Naming

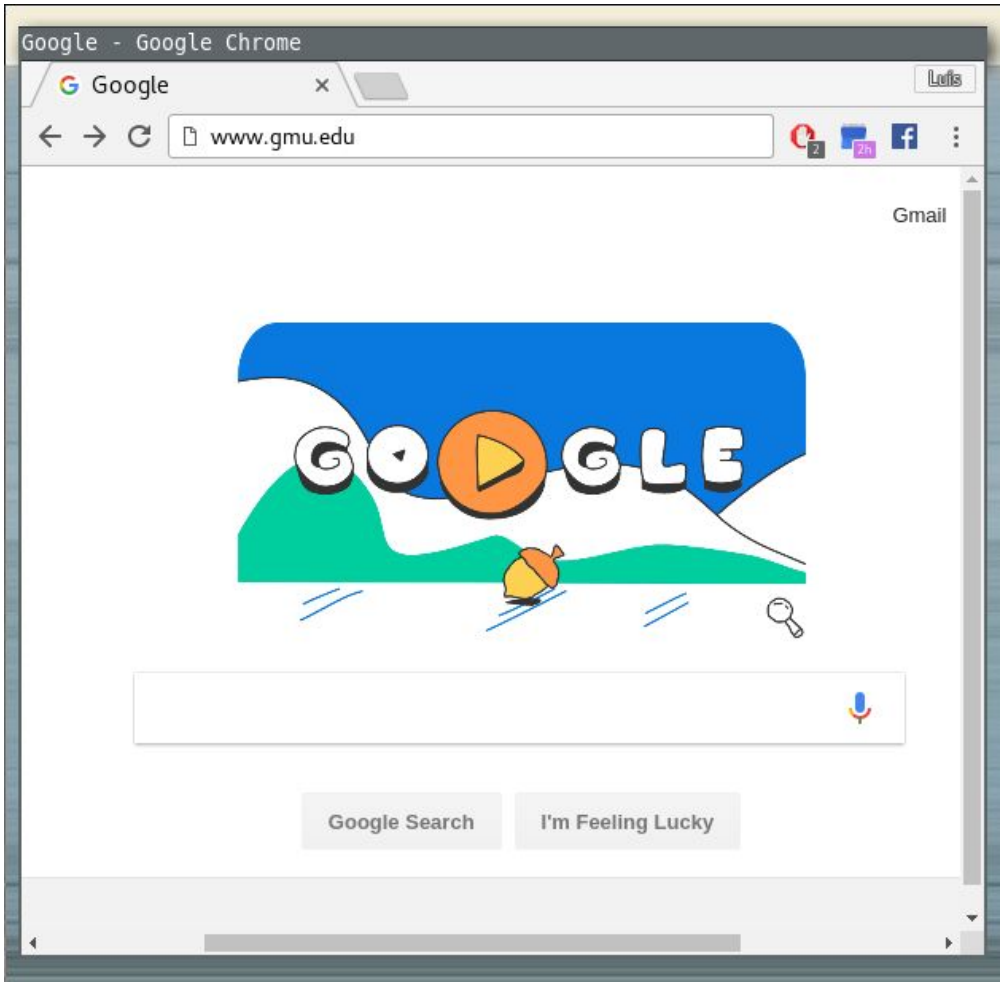
CS 475, Spring 2018
Concurrent & Distributed Systems

Slides by Luís Pina (lpina2@gmu.edu)

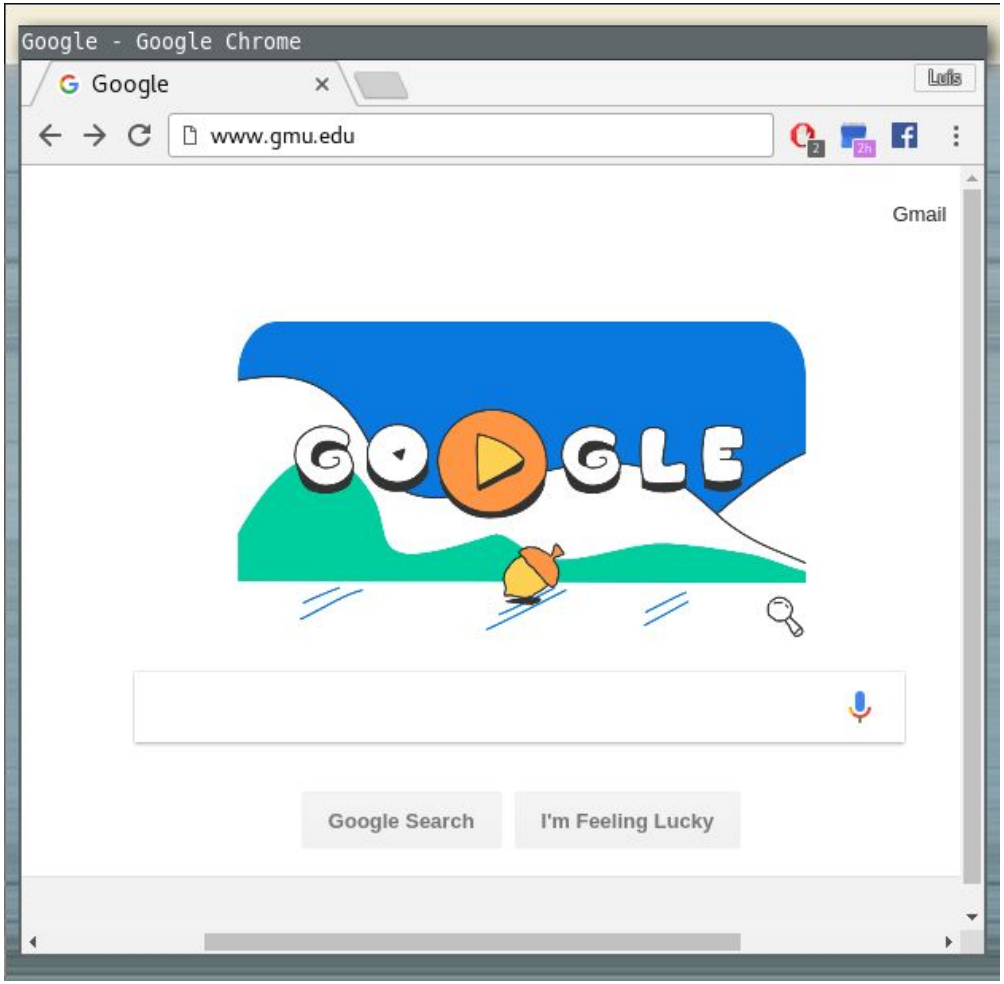
Domain Name System

Name Discovery

- What happens after typing the name of the host?



- What happens after typing the name of the host?
- The internet routes packets using the Internet Protocol (IP)
- Each host has its own IP address (e.g., `127.0.0.1`)
- We did not provide any IP address
  - We provided a name
- How can we find IP addresses from their name?
- **Domain Name System (DNS)**



Domain Name System

- Obvious solution: Local file
  - Keep local copy of mapping from all hosts to all IPs (e.g., `/etc/hosts`)
  - Hosts change IPs regularly: Download file frequently
  - IPv4 space is now full
    - 32-bits: 4,294,967,296 addresses
    - At 1 byte per address, file would be 4GB
    - Not a lot of disk space (now, DNS introduced in the late 80s)

Domain Name System


your Tandy 1000s/3000/4000 or PC Compatible on a user-installable card. Includes manual and software for easy installation. The easy way to get hard disk power for your computer system. **25-4059** **799.00**

Add an External Hard Disk Drive

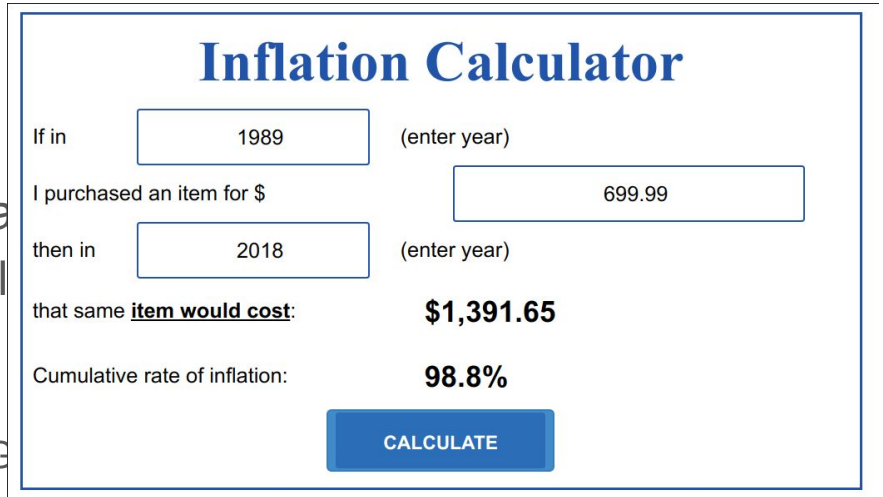
699.00 Low As $40 Per Month*

- **Alternative Expansion Option**
- **Allows Greater Data Storage**

20-Megabyte External Hard Disk Drive. (Cable Kit and installation required for secondary unit.) Requires Hard Disk Controller Board (25-1007).
25-1041 . . . . . **699.00**

Hard Disk Controller Board. For Tandy 1000 SX/SL and original Tandy 1000 only. Allows you to add hard disk drives for up to 40 million characters of storage. Includes cable for use with 10 or 20-megabyte hard disks.
25-1007 . . . . . **299.95**

180 PRICES APPLY AT PARTICIPATING RA

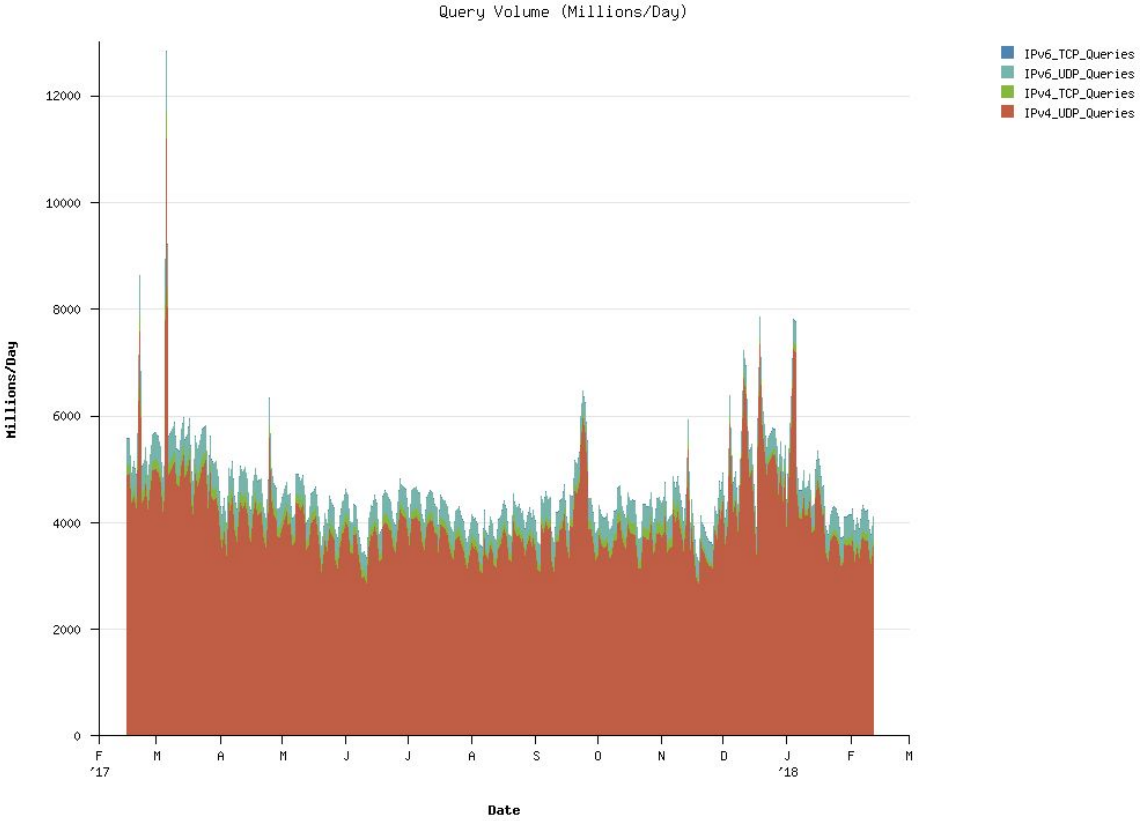


We need 200x of these to hold 4GB: $270K+

Domain Name System

- Obvious solution: Local file
  - Keep local copy of mapping from all hosts to all IPs (e.g., `/etc/hosts`)
  - Hosts change IPs regularly: Download file frequently
  - IPv4 space is now full
    - 32-bits: 4,294,967,296 addresses
    - At 1 byte per address, file would be 4GB
    - Not a lot of disk space (now, DNS introduced in the late 80s)
    - But a lot of constant internet bandwidth
  - More names than IPs
    - Aliases
  - **Not scalable!**

Domain Name Syst



- Obvious solution: L
  - Keep local copy of m
  - Hosts change IPs reg
  - IPv4 space is now fu
    - 32-bits: 4,294,967,29
    - At 1 byte per address
    - Not a lot of disk spac
    - But a lot of constant i
  - More names than IPs
    - Aliases
  - **Not scalable!**

http://a.root-servers.org/static/index.html

- Obvious solution: Well known centralized server
  - Single point of failure
  - Traffic volume
  - Access time
  - **Not scalable!**

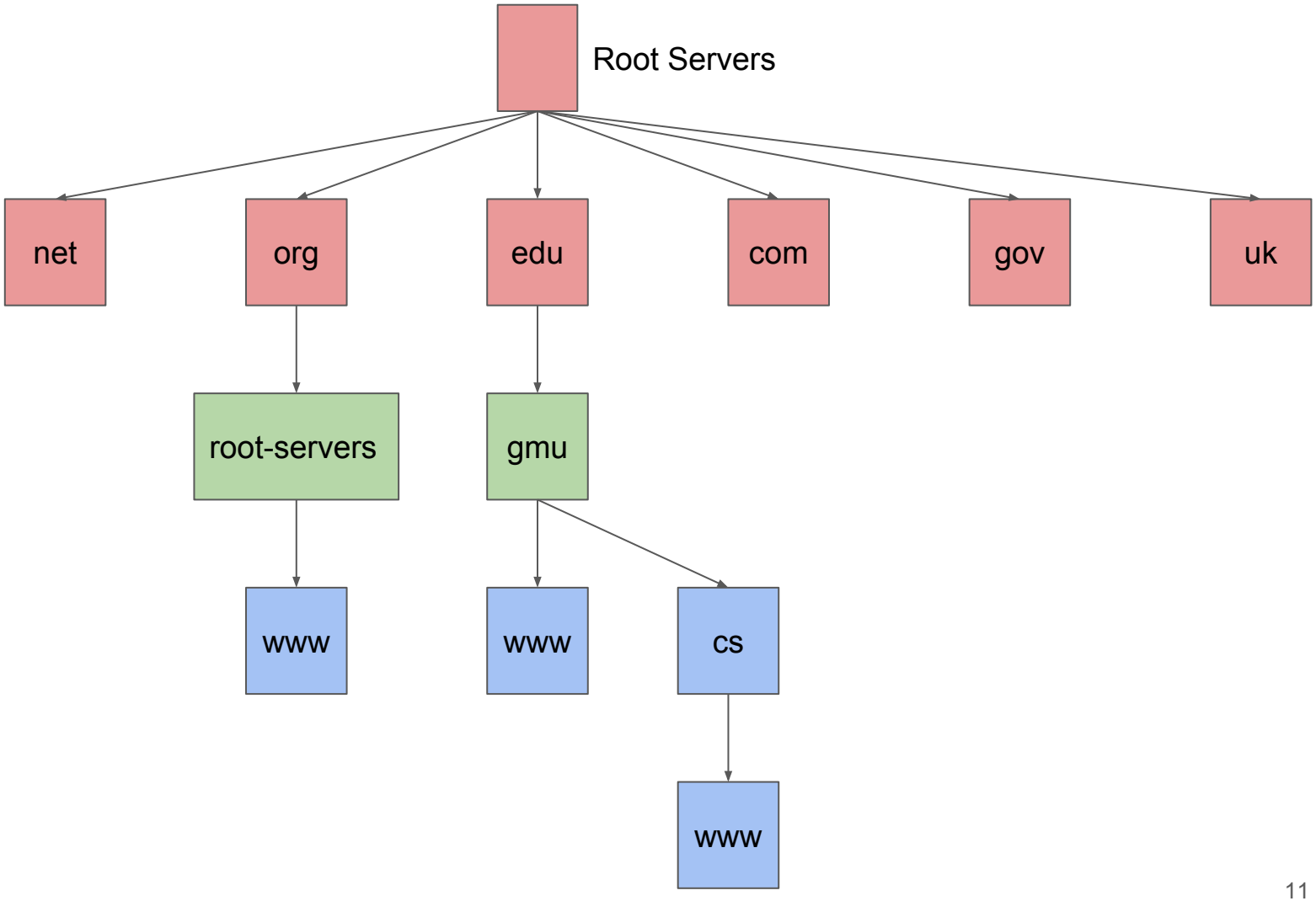
Domain Name System

- Obvious solution: Local file
  - Keep local copy of mapping from all hosts to all IPs (e.g., `/etc/hosts`)
  - Hosts change IPs regularly: Download file frequently
  - IPv4 space is now full
    - 32-bits: 4,294,967,296 addresses
    - At 1 byte per address, file would be 4GB
    - Not a lot of disk space (now, DNS introduced in the late 80s)
    - But a lot of constant internet bandwidth
  - More names than IPs
    - Aliases
  - **Not scalable!**
- Obvious solution: Well known centralized server
  - Single point of failure
  - Traffic volume
  - Access time
  - **Not scalable!**

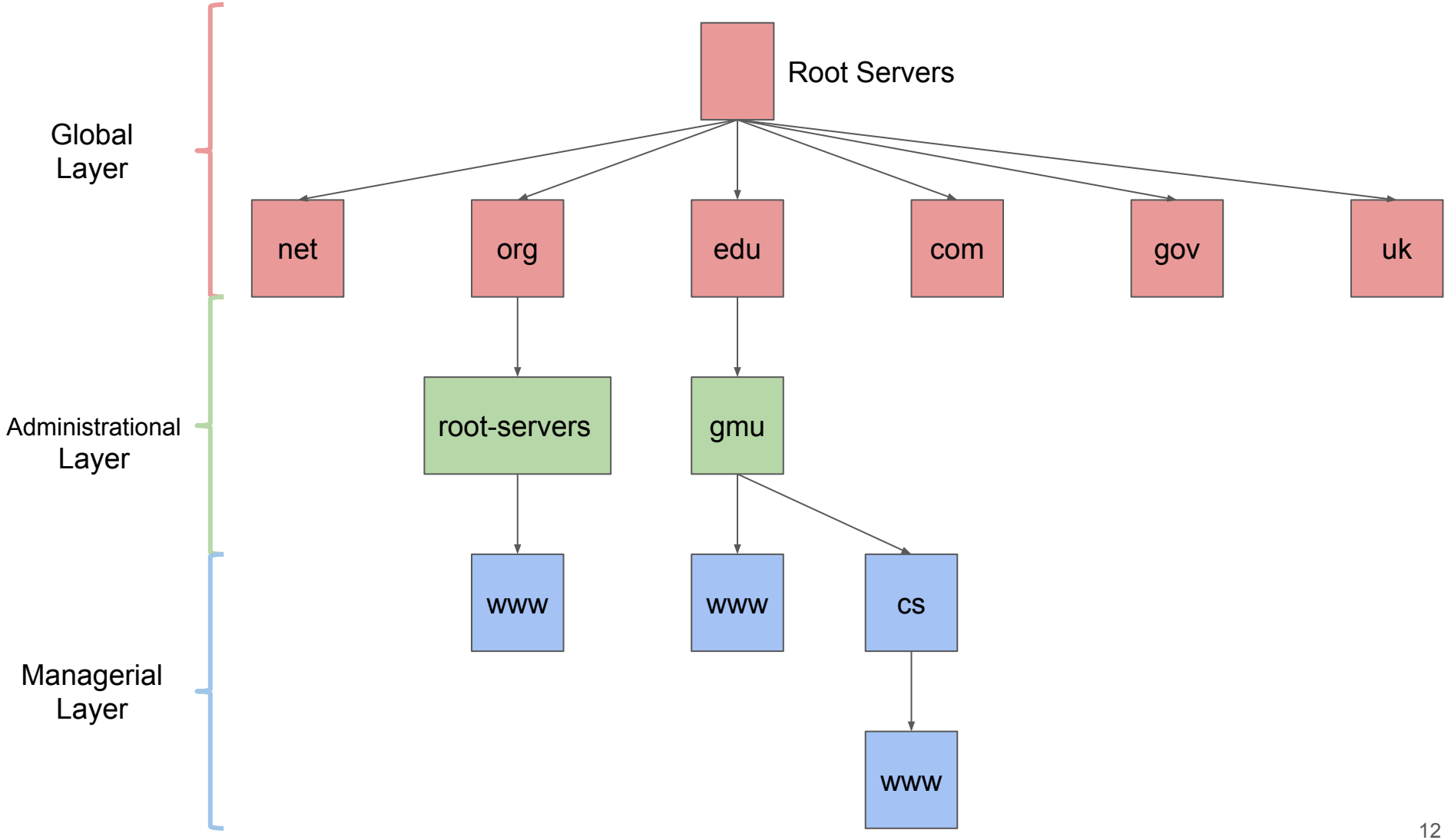
Domain Name System

- Goals
  - Scalable
  - Robust
    - High availability
  - Decentralized maintenance
  - Global scope
    - Names mean the same thing everywhere
- Non-goals
  - Atomicity
  - Strong consistency

