Introduction to "Domain Name Service" (DNS)

1. Host name to IP number mapping was originally done by downloading a static file

2. The UNIX version of this file is `/etc/hosts` (the file we ftp'd in was called hosts.txt

Introduction to the Domain Name Service (DNS)

3. The central file was maintained by the Stanford Research Institute Network Information Center (SRI-NIC)

Introduction to the Domain Name Service (DNS)

4. As the Internet grew this scheme became unworkable
   - ☞ The size of the file became too large
   - ☞ The load on SRI-NIC site became too heavy
   - ☞ The file was always inconsistent with reality
   - ☞ Hostname collisions became frequent (anyone could name their machine "whitehouse.gov" if they wanted to)

Overview of DNS

In 1984 Paulk Mockapetris of USC designed the architecture of DNS. It's based on the idea of "resource records".

Overview of DNS

The InterNIC was set up to manage DNS; however, this responsibility was given over to ICANN (Internet Corporation for Assigned Names and Numbers), and now the "InterNIC" is just a website. One of the most important activities of ICANN is to accredit registrars.

Overview of DNS

Today, there are hundreds of registrars, and prices for domain names have dropped under $10.

Features

- ☞ Local control: each segment is updated locally
- ☞ Global access: each segment is available (almost) immediately to the rest of the world upon update

Features

- ☞ Robustness: achieved through replication
- ☞ Adequate performance: is achieved through caching

Software

☞ Servers: called name servers, contain information about some segment of the network and make it available to clients (“BIND” = “Berkeley Internet Name Daemon”, includes “named”, libraries, “nslookup”, “dig”, “host”)

Software

☞ Client: resolvers, a set of library routines that resolve names by accessing a server (originally a separate library, like `libresolv.a`, now usually part of `libc.a`)

Software

☞ Domain name server software is also available for non-UNIX platforms, such as Windows 2008 and Macintosh OS X.

Domain structure

- ☞ Similar to the structure of a hierarchical file system
- ☞ The root's name is the null label " " but is written as a single dot ".".

Domain structure

- ☞ Each node represents a 'domain'
- ☞ Every domain is named

Domain structure

☞ The full domain name is the sequence of labels from the domain to the root, separated by periods

Domain structure

☞ Unlike a file system pathname the name is read from leaf to root (right to left rather than left to right)

```
xi.cs.fsu.edu
```

Domain management

- Each domain may be managed by a different organization
- The organization may divide itself into subdomains

Domain management

- ☞ Then delegate responsbility for maintaining them
- ☞ ICANN (currently) manages the "provisioning" of top-level domains

Domain management

What is ICANN?

The Internet Corporation for Assigned Names and Numbers (ICANN) is responsible for managing and coordinating the Domain Name System (DNS) to ensure that every address is unique and that all users of the Internet can find all valid addresses. It does this by overseeing the distribution of unique IP addresses and domain names. It also ensures that each domain name maps to the correct IP address.

ICANN is also responsible for accrediting the domain name registrars. "Accredit" means to identify and set minimum standards for the performance of registration functions, to recognize persons or

entities meeting those standards, and to enter into an accreditation agreement that sets forth the rules and procedures applicable to the provision of Registrar Services.

Host names

- Each host on a network has a domain
- The domain points to information about the host

Host names

- This may include:
  - An IP address (A records)
  - Mail routing information (MX records)

Host names

- Aliases which point to the real ("canonical") host name (CNAME records)

The domain name space

- There may be any number of branches at a node
- Some implementations limit the tree's depth

The domain name space

- Each name may contain up to 63 characters
- The suggested length is 12 or less characters

The domain name space

- A domain name that is written relative to the root is called a 'fully-qualified domain name' - FQDN
- Names without trailing dots ("leading dots") are sometimes interpreted as relative to some domain other than root

The domain name space

- ☞ Sibling nodes must have unique names
- ☞ The name of a domain is the domain name of the node at the top of the domain (example purdue.edu)

The domain name space

- Again, similar to a file system
- A node is in multiple domains

The domain name space

- So, a domain is just a subtree of the domain name space ("subdomain")
- Must not use "_", although other similiar naming schemes **(prominently, such as NIS)** have allowed this.

