Domain Name System (DNS)

DNS Fundamentals
AfNOG 2006

Ayitey Bulley
extended by Phil Regnauld

Computers use IP addresses. Why do we need names?

- Names are easier for people to remember
- Computers may be moved between networks, in which case their IP address will change.

The old solution: HOSTS.TXT

- A centrally-maintained file, distributed to all hosts on the Internet

```
SPARKY                  128.4.13.9
UCB-MAILGATE            4.98.133.7
FTPHOST                 200.10.194.33
... etc
```

- This feature still exists:
  - /etc/hosts (UNIX)
  - c:\windows\system32\drivers\etc\hosts

hosts.txt does not scale

✗ Huge file (traffic and load)
✗ Name collisions (name uniqueness)
✗ Consistency
✗ Always out of date
✗ Single point of Administration
✗ Did not scale well

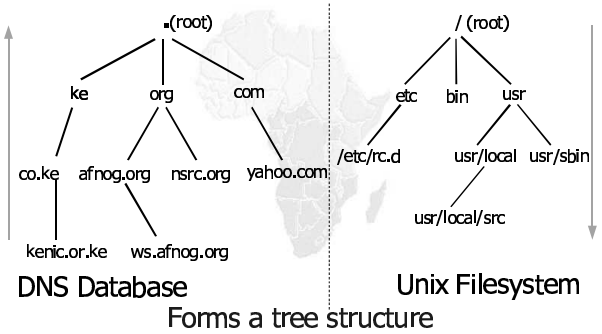
The Domain Name System was born

- Invented by Paul Mockapetris at the University of Southern California
- RFC882, 883 published in November 1983:
  - DOMAIN NAMES - CONCEPTS and FACILITIES
  - DOMAIN NAMES - IMPLEMENTATION and SPECIFICATION

The Domain Name System was born

- DNS is a distributed database for holding name to IP address (and other) information
- Distributed:
  - Shares the Administration (delegation)
  - Shares the Load (multiple NS)
- Robustness and performance achieved through:
  - replication (master/slave protocol)
  - caching of answers
- Employs a client-server architecture
- A critical piece of the Internet's infrastructure

DNS is Hierarchical



Forms a tree structure

DNS is Hierarchical (contd.)

- Globally unique names
- Administered in zones (parts of the tree)
- You can give away ("delegate") control of part of the tree underneath you
- Example:
  - afnog.org on one set of nameservers
  - ws.afnog.org on a different set
  - e1.ws.afnog.org on another set

Domain Names are (almost) unlimited

- Max 255 characters total length
- Max 63 characters in each part
  - RFC 1034, RFC 1035
- If a domain name is being used as a host name, you should abide by some restrictions
  - RFC 952 (old!)
  - a-z 0-9 and minus (-) only
  - No underscores (_)

Using the DNS

- A Domain Name (like www.ws.afnog.org) is the KEY to look up information
- The complete name is called an FQDN (Fully Qualified Domain Name)
- The result is one or more RESOURCE RECORDS (RRs)
- There are different RRs for different types of information
- You can ask for the specific type you want, or ask for "any" RRs associated with the domain name

Commonly seen Resource Records (RRs)

- A (address): map hostname to IP address
- PTR (pointer): map IP address to hostname
- MX (mail exchanger): where to deliver mail for *user@domain*
- CNAME (canonical name): map alternative hostname to real hostname
- TXT (text): any descriptive text
- NS (name server), SOA (start of authority): used for delegation and management of the DNS itself

A Simple Example

- Query: `www.afnog.org.`
- Query type: `A`
- Result:

```
www.afnog.org.    14400    IN    A       196.216.2.4
```

- In this case a single RR is found, but in general, multiple RRs may be returned.
  - (IN is the "class" for INTERNET use of the DNS, this is the default for queries)

Possible results from a Query

- Positive
  - one or more RRs found
- Negative
  - definitely no RRs match the query
- Server fail
  - cannot find the answer
- Refused
  - not allowed to query the server
- NOTE: positive and negative ARE answers, even if the server returns 0 records.

