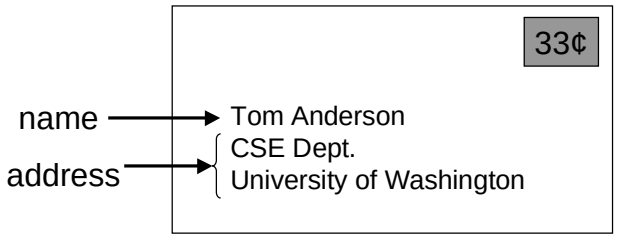
CSE/EE 461 Lecture 17
Domain Name System

Tom Anderson
tom@cs.washington.edu

Peterson, Chapter 9.1

Names and Addresses



- Names are identifiers for objects/services (high level)
- Addresses are locators for objects/services (low level)
- Resolution is the process of mapping name to address
- But addresses are really lower-level names
  - e.g., NAT translation from a virtual IP address to physical IP

Naming in Systems

- Ubiquitous
  - Files in filesystem, processes in OS, pages on the web, …
- Decouple identifier for object/service from location
  - Hostnames provide a level of indirection for IP addresses
- Naming greatly impacts system capabilities and performance
  - Ethernet addresses are a flat 48 bits
    - flat → any address anywhere but large forwarding tables
  - IP addresses are hierarchical 32/128 bits

Internet Hostnames

- Hostnames are human-readable identifiers for end-systems based on an administrative hierarchy
  - decouple identifier for object/service from its location
  - ex: june.cs.washington.edu, yahoo.com
- IP addresses are a fixed-length binary encoding for end-systems based on their position in the network
  - 128.95.2.106 is june's IP address
  - 216.115.109.6 is one of yahoo.com's IP addresses

Original Hostname System

- When the Internet was really young …
- Flat namespace
  - Simple (host, address) pairs
- Centralized management
  - Updates via a single master file called HOSTS.TXT
  - Manually coordinated by the Network Information Center (NIC)
- Resolution process
  - Look up hostname in the HOSTS.TXT file

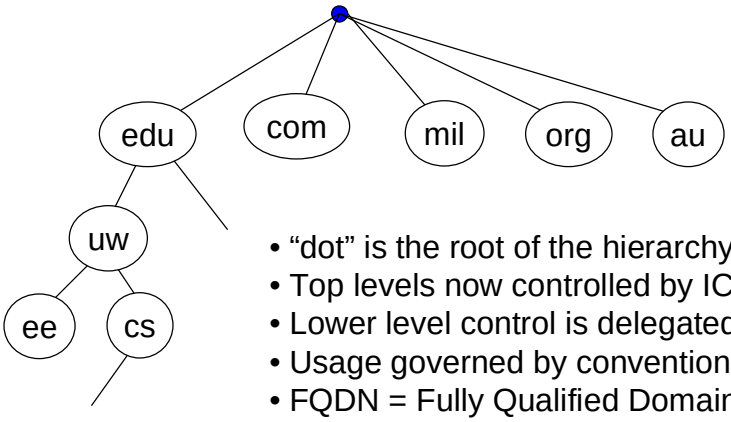
Scaling Problems

- Coordination
  - Between all users to avoid conflicts
- Inconsistencies
  - Between updated and old versions of file
- Reliability
  - Single point of failure
- Performance
  - Competition for centralized resources

Domain Name System (DNS)

- Developed by Mockapetris and Dunlap, mid-80's
- Namespace is hierarchical
  - Allows much better scaling of data structures
  - e.g., june.cs.washington.edu
- Namespace is distributed
  - Decentralized administration and access
  - e.g., june managed by CSE
- Resolution is by query/response
  - With replicated servers for redundancy
  - With heavy use of caching for performance

DNS Hierarchy



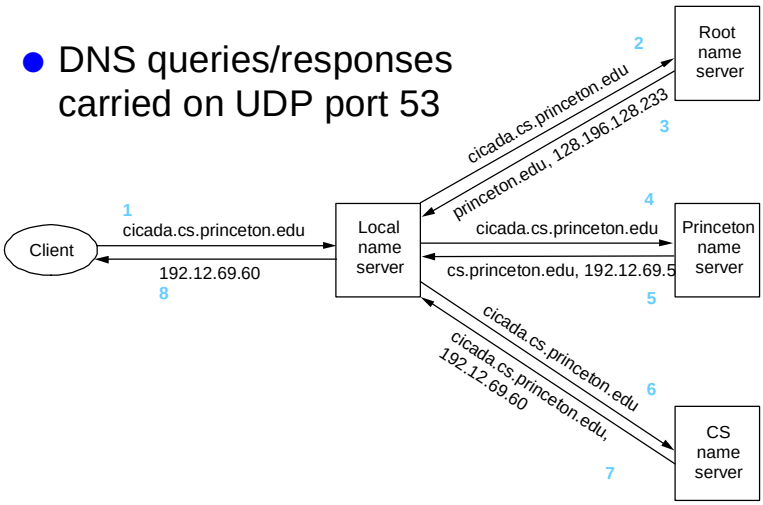
- "dot" is the root of the hierarchy
- Top levels now controlled by ICANN
- Lower level control is delegated
- Usage governed by conventions
- FQDN = Fully Qualified Domain Name