Domain Name System



Domain Name System



Domain Name System - Root Servers

- 13 root servers
  - `[a-m].root-servers.org`
  - E.g., `d.root-servers.org`
- Handled by 12 entities
- How many physical servers?
  - a) Less than 13
  - b) 13
  - c) Tens
  - d) Hundreds
  - e) Thousands
  - f) Millions

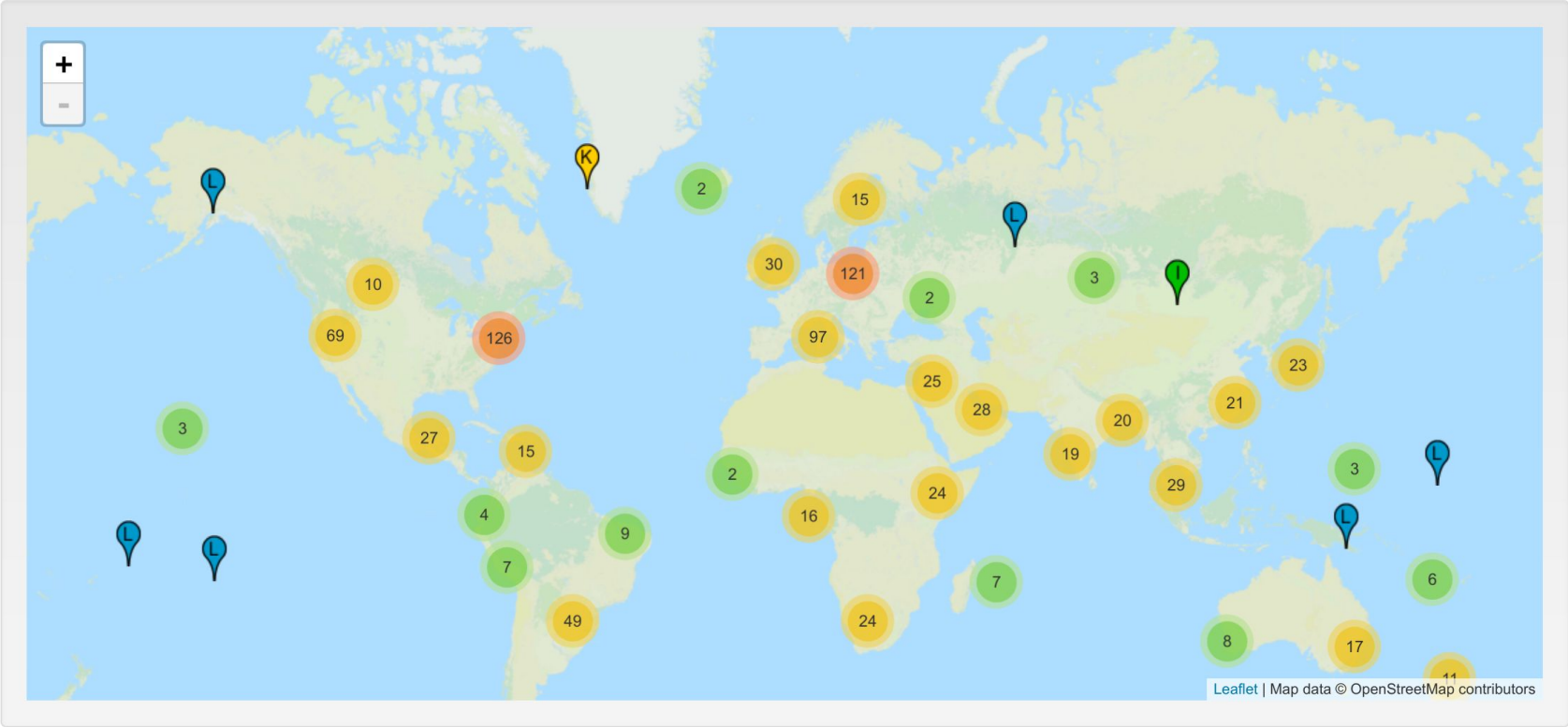
Verisign, Inc.	a
Information Sciences Institute	b
Cogent Communications	c
University of Maryland	d
NASA Ames Research Center	e
Internet Systems Consortium, Inc.	f
U.S. DOD Network Information Center	g
U.S. Army Research Lab	h
Netnod	i
Verisign, Inc.	j
RIPE NCC	k
ICANN	l
WIDE Project	m

Domain Name System - Root Servers

- 13 root servers
  - `[a-m].root-servers.org`
  - E.g., `d.root-servers.org`
- Handled by 12 entities
- How many physical servers?
  - ~~a) Less than 13~~
  - ~~b) 13~~
  - ~~c) Tens~~
  - d) Hundreds
    - 980
  - ~~e) Thousands~~
  - ~~f) Millions~~

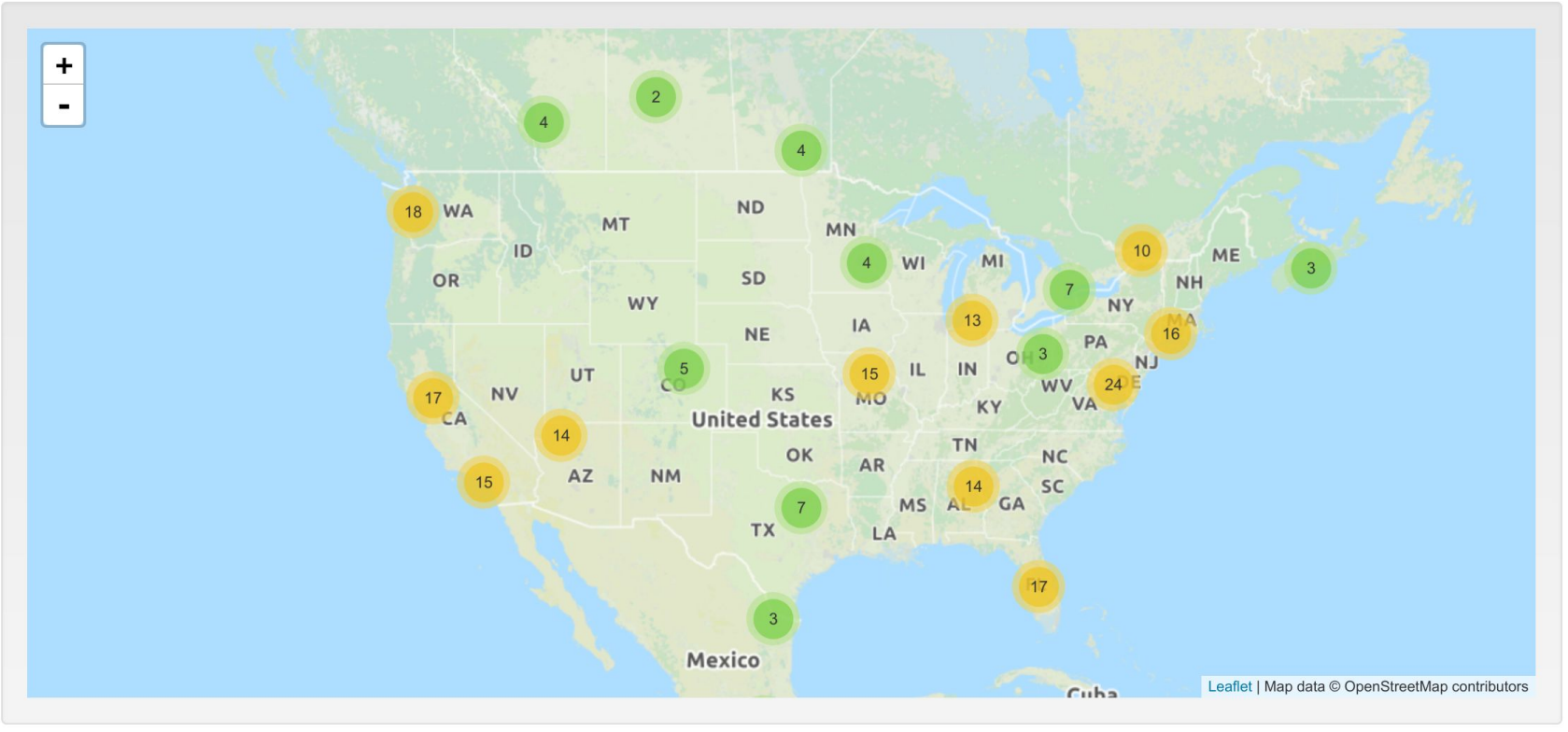
Verisign, Inc.	a	5
Information Sciences Institute	b	2
Cogent Communications	c	10
University of Maryland	d	128
NASA Ames Research Center	e	191
Internet Systems Consortium, Inc.	f	193
U.S. DOD Network Information Center	g	6
U.S. Army Research Lab	h	2
Netnod	i	60
Verisign, Inc.	j	159
RIPE NCC	k	58
ICANN	l	157
WIDE Project	m	9

