

# Domain Name System (DNS)



## Session 2: Resolver Operation and debugging

Michuki Mwangi

AfNOG Workshop, AIS 2018, Dakar

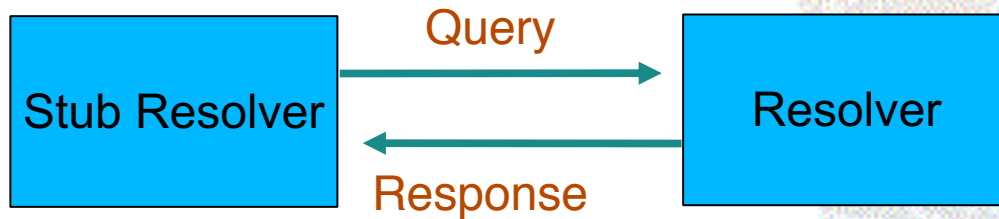


# DNS Resolver Operation



# How Resolvers Work (1)

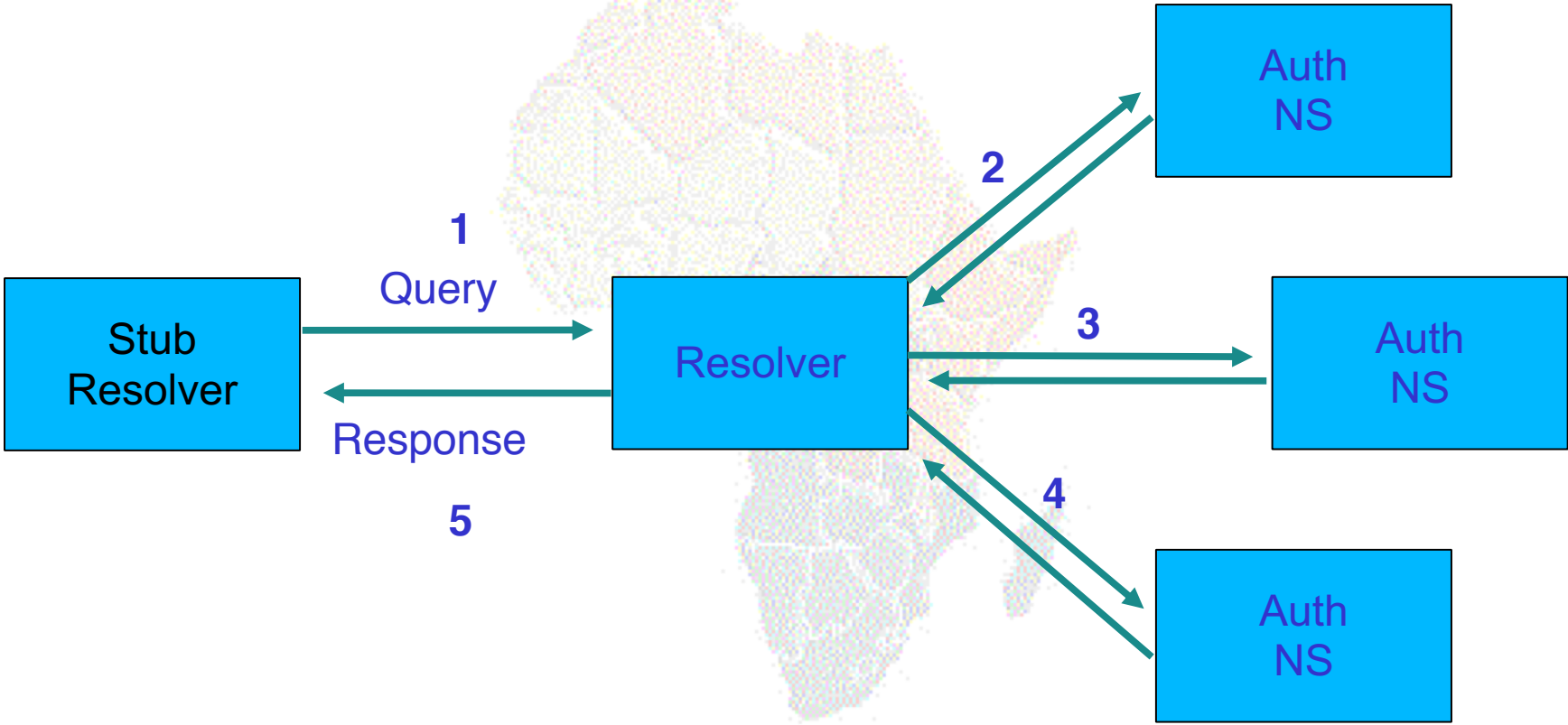
- If we've dealt with this query before recently, answer is already in the cache - easy!



