**Peer-to-Peer With Distributed Index (P2P-DI) System for Downloading RFCs**

Arunkumar Krishnamoorthy – akrish12

Harish Annamalai Palaniappan - hpalani

**Introduction:**

We are implementing a peer-to-peer system with a distributed index. All communication among the peers or between a peer and the registration server will take place over the TCP. Peer-to-Peer system is used to partition the tasks between the peers. The main advantages of P2P system are that it is easy to set up and all nodes can act as clients and servers. It is also quite common in places where they don’t use a dedicated file server.

**Overview:**

In this project, we use a registration server and 6 peers, which can act as both the client and the server. The peer must register with the Registration Server to get to know the active list of peers. This enables the peer to contact the other peer servers, from which it can either download the RFCs or get the RFC Index. When the peer wants to leave the system, again it must inform the Registration Server before leaving. Unique cookie number is assigned to every peer, when the peer registers for the first time in the Registration Server. In the peer server and in the RS Server, the data is stored using the Linked List data structure.

**Implementation:**

The peer to peer system is designed in an user interactive manner. The client gets the host name from the user, its own RFC server’s port number and the message type also from the user.

The message types are Register, PQuery, Leave, KeepAlive. Depending on the message type, the Registration Server will carry out the specific operations.

When the peer is registering for the first time in the Registration Server, it won’t have a cookie, hence an unique cookie number is generated and assigned to the cookie. The client stores the cookie information and data in a file, so that even after the execution is over, the client can identify with the help of the cookie number. When the process of appending the data of the peer to the linked list takes place and if the data of the peer is already present, then we get a message saying ‘Already Registered’. Each record of the peer list consists of the hostname, which is received from the user, the port number, which is also received from the user. The active flag is set to True. The Time to Live variable is set to 7200 seconds. The recent register time is set with the help of the time module. Instance is made use to find the number of times the peer has been active.

The PeerList class is used to store and display the data contents of the peer. The pointers of the various peers are stored in the NodePeer class. The checkpeer\_List() function is used to check about the active peer list and peer if they’re in active state or not. When the peer leaves the system, the activity flag is changed to False with the help of setinactive(). The calculate\_instance function is used to calculate the number of times the peer has been active. Every instance the peer becomes active, the value of the variable is incremented by 1. The check\_peer() is used to check whether the peer has registered or not. If there are no peers, then the message ‘No peer registered’ is displayed. Register is used to register in the Registration Server, PQuery is used to know the list of active peers and Leave is used when a peer wants to the leave the system.

Each peer locally maintains a RFC Index which initially contains only the RFCs stored locally at the peer. Each record of RFC consists of the RFC number, the title of the RFC, hostname of the peer containing the RFC and the TTL field.

The request message format for Register Message Type is:

GET Register P2P/DI-1.1 <cr> <lf>

Host <Host\_name> <cr> <lf>

Port <Port\_No> <cr> <lf>

Cookie <Cookie> <cr> <lf>

The message format for PQuery Message Type is:

GET PQuery P2P/DI-1.1 <cr> <lf>

Host <Host\_name> <cr> <lf>

Port <Port\_No> <cr> <lf>

Cookie <Cookie> <cr> <lf>

The message format for Leave Message Type is

POST Leave P2P/DI-1.1 <cr> <lf>

Host <Host\_name> <cr> <lf>

Port <Port\_No> <cr> <lf>

Cookie <Cookie> <cr> <lf>

The request message format for KeepAlive Message Type is:

POST KeepAlive P2P/DI-1.1 <cr> <lf>

Host <Host\_name> <cr> <lf>

Port <Port\_No> <cr> <lf>

Cookie <Cookie> <cr> <lf>

The RFC Client and Server is used to download the RFC files and write them into CSV files. It also helps in maintaining an index. Threading is used in the RFC Server to handle multiple users at the same time. The ClassRFCIndex is used to maintain the RFC number, title of the RFC and the hostname of the peer. The RFC Server is contacted for two main purposes. One is to get an index of the RFC files stored in the peer and other is to download the RFCs. To download files from the other peer, the RFC Server must be contacted. Every time when a peer contacts the RFC Server, a new thread is created. The RFC Client is used to query the RFCs. It is used to contact other servers and obtain the RFCs. The class PeerRFCList is used to add the peer index to the list and the corresponding message is displayed. The show\_rfc\_list is used to display the files and the delete operation is carried out by the delete\_rfc\_list. The function load\_data is used to write data into the CSV files. Incase it’s the first instance , the data is written without any issues, in case of duplicate data, it displays as ‘Already Present’ indicating duplicate data. When the action is RFCQuery , it is used to get RFC Index from the other peer and when action is GetRFC, it is used to download the specific RFC. The while loop is used to indicate the successful downloading of data.