Content Delivery Networks

- Replication is a huge benefit to availability, scalability, and performance
  - Can spread the load
  - Places content closer to clients (less latency)

  - Caching is a form of opportunistic replication
    - .. but what if a given organization doesn't have a forward proxy?
    - .. what if content provider and wants its content always replicated?

    - Idea: Caching and replication as a service — "CDNs 1.0"

CDNs “1.0”

- Large-scale distributed storage infrastructure
  - (Usually) administered by one entity
  - e.g., Akamai has 275,000+ servers in 136 countries
- Any server can host content for the many clients of the CDN (virtual hosting)
- How does content provider get its data onto Akamai’s servers?
- Two major ways
  - Pull
  - Push
  - .. we’ll come back to these in a moment

CDNs "1.0" - the basic idea

- Content provider buys service from a CDN, e.g., Akamai

- CDN creates new domain names for the customer content provider
  - e.g., e12596.dscj.akamaiedge.net for cnn.com
  - The CDN's DNS servers are authoritative for the new domains

- Content provider modifies its content so that embedded URLs reference the new domains
  - "Akamaize" content
  - e.g.: http://www.cnn.com/some-photo.jpg becomes http://e12596.dscj.akamaiedge.net/some-photo.jpg

- Initial request goes to CNN (e.g., for main http://www.cnn.com page)
  - .. but embedded links go to Akamai, which handles DNS resolution for URL
  - .. Akamai DNS servers pick one of their 275,000+ servers to serve it
  - (based on IP geolocation, server load, etc.)

CDNs “1.0” - the basic idea

- Content provider buys service from a CDN, e.g., Akamai

- CDN creates new domain names for the customer content provider
  - e.g., e12596.dscj.akamaiedge.net for cnn.com
  - The CDN’s DNS servers are authoritative for the new domains

- Content provider modifies its content so that embedded URLs reference the new domains
  - “Akamaize” content
  - e.g.: http://www.cnn.com/some-photo.jpg becomes http://e12596.dscj.akamaiedge.net/some-photo.jpg

- Initial request goes to CNN (e.g., for main http://www.cnn.com page)
  - .. but embedded links go to Akamai, which handles DNS resolution for URL
  - .. Akamai DNS servers pick one of their 275,000+ servers to serve it
  - (based on IP geolocation, server load, etc.)

How do you get content onto the CDN servers?

- Pull
  - Akamai servers act like a cache
  - Content provider gives CDN “origin” URL
  - When a client requests from Akamai
    - .. if cached, serve it
    - .. if not cached, request (“pull”) from origin, cache it, serve it

- Push
  - Akamai servers just act like normal servers
  - Content provider uploads content to CDN (“pushes” their content)
  - When a client requests from Akamai, just serve like any web server

- Various tradeoffs
  - Short version: pull is less work for content provider but push gives more control

DNS

How do you resolve a name into an IP?

In olden times (1980s)

- all host to address mappings were in a file called hosts.txt
- in /etc/hosts
- Had to download regularly
- *still useful for certain situations. /etc/hosts takes precedence
    - https://raw.githubusercontent.com/StevenBlack/hosts/master/hosts

Problem:

- Scalability in terms of
    - Management
    - Availability
    - Consistency

How do you resolve a name into an IP?

In olden times (1980s)

- all host to address mappings were in a file called hosts.txt
- in /etc/hosts
- *still usef precedence
    - h ts/master/hosts

Problem:

- Scalabilit
    - Manag
    - Availability
    - Consistency

What do you do when you need scalability?

How do you resolve a name into an IP?

In olden times (1980s)

- all host to address mappings were in a file called hosts.txt
- in /etc/hosts
- *still usef precedence
    - h ts/master/hosts

Problem:

- Scalabilit
    - Manag
    - Availability
    - Consistency

What do you do when you need scalability?
a hierarchical structure

To scale, DNS adopt three intertwined hierarchies

naming structure	hierarchy of addresses https://ee.hawaii.edu/home/
Management	hierarchy of authority over names
Infrastructure	hierarchy of DNS servers

