Domain Name System (DNS)

Session-1: Fundamentals

Ayitey Bulley
abulley@ghana.com

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.

HOSTS.TXT (The old solution)

- 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\hosts

What was wrong with HOSTS.TXT

✗ Traffic and load
✗ Name collisions(Name uniqueness)
✗ Consistency
✗ Allways out of date
✗ Single point of Administration
✗ Did not scale well

✔Thus DNS was born..... ;o)

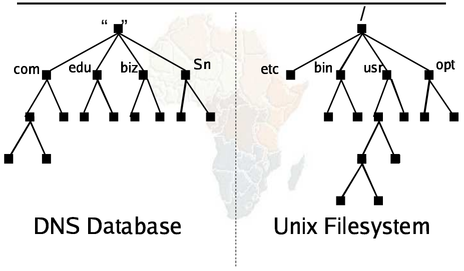
What is DNS?

- DNS is a distributed database for holding name to IP address (and other) information
  - Shares Administration
  - Shares Load
- Robustness and performance achieved through replication and caching
- Employs a client-server architecture
  - Name servers constitute the server half of the client-server mechanism
  - Resolvers constitute the client half of the client-server mechanism
- A critical piece of the Internets infrastructure

Hierarchical Structure of DNS

- Very similar to the structure of the UNIX file system
- Pictured as an inverted tree with root node at the top
- Each node in the tree has a text label
- The null label “” is reserved for the root node
- Root node is written as a single dot (.)

Hierarchical Structure of DNS (contd.)



Hierarchical Structure of DNS (contd.)

- Hostnames are globally unique
- 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
  - t1.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.ghana.com.gh) is the KEY to look up information
- 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 name
- 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.tiscali.co.uk.`
- Query type: `A`
- Result:

```
www.tiscali.co.uk.  2880  IN  A        212.74.101.10
```

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)

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)

