International Conference on Peer-to-Peer Computing, IEEE (2002)

[2] http://en.wikipedia.org/wiki/Peer-to-peer

There were two major downsides:

a) One was that every computer would have synchronize with the master, and this depended on the bandwidth in case of larger files;

b) Second there had to be a central computer up and running in order to benefit from the service provided by this implementation.

Decentralized synchronizing network

This was a similar approach with the difference that the master computer was holding only the IP list of the computers that could connect for synchronizing. One could not synchronize the content of two computers if any of those was missing from the list. As in the previous situation both computers had to access the master for identification before they could exchange content.

But what about a network that has no internet connection?

Current architecture

This problem was resolved considering that one user doesn't necessarily need internet connection wherever he is. He may want to synchronize content from his notebook with another computer that has no internet access at the moment. The authentication cannot be done in this case.

We choose to authenticate between the computers that need synchronization. Initially each computer has to be configured manually with authorization keys and IP addresses. There is a Bash script that runs periodically through Linux cron in order to prevent large amount of data being synchronized at once.

Architecture and applications

The applications and components used to develop the system are: SSH, Unison, Linux cron daemon and a Bash console script. We are going to discuss each of them in detail as following.

Secure Shell (SSH)

SSH is a network protocol that allows data to be exchanged using a secure channel between two networked devices.[3] Used primarily on Linux and Unix based systems to access shell accounts, SSH was designed as a replacement for Telnet and other insecure remote

[3] RFC 4252

shells, which send information, notably passwords, in plaintext, leaving them open for interception.[4] The encryption used by SSH provides confidentiality and integrity of data over an insecure network, such as the Internet.

SSH uses public-key cryptography to authenticate the remote computer and allow the remote computer to authenticate the user, if necessary.

SSH is typically used to log into a remote machine and execute commands, but it also supports tunneling, forwarding TCP ports and X11 connections; it can transfer files using the associated SFTP or SCP protocols. SSH uses the client-server model.

The standard TCP port 22 has been assigned for contacting SSH servers.

An SSH client program is typically used for establishing connections to an SSH daemon accepting remote connections. Both are commonly present on most modern operating systems, including Mac OS X, Linux, FreeBSD, Solaris and OpenVMS. Proprietary, freeware and open source versions of various levels of complexity and completeness exist.[5]

Unison (file synchronizer)

Unison is a file synchronization program. It is used for synchronizing files between two directories, either on one computer, or between a computer and another storage device (e.g. another computer, or a removable disc). It runs on Unix-like operating systems (including Linux, Mac OS X, and Solaris), as well as on Windows.

- It runs on many operating systems, and can synchronize files across platforms, so that for instance a Windows laptop may be synchronized with a Unix server.
- It detects 'conflicts' where a file has been modified on both sources, and displays these to the user
- It communicates over the TCP/IP protocol so that any two machines with an internet connection can be synchronized. This also means that the data transferred can be secured by tunneling over an encrypted SSH connection.
- It uses the rsync algorithm developed by Andrew Tridgell. This algorithm transfers only the parts of a file that have changed, and so is faster than copying the whole file.
- It is designed to be robust in the event of a program or system crash or a communication failure.
- It is open-source.

[4] http://www.serverwatch.com/news/print.php/3551081

[5] http://en.wikipedia.org/wiki/Secure_Shell

- It is written in the Objective Caml language.

File synchronization tools such as Unison are similar to version control tools (CVS, Subversion, etc.), distributed filesystems (Coda, etc.), and mirroring utilities (rsync, etc.), in that all these attempt to keep sets of files synchronized. However file synchronization tools can deal with modifications to both versions of the directory structure, without the overhead of version control.

Cron

Cron is a time-based job scheduler in Unix-like computer operating systems. The name cron comes from the word chronograph (a time-piece). Cron enables users to schedule jobs (commands or shell scripts) to run automatically at a certain time or date. It is commonly used to automate system maintenance or administration, though its general purpose nature means that it can be used for other purposes, such as connecting to the Internet and downloading email.[6]

Cron is driven by a crontab, a configuration file that specifies shell commands to run periodically on a given schedule.

