Naming

- How do we efficiently locate resources?
  - DNS: name → IP address
  - Service location: description → host
- Other issues
  - How do we scale these to the wide area?
  - How to choose among similar services?

Overview

- DNS
- Service location
Obvious Solutions (1)

Why not centralize DNS?
• Single point of failure
• Traffic volume
• Distant centralized database
• Single point of update

• Doesn’t scale!

Obvious Solutions (2)

Why not use /etc/hosts?
• Original Name to Address Mapping
• Flat namespace
• /etc/hosts
• SRI kept main copy
• Downloaded regularly
• Count of hosts was increasing: machine per domain → machine per user
• Many more downloads
• Many more updates

Domain Name System Goals
• Basically building a wide area distributed database
• Scalability
• Decentralized maintenance
• Robustness
• Global scope
  • Names mean the same thing everywhere
• Don’t need
  • Atomicity
  • Strong consistency

DNS Records

RR format: (class, name, value, type, ttl)

• DB contains tuples called resource records (RRs)
• Classes = Internet (IN), Chaosnet (CH), etc.
• Each class defines value associated with type

FOR IN class:

• Type=A
  • name is hostname
  • value is IP address

• Type=NS
  • name is domain (e.g. foo.com)
  • value is name of authoritative name server for this domain

• Type=CNAME
  • name is alias name for some “canonical” (the real) name
  • value is canonical name

• Type=MX
  • name is hostname of mailserver associated with name
DNS Design: Hierarchy Definitions

- Each node in hierarchy stores a list of names that end with same suffix
  - Suffix = path up tree
  - E.g., given this tree, where would following be stored:
    - Fred.com
    - Fred.edu
    - Fred.cmu.edu
    - Fred.cmcl.cs.cmu.edu
    - Fred.cs.mit.edu

DNS Design: Zone Definitions

- Zone = contiguous section of name space
  - E.g., Complete tree, single node or subtree
  - A zone has an associated set of name servers

DNS Design: Cont.

- Zones are created by convincing owner node to create/delegate a subzone
  - Records within zone stored multiple redundant name servers
    - Primary/master name server updated manually
    - Secondary/redundant servers updated by zone transfer of name space
      - Zone transfer is a bulk transfer of the “configuration” of a DNS server – uses TCP to ensure reliability
  - Example:
    - CS.CMU.EDU created by CMU.EDU administrators

Servers/Resolvers

- Each host has a resolver
  - Typically a library that applications can link to
  - Local name servers hand-configured (e.g. /etc/resolv.conf)
- Name servers
  - Either responsible for some zone or…
  - Local servers
    - Do lookup of distant host names for local hosts
    - Typically answer queries about local zone
**DNS: Root Name Servers**

- Responsible for "root" zone
- Approx. dozen root name servers worldwide
  - Currently (a-m).root-servers.net
- Local name servers contact root servers when they cannot resolve a name
  - Configured with well-known root servers

**DNS Message Format**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification</td>
<td>12 bytes of data used for matching request and response</td>
</tr>
<tr>
<td>No. of Questions</td>
<td>Number of questions in the message</td>
</tr>
<tr>
<td>No. of Answer RRs</td>
<td>Number of answer resource records in the message</td>
</tr>
<tr>
<td>No. of Authority RRs</td>
<td>Number of authority resource records in the message</td>
</tr>
<tr>
<td>No. of Additional RRs</td>
<td>Number of additional resource records in the message</td>
</tr>
<tr>
<td>Questions</td>
<td>Questions and answers to the query</td>
</tr>
<tr>
<td>RRs in response to query</td>
<td>Resource records in response to the query</td>
</tr>
<tr>
<td>Authority</td>
<td>Authority records in the message</td>
</tr>
<tr>
<td>Additional Info</td>
<td>Additional resource records that may be used</td>
</tr>
</tbody>
</table>

**DNS Header Fields**

- **Identification**
  - Used to match up request/response
- **Flags**
  - 1-bit to mark query or response
  - 1-bit to mark authoritative or not
  - 1-bit to request recursive resolution
  - 1-bit to indicate support for recursive resolution

**Typical Resolution**

Client \[www.cs.cmu.edu\] → Local DNS server \[ns1.cs.cmu.edu\] → Root & Edu DNS server \[ns1.cs.cmu.edu\] → www.cs.cmu.edu DNS server

