DNS 101

Domain names are the “lingua franca” of the Internet.

- provides a symbolic veneer over the IP address space
- names for specific nodes, e.g., fran.cs.duke.edu
- names for service aliases (e.g., www, mail servers)

- Almost every Internet application uses domain names when it establishes a connection to another host.

The Domain Name System (DNS) is a planetary name service that translates Internet domain names.

maps \(<node\ name\rangle\) to \(<IP\ address\rangle\)

(mostly) independent of location, routing etc.

---

Domain Name Hierarchy

DNS name space is hierarchical:
- fully qualified names are “little endian”
- scalability
- decentralized administration
- domain are naming contexts
  replaces flat hosts.txt namespace

How is this different from hierarchical directories in distributed file systems? Do we already know how to implement this?
DNS Implementation 101

DNS protocol/implementation:
- UDP-based client/server
- client-side resolvers typically in a library `gethostbyname`, `gethostbyaddr`
- cooperating servers query-answer-referral model
- forward queries among servers
- server-to-server may use TCP ("zone transfers")
- common implementation: BIND

DNS Name Server Hierarchy

DNS servers are organized into a hierarchy that mirrors the name space.

Specific servers are designated as authoritative for portions of the name space.

Servers may delegate management of subdomains to child name servers.

Parents refer subdomain queries to their children.

Subdomains correspond to organizational (administrative) boundaries, which are not necessarily geographical.

Servers are bootstrapped with pointers to selected peer and parent servers.

Resolvers are bootstrapped with pointers to one or more local servers; they issue recursive queries.
DNS: The Big Issues

1. How to establish naming contexts?
   I want to use short, unqualified names like whiteout instead of whiteout.cs.duke.edu when I’m in the cs.duke.edu domain.

2. What about trust? How can we know if a server is authoritative, or just an impostor?
   What happens if a server lies or behaves erratically? What denial-of-service attacks are possible? What about privacy?

3. What if an “upstream” server fails?

4. Is the hierarchical structure sufficient for scalability?
   more names vs. higher request rates

---

DNS Caching

Caching of query responses allows subsequent queries to bypass the roots of the server hierarchy.

Each response is stamped with a time-to-live (TTL) to limit damage from stale cache entries.

What about negative caching: is it worthwhile to cache negative responses?
DNS Replication

Every DNS domain has or should have at least one secondary name server replica.
- configure peers to offload queries from primary
- serve as authoritative backup

Secondary replicas keep themselves up to date by periodically fetching/refreshing the entire naming database via zone transfer (TCP).

The primary database is timestamped with a "serial number" to short-circuit if no updates have occurred since last zone transfer.

How to load-balance the secondaries?
What if primary is overloaded with too many secondaries requesting zone transfers?

Reverse Translation

152

3

140

5
(prophet)

152.3.140.5
Directory Services (e.g., LDAP)

A directory service is a souped-up a name service.
- read-mostly access to named entries with unique, global distinguished names
- clients see a uniform view of a hierarchical name space
  authority to serve the name space is partitioned across a collection of servers
    partitioning reflects geographical or organizational boundaries
    context-based lookups with referrals
- simple updates: add/delete/update single entry
- large-scale caching/replication with soft consistency

Attributes and Searching

Directory services are augmented with support for attributes.
  e.g., LDAP: Lightweight Directory Access Protocol (X.500)
- An entry is a named collection of attributes.
- An attribute is a typed collection of values, whose format is defined by its type.
  e.g., printer, name=buzzard, location=LSRC 312, resolution=600dpi
  Attributes are more useful if their types are standardized.
Attributes can be used as the basis for searches that find an object with specified properties.
- can specify filters, scope of search, etc.
- goal: attribute-based definition of services