Bash

Bash is a free software Unix shell written for the GNU Project. Its name is an acronym which stands for Bourne-again shell.[7] The name is a pun on the name of the Bourne shell (sh), an early and important Unix shell written by Stephen Bourne and distributed with Version 7 Unix circa 1978,[8] and the phrase "born again." Bash was created in 1987 by Brian Fox. In 1990 Chet Ramey became the primary maintainer.[9]

Bash is the shell for the GNU operating system from the GNU Project. It can be run on most Unix-like operating systems. It is the default shell on most systems built on top of the Linux kernel as well as on Mac OS X and Darwin. It has also been ported to Microsoft Windows using Subsystem for UNIX-based Applications (SUA), or POSIX emulation provided by Cygwin and MSYS. It has been ported to MS-DOS by the DJGPP project and to Novell NetWare.

[6] http://en.wikipedia.org/wiki/Cron

[7] C Programming by Al Stevens, Dr. Dobb's Journal, July 1, 2001

[8] Fresh Faces by Rosalyn Lum, Dr. Dobb's Journal, June 1, 2005

[9] Ramey, Chet (1994-08-01). "Bash - the GNU shell (Reflections and Lessons Learned)". Linux Journal. http://www.linuxjournal.com/article/2800#N0xa50890.0xb46380. Retrieved 2008-11-13.

The Bash command syntax is a superset of the Bourne shell command syntax. The vast majority of Bourne shell scripts can be executed by Bash without modification, with the exception of Bourne shell scripts stumbling into fringe syntax behavior interpreted differently in Bash (nested parentheses broke under Bash, for example, in the Mozilla startup script some years back, or attempting to run a system command matching a newer bash builtin, etc. Bash command syntax includes ideas drawn from the Korn shell (ksh) and the C shell (csh) such as command line editing, command history, the directory stack, the $RANDOM and $PPID variables, and POSIX command substitution syntax $(…). When used as an interactive command shell and pressing the tab key, Bash automatically uses command line completion to match partly typed program names, filenames and variable names.

Bash supports here documents just as the Bourne shell always has. However, since version 2.05b Bash can redirect standard input (stdin) from a "here string" using the <<< operator.

Bash 3.0 supports in-process regular expression matching using a syntax reminiscent of Perl.[10]

Components roles and features

SSH

SSH is a protocol that can be used for many applications across many platforms including UNIX, Microsoft Windows, Apple Mac and Linux. Some of the applications below may require features that are only available or compatible with specific SSH clients or servers. For example, using the SSH protocol to implement a VPN is possible, but presently only with the OpenSSH server and client implementation.

- For login to a shell on a remote host (replacing Telnet and rlogin)
- For executing a single command on a remote host (replacing rsh)
- For copying files from a local server to a remote host. See SCP, as an alternative for rcp
- In combination with SFTP, as a secure alternative to FTP file transfer
- In combination with rsync to backup, copy and mirror files efficiently and securely
- For port forwarding or tunneling a port (not to be confused with a VPN which routes packets between different networks or bridges two broadcast domains into one.).

[10] http://en.wikipedia.org/wiki/Bash

- For using as a full-fledged encrypted VPN. Note that only OpenSSH server and client supports this feature.
- For forwarding X11 through multiple hosts
- For browsing the web through an encrypted proxy connection with SSH clients that support the SOCKS protocol.
- For securely mounting a directory on a remote server as a filesystem on a local computer using SSHFS.
- For automated remote monitoring and management of servers through one or more of the mechanisms as discussed above.
- For secure collaboration of multiple SSH shell channel users where session transfer, swap, sharing, and recovery of disconnected sessions is possible.[11]

Unison

Unison allows the same version of files to be maintained on multiple computing devices. In other words, when two devices are synchronized, the user can be sure that the most current version of a file is available on both devices, regardless of where it was last modified.