Domain Name System - Root servers



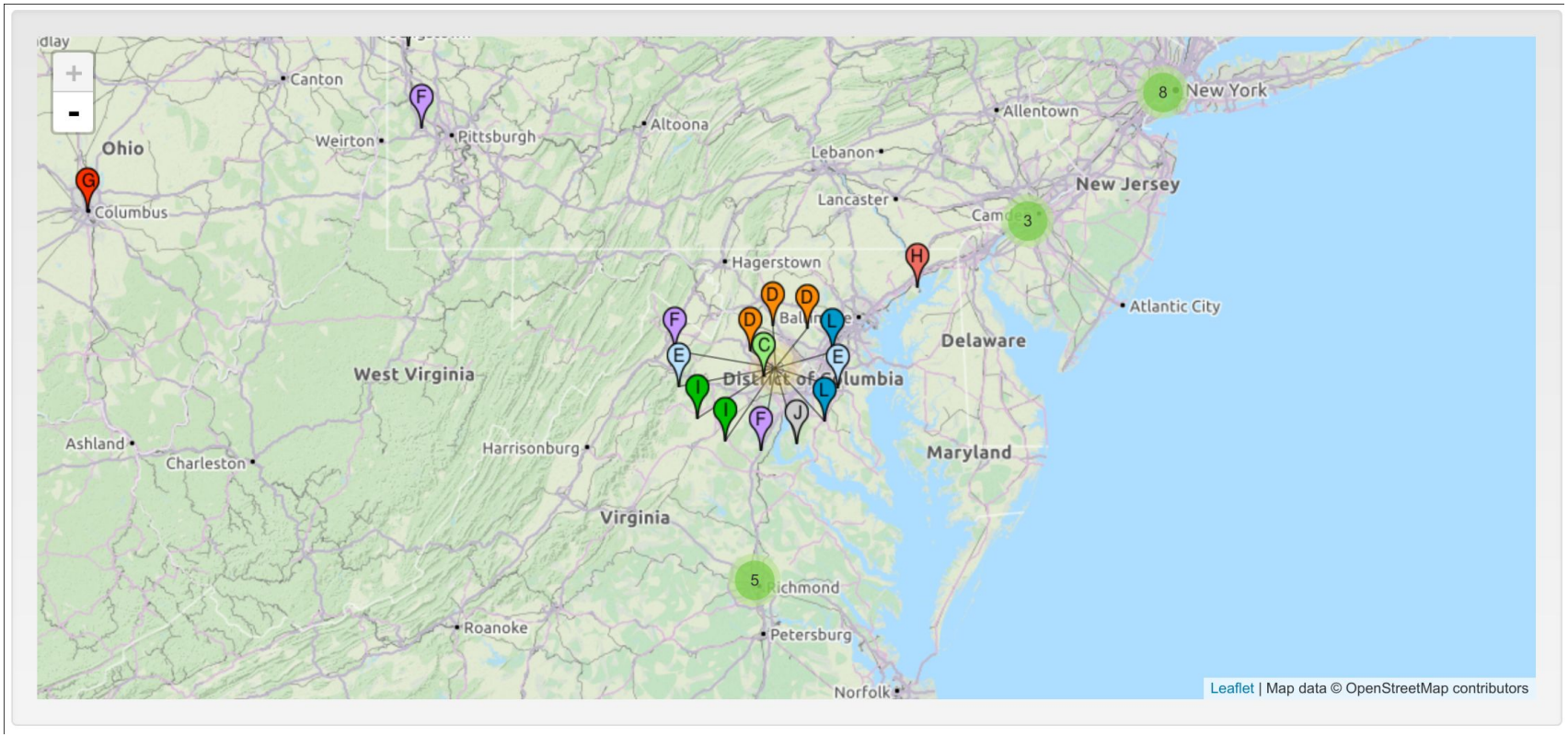
www.root-servers.net

Domain Name System - Root servers



www.root-servers.net

Domain Name System - Root servers

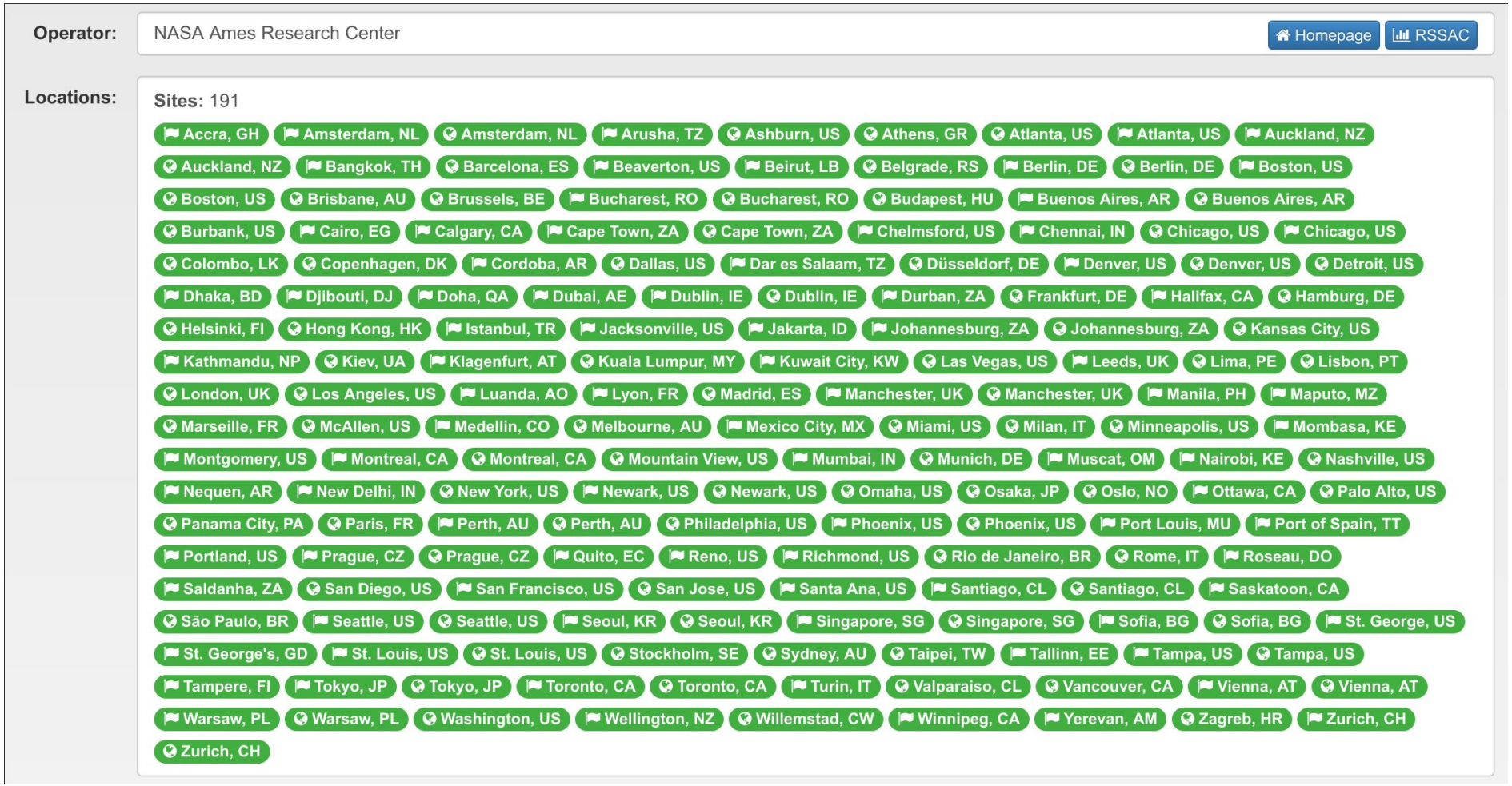


www.root-servers.net

Domain Name System - Root servers

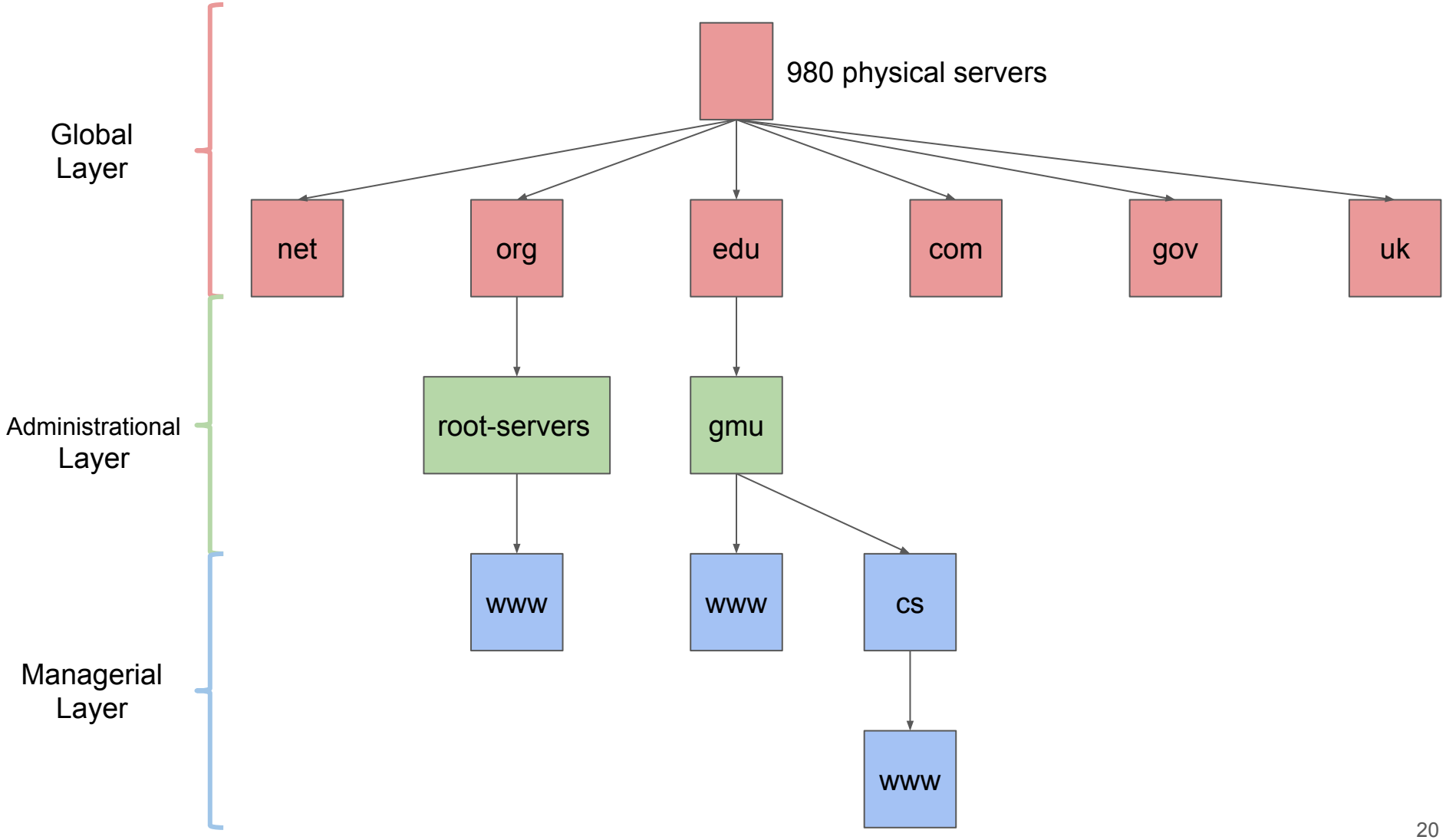
www.root-servers.net

Domain Name System - Root servers

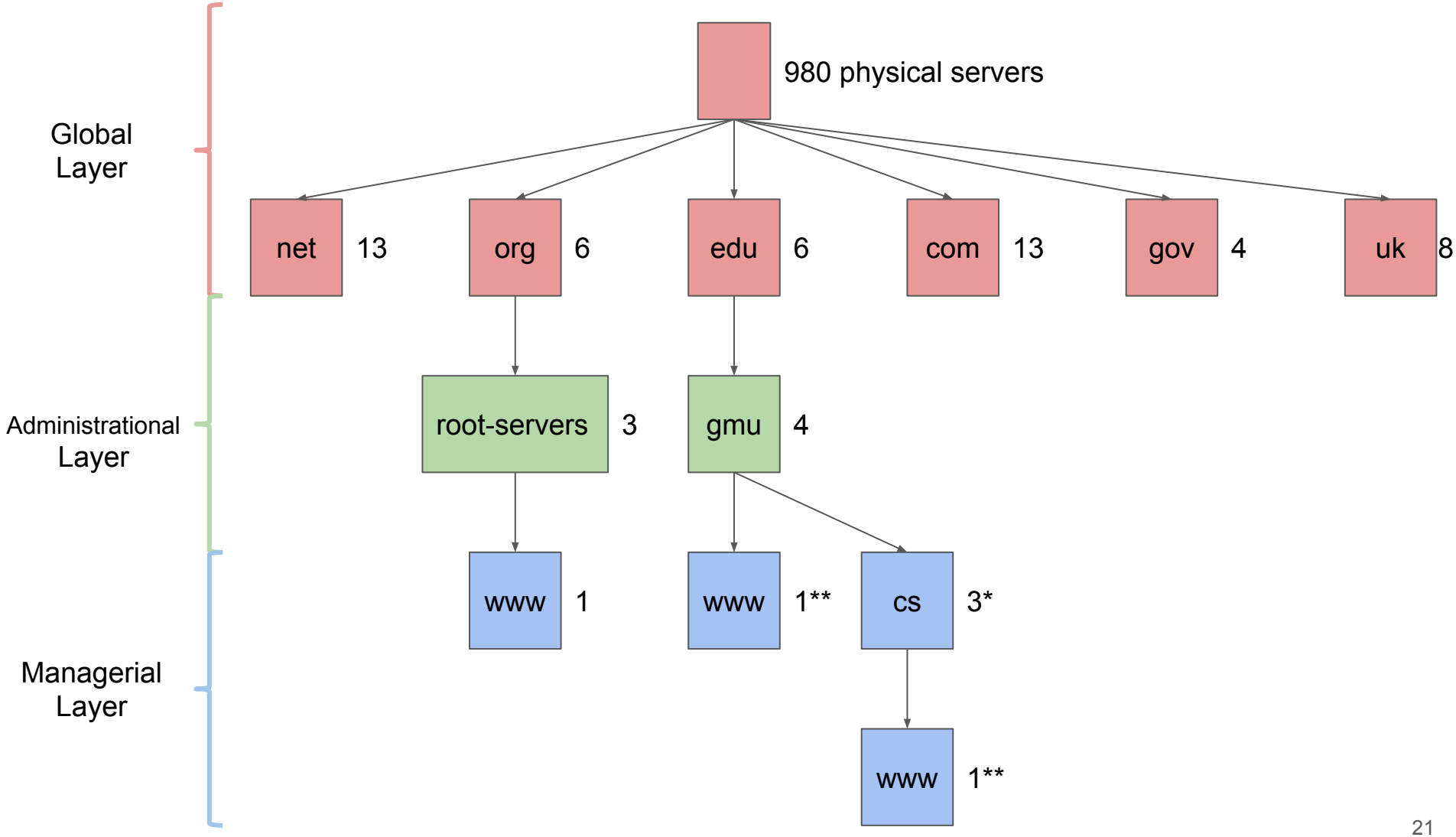


www.root-servers.net

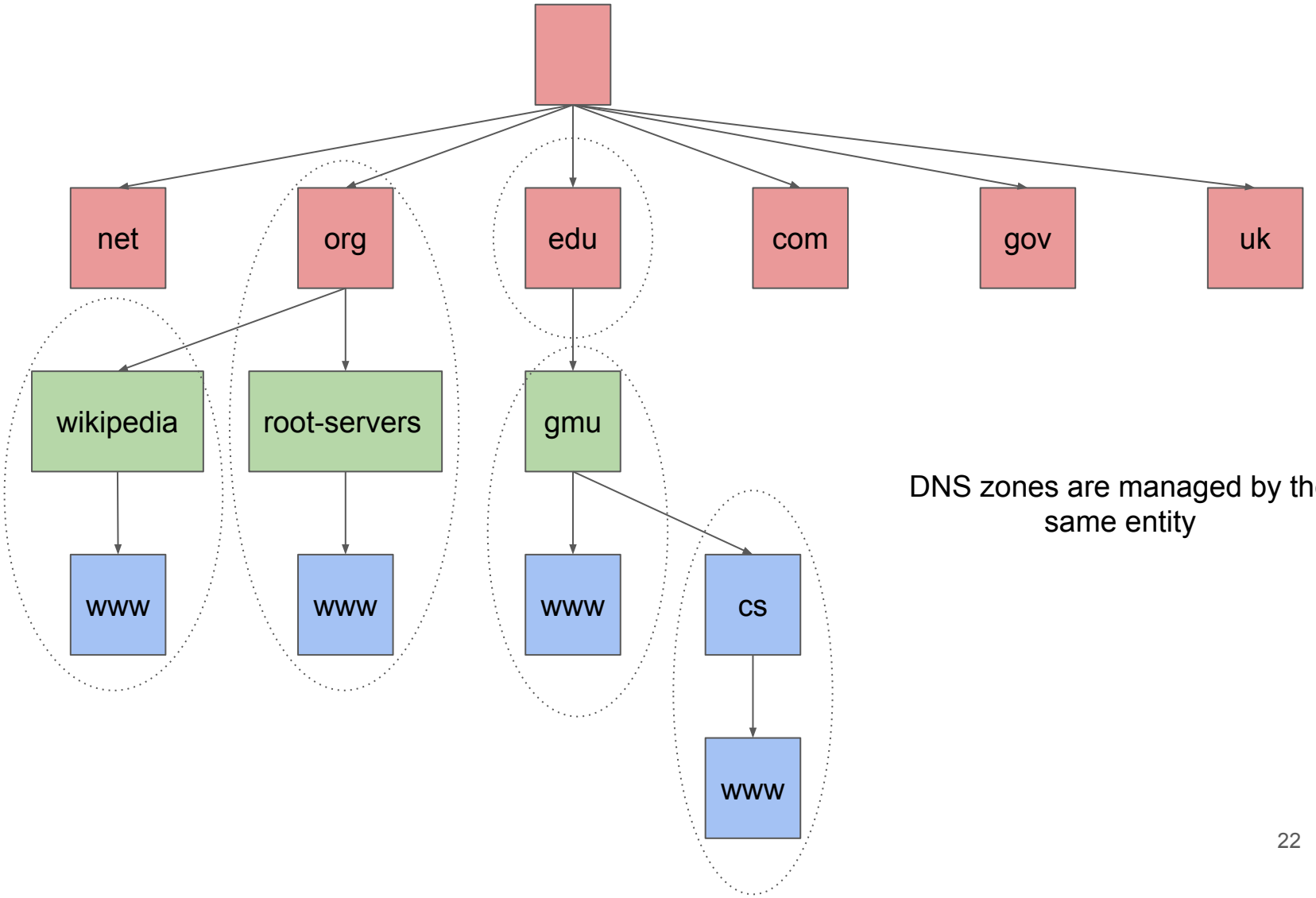
Domain Name System - Scale



Domain Name System - Scale



Domain Name System - Zones



DNS zones are managed by the same entity

Domain Name System - Questions/Answers

- DNS message format is the same for questions and answers
  - Header
    - ID
      - Question/Answer have the same ID
    - Flags
      - 1 bit for question/answer, etc.
    - Number of questions
    - Number of answers
    - Number of authority RRs (Resource Records)
    - Number of additional RRs
  - Questions
  - Answers
  - Authority
  - Additional Info

Domain Name System - Resource Records (RRs)

- RR format: (class, name, value, type, ttl)
  - Class: Internet (IN)
  - Type
    - A (AAAA for IPv6)
      - name is hostname
      - value is IP address
    - NS
      - name is domain
      - value is authoritative server for domain
    - CNAME
      - name is alias for some canonical (real) name
      - value is canonical name
    - And more...

Domain Name System - Example Query

dig(1) - Linux man page

Name

dig - DNS lookup utility

Synopsis

dig [@server] [**-b** *address*] [**-c** *class*] [**-f** *filename*] [**-k** *filename*] [**-m**] [**-p** *port#*] [**-q** *name*] [**-t** *type*] [**-x** *addr*] [**-y***[hmac:]name:key*] [**-4**] [**-6**] [name] [type] [class] [queryopt...]
dig [**-h**]

dig [global-queryopt...] [query...]

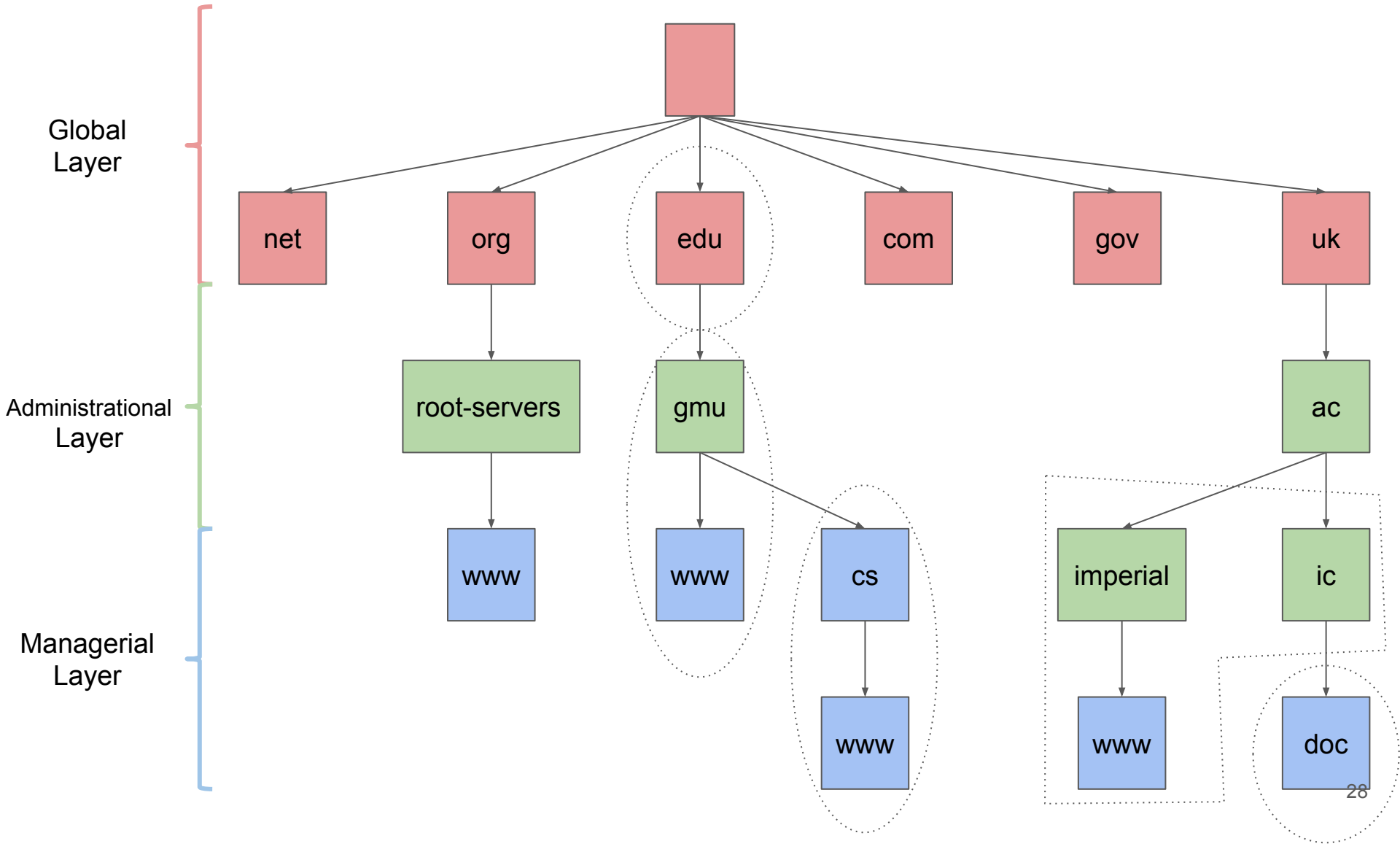
Domain Name System - Example Query

```
> dig www.gmu.edu a
; <<>> DiG 9.11.2 <<>> www.gmu.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 20159
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 3, ADDITIONAL: 3

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.gmu.edu.                   IN      A

;; ANSWER SECTION:
www.gmu.edu.            77455   IN      CNAME   jiju3.gmu.edu.
jiju3.gmu.edu.          46534   IN      A       129.174.1.59

;; AUTHORITY SECTION:
gmu.edu.                86013   IN      NS      eve.gmu.edu.
gmu.edu.                86013   IN      NS      uvaarpa.virginia.edu.
gmu.edu.                86013   IN      NS      magda.gmu.edu.

;; ADDITIONAL SECTION:
eve.gmu.edu.            3219    IN      A       129.174.253.66
magda.gmu.edu.          1993    IN      A       129.174.18.18
uvaarpa.virginia.edu.   84640   IN      A       128.143.2.7

;; Query time: 2 msec
;; SERVER: 192.168.1.1#53(192.168.1.1)
;; WHEN: Thu Feb 15 10:19:24 EST 2018
;; MSG SIZE  rcvd: 212
```

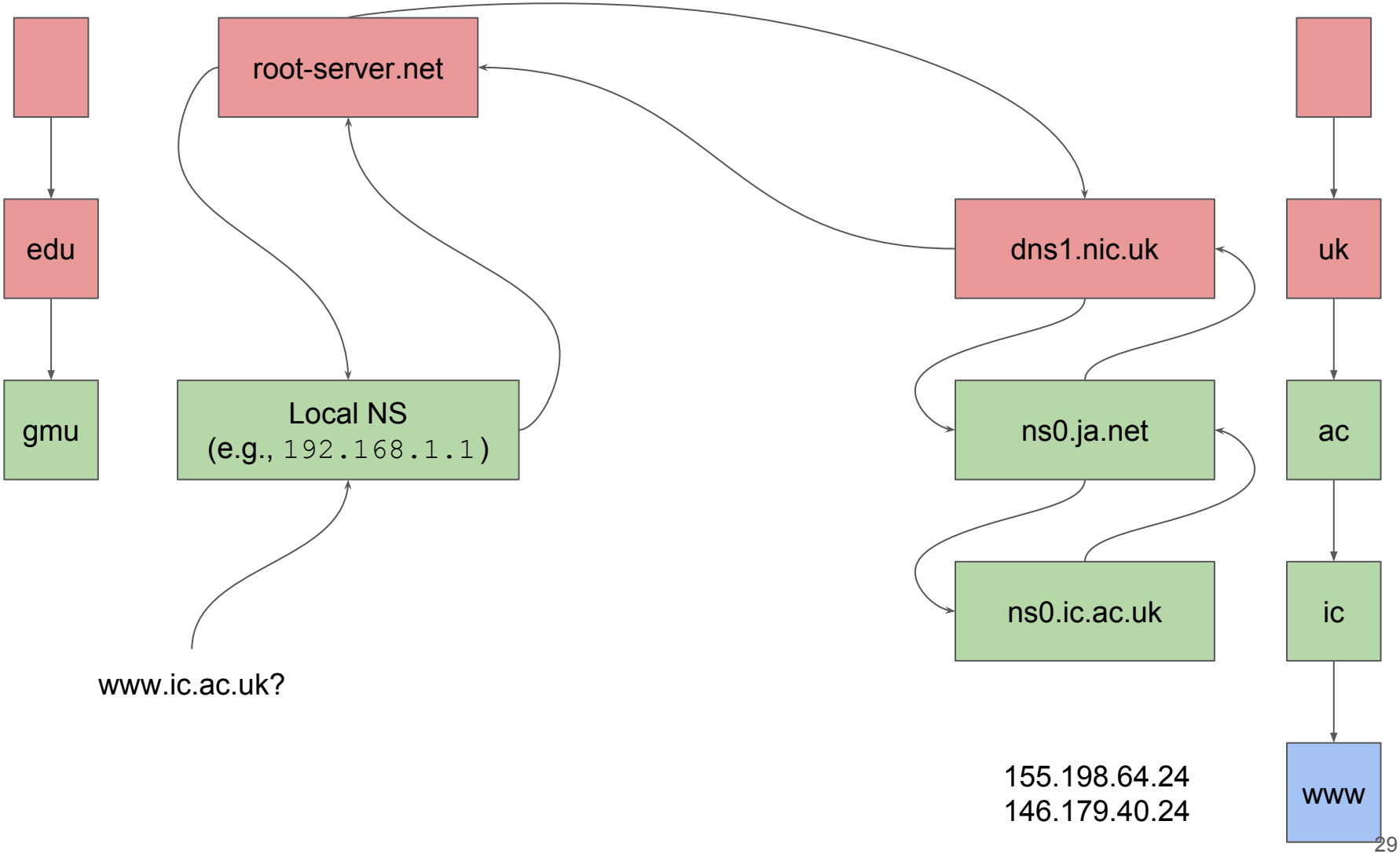
Domain Name System - Example Query

```
> dig www.ic.ac.uk a
; <<>> DiG 9.11.2 <<>> www.ic.ac.uk a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28622
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.ic.ac.uk.                  IN      A

;; ANSWER SECTION:
www.ic.ac.uk.           271     IN      CNAME   wrp.cc.gslb.ic.ac.uk.
wrp.cc.gslb.ic.ac.uk.   1       IN      A       146.179.40.24

;; Query time: 1 msec
;; SERVER: 192.168.1.1#53(192.168.1.1)
;; WHEN: Thu Feb 15 10:23:52 EST 2018
;; MSG SIZE  rcvd: 83
```