Name Space Delegation

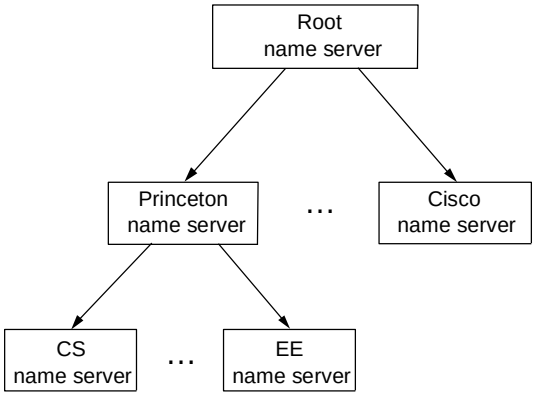
- Each organization controls its own name space ("zone" = subtree of global tree)
  - each organization has its own name servers
    - replicated for availability
  - name servers translate only names within their organization
    - client lookup proceeds step-by-step
  - example: washington.edu
    - contains IP addresses for all its hosts (www.washington.edu)
    - contains pointer to its subdomains (cs.washington.edu)

DNS Lookups/Resolution

- DNS queries/responses carried on UDP port 53



Hierarchy of Nameservers

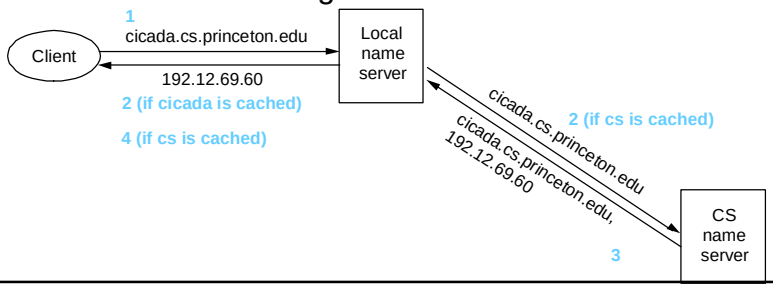


DNS Bootstrapping

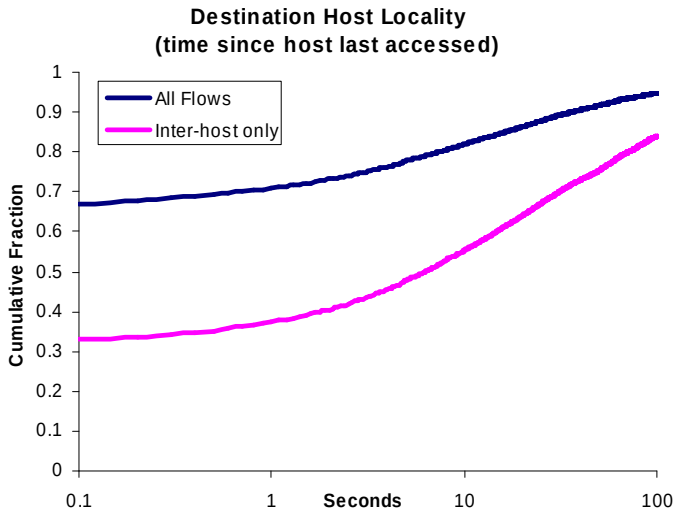
- Need to know IP addresses of root servers before we can make any queries
- Addresses for 13 root servers ([a-m].root-servers.net) handled via initial configuration (named.ca file)

DNS Performance: Caching

- DNS query results are cached at local proxy
  - quick response for repeated translations
  - lookups are the rare case
  - vastly reduces load at the servers
  - what if something new lands on slashdot?



DNS Cache Effectiveness



DNS Cache Consistency

- How do we keep cached copies up to date?
  - DNS entries are modified from time to time
    - to change name -> IP address mappings
    - to add/delete names
- Cache entries invalidated periodically
  - each DNS entry has time-to-live (TTL) field: how long can the local proxy can keep a copy?
  - if entry accessed after the timeout, get a fresh copy from the server
  - how do you pick the TTL?
  - how long after a change are all the copies updated?

