Gnutella Peer2Peer File System

Through JAVA RMI

Manohar Kotapati

CS Graduate Student

TEXAS TECH UNIVERSITY

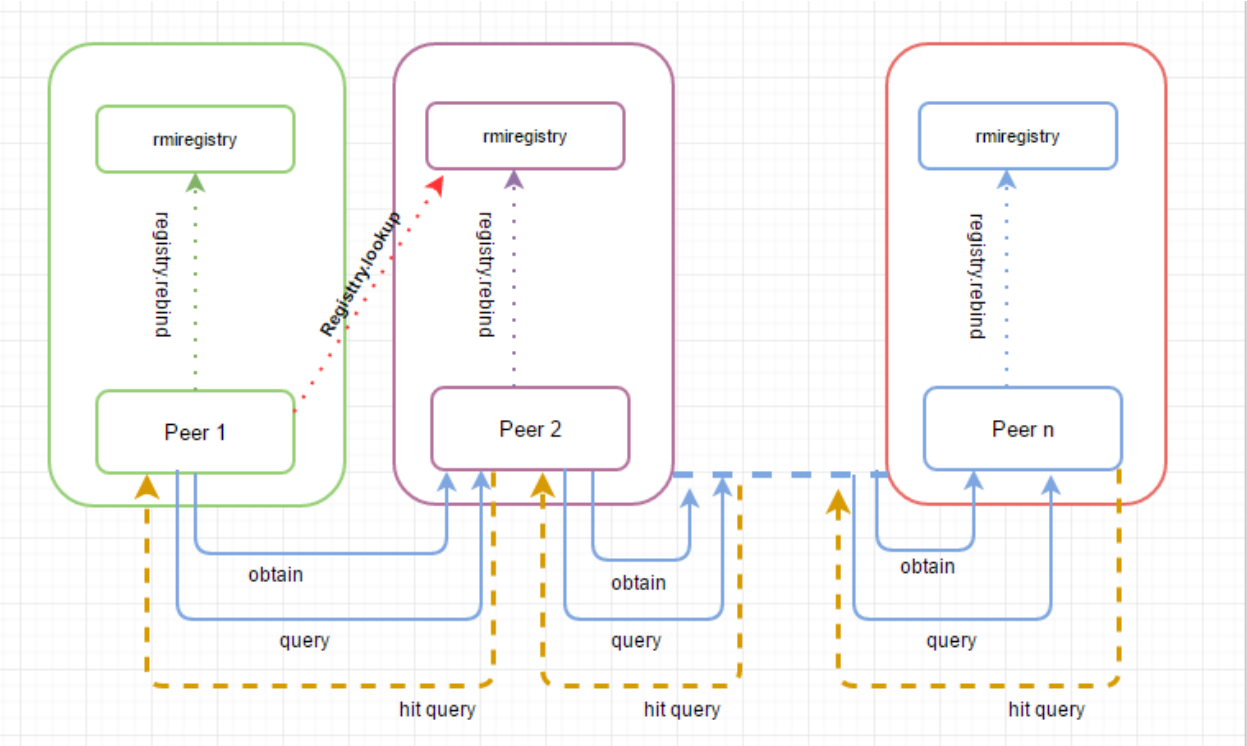
Submitted on 18th April, 2016.

Table of Contents

1. Project Statement

Gnutella-style peer-to-peer (P2P) system: Each peer should be both a server and a client. As a client, it provides interfaces through which users can issue queries and view search results. As a server, it accepts queries from other peers, checks for matches against its local data set, and responds with corresponding results. In addition, since there's no central indexing server, search is done in a distributed manner. Each peer maintains a list of peers as its neighbor. Whenever a query request comes in, the peer will broadcast the query to all its neighbors in addition to searching its local storage (and responds if necessary)

2. Design



Gnutella style Peer to Peer file management design with RMI

3. Implementation

3.1 Config

As we are not implementing the dynamic initialization of the network as in Gnutella. To keep things simple, assume that the structure of the P2P network is static. Declare the neighbor peer details in the config.properties file. Maintain one config file for each topology i.e. Mesh topology and star topology.

E.g. Mesh topology

```
20
21
22  # Neighbor details
23  #3X3 Mesh
24  peerid.1.neighbors=peerid.2,peerid.4
25  peerid.2.neighbors=peerid.1,peerid.3,peerid.5
26  peerid.3.neighbors=peerid.2,peerid.6
27  peerid.4.neighbors=peerid.1,peerid.5,peerid.7
28  peerid.5.neighbors=peerid.2,peerid.4,peerid.6,peerid.8
29  peerid.6.neighbors=peerid.3,peerid.5,peerid.9
30  peerid.7.neighbors=peerid.4,peerid.8
31  peerid.8.neighbors=peerid.5,peerid.7,peerid.9
32  peerid.9.neighbors=peerid.6,peerid.8
33
34  #6 star
```

3.2 gnutellaP2P Package

This package consists of the necessary classes for the peer to participate in the network.

PeerInterface.java – Interface with three remote methods

- *obtain(filename)*
- *query(fromPeerId,msgId,fielname)*

Peer.java

- It creates the RMI registry for PeerInterfaceRemote on a specific port that provided by the user. So that peer can act as a server on rmi://<ip>:<port>/peerServer
- And act as client by interacting with the user and provides the services of file search. The steps of interaction are as follows
    - Enter the directory path that is going to be shared with other peers

- Prompts for file name to search
- Get the neighbor peers from the config file and get the result from those peers.
- Display the list of peers containing the requested file.
- Prompt to enter the peer id from where the file should be downloaded
- Establish connection with peer and download the file.

PeerInterfaceRemote.java – Implemented the logic for the remote methods defined in the PeerInterface.

- *query(fromPeerId,msgId,fielname)*

  This remote method takes the filename, message id as arguments. It checks whether the peer already processed that *msgId* or not. If it is already processed, it will skip the process and return to the called client. If message id is not there with the peer then it will perform the search locally. It will establish the connection with its neighbor peers and will get the results. All these results will be sent back to the called client.

- *Obtain(filename)*

  This remote method takes the file name as an argument. It will send the file contents as bytes and client will create a new file with these bytes of data.

HitQuery.java

This class is used to handle the details of the result of every query. It contains properties of the peer details of the requested file and the path through which search performed.

3.2 Performance Evaluation.

Assume mesh network of 3X3 peers. Find the response time of Peer1 for 1000 requests in following scenarios

- When it is the only peer performing the search in the network.
- When there is one more peer performing the search in the network
- When there are two more peers performing the search in the network

Modify the class PerformanceLauncher.java file as per each scenario given above.

PerformanceEvalution.java file handles the request for 1000 sequential requests and find the average response time.

4. Instructions to run the software:

Please follow the below instructions to run the software.

1. **Config:** Modify the config.properties file as per the required network topology
2. **Run the peer**: Run the class Peer
   E.g. Java gnutellaP2P/Peer
   We can run more than one peers with different ports in the same machine.

Note: If you are testing the whole software in one machine, then maintain separate command interfaces for server and client.

5. Improvements

Currently peers are configured in config file. It can be enhanced by configuring the peers in the network dynamically.
Algorithm used to search array can also be improved.