Client \[www.cs.cmu.edu\] → Local DNS server \[ns1.cs.cmu.edu\] → ns1.cs.cmu.edu DNS server
**Typical Resolution**

- Steps for resolving www.cmu.edu
  - Application calls gethostbyname() (RESOLVER)
  - Resolver contacts local name server (S₁)
  - S₁ queries root server (S₂) for www.cmu.edu
  - S₂ returns NS record for cmu.edu (S₃)
  - What about A record for S₃?
    - This is what the additional information section is for
    - PREFETCHING
  - S₁ queries S₃ for www.cmu.edu
  - S₃ returns A record for www.cmu.edu
  - Can return multiple A records → what does this mean?

**Lookup Methods**

- Recursive query:
  - Server goes out and searches for more info (recursive)
  - Only returns final answer or “not found”
- Iterative query:
  - Server responds with as much as it knows (iterative)
  - “I don’t know this name, but ask this server”

Workload impact on choice?
- Local server typically does recursive
- Root/distant server does iterative

**Workload and Caching**

- What workload do you expect for different servers/names?
  - Why might this be a problem? How can we solve this problem?
- DNS responses are cached
  - Quick response for repeated translations
  - Other queries may reuse some parts of lookup
- NS records for domains
- DNS negative queries are cached
  - Don’t have to repeat past mistakes
  - E.g. misspellings, search strings in resolv.conf
- Cached data periodically times out
  - Lifetime (TTL) of data controlled by owner of data
  - TTL passed with every record

**Typical Resolution**

Client → Local DNS server → ns1.cmu.edu DNS server → www.cs.cmu.edu DNS server → root & edu DNS server
Subsequent Lookup Example

Client -> Local DNS server

ftp.cs.cmu.edu

root & edu DNS server

cmu.edu DNS server

cs.cmu.edu DNS server

Reliability

- DNS servers are replicated
  - Name service available if one replica is up
  - Queries can be load balanced between replicas
- UDP used for queries
  - Need reliability → must implement this on top of UDP!
  - Why not just use TCP?
- Try alternate servers on timeout
  - Exponential backoff when retrying same server
- Same identifier for all queries
  - Don’t care which server responds

Reverse Name Lookup

- 128.2.206.138?
  - Lookup 138.206.2.128.in-addr.arpa
  - Why is the address reversed?
  - Happens to be www.intel-iris.net and mammoth.cmcl.cs.cmu.edu → what will reverse lookup return? Both?
    - Should only return name that reflects address allocation mechanism

Prefetching

- Name servers can add additional data to any response
- Typically used for prefetching
  - CNAME/MX/NS typically point to another host name
  - Responses include address of host referred to in "additional section"
Mail Addresses

- MX records point to mail exchanger for a name
  - E.g. mail.acm.org is MX for acm.org
- Addition of MX record type proved to be a challenge
  - How to get mail programs to lookup MX record for mail delivery?
  - Needed critical mass of such mailers

DNS Experience

- One of the greatest challenges seemed to be getting good name server implementations
  - Developers were typically happy with “good enough” implementation
  - Challenging, large scale, wide area distributed system
    - Like routing, but easier to have broken implementations that work

DNS Experience

- Common bugs
  - Looped NS/CNAME record handling
  - Poor static configuration (root server list)
  - Lack of exponential backoff
  - No centralized caching per site
    - Each machine runs on caching local server
    - Why is this a problem?
- Solution
  - Monitor for misbehaving name servers?

DNS Experience

- 23% of lookups with no answer
  - Retransmit aggressively → most packets in trace for unanswered lookups!
  - Correct answers tend to come back quickly/with few retries
- 10 - 42% negative answers → most = no name exists
  - Inverse lookups and bogus NS records
- Worst 10% lookup latency got much worse
  - Median 85→97, 90th percentile 447→1176
  - Increasing share of low TTL records → what is happening to caching?
DNS Experience

- Hit rate for DNS = 80% \(\rightarrow 1-\frac{\#DNS}{\#connections}\)
  - Most Internet traffic is Web
  - What does a typical page look like? \(\rightarrow\) average of 4-5 imbedded objects \(\rightarrow\) needs 4-5 transfers \(\rightarrow\) accounts for 80% hit rate!
  - 70% hit rate for NS records \(\rightarrow\) i.e. don’t go to root/gTLD servers
    - NS TTLs are much longer than A TTLs
    - NS record caching is much more important to scalability
  - Name distribution = Zipf-like \(= \frac{1}{x^a}\)
  - A records \(\rightarrow\) TTLs = 10 minutes similar to TTLs = infinite
  - 10 client hit rate = 1000+ client hit rate

