L9 Naming, DNS and Chord

Access Point

Access point: way of contacting resource in a networked system

Address

Name of an access point, which can change over time

Names

- Flat

  "Opaque" identifier, not indication as to location

  Eg: phone numbers, Ethernet

- Structured

  Location encoded in the address

  IP address

Naming Hierarchy for Scale

1. **Host name**

   Domain: register for each top-level domain (americauniversity.edu)

   Host name: local administrator assigns to each host (www)

   The host name represents the network or system used to deliver a user to a certain address or location. The domain name represents the site or project that the user is accessing.

   To make things even more confusing, you can refer to the full address in its entirety as a hostname. For example, with www.bleepingcomputer.com:

   - Bleepingcomputer.com is the domain name.
   - www is the hostname
   - www.bleepingcomputer.com is the hostname as well!!!

Full Address	Hostname	Domain Name/Subdomain	TLD
www.bleepingcomputer.com	www	bleepingcomputer.com	com
bleepingcomputer.com	bleepingcomputer.com	bleepingcomputer.com	com
www.google.net	www	google.net	net
my.www.bleepingcomputer.com	my	www.bleepingcomputer.com	com

2. **IP Address**

Prefix: : ICANN, regional Internet registries, and ISPs

Hosts: static configuration, or dynamic using DHCP

3. **MAC address 58:B0:35:F2:3C:D9**
    - OIDs : assigned to vendors by the IEEE (58:B0:35)
    - Adapters : assigned by the vendor from its block (F2:3C:D9)

Mapping between identifiers

1. **DNS**

    Convert between host name and IP address

2. **ARP**

    IP address => MAC address

    To enable communication with the local area network

3. **DHCP**
    - Automatic host boot-up process
    - Assign a unique IP address given a MAC address
    - Tell other stuff about then local area network

Hierarchical naming and DNS

1. DNS host name

- Mnemonic name
- Variable length(alphabet)
- Provides little info about location
- Host name aliasing, which points to canonical hostname
- Email look up domain's mail server by fomain name

2. IP address

- Numerical address appreciated by routers
- Fixed length
- Hierarchical address space, related to host location

Original Design of DNS

Networked computers used local files to map hostnames to IP addresses. On Unix systems this file was named `/etc/hosts` or “the hosts file”

Each line includes IP address and DNS name

SRI keeps the master copy and every host download it regularly

But a single server doesn't scale, Traffic implosion(regular lookups and updates), Single point of failure, We need a distributed, hierarchical collection of servers == >**DNS!!**

Goals of DNS

Let it be a wide-area distributed database.

Scalability, robustness, global scope(host name means the same everywhere), distributed updates and queries, Good performance

But strong consistency is not necessary

DNS

Hierarchical namespace divided into zones, which are distributed over a collection of DNS servers

- Root servers (13 worldwide)

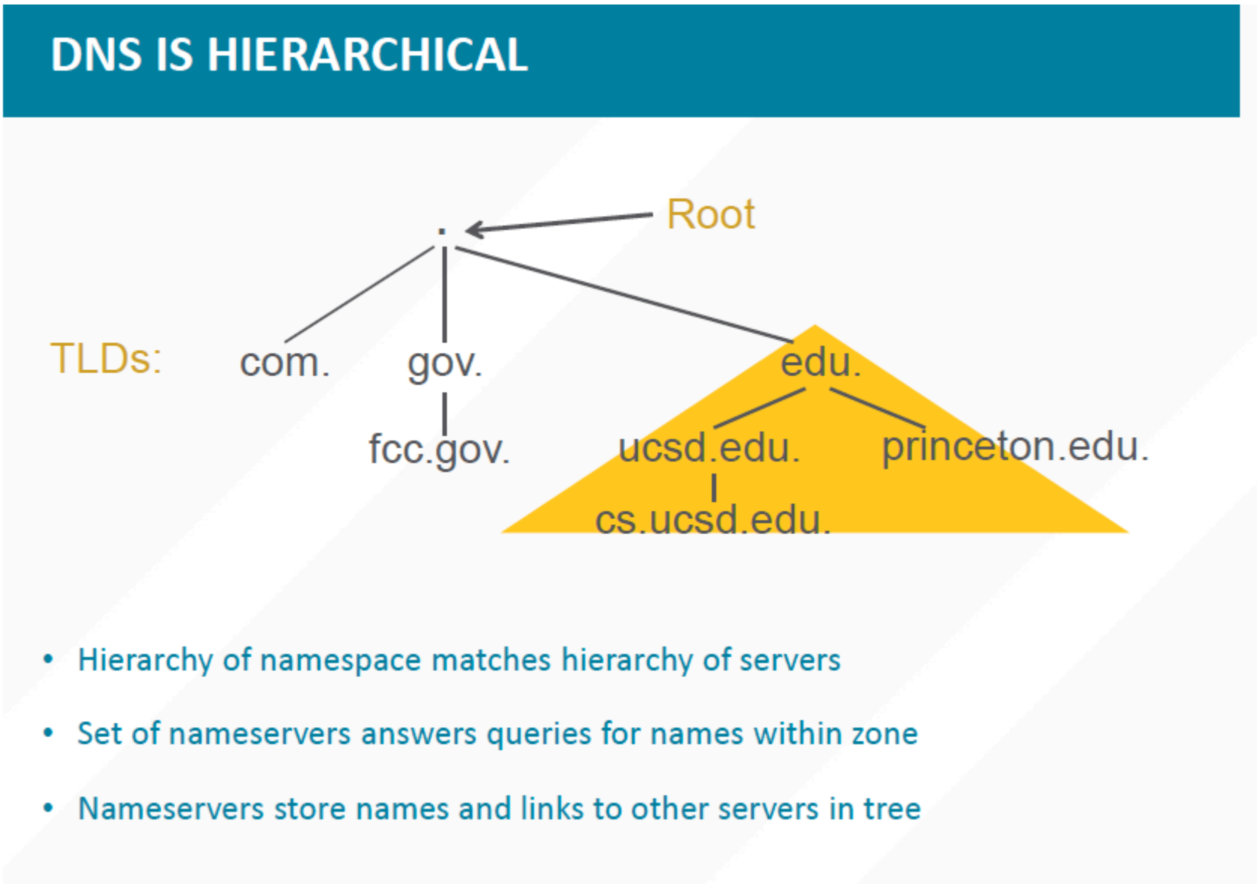
  Each server is really a cluster of servers (some geographically distributed), replicated via IP anycast
- Top-level domain(TLD) servers

  Responsible for com, org, net, edu , etc , and all top level country domains: uk , fr , ca, jp
- Authoritative DNS servers

  An organization’s DNS servers, providing authoritative information for that organization May be maintained by organization itself, or ISP

Performing the translations:

- Local DNS servers located near clients
- Resolver software running on clients



Local name servers

Each ISP(company, or university) has one, also called default or caching name server

When host makes DNS query, query is sent to its DNS server

Acts like a proxy, forwards query into hierarchy

DNS Resource Records

DNS is a distributed database storing resource records

Resource Records

(name , type, value, time to live)

Type = A (address)

- **name** = hostname
- **value** is IP address

Type = CNAME

- **name** = alias for some "canonical" (real) name
- **value** is canonical name

Type = NS (name server)

- **name** = domain (e.g. princeton.edu)
- **value** is hostname of authoritative name server for this domain

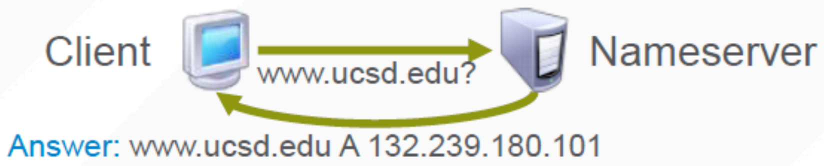
Type = MX (mail exchange)