# What if the answer is not in the cache?

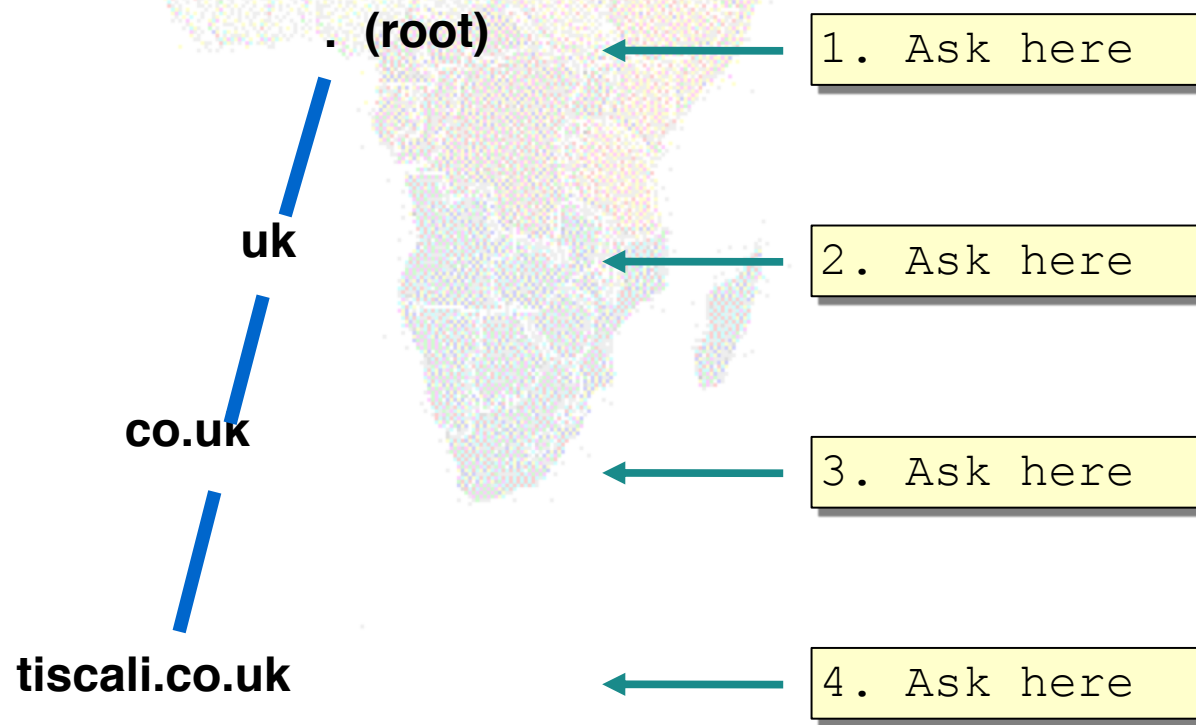
- DNS is a distributed database: parts of the tree (called "zones") are held in different servers
- They are called "authoritative" for their particular part of the tree
- It is the job of a caching nameserver to locate the right authoritative nameserver and get back the result
- It may have to ask other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

