P2P DHT

Programming Language: Python

Python Version: v.3.6.0

YouTube Demo: https://www.youtube.com/watch?v=ZM65C-gIgws&t=88s

Explanation with the same graph shown in project requirement:

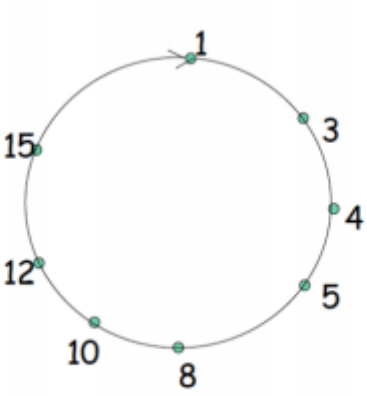

Figure 1. DHT configuration.

In this instance, my implementation will create 8 processes (1 for each peer). For each peer, it will create a peer class to store data names and relationships. Stored data include peer names, successor names and second successor names. Predecessors are not going to be initialized when program start, they will only be noted by ping requests.

At the beginning of program running, two threads will be created, one for UDP listening and the other one for TCP listening. And the peer class will be assign to these two threads so that they have ability of operating data in peers.

Ping Request

After a while, when time passed beyond interval, peer will send a ping request to its successor and another ping to its second successor to know their living status. Time interval is initialized to be 5 seconds, every time peer received correct response from its successor, time interval will be incrementing to be squared-root 2 of its old value, therefore if both successor and second successor are correctly being detected as alive, time interval will be doubled until it reaches its maxim value. Maxim value of time interval in this code will be **20 seconds** for testing, because no one want to want to long. But if it is deployed into real world, I will set the maxim value to be **5 minutes** to save the data flow.

Ping request will be sent or received using UDP, they will use the same socket of peer number plus 50,000. When server listened a request, it will create a thread to send a response message and assign the same socket to it using UDP too.

File Request

File request here is using TCP transmission. Here we use peer 12 as example, if peer 12 want file 0269, the file name 0269 will be hashed using module 256, turned out to be 13. Then peer 12 pass the request to peer 15, peer 15 found that 13 is between 12 (its predecessor) and 15 (itself), then it

contains the file and will send file directly to peer 12 (address of original peer will always be contained in message, so that the peer which has the file can know who want this file and able to send back directly). A trick here for find if a file with large number is stored in the first peer, here is peer 1, is that if a predecessor has larger number than it and file module result is greater than number of predecessor, then the first peer will contain the file.

Departure

When a peer wants to departure from network, it will tell its predecessors (predecessor addresses were learned from ping requests). Information of successors of this peer will be deliver with message sent with TCP, so that its predecessors can know how to reset their new successors and new second successors. After predecessors are noted, they will send back acknowledges with TCP to peer to inform it they have already known. And they will ping their new successors with UDP immediately. After both first predecessor and second predecessor send back acknowledges, peer will close UDP and TCP socket then close the program.

Kill

When UDP ping request do not receive response, with timeout interval 1 second, it will resent ping request again. After duplicate request send 3 times, the peer which should received this ping request will be identified as dead. Then program will command TCP thread to reset its new successor and new second successor.

Synchronize

There is a synchronize problem when a peer departure from network. For example, if peer 8 departure, peer 4 and peer 5 will be noted, then peer 5 will reset its new successors and peer 4 will ask peer 5 to get its new second successor. Therefore, if peer 4 ask before peer 5 reset, that how this synchronize problem potentially happens. My design is that peer 4 will check if the response from peer 5 (the new successor) is the same as its old one, if that is, it implies that peer 5 has not been reset completely, so it will resend the ask request again.

Choice of retransmission times

The reason of choose 3 times as retransmission in ping request in simple, same as TCP quick-retransmission to ensure network working nice and quick!