Unison shares a number of features with tools such as configuration management packages (CVS, PRCS, Subversion, BitKeeper, etc.), distributed filesystems (Coda, etc.), uni-directional mirroring utilities (rsync, etc.), and other synchronizers (Intellisync, Reconcile, etc). However, there are several points where it differs:

- Unison runs on both Windows and many flavors of Unix (Solaris, Linux, OS X, etc.) systems. Moreover, Unison works across platforms, allowing you to synchronize a Windows laptop with a Unix server, for example.
- Unlike simple mirroring or backup utilities, Unison can deal with updates to both replicas of a distributed directory structure. Updates that do not conflict are propagated automatically. Conflicting updates are detected and displayed.
- Unlike a distributed filesystem, Unison is a user-level program: there is no need to modify the kernel or to have superuser privileges on either host.
- Unison works between any pair of machines connected to the internet, communicating over either a direct socket link or tunneling over an encrypted ssh connection. It is careful with network bandwidth, and runs well over slow links such as PPP connections. Transfers of small updates to large files are optimized using a compression protocol similar to rsync.

[11] Article: GSW UTS Team Services Retrieved 2009-12-15

- Unison is resilient to failure. It is careful to leave the replicas and its own private structures in a sensible state at all times, even in case of abnormal termination or communication failures.
- Unison has a clear and precise specification.
- Unison is free; full source code is available under the GNU Public License.

Cron

Cron is used for automating tasks in the Linux/Unix Environment like:

- Using crontab to execute a shell script periodically.
- Automating network access using ssh and keys.
- Backing up a mysql database periodically.
- Doing a remote action based on a real time event.

Bash

Bash has several features that can be used:

- Command Line Options: Command line options that you can give to Bash.
- Bash Startup Files: When and how Bash executes scripts.
- Is This Shell Interactive? : Determining the state of a running Bash.
- Bash Built-ins: Table of built-ins specific to Bash.
- The Set Built-in: This built-in is so overloaded it deserves its own section.
- Bash Variables: List of variables that exist in Bash.
- Shell Arithmetic: Arithmetic on shell variables.
- Printing a Prompt: Controlling the PS1 string.

Use cases and scenario

The system presented here comes in handy for those computer users who work on multiple files located on different stations and in different locations. The system is designed to synchronize a folder on multiple computers belonging to the same user but not all at once. The files will be redundant, meaning that they will be present on every computer that the person uses.

The first example

In the following we will present an example considering:

a) We have a computer user with three stations:

  - Laptop computer (C1)
  - Home desktop computer (C2)
  - Computer at the work place (C3)

b) The goal of the project is for the person to have in any location the files(F1, F2, F3…) he is working on (documents, music files, pictures,…);

c) For the system to work we need at least 2 computers to be turned on at the same time;

d) We consider that the person will work on the files from one computer at a time;

e) All computers have network access and capabilities;

We start with:

- C1 has F1, F2, F3
- C2 has F4
- C3 has no user files

We end with:

- C1, C2, C3 will have all and the same files

Step 1

The computer user is at home (C1, C2) where he's using F4 on his desktop computer. The laptop computer C1 is powered on when the Cron job will execute the Bash script and the folder synchronization will start. When the job is finished C1 and C2 will contain the same F1, F2, F3, F4 files. He powers down the C2 and goes to work.

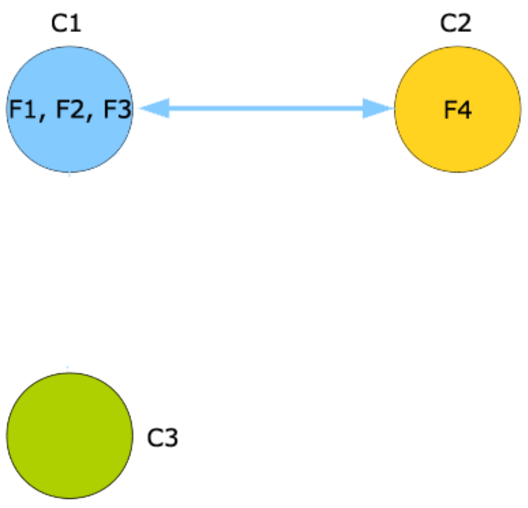

Fig. 2

Step 2

User arrives at the work place where the synchronization between C1 and C3 will result in C3 having all 4 files (F1 – F4).

