Introduction to the DNS system

Presented by Joe Abley
SANOG 4, 2004

Purpose of naming

- Addresses are used to locate objects
- Names are easier to remember than numbers
- You would like to get to the address or other objects using a name
- DNS provides a mapping from names to resources of several types

Names and addresses in general

- An address is how you get to an endpoint
  - Typically, hierarchical (for scaling):
    - 950 Charter Street, Redwood City CA, 94063
    - 204.152.187.11, +1-650-381-6003
- A “name” is how an endpoint is referenced
  - Typically, no structurally significant hierarchy
  - “David”, “Tokyo”, “itu.int”

Naming History

- 1970’s ARPANET
  - Host.txt maintained by the SRI-NIC
  - pulled from a single machine
  - Problems
    - traffic and load
    - Name collisions
    - Consistency
- DNS reated in 1983 by Paul Mockapetris (RFCs 1034 and 1035), modified, updated, and enhanced by a myriad of subsequent RFCs

DNS

- A lookup mechanism for translating objects into other objects
- A globally distributed, loosely coherent, scalable, reliable, dynamic database
- Comprised of three components
  - A “name space”
  - Servers making that name space available
  - Resolvers (clients) which query the servers about the name space

DNS Features: Global Distribution

- Data is maintained locally, but retrievable globally
  - No single computer has all DNS data
- DNS lookups can be performed by any device
- Remote DNS data is locally cachable to improve performance
DNS Features: Loose Coherency

- The database is always internally consistent
  - Each version of a subset of the database (a zone) has a serial number
    - The serial number is incremented on each database change
- Changes to the master copy of the database are replicated according to timing set by the zone administrator
- Cached data expires according to timeout set by zone administrator

DNS Features: Scalability

- No limit to the size of the database
  - One server has over 20,000,000 names
    - Not a particularly good idea
- No limit to the number of queries
  - 24,000 queries per second handled easily
- Queries distributed among masters, slaves, and caches

DNS Features: Reliability

- Data is replicated
  - Data from master is copied to multiple slaves
- Clients can query
  - Master server
  - Any of the copies at slave servers
- Clients will typically query local caches
- DNS protocols can use either UDP or TCP
  - If UDP, DNS protocol handles retransmission, sequencing, etc.

DNS Features: Dynamicity

- Database can be updated dynamically
  - Add/delete/modify of any record
- Modification of the master database triggers replication
  - Only master can be dynamically updated
    - Creates a single point of failure

DNS Concepts

- Next slides are about concepts
- After this set of slides you should understand
  - How the DNS is built
  - Why it is built the way it is
  - The terminology used throughout the course

Concept: DNS Names 1

- The namespace needs to be made hierarchical to be able to scale.
- The idea is to name objects based on
  - location (within country, set of organizations, set of companies, etc)
  - unit within that location (company within set of company, etc)
  - object within unit (name of person in company)
### Concept: DNS Names 2

**How names appear in the DNS**

- **Fully Qualified Domain Name (FQDN)**
  - **WWW.RIPE.NET.**
  - Labels separated by dots
  - Note the trailing dot

- DNS provides a mapping from FQDNs to resources of several types

- Names are used as a key when fetching data in the DNS

### Concept: Resource Records

- The DNS maps names into data using Resource Records.

- **Resource Record**
  - **www.ripe.net. ... A 10.10.10.2**
  - **Address Resource**

- More detail later

### Concept: Domains

- Domains are "namespaces"
- Everything below .com is in the com domain.
- Everything below ripe.net is in the ripe.net domain and in the net domain.

### Concept: Zones and Delegations

- Zones are "administrative spaces"
- Zone administrators are responsible for portion of a domain’s name space
- Authority is delegated from a parent and to a child

---

**Delegation**

- Administrators can create subdomains to group hosts
  - According to geography, organizational affiliation or any other criterion

- An administrator of a domain can delegate responsibility for managing a subdomain to someone else
  - But this isn’t required

- The parent domain retains links to the delegated subdomain
  - The parent domain “remembers” who it delegated the subdomain to
**Concept: Name Servers**

- Name servers answer ‘DNS’ questions.