To scale, DNS adopt three intertwined hierarchies

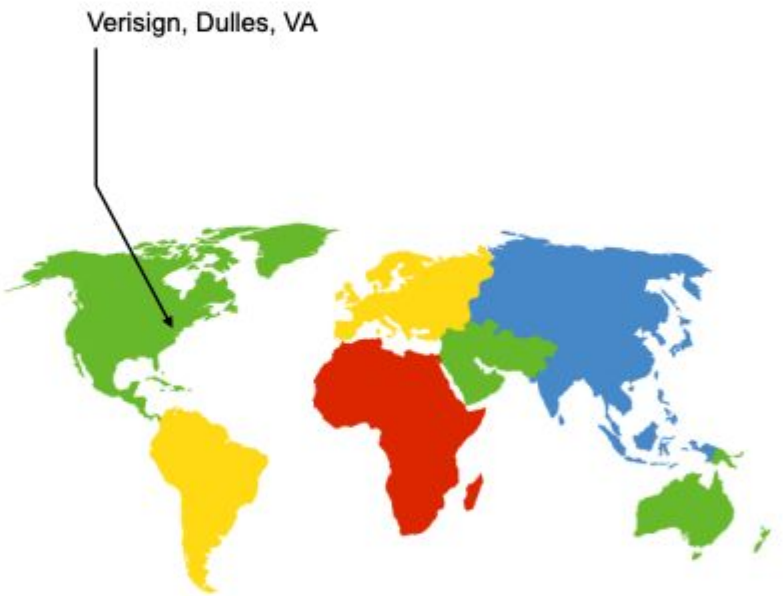
naming structure	hierarchy of addresses https://ee.hawaii.edu/home/
Management	hierarchy of authority over names
Infrastructure	hierarchy of DNS servers

DNS root

Located in Virginia, USA

Every server knows the address of root servers - needed for bootstrap https://www.internic.net/domain/named.root

How do we make the root scale?



DNS root

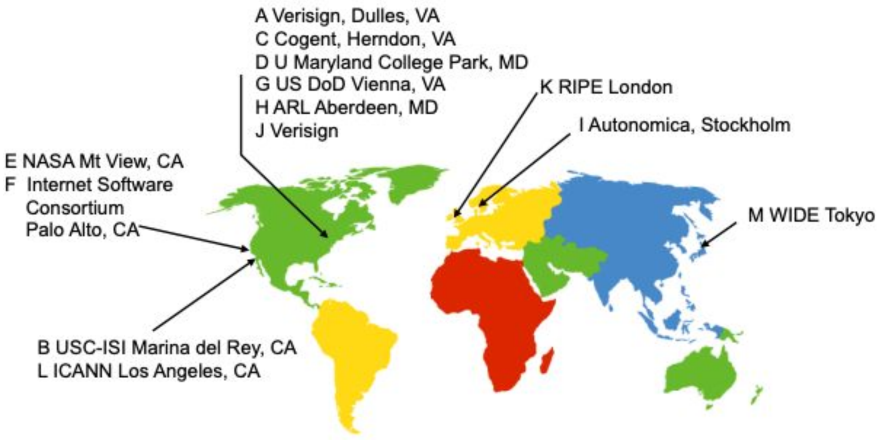
13 root servers (see http://www.root-servers.org/)

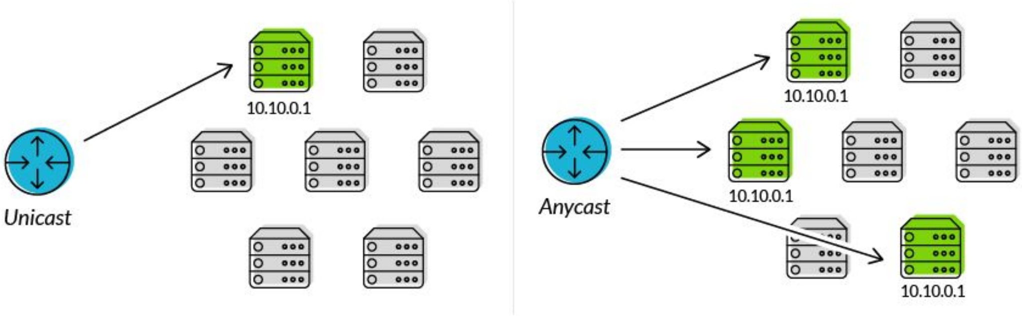
Labeled A through M

- Does this scale?

