Design Document

This assignment includes designing a simple Gnutella P2P file sharing system. This file sharing system has one components: Peer. The Gnutella is a decentralized peer-to-peer system, where peers are connected to one another forming a network.

Peer: A peer act as a server and a client. As a client, it can request for the file information from its neighbour and as a server it allows other users to download files from it.

To implement this simple Gnutella P2P file sharing system, Java RMI API, an object-oriented equivalent of RPC has been used. The communication between the client and server happens through the remote method invocation. This RMI application comprises of components which make a Peer act as Server and a Client at the same time. The Server program uses remote objects, and make these remote objects accessible to the Client when methods are invoked on these objects. The Client program obtains remote references and invoke methods. The communication between Peers is provided by RMI and passes information back and forth, often referred to as distributed object application.

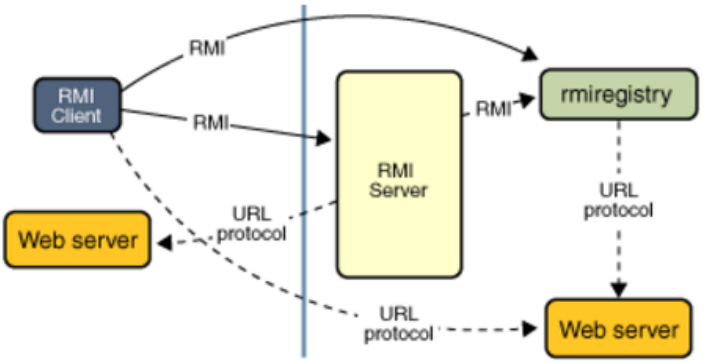

Fig 1. Mechanism of RMI

This application has one section: Client

Client: The Client section contains four elements, *InClientIF*, an Interface for the Peer, *PeerImp*, Implementation class for peer interface *InClientIF*, *Peer*, a client program for the peers, and *MsgDetails*, a getter setter class for message among the peer.

PeerImp: This implements the method of the Client Interface.

- *downloadFile (String fileName, String peerName):* When one peer wants to download the file from another peer, downloadFile method is invoked. When peer receives a download request, it behaves as server for the peer and the requesting peer act as client. This method writes the requested file in the requester's folder path.
- *query ((String msgID, int intTTL, String fileName)): When* one peer wants to search a file in the Gnutella file sharing system, query method is invoked. The request gets broadcasted among its neighbours across the network.

- *queryhit ((String msgID, int intTTL, String fileName, String peerIP, int portNum)):* When the search message is broadcasted among the peers, each peer search the file in their local index. Once the file is found, this method is invoked. The detail of the peer is sent for downloading the file for the requested Peer. Once the downloading request is received, a connection is established between the peers and file gets downloaded for the requester Peer.

MsgDetails: This implements the getter setter class for message details. This is used to fetch details of the message such as message Id, filename, ip address, TTL and port number.

Peer: This handles the Peer implementation for the application. This runs with the main method which takes port number as input while running the application. When the peer is executed two threads run simultaneously, they are: PeerBroadCastReq, and PeerServer. PeerClient communicates with the indexing server, PeerServer keeps listening to the assigned port for any download request, and PeerBroadCastReq handles the broadcasting of the search request among the neighbour peers.

The Peer provides the user the following options:

- Register neighbour port
- Lookup for files in N/W
- Show connected peer

While registering the peer with application, it looks for its neighbours in the network file (maintained in text congif file where different topology can be maintained such as linear and star) and uses that information for connection to the Gnutella system. The neighbour information for the Peer is stored.

While searching for the files, client enters the filename. On the search request, the search message is broadcasted among the neighbour which was maintained while registering. If the file is found with some peer, connection is established and the requested file is downloaded. When the download request is initiated, the PeerServer thread responds and the files is downloaded from the other peer.

While initiating Show connected peer request the peer from the application displays the neighbours it is connected to.