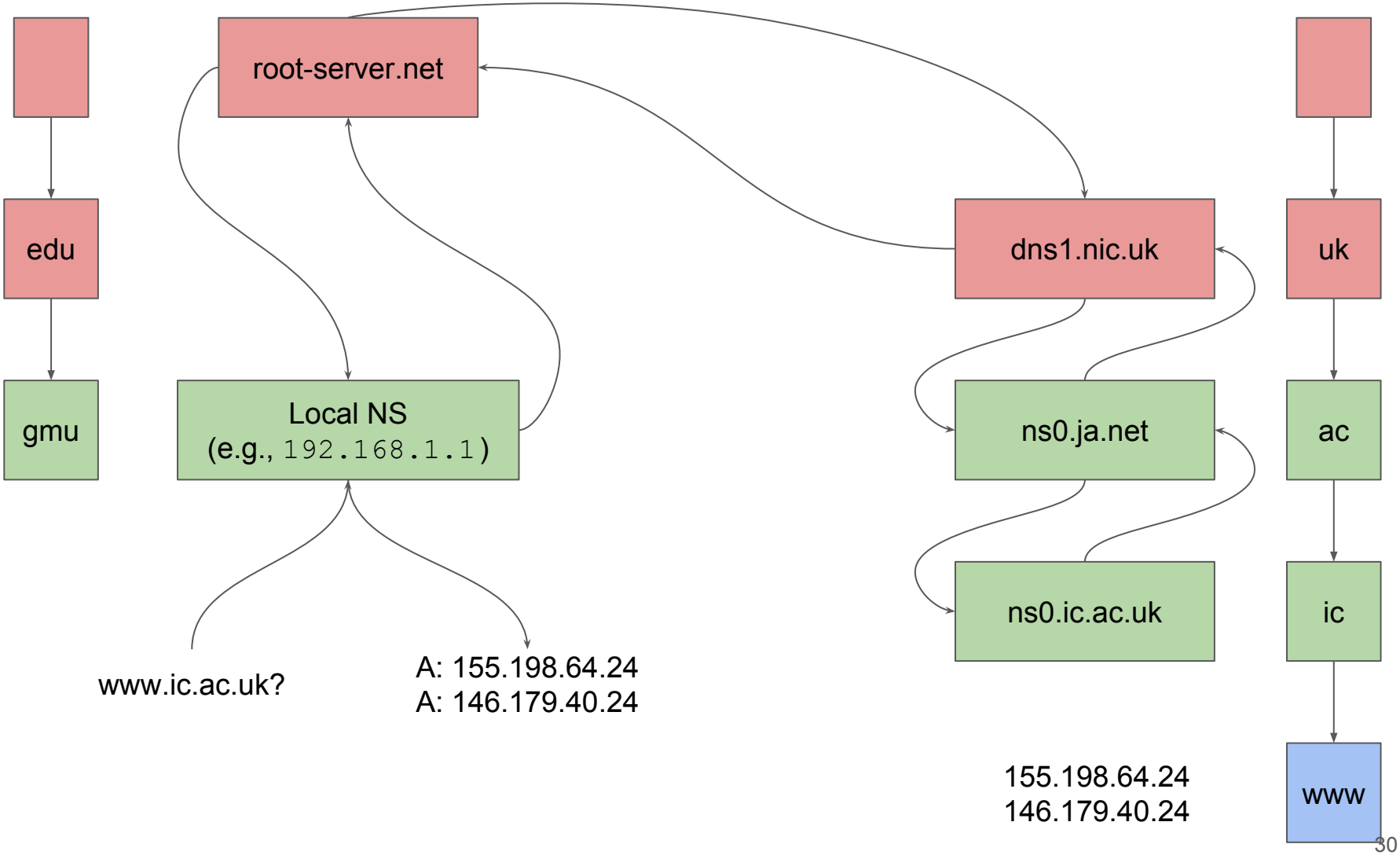
Domain Name System



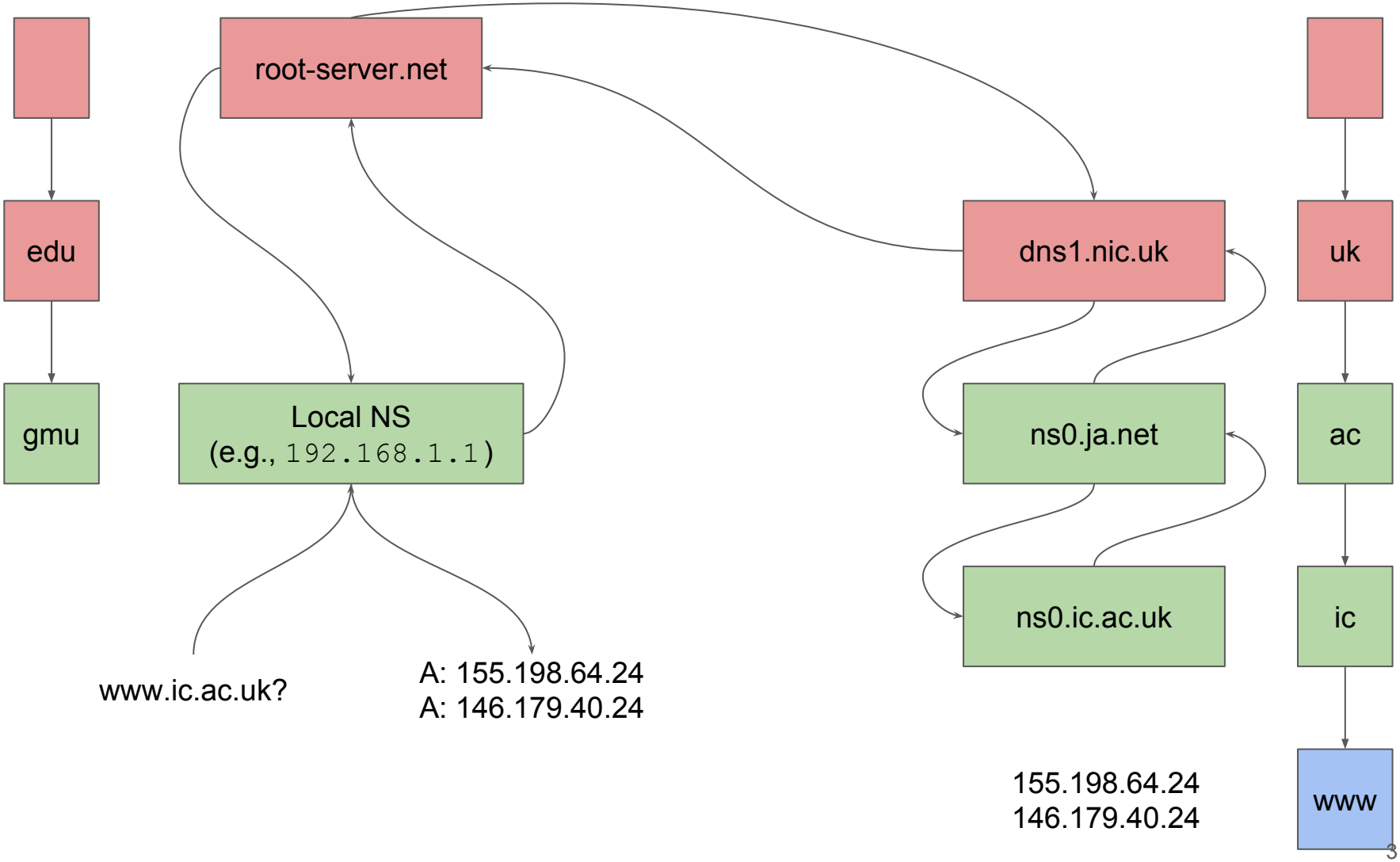
Domain Name System - Resolution



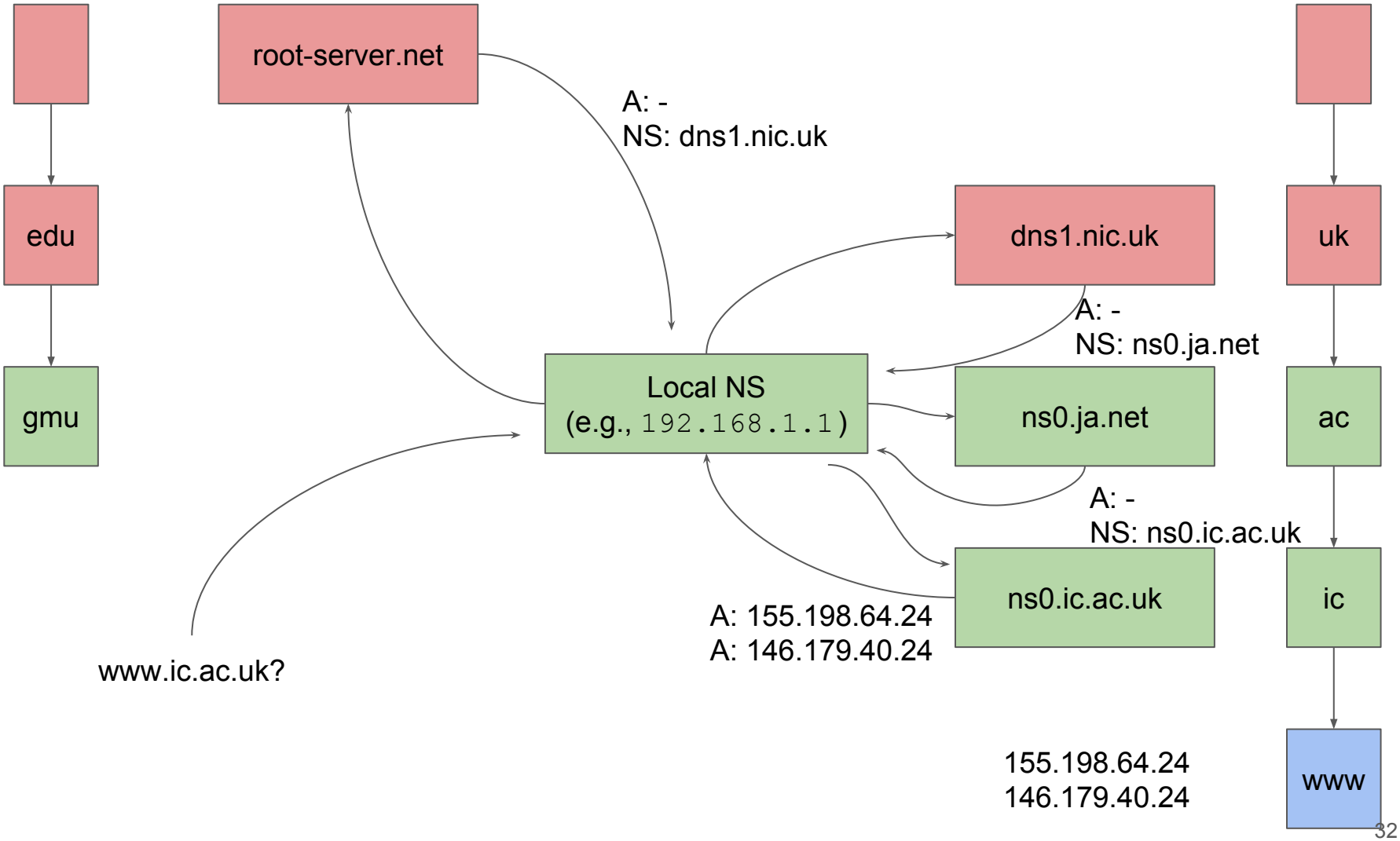
Domain Name System - Resolution



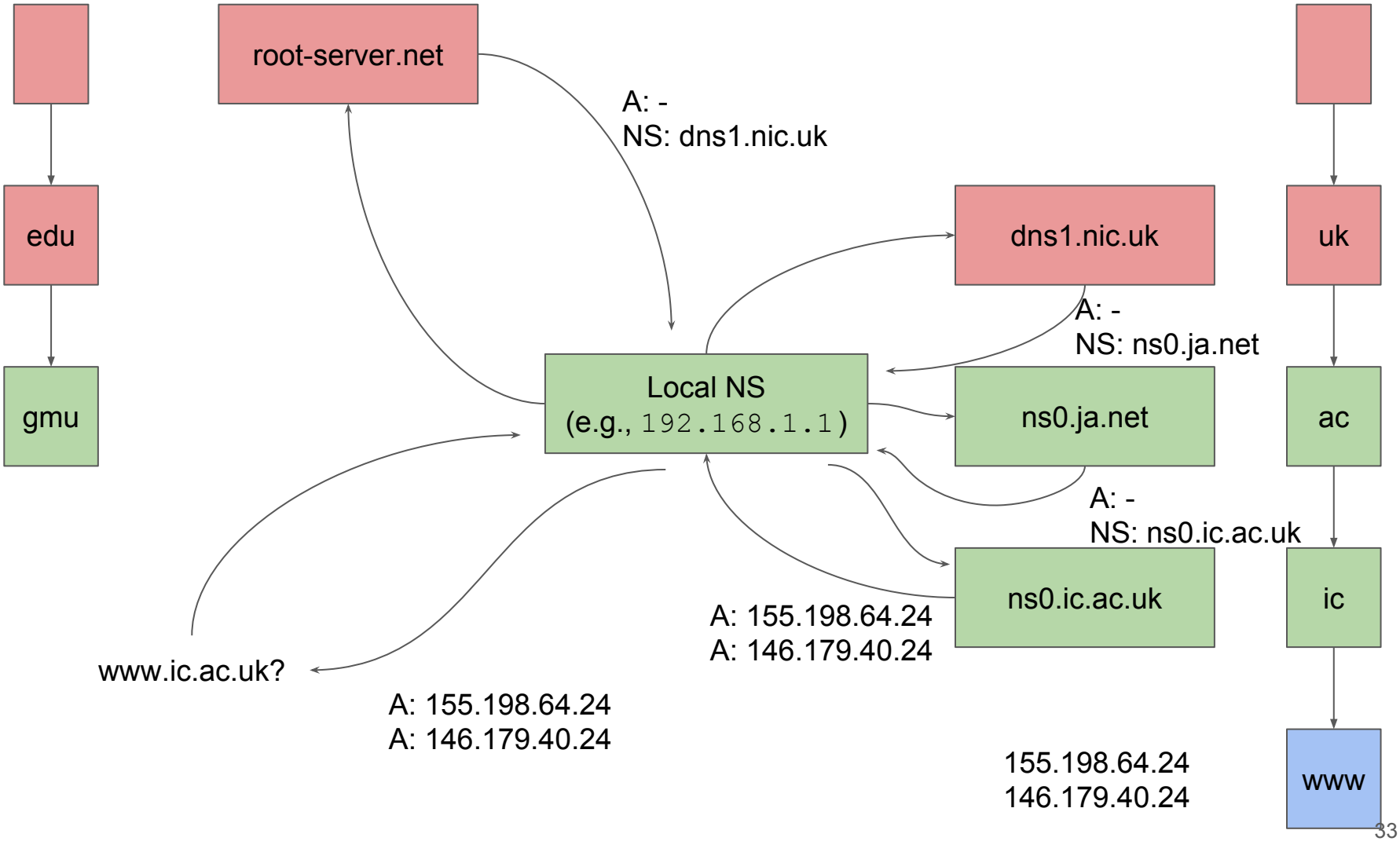
Domain Name System - (Recursive) Resolution



Domain Name System - Iterative Resolution

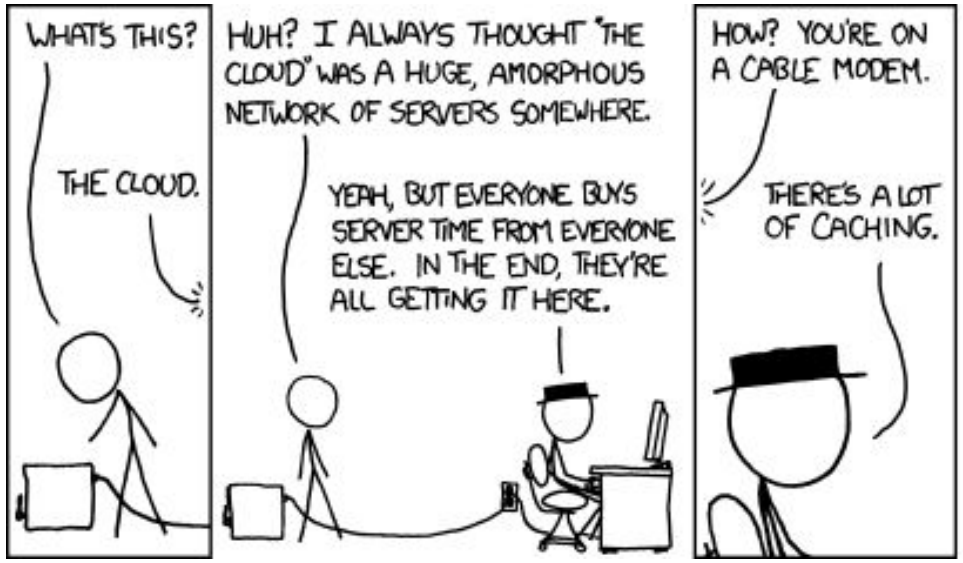


Domain Name System - Iterative Resolution



Domain Name System - Example Query

```
> cat /etc/resolv.conf #which DNS server am I using
# Generated by resolvconf
search fios-router.home
nameserver 192.168.1.1
> dig @192.168.1.1 www.ic.ac.uk a #recursive query
...
> dig +norecurse @192.168.1.1 www.ic.ac.uk a
# same answer, why?
# what if I had tried this first?
```



https://xkcd.com/908/

Domain Name System - Caching

Some gamers steamed over alleged Valve anti-cheat DNS spying - CSO
https://www.csoonline.com/.../some-gamers-steamed-over-alleged-valve-anti-cheat-dn...
Feb 16, 2014 - Goes through all your **DNS Cache** entries (ipconfig /displaydns); Hashes each one with MD5; Reports back to VAC Servers. **Valve** is not the only company that uses an **anti-cheat** system, but it is perhaps one of the most highly regarded companies as countless millions of gamers have Steam. Various ...

Reddit user claims Valve Anti-Cheat scans your DNS cache - PC ...
https://gamefaqs.gamespot.com/boards/916373-pc/68593356
For PC on the PC, a GameFAQs message board topic titled "Reddit user claims **Valve Anti-Cheat** scans your **DNS cache**".

Valve Anti-Cheat seems to scan your DNS cache, but probably doesn ...
https://www.neogaf.com › Discussions › Gaming Discussion
Feb 16, 2014 - Trust is a critical part of a multiplayer game community - trust in the developer, trust in the system, and trust in the other players. **Cheats** are a negative sum game, where a minority benefits less than the majority is harmed. There are a bunch of different ways to attack a trust-based system including writing a ...

Report: Valve anti-cheat scans your DNS history - Player Attack
https://www.playerattack.com/news/.../report-valve-anti-cheat-scans-your-dns-history/
Feb 17, 2014 - Even if you've never actively visited a **cheat** website, there may be traces of them in your DNS, and that's what VAC is reportedly now looking for. The news was first posted to the **Counter**-Strike: Global Offensive Reddit, explaining that VAC now: Goes through all your **DNS Cache** entries (ipconfig ...

IS IT OK THAT VAC SCANS YOUR DNS CACHE? :: VAC Discussion - Steam...
steamcommunity.com › Steam Forums › VAC Discussion
Feb 15, 2014 - 16 posts - 7 authors
Valve ANSWER THIS! http://www.ghacks.net/2014/02/16/steams-vac-protection-now-scans-ans-transfers-**dns-cache**/ What is going on with this DNS spying? We all know that various **anti-cheat** programs do check your DNS, some of them don't really collect data though. It has been proven that VAC collects ...

Domain Name System - Caching

There are a number of **kernel-level paid cheats** that relate to [this Reddit thread](). Cheat developers have a problem in getting cheaters to actually pay them for all the obvious reasons, so they start creating DRM and anti-cheat code for their cheats. These cheats **phone home to a DRM server that confirms that a cheater has actually paid** to use the cheat.

VAC checked for the presence of these cheats. If they were detected VAC then checked to see which cheat DRM server was being contacted. This second check was done by **looking for a partial match to those (non-web) cheat DRM servers in the DNS cache**. If found, then hashes of the matching DNS entries were sent to the VAC servers. The match was double checked on our servers and then that client was marked for a future ban. Less than a tenth of one percent of clients triggered the second check. 570 cheaters are being banned as a result.

