Journal of Basic and Applied Engineering Research
Print ISSN: 2350-0077; Online ISSN: 2350-0255; Volume 1, Number 7; October, 2014 pp. 76-78

http://www.krishisanskriti.org/jbaer.html

File Sharing using Python in Peer to Peer Networks

Shantanu Khandelwal

University Institute of Engineering and Technology, Panjab University, Chandigarh, India

Abstract: **Peer-to-Peer (P2P) systems are widely used as "file-swapping" networks to support distributed content sharing. Quite a good number of P2P networks for file sharing have been developed and deployed. This paper presents content sharing in P2P networks using python SimpleHTTPServer.**

Keywords: **file sharing; multiprocessing; P2P Networks; python; SimpleHTTPServer**

1. INTRODUCTION

Peer to peer networking has emerged as a viable business model and systems architecture for Internet scale applications. It is an effective way to build applications that connect millions of users across the globe without reliance on specially deployed servers. It is an alternative to distributed computing which works on client/server model. All the nodes here are equal. In peer-to-peer (P2P) networks two or more computers are connected with each other in such a way that they can share resources without the need for a server by direct exchange between systems. The networks can be small or large ranging from a few computers connected through USB or a small network in any organization or a network on the internet in which the computers have relationships with other computers through special protocols and applications. The existing computing power and network connections are used to benefit the entire organization by sharing computer resources and services.

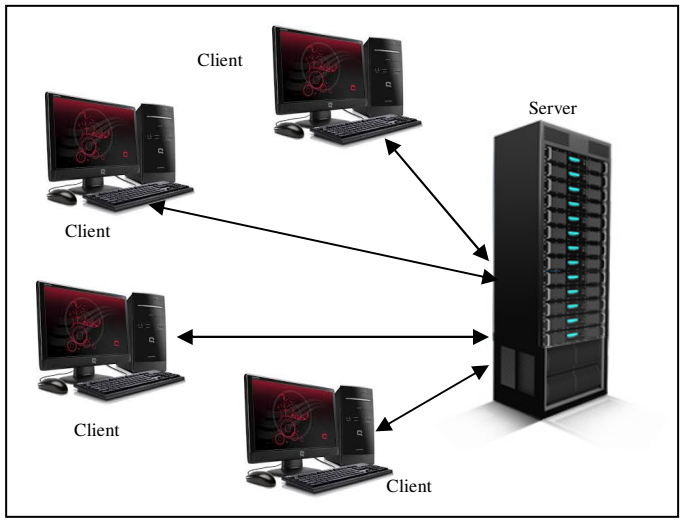

Fig. 1: The Client/Server Model

Decentralization, which is a key feature in P2P networks has several implications. There are numerous benefits P2P networks can provide such as scalability, reliability and ease of distribution. They have built-in fault tolerance, replication and load balancing capability. This capability comes from redundancy and shared responsibility. Although access control is difficult in P2P networks.

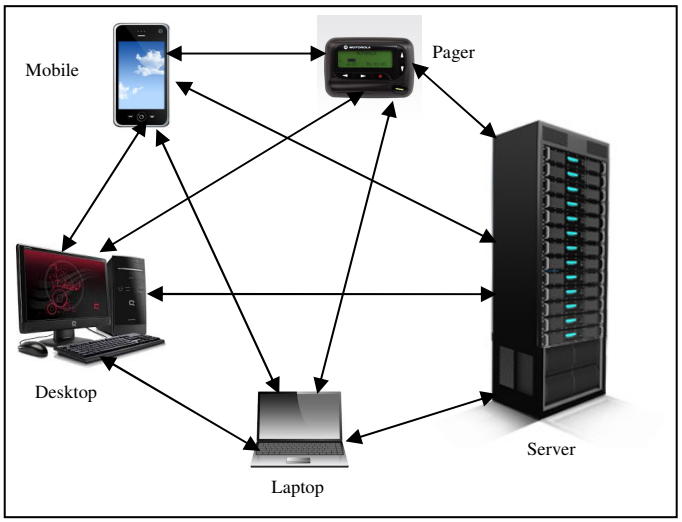

Fig. 2: The Peer-to-Peer Model

All the three valuable fundamental assets of Internet, namely information, bandwidth and computing resources are underutilized as Internet is a client /server model. It is difficult to find information and it is impossible to create index and catalog. With regards to Bandwidth, the hot links become hotter and the cold ones remain cold. As far as computing resources are concerned, the heavily loaded nodes become more overloaded and the idle nodes never become heavily loaded. Centralized systems and client-server systems operate in controlled environments but in a P2P network the users can openly share files and devices.