To scale root servers, operators rely on BGP anycast

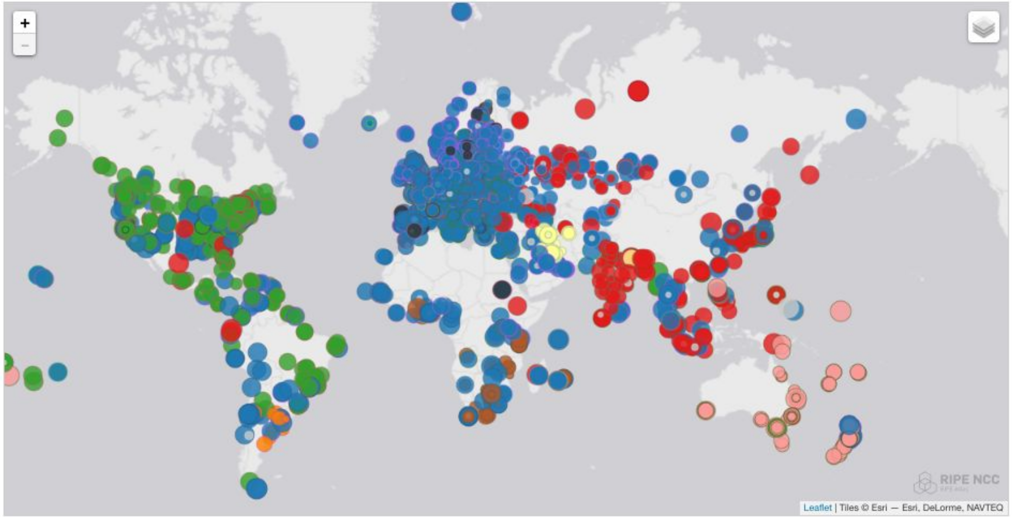
- Routing finds shortest-paths
- If several locations announce the same prefix, then routing will deliver the packets to the "closest" location
- This enables seamless replications of resources





To scale root servers, operators rely on BGP anycast

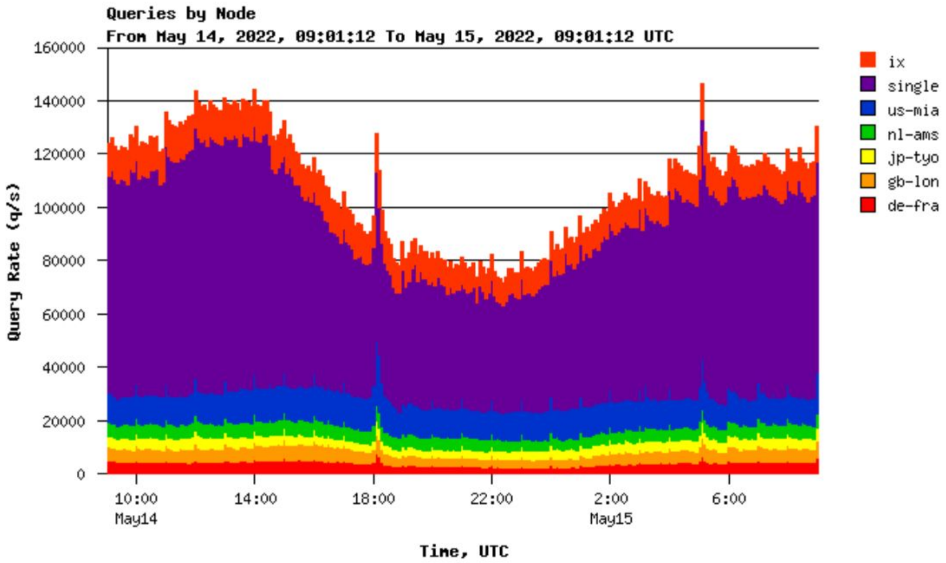
- K root (RIPE) anycast
  - Color == server used
- BGP is mediocre at this!



