P2P Distributed File Sharing System - I
Design Specification

Agnivo Saha, Anurag Anand, Biswajit Paria, Prithwish Mukherjee

- **Components**
  - There are 2 super-peers, one primary and one secondary. Fault-model is that only one of the 2 super-peers can crash, in which case the secondary super peer will become the primary super-peer.
  - For the purpose of experimentation, a node must have a flag with values 0 or 1 indicating whether it qualifies to be a normal node opeer respectively. Whether or not a node qualifies to be a peer will depend on parameters like system memory, processing speed etc.
  - *Super-peer*:
    - Its IP address is known to all of the nodes in the system.
    - It decides which of the nodes are normal nodes and which are peers.
    - It will choose new peers if the number of peers is less than a constant fraction of the total number of nodes.
    - It will contain information about online peers and nodes and will keep their status updated using heartbeat messages.
  - *Peer*
    - It has list of files shared by a subset of nodes in the network.
    - It receives file search requests from other nodes. If it has an entry for the requested file, it sends the IP address and path of the file to the node requesting.
    - It maintains the set of active nodes among the nodes which have shared their files with it. Periodically, it removes the entries of the nodes which are disconnected for long.
  - *Normal node*
    - A normal node will contain which peers have got its file location information and super peer will use this to replicate the information if one the peers goes down.
    - Search will be distributed as search queries will be sent to only a fixed number of peers given by the search quorum of a node.
    - A normal node has a share quorum and a query quorum. The size of the share quorum is k and the query quorum is of size $k + 1$ when the total number of peers in the system is $2k$.

    - A normal node shares its file list to be indexed at all the k peers in its share quorum.
    - As in between any share quorum and query quorum of two nodes there is at least one common peer, so if we query in $k + 1$ peers, then we ensure that all files are searched.

- **Variables**
  - Super-peer
    - String super_peer_ip
    - Set<node_ip> peers: the set of active peers.
    - Set<node_ip> normals: the set of active normal nodes.
  - Peers
    - String peer_ip
    - Map<file_hash, (file_name, valid_bit, node_ip)>: maintains the file location and its validity.
    - List<string> file_names: list of file names it has in its memory.
    - Queue<(node_ip, query)> requests: queue of outstanding requests.
  - Normal node
    - String super_peer: super_peer ip.
    - Set<String> replica_peers: The peer IPs with which the shared file list has been shared.
    - Int query_quorum: the size of the read quorum. While querying, we randomly query query_quorum number of peers.
    - List<string> shared_files: list of files that have been shared.

- **Messages**
  - Super-peer
    - I_AM_SUPER(node_ip): Informs other nodes that it is the super peer (in case of a super-peer failure)
    - BE_PEER(peer_ip): Instructs an eligible node to become a peer.
    - RETURN_PEER_LIST(normal_ip, peer_list): returns peer list to a normal node that has just come online.
    - ARE_YOU_ALIVE(peer_ip): to periodically check whether the peer is alive and kicking.
  - Peers
    - I_AM_PEER(super_ip): to let know that the node agrees to be a peer.
    - I_AM_ALIVE(super_ip): to let know that the peer is online.
    - FILES_SHARED_ACK(node_ip): to acknowledge a normal node that the file list was succesfully shared.
    - ARE_YOU_ALIVE(node_ip): to check whether a normal node is alive.

  - Normal node
    * I_AM_ONLINE(super_ip): to let know that it has come online.
    * I_AM_OFFLINE(super_ip): to let know that it is going offline, and its files need to be invalidated.
    * I_AM_ALIVE(peer_ip): to let know that the node is still online.
    * SHARE_MY_FILES(peer_ip, shared_files): to share its file list with the peer for making them searchable.
    * DELETE_MY_FILES(peer_ip, deleted_files): to delete file names from the shared list when a file is deleted locally.

- **Fault Tolerance Model**
  - Can tolerate one super-peer crash.
  - There can be inconsistent information about the shared files due to crashes but eventually it will be consistent.
  - During faults searches might not return files even when they exist and match the queries. But will work well again once the effect of the fault has been settled.
  - Searches may return files which do not exist, as the nodes can go offline at any arbitrary time. But eventually the non-existing files will be removed from the shared list.

- **Platform**
  - Operating system: Linux based
  - Programming language: Python
  - Interface: Command line

- **Demonstration**
  - Multiple processes will be spawned, each process representing a node. The spawned processes start their own sub-processes performing almost independent functions like sending heartbeat messages, searching the file lists, sharing files, managing the file list, etc.
  - The flag of the node representing whether it can be a peer will be generated randomly. 10% peer nodes and 90% normal nodes.
  - A list of file names will be generated randomly and distributed randomly among the nodes.
  - File queries will be generated randomly. Queries will be generated such that the results are substantial.
  - Nodes come online and go offline according to a poisson process with predefined parameters.
  - Search and share actions also occur according to a poisson process.