# Design Document

# 

We have used the Java Sockets to implement for the interprocess communication in our Gnutella File Sharing System. We have kept the TTL to 7 by default which will decrement when it hops from one hop to another .

Program design :

Each peer contains :

1. A client thread (peer acting as a client, ie, sending requests to other peers)

2. A server thread (peer acting as server I.e, accepting request from other peers)

3. a watcher thread (to monitor events on the peers master folder and notify neighbors when there is a change)

4. A pull thread

Each peer also maintains :

1. query messages it has seen

2. queryhit messages it has seen

3. invalidation messages it has seen

4. all files residing on the peer and their metadata

5. all its neighbors (initialized on bootup from the peer configuration file)

PUSH BASED:

1. On the peer bootup, a watcher thread is created to monitor the peer’s master folder.

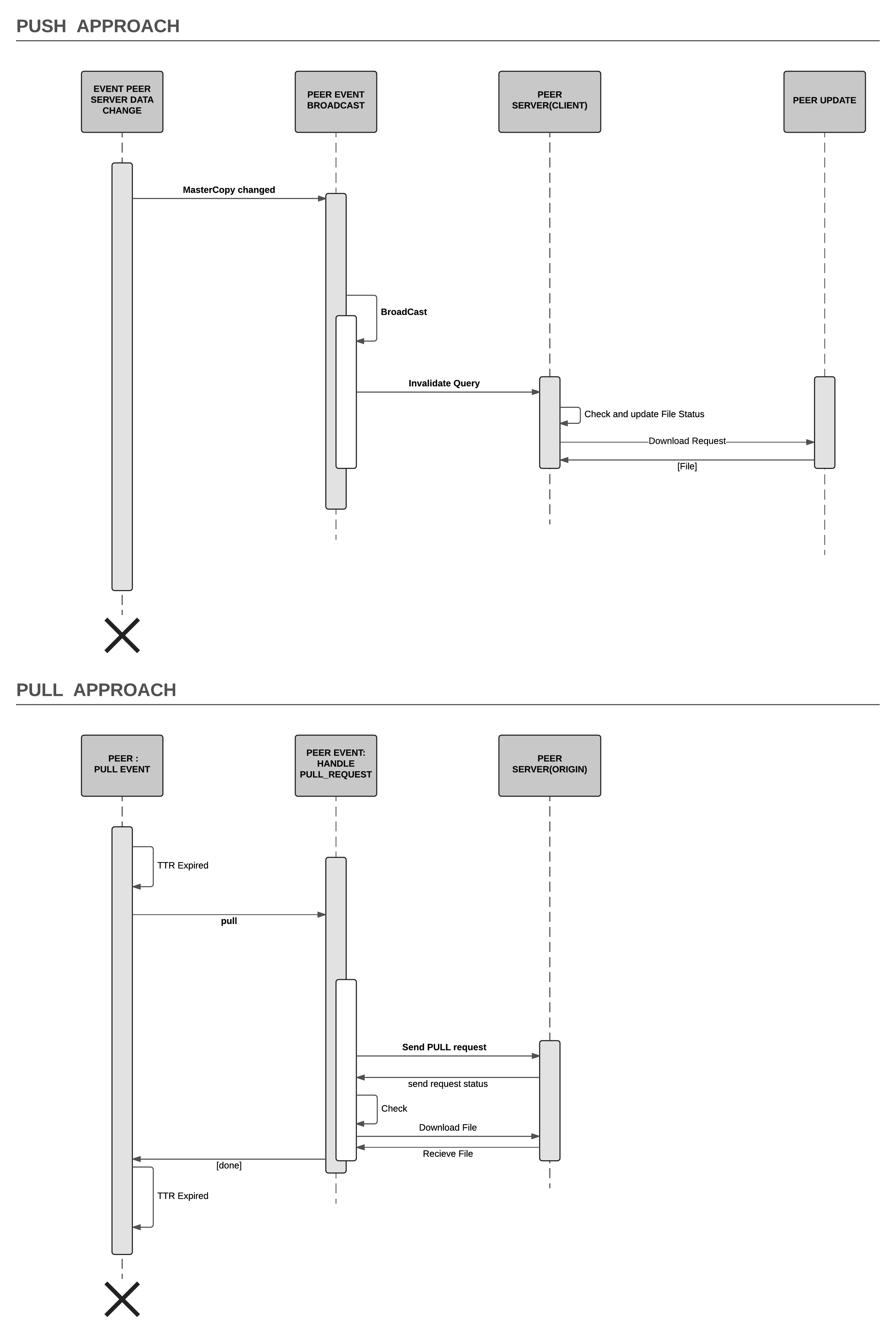
2. the watcher thread continously monitors the folder for any changes, and send out a message to the peer when any file is modified/deleted/created.

