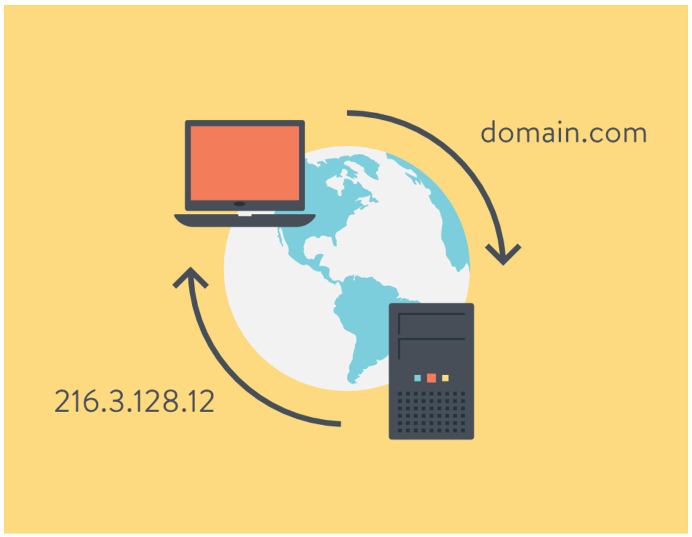
Distributed Systems (Spring 2022)

Domain Name System (DNS) using Distributed Hash Tables (DHT)

Samartha S M (2018101094)
Aditya Gupta (2018102010)
Amey Kudari (2018101046)

Domain Name System (DNS)

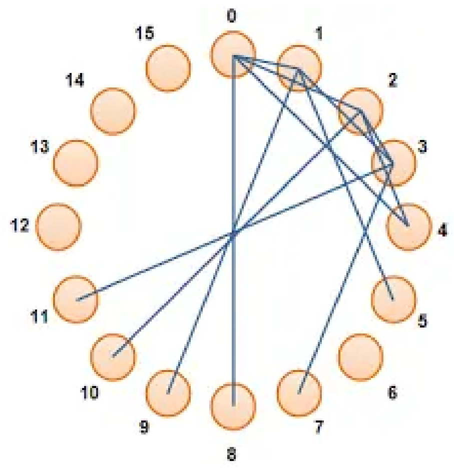
The **Domain Name System** (**DNS**) is the hierarchical and decentralized naming system used to identify computers reachable through the Internet or other Internet Protocol (IP) networks. The resource records contained in the DNS associate domain names with other forms of information. These are most commonly used to map human-friendly domain names to the numerical IP addresses computers need to locate services and devices using the underlying network protocols, but have been extended over time to perform many other functions as well. The Domain Name System has been an essential component of the functionality of the Internet since 1985.



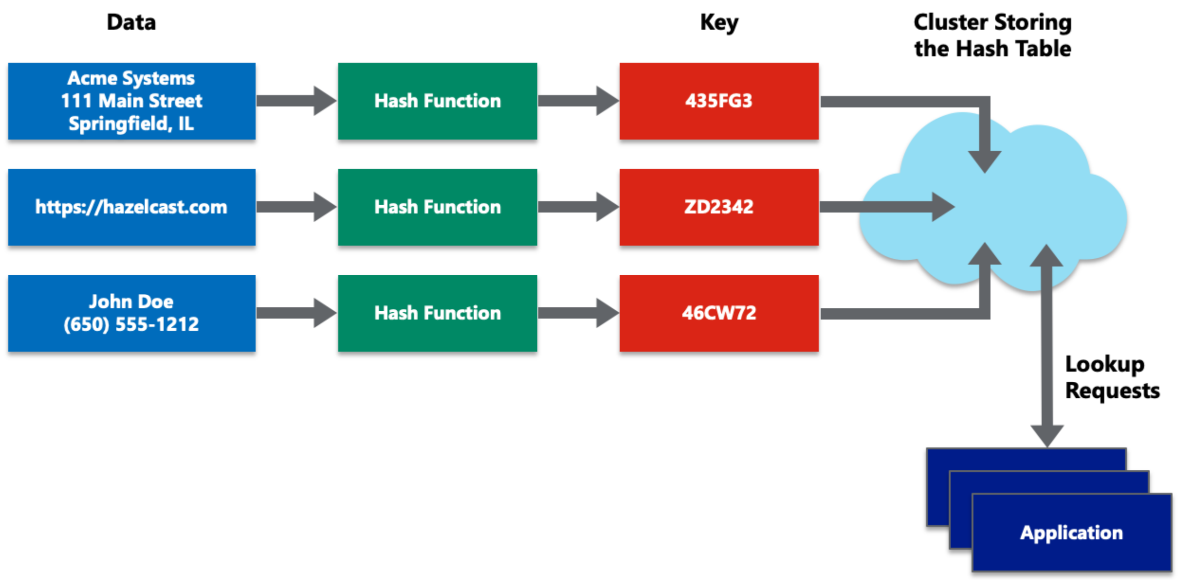
An often-used analogy to explain the Domain Name System is that it serves as the phone book for the Internet by translating human-friendly computer hostnames into IP addresses. For example, the domain name www.example.com translates to the addresses *93.184.216.34* (IPv4) and *2606:2800:220:1:248:1893:25c8:1946* (IPv6). The DNS can be quickly and transparently updated, allowing a service's location on the network to change without affecting the end users, who continue to use the same hostname. Users take advantage of this when they use meaningful Uniform Resource Locators (URLs) and e-mail addresses without having to know how the computer actually locates the services.

