PROJECT REPORT

Domain Name System using Distributed Hash Table

Submitted by:

Team Number - 36

Virat (2019201033)

Sumit (2019201034)

Divyani (2019201028)

Ann (2019201024)

Javed (2019201035)

Deeksha (2019201068)

1 Introduction

The Domain Name System (DNS) is a hierarchical and decentralized naming system for computers, services, or other resources connected to the Internet. It is a standard protocol that helps Internet users discover websites using human readable addresses.DNS lets you type the address of a website and automatically discover the Internet Protocol(IP) address for that website. We need to implement a DNS using DHT. We are using the Chord protocol to set up the DHT.

2 Distributed Hash Table

A DHT is a decentralized storage system that provides lookup and storage schemes similar to a hash table,storing key-value pairs. Responsibility for maintaining the mapping from keys to values is distributed among the nodes, in such a way that a change in the set of participants causes a minimal amount of disruption. This allows a DHT to scale to extremely large numbers of nodes and to handle continual node arrivals, departures, and failures.

2.1 Properties

- Decentralized Autonomous - The nodes collectively form the system without any central coordination.
- Fault Tolerant - The system should be reliable (in some sense) even with nodes continuously joining, leaving, and failing.
- Scalable - The system should function efficiently even with thousands or millions of nodes.

2.2 DHT's Functions

- put(key, value)
- get(key)

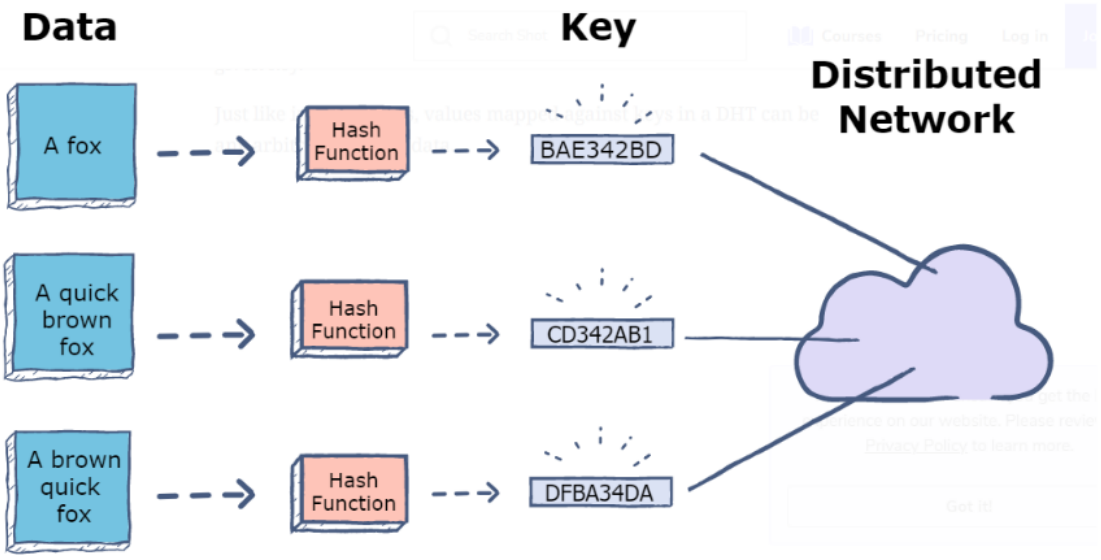

Figure 1: DHT

3 Chord Protocol

Chord is based on consistent hashing, which assigns hash keys to nodes in a way that doesn't need to change much as nodes join and leave the system. Chord is based on consistent hashing, which assigns hash keys to nodes in a way that doesn't need to change much as nodes join and leave the system. In a distributed system, consistent hashing helps in solving the following scenarios:

- To provide elastic scaling (a term used to describe dynamic adding/removing of servers based on usage load) for cache servers.
- Scale out a set of storage nodes like NoSQL databases.

In an N-node network, each node maintains information about only O(log N) other nodes, and a lookup requires O(log N) messages. Three features that distinguish Chord from many other peer-to-peer lookup protocols are its simplicity, provable correctness, and provable performance.

