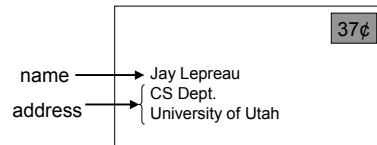


Domain Name System and Cache Consistency

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
 - hierarchy → smaller routing tables but constrained locations

Internet Hostnames

- Hostnames are human-readable identifiers for end-systems based on an administrative hierarchy
 - decouple identifier for object/service from its location
 - ex: www.cs.utah.edu, yahoo.com
- IP addresses are a fixed-length binary encoding for end-systems based on their position in the network
 - 155.99.25.15 is gluttony's (www's) IP address
 - 66.218.71.198 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

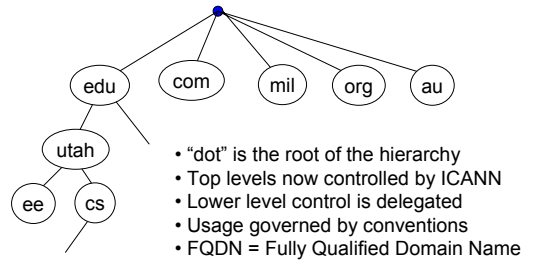
Scaling Problems

- 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., famine.cs.utah.edu
- Namespace is distributed
 - Decentralized administration and access
 - e.g., famine managed by CS facility
- Resolution is by query/response
 - With replicated servers for redundancy
 - With heavy use of caching for performance

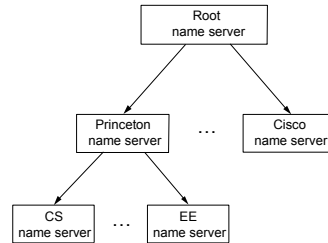
DNS Hierarchy



Name Space Delegation

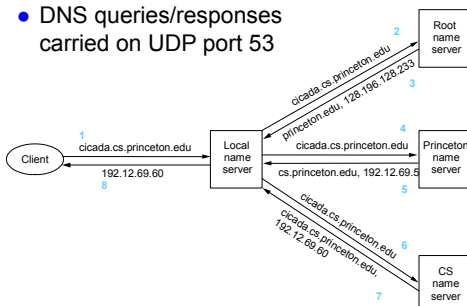
- Each organization controls its own name space ("zone" = subtree of global tree)
 - each organization has its own name servers
 - replicated for availability
 - name servers translate only names within their organization
 - client lookup proceeds step-by-step
 - example: utah.edu
 - contains IP addresses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

Hierarchy of Nameservers



DNS Lookups/Resolution

- DNS queries/responses carried on UDP port 53

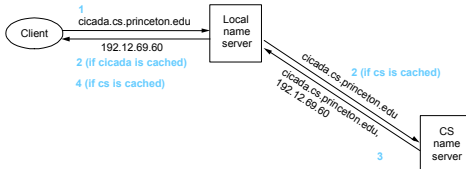


DNS Bootstrapping

- 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 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

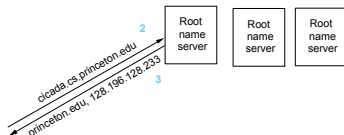
- Caches used in DNS, web proxies, reverse web proxies, network file systems, ...
 - How do you keep cached copy up to date?
- Alternatives:
 - User-driven (HTTP)
 - user hits "reload" to fetch latest version
 - Timeouts (DNS, HTTP 1.0, NFS)
 - client fetches periodically; in meantime can be out of date
 - "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?

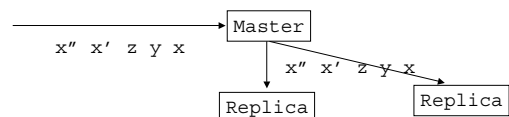
DNS Availability

- 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

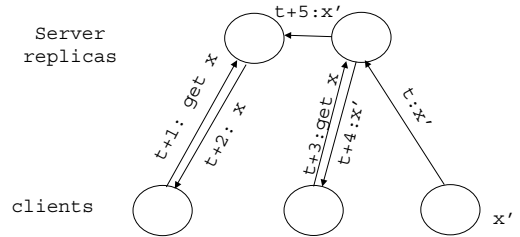
- How do we keep multiple copies of a database consistent?
- Apply same sequence of updates to each copy, *in the same order*
 - Example: send updates to master; master copies exact sequence of updates to each replica



