DNS Session 1: Fundamentals

Based on Brian Candler's materials

ISOC CCTLD workshop

Computers use IP addresses. Why do we need names?

- Easier for people to remember
  - Especially true for IPv6
- Computers may be moved between networks, in which case their IP address will change

Old solution: hosts.txt

- A centrally-maintained file, distributed to all hosts on the Internet
- This feature still exists
  - `/etc/hosts` [Unix]
  - `c:\windows\system32\drivers\etc\hosts` [Windows]

```
128.4.13.9          SPARKY
4.98.133.7          UCB-MAILHOST
200.10.194.33       FTPHOST
```

hosts.txt doesn't scale

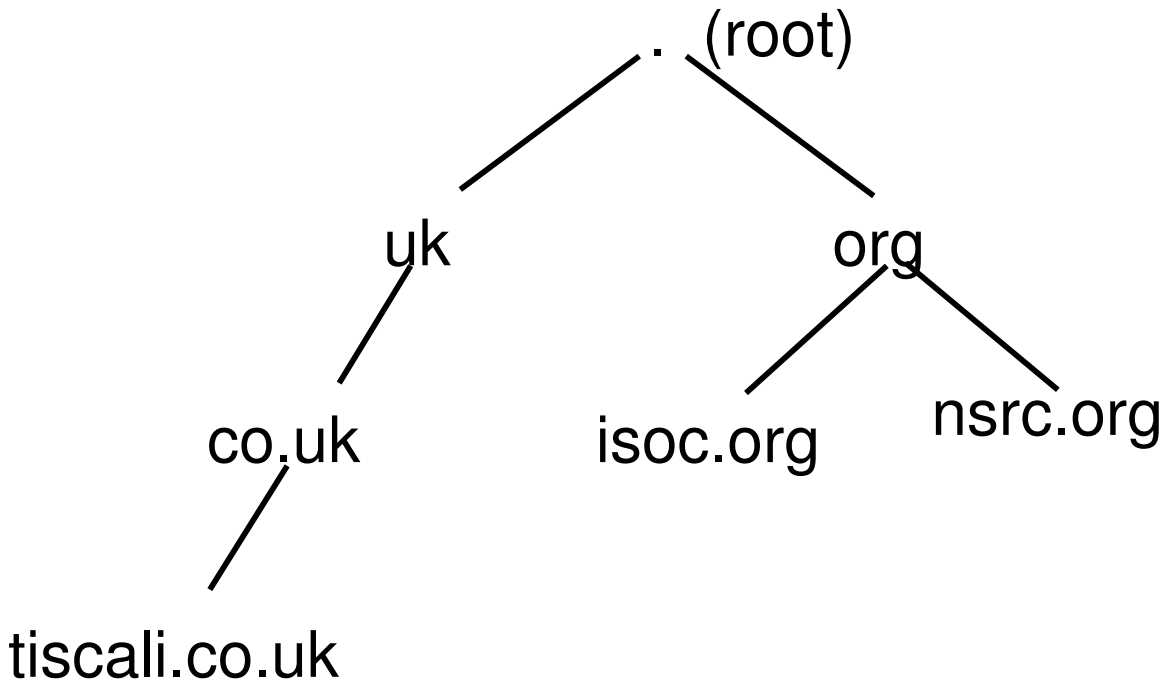
- ✗ Huge file
- ✗ Needs frequent copying to ALL hosts
- ✗ Consistency
- ✗ Always out-of-date
- ✗ Name uniqueness
- ✗ Single point of administration

The domain name system was born

- DNS is a Distributed Database for holding name to IP address (and other) information
- Distributed:
  - Shares the administration
  - Shares the load
- Robustness and performance through:
  - Replication
  - Caching
- A *critical piece of Internet infrastructure*

DNS is Hierarchical

- Forms a tree structure



DNS is Hierarchical (2)

- Gives globally unique names
- Administered in "zones" (parts of the tree)
- You can give away ("delegate") control of part of the tree underneath you
- Example:
  - isoc.org on one set of nameservers
  - dnsws.isoc.org on a different set
  - foobar.dnsws.isoc.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.tiscali.co.uk) 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 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

Simple example

- Query: www.tiscali.co.uk
- Query type: A
-

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

- In this case just 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

- Positive
  - one or more RRs found
- Negative
  - definitely no RRs match the query
- Server fail
  - cannot contact anyone who knows the answer