He creates a presentation (F5) on C3. Through the synchronization F5 will be transferred on laptop computer C1, that the user is going to use at a client meeting.

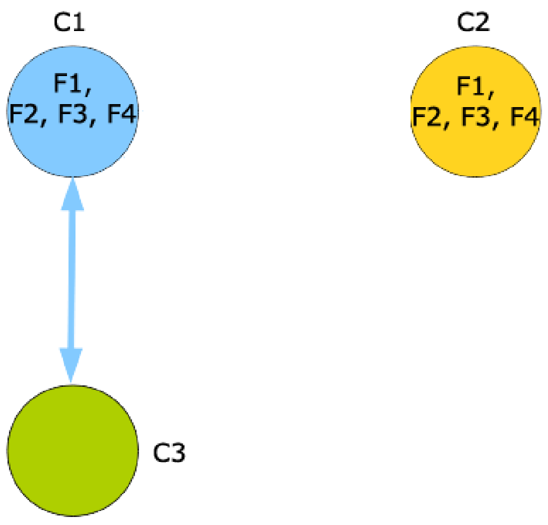

Fig. 3

Step 3

Once he arrives home, after C1 and C2 synchronize he can edit file F5 in the comfort provided by the Desktop computer C2.

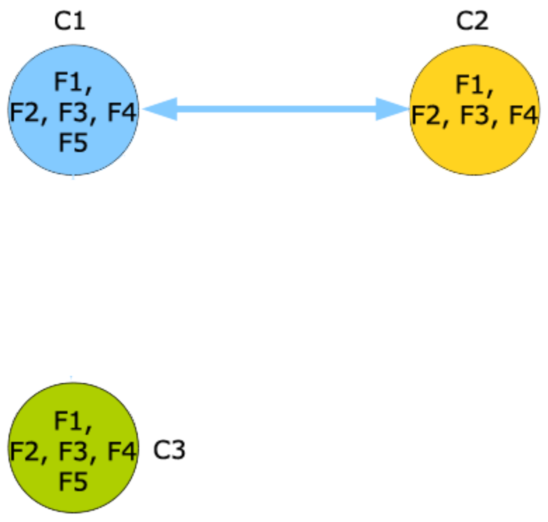

Fig. 4

Second example

In this example we have a user with 4 computer stations as follows:

a) Computers have been assigned letters: A, B, C and D;

b) Files have numbers assigned: 1, 2, 3, 4;

c) All computers have network access and capabilities.

d) The goal is to synchronize files on all four machines over the network.

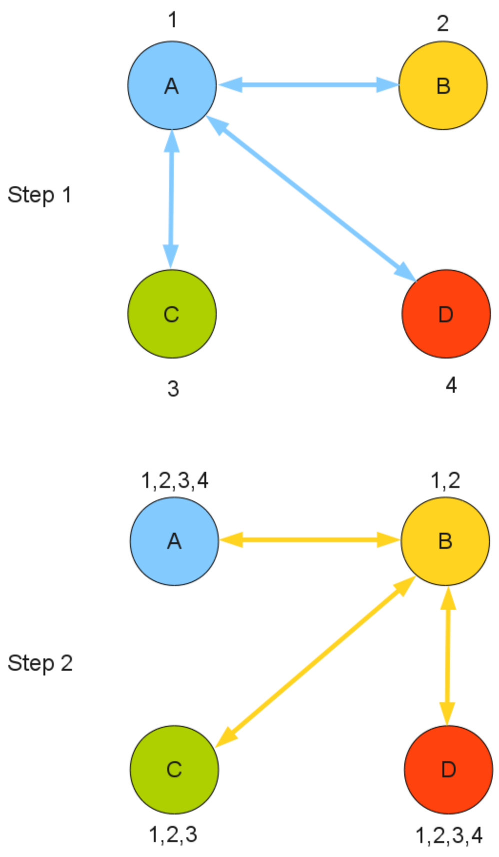

Fig. 5

Each computer has 1 different file from the others: Computer A stores file 1, computer B stores file 2, Computer C stores file 3 and so on.

