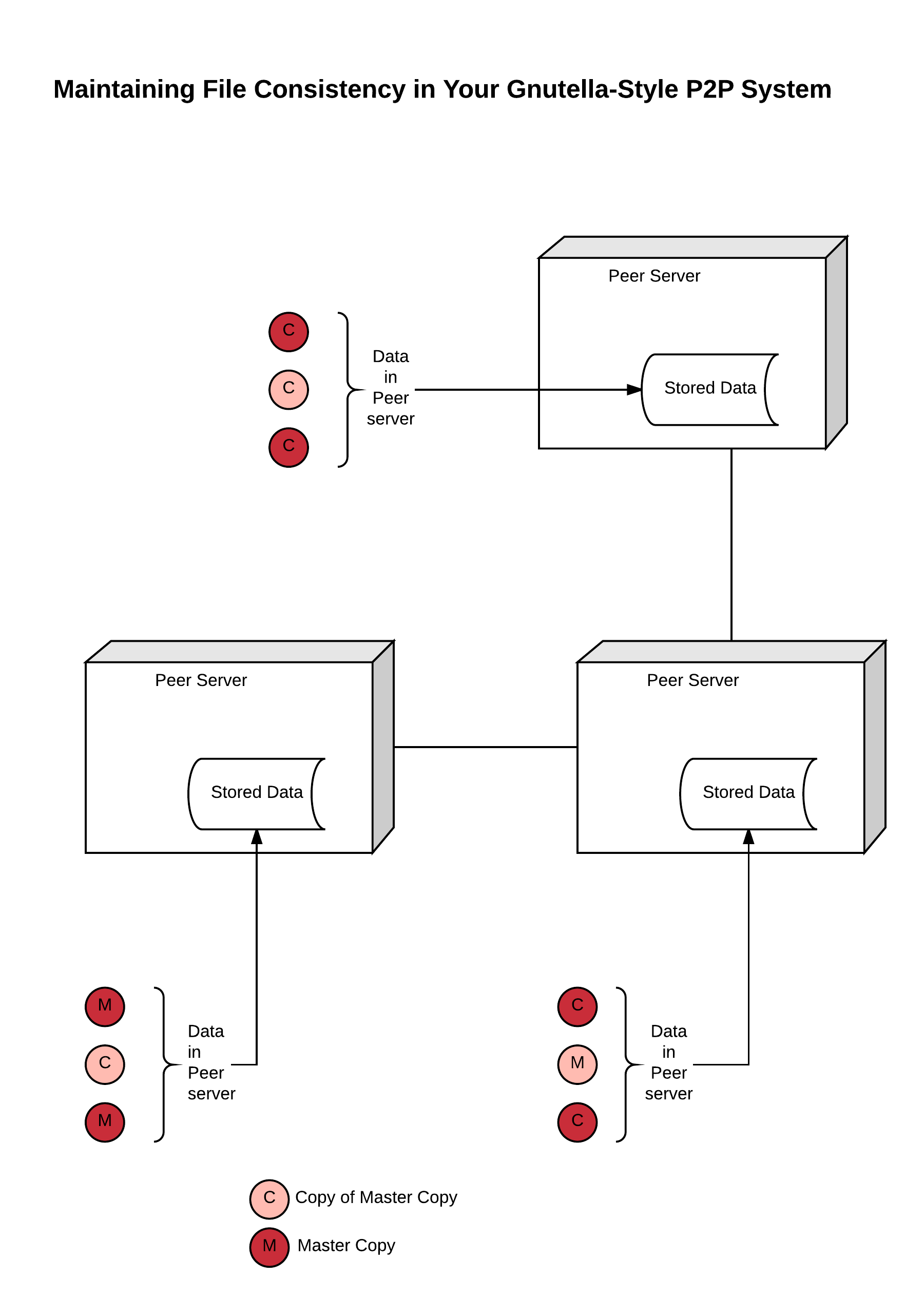
# Design Document



We have used the Java Sockets to implement for the inter process 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

## PUSH BASED:

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

2. the watcher thread continuously 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.