- **name** = domain
- **value** is name of mail server for that domain

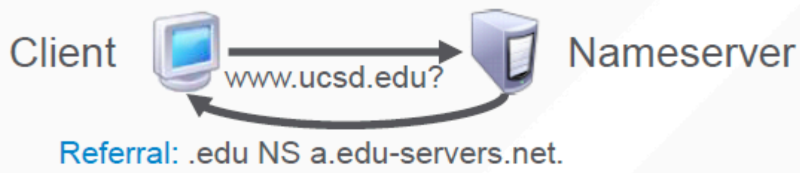
DNS queries

DNS IN OPERATION

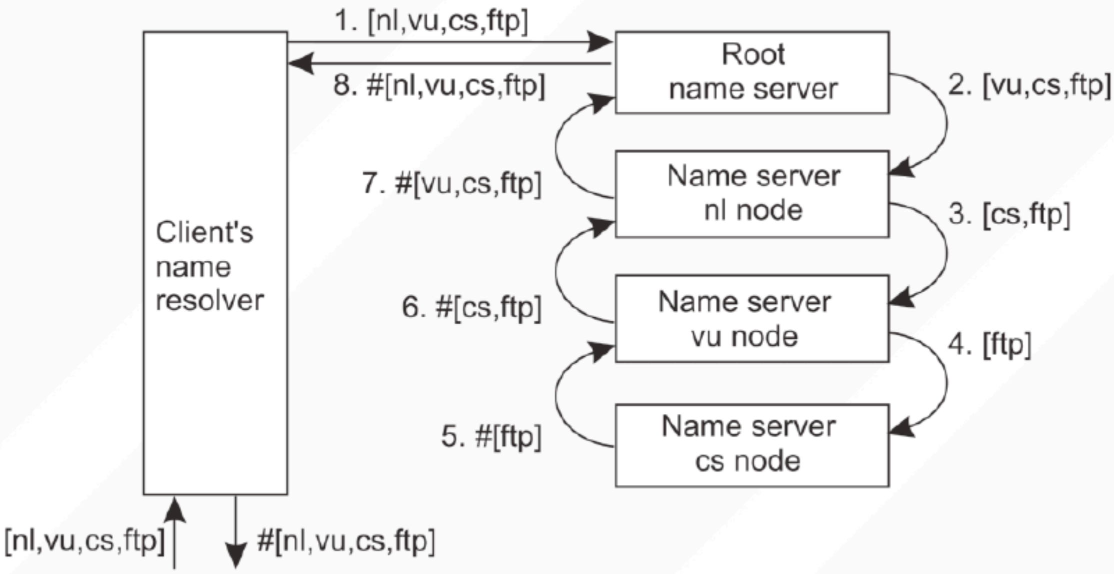
- Most queries and responses are UDP datagrams
  - Two types of queries:
- ***Recursive***: Nameserver responds with answer or error



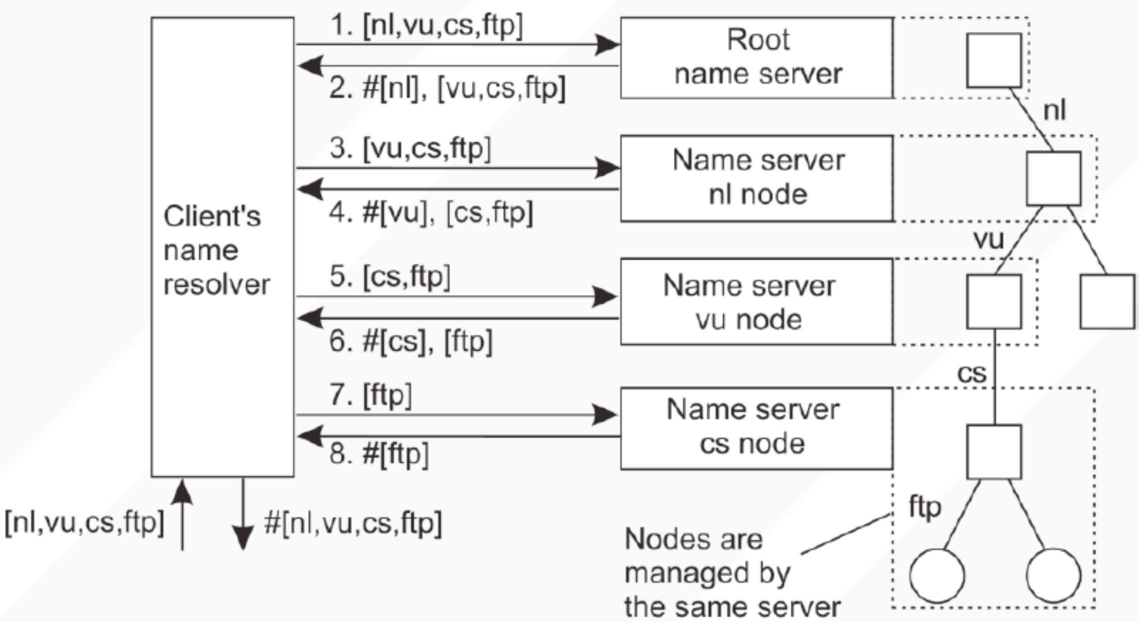
- ***Iterative***: Nameserver may respond with a referral



RECURSIVE DNS LOOKUP



ITERATIVE DNS LOOKUP



Recursive name resolution of [*nl*, *vu*, *cs*,*ftp*]

Server for node	Should resolve	Looks up	Passes to child	Receives and caches	Returns to requester
cs	[*ftp*]	#[*ftp*]	—	—	#[*ftp*]
vu	[*cs*, *ftp*]	#[*cs*]	[*ftp*]	#[*ftp*]	#[*cs*] #[*cs*, *ftp*]
nl	[*vu*, *cs*, *ftp*]	#[*vu*]	[*cs*, *ftp*]	#[*cs*] #[*cs*, *ftp*]	#[*vu*] #[*vu*, *cs*] #[*vu*, *cs*, *ftp*]
root	[*nl*, *vu*, *cs*, *ftp*]	#[*nl*]	[*vu*, *cs*, *ftp*]	#[*vu*] #[*vu*, *cs*] #[*vu*, *cs*, *ftp*]	#[*nl*] #[*nl*, *vu*] #[*nl*, *vu*, *cs*] #[*nl*, *vu*, *cs*, *ftp*]

DNS caching

Caching reduce time overhead

- The top level servers very rarely change
- Popular sites visited often

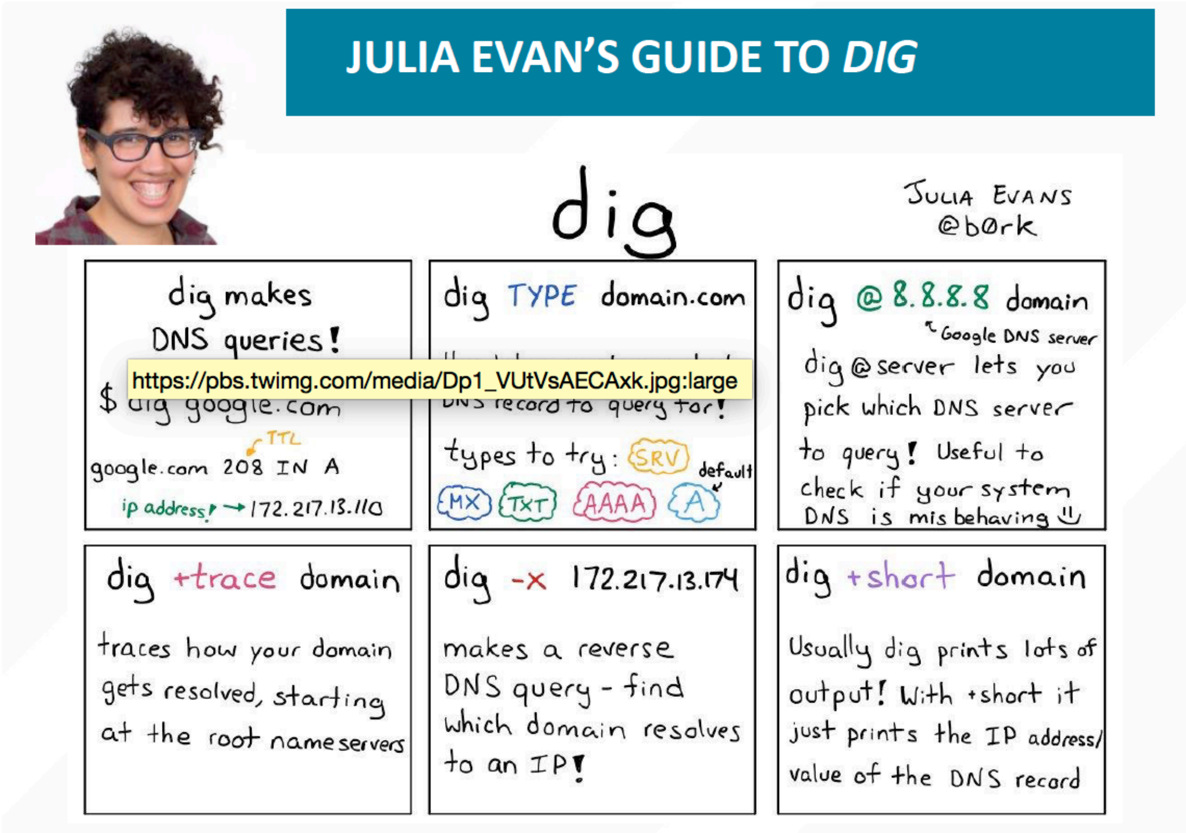
Local DNS server often has the information cached