4 Implementation

We followed the approach from the paper: Chord-A Scalable Peer-to-peer Lookup Protocol for Internet Applications. The Chord protocol supports just one operation: given a key, it will determine the node responsible for storing the key's value. Chord does not itself store keys and values.

4.1 Phase 1 - INITIATION MODULE

This module sets up the network by running the Initialization procedure. Operations Performed are :

- Setting Predecessor of the node as NULL. Predecessor is the previous node on the identifier circle.
- Setting Successor of the node as itself.Successor is the next node on the identifier circle;finger[1]:node
- Initializing Data Structures.That is updating the finger table and updating key- value Pair.

4.2 Phase 2 - Node JOIN

Whenever a new node joins in the network, these steps are followed:

- New server sends the join request.
- Active server adds the new server to the finger table.
- Server already in the network updates its Finger Table.
- New node updates its finger table.
- Server 1 Sends some Key Pairs to Server 2(new server).

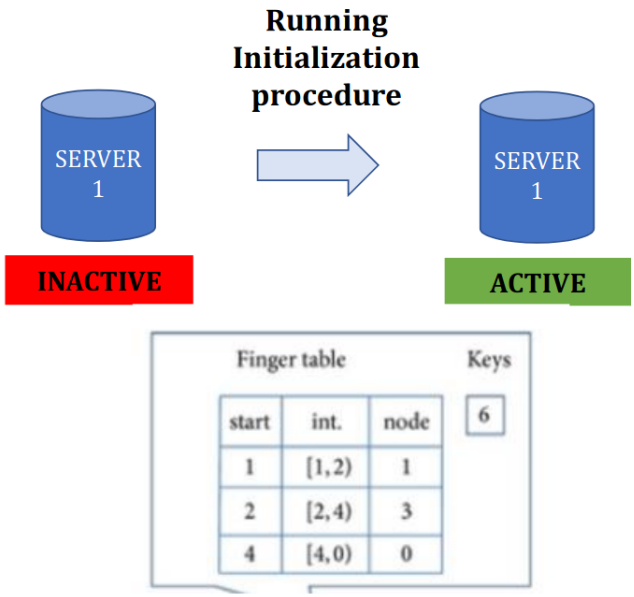

Figure 2: Initialization

4.2.1 STABILIZE

Every node runs stabilize() periodically to learn about newly joined nodes. Each time node n runs stabilize(), it asks its successor for the successor's predecessor p, and decides whether p should be n's successor instead. This would be the case if node p recently joined the system. In addition, stabilize() notifies node n's successor of n's existence, giving the successor the chance to change its predecessor to n. The successor does this only if it knows of no closer predecessor than n.

4.3 Phase 3: CLIENT URL QUERY

When a client requests for an URL for one of the servers in the network it checks for the key value pair in its own data. If the node finds, it will send the IP address back to the client.Otherwise it runs a lookup algorithm.

Steps for search query:

- Request with url to the load balancer.
- Load balancer will forward the request to one of the servers in chord.