Cache Consistency Alternatives

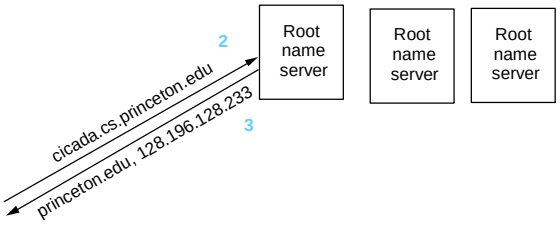
- Caches used in DNS, web proxies, reverse web proxies, network file systems, …
  - How do you keep cached copy up to date?
- Alternatives:
  - User-driven (HTTP)
    - user hits “reload” to fetch latest version
  - Timeouts (DNS, HTTP 1.0, NFS)
    - client fetches periodically; in meantime can be out of date
  - “If modified-sense” with timeouts (HTTP 1.1)
    - client periodically queries server; download if changed

Callback Cache Consistency

- Can we achieve strict consistency?
  - same as if server was accessed each time
- Callback-based caching
  - server keeps track of everyone who has a copy
  - before applying an update
    - send message to each copy to invalidate it
    - clients then refetch new version next time it is needed
  - what about scalability?

Availability

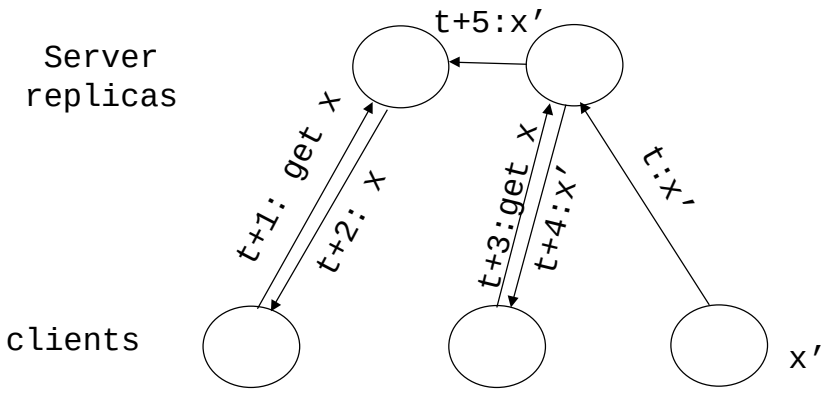
- What happens if DNS service is not working?
- DNS servers are replicated
  - name service available if at least one replica is working
  - queries load balanced between replicas



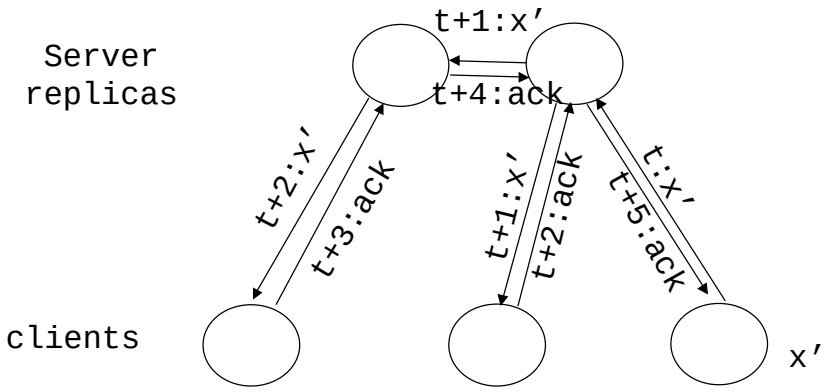
Replica Consistency

- Need to keep database consistent across all replicas as well as all caches
- DNS solution: eventual consistency
  - changes made to a master server
  - copied in the background to other replicas
  - in meantime can get inconsistent results, depending on which replica you consult
- Alternative: strict consistency
  - before making a change, notify all replicas to stop serving the data temporarily (and invalidate any copies)
  - broadcast new version to each replica
  - when everyone is updated, allow servers to resume

Eventual Consistency Example



Sequential Consistency Example



Write doesn't complete until all copies invalidated or updated

Building on the DNS

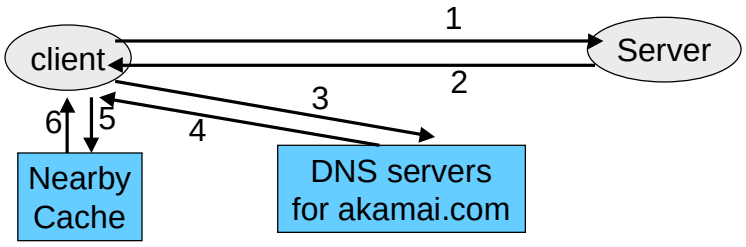
- Email: tom@cs.washington.edu
  - DNS record for tom in the domain cs.washington.edu, specifying where to deliver the email
- Uniform Resource Locators (URLs) name for Web pages
  - e.g., www.cs.washington.edu/homes/tom
  - Use domain name to identify a Web server
  - Use "/" separated string for file name on the server (or program to run to generate the page)

Future Evolution of the DNS

- Design constrains us in two major ways that are increasingly less appropriate
- Static host to IP mapping
  - What about mobility (Mobile IP) and dynamic address assignment (DHCP)?
- Location-insensitive queries
  - Many servers are geographically replicated; “yahoo.com” doesn’t refer to a single machine or even a single location (want closest server)

Akamai

- Use the DNS to effect selection of a nearby Web cache



- Names no longer mean same thing everywhere