They cache responses to queries

Responses have the field time-to-live(TTL)

Server deletes cached entry after TTL expires

Tool DIG

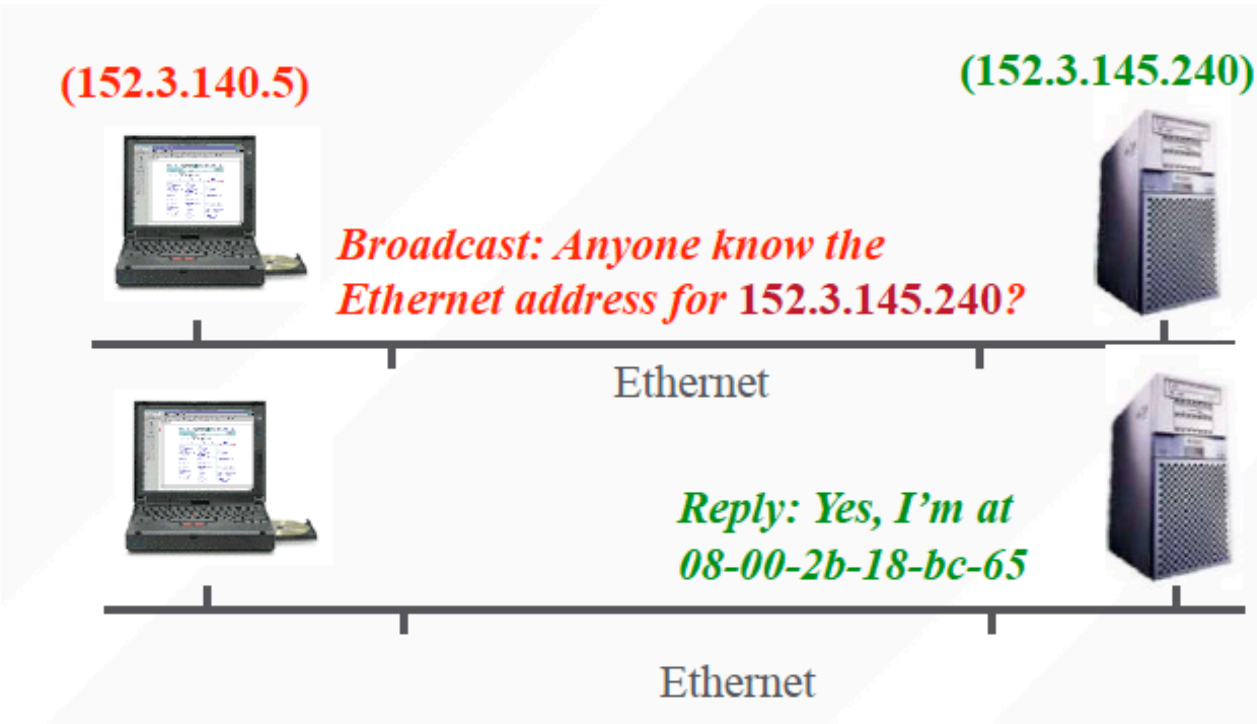


ARP

Translates IP address to link-level address(eg: Ethernet addr)

Broadcast request over network for IP => link-level mapping

Maintain cache



DHCP

We use DHCP to complete Auto-configuration

- Host doesn't have an IP address yet
- Host doesn't know who to ask for an IP address,So, host doesn't know what destination address to use

We install a DHCP server on LAN to allocate new host IP address

Broadcast-based LAN protocol algorithm

• Host broadcasts "DHCP discover" on LAN (e.g. Ethernet broadcast) • DHCP server responds with "DHCP offer" message • Host requests IP address: "DHCP request" message • DHCP server sends address: "DHCP ack" message w/IP address

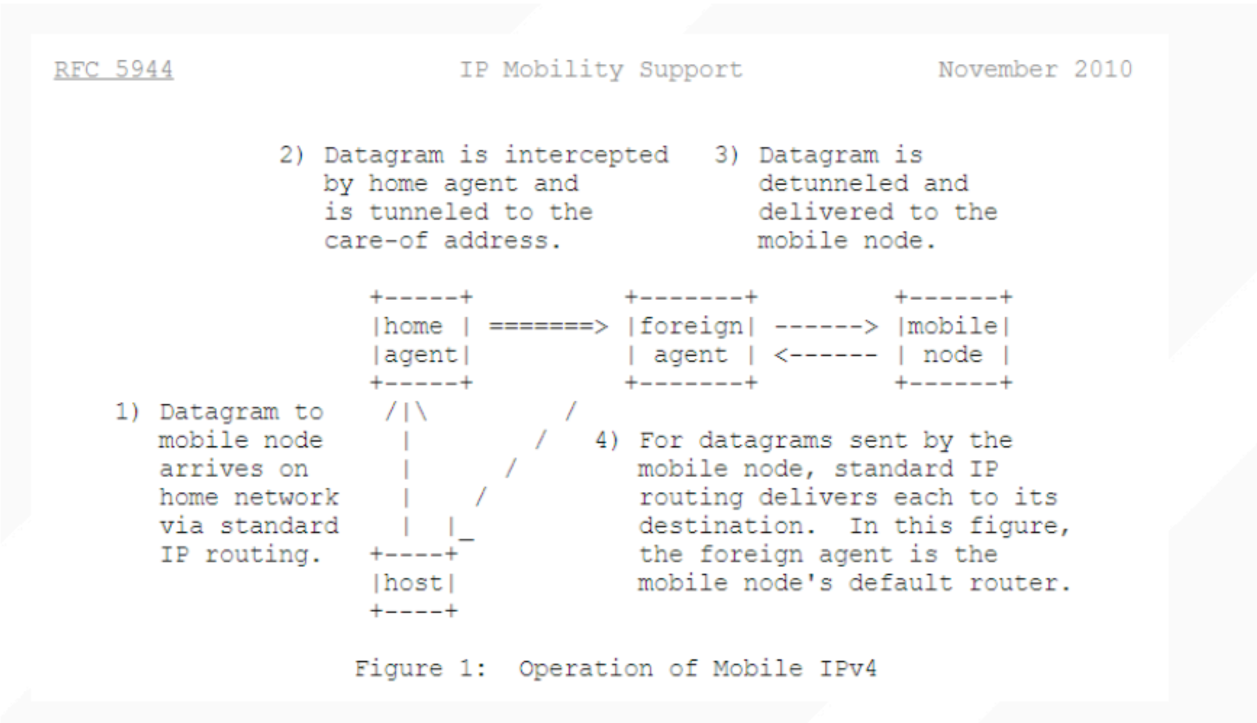
We use **timeout** in case host goes away

Address is a lease not a grant!

IP Mobility

Keep user's IP address the same even if the user's geographical address keeps changing

Home/Foreign Agent Forwarding



```
RFC 5944                 IP Mobility Support              November 2010

                 2) Datagram is intercepted   3) Datagram is
                    by home agent and            detunneled and
                    is tunneled to the           delivered to the
                    care-of address.             mobile node.

                      +-----+           +-------+         +------+
                      |home | =======>  |foreign| ------> |mobile|
                      |agent|           | agent | <------ | node |
                      +-----+           +-------+         +------+
       1) Datagram to   /|\         /
          mobile node    |        /   4) For datagrams sent by the
          arrives on     |      /        mobile node, standard IP
          home network   |    /          routing delivers each to its
          via standard   |   |_          destination.  In this figure,
          IP routing.  +----+            the foreign agent is the
                       |host|            mobile node's default router.
                       +----+

                   Figure 1:  Operation of Mobile IPv4
```