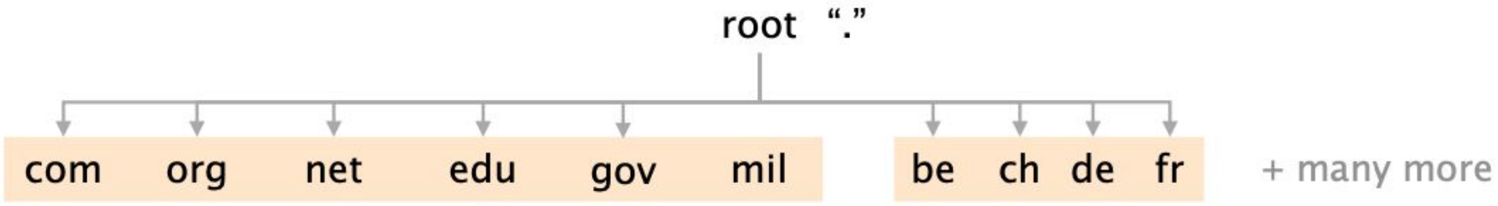
DNS scale

Each instance receives up to 80k queries per second

summing up to a few billions of queries per day



Top Level Domain (TLDs) sit below the root

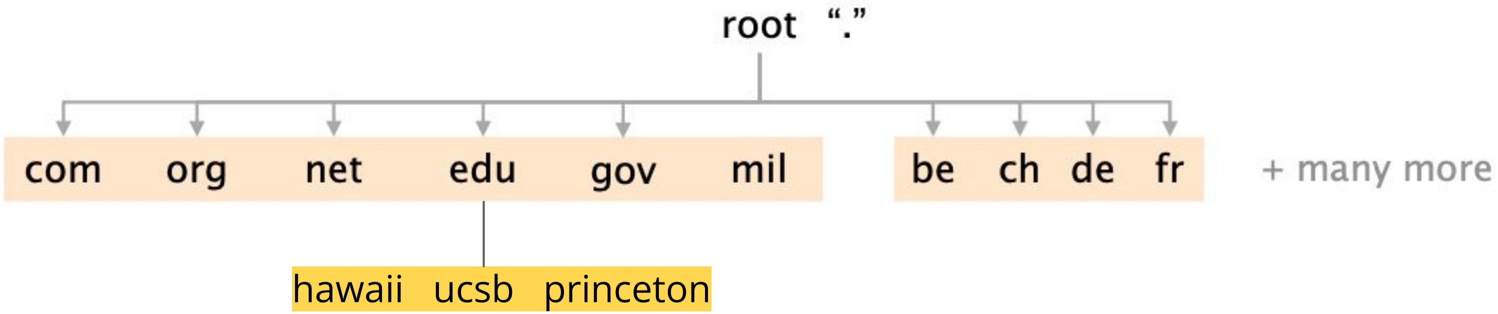


Each root knows the address of all TLD servers

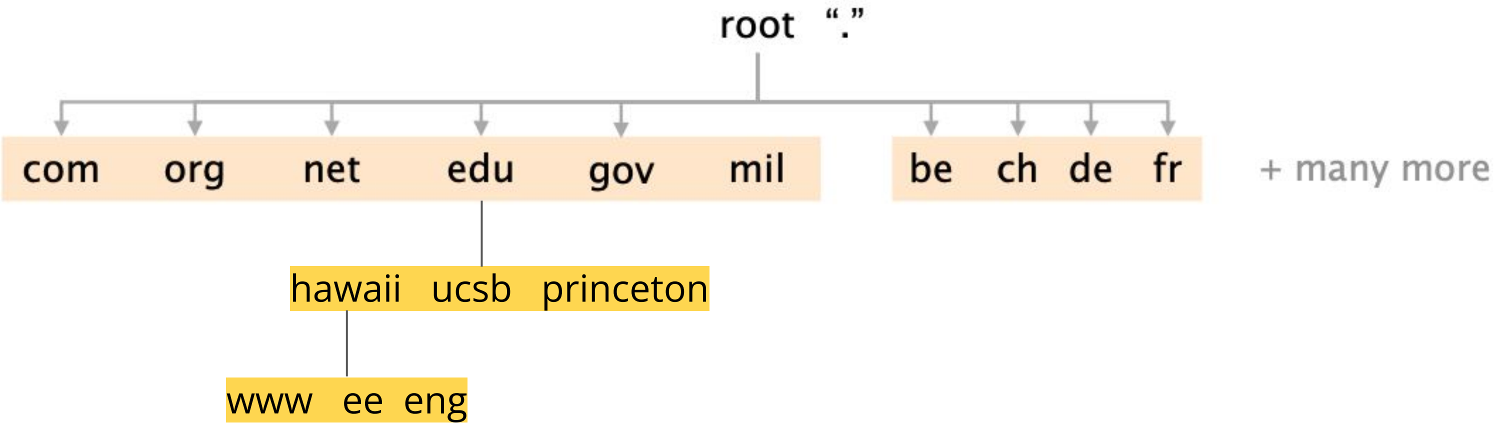
TLD and Authoritative DNS servers

- Top-level domain (TLD) servers
  - Generic domains (e.g., com, org, edu)
  - Country domains (e.g., uk, fr, cn, jp)
  - Special domains (e.g., arpa)
  - Typically managed professionally
    - Network Solutions maintains servers for “com”
    - Educause maintains servers for “edu”
- Authoritative DNS servers
  - Provide public records for hosts at an organization
  - For the organization’s servers (e.g., Web and mail)