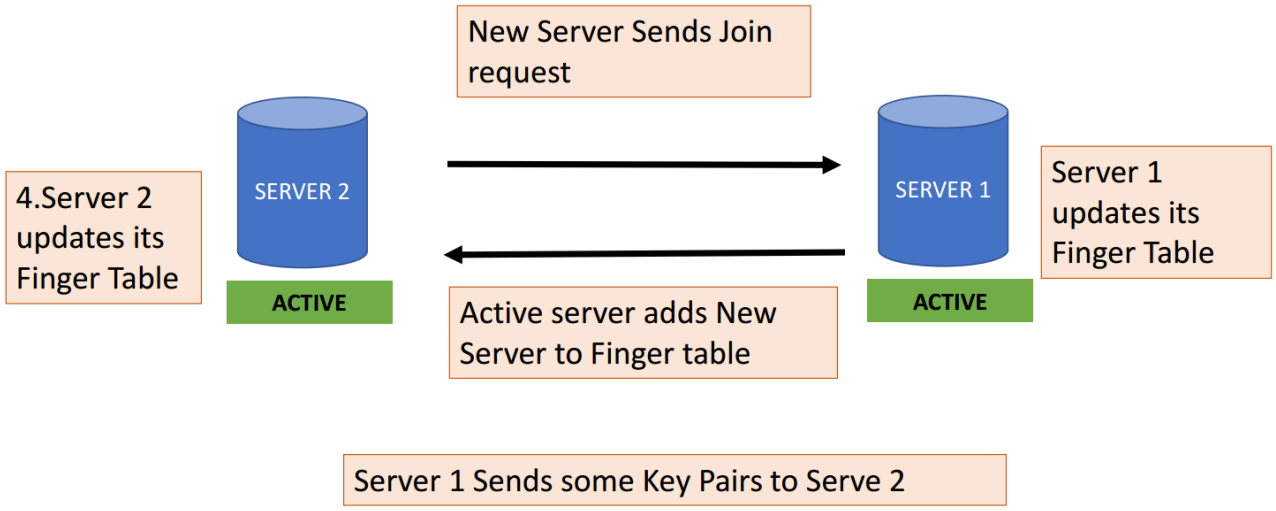

Figure 3: Node Join

- Node in the chord will calculate hashed value and search for successor of the url.
- If hashed valued matches with any successor then request will be forwarded to that node otherwise it will be forwarded to the last node in its finger table.
- Matched node will search the url in its own url-ip map and return it and if url is not present in that case it will return "Address not found".

Figure 5 shows an example in which node 8 performs a lookup for key 54. Node 8 invokes find successor for key 54 which eventually returns the successor of that key, node 56. The query visits every node on the circle between nodes 8 and 56. The result returns along the reverse of the path followed by the query. Figure 5 shows the path taken by a query from node 8 for key 54 For example Figure 6 shows the The finger table entries for node 8.The i th entry in the table at node n contains the identity of the first node s that succeeds n by at least

$$2^{i-1}$$

on the identifier circle.A finger table entry includes both the Chord identifier and the IP address and port number of the relevant node. The first finger