Gabe Newell, Valve's CEO

https://www.reddit.com/r/gaming/comments/1y70ej/valve_vac_and_trust/

Domain Name System - Example Query

```
> cat /etc/resolv.conf #which DNS server am I using
# Generated by resolvconf
search fios-router.home
nameserver 192.168.1.1
> dig @192.168.1.1 www.ic.ac.uk a #recursive query
...
> dig +norecurse @192.168.1.1 www.ic.ac.uk a
# same answer, why?
# what if I had tried this first?

> dig @a.root-servers.net www.ic.ac.uk a
; <<>> DiG 9.11.2 <<>> @a.root-servers.net www.ic.ac.uk a
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 682
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 8, ADDITIONAL: 14
;; WARNING: recursion requested but not available

;; AUTHORITY SECTION:
uk.                     172800  IN      NS      dns1.nic.uk.
(7 more)

;; ADDITIONAL SECTION:
dns1.nic.uk.            172800  IN      A       213.248.216.1
dns1.nic.uk.            172800  IN      AAAA    2a01:618:400::1
...
```

Domain Name System - Manual Iterative Resolution

```
> dig @dns1.nic.uk www.ic.ac.uk a
;; AUTHORITY SECTION:
ac.uk.                  172800  IN      NS      ns0.ja.net.
...

> dig @ns0.ja.net www.ic.ac.uk a
;; AUTHORITY SECTION:
ic.ac.uk.               86400   IN      NS      ns0.ic.ac.uk.
...

> dig @ns0.ic.ac.uk www.ic.ac.uk a
;; ANSWER SECTION:
www.ic.ac.uk.           300     IN      CNAME   wrp.cc.gslb.ic.ac.uk.

;; AUTHORITY SECTION:
gslb.ic.ac.uk.          86400   IN      NS      gslb1.net.ic.ac.uk.

> dig @gslb1.net.ic.ac.uk wrp.cc.gslb.ic.ac.uk a
;; ANSWER SECTION:
wrp.cc.gslb.ic.ac.uk.   30      IN      A       146.179.40.24
```

Domain Name System - Load Balancing

```
> dig ic.ac.uk a
;; ANSWER SECTION:
ic.ac.uk.               86363   IN      A       155.198.63.21
ic.ac.uk.               86363   IN      A       155.198.30.71
ic.ac.uk.               86363   IN      A       129.31.100.150
ic.ac.uk.               86363   IN      A       155.198.30.55
ic.ac.uk.               86363   IN      A       146.179.32.12
ic.ac.uk.               86363   IN      A       129.31.47.2
ic.ac.uk.               86363   IN      A       155.198.30.98
ic.ac.uk.               86363   IN      A       146.179.32.37
ic.ac.uk.               86363   IN      A       129.31.22.11

> dig ic.ac.uk a
;; ANSWER SECTION:
ic.ac.uk.               86340   IN      A       129.31.47.2
ic.ac.uk.               86340   IN      A       155.198.30.55
ic.ac.uk.               86340   IN      A       129.31.22.11
ic.ac.uk.               86340   IN      A       155.198.30.71
ic.ac.uk.               86340   IN      A       146.179.32.37
ic.ac.uk.               86340   IN      A       155.198.63.21
ic.ac.uk.               86340   IN      A       146.179.32.12
ic.ac.uk.               86340   IN      A       129.31.100.150
ic.ac.uk.               86340   IN      A       155.198.30.98
```

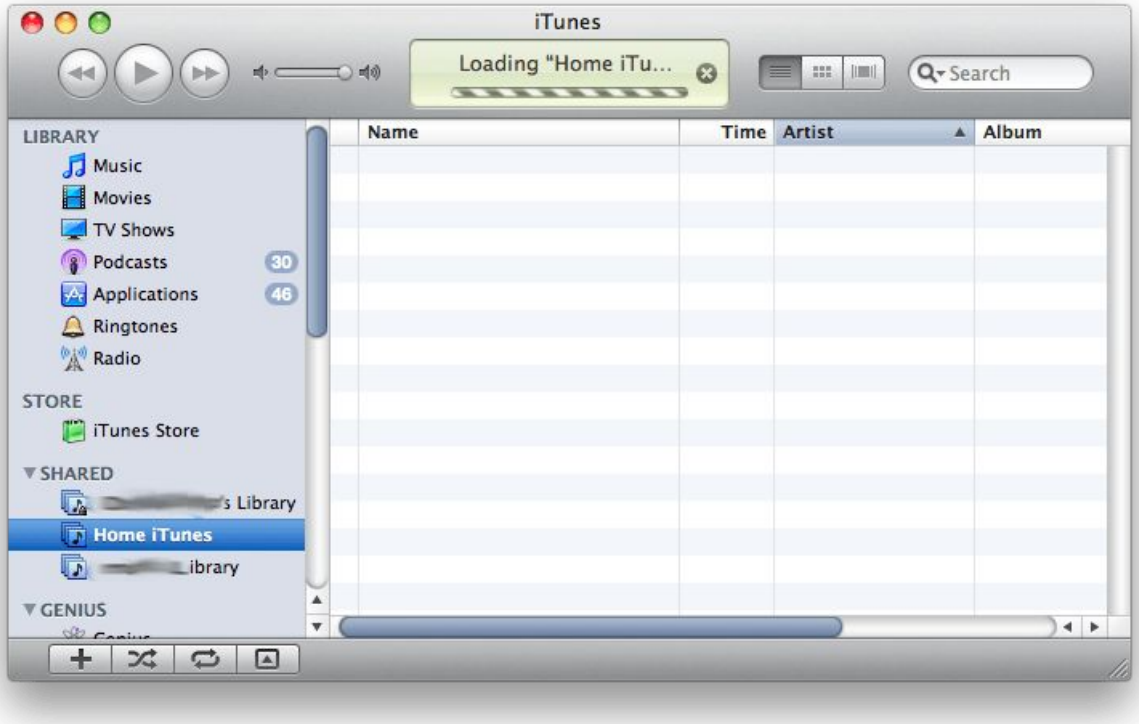
Different order: Load balancing requests amo all available servers

~~Domain Name System~~

Name Discovery

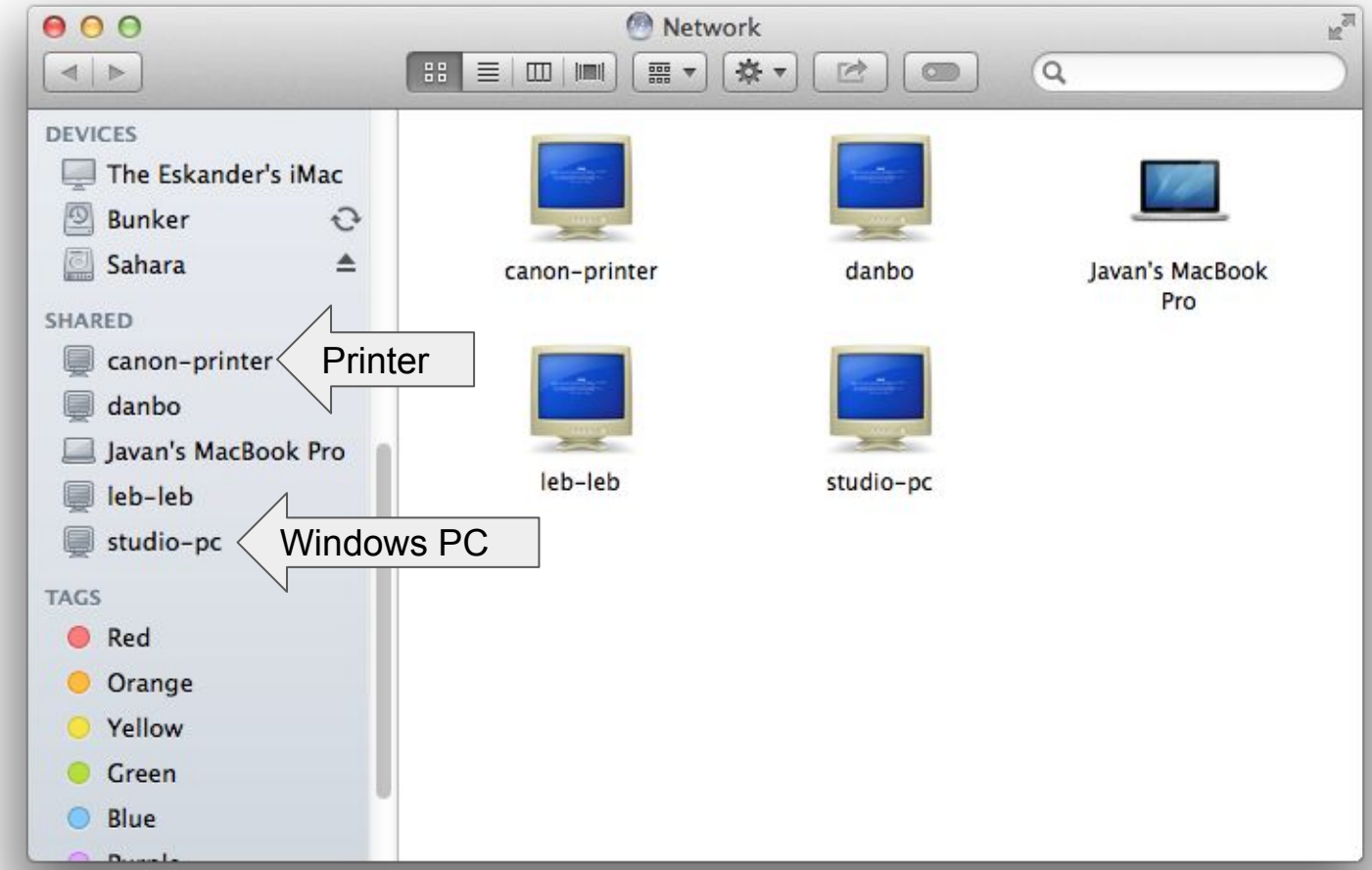
Name Discovery

- No configuration
- Automatic discovery of other hosts on the same network
- Does this use DNS?
  - Global/Local



How does this work?

Name Discovery

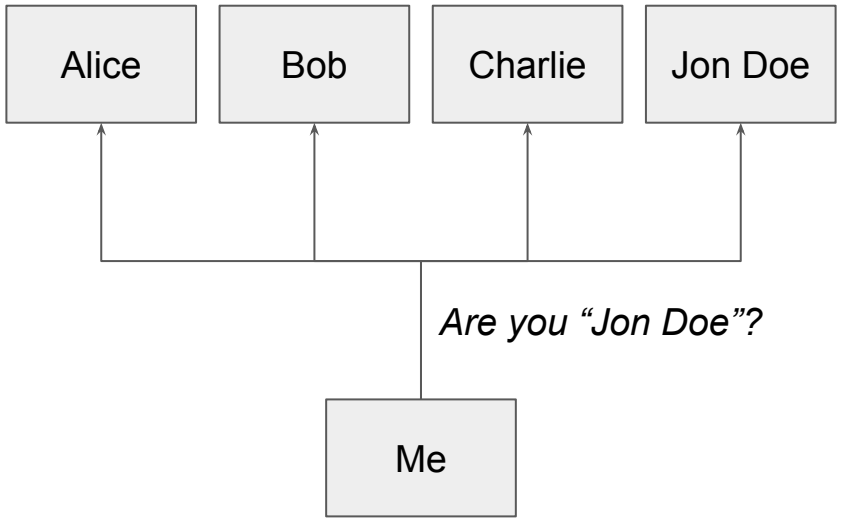


Name Discovery

- DNS requires knowing the address of the root servers
- Also requires names to be centrally registered and globally available
- Useful, but not for every scenario
  - Hosts on an LAN
    - Typically router provides
      - Dynamic Host Configuration Protocol (DHCP) server
      - Local DNS server
    - What about unmanaged LANs?
  - IOT devices
    - “Just work” without any manual configuration
    - I.e., automatically discovered
  - In both cases, names should be local to their LAN and available after joining the LAN
    - DNS is global and takes some time to react to changes (e.g., up to 24h)

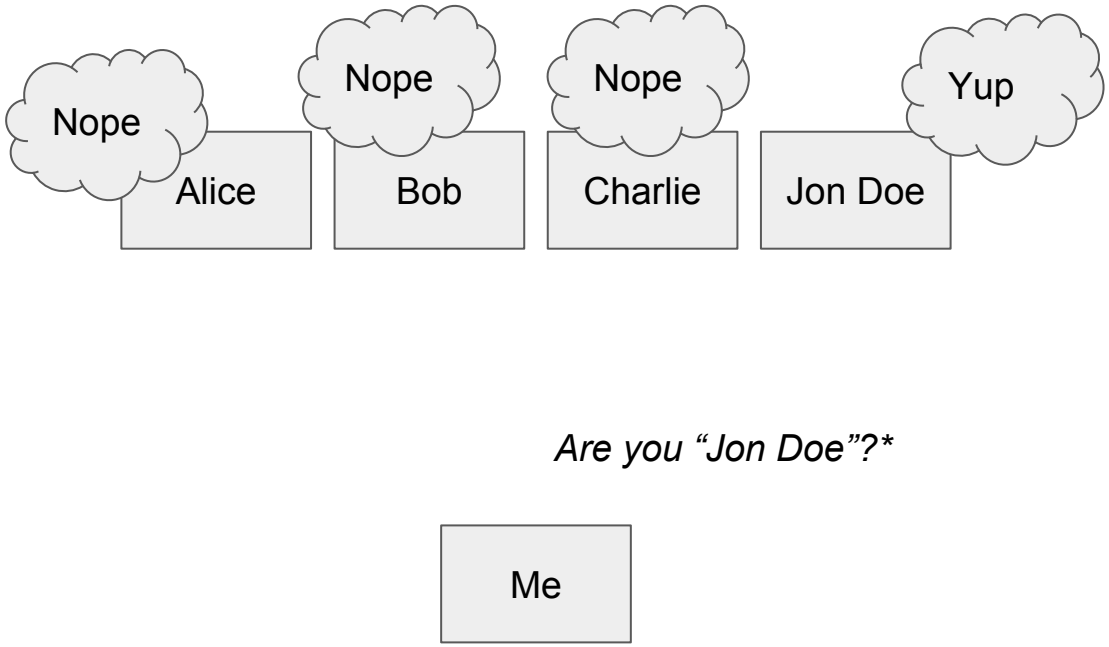
Name Discovery - Broadcasting

- Broadcast the ID, requesting the entity to answer with its address
  - “Is Jon Doe here?”



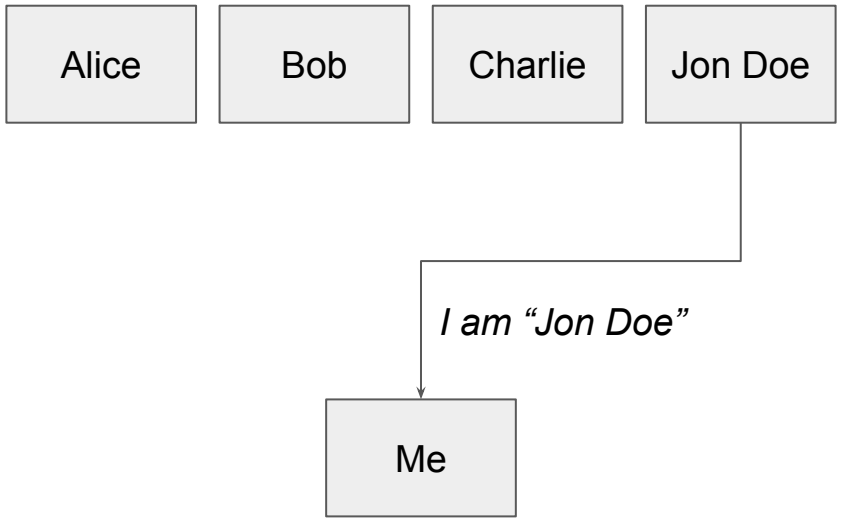
Name Discovery - Broadcasting

- Broadcast the ID, requesting the entity to answer with its address
  - “Is Jon Doe here?”



Name Discovery - Broadcasting

- Broadcast the ID, requesting the entity to answer with its address
  - “Is Jon Doe here?”



Name Discovery - Broadcasting

- Broadcast the ID, requesting the entity to answer with its address
  - “Is Jon Doe here?”
- Simple
- Requires all entities to listen to incoming requests
- Not very scalable, network traffic wasted by name requests
  - But LAN is relatively small, so that’s OK

Name Discovery - Broadcasting

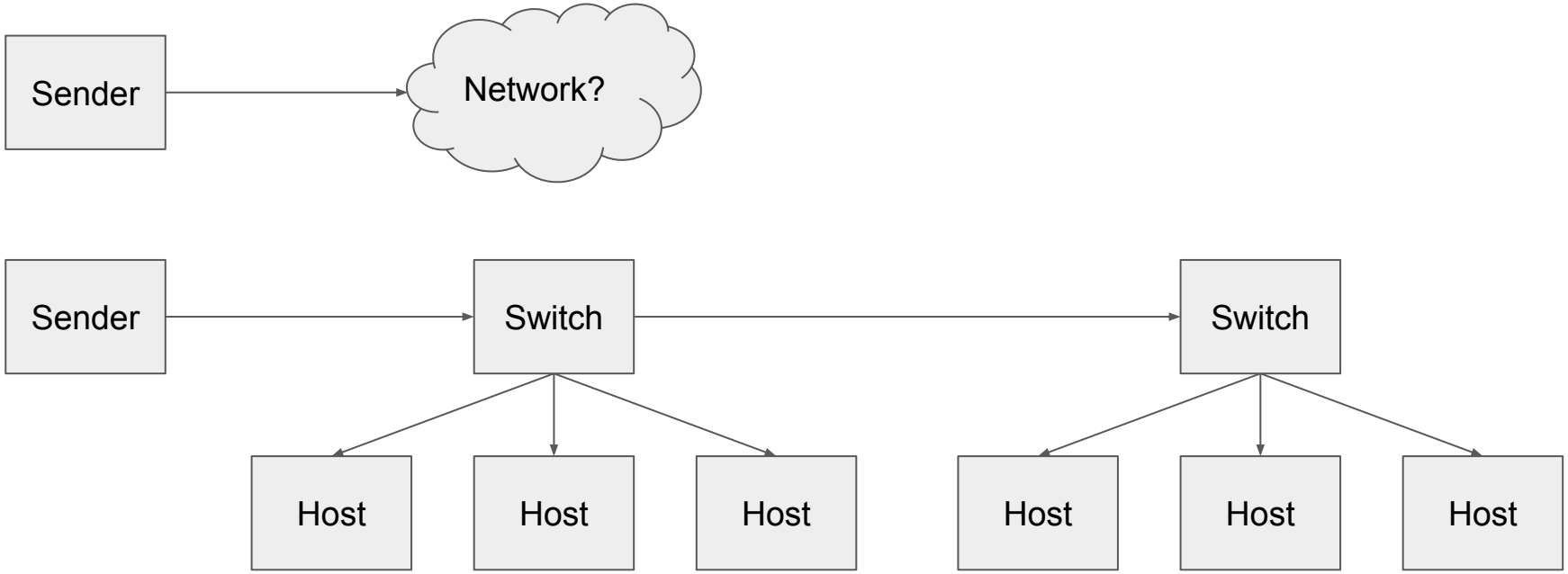
- Unmanaged networks (i.e., no router)
  - The **Address Resolution Protocol (ARP)**, used to map IP addresses (e.g., 127.0.0.1) to MAC addresses (e.g., 00:00:00:00:00:00), uses this approach
- Managed networks
  - The **Dynamic Host Configuration Protocol (DHCP)** allows a router to assign IP addresses to all hosts
    - And set up the local name server
  - Newly connected host broadcast a **DCHP discover** message on the network to discover if there's any DHCP server
  - DHCP server replies, and host now knows its IP and how to use DNS
- Service discovery (iTunes, etc.) does not require a router

Broadcast - Multicast

- Unicast connections are one-to-one (or point-to-point)
  - One sender, one receiver
  - E.g., HTTP traffic, server sends pages to the clients
- Multicast connections are **one-to-many**
  - One sender, many receivers
  - E.g., Address Resolution Protocol (ARP) shown earlier

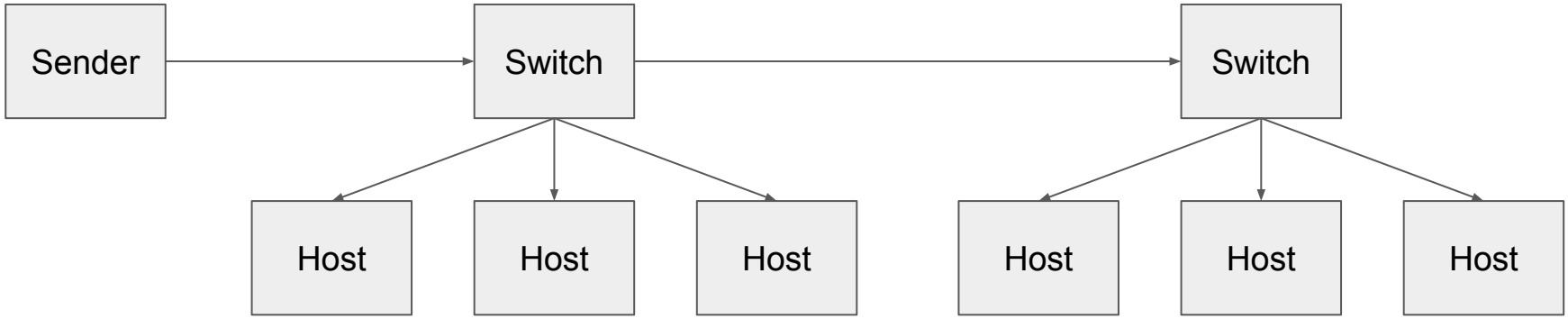
Broadcast - Multicast

- Multicast does not require sender knowing the receivers
  - Unicast requires sender to establish a connection to the receiver