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 193.194.185.15
Domain name: 15.185.194.193.in-addr.arpa.
Query Type: PTR
Result: ashanti.gh.com.
```

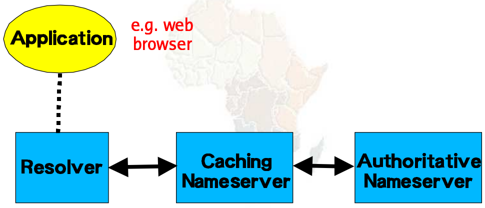
Known as a "reverse DNS lookup" (because we are looking up the name for an IP address, rather than the IP address for a name)

?

DNS is a Client-Server application

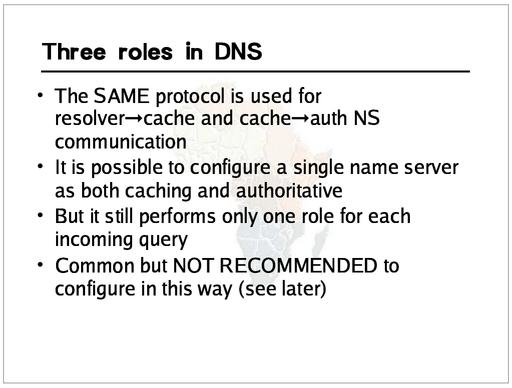
- (Of course - it runs across a network)
- Requests and responses are normally sent in UDP packets, port 53
- Occasionally uses TCP, port 53
  - for very large requests (larger than 512-bytes) e.g. zone transfer from master to slave

There are three roles involved in DNS



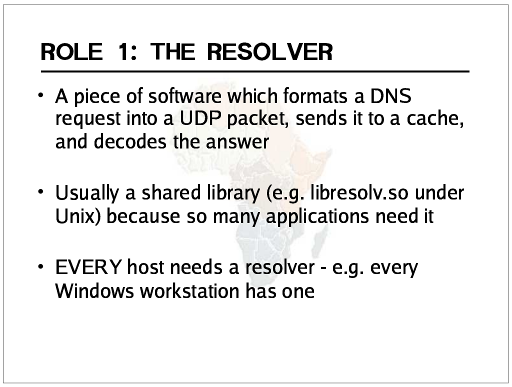
Three roles in DNS

- 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



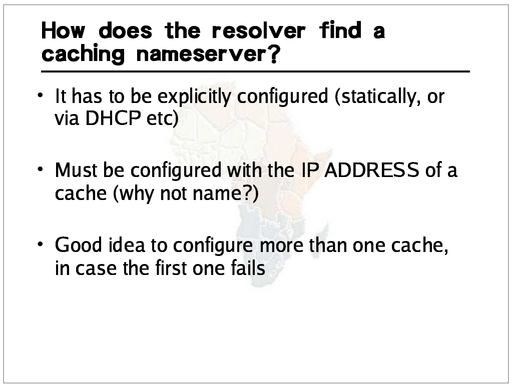
Three roles in DNS

- 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 (see later)



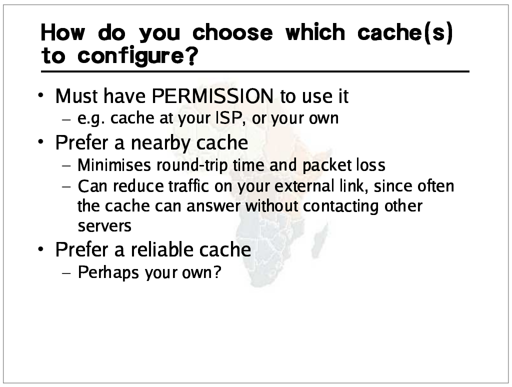
ROLE 1: THE RESOLVER

- 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
- EVERY host needs a resolver - e.g. every Windows workstation has one



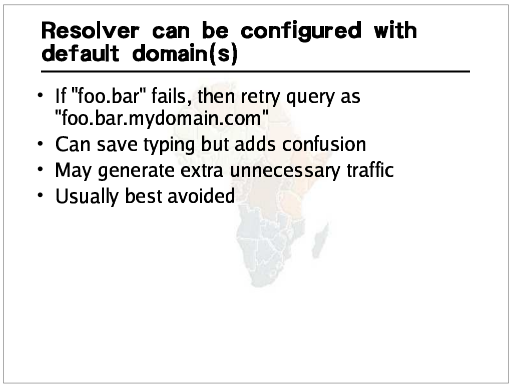
How does the resolver find a caching nameserver?

- 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



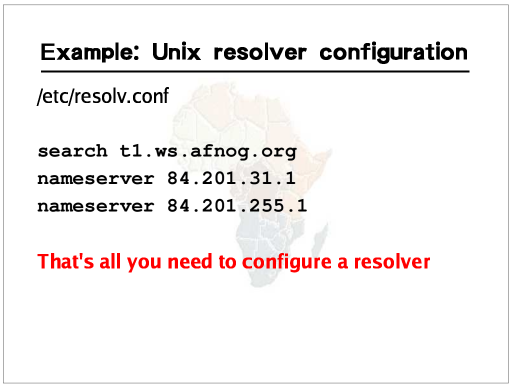
How do you choose which cache(s) to configure?

- Must have PERMISSION to use it
  - e.g. cache at your ISP, or your own
- 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)

- 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



Example: Unix resolver configuration

/etc/resolv.conf

```
search t1.ws.afnog.org
nameserver 84.201.31.1
nameserver 84.201.255.1
```

That's all you need to configure a resolver

Testing DNS

- Just put "www.yahoo.com" in a web browser?
- Why is this not a good test?

The BIND dig utility

- **Syntax**

dig [@server] domain [q-type] [other options]

Server – The server you want to use to resolve the query (defaults to servers listed in /etc/resolv.conf)

Domain - a name in the Domain Name System

q-type - is one of (a,any,mx,ns,soa,hinfo,axfr,txt,...) [default: a]

- Examples

```
# dig @84.201.255.1 ws.afnog.org. a
# dig @noc.t1.ws.afnog.org. ws.afnog.org. a
# dig @noc.ws.afnog.org. -x 84.201.31.1
# man dig
```

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 tiscali.co.uk.
  -- defaults to query type "A"
dig tiscali.co.uk. mx
  -- specified query type
dig @212.74.112.66 tiscali.co.uk. mx
  -- send to particular cache (overrides
     /etc/resolv.conf)
```

