# CS550 Programming Assignment 3 (PA#3)
# **Maintaining File Consistency in the Hierarchical Gnutella-style P2P File Sharing System**

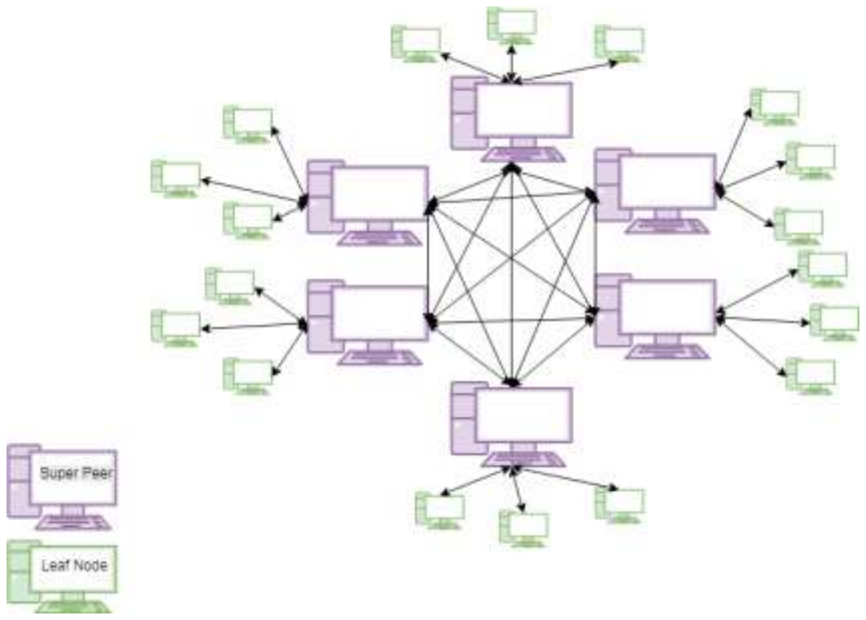
## **Abstract:**

Gnutella was the very first decentralized P2P file sharing system. In Gnutella, there are no explicitly defined servers but rather the clients are used to search and retrieve data using messages. These clients are called peers where they communicate directly with each other. Since there is no central indexing server, the search is done in a distributed manner. Each peer also maintains the list of neighbouring peers.

In the hierarchical Gnutella-style P2P system, there are two types of peers. One is called a leaf-node where it has only one connection open. And the other is called a super peer where it acts like a proxy indexing server and is connected to the other super peers. All leaf nodes register the files in the connected super-peer. Whenever a query request comes from a leaf node, the super-peer first searches the local storage i.e. checking whether it is present in its connected leaf nodes. If the file is not present in the local storage it broadcasts to all the super-peer neighbours.

In the programming assignment we are adding two types of File consistency mechanisms to the hierarchical Gnutella P2P File sharing system. The objective of the assignment is to ensure that all the copies of the file in the P2P system are consistent with one another.

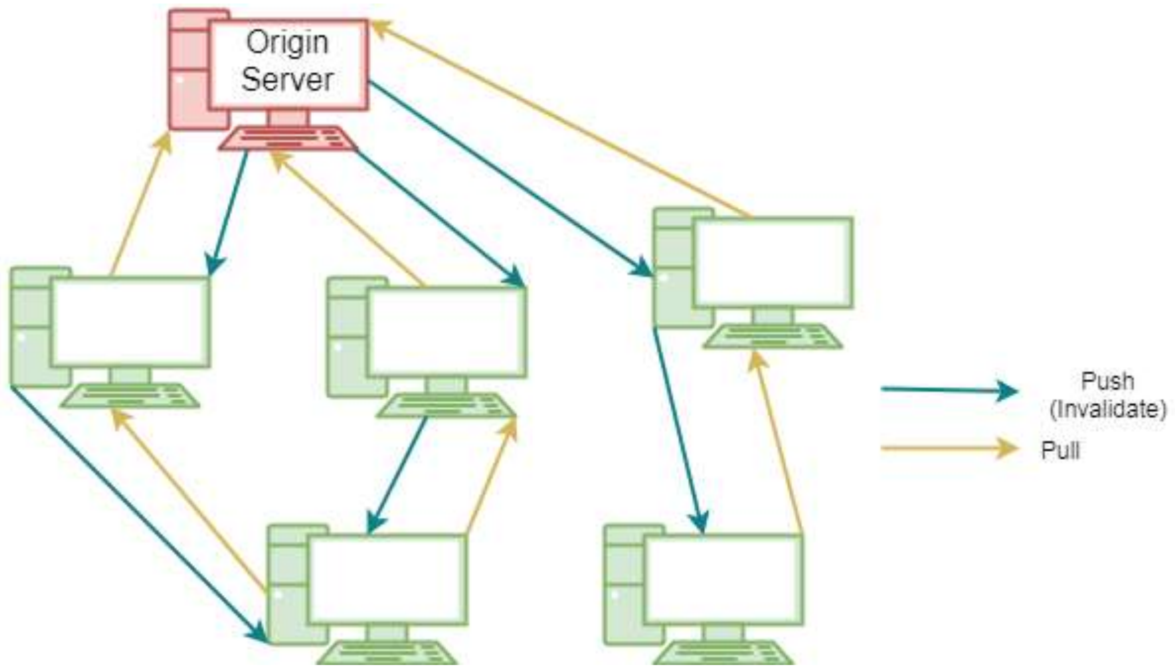
## **System Design of Hierarchical Gnutella:**



The Above diagram illustrates the hierarchical style P2P network where there are two types of peers, namely, the leaf nodes and the Super-peer.