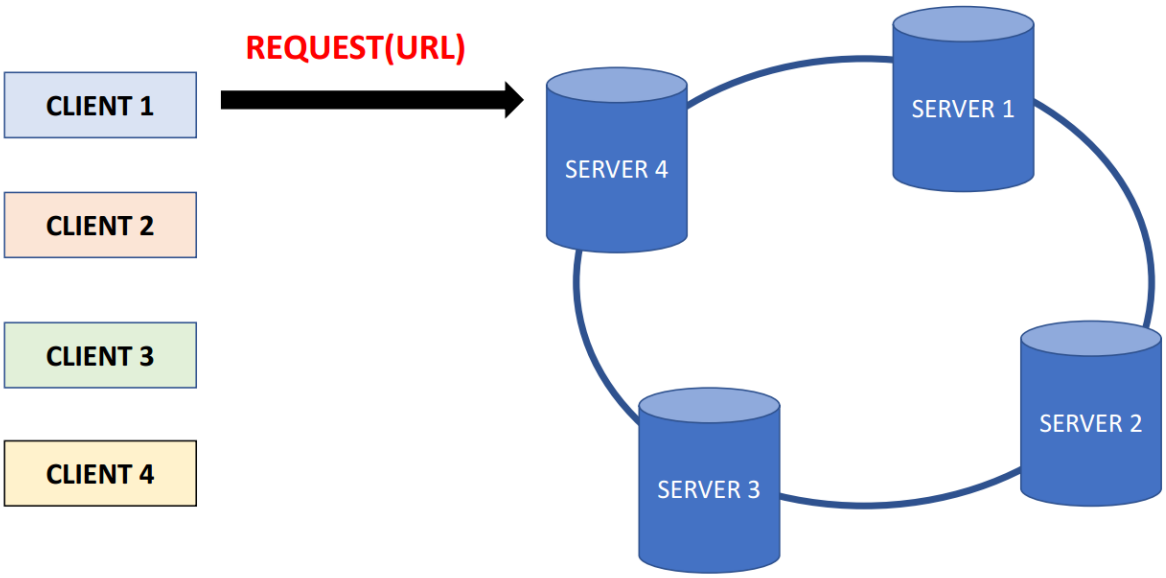

Figure 4: Client requests IP

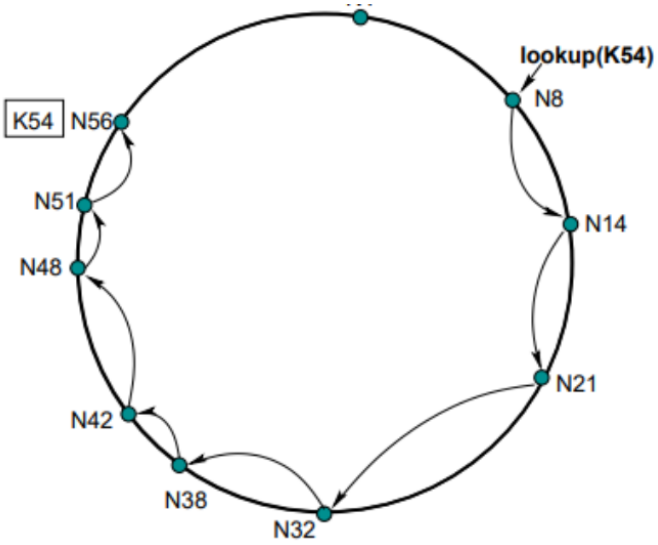

Figure 5:

of n is the immediate successor of n on the circle; for convenience we often refer to the first finger as the successor. Figure 7 shows the path taken by a

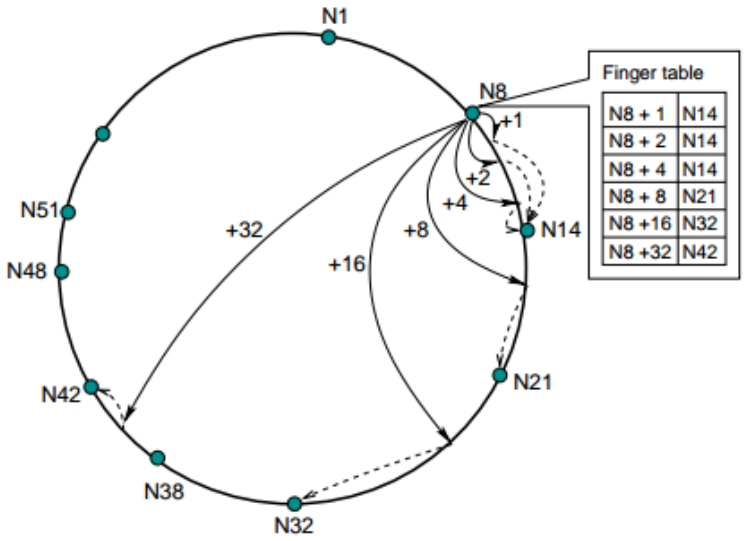

Figure 6: The finger table entries for node 8

query from node 8 for key 54.

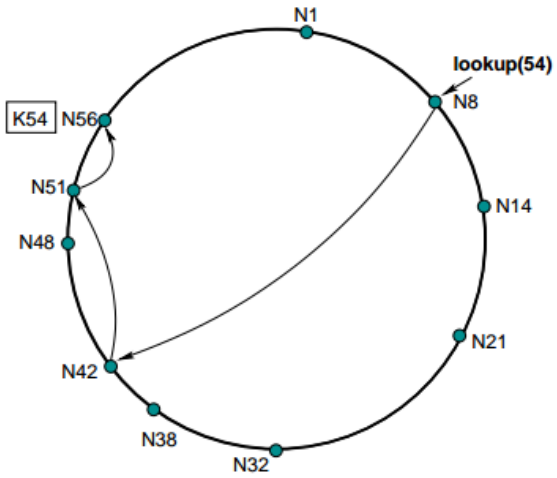

Figure 7: The path taken by a query from node 8 for key 54

5 References

[1] Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications Ion Stoicay, Robert Morrisz, David Liben-Nowellz, David R. Kargerz, M. Frans Kaashoekz, Frank Dabekz, Hari Balakrishnanz

[2] AJMANI, S., CLARKE, D., MOH, C.-H., AND RICHMAN, S. ConChord: Cooperative SDSI certificate storage and name resolution. In First International Workshop on Peer-to-Peer Systems (Cambridge, MA, Mar. 2002).