How do you use an IP address as the key for a DNS query

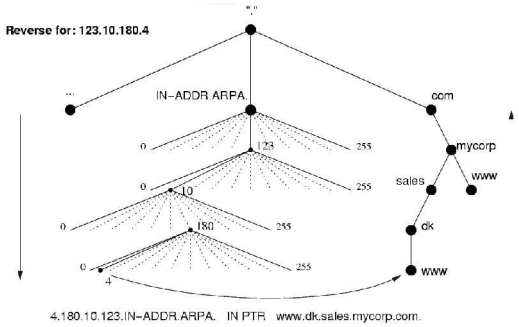
- Convert the IP address to dotted-quad
- Reverse the four parts
- Add ".in-addr.arpa." to the end; special domain reserved for this purpose

e.g. to find name for 123.10.180.4

Domain name: `4.180.10.123.in-addr.arpa.`
Query Type: `PTR`
Result: `www.dk.sales.mycorp.com.`

Known as a "reverse DNS lookup" (because we are looking up the name for an IP address, rather than the IP address for a name)

How do you use an IP address as the key for a DNS query

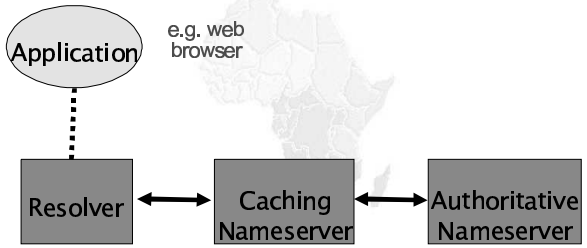


Any Questions?

DNS is a Client-Server application

- (Of course - it runs across a network)
- Requests and responses are normally sent in UDP packets, port 53
- Occasionally uses TCP, port 53
  - for very large requests (larger than 512-bytes) e.g. zone transfer from master to slave or an IPv6 AAAA (quad A) record.

There are three roles involved in DNS



Three roles in DNS

- RESOLVER
  - Takes request from application, formats it into UDP packet, sends to cache
- CACHING NAMESERVER
  - Returns the answer if already known
  - Otherwise searches for an authoritative server which has the information
  - Caches the result for future queries
  - Also known as RECURSIVE nameserver
- AUTHORITATIVE NAMESERVER
  - Contains the actual information put into the DNS by the domain owner

Three roles in DNS

- The SAME protocol is used for resolver <-> cache and cache <-> auth NS communication
- It is possible to configure a single name server as both caching and authoritative
- But it still performs only one role for each incoming query
- Common but NOT RECOMMENDED to configure in this way (we will see why later).

ROLE 1: THE RESOLVER

- A piece of software which formats a DNS request into a UDP packet, sends it to a cache, and decodes the answer
- Usually a shared library (e.g. libresolv.so under UNIX) because so many applications need it, already included in the C library in some OSes (FreeBSD for example)
- EVERY host needs a resolver - e.g. every Windows workstation has one

How does the resolver find a caching nameserver?

- It has to be explicitly configured (statically, or via DHCP etc)
- Must be configured with the IP ADDRESS of a cache (why not name?)
- Good idea to configure more than one cache, in case the first one fails

How do you choose which cache(s) to configure?

- Must have PERMISSION to use it
  - e.g. cache at your ISP, or your own
  - not every server on the Internet wants to answer your queries – think SMTP and email sending.
- Prefer a nearby cache
  - Minimises round-trip time and packet loss
  - Can reduce traffic on your external link, since often the cache can answer without contacting other servers
- Prefer a reliable cache
  - Perhaps your own?

Resolver can be configured with default domain(s) to search

- If "foo.bar" fails, then retry query as "foo.bar.mydomain.com"
- Can save typing but adds confusion
- May generate extra unnecessary traffic
- Usually best avoided – but depends on the type of organization

Example: Unix resolver configuration

/etc/resolv.conf

```
search e0.ws.afnog.org
nameserver 196.200.218.100
nameserver 196.200.222.1
```

That's all you need to configure a resolver

Testing DNS

- Just put "www.yahoo.com" in a web browser?
- Is this a good test ? Why ? What could be some problems ?