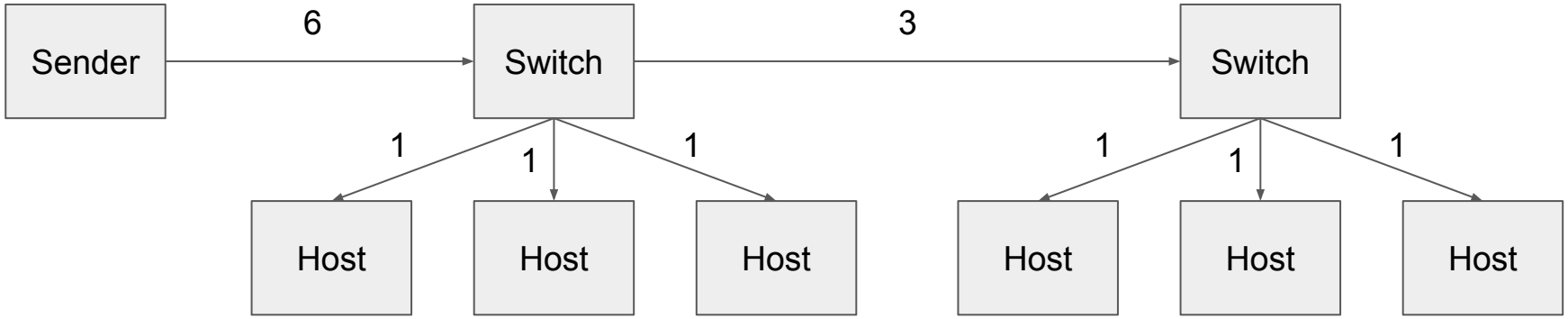
Multicast

- Multicast increases the efficiency of networks
  - Point-to-point broadcast requires the sender to send N copies of the message
  - Multicast broadcast only sends one copy
  - Network switches replicate the traffic faster and more efficiently



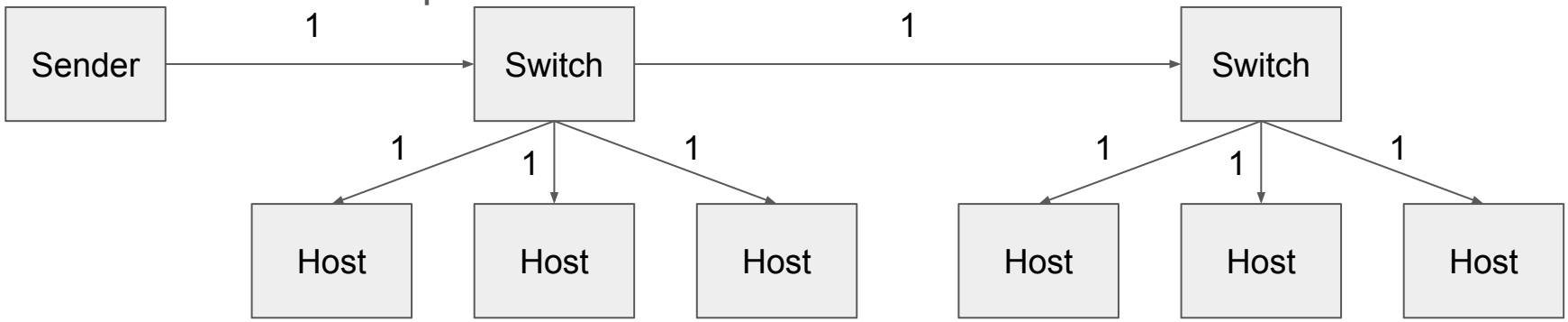
Multicast

- Multicast increases the efficiency of networks
  - Point-to-point broadcast requires the sender to send N copies of the message
  - Multicast broadcast only sends one copy
  - Network switches replicate the traffic faster and more efficiently
  - Unicast: 15 copies



Multicast

- Multicast increases the efficiency of networks
  - Point-to-point broadcast requires the sender to send N copies of the message
  - Multicast broadcast only sends one copy
  - Network switches replicate the traffic faster and more efficiently
  - Unicast: 15 copies
  - Multicast: 8 copies



Multicast on IPv4

- Special multicast addresses
  - Leading bits are `1110`
  - From `224.0.0.0` to `239.255.255.255`
- Local subnetwork: `224.0.0.0` to `224.0.0.255`
  - **Not routable**: Internet routers will discard them
  - Router should not forward these to the internet
- Several classes of multicast addresses
  - Most not used
- Well known multicast addresses
  - `224.0.1.1`: Network Time Protocol (NTP) - Routable!
  - `224.0.0.251`: Multicast DNS (mDNS)
  - `255.255.255.255`: DHCP Discover
- Multicast on IPv6 is similar

Multicast in practice - Multicast DNS (mDNS)

- Resolve DNS records without any DNS server
- All hosts on a local network listen to IPv4 address `224.0.0.251` on port `5353`
  - Address `FF02::FB` for IPv6
- DNS query sent to this address is broadcast among all hosts
  - Host with the correct name answers the query

DNS Service Discovery

- DNS Service Discovery (DNS-SD)
  - Service discovery based on mDNS
- Discover a service on the network (e.g., scanner)
  - Send a query for a PTR Record for the domain:
    `<service>.<transport>.local`
    - E.g., `_scanner._tcp.local`
  - Receive list of PTR RRs if service exists on the network
    - `<instance>.<service>.<transport>.local`
    - E.g., `Deskjet\0323520\032series\032[5C41EC]._scanner._tcp.local`
- `dig @224.0.0.251 -p 5353 _scanner._tcp.local ptr`

mDSN - Example with dig

> dig @224.0.0.251 -p 5353 _scanner._tcp.local ptr
; <<>> DiG 9.11.2 <<>> @224.0.0.251 -p 5353 _scanner._tcp.local ptr
; (1 server found)
;; global options: +cmd
;; Got answer:
;; WARNING: .local is reserved for Multicast DNS
;; You are currently testing what happens when an mDNS query is leaked to DNS
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10754
;; flags: qr aa; QUERY: 1, ANSWER: 1

;; QUESTION SECTION:
;_scanner._tcp.local. IN PTR

;; ANSWER SECTION:
_scanner._tcp.local. 10 IN PT ocal.

;; ADDITIONAL SECTION:
Deskjet\0323520\032series\032[5C41 EC.local.
Deskjet\0323520\032series\032[5C41 20 series"
"mfg=HP" "mdl=Deskjet 3520 series"
"UUID=1c852a4d-b800-1f08-abcd-28
HP28924A5C41EC.local. 10 IN
HP28924A5C41EC.local. 10 IN

;; Query time: 185 msec
;; SERVER: 192.168.1.160#5353(224.0.0.251)
;; WHEN: Thu Feb 22 11:23:58 EST 2018
;; MSG SIZE rcvd: 340

mDSN - Example with dig

> dig @224.0.0.251 -p 5353 _scanner._tcp.local ptr

```
; <<>> DiG 9.11.2 <<>> @224.0.0.251 -p 5353 _scanner._tcp.local ptr
; (1 server found)
;; global options: +cmd
;; Got answer:
;; WARNING: .local is reserved for Multicast DNS
;; You are currently testing what happens when an mDNS query is leaked to DNS
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10754
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 4

;; QUESTION SECTION:
;_scanner._tcp.local.          IN      PTR

;; ANSWER SECTION:
_scanner._tcp.local.    10      IN      PTR     Deskjet\0323520\032series\032[5C41EC]._scanner._tcp.local.

;; ADDITIONAL SECTION:
Deskjet\0323520\032series\032[5C41EC]._scanner._tcp.local. 10 IN SRV 0 0 8080 HP28924A5C41EC.local.
Deskjet\0323520\032series\032[5C41EC]._scanner._tcp.local. 10 IN TXT "txtvers=1" "ty=Deskjet 3520 series"
"mfg=HP" "mdl=Deskjet 3520 series" "adminurl=http://HP28924A5C41EC.local." "note="
"UUID=1c852a4d-b800-1f08-abcd-28924a5c41ec" "button=T" "flatbed=T"
HP28924A5C41EC.local.   10      IN      A       192.168.1.160
HP28924A5C41EC.local.   10      IN      AAAA    fe80::2a92:4aff:fe5c:41ec

;; Query time: 185 msec
;; SERVER: 192.168.1.160#5353(224.0.0.251)
;; WHEN: Thu Feb 22 11:23:58 EST 2018
;; MSG SIZE  rcvd: 340
```

DNS-SD in practice - Zeroconf

- Modern Operating Systems all have a zeroconf daemon
  - Apple: *Bonjour* protocol
    - mDNSResponder released as open source, used by Android
  - Microsoft:
    - *Netbios* (not mDNS)
      - Until Windows XP (at least?)
    - *Link-Local Multicast Name Resolution* (LLMNR)
      - From Windows Vista
  - GNU/Linux
    - *Avahi* service
- Building block of modern IOT devices

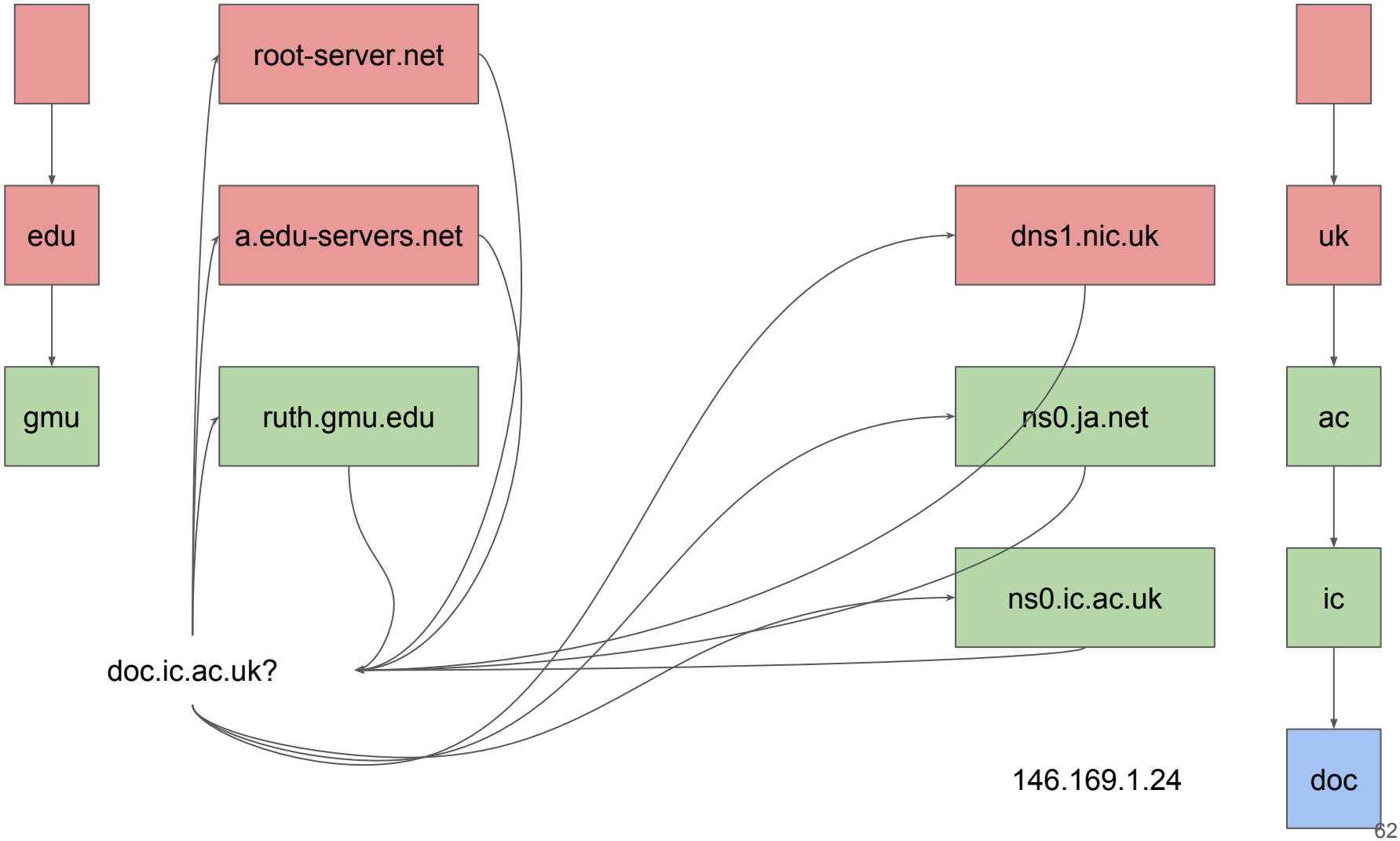
Conclusion

- Resolving names requires a large scale distributed system
  - Domain Name System (DNS)
  - Distributed between several entities and among all continents
  - World-wide scale
  - High availability
- Sometimes we need local names
  - Peer-to-peer protocol based on DNS and on local network multicast
  - mDNS: Each host is responsible for its own records
  - Distributed, small scale, high availability zeroconf services
    - Bonjour, Avahi, mDNS responder, LLMNR
    - Building block of IOT

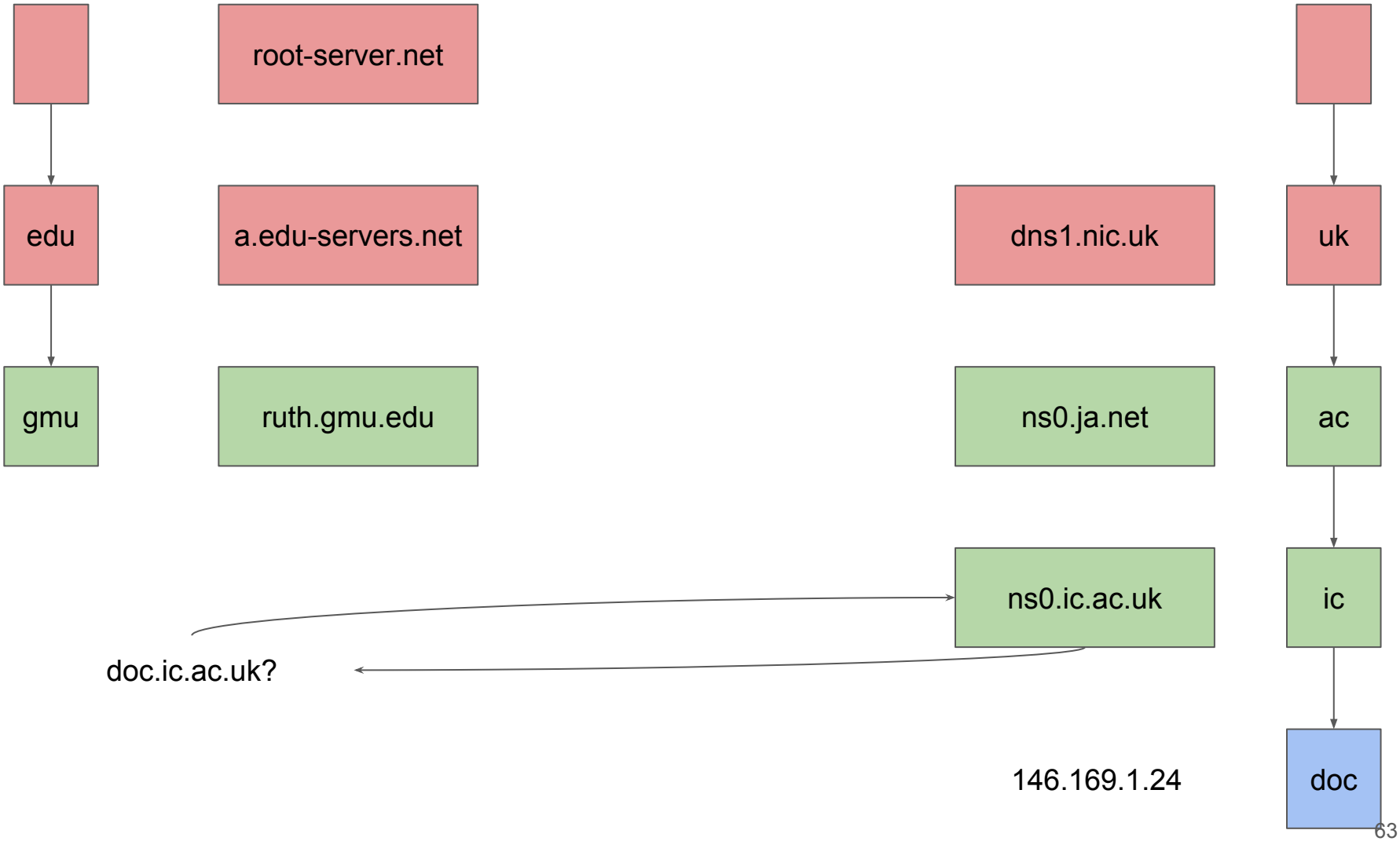
Questions?
Comments?

Contact me:
Luís Pina
lpina2@gmu.edu

Domain Name System - Iterative Resolution



Domain Name System - Iterative Resolution (caching)



Outline

- What is a: name, address, access point, pure name, identifier
- Flat names (pure names)
  - Broadcast, foward pointers, home-base
  - DHT, chord, expand example of finding an address, discuss how to add/remove nodes
  - Hierarchical Location Services (HLS), scaling
- Structured names
  - Name space: leaf node, directory node
  - UNIX paths: absolute/relative names, linking, mounting, example with SSHFS instead of NFS
    - Distinction between paths and inode numbers, they are the same thing but inode numbers are pure names
  - DNS: layers, scaling, caching
    - Anecdote about Steam cheat detection by querying OS DNS cache for known cheat sites

Concepts

Name resolution: Map a name (or an identifier) to an address

Name: Sequence of bits/characters that is used to refer to an entity

Entity: *Anything* (node in a network, memory location, file on disk, etc.)

Address: Name of an access point (`*name = 0xDEADBEEF`, what about `&name`?)

Access Point: Special entity that can operate on another entity

A telephone is the **access point** of a person, a telephone

Concepts

Name resolution: Map a name (or an identifier) to an address

Identifier: Unique name for an entity, with the following rules:

1. An identifier refers to at most one entity
2. Each entity is referred to by at most one identifier
3. An identifier always refers to the same entity (i.e., is never reused)

Which are identifiers? Telephone number, URL, credit card number, social security number, file path

Concepts

Name resolution: Map a name (or an identifier) to an address

Pure (flat) name: The contents of the name do not contain any information about how to locate the access point of the named entity

Structured name: The contents of the name ~~do not~~ contain any information about how to locate the access point of the named entity

Which are pure? Telephone number, URL, credit card number, social security number, file path

Concepts

Name resolution: Map a name (or an identifier) to an address

What is the most straightforward/simple/direct implementation of name resolution?

A table from names to addresses!