  - Can be maintained locally or by a service provider

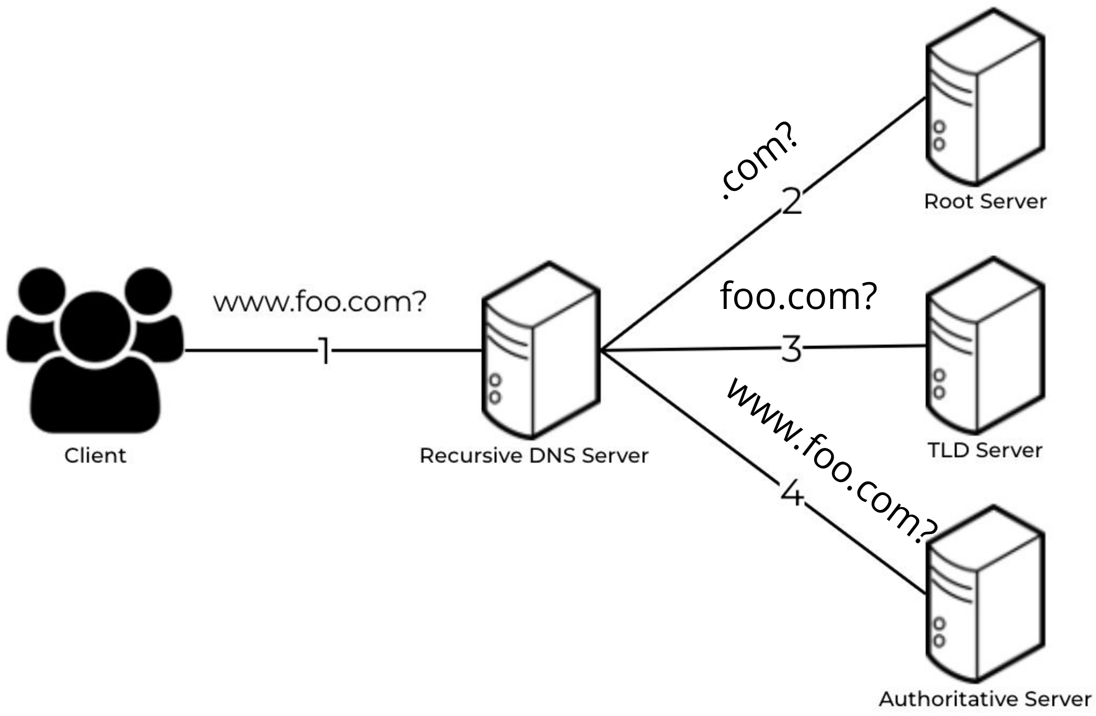
Domains are subtrees



A name, e.g. ee.hawaii.edu, represents a leaf-to-root path in the hierarchy



DNS Hierarchy



To ensure availability, each domain must have at least a primary and secondary DNS server

Ensure name service availability as long as one of the servers is up

DNS queries can be load-balanced across the replicas

On timeout, client use alternate servers exponential backoff when trying the same server

Overall, the DNS system is highly scalable, available, and extensible

Scalable	#names, #updates, #lookups, #users, but also in terms of administration
Available	domains replicate independently of each other
Extensible	any level (including the TLDs) can be modified independently