The trailing dot

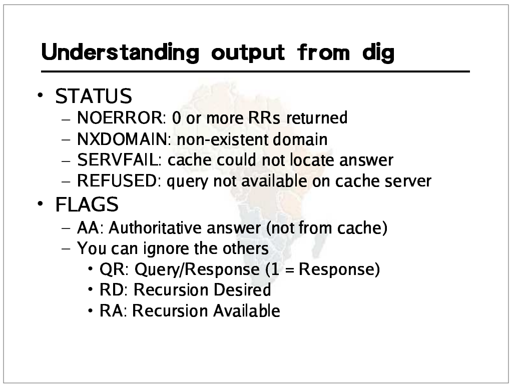
```
dig tiscali.co.uk.
```

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

```
ns# dig @84.201.31.1 www.gouv.bj a

; <<>> DiG 8.3 <<>> @84.201.31.1 www.gouv.bj a
; (1 server found)
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 4, ADDITIONAL: 3
;; QUERY SECTION:
;;      www.gouv.bj, type = A, class = IN

;; ANSWER SECTION:
www.gouv.bj.            1D IN CNAME     waib.gouv.bj.
waib.gouv.bj.           1D IN A         208.164.179.196

;; AUTHORITY SECTION:
gouv.bj.                1D IN NS        rip.psg.com.
gouv.bj.                1D IN NS        ben02.gouv.bj.
gouv.bj.                1D IN NS        nakayo.leland.bj.
gouv.bj.                1D IN NS        ns1.intnet.bj.

;; ADDITIONAL SECTION:
ben02.gouv.bj.          1D IN A         208.164.179.193
nakayo.leland.bj.       1d23h59m59s IN A  208.164.176.1
ns1.intnet.bj.          1d23h59m59s IN A  81.91.225.18

;; Total query time: 2084 msec
;; FROM: noc.t1.ws.afnog.org to SERVER: 84.201.31.1
;; WHEN: Sun Jun  8 21:18:18 2003
;; MSG SIZE  sent: 29  rcvd: 221
```

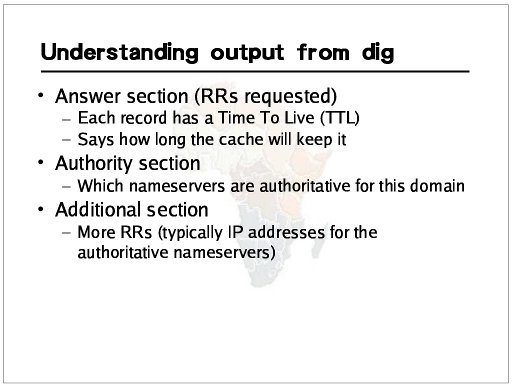
Understanding output from dig

- Queries using the dig utility outputs a lot of information, however the most important for us are
  - Status
  - Flags
  - Answer Section
  - Authority Section
  - Additional Section
  - TTL
  - Total query time
  - “From .... To .... Server” .... Section



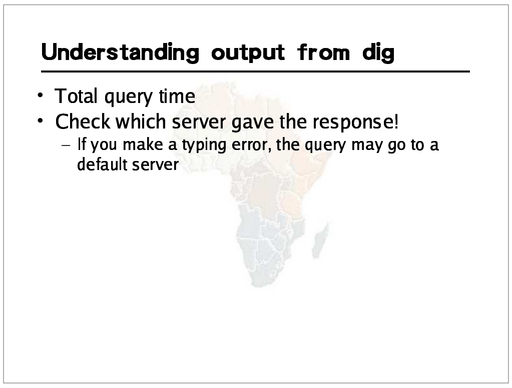
Understanding output from dig

- 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



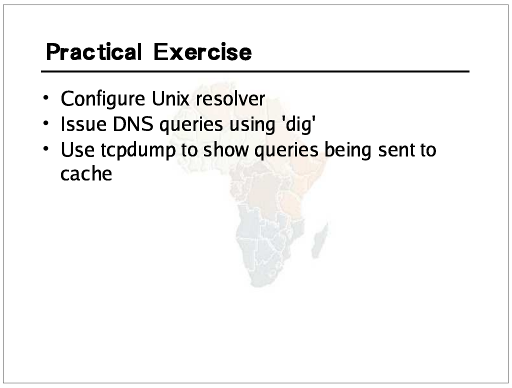
Understanding output from dig

- 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)



Understanding output from dig

- 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