Flat naming - Forwarding Pointers

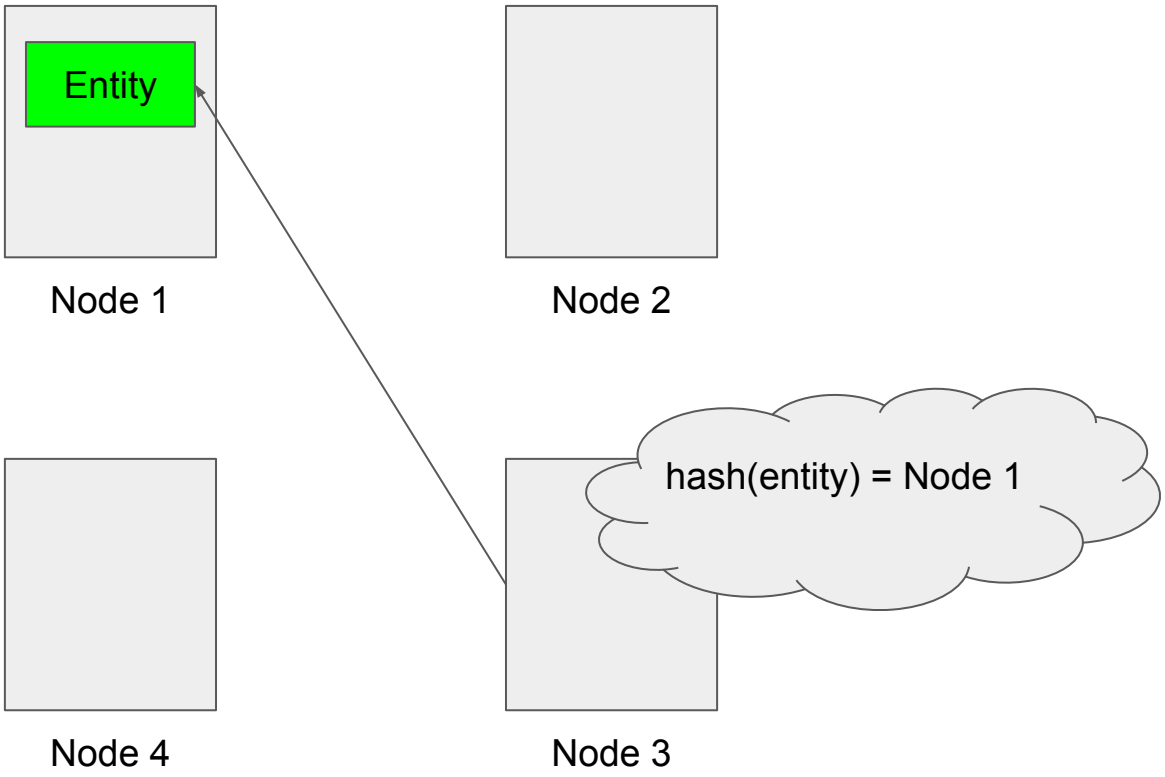
How to resolve flat (pure) names?

- You are designing a system with a number of nodes
- Each node keeps a collection of entities (e.g., objects)
- Each entity is only in one node at each time
- You can find out the first node that stored the entity (e.g., hash)
- Optimization: Move entities to the nodes that are using them

- How can you find an entity?

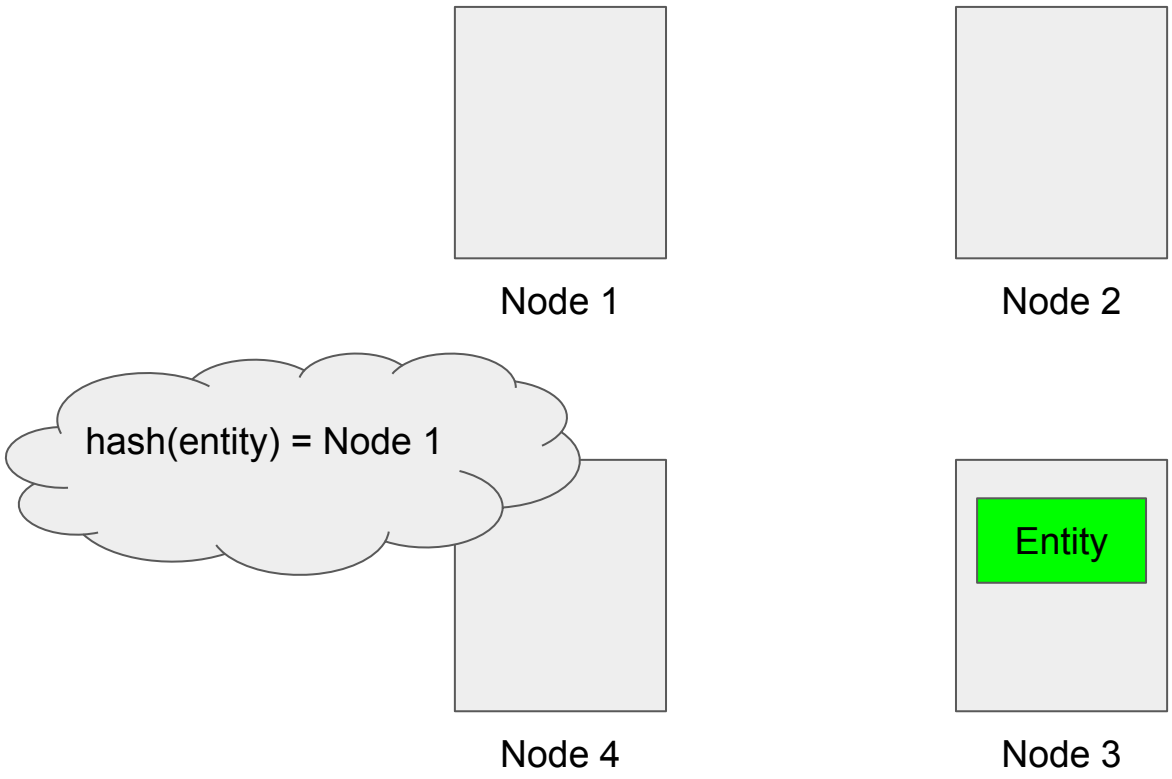
Flat naming - Forwarding Pointers

How to resolve flat (pure) names?



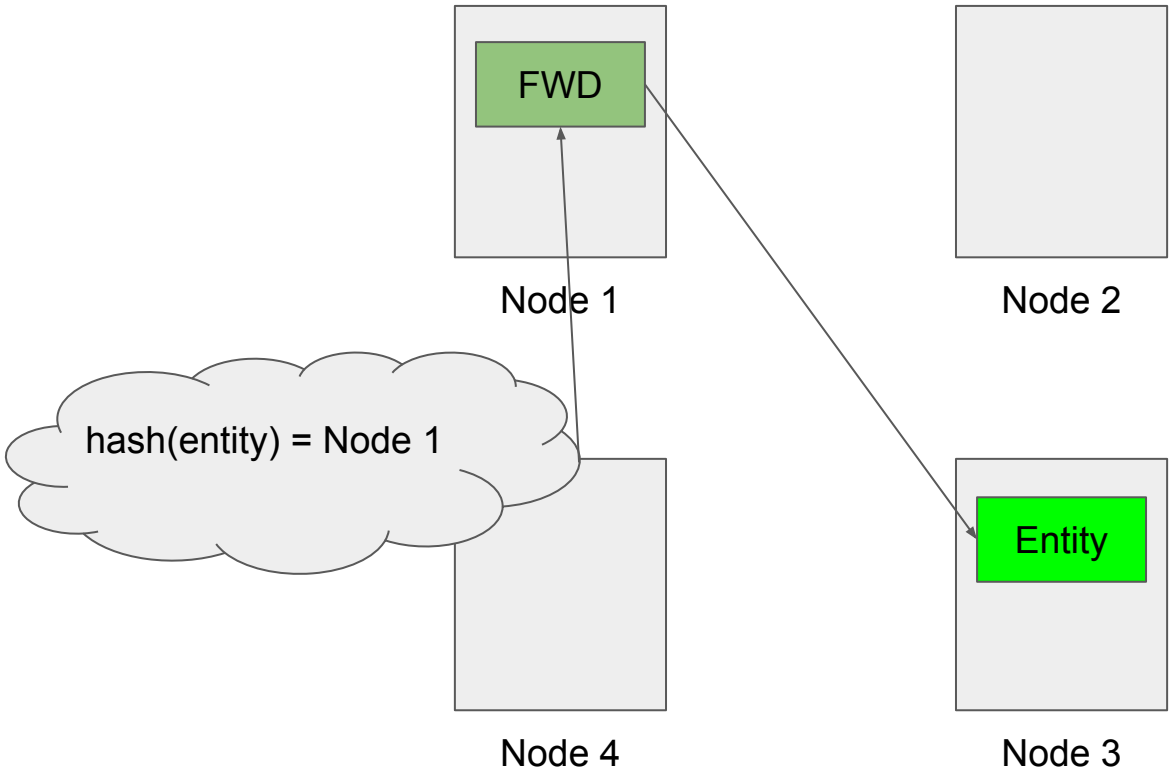
Flat naming - Forwarding Pointers

How to resolve flat (pure) names?



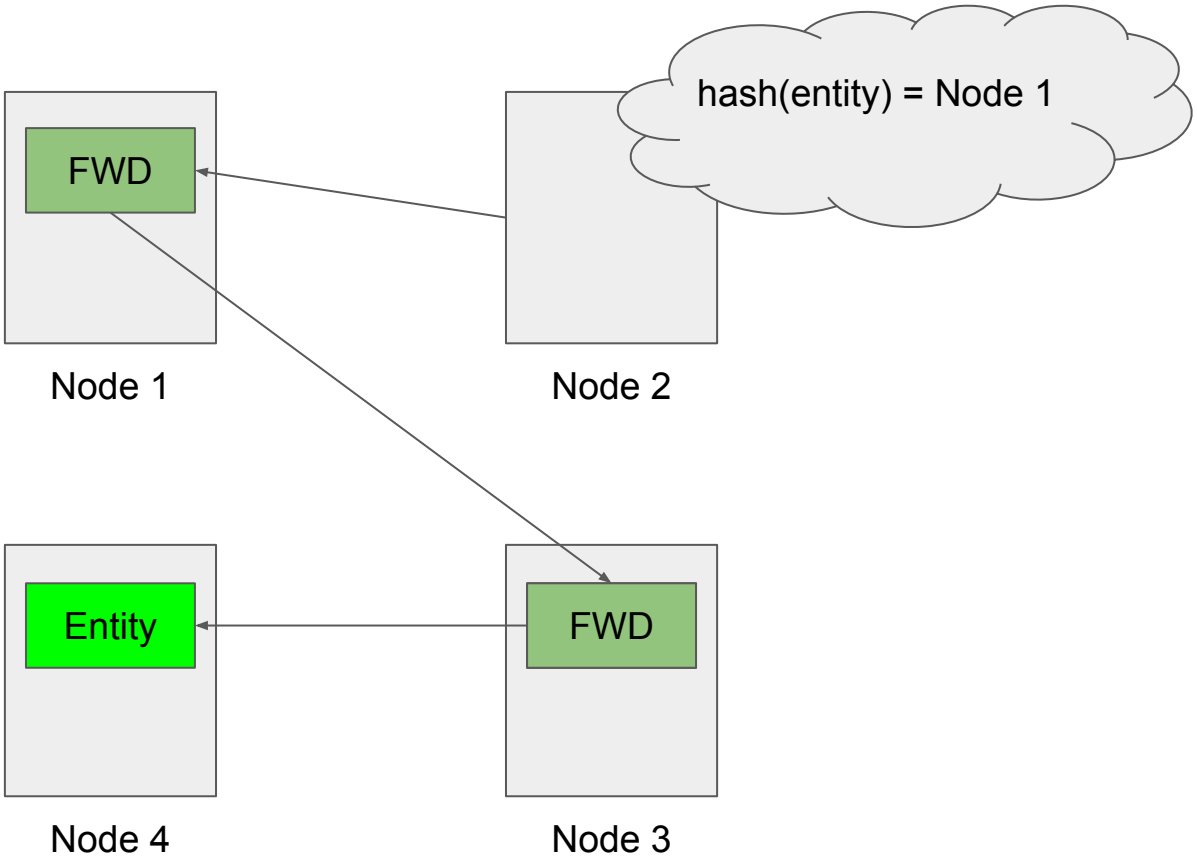
Flat naming - Forwarding Pointers

How to resolve flat (pure) names?



Flat naming - Forwarding Pointers

How to resolve flat (pure) names?



Flat naming - Forwarding Pointers

How to resolve flat (pure) names?

- When an entity moves, it leaves behind a pointer to its next location
- Simple: Just follow chain of pointers to find an entity
- Long chain can be too expensive to follow
- All nodes need to keep resources (e.g., memory) to keep the chain
- Not fault tolerant: If a node becomes unavailable, chain is broken

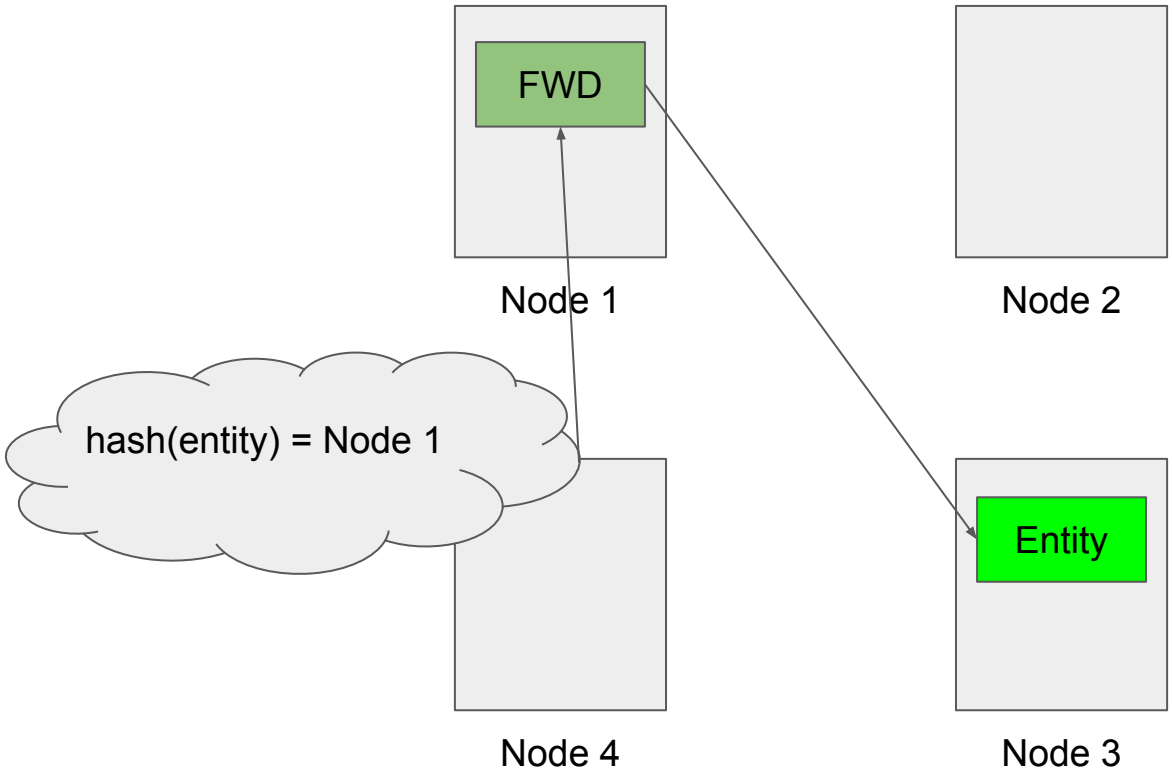
Flat naming - Forwarding Pointers

How to resolve flat (pure) names?

- How to keep chains short?
- Idea: Shorten chain at every access

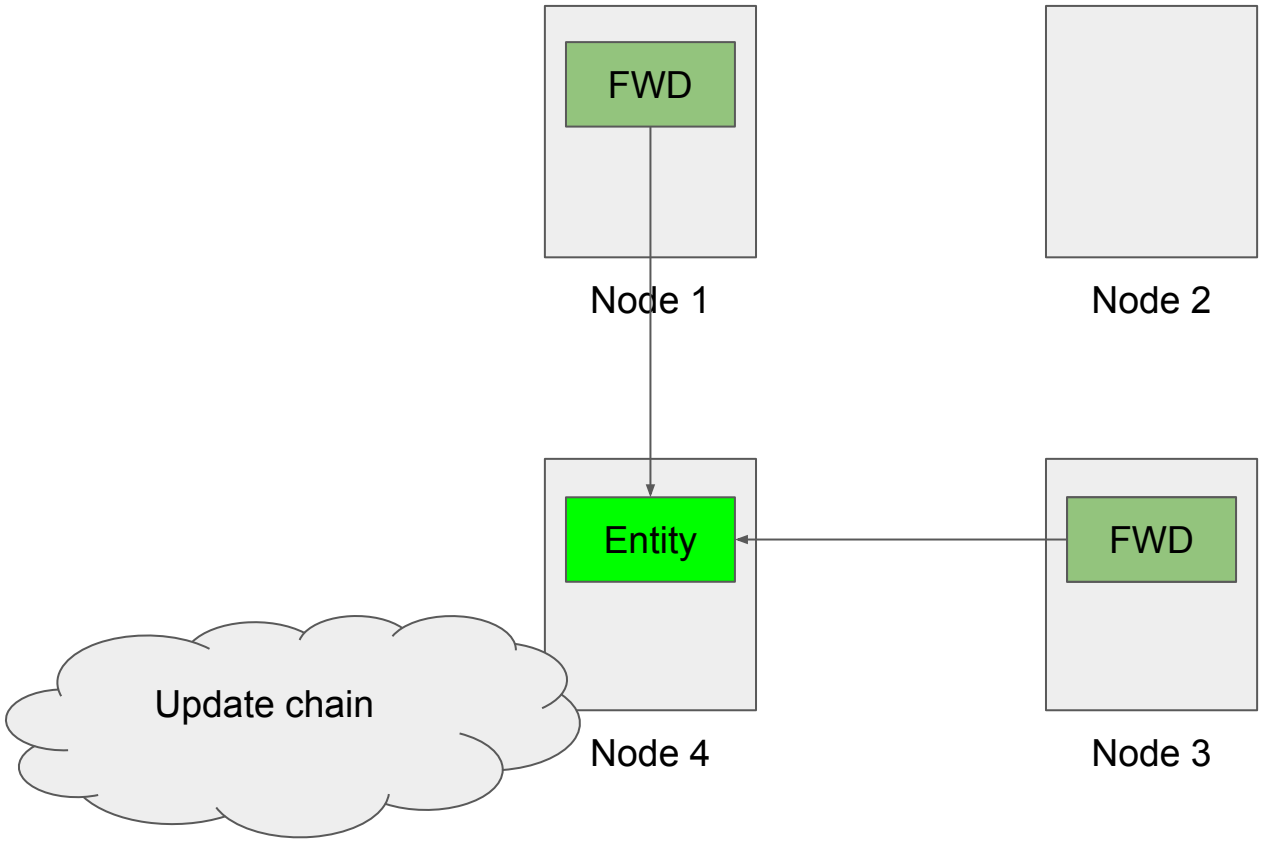
Flat naming - Forwarding Pointers

How to resolve flat (pure) names?



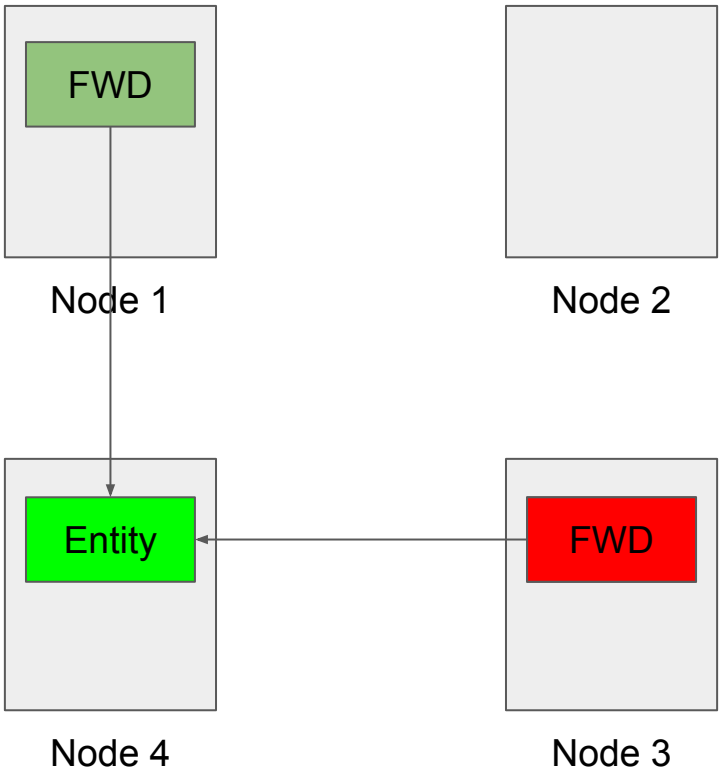
Flat naming - Forwarding Pointers

How to resolve flat (pure) names?



Flat naming - Forwarding Pointers

How to resolve flat (pure) names?



Garbage!