1. Super Peer: The super peer behaves as a proxy indexing server for the leaf nodes. It maintains the list of all the files present in the local storage i.e. present in the connected leaf nodes. It also maintains its neighbours list.
2. Leaf Node: The leaf node registers its files to the connected Super peer.

**System Design for Maintaining File Consistency in Hierarchical Gnutella:**



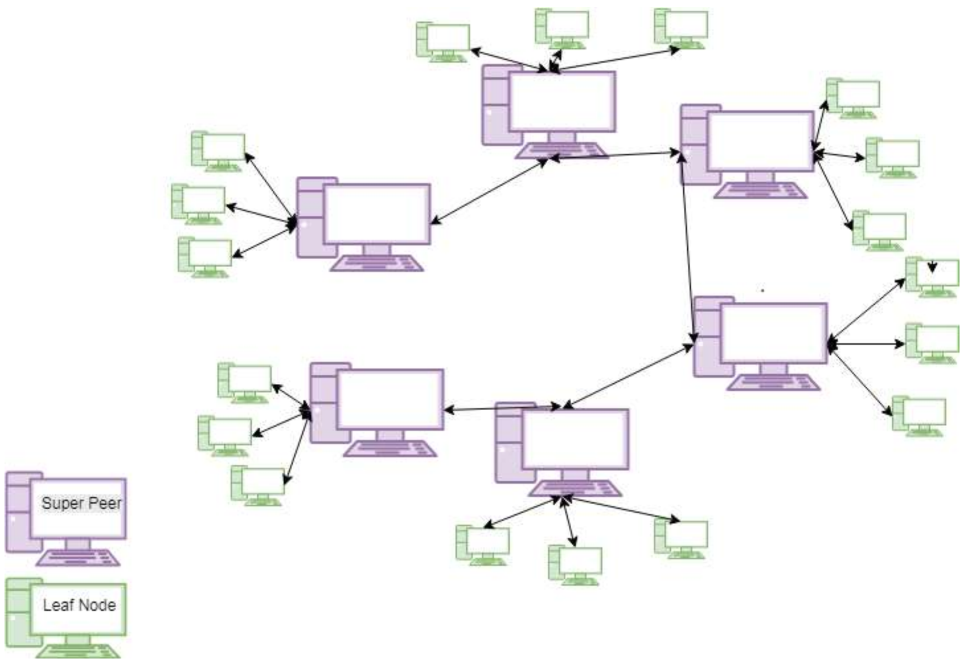
**Implementation:**

*Configuration:*

We initialize and define the topology structure in the config properties file. We also declare the peer details, i.e. associated port number, IP address and the shared directory in the file.

The hierarchical gnutella has super peers. We define super peers as the peers which have more than one neighbours. The peers with only one neighbour are the leaf nodes that are connected to the super peer (the neighbour which is mentioned).

An Example of a Linear Topology:



We are implementing two approaches to maintain file consistency in the Gnutella network, namely push and pull approach. We must set only one of these in the configuration file. We must also mention the TTR value, i.e. Time to Refresh. This value is used in the pull approach, that is, the peers polls the origin server after certain time as specified in the Time to refresh property.

*Components of the Gnutella P2P System:*

1. Peer:

   There are two types of peers in the system, namely a Leaf node and a super Peer.

   i. Leaf node: A leaf node is a normal client which registers its files to the super peer. It has methods to search for a file and download it.

   ii. Super Peer: A Super peer acts like a proxy indexing server to the leaf nodes. It maintains the list of all the files which are registered by the peer. Additionally it also maintains the list of all the neighbouring super peers.

2. Peer Interface:

   a. Obtain:

      The method used to download the file from one peer to another.

   b. Query:

      This is used to send out a query message asking for a file, where this is first checked in the local storage (Super peer's proxy index server) if found it returns else it broadcasts the query message to all the neighbours of the Super peer.

   c. Registry:

      A leaf node will register all the files it has with the super peer.

   d. Invalidation:

      This is used in the pull approach which push messages sends invalidate message to all the leaf nodes, where in turn the leaf nodes checks the consistency of the file.

   e. Poll Server:

      This is used by the leaf nodes to poll the server to check for file consistency.

3. Hit Query:

   When a client sends a query asking for a desired file, it is broadcasted when the file is not present on the local storage. When the desired file is found on the network, the Hit query is executed which basically returns the location of the desired file i.e. the leaf node's IP address and port number. It also maintains the traversed path of the query message.

*Adding File Consistency to the Gnutella P2P System:*

File consistency are implemented using two approaches.

1. Push:
   Whenever the master copy of the file in modified in the leaf node (Origin server), the origin server broadcasts a PUSH message for the file which invalidates the copies of the file present in other peers. On receiving the invalidate message, each peer will check if it has a copy of the file, if the file is present, then it is considered as stale and its discarded.
2. Pull:
   In the pull approach, each peer can initiate a Pull message to the origin server to see if the cached file is stale or not. On receiving the pull message, the origin server will check whether the master copy is updated or not. If the origin server has a newer version of the cached file, it sends back a file out of date message and the peer discards the cached copy and notifies the user.

**Design Trade-Offs:**

1. Currently the network topology is statically defined and initialized using a configuration file.
2. There is no sophisticated searching algorithm present, but rather we only use exact string match.
3. The invalidate message currently broadcasts to all the peers in the network irrespective whether they have the peer or not.

**Further Proposed Improvements:**

1. One area of improvement is removal of static initialization. One could enhance it by configuring the peer in the network dynamically.
2. The ping and pong messages have not been implemented in the above system where one could survey the network and get to know the neighbouring peers, instead this is also initialized in the configuration file.
3. Time to live parameter is constant in the whole network, it does not grow as there is growth in the network.
4. The origin server can maintain a list of all the peers which have the cached copy and can only send push messages to those peers thereby reducing the network bandwidth.