# How caching NS works (2)



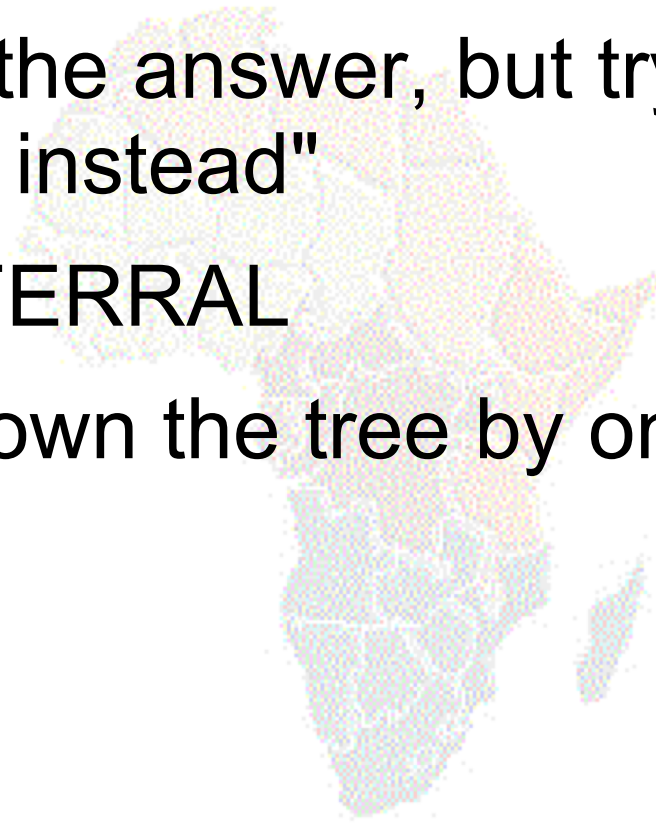
# How does it know which authoritative nameserver to ask?

- It follows the hierarchical tree structure
- e.g. to query "www.tiscali.co.uk"



# Intermediate nameservers return "NS" resource records

- "I don't have the answer, but try these other nameservers instead"
- Called a REFERRAL
- Moves you down the tree by one or more levels



# 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 resolver may happen also to be an authoritative nameserver for a particular query. In that case it will answer immediately without asking anywhere else. We will see later why it's a better 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/unbound/unbound.conf.d/root-hints.conf
```