Flat Naming and P2P newtorks

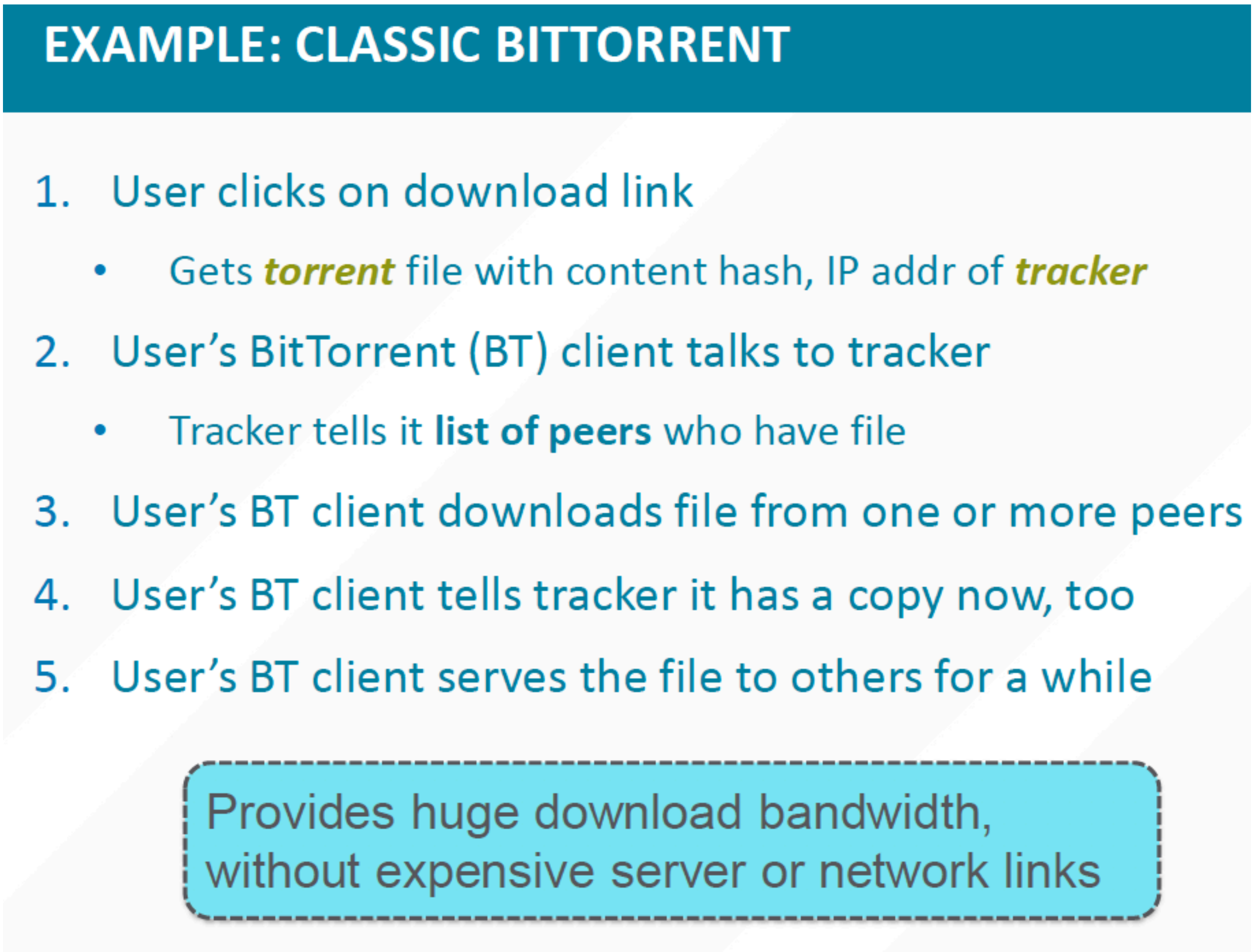
A distributed system architecture:

- No centralized control

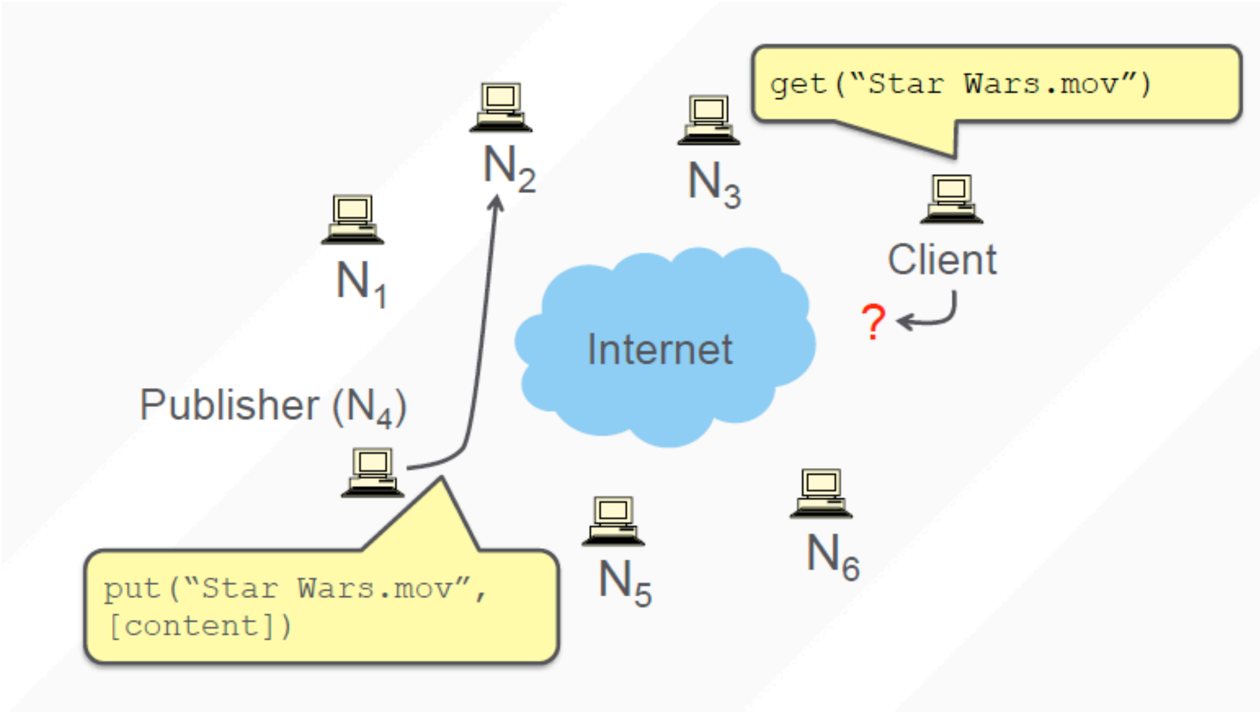
- Nodes are roughly symmetric in function
- Large number of unreliable nodes