Distributed Hash Table (DHT)

A **distributed hash table** (**DHT**) is a distributed system that provides a lookup service similar to a hash table: key-value pairs are stored in a DHT, and any participating node can efficiently retrieve the value associated with a given key. The main advantage of a DHT is that nodes can be added or removed with minimum work around re-distributing keys. *Keys* are unique identifiers which map to particular *values*, which in turn can be anything from addresses, to documents, to arbitrary data. 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.

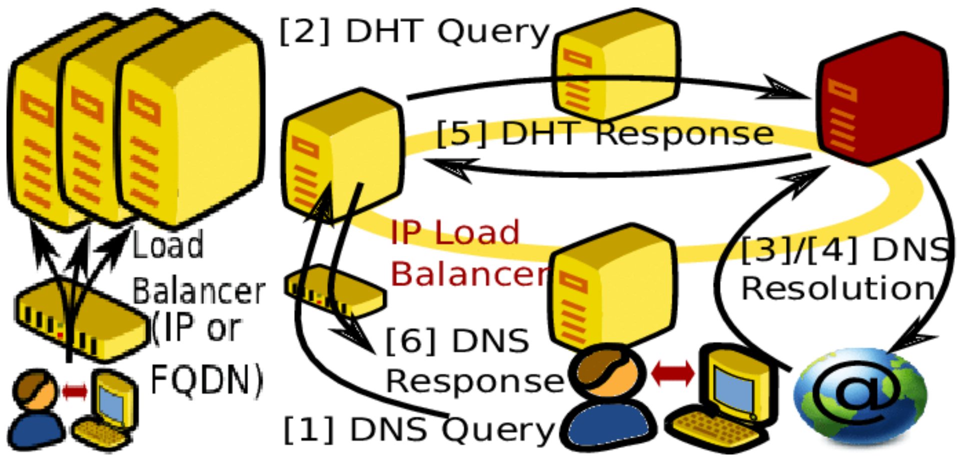


DHTs form an infrastructure that can be used to build more complex services, such as anycast, cooperative web caching, distributed file systems, domain name services, instant messaging, multicast, and also peer-to-peer file sharing and content distribution systems. Notable distributed networks that use DHTs include BitTorrent's distributed tracker, the Kad network, the Storm botnet, the Tox instant messenger, Freenet, the YaCy search engine, and the InterPlanetary File System.



Domain Name System (DNS) using Distributed Hash Table (DHT)

The current Domain Name System (DNS) follows a hierarchical tree structure. Several recent efforts proposed to re-implement DNS as a peer-to-peer network with a flat structure that uses Distributed Hash Table (DHT) to improve the system availability.



In this project, we have implemented a Domain Name System (DNS) using Kademlia Distributed Hash Table (DHT). The mappings between human friendly domain names and corresponding IP addresses are stored in peer storage (DHT) to improve the efficiency. The code is implemented in python.

How to run?

The DNS server is kept running on localhost (127.0.0.1) on port 53 as port 53 is default port for DNS protocol with TCP and UDP. When the DNS server is started, it starts bootstrapping the nodes and then imports the data (domain - IP map) into the nodes.

To start the DNS server, run the following command

```
$ sudo python3 dns_dht.py --num_nodes <num_dht_nodes> --data_path <domains-ip_path>
```

To query for a domain, run the following command

```
$ dig <domain_name> @127.0.0.1
```

As the DNS server is running on localhost (127.0.0.1) and port 53, any DNS queries sent to 127.0.0.1 will be redirected to port 53 by default on 127.0.0.1 host. The DNS response is built and sent back to the client address after the IP address is found from the DHT node.

Example Output

Google.com (142.250.205.238)

```
; <<>> DiG 9.16.1-Ubuntu <<>> google.com @127.0.0.1
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23577
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             400     IN      A       142.250.205.238

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sat Apr 30 23:14:51 IST 2022
;; MSG SIZE  rcvd: 44
```

Pros of using DHT for DNS

- The main advantage of a DHT is that nodes can be added or removed with minimum work around re-distributing keys.
- Scalable, self-organizing
- Lack of hierarchy
    - Hard to attack a set of domain names
    - Handle flash-crowd effects well
    - No central points of failure
    - Network routes around failures
- DNS servers --- mostly homogeneous
- Can design backward-compatible DHT-based DNS

Cons of using DHT for DNS

- Network outages are poorly handled
- Certain functionality is lost
- Solving the wrong problem
- Performance improvements are not due to DHTs
    - But rather to heavy replication