Hosts

- Where are the hosts?
- A domain name is just an index into the DNS database

Hosts

- ☞ The 'hosts' are domain names that point to individual machine information
- ☞ The hosts are related 'logically' usually by geography or organization

Hosts

- They are NOT necessarily related by network or IP address or hardware type
- You could have 10 different hosts on 10 different networks in ten different countries all in the same domain (hp.com)

Hosts

- Nodes at the leaves of the tree usually represent individual hosts

- Interior nodes may point to both host information and to subdomain information For example, "hp.com" is both the name of a domain and the name of a machine that routes mail

The domain name space

- Terms
  - top-level domain (TLD): a child of root (edu)
  - first-level domain: a child of root (edu)
  - second-level domain: a child of 1st level domain (fsu.edu)

The domain name space

The domain name space

- Naming rules - the original 7 top-level domains were:
  - com - commerical organizations
  - edu - educational organizations
  - gov - governmental bodies
  - mil - military organizations
  - net - networking organizations
  - org - non-commercial organizations
  - int - international organizations

The domain name space

- International names (ISO 3166-1 names)
  - 2-letter designations are reserved for each country (e.g.: DE - Germany, DK - Denmark, CH - Switzerland)
  - Each country may organize its domain space however it wishes

The domain name space

- For example, Australia uses `edu.au` and `com.au`
- And Britain uses `co.uk` - corporations, `ac.uk` - academic community
- And the U.S. uses states: `fl.us`, then cities: `tlh.fl.us`

Name servers

- Zones
  - A program that stores information about the domain name space is called a Domain Name Server
  - A name server generally has complete information about some part of the domain name space

Name servers

- The subspace is called a 'zone'
- The server is said to have 'authority' for one or more zones
- What is the difference between a zone and a domain?
  [A domain may be composed of one or more zones, but not vice versa]

Name servers

- Types of name servers
  - Primary master → Gets the data for its zones from flat data (text) files

Name servers

- Secondary master
  - Gets the data for its zone from another server
  - It periodically updates its local data by copying the primary master's files
  - This is called a 'zone transfer'

Name servers

- Generally keep more than one name server for any given zone (1) Redundancy: fault tolerance (2) Load: localize it as much as possible

Name service clients

- These are the clients that access name servers
- In BIND these are a set of library routines

Name service clients

☞ These are compiled (or linked via shared library) into ssh, sftp, scp, telnet, etc. so that these programs will use DNS to resolve names ("gethostbyname()" and others)

Duties of a simple resolver

- Sometimes called a 'stub resolver'
- Querying a name server
- Interpreting the response
- Resend a response
- Returning a reply to the program that it is servicing

How does the name server resolve names

☞ If the name is in the name server's zone then it can give the resolver an immediate 'authoritative' response

How does the name server resolve names

☞ If not, then the name server must search the domain name space for an answer

Root name servers

- ☞ Technical operation information about the root servers can be found at `http://www.root-servers.org/`
- ☞ The root name servers are authoritative for the top-level domains (edu, org, us, dk, etc.)

Root name servers

Operator	Location	IP
Verisign	VA	198.41.0.4
ISI	CA	192.228.79.201
		2001:478:65::53

Root name servers

Operator	Location	IP
Cogent	VA; CA; NY; IL	192.33.4.12
UMD	MD	128.8.10.90
NASA	CA	192.203.230.10

Root name servers

Operator	Location	IP
ISC	Canada; CA; NY; Spain; Hong Kong Italy; NZ Brazil; China SK; Russia; Taipei Dubai; France; Singapore Portugal; ZA	192.5.5.241 2001:500::1035

Root name servers

Operator	Location	IP
ISC (cont'd)	Tel Aviv; Jakarta	
	Germany; Japan	
	CZ; Netherlands	
	Kenya; India	
	Britain; Chile	
	Pakistan	

Root name servers

Operator	Location	IP
DOD NIC	VA	192.112.36.4
Army	MD	128.63.2.53
		2001:500:1::803f:235