```
server:  
  root-hints: /var/lib/unbound/named.root
```

```
/var/lib/unbound/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
```



# Where did named.root come from?

- `ftp://ftp.internic.net/domain/named.cache`
- Worth checking every 6 months or so for updates





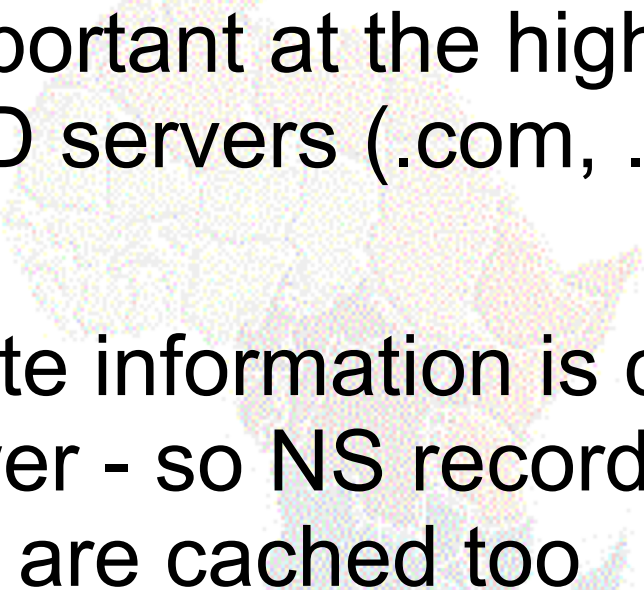
# Demonstration

- `dig +trace www.tiscali.co.uk.`
- Instead of sending the query to the cache, "dig +trace" traverses the tree from the root and displays the responses it gets
  - `dig +trace` is a bind 9 feature
  - useful as a demo but not for debugging

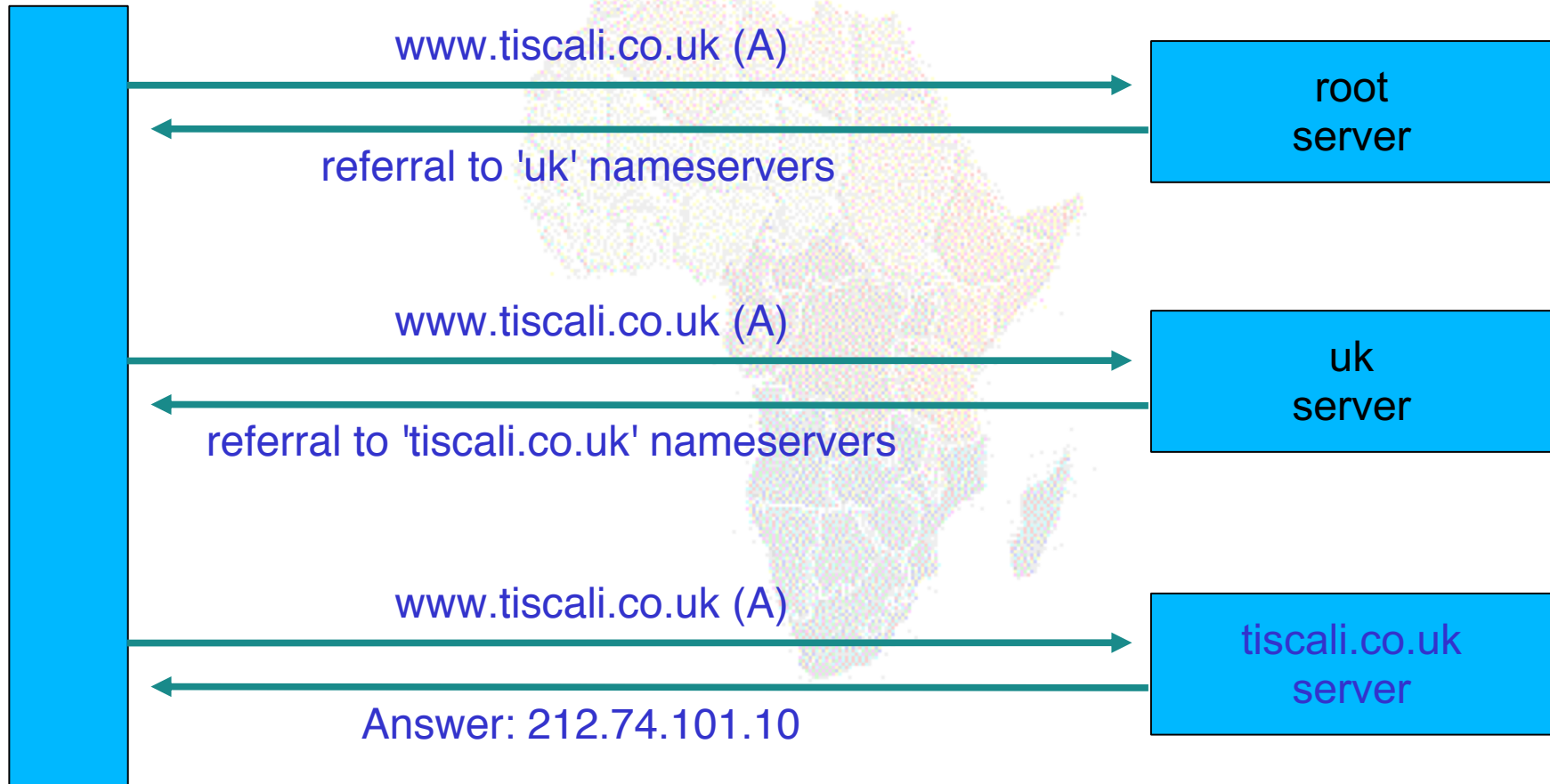
# Distributed systems have many points of failure!

- 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
  - Individual root servers are distributed all over the place using anycast

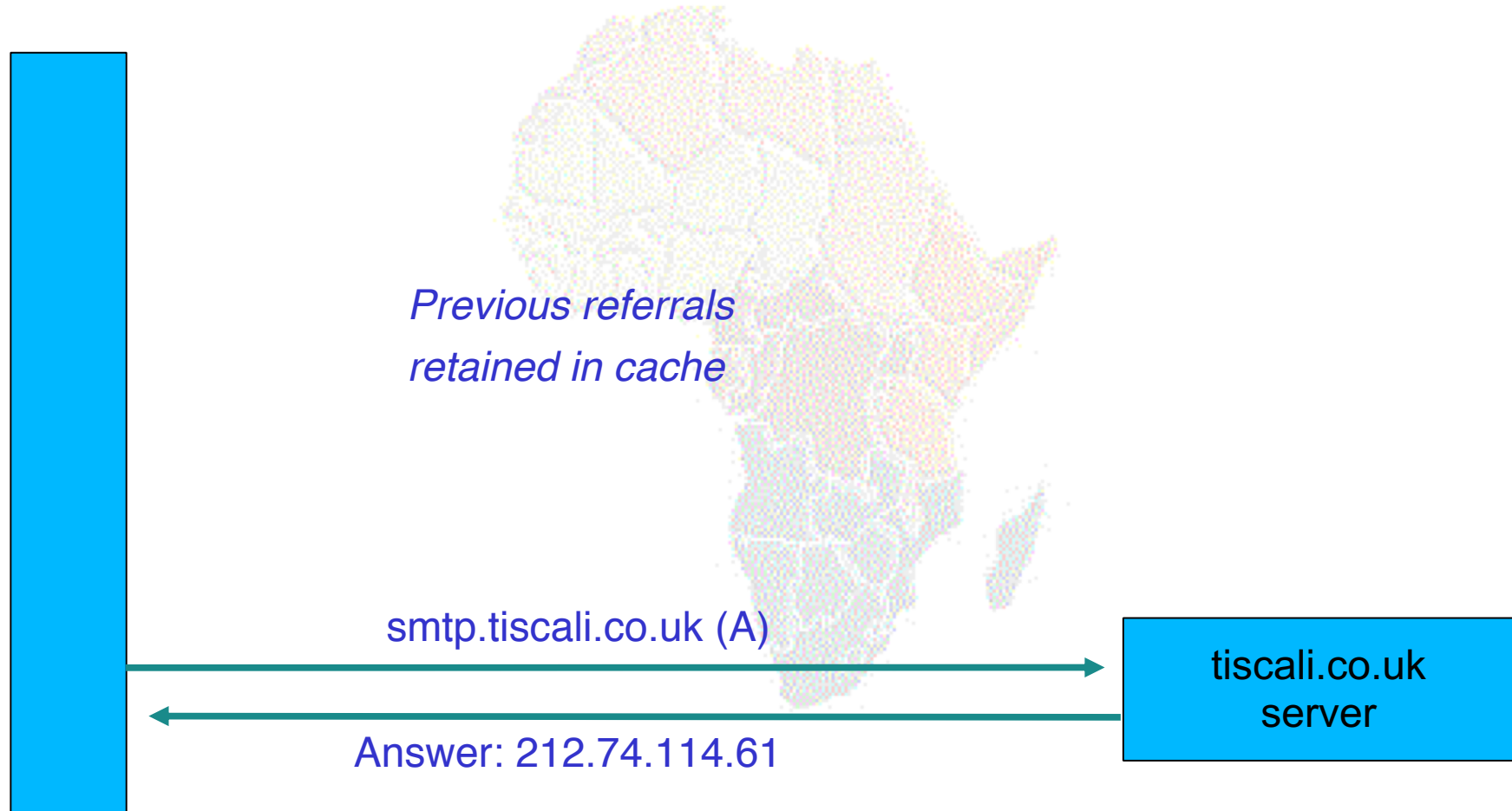
# Caching reduces the load on auth nameservers

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

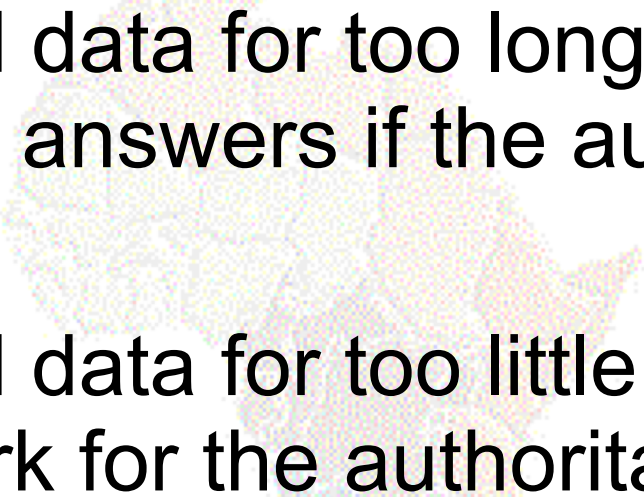
# Example 1: www.tiscali.co.uk (on an empty cache)



# 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 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



Any questions?





# DNS Debugging

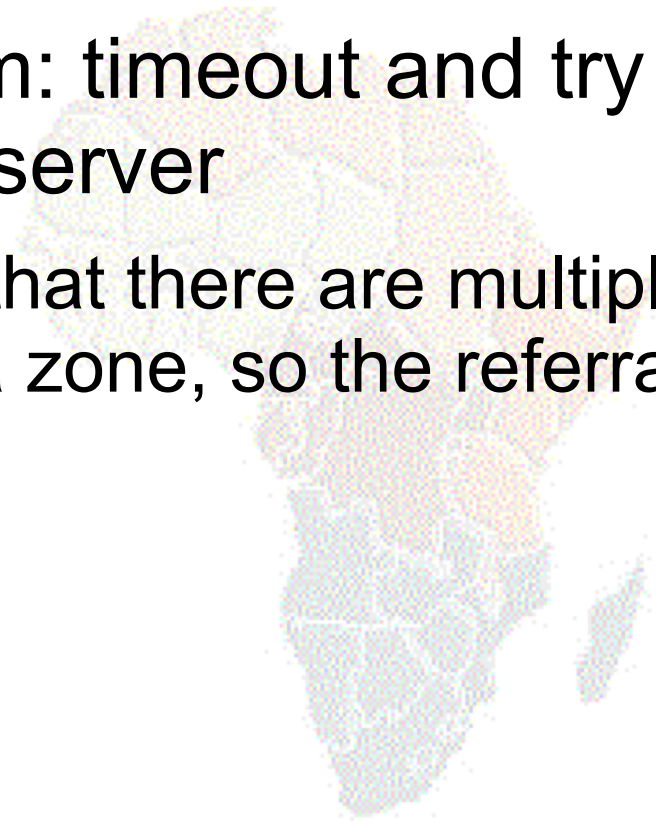
# What sort of problems might occur when resolving names in DNS?

- 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 to a nameserver which is not authoritative for this zone

- Bad error. Called "Lame Delegation"
- Query cannot proceed - server can give neither the right answer nor the right delegation
- Typical error: NS record for a zone 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)
- Problem: if the two sets aren't the same, then which is right?
  - Leads to unpredictable behaviour
  - Caches could use one set or the other, or the union of both



## (6) Mixing caching and authoritative nameservers

- Consider when 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!
- This is a very strong reason for having separate machines for authoritative and caching NS
  - Another reason is that an authoritative-only NS has a fixed memory usage

# (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 resolver!

- 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 resolver is easy; running authoritative nameservice properly requires great attention to detail
- But nothing makes the helpdesk phone ring quite like a broken resolver

# 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

# 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.      3600    IN      A       1.2.3.4
```

