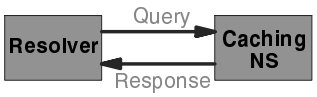
DNS Session 2: Operation of recursive (caching) nameserver

Brian Candler
Tiscali UK Ltd

How caching NS works (1)

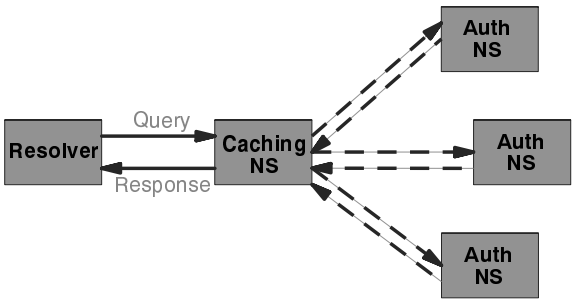


If we've dealt with this query before recently, answer is already in cache - easy!

What if the answer is not in the cache?

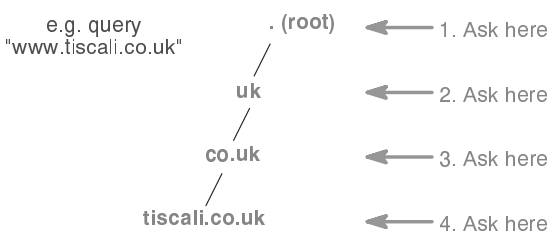
- ✓ DNS is a distributed database: parts of the tree are held in different servers
- ✓ They are called "authoritative" for their particular part of the tree
- ✓ It is the job of a caching nameserver to locate the right authoritative nameserver and get back the result
- ✓ It may have to ask other nameservers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

How caching NS works (2)



How does it know which auth nameserver to ask?

- ✓ It follows the hierarchical tree structure



Intermediate nameservers return a "NS" resource record

- ✓ "I don't have the answer, but try this other nameserver instead"
- ✓ Called a REFERRAL

Eventually this process will either:

- ✓ Find an authoritative nameserver which knows the answer (positive or negative)
- ✓ Not find any working nameserver: SERVFAIL
- ✓ End up at a faulty nameserver - either cannot answer and no further delegation, or wrong answer!

(Note: the caching nameserver may happen also to be an authoritative nameserver for the query. In that case it can answer immediately without asking anywhere else. We will talk later why it's a good idea to have separate machines for caching and authoritative nameservers)

How does this process start?