3. Peer upon receiving the notification update the metadata of the files it is holding and send out an INVALIDATION message to all its neighbor about the changes.

4. Peer upon receiving the INVALIDATION message, if the file was downloaded from the peer before, the file is marked as stale. And again the message is forwarded to all its neighbors.

PULL BASED:

Sequence Diagram :



## Use case supported:

1. lookup for a file by entering the file name.

2. download a file by entering file name, peer address and port from where to downloaded.

3. Refresh a stale file

## Configuration file-

We have kept list of neighbors in a separate like ***portNumberconfig.file***

Below configuration for linear & star topology, where ip is the ip address of neighbor.

## Linear Topology

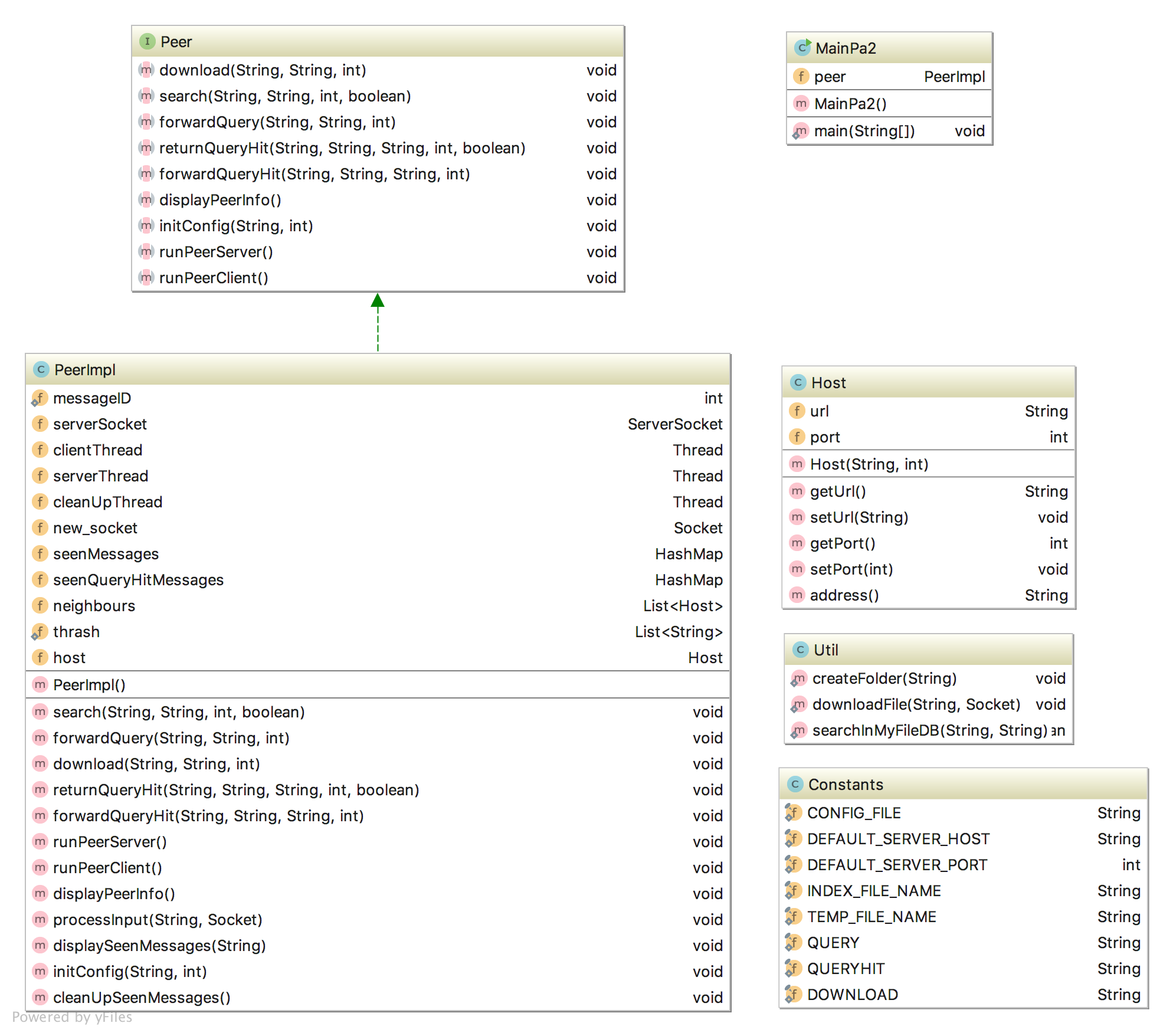
|  |  |
| --- | --- |
|  | File Contents:  **1config.file**  ip:2  **2config.file**  ip:1  ip:3  **3config.file**  ip:2  ip:4  **4config.file**  ip:3 |