- Several types of name servers
  - Authoritative servers
    - master (primary)
    - slave (secondary)
  - (Caching) recursive servers
    - also caching forwarders
  - Mixture of functionality

**Concept: Authoritative Name Server**

- Give authoritative answers for one or more zones.
- The master server normally loads the data from a zone file
- A slave server normally replicates the data from the master via a zone transfer

**Concept: Recursive Server**

- Recursive servers do the actual lookups; they ask questions to the DNS on behalf of the clients.
- Answers are obtained from authoritative servers but the answers forwarded to the clients are marked as not authoritative
- Answers are stored for future reference in the cache

**Concept: Resolvers**

- Resolvers ask the questions to the DNS system on behalf of the application.
- Normally implemented in a system library (e.g., libc)
  - `gethostbyname(char *name);`
  - `gethostbyaddr(char *addr, int len, type);`

**Concept: Resolving Process & Cache**

- **Question:** www.ripe.net A
- **Root Server:**
  - www.root-servers.net
- **Caching Forwarder (recursive):**
  - www.caching-forwarder.net
- **GTLDServer:**
  - www.gtld-servers.net
- **RIPE Server:**
  - ns.ripe.net
- **Add to cache:**
  - 192.168.5.10

**Concept: Resource Records (more detail)**

- Resource records consist of its name, its TTL, its class, its type and its RDATA
- TTL is a timing parameter
- IN class is widest used
- There are multiple types of RR records
- Everything behind the type identifier is called rdata

- **Example:** `www.ripe.net. 3600 IN A 10.10.10.2`
### Example: RRs in a zone file

```
ripe.net. 7200 IN      SOA     ns.ripe.net.    olaf.ripe.net. (
2001061501 ; Serial
43200 ; Refresh 12 hours
14400 ; Retry 4 hours
345600 ; Expire 4 days
7200 ; Negative cache 2 hours
)
ripe.net. 7200  IN    NS      ns.ripe.net.
ripe.net. 7200  IN    NS      ns.eu.net.
pinkje.ripe.net. 3600  IN    A       193.0.1.162
host25.ripe.net. 2600  IN    A       193.0.3.25
```

### Resource Record: SOA

- **Label**: part of the domain name.
- **ttl**: time to live (in seconds).
- **class**: internet.
- **type**: SOA (Start of Authority).
- **rdata**: details of the SOA record.

```
ripe.net. 3600 IN SOA  A.GTLD-SERVERS.net. nstld.verisign-grs.com. (2002021301; serial 30M; refresh 15M; retry 1W; expiry 1D); neg. answ. ttl
```

### Resource Record: NS

- **Label**: part of the domain name.
- **ttl**: time to live (in seconds).
- **class**: internet.
- **type**: NS (Name Server).
- **rdata**: details of the NS record.

```
pinkje.ripe.net. 3600 IN A 193.0.1.162
host25.ripe.net. 2600 IN A 193.0.3.25
```

### Concept: TTL and other Timers

- **TTL**: time to live (in seconds).
- **Master server**: authoritative server.
- **Contact address**: where to send queries.
- **Version number**: identifies changes.
- **Timing parameter**: values for refreshing, retrying, etc.

### Places where DNS data lives

- **Registry DB**: authoritative data.
- **Master server**: authoritative server.
- **Slave server**: secondary server.
- **Cache server**: reduces load and response times.

### To remember...

- **Multiple authoritative servers**: to distribute load and risk.
- **Caches**: reduce load to authoritative servers and response times.
- **SOA timers and TTL**: need to be tuned to needs of zone.

- **TTL** is a timer used in caches.
  - An indication for how long the data may be reused.
  - Data that is expected to be ‘stable’ can have high TTLs.
- **SOA timers** are used for maintaining consistency between primary and secondary servers.

Changes in DNS do not propagate instantly!

Might take up to refresh to get data from master.

Not going to net if TTL>0

Upload of zone data is local policy.

Registry DB

Master

Slave

Slave server

Not going to net if TTL>0

Cache server
What have we learned
What are we about to learn

- We learned about the architecture:
  - resolvers,
  - caching forwarders,
  - authoritative servers,
  - timing parameters