P2P Adoption

- Client-to-client file sharing
- Digital currency
- Video telephony

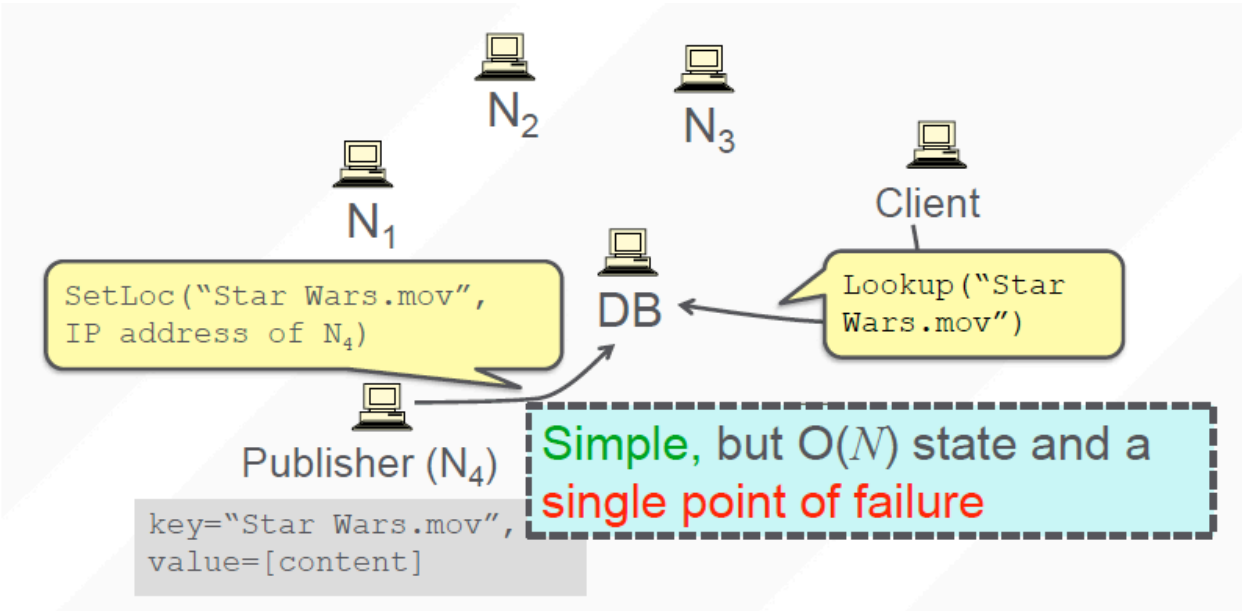


Flat name lookup problem

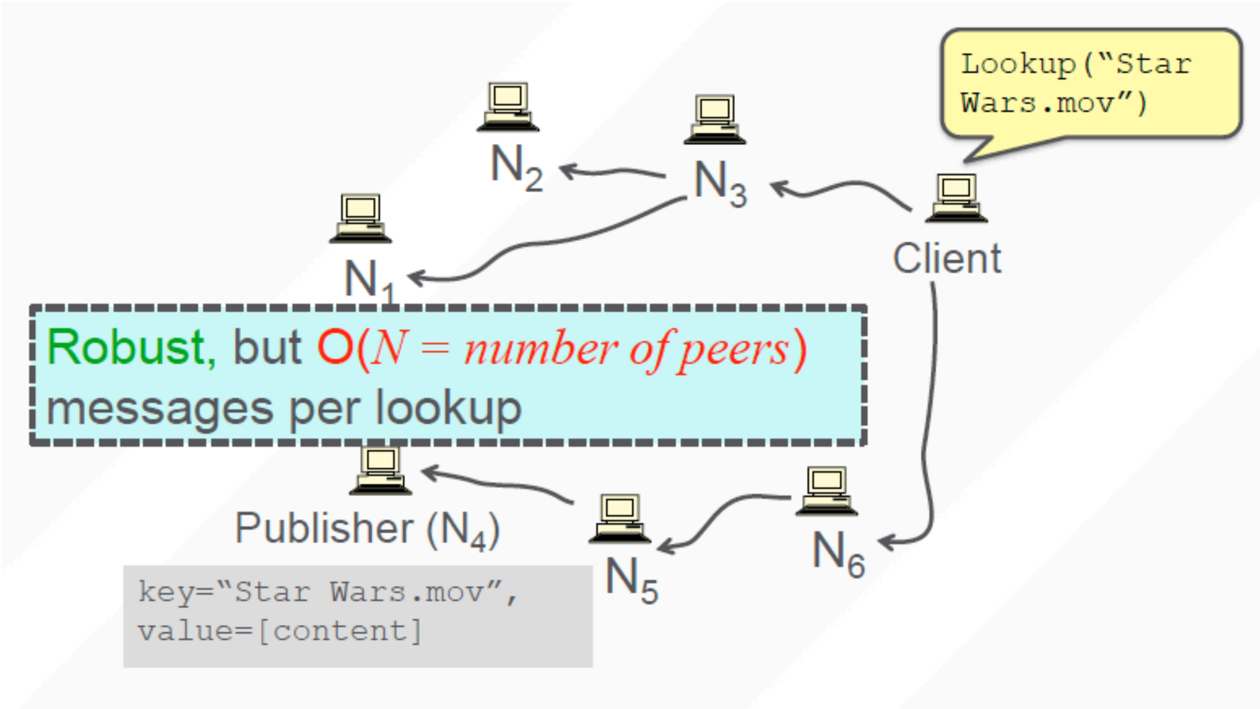


How to find N2?

