Part2Part

By Mihai Manolache

8 December, 2019

1 Introduction

Part2Part is a peer-to-peer file sharing application. Peer-to-peer (P2P) computing or networking is a distributed application architecture that partitions tasks or workloads between peers. Peers are equally privileged, equipotent participants in the application. They are said to form a peer-to-peer network of nodes. Peers 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 by servers or stable hosts. Peers are both suppliers and consumers of resources, in contrast to the traditional client-server model in which the consumption and supply of resources is divided.[1]

1.1 Motivation

The traditional client-server architecture has some siginificant problems such as:
-lack of robustness: services are fragile, because, in the pure form of client-server architecture, the availability of a service is dependent on a single server running on a single device and a small number of communication links that connect that device out to the remote devices, resulting in a bottleneck;
-lack of resilience: services, once broken, cannot be recovered quickly, because they remain unavailable until the relevant device and/or network has been returned to a workable state;
lack of scalability: as resources supporting a client-server scheme are expanded, the number of users that can be supported does not grow as quickly. Hence very large schemes are very expensive;
-incapacity to service levels of demand that are very high relative to the processor and network resources. Peaks may arise on a daily, monthly or yearly cycle, as a result of external conditions such as bad weather, or as a result

of an entertainment, marketing or sporting event;
-vulnerability to attack, especially denial of service, but also masquerade and data pollution.
The P2P architecture aims to solve these problems by harnessing the 'power at the edge of the net'.[2]
Aims and benefits of P2P include: efficient use of resources, scalability, reliability, easy administration, anonymity, dynamism.[3]

2 Technologies

In order to provide a smooth experience and concurency, serve multiple requests etc. multithreading will be used. Multithreading will be achieved by using the POSIX thread libraries.The POSIX thread libraries are a standards based thread API for C/C++. It allows one to spawn a new concurrent process flow. It is most effective on multi-processor or multi-core systems where the process flow can be scheduled to run on another processor thus gaining speed through parallel or distributed processing. Threads require less overhead than "forking" or spawning a new process because the system does not initialize a new system virtual memory space and environment for the process.[4]
Comunication between the nodes will be done by using the Transmission Control Protocol(TCP). TCP provides reliable, ordered, and error-checked delivery of a stream of octets (bytes) between applications running on hosts communicating via an IP network.[5]

3 Use cases

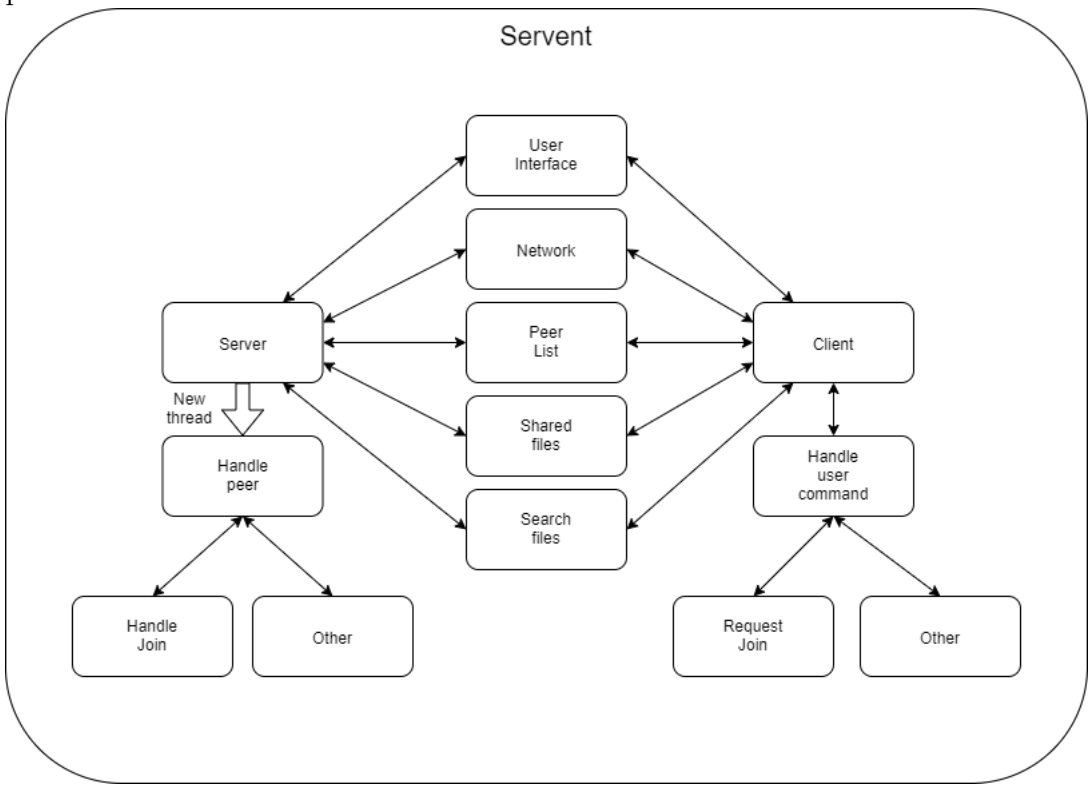
The user is anybody who has a machine running linux and wants to either get a file or share a file. Interacting with the application will be done through the linux command-line by inputing a command and receiving some text confirmation.

- USE CASE 1. The user starts the application and requests to join the network by connecting to a known node already in the network.
- USE CASE 2. The user exits the network either willingly or because a failure has occured. The network will remove the user from the list of peers.

- USE CASE 3. The user searches the network to see if a file exists. This can be done by file name. The application will return a list of peers that have the desired file.
- USE CASE 4. The user publishes a file from his machine.
- USE CASE 5: The user removes a file from the list of files he provides to the network.
- USE CASE 6: The user requests to download a file from the network.
- USE CASE 7: Download of a file fails and the user is notified.

4 Architecture

All the peers/nodes in the network are servents(have both a SERVer process and a cliENT process). The structure of the network is unorganized, there are no supernodes or central nodes. Joining the network is done by connecting to a known node already in the network and receiving a list of initial peers from it.



- Client: This module will handle asking other peers for different services(join, search file, download file) as requested by the user.
- Server: This module will handle servicing other peers request's.
- User Interface: Consisting only of the command-line. Will forward messages to both the client(ex: join) and the server(ex: stop service).
- Network: Offers methods to the client and the server to comunicate with other peers over TCP.
- Search files: Searching for files is done by initiating a BFS with a time-to-live(a limited amount of hops).
- Shared files: A list of files on the user's device that are to be shared with other peers.
- Peer list: A list of the IPs and Ports of known peers in the network.

5 Conclusion

The project architecture is not very efficient but for a small scale network it should be sufficient. For larger scale networks more complex ways of managing the network and searching in it might be needed. The current architecture manages however to provide many of the benefits of peer-to-peer applications such as reliability, easy administration, anonymity and better use of resources and scalability compared to a pure server-client architecture.

References

[1] Wikipedia Commons
Definition of P2P . `https://en.wikipedia.org/wiki/Peer-to-peer/`

[2] Peer-to-Peer (P2P) - An Overview `http://www.rogerclarke.com/EC/P2POview.html#PADef`

[3] P2P Paradigm. `https://profs.info.uaic.ro/~computernetworks/files/11rc_ParadigmaP2P_En.pdf`

[4] POSIX thread (pthread) libraries `https://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html`

[5] Transmission Control Protocol `https://en.wikipedia.org/wiki/Transmission_Control_Protocol`