Every caching nameserver is seeded with a list of root servers

`/etc/namedb/named.conf`

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

`/etc/namedb/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
```

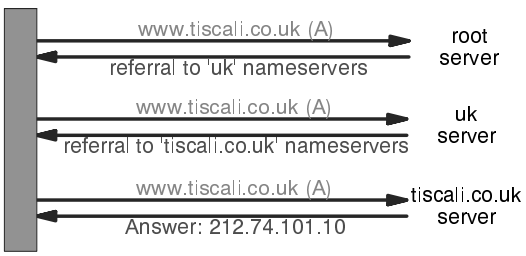

Distributed systems have many points of failure!

- ✓ 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 busy - there are about 13 of them!

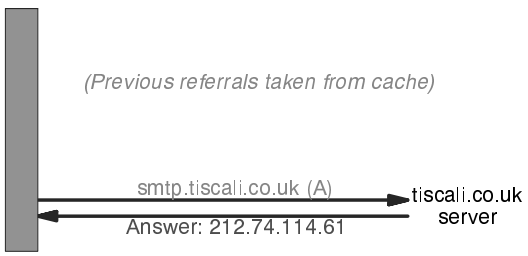
Caching reduces the load on auth nameservers

- ✓ Especially important at the higher levels: root servers, GTLD servers (.com, .net etc)
- ✓ All intermediate information is cached as well as the final answer - so NS records from REFERRALS are cached too

Example 1: www.tiscali.co.uk (on an empty cache)



Example 2: smtp.tiscali.co.uk (after previous example)



Caches can be a problem if data becomes stale

- ✓ If caches hold data for too long, they may give out 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 can control 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)

(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

What sort of problems might happen when a caching nameserver is operating?

- ✓ Remember that following referrals is in general a multi-step process
- ✓ Remember the caching

(1) One authoritative server is down or unreachable

- ✓ Not a problem: timeout and try the next authoritative server (remember that there are multiple authoritative servers for a zone, so the referral returns multiple NS records)

(2) *ALL* authoritative servers are down or unreachable!

- ✓ This is bad; query cannot complete
- ✓ Make sure all nameservers not on the same subnet (switch/router failure)
- ✓ Make sure all nameservers not in the same building (power failure)
- ✓ Make sure all nameservers not even on the same Internet backbone (failure of upstream link)
- ✓ For more detail read RFC 2182

(3) Referral points to a nameserver which is not authoritative for this zone

- ✓ Bad error. Called "Lame Delegation"
- ✓ Query cannot proceed - server does not have either the right answer or the right delegation
- ✓ Typical error: NS record points to a caching nameserver which has not been set up as authoritative for that zone
- ✓ Or: syntax error in zone file means that nameserver software ignores it

(4) Inconsistencies between authoritative servers

- ✓ If auth servers don't have the same information then you will get different information depending on which one you picked (random)
- ✓ Because of caching, these problems can be very hard to debug. Problem is intermittent.

(5) Inconsistencies in delegations

- ✓ NS records in the delegation do not match NS records in the zone file (we will write zone files later)
- ✓ Which is right?

(6) Mixing caching and authoritative nameservers

- ✓ If caching nameserver contains an old zone file, but customer has transferred their DNS somewhere else
- ✓ Caching nameserver responds immediately with the old information, even though NS records point at a different ISP's authoritative nameservers which hold the right information!

(7) Inappropriate choice of parameters

- ✓ e.g. TTL set either far too short or far too long

These problems are not the fault of the caching server!

- ✓ They all originate from bad configuration of the AUTHORITATIVE name servers
- ✓ Many of these mistakes are easy to make but difficult to debug, especially because of caching
- ✓ Running a caching server is easy: running authoritative nameservice properly requires great attention to detail

How to debug these problems?

- ✓ We must bypass caching
- ✓ We must try *all* N servers for a zone (a caching nameserver stops after one)
- ✓ We must bypass recursion to test all the intermediate referrals
- ✓ "dig +norec" is your friend

```
dig +norec @1.2.3.4 foo.bar. a
```

Server to query | Domain | Query Type

25

How to interpret responses (1)

- ✓ Look for "status: NOERROR"
- ✓ "flags .... **aa**" means this is an Authoritative Answer (i.e. not cached)
- ✓ "ANSWER SECTION" gives the answer
- ✓ If you get back just NS records: it's a referral

```
;; ANSWER SECTION
foo.bar.  1H IN A  1.2.3.4
```

domain name | TTL | answer

2

How to interpret responses (2)

- ✓ If you get back "status: NXDOMAIN" it's a successful negative answer (i.e. the requested RR definitely does not exist). You should get a SOA record as well.
- ✓ "status: SERVFAIL" means all authoritative servers dead (seen only for recursive lookups)

27

How to debug a domain using "dig +norec" (1)

- ✓ 1. Start at any root server

```
dig +norec @a.root-servers.net. www.tiscali.co.uk. a
```

2. For a referral, note the NS records returned
3. Repeat the query for *all* NS records.
4. Go back to step 2, until you have got the final answers to the query

2

How to debug a domain using "dig +norec" (2)

- ✓ 5. Check all the answers have "flags: aa" and that answers from a group of authoritative nameservers are consistent with each other

6. Note that NS records are names not IP addresses. So now check every NS record maps to the correct IP address using the same process!

29

How to debug a domain using "dig +norec" (3)

- ✓ Tedious, requires patience and accuracy, but it pays off
- ✓ Learn this first before playing with more automated tools, e.g. http://zonecheck.nic.fr/v2/

3