Commonly seen 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

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

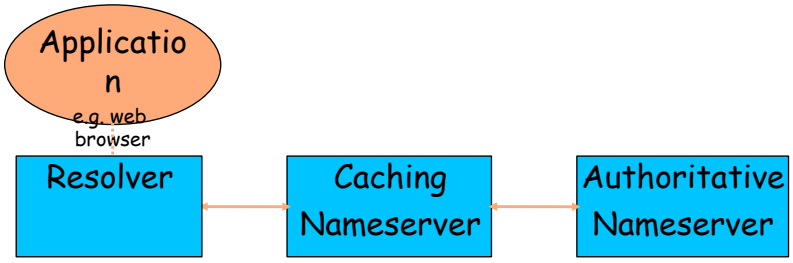
- Convert the IP address to dotted-quad
- Reverse the four parts
- Add ".in-addr.arpa" to the end (special domain reserved for this purpose)
- e.g. to find name for 212.74.101.10

```
10.101.74.212.in-addr.arpa.
            → PTR www.tiscali.co.uk.
```

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, 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

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

Example: Unix resolver configuration

- /etc/resolv.conf

```
Search cctld.or.ke
nameserver 196.216.0.21
```

- That's all you need to configure a resolver

```
# dig www.gouv.bj. a
; <<>> DiG 9.3.0 <<>> www.gouv.bj a
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2462
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 4, ADDITIONAL: 4
;; QUESTION SECTION:
;www.gouv.bj                    IN   A

;; ANSWER SECTION:
www.gouv.bj.            86400   IN   CNAME  waib.gouv.bj.
waib.gouv.bj.           86400   IN   A      81.91.232.2

;; AUTHORITY SECTION:
gouv.bj.                86400   IN   NS     rip.psg.com.
gouv.bj.                86400   IN   NS     ben02.gouv.bj.
gouv.bj.                86400   IN   NS     nakayo.leland.bj.
gouv.bj.                86400   IN   NS     ns1.intnet.bj.

;; ADDITIONAL SECTION:
ben02.gouv.bj.          86400   IN   A      81.91.232.1
nakayo.leland.bj.       18205   IN   A      81.91.225.1
ns1.intnet.bj.          18205   IN   A      81.91.225.18
rip.psg.com.            160785  IN   A      147.28.0.39
```

Interpreting the results: header

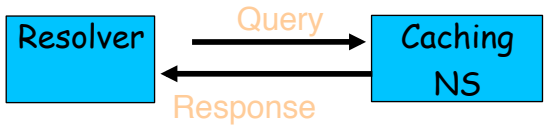
- STATUS
  - NOERROR: 0 or more RRs returned
  - NXDOMAIN: non-existent domain
  - SERVFAIL: cache could not locate answer
- FLAGS
  - AA: Authoritative answer (not from cache)
  - You can ignore the others
    - QR: Query or Response (1 = Response)
    - RD: Recursion Desired
    - RA: Recursion Available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

Interpreting the results

- 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 addrs for authoritative NS)
- Total query time
- Check which server gave the response!
  - If you made a typing error, the query may go to a default server

How caching NS works (1)

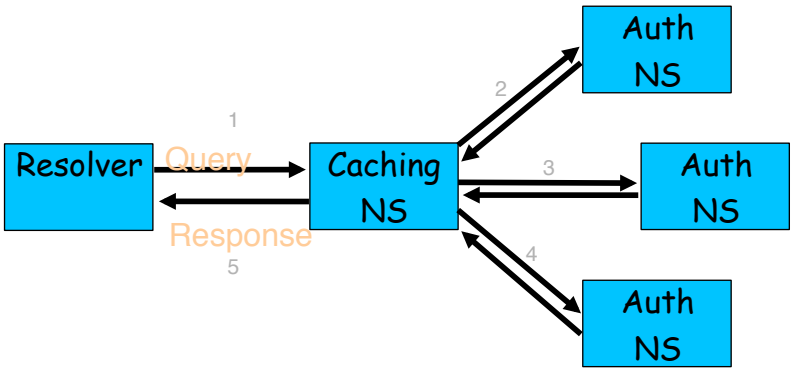
- If we've dealt with this query before recently, answer is already in the cache - easy!



What if the answer is not in the cache?

- DNS is a distributed database: parts of the tree (called "zones") are held in different servers
- They are called "authoritative" for their particular part of the tree
- It is the job of a caching nameserver to locate the right authoritative nameserver and get back the result
- It may have to ask other nameservers first to locate the one it needs

How caching NS works (2)



How does this process start?

- Every caching nameserver is seeded with a list of root servers

/usr/local/etc/named.conf

```
zone "." {
        type hint;
        file "named.root";
}
```

named.root

```
.                       3600000      NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.     3600000      A     198.41.0.4

.                       3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.     3600000      A     128.9.0.107

.                       3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.     3600000      A     192.33.4.12
;... etc
```

Distributed systems have many points of failure!