Root Zone

- Generic Top Level Domains (gTLD) = .com, .net, .org, etc…
- Country Code Top Level Domain (ccTLD) = .us, .ca, .fi, .uk, etc…
- Root server ((a-m).root-servers.net) also used to cover gTLD domains
  - Load on root servers was growing quickly!
  - Moving .com, .net, .org off root servers was clearly necessary to reduce load \(\rightarrow\) done Aug 2000

New gTLDs

- .info \(\rightarrow\) general info
- .biz \(\rightarrow\) businesses
- .aero \(\rightarrow\) air-transport industry
- .coop \(\rightarrow\) business cooperatives
- .name \(\rightarrow\) individuals
- .pro \(\rightarrow\) accountants, lawyers, and physicians
- .museum \(\rightarrow\) museums
  - Only new one actives so far = .info, .biz, .name

New Registrars

- Network Solutions (NSI) used to handle all registrations, root servers, etc…
  - Clearly not the democratic (Internet) way
  - Large number of registrars that can create new domains \(\rightarrow\) However, NSI still handle root servers
Overview

- **DNS**

- **Service location**

Service Location

- What if you want to lookup services with more expressive descriptions than DNS names
  - E.g. please find me printers in cs.cmu.edu instead of laserjet1.cs.cmu.edu

- What do descriptions look like?
- How is the searching done?
- How will it be used?
  - Search for particular service?
  - Browse available services?
  - Composing multiple services into new service?

Service Descriptions

- Typically done as hierarchical value-attribute pairs
  - Type = printer → memory = 32MB, lang = PCL
  - Location = CMU → building = WeH

- Hierarchy based on attributes or attributes-values?
  - E.g. Country → state or country=USA → state=PA and country=Canada → province=BC?

- Can be done in something like XML

Service Discovery (Multicast)

- Services listen on well known discovery group address
- Client multicasts query to discovery group
- Services unicast replies to client

- Tradeoffs
  - Not very scalable → effectively broadcast search
  - Requires no dedicated infrastructure or bootstrap
  - Easily adapts to availability/changes
  - Can scope request by multicast scoping and by information in request
Service Discovery (Directory Based)

- Services register with central directory agent
  - Soft state registrations must be refreshed or the expire
- Clients send query to central directory replies with list of matches
- Tradeoffs
  - How do you find the central directory service?
  - Typically using multicast based discovery!
  - SLP also allows directory to do periodic advertisements
  - Need dedicated infrastructure
  - How do directory agents interact with each other?
  - Well suited for browsing and composition knows full list of services

Service Discovery (Routing Based)

- Client issues query to overlay network
  - Query can include both service description and actual request for service
- Overlay network routes query to desired service(s)
- If query only description, subsequent interactions can be outside overlay (early-binding)
- If query includes request, client can send subsequent queries via overlay (late-binding)
  - Subsequent requests may go to different services agents
  - Enables easy fail-over/mobility of service
- Tradeoffs
  - Routing on complex parameters can be difficult/expensive
  - Can work especially well in ad-hoc networks
  - Can late-binding really be used in many applications?

Wide Area Scaling

- How do we scale discovery to wide area?
  - Hierarchy?
- Hierarchy must be based on attribute of services
  - All services must have this attribute
  - All queries must include (implicitly or explicitly) this attribute
- Tradeoffs
  - What attribute? Administrative (like DNS)? Geographic? Network Topologic?
  - Should we have multiple hierarchies?
  - Do we really need hierarchy? Search engines seem to work fine!

Other Issues

- Dynamic attributes
  - Many queries may be based on attributes such as load, queue length
  - E.g., print to the printer with shortest queue
- Security
  - Don’t want others to serve/change queries
  - Also, don’t want others to know about existence of services
  - Srini’s home SLP server is advertising the $50,000 MP3 stereo system (come steal me!)
Next Lecture: Midterm

- Closed book
- Up through multi-hop wireless
- Question types
  - Multiple choice
  - Short answer
  - Multi-part questions