At the end of the synchronization steps all four files (1, 2, 3, 4) are stored on every computer (A – D).

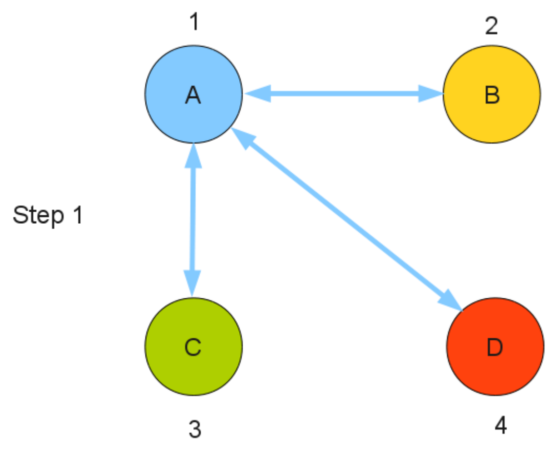

Fig. 6

Step 1

Computer A is synchronized with B, C and D in that particular order. The result is:

- A – has 1, 2, 3, 4;
- B – has 1, 2;
- C – has 1, 2, 3;
- D – has 1, 2, 3;

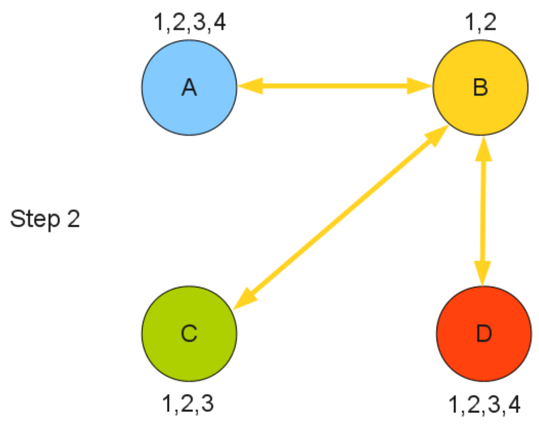

Fig. 7

Step 2

Computer B has to synchronize with computer A, C, D in this particular order for every machine (A – D) to have the same file content when finished.

Conclusions

- The system can provide a easy to use method for synchronizing a “home” directory for example.
- It is a secure way to synchronize files because of SSH support;
- It is scalable to any number of computer a user may have or want to work with;
- There is no overhead because the synchronization is made periodically due to the cron daemon;
- If one machine is compromised, it can be redrawn it’s corresponding SSH key;

Future developments

As future developments are concerned the system can provide:

- Multi user capabilities;
- File merging and conflict resolution;
- Windows support;
- A centralized way to handle keys and stations IP;
- Full User Interface;

References

http://en.wikipedia.org/

http://www.cis.upenn.edu/~bcpierce/unison/

http://www.cis.upenn.edu/~bcpierce/papers/index.shtml#File%20Synchronization

http://www.linuxjournal.com/article/7712

http://www.cs.utah.edu/dept/old/texinfo/bash/features_toc.html

http://www.openssh.com/

http://www.ssh.com/

http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

http://linux.die.net/man/5/crontab

http://www.computerhope.com/unix/ucrontab.htm

http://troy.jdmz.net/cron/

http://kevin.vanzonneveld.net/techblog/article/schedule_tasks_on_linux_using_crontab/

http://www.scrounge.org/linux/cron.html

Glossary

P2P – Peer to Peer

IP – Internet Protocol

TCP – Transmission Control Protocol

SSH – Secure Shell

X11 – X Window System

SFTP – Secure File Transfer Protocol

SCP – Secure Copy

CVS – Concurrent Versions System

PRCS – Project Revision Control System