Domain name

TTL

Answer

# How to interpret responses (2)

- *"status: NXDOMAIN"*
  - OK, negative (the name does not exist). You should get back an SOA
- *"status: NOERROR"* with an empty answer section
  - OK, negative (name exists but no RRs of the type requested). Should get back an SOA
- Other status may indicate an error
- Look also for *Connection Refused* (DNS server is not running or doesn't accept queries from your IP address) or *Timeout* (no answer)



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

1. Start at any root server: [a-m].root-servers.net

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

*Remember the trailing dots!*

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

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

1. Check all the results from a group of authoritative nameservers are consistent with each other
2. Check all the final answers have "flags: aa"
3. Note that the NS records point to names, not IP addresses. So now check every NS record seen maps to the correct IP address using the same process!!



# 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
  - Such as:
    - <http://www.squish.net/dnscheck/>
    - <http://www.zonecheck.fr/>
  - These tools all have limitations, none is perfect

# Practical



***Worked examples***

# Building your own resolver

- We will be using unbound, software written by NLNet Labs, [www.nlnetlabs.nl](http://www.nlnetlabs.nl)
  - There are other options, e.g. BIND9
- Unbound is a dedicated resolver, and runs on most server operating systems
  - Debian: `apt-get install unbound`
- Question: what sort of hardware would you choose when building a resolver?

# Improving the configuration

- Limit client access to your own IP addresses only
  - No reason for other people on the Internet to be using your cache resources

```
/etc/unbound/unbound.conf.d/clients.conf
```

```
server:  
  access-control: 197.4.137.0/24 allow  
  access-control: 2001:43f8:220:219::/64 allow
```

# Managing a resolver

- `# service unbound start`
- `# unbound-control status`
- `# unbound-control reload`
  - After config changes; causes less disruption than restarting the daemon
- `# unbound-control dump_cache`
  - dumps current cache contents to standard out (redirect to a file if you want the output in a file)
- `# unbound-control flush .`
  - Destroys the cache contents from the root all the way down; don't do on a live system!

# Absolutely critical!

- `tail /var/log/daemon.log`
  - after any nameserver changes and reload/restart
- A syntax error may result in a nameserver which is running, but not in the way you wanted
- check your log files

# Practical

- Build a resolver
- Examine its operation