- So each zone has two or more authoritative nameservers for resilience
- They are all equivalent and can be tried in any order
- Trying stops as soon as one gives an answer
- Also helps share the load
- The root servers are very busy
  - There are currently 13 of them (each of which is a large cluster)

Caches can be a problem if data becomes stale

- If caches hold data for too long, they may give out the wrong answers if the authoritative data changes
- If caches hold data for too little time, it means increased work for the authoritative servers

The owner of an auth server controls how their data is cached

- Each resource record has a "Time To Live" (TTL) which says how long it can be kept in cache
- The SOA record says how long a negative answer can be cached (i.e. the non-existence of a resource record)
- Note: the cache owner has no control - but they wouldn't want it anyway

A compromise policy

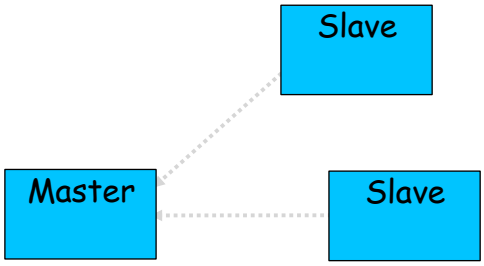
- Set a fairly long TTL - 1 or 2 days
- When you know you are about to make a change, reduce the TTL down to 10 minutes
- Wait 1 or 2 days BEFORE making the change
- After the change, put the TTL back up again

DNS Replication

- For every domain, we need more than one authoritative nameserver with the same information (RFC 2182)
- Data is entered in one server (Master) and replicated to the others (Slave(s))
- Outside world cannot tell the difference between master and slave
  - NS records are returned in random order for equal load sharing
- Used to be called "primary" and "secondary"

Slaves connect to Master to retrieve copy of zone data

- The master does not "push" data to the slaves



When does replication take place?

- Slaves poll the master periodically - called the "Refresh Interval" - to check for new data
  - Originally this was the only mechanism
- With new software, master can also notify the slaves when the data changes
  - Results in quicker updates
- The notification is unreliable (e.g. network might lose a packet) so we still need checks at the Refresh Interval

Serial Numbers

- Every zone file has a Serial Number
- Slave will only copy data when this number **INCREASES**
  - Periodic UDP query to check Serial Number
  - If increased, TCP transfer of zone data
- It is your responsibility to increase the serial number after every change, otherwise slaves and master will be inconsistent

Recommended serial number format: YYYYMMDDNN

- YYYY = year
- MM = month (01-12)
- DD = day (01-31)
- NN = number of changes today (00-99)
  - e.g. if you change the file on 5th March 2004, the serial number will be 2004030500. If you change it again on the same day, it will be 2004030501.

Configuration of Master

- /usr/local/etc/named.conf points to zone file (manually created) containing your RRs
- Choose a logical place to keep them
  - e.g. /var/cctld/master/cctld.or.ke
  - or /var/cctld/master/ke.or.cctld

```
zone "example.com" {
        type master;
        file "/var/cctld/master/example.com";
        allow-transfer { 192.188.58.126;
                         192.188.58.2; };
};
```

Configuration of Slave

- named.conf points to IP address of master and location where zone file should be created
- Zone files are transferred automatically
- Don't touch them!

```
zone "example.com" {
        type slave;
        masters { 192.188.58.126; };
        file "/var/cctld/slave/example.com";
        allow-transfer { none; };
};
```

Master and Slave

- It's perfectly OK for one server to be Master for some zones and Slave for others
- That's why we recommend keeping the files in different directories
  - /var/cctld/master/
  - /var/cctld/slave/
    - (also, the slave directory can have appropriate permissions so that the daemon can create files)

allow-transfer { ... }

- Remote machines can request a transfer of the entire zone contents
- By default, this is permitted to anyone
- Better to restrict this
- You can set a global default, and override this for each zone if required

```
options {
    allow-transfer { 127.0.0.1; };
};
```

Structure of a zone file

- Global options
  - $TTL 1d
  - Sets the default TTL for all other records
- SOA RR
  - "Start Of Authority"
  - Housekeeping information for the zone
- NS RRs
  - List all the nameservers for the zone, master and slaves
- Other RRs
  - The actual data you wish to publish

Shortcuts

- If the Domain Name does not end in a dot, the zone's own domain ("origin") is appended
- A Domain Name of "@" means the origin itself
- e.g. in zone file for example.com:
  - `@` **means** `example.com.`
  - `www` **means** `www.example.com.`

Format of the SOA record

```
$TTL 1d

@  1h  IN  SOA  ns1.example.net. brian.nsrc.org. (
             2004030300      ; Serial
             8h              ; Refresh
             1h              ; Retry
             4w              ; Expire
             1h )            ; Negative

       IN  NS  ns1.example.net.
       IN  NS  ns2.example.net.
       IN  NS  ns1.othernetwork.com.
```

Format of the SOA record

- `ns1.example.net.`
  - hostname of master nameserver
- `brian.nsrc.org.`
  - E-mail address of responsible person, with "@" changed to dot, and trailing dot
- Serial number
- Refresh interval
  - How often Slave checks serial number on Master
- Retry interval
  - How often Slave checks serial number if the Master did not respond

Format of the SOA record (cont)

- Expiry time
  - If the slave is unable to contact the master for this period of time, it will delete its copy of the zone data
- Negative / Minimum
  - Old software used this as a minimum value of the TTL
  - Now it is used for negative caching: indicates how long a cache may store the non-existence of a RR
- RIPE-203 has recommended values
  - http://www.ripe.net/ripe/docs/dns-soa.html

Format of NS records

- List all authoritative nameservers for the zone - master and slave(s)
- Must point to HOSTNAME not IP address

```
$TTL 1d

@  1h  IN  SOA  ns1.example.net. brian.nsrc.org. (
             2004030300     ; Serial
             8h             ; Refresh
             1h             ; Retry
             4w             ; Expire
             1h )           ; Negative

       IN  NS  ns1.example.net.
       IN  NS  ns2.example.net.
       IN  NS  ns1.othernetwork.com.
```

Format of other RRs

- `IN  A  1.2.3.4`
- `IN  MX  10 mailhost.example.com.`
  - The number is a "preference value". Mail is delivered to the lowest-number MX first
  - Must point to HOSTNAME not IP address
- `IN  CNAME  host.example.com.`
- `IN  PTR  host.example.com.`
- `IN  TXT  "any text you like"`