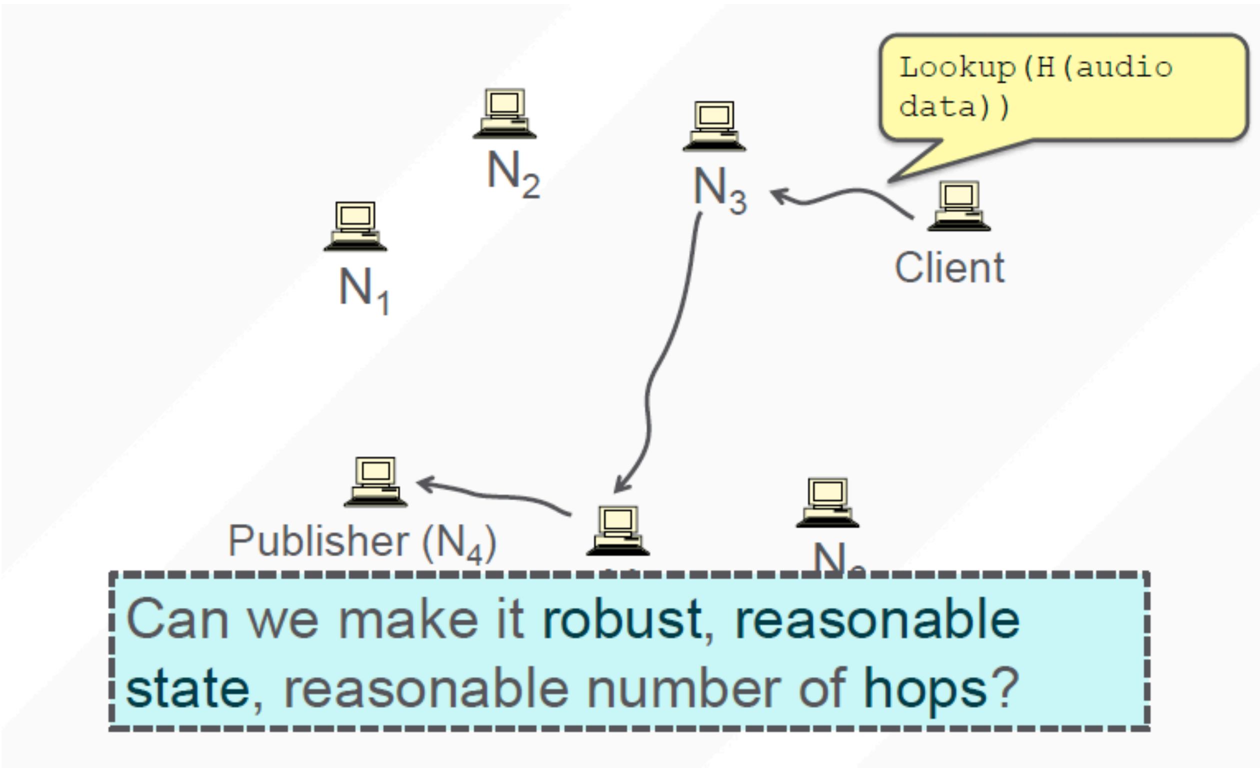
Solution1: centralized lookup



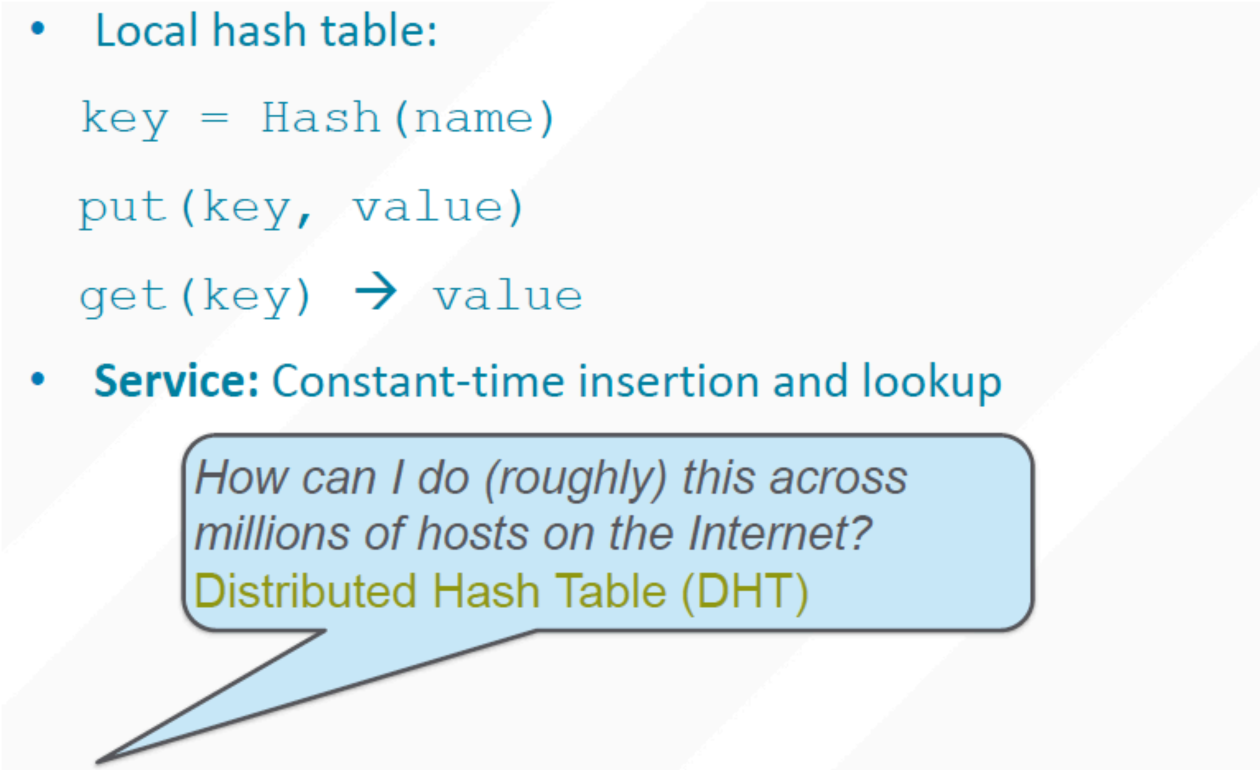
Solution2: Flooded queries

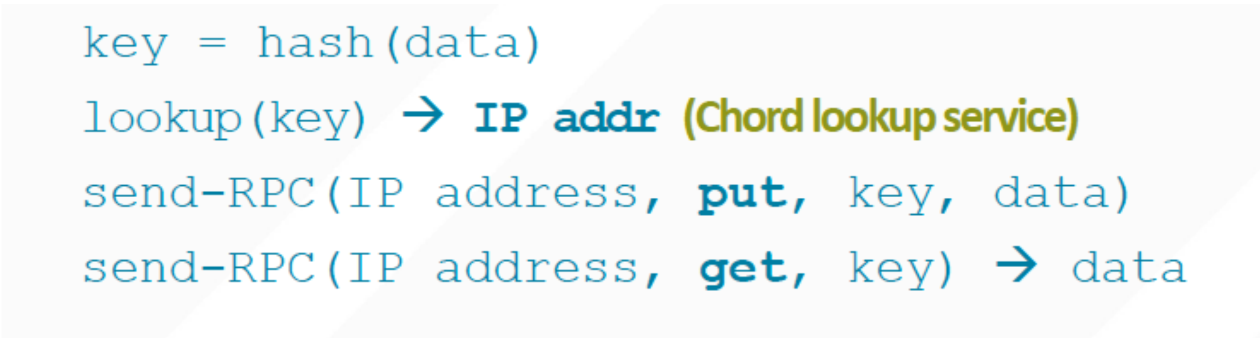


Solution3: Routed DHT queries(Chord)



DHT





We also need to partition the data in truly large-scale distributed systems

Chord

- Interface: lookup(key) --> IP address
- Efficient: O(log N) messages per lookup N is the total number of servers
- Scalable: O(log N) state per node
- Robust: survives massive failures

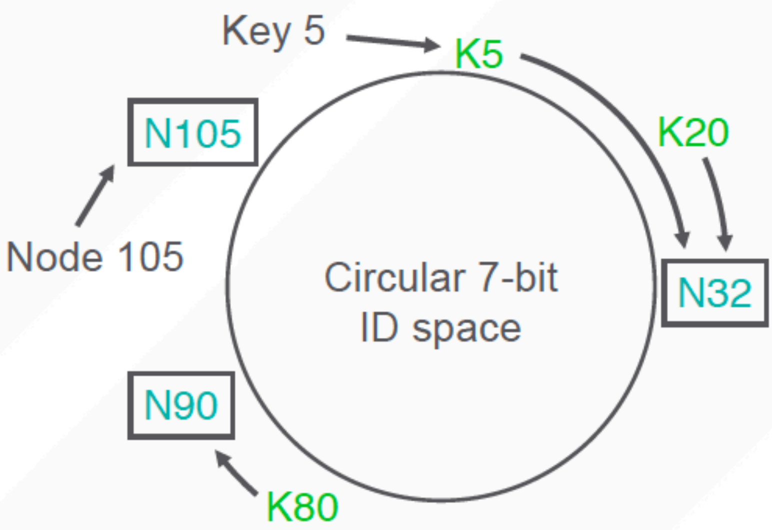
Identifiers

Key identifier = SHA 1(key)

Node identifier = SHA 1(IP address)

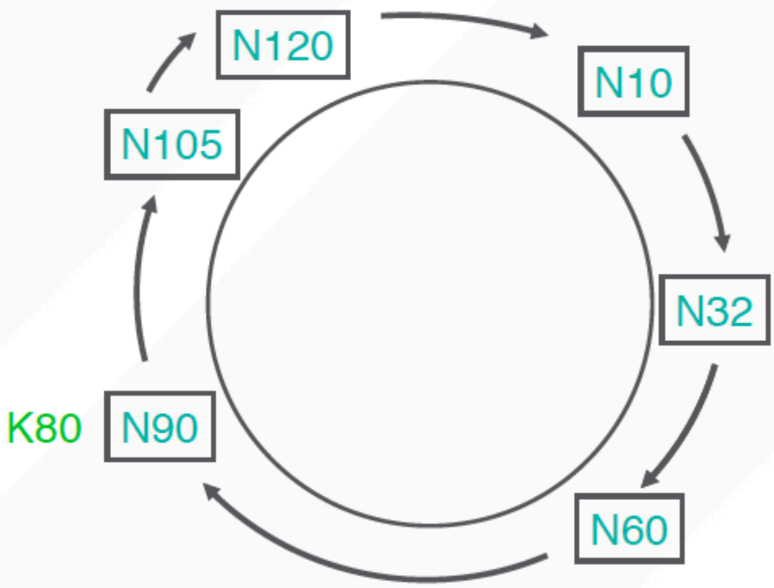
Map key IDs to node IDs

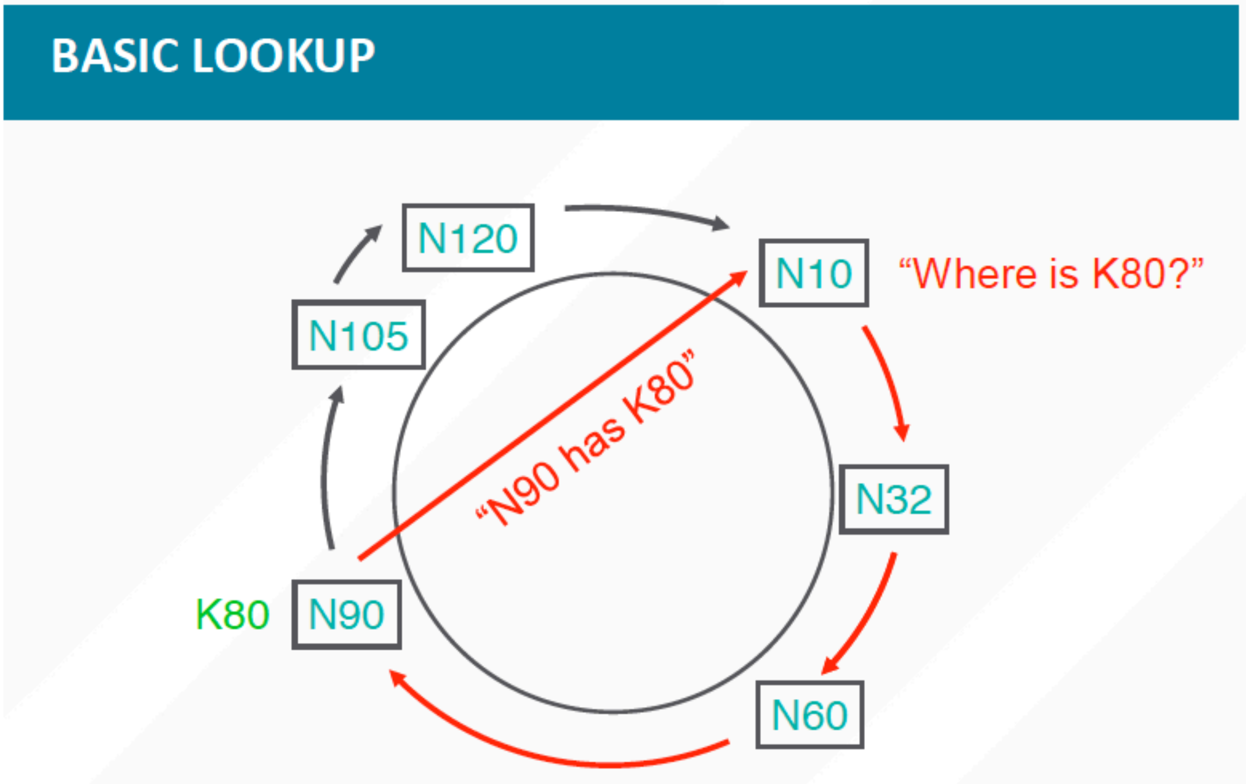
CONSISTENT HASHING [KARGER '97]



