**Design Document**

A Simple Distributed Hash Table.

**Pavankumar Shetty**

**CWID-A20354961**

**CS-550 Fall 2015**

**Main Campus**

**Table of Contents**

1. Introduction -----------------------------------------------------------------3
2. Languages and Platform ---------------------------------------------------4
3. Architecture -----------------------------------------------------------------4
4. Design principle  -----------------------------------------------------------5
5. Improvements and extensions --------------------------------------------6

# Introduction

* Distributed Hash Table is one of the important algorithms used in distributed systems. DHT is used in P2Psystems.
* Generally, Hash tables are used to store key value pairs. All it needs is a key, value and an efficient hash function, where hash function maps the key to some space in hash table and the value is stored. Thus helps in distributing the keys evenly across the table. However, it requires a lot of memory (or space) to accommodate the entire table, even if most of the table is empty. But it gives out the best time for searching a key from the massive table structure. So to accommodate large number of keys, we will have to divide keys in to different subsets, and map those subsets of keys to a bucket, each bucket can reside in a separate machine/node. Distributed Hash Table uses buckets to distribute the (key, value) pair. This is where DHT is different from Hash Table.
* It automatically distributes the load to all the existing buckets. There is no need of central server in this system.

Here we are trying to implement a similar DHT, but a simpler one.

It is done by using hashcode() and modulus operation on key i.e. hash function is (key.hashCode()\*primeNumber) mod n, where n is the number of servers.

Prime Number is used because primes are un-divisible by any other numbers. Primes are unique numbers. The product of a prime with any other number has the best chance of being unique, due to the fact, this property is used in hashing functions.

Overview and Key points of the experiment:

* 8 participants in this system.
* Each client acts as a server and client.
* Server exposes its distributed hash table for the entire system. And allows put, get and delete operations on its DHT.
* Client uses the interface and performs put, get and delete operations on different client’s server DHT. Client then connects to the peer who has the file and downloads the file from it.
* The main hashing used here is, (key.hashCode()\*11)mod 8 and later the server number is determined from this hashing function.

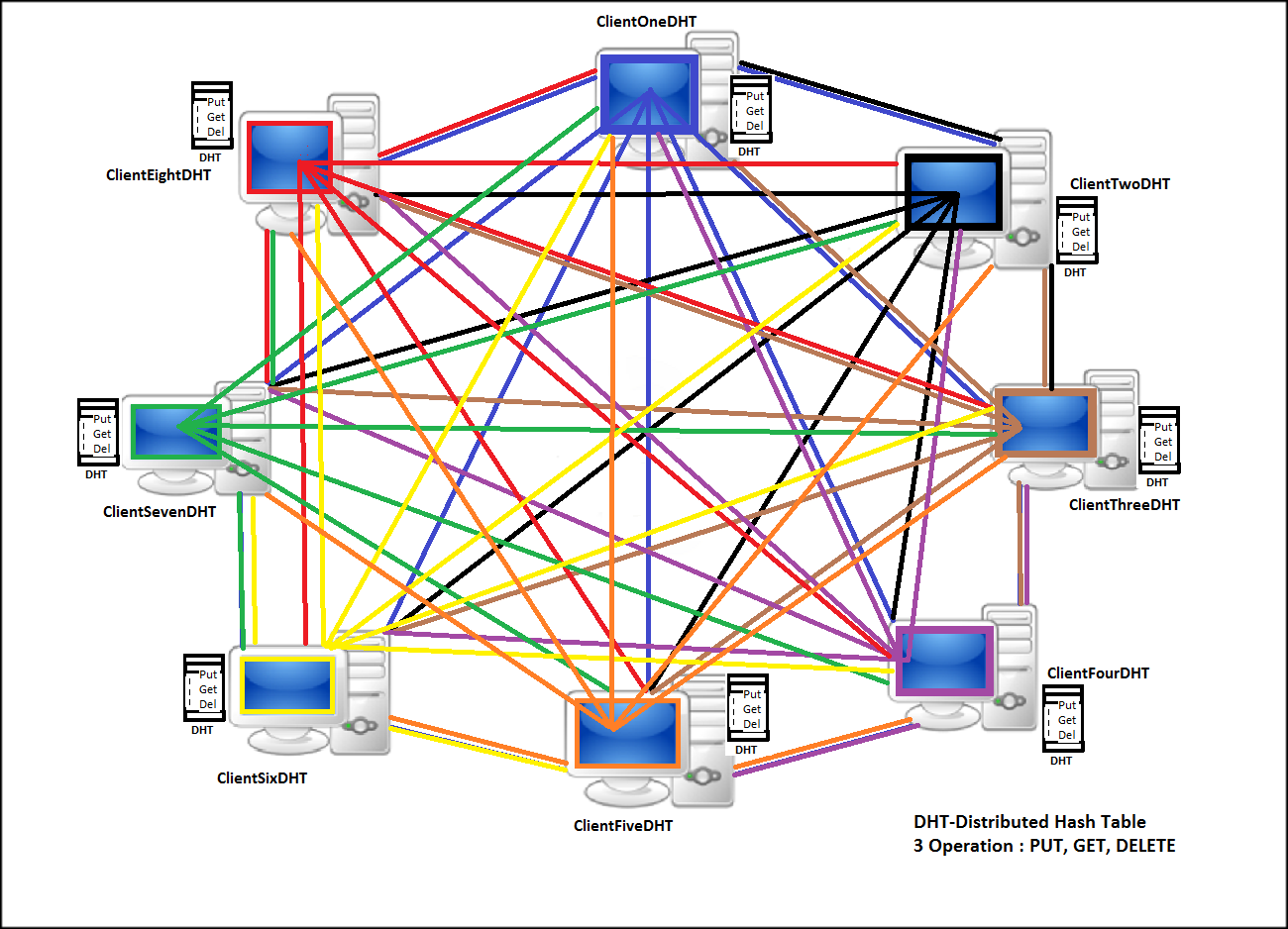
**Language & Platform**

**Language used:** JAVA programming, which covers MultiThreading, Socket Programming.

**Platform:** Built on Linux environment.

**Protocol used:**  TCP/IP.

**Architecture**

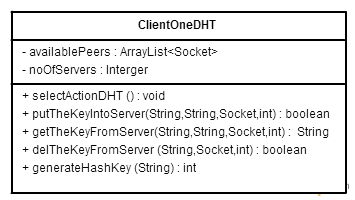


## Design Principle

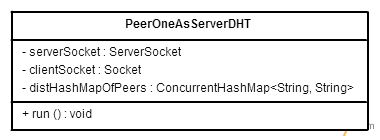
Source code implementation:

**CLIENT 1:**

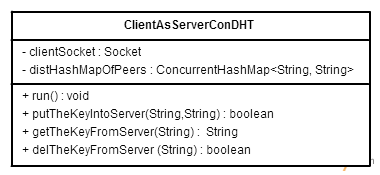
**Public class ClientOneDHT{}** implements

* Client performs 3 operations. Put, get, delete.
* Key is hashed and then server is determined. ****

**Public class PeerOneAsServerDHT{}** implements server part of client, which opens up its DHT for the system to access it and perform 3 operations on it.



**Public class ClientAsServerConDHT{}** implements Client server’s thread part to handle each operations handled by each client.



**Config.properties** file includes ipaddress of all the 8 clients.

**build.xml** file has been generated to automate the execution.

**Improvements and Extensions**

Any node failure in the entire system, can be handled as an improvement to the current experiment. And also maintaining a replica of all the key value pairs in each client could be considered as extension to the current version.