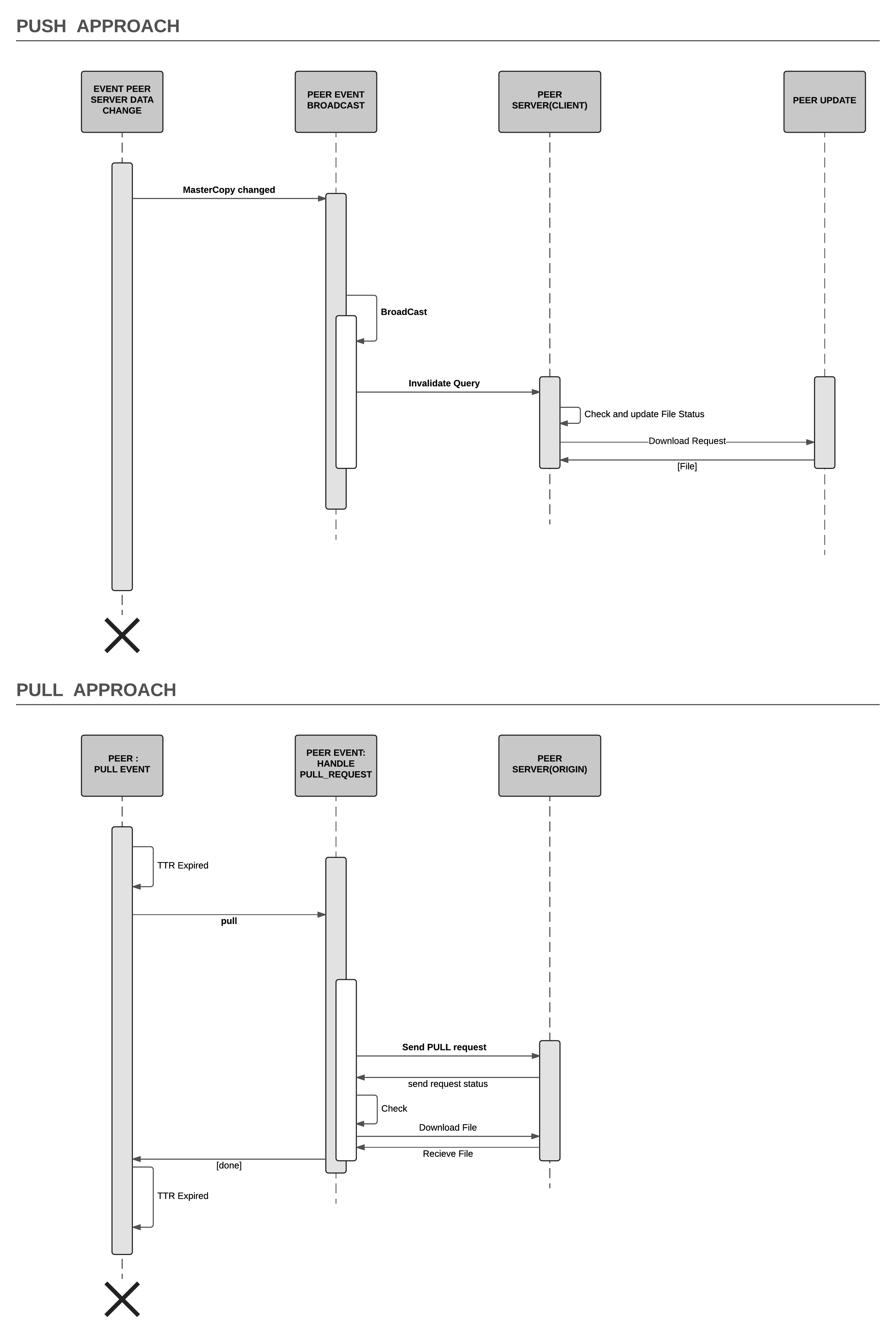
# PULL BASED:

1. The algorithms checks every cached file for every specified TTR. So the peer will poll its file by sending the request to Peer Server.

2. The peer Server checks and if the poll message is not valid then it will send the new meta data.

3. Upon receiving the meta data, it will update the file status to stale.

4. Only when user initiates refresh, then it will pull the file from the server.



# 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.

4. Once file is stale in pull, then it won’t poll the request until it downloads the file. So it avoids too many poll requests.

# Possible improvements and extensions to your program

1. We can store the files meta data into database, so that application will remember what it has.
2. We can implement the auto-refresh for PULL mechanism, so that user do not require to refresh manually.
3. All its neighbors (initialized on bootup from the peer configuration file)

## Problem report:

1. Watch thread is tested in Unix and Mac, but we need to modify it to work for Windows.
2. File name is considered as unique, so if you put two file of same name it will not work as desired.
3. Peer Downloads file should be there in each peer folder, otherwise it will not work