2. FILE SHARING

File sharing is the public or private sharing of computer data such as movies, games, books, music in a network using a specialized P2P software program. Various levels of access privileges can be given for the files shared. The same files can

be used by different users in different ways. Users may be able to read, write, copy, modify, and even print it. The advantage of file sharing in peer to peer networks is that files are not saved on any other server so they are secure. There is no requirement of Internet.

3. LITERATURE SURVEY

A large number of P2P file sharing programs are increasingly popular. Shareaza, Bit Torrent, Ares, BearShare, Kazaa, Morpheus, Limewire, eDonkey, eMute, WinMx are some of them. Each program is characterized by either large setup size, slow speed, too much CPU usage, embedded advertising, grind to hault, wasted data downloaded, incompatibility. Also, some of them run only on one or two platforms, do not allow surfing the Internet or otherwise utilize the network when files are uploaded or downloaded.

4. APPROACH

For file sharing we have used multi-threading programming and execution model which provides a useful abstraction of concurrent execution. The advantage of the multi-threaded environment is parallel and faster execution. Using multiprocessing, the application can manage multiple requests by the same user or different users and remain responsive to input. If single-threaded program is used and, if the main execution thread blocks on a long-running task, the entire application can freeze at any time. If LAN is available between sender and receiver then it uses it. But even if LAN is not available it can use appropriate hardware like WIFI adapter available in laptops and create a hotspot on the fly with WPA2-PSK encryption technique. In either case, the module generates a URL that you have to share with the receiver.

File sharing using Python SimpleHTTPServer does not require any receiver side software. Only a web browser is required at the receiver side which is readily available. The script for file sharing coded in Python 2.7 is stored at the sender side. Since Python is platform independent, this script is also platform independent. The sender only requires a fair knowledge of directory structure to send the files. The script generates a URL which depends on the local IP address of the sender. If Internet is available at the time of file sharing, URL shortening services like tinyURL can be used which generates a small and readable URL and if not then the URL generated by the script is conveyed to the receiver side by any suitable means.

A. SimpleHTTPServer

SimpleHTTPRequestHandler class which has its interface compatibility with **BaseHTTPServer.BaseHTTP Request Handler** is defined by the **SimpleHTTP Server** module. The **SimpleHTTPServer** module defines the following class:

Class

SimpleHTTPServer.**SimpleHTTPRequestHandler**(*request*, *client_address*, *server*)

This class maps the directory structure to HTTP requests and files are served from the current directory and below. Parsing the request, is done by the base class **BaseHTTPServer.BaseHTTPRequestHandler**. **do_GET()** and **do_HEAD() f**unctions are implemented by this class

B. SimpleHTTPRequestHandler

Following methods are defined by The SimpleHTTPRequestHandler class :

- do_HEAD() To serve the 'HEAD' request type this method is called. This would send the headers for the requests which are equivalent to GET request.
- do_GET() This method is used to handle GET request. This gives relative path to the current working directory which is then mapped to a local file. It is also used to place all the parameters on the URL.

After the request is mapped to a directory, a file named index.html or index.htm (in that order) is searched in the current working directory. If the file index.html is found, its contents are returned and if not then a directory listing is displayed by calling the list_directory() method. The list_directory() method uses os.listdir() so as to scan the directory, and a 404 error response is returned if the listdir() fails. If the request was mapped to a file, the requested fileis opened and it's contents are sent. If there is any error then an IOError exception in opening the requested file is mapped to a 404, 'File not found' error. The guess_type() method is then called to guess the content type, The guess_type() method in turn uses the *extensions_map* variable.

C. Multiprocessing

The Multiprocessing package uses an Application Programming interface to support the called processes. It is like a threading module. The package thus allows concurrency at both local and remote level. The Multiprocessing package thus avoids the Global Interpreter Lock because in Multiprocessing sub processes are used instead of threads. Thus full advantage of multiple processors can be achieved by the programmer by the usage of multiprocessing module on a given machine. It is platform independent and can run on both Unix and Windows.

