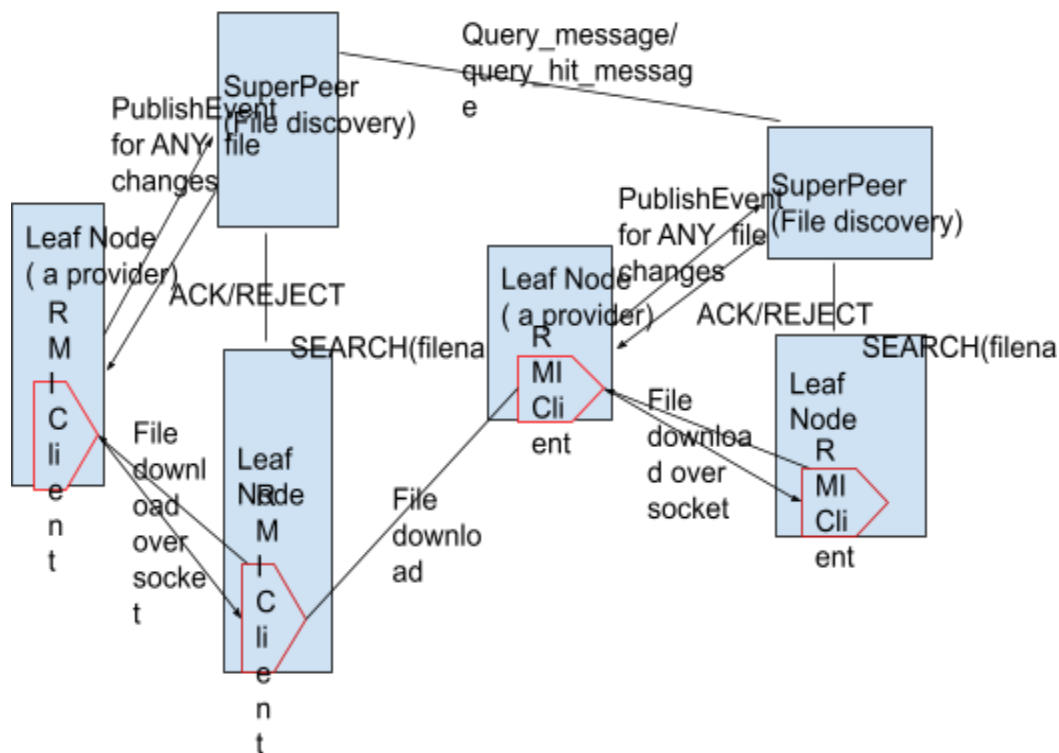


## DESIGN DOCUMENT



Fig(a). P2PSystem network topology

## OVERVIEW

### File Download

The overall design of the project is as above. The RMISuperPeerClient, acts as the client part of the LeafNode server requests for the download of files once it is aware of the file server host and port address and this happens over TCP sockets. This was chosen design to leverage the multithreaded server and multiple client scenario for file transfers. RMI protocol was used for all interactions between SuperPeers and LeafNode due to its lightweight multithreaded ability.

### QUERY MESSAGE

When a query message is sent by the leaf node it first interacts with the super peer it is connected to and checks if the super peer knows any leaf nodes in its own cluster that may have the file associated with the query. If not the message is broadcasted by the super peer to all other super peers connected to it in the Gnutella network and the time to live is decremented by 1. The Query message is also stored in the super peers state so that any future requests on for the same Query message are ignored to break the redundancy because this super peer has already broadcasted to the other super peers in the network. The 'associated array' cache being used is a least frequently used cache so if some messages are not queried often they may be removed from the cache and the super peer may introduce some message redundancy

should these messages get forwarded to it in the future. The cache eviction policy was chosen because less frequently seen messages may be removed from the super peer to save working memory of the system.

## **QUERY HIT MESSAGE**

### *During Query Process*

After having broadcasted the messages to all the super peers in the network a 'serving' super peer checks to see if the filename associated with the query is in its local "coordinator" map. If so, some leaf node connected to this gnutella super peer still has the file and hence a query hit message is sent back to the super peer that had originated in this query message on the 'serving' super peer. Again, the time to live is decremented and forwarded to the originator of the request.

### *On receipt of Query Hit Message*

The originator of the message can be found in the associative array that records the requester of the query message at the 'serving' super node. Again, this the same cache has a least frequently used eviction policy so it may be possible that query hit messages may not be forwarded at all if the message has been evicted from the cache.

## **Publish File Event**

When there changes made to a file such as create or modify then these events are broadcasted to the SuperPeer that will record these changes internally in a thread safe hashmap that is guarded by synchronized. The hashmap has keys that designate the filename and the values are all the host and ports of peers where this file has been discovered either by creation or modification. On modification, the software is not aware of which version of the file is stale so all the peers that have last registered will continue as values for the coordinator map.

## **Search**

The search for the filename is an exact match search on the filename.

The oldest(in time) peer to have been discovered for this filename at a super peer will be returned as a candidate for other peers to download the files. Peers are not deleted from this map unless they explicitly call deregister. The files will be directly downloadable by the requesting leaf node from the serving leaf node by calling Obtain. This Obtain runs in an asynchronous thread that uses sockets to download the file.

## **FUTURE WORK**

This design works well for a single coordinator but if the coordinator fails the entire system is out of reach. It has a single point of failure which shouldn't be the case in the P2P scenario. The peers should for a distributed hash table rather than a single hash table to make this fault

tolerant and also scalable in the set of keys being shared amongst subgroups of peers which would reduce the number of messages that are passed for discovery.

Stateful Load balancing using network bandwidth could be performed on the set of SuperPeer to reduce latency in the scenarios where for a set of peers that have knowledge about a filename could split the load of download requests.

Also, the design could have included leader election to choose a new SuperPeer inside a Gnutella cluster in terms of a failure or have had more than coordinator as a redundancy.

The files could also be sharded in such a way that a certain set of SuperPeers are directly responsible for the file by the filename by applying a hash function an evenly distributing the files. Then the super peers will apply replication using a finite state machine to make sure two Gnutella Clusters sharing the same shards will always be synchronized using a consensus algorithm.

More work can also be done to support timestamping as the files are out of sync on some Gnutella leaf nodes having stale data but sharing the same file server name.