Key is stored at its successor: node with next-higher ID

CHORD: SUCCESSOR POINTERS





Psuedo Code

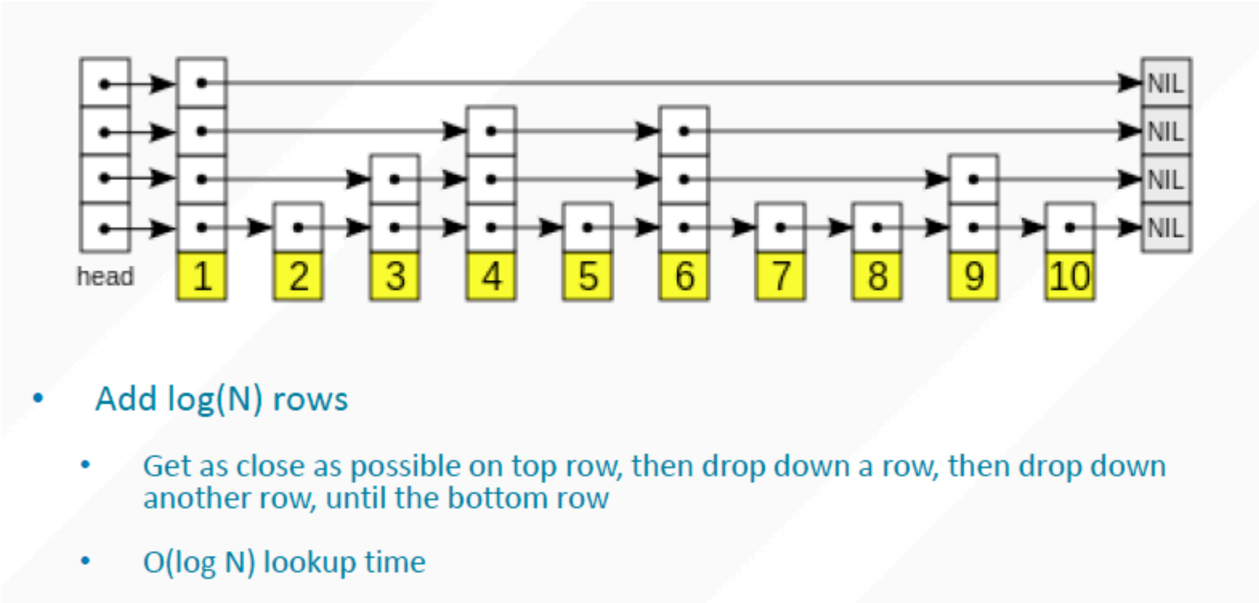
```
Lookup(key-id)
   succ ← my successor
   if my-id < succ < key-id //next hop
     call Lookup(key-id) on succ
   else                 //done
    return succ
```

Problem: Forwarding through successor is slow

• Data structure is a linked list: O(n) • Idea: Can we make it more like a binary search? • Need to be able to halve distance at each step