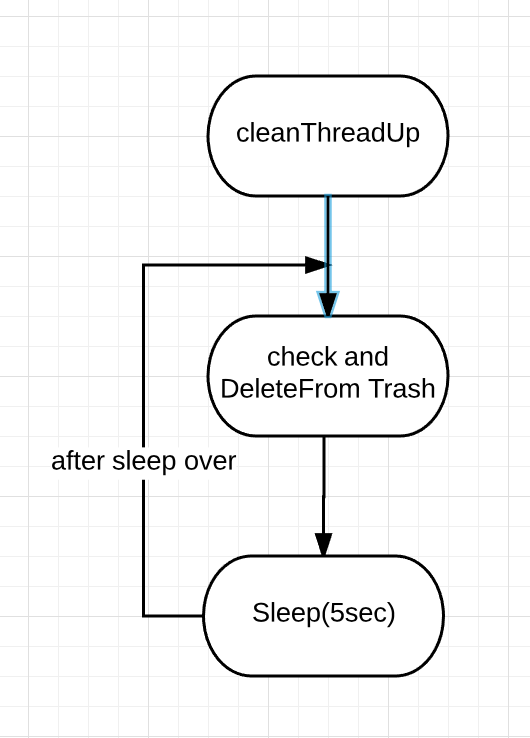
## Star Topology

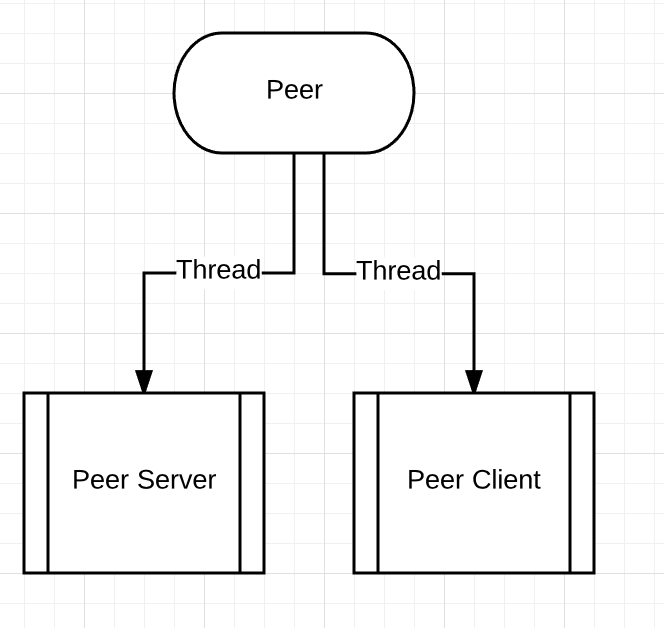
Peer Configuration File for star topology

|  |  |
| --- | --- |
|  | File Contents:  **1config.file**  ip:4  ip:3  ip:2  ip:5  **2config.file**  ip:1  **3config.file**  ip:1  **4config.file**  ip:1  **5config.file**  ip:1 |

# Class Diagram :







Manual text

Steps to run our program :  
  
1. Place the shared shell script "run.sh" , shared jar file and corresponding configuration file in a seperate folder for each peer.  
 For example :  
 if I want to create a peers running on port 52001, I will create a folders(any name) and place "run.sh" , jar file and its config file "52001config.file" in it.  
2. Run the shell script (./run.sh)  
3. Default config? Y/N, Say N  
4. Enter ipaddress of the system where peer is running (if it this system itself, enter "localhost")  
5. Enter port where this peer has to run.. (I will say 52001 if I have placed 52001config.file)  
6. later select options accordingly  
7. If you download any files it will be stored in "sharedFolder(ip:port)". in my case "sharedFolderlocalhost:52001"  
8. Any file shared by this peer should also be stored in this folder.. this folder will be created on peer's bootup

Example tree structure -