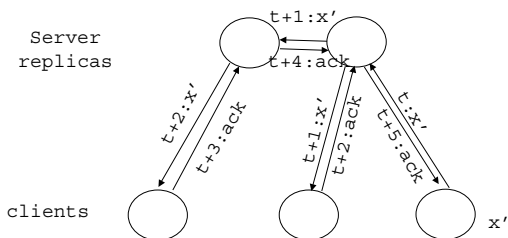
Replica Consistency

- While updates are propagating, which version(s) are visible?
- 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: lepreau@cs.utah.edu
 - DNS record for lepreau in the domain cs.utah.edu, specifying where to deliver the email
- Uniform Resource Locators (URLs) name for Web pages
 - e.g., www.cs.utah.edu/~lepreau
 - 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)

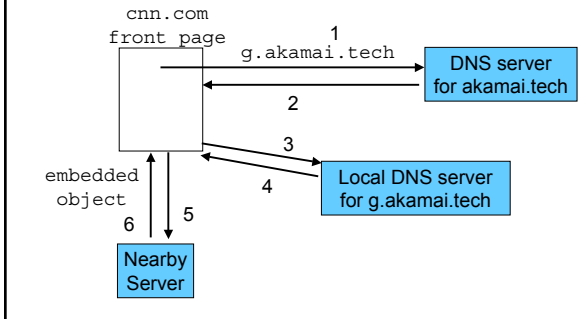
Future Evolution of the DNS

- 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 DNS to select a nearby Web cache
 - Front page points to g.akamai.tech
 - Special DNS server for akamai.tech, points to local akamai DNS server
 - return different server based on client location
 - use long TTL assuming clients don't move
 - Local DNS server points to local web server
 - use short TTL to allow rapid load balancing
 - Local server returns data
- Names no longer mean same thing everywhere

Akamai Example



Course Wrapup

Course Topics

- Internet architecture
 - how a web request works, from click to display
 - DNS lookup, connection setup, request/response to server, IP routing, media access, wire transmission ...
 - end to end principle
- Link layer
 - Encoding
 - Checksums and CRC's
 - Media access (Ethernet)

Course Topics

- Routing (IP)
 - forwarding and addressing mechanics
 - Virtual circuits vs. connectionless (ATM)
 - link state and distance vector routing (OSPF)
 - interdomain routing (BGP)
 - NATs and server load balancing
- Transport (TCP)
 - ARQ and sliding window
 - Connection setup/teardown and flow control
 - Congestion control: RTT estimation and window size
 - Remote procedure call

Course Topics

- Services
 - DNS lookup, caching and replication
 - distributed cache coherence

Internet Design Principles (1)

“Open Network Architecture”
[Cerf and Kahn 1974]

- Minimalism, autonomy
- Best-effort services
- Stateless routers
- Decentralized control

Internet Design Principles (2)

- End to end principle
 - Expect failures to occur at every step, so end hosts must be ultimately responsible for error recovery
 - example: TCP checksum, sliding window
- Soft state
 - if possible, state should be recoverable after a failure
 - example: link state routing messages are resent periodically, whether needed or not
- Design for scalability
 - using backoff: Ethernet, TCP congestion control
 - using hierarchy: IP addresses, DNS, routing (BGP)
 - using neighbors: Distance vector, Utella

Some Principles

- Transport
 - Reliable communication over unreliable network layer
 - Connection estab/teardown and handshaking
 - Congestion control
 - Flow control
 - Multiplexing
- Network
 - Finding “good” paths between routers
 - How to interconnect very large numbers of systems
- Data link layer
 - How to deal with a multiple access channel

Rules of Thumb [not in lecture]

- Some of these are from the trenches, some are widely-recognized “principles”
- Always use a version number in any protocol!
- [Maybe more later...]

Engineering Sermon: Simplicity

Simplicity is an absolute good, not a tradeoff

- Reasons for simplicity:
 - Can't sell it if you don't understand it
 - Easier to build, easier to maintain
 - Make things faster by making them simpler
 - Cheaper and quicker to market
- How do you make things simpler?
 - Use creativity to avoid complexity!
 - Don't accept complexity; design then code
 - Be extremist. What is the simplest possible design?

One Future Problem: Reliability

- Internet has ~ 98-99% uptime
 - measured end to end: can two hosts communicate?
 - telephone network: 99.99% uptime
 - air traffic control: 99.999% uptime
- How do we build more reliable systems?
 - Internet effective at masking router/link failures
 - Remaining failures are operational mistakes, programming errors, malicious attacks
 - Need to design for robustness to start (Tom Anderson of UW)
 - “Waste” resources to get other values

Another Problem: very tough to change the Internet!

How to cope:

- Make small tweaks (little impact)
- Design for alternative standalone network (little or no impact)

Better ways to cope:

- Go on top: overlay networks
- Go where constraints are different and it's not set in stone:
 - Wireless and mobile
 - Sensor nets (data centric)

New ideas in these areas are just emerging. Exciting!