Testing DNS with "dig"

- "dig" is a program which just makes DNS queries and displays the results
- Better than "nslookup", "host" because it shows the raw information in full

```
dig ws.afnog.org.
  -- defaults to query type "A"
dig afnog.org. mx
  -- specified query type
dig @196.200.222.1 afnog.org. mx
  -- send to particular cache (overrides
     /etc/resolv.conf)
```

The trailing dot

```
dig ws.afnog.org.
```

- Prevents any default domain being appended
- Get into the habit of using it always when testing DNS
  - only on domain names, not IP addresses or e-mail addresses

```
ns# dig @147.28.0.39 www.afnog.org. a

; <<>> DiG 9.3.2 <<>> @147.28.0.39 www.afnog.org
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4620
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 2

;; QUESTION SECTION:
;www.afnog.org.                 IN      A

;; ANSWER SECTION:
www.afnog.org.          14400   IN      A       196.216.2.4

;; AUTHORITY SECTION:
afnog.org.              14400   IN      NS      rip.psg.com.
afnog.org.              14400   IN      NS      austin.gh.com.
afnog.org.              14400   IN      NS      ns-ext.isc.org.
afnog.org.              14400   IN      NS      ns-sec.ripe.net.

;; ADDITIONAL SECTION:
rip.psg.com.            77044   IN      A       147.28.0.39
austin.gh.com.          14400   IN      A       196.3.64.1

;; Query time: 708 msec
;; SERVER: 147.28.0.39#53(147.28.0.39)
;; WHEN: Wed May 10 15:05:55 2006
;; MSG SIZE  rcvd: 182
```

Understanding output from dig

- STATUS
  - NOERROR: 0 or more RRs returned
  - NXDOMAIN: non-existent domain
  - SERVFAIL: cache could not locate answer
  - REFUSED: query not available on cache server
- FLAGS
  - AA: Authoritative answer (not from cache)
  - You can ignore the others
    - QR: Query/Response (1 = Response)
    - RD: Recursion Desired
    - RA: Recursion Available
- ANSWER: number of RRs in answer

Understanding output from dig

- Answer section (RRs requested)
  - Each record has a Time To Live (TTL)
  - Says how long the cache will keep it
- Authority section
  - Which nameservers are authoritative for this domain
- Additional section
  - More RRs (typically IP addresses for the authoritative nameservers)
- Total query time
- Check which server gave the response!
  - If you make a typing error, the query may go to a default server

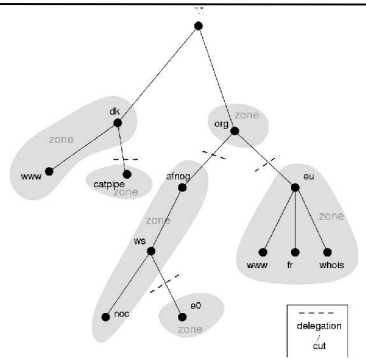
Practical Exercise

- Configure UNIX resolver
- Issue DNS queries using 'dig'
- Use tcpdump to show queries being sent to cache

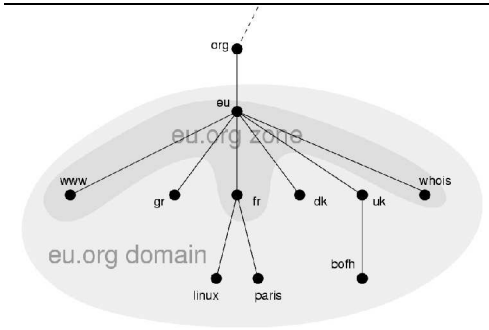
Delegation

- We mentioned that one of the advantages of DNS was that of distribution through shared administration. This is called delegation.
- We delegate when there is an administrative boundary and we want to turn over control of a subdomain to:
  - a department (within a company)
  - a company (within a TLD)
  - a country (a ccTLD)

Delegation



Delegation



Delegation

- Creating a delegation is easy, we will see this later, as time allows:
  - create the subdomain (the zone) on the server which will answer authoritatively for it (the auth NS).
  - create NS records for the zone to be delegated, pointing to the auth NS.
- That's pretty much all there is to it.