Root name servers

- They can point you to the name servers for each of the top-level domains
- They, in turn can point you to their subdomains, etc. until the name is resolved

Root name servers

☞ This scheme puts a lot of importance on the root-level servers

Recursion

- ☞ The first name server can make multiple requests
- ☞ Successive requests refer the first server to another machine
- ☞ A local server generally responds to a 'recursive query'

Recursion

- ☞ This is because the local 'dumb' resolver is not smart enough to follow any referrals
- ☞ A recursive query places most of the work on a single name server

Recursion

☞ When a recursive query is made the name server is obliged to go find the answer or return an error message

Mapping addresses to names ("reverse lookups")

- What if you have an IP number and want to find the host name?
  - This is useful to make output more readable
  - Used for some security checks

Mapping addresses to names ("reverse lookups")

- ☞ This was easy with the old `/etc/hosts` tables
- ☞ The DNS data is indexed by name

Mapping addresses to names ("reverse lookups")

- Could do an exhaustive search
- The clever solution

Mapping addresses to names ("reverse lookups")

- Create a part of the domain name space that uses addresses as names

Mapping addresses to names ("reverse lookups")

- For example type:

```
# nslookup 128.186.120.2
Server:                 128.186.120.179
Address:        128.186.120.179#53

2.120.186.128.in-addr.arpa        name = diablo.cs.fsu.edu.
```

Mapping addresses to names ("reverse lookups")

Now, as you can see, newer **nslookup** versions will do this automatically. (Of course, you get many versions today of **nslookup** that announce that nslookup is "deprecated".)

Caching

- ☞ Each time a local name server processes a recursive query it learns a lot of information
- ☞ This is cached which speeds up successive queries

Caching

☞ Example:

- ➳ Say our server has already looked up the address of `eecs.berkeley.edu`
- ➳ This means it has cached the name servers for both `eecs.berkeley.edu` and `berkeley.edu`
- ➳ If we now make a query for `baobab.cs.berkeley.edu` the local server can skip the root-level query and go right to `berkeley.edu`

Caching

- time to live (TTL)
- TTL is the amount of time that information is cached before it is discarded

Caching

☞ The trade-off is between consistency and performance

☞ Remember, caching is also performed by other actors: in particular, **nscd** can cache names, and it uses its own TTLs, not those of the actual RRs. Applications such as browsers often also do a good bit of caching of ip values.

Configuring DNS: Client side

Setting up clients

☞ configure /etc/resolv.conf

```
domain cs.fsu.edu
; nu.cs.fsu.edu
nameserver  128.186.121.10
; mailer.cc.fsu.edu
nameserver  128.186.6.103
; trantor.umd.edu
nameserver  128.8.10.14
```

Overview of DNS

The client will try the nameservers in order: "nu", then "mailer", then "trantor"

- ☞ you can comment out nu and/or mailer then use nslookup and see results
- ☞ or put a bogus address in the first entry to see if the resolver tries number 2
- ☞ the changes take effect immediately

```
nslookup chi
Server:  TRANTOR.UMD.EDU
Address:  128.8.10.14

Name:    chi.cs.fsu.edu
Address:  128.186.121.20
```

The `named.conf` file

```
//
// named.conf for Red Hat caching-nameserver
//

options {
        directory "/var/named";
        dump-file "/var/named/data/cache_dump.db";
        statistics-file "/var/named/data/named_stats.txt";
        /*
         * If there is a firewall between you and nameservers you want
         * to talk to, you might need to uncomment the query-source
         * directive below.  Previous versions of BIND always asked
         * questions using port 53, but BIND 8.1 uses an unprivileged
         * port by default.
```

```
         */
         // query-source address * port 53;
};

//
// a caching only nameserver config
//
controls {
        inet 127.0.0.1 allow { localhost; } keys { rndckey; };
};

zone "." IN {
        type hint;
        file "named.ca";
};

zone "localdomain" IN {
        type master;
        file "localdomain.zone";
        allow-update { none; };
```

```
};

zone "localhost" IN {
        type master;
        file "localhost.zone";
        allow-update { none; };
};

zone "0.0.127.in-addr.arpa" IN {
        type master;
        file "named.local";
        allow-update { none; };
};

zone "0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.
         0.0.0.0.0.0.0.ip6.arpa" IN {
        type master;
        file "named.ip6.local";
        allow-update { none; };
};
```

```
zone "255.in-addr.arpa" IN {
        type master;
        file "named.broadcast";
        allow-update { none; };
};

zone "0.in-addr.arpa" IN {
        type master;
        file "named.zero";
        allow-update { none; };
};

include "/etc/rndc.key";
```