But Node 3 doesn't know if other nodes point to it

Distributed Garbage Collection is hard

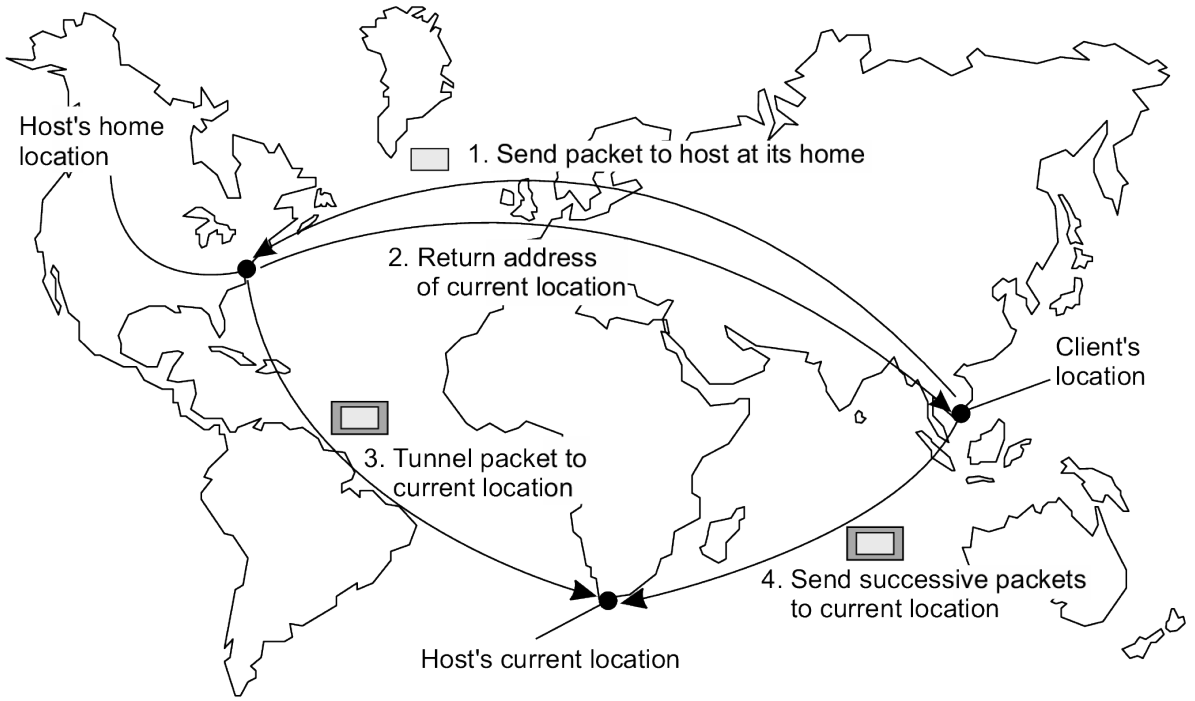
Flat naming - Home-base approach

How to resolve flat (pure) names?

- Each entity always has a home location which keeps track of its address
- Example: Mobile IP
- Each mobile host as same IP address
- Home agent receives all traffic for that IP and tracks host
  - If on the same LAN, forward
  - If not, tunnel IP packet to host's current real IP
  - Fully transparent to applications
  - Increase in communication latency
  - Home location must always be available

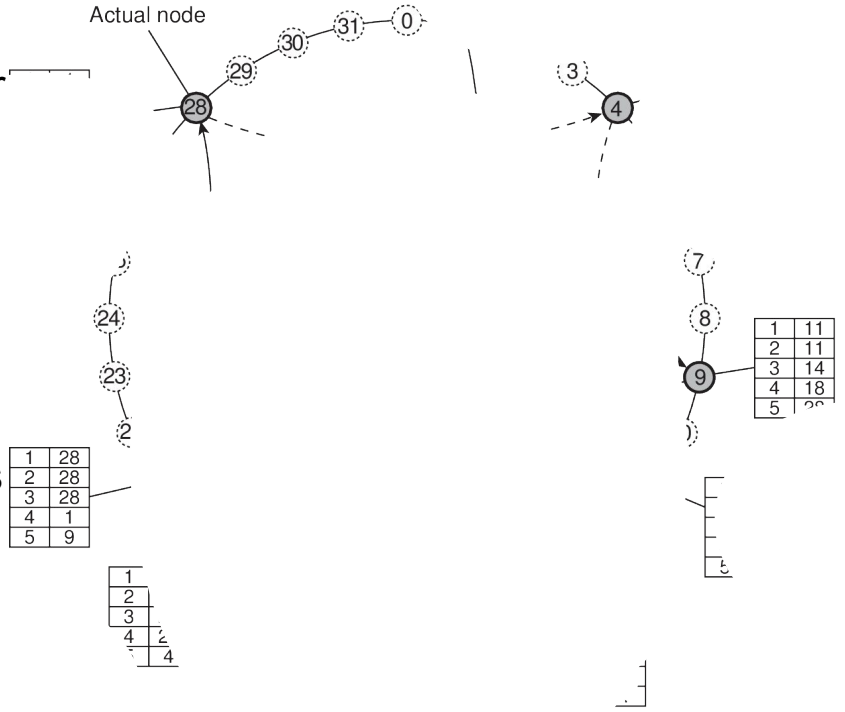
Flat naming - Home-base approach

How to resolve flat (pure) names?



Flat naming - Distributed Hash Table (Chord)

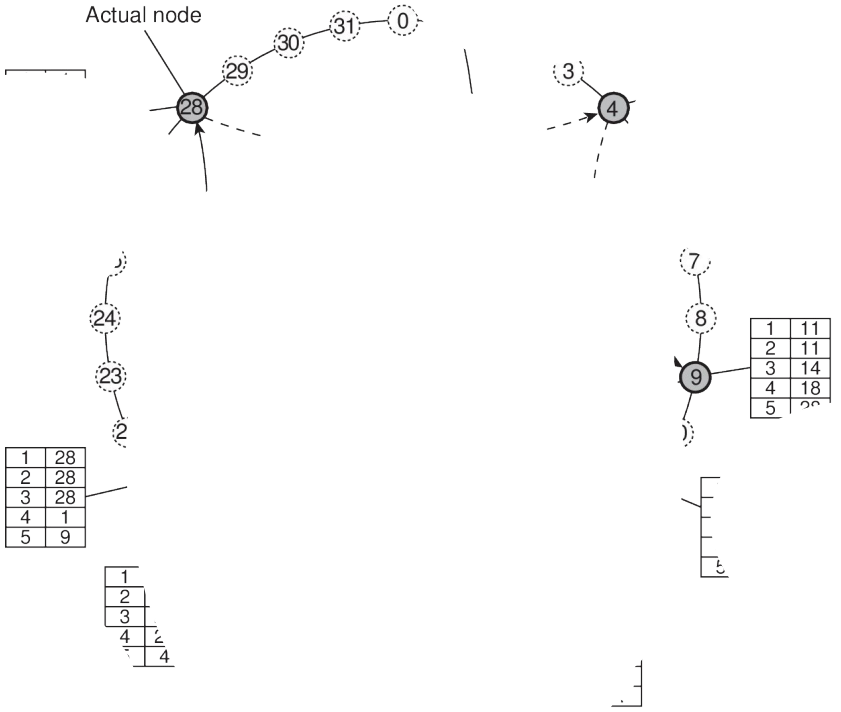
- *m*-bit identifier space
  - Each node has a random identifier
  - Each entity has an *m*-bit key: *k*
  - Entity k belongs to: *min(id) > k*
- Obvious approach: linear scan
  - Each node knows its successor and predecessor
  - To find key k, send it right or left
  - Lookup in O(*N*) steps, for *N* nodes
- Can we do better?



Flat naming - Distributed Hash Table (Chord)

- Finger tables
  - Contains $s \leq m$ entries
  - $FT_p[i] = succ(p + 2^{i-1})$
- Node *i* is the first node succeeding *p* by at least

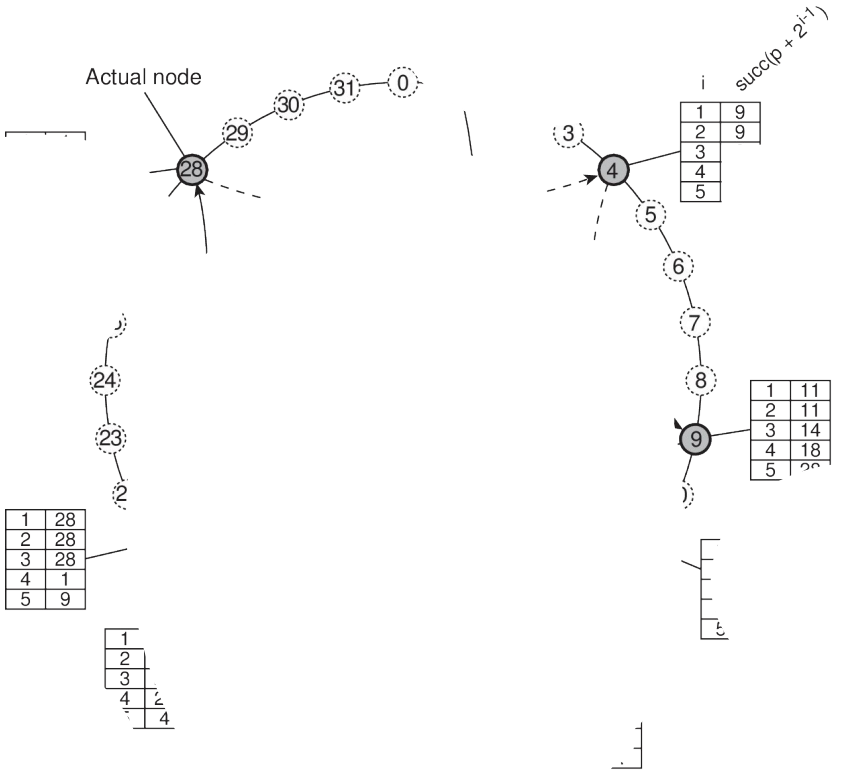
$$2^{i-1}$$



Flat naming - Distributed Hash Table (Chord)

- Finger tables
  - Contains $s \leq m$ entries
  - $FT_p[i] = succ(p + 2^{i-1})$
- Node *i* is the first node succeeding *p* by at least

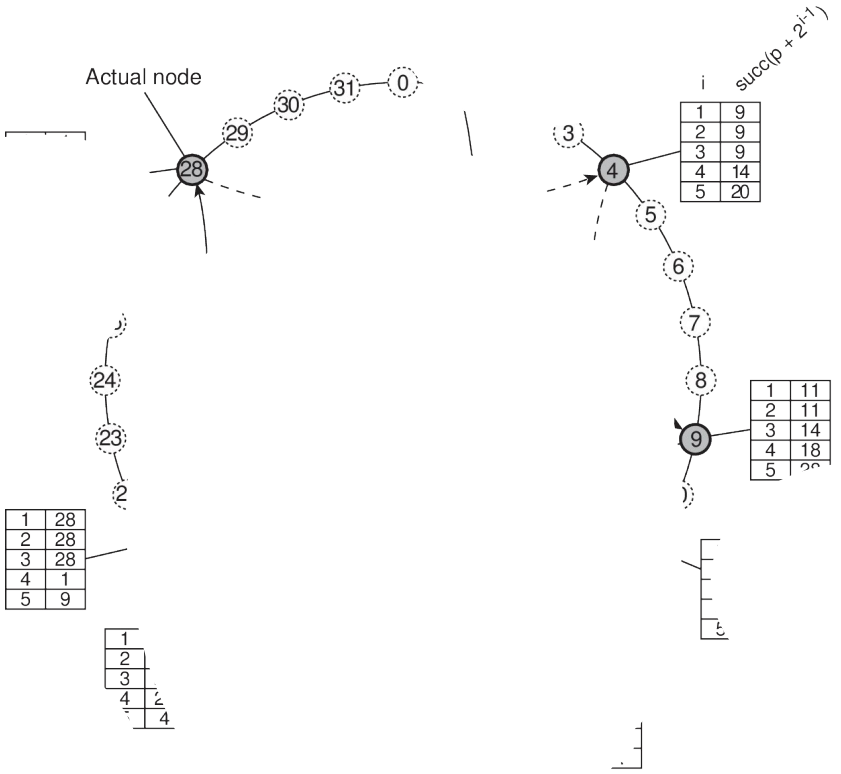
$$2^{i-1}$$



Flat naming - Distributed Hash Table (Chord)

- Finger tables
  - Contains $s \leq m$ entries
  - $FT_p[i] = succ(p + 2^{i-1})$
- Node *i* is the first node succeeding *p* by at least

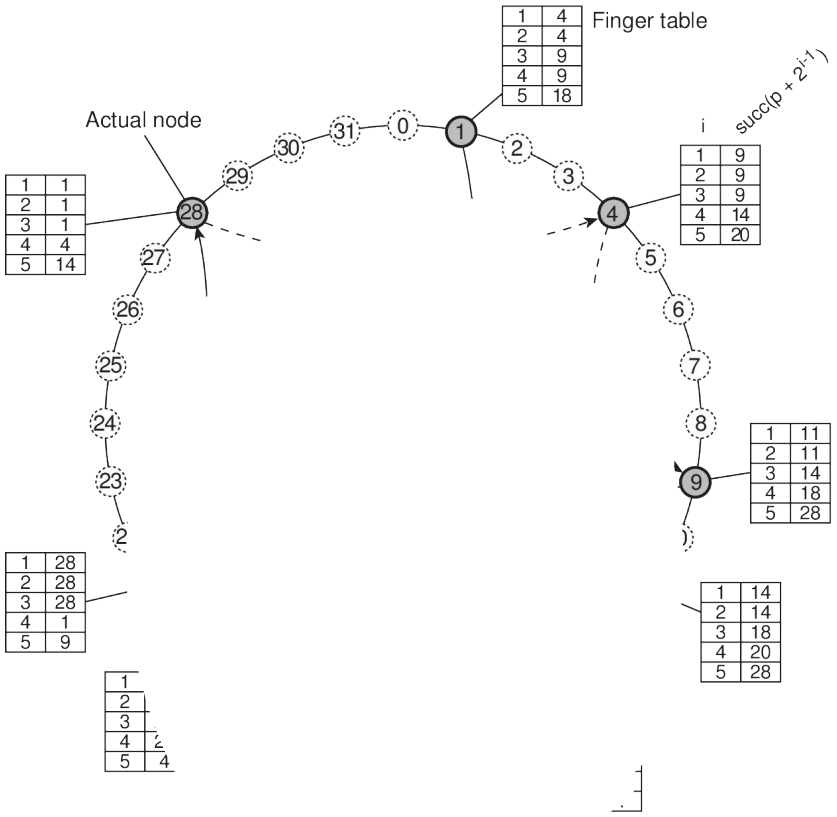
$$2^{i-1}$$



Flat naming - Distributed Hash Table (Chord)

- Finger tables
  - Contains $s \leq m$ entries
  - $FT_p[i] = succ(p + 2^{i-1})$
- Node *i* is the first node succeeding *p* by at least

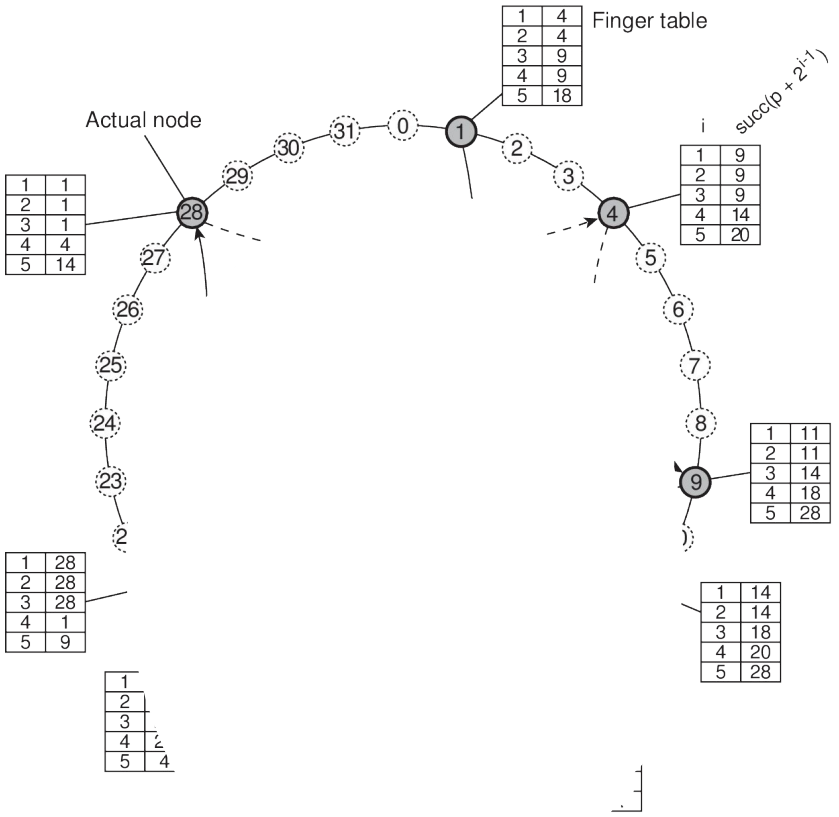
$$2^{i-1}$$



Flat naming - Distributed Hash Table (Chord)

- Finger tables
  - Contains $s \leq m$ entries
  - $FT_p[i] = succ(p + 2^{i-1})$
- Node *i* is the first node succeeding *p* by at least
- Lookup: 2^{i-1}

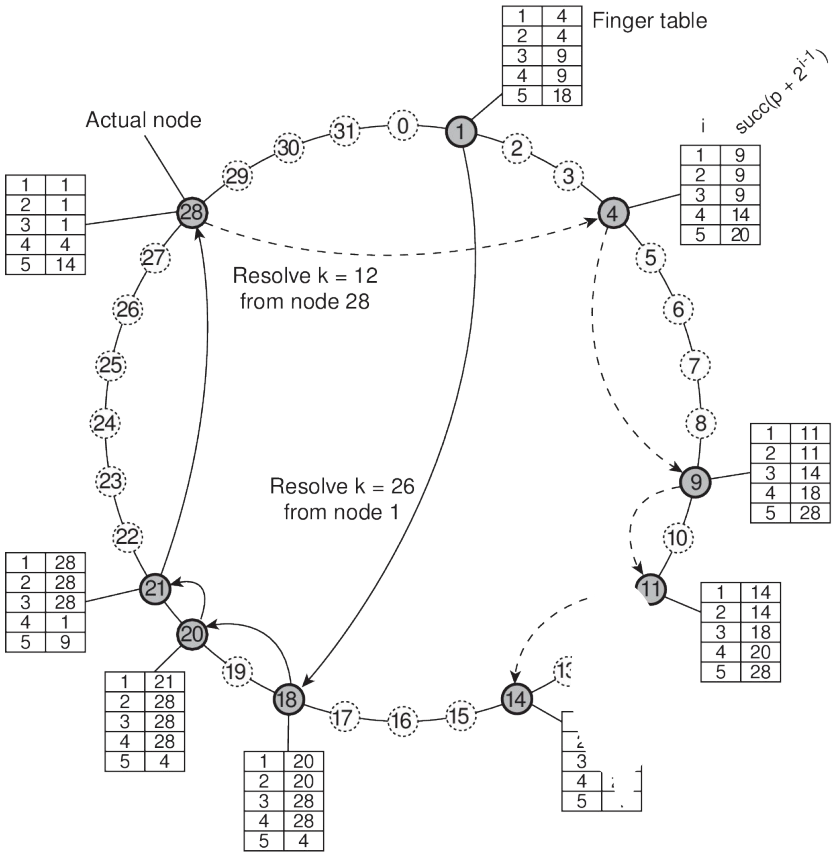
$$q = FT_p[j] \leq k < FT_p[j+1]$$



Flat naming - Distributed Hash Table (Chord)

- Finger tables
  - Contains $s \leq m$ entries
  - $FT_p[i] = succ(p + 2^{i-1})$
- Node *i* is the first node succeeding *p* by at least
- Lookup: 2^{i-1}

$$q = FT_p[j] \leq k < FT_p[j+1]$$



Flat naming - Distributed Hash Table (Chord)

- Finger tables
  - Contains $s \leq m$ entries
  - $FT_p[i] = succ(p + 2^{i-1})$
- Node *i* is the first node succeeding *p* by at least
- Lookup: 2^{i-1}
- Lookup in: $q = FT_p[j] \leq k < FT_p[j+1]$

$$O(log(N))$$