An API similar to the threading module is used by the Multiprocessing package that supports spawning processes. Both local and remote concurrency are offered by the multiprocessing package, which effectively side-steps the Global Interpreter Lock by using sub processes instead of threads[4]. Due to this, the multiprocessing module allows the programmer to fully leverage multiple processors on a given machine.

In Multiprocessing, after creating a Process object and the its start() method is called new processes are generated. The new

processes generated follow the Application Programming interface of threading.Thread.

- start() Starts the process's activity. This method is called only one time for every process object. After this method is over the object's run() method is called in a separate process.
- join([*timeout*]) When this method is called, the thread that called it is blocked. When the join method is terminated either automatically or timeout, the calling thread is unblocked. Unlike start() method, the join() method can be called any number of times once it is started. Also joining of a process with itself is avoided as it will lead to deadlock

To implement the No-Pendrive script simpleHTTPServer of Pyhton module(package)[2], [3] has been used which is a built in web server in Python that provides standard GET and HEAD request handlers. An advantage with the built-in HTTP server is that we do not have to install and configure anything. The only thing we need is to have Python installed.

D. SimpleHTTPServer Usage

```
To only serve on localhost
import sys
import BaseHTTPServer
from SimpleHTTPServer
import SimpleHTTPRequestHandler
HandlerClass = SimpleHTTPRequestHandler
ServerClass = BaseHTTPServer.HTTPServer
Protocol  = "HTTP/1.0"
if sys.argv[1:]:
port = int(sys.argv[1])
else:
port = 8000
server_address = ('127.0.0.1', port)
HandlerClass.protocol_version = Protocol
httpd = Server-Class(server-address, Handler-Class)
sa = httpd.socket.getsockname()
print "Serving HTTP on", sa[0], "port", sa[1], "..."
httpd.serve_forever()
```

E. Multiprocessing Usage

```
from multiprocessing import Process
import os
def info(title):
print title
print 'module name:', __name__
if hasattr(os, 'getppid'): # only available on Unix
 print 'parent process:', _os.getppid()
print 'process id:', _os.getpid()
def f(name):
info('function f')
print 'hello', name
if __name__ == '__main__':
info('main line')
p = Process(target=f, args=('bob', ))
p.start()
p.join()
```

F. NoPendrive Script Usage

Assume that we would like to share the directory /home/somedir and sender IP address is 192.168.10.2
Open up a terminal and type: $ nopendrive
The module requires port number and the directory name to share as input. The module will then generate some URL such as http://192.168.10.2:8000. Convey this URL to the receiver to receive the file sent. If the client is using any download manager and becomes offline between file sharing, then he can continue from where he left off. But if the server goes offline in between, the entire file has to be resent again.

5. CONCLUSION

Python's SimpleHTTPServer is a great way to serve the contents of the current directory from the command line: An advantage with the built-in HTTP server is that we do not have to install and configure anything. A super-simple script lets us instantly share files across a local network. The application is very light weight. It size is just about 2 MB. The application is free from the discovery process in peer to peer architectures. Any unused port can be used for file sharing between the server and the client.

P2P file sharing system using python has not been plagued by several problems for users. The providers of leading P2P file applications can earn revenue from third parties by embedding spyware and malware into the applications. Users then find their computers infected with such software immediately after installing the P2P application. Secondly, a large amount of corrupted or polluted content has been published in file sharing systems and it is difficult for users to distinguish such content from the original digital content they seek[1]. Also, the leading P2P file sharing systems have not adopted mechanisms to protect licensed contents or collect payments for transfers on behalf of copyright owners. Several ventures seek to legitimate P2P file sharing for licensed contents by incorporating techniques for Digital Rights Management (DRM) and super distribution in P2P distribution architectures. When such techniques are used, content is required to be encrypted. So even if it can be freely distributed purchasing an encrypted license file to render the media is required by the user.

REFERENCES

[1] John F. Buford, Heather Yu, Eng Keong Lua "P2P Networking and Applications"

[2] http://www.linuxjournal.com/content/tech-tip-really-simple-http-server-python

[3] http://docs.python.org/2/library/simplehttpserver.html

[4] http://docs.python.org/2/library/multiprocessing.html