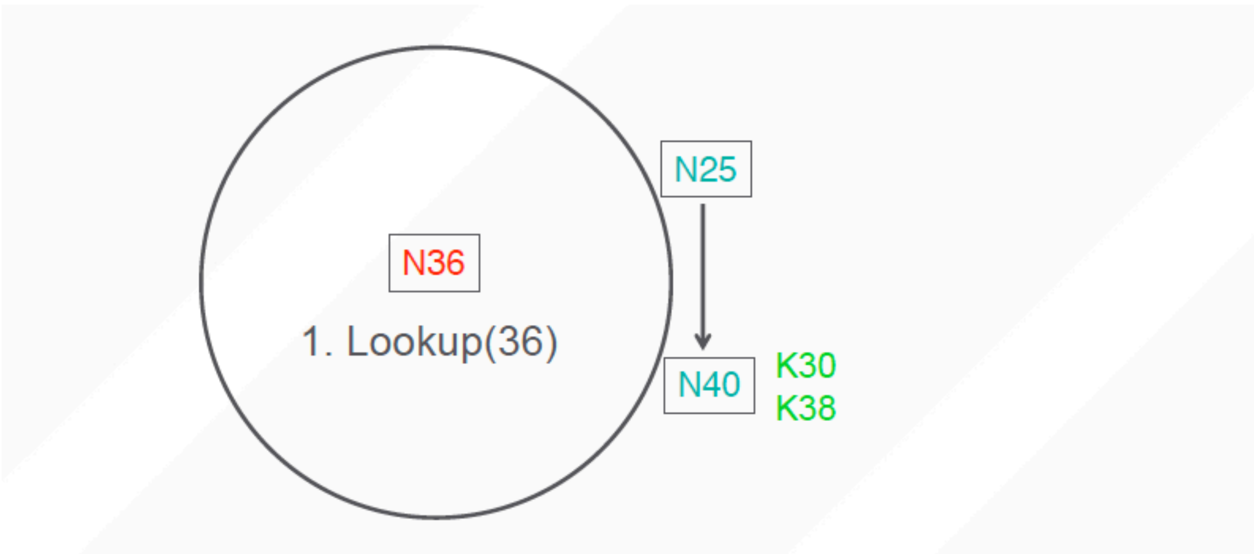
Improve Performance

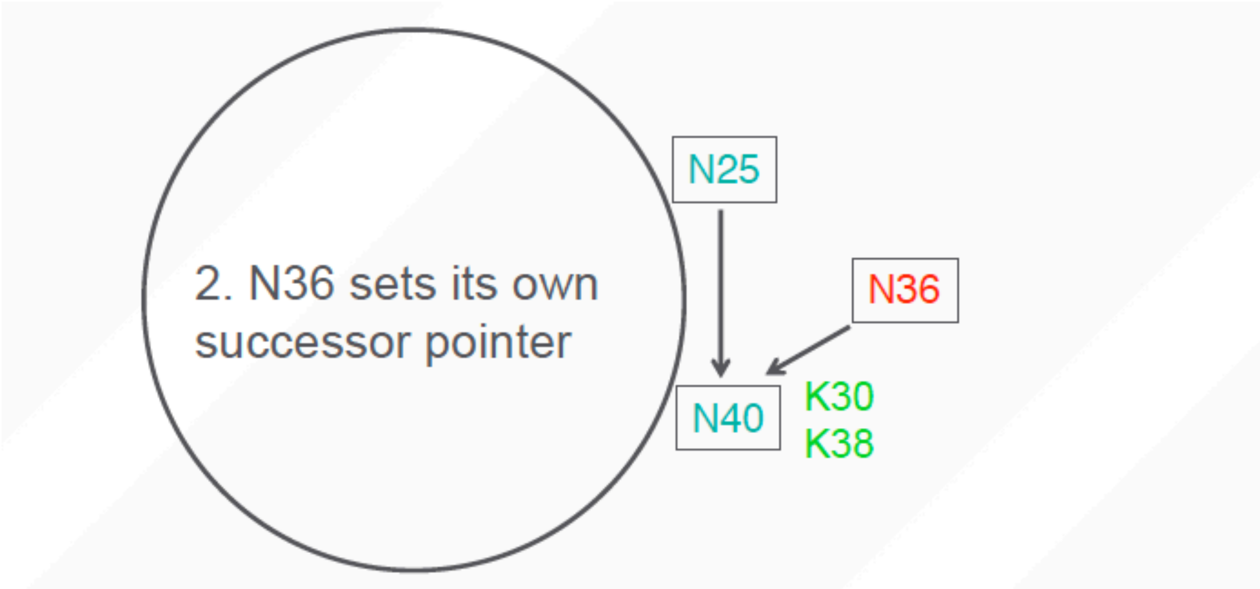
Convert linked list to skip list

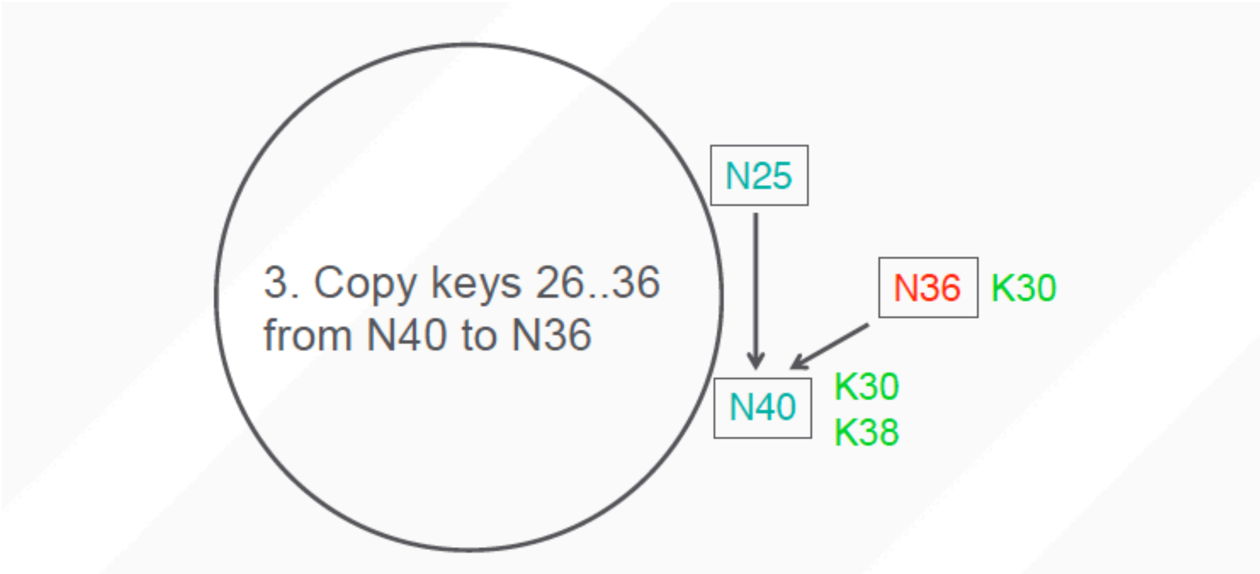


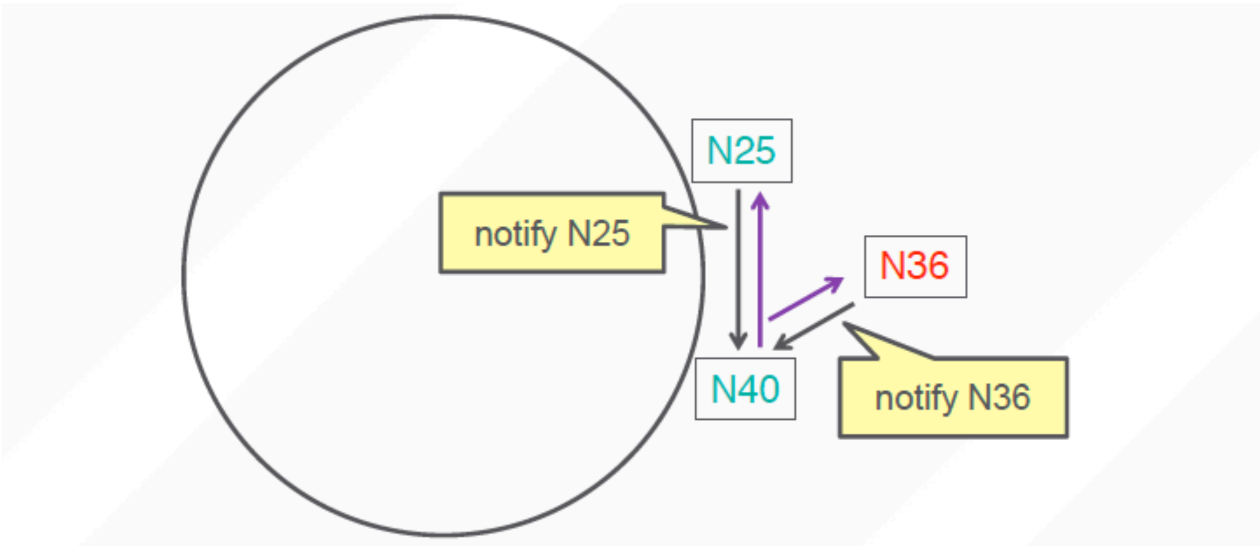
- For a million nodes, it's 20 hops
- If each hop takes 50 milliseconds, lookups take a second
- If each hop has 10% chance of failure, it's a couple of timeouts
- So in practice log(n) is better than O(n) but not great

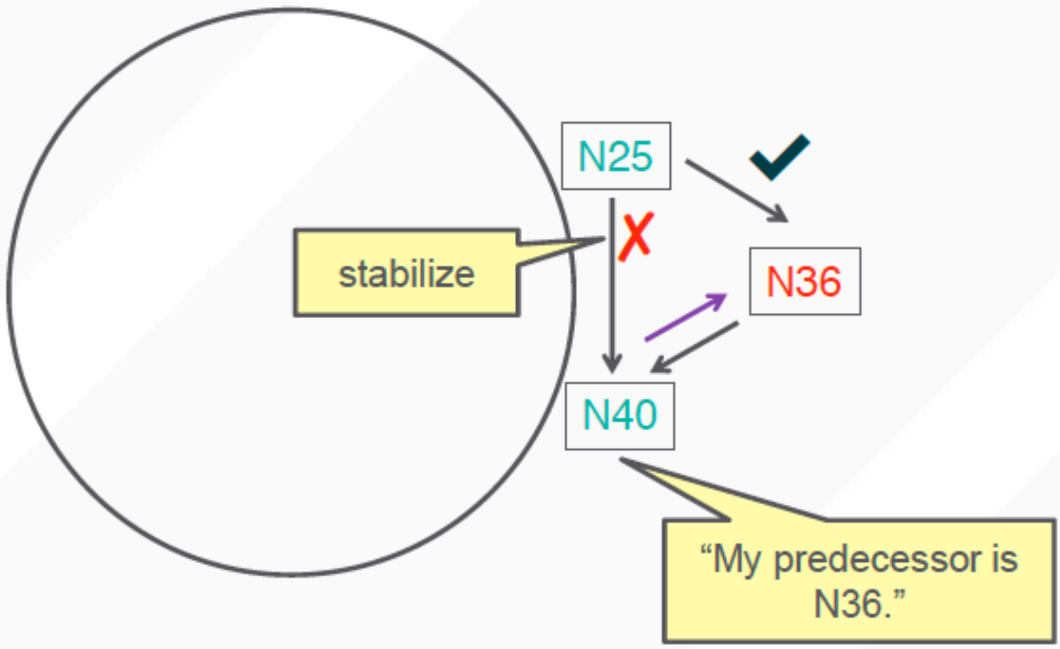
Join

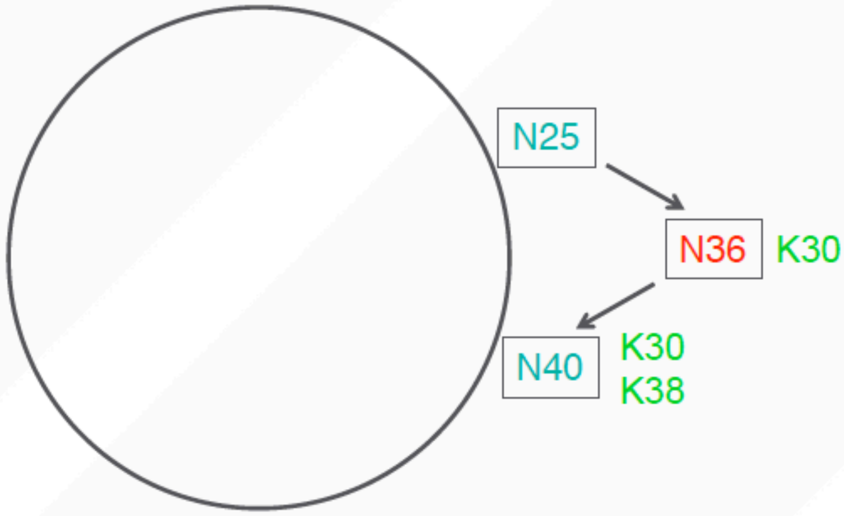



2. N36 sets its own successor pointer
N25
N36
N40
K30
K38


3. Copy keys 26..36 from N40 to N36
N25
N36
K30
N40
K30
K38


N25
notify N25
N36
N40
notify N36





- Predecessor pointer allows link to new node
- Update finger pointers in the background
- Correct successors produce correct lookups

WHAT DHTS GOT RIGHT

- **Consistent hashing**
  - Elegant way to divide a workload across machines
  - Very useful in clusters: actively used today in Amazon Dynamo and other systems
- **Replication** for high availability, efficient recovery after node failure
- **Incremental scalability:** "add nodes, capacity increases"
- **Self-management:** minimal configuration
- **Unique trait:** no single server to shut down/monitor