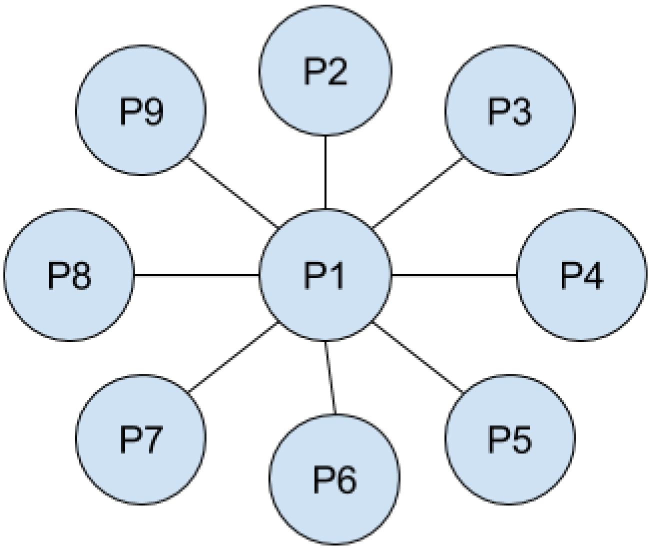
File Sharing Peer to Peer
Gnutella Style

This system aims to allow connected peer to search and downloads files. Peers are connected together as a star topology. There is no placed server in order to manage the peers activities. However, each peer works as a client and server in the same time. The peer has the ability to listen to others requests. Moreover, it has the ability to send requests.

In this system, peer one placed in the middle while the others connected statically to peer one.

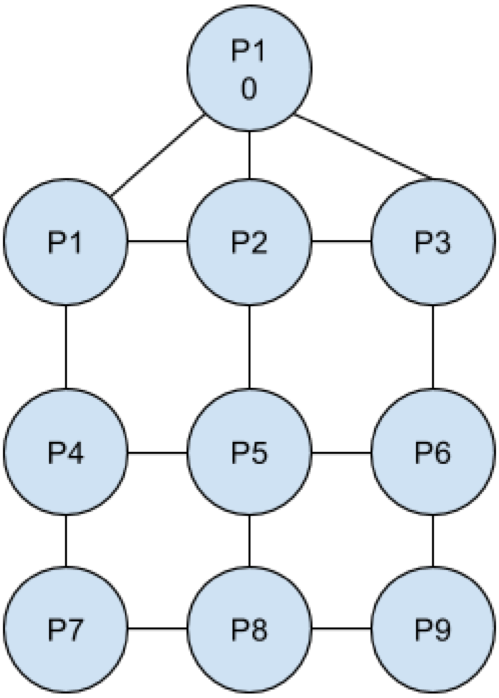


The topology structure is predefined in a config.txt file in the system as following:

```
p1 localhost 60000 p2-p3-p4-p5-p6-p7-p8-p9-p10
p2 localhost 60001 p1
p3 localhost 60002 p1
p4 localhost 60003 p1
p5 localhost 60004 p1
p6 localhost 60005 p1
p7 localhost 60006 p1
p8 localhost 60007 p1
p9 localhost 60008 p1
p10 localhost 60009 p1
```

The separate structure design give the user the needed flexibility to manipulate the peer structure as desired. Moreover, the config.txt contains the peer IP and ports number. They can be changed if needed.

Because of this kind of flexibility, the same code run the 2D-mesh topology based on the user input.



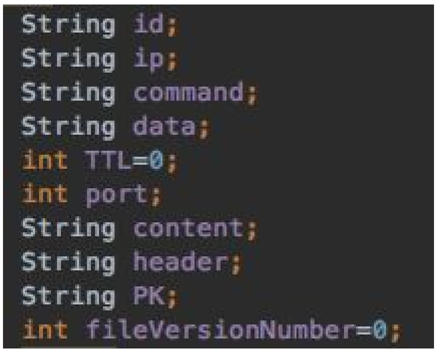
The configuration of the 2D-Mesh located at the project folder and named as ‘MeshConfig’.

```
p1 localhost 60000 p2-p4-p10
p2 localhost 60001 p1-p3-p10-p5
p3 localhost 60002 p10-p2-p6
p4 localhost 60003 p1-p5-p7
p5 localhost 60004 p2-p4-p6-p8
p6 localhost 60005 p3-p9-p5
p7 localhost 60006 p4-p8
p8 localhost 60007 p7-p5-p9
p9 localhost 60008 p6-p8
p10 localhost 60009 p1-p2-p3
```

When a user query a search request for a file, he will send a broadcast message to his neighbours who are going to do the same if they do not own the requested file.
If a file found, the file owner will send back a hit query the requester through the addresses that have been stored in the header message of all passby nodes.
If a node received the same message one more time, it will drop it immediately. This action is done by doing the following steps:

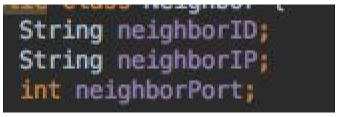
1. When a new message created, attache the PK (peer ID+time stamp) along with message, and store it in the PK field.
2. For any broadcasted message, record the PK in a list.
3. For every received message, compare the PK with the stored PKs. If found, drop the message. Else, broadcast the message.

The message structure in the system is as following:



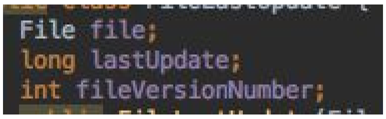
```
String id;
String ip;
String command;
String data;
int TTL=0;
int port;
String content;
String header;
String PK;
int fileVersionNumber=0;
```

The neighbor structure is as following:



```
String neighborID;
String neighborIP;
int neighborPort;
```

The file object structure is as following:



```
File file;
long lastUpdate;
int fileVersionNumber;
```

Enhancement suggestion for the system:

Currently, if a peer is out, the structure link will be broken, and some peers will be unreachable. Therefore, applying a disaster mechanism for the out-of-service peers will make the system more reliable. For example, if a peer sense the medium before sending, and then notify all if a node is unavailable, we may connect the broken link to an alternative one.