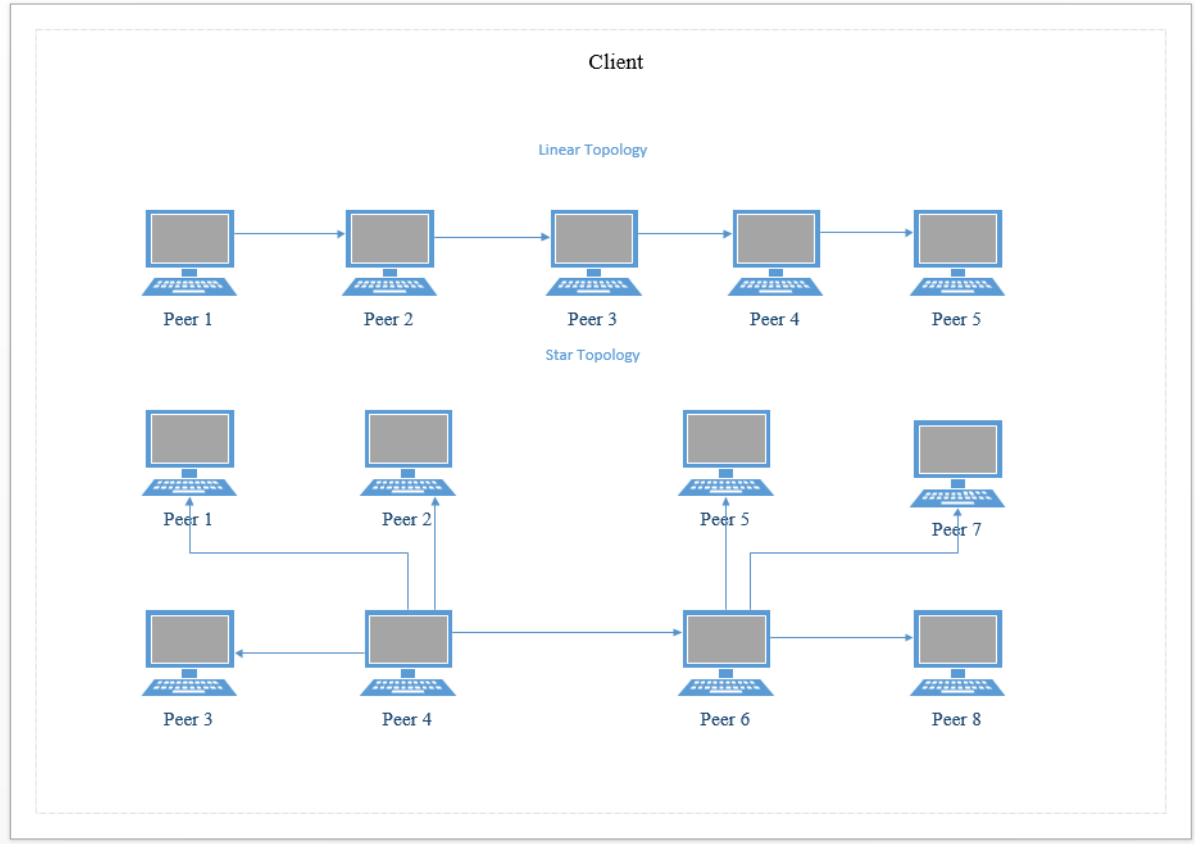

Fig 2: The Gnutella network of P2P file sharing system

Possible Improvements and Extensions.

1. In the current design, user inputs the port number while registering with the Index Server. It can be designed so that the port number is assigned by the server. This will avoid the manual entry of peer name and port every time done by the Client.

2. The display can be improved. GUI can be built so that it makes simple for the User to use the application

3. While receiving any request by the server no buffer has been used. Adding the buffer at the time of receiving request by the server will help server handle too many requests. The request will not die.

4. In the current authentication of the client has not been implemented. Anyone can connect to the server. The index server can authenticate only through authorized connection. To achieve this a signup option can be provided to the peer, which will avoid any unauthorized peer to access the application.

5. The application is using flat file to maintain the network details for the application which is static. Dynamic network approach can be built. This will take care of the data loss which might occur due to server crash or at the time of server restart.

6. While accepting the download request from one peer and establishing a connection, the availability of the peer is not checked. The availability check can be added, so that will it will download the file only for the available peer.