- Setting up a caching-only server: used to be more popular, now **nscd** is more popular

- Still it is very easy to do these days: **yum -y caching-nameserver**, then just turn on default installation **/etc/init.d/named start** and change `/etc/resolv.conf`

```
[root@sophie root]# nslookup
> www.yahoo.com
Server:         127.0.0.1
```

```
Address:        127.0.0.1#53

Non-authoritative answer:
www.yahoo.com   canonical name = www.yahoo.akadns.net.
Name:   www.yahoo.akadns.net
Address: 68.142.226.43
Name:   www.yahoo.akadns.net
Address: 68.142.226.45
Name:   www.yahoo.akadns.net
Address: 68.142.226.50
Name:   www.yahoo.akadns.net
Address: 68.142.226.35
Name:   www.yahoo.akadns.net
```

```
Address: 68.142.226.38
Name:    www.yahoo.akadns.net
Address: 68.142.226.39
Name:    www.yahoo.akadns.net
Address: 68.142.226.41
Name:    www.yahoo.akadns.net
Address: 68.142.226.42
>
```

Logging and named

errors: like most daemons, **named** errors (and other information) are routed through syslog, which you control wtih `/etc/syslog.conf`:

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                                 /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;news.none;authpriv.none;cron.none              /var/log/messages
```

```
# The authpriv file has restricted access.
authpriv.*                                              /var/log/secure

# Log all the mail messages in one place.
mail.*                                                  /var/log/maillog


# Log cron stuff
cron.*                                                  /var/log/cron

# Everybody gets emergency messages
*.emerg                                                 *

# Save news errors of level crit and higher in a special file.
uucp,news.crit                                          /var/log/spooler

# Save boot messages also to boot.log
local7.*                                                /var/log/boot.log

#
```

```
# INN
#
news.=crit                                        /var/log/news/news.crit
news.=err                                         /var/log/news/news.err
news.notice                                       /var/log/news/news.notice
```

And here is what you see in /var/log/messages

```
[root@sophie root]# egrep -i named /var/log/messages
Feb 14 10:18:20 sophie named[7597]: starting BIND 9.2.4 -u named -t /var/named/chroo
Feb 14 10:18:20 sophie named[7597]: using 1 CPU
Feb 14 10:18:20 sophie named: named startup succeeded
Feb 14 10:18:20 sophie named[7597]: loading configuration from '/etc/named.conf'Feb
Feb 14 10:18:20 sophie named[7597]: listening on IPv4 interface lo, 127.0.0.1#53Feb
Feb 14 10:18:20 sophie named[7597]: command channel listening on 127.0.0.1#953
Feb 14 10:18:20 sophie named[7597]: zone 0.in-addr.arpa/IN: loaded serial 42
Feb 14 10:18:20 sophie named[7597]: zone 0.0.127.in-addr.arpa/IN: loaded serial 199
Feb 14 10:18:20 sophie named[7597]: zone 255.in-addr.arpa/IN: loaded serial 42
Feb 14 10:18:20 sophie named[7597]: zone 0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0
```

```
Feb 14 10:18:20 sophie named[7597]: zone localdomain/IN: loaded serial 42
Feb 14 10:18:20 sophie named[7597]: zone localhost/IN: loaded serial 42
Feb 14 10:18:20 sophie named[7597]: running
```

DNS Security

- basic: remove HINFO fields
- "basic": limit zone transfers
- use keys and **rndc**