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.
```

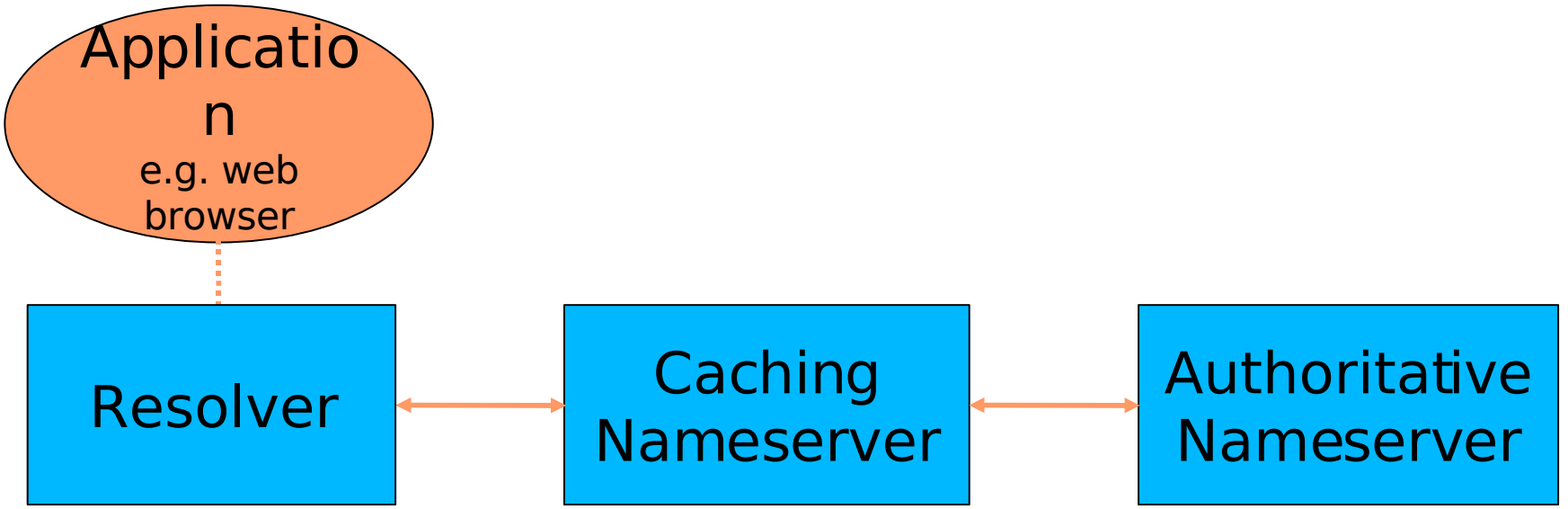
Any questions?

?

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, 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↔authoritative NS communication
- It is possible to configure a single nameserver 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
  - Can you run one better than your ISP?

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 cctld.or.ke
nameserver 196.216.0.21
```

- 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?

Testing DNS with "dig"

- "dig" is a program which just makes DNS queries and displays the results
  - Better for debugging than "nslookup" and "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 specific 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
  - but only on domain names, not IP addresses or E-mail addresses

```
# 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

;; Query time: 200 msec
;; SERVER: 212.74.112.67#53(212.74.112.67)
;; WHEN: Tue Dec 28 19:50:01 2004
;; MSG SIZE  rcvd: 237
```

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

Practical Exercise

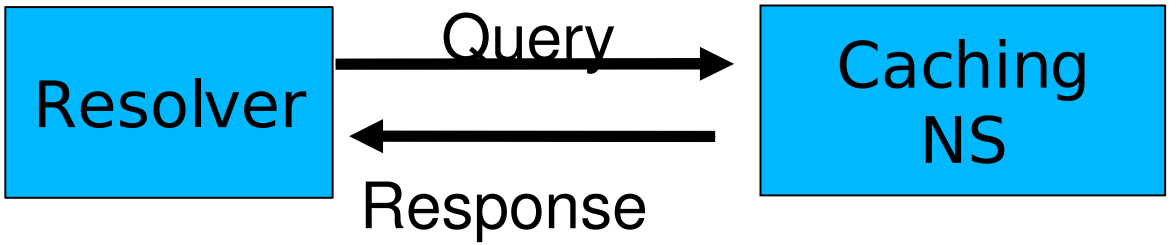
- dig
- download and install BIND
- rndc config

DNS Session 2: DNS cache operation and DNS debugging

Based on Brian Candler's materials

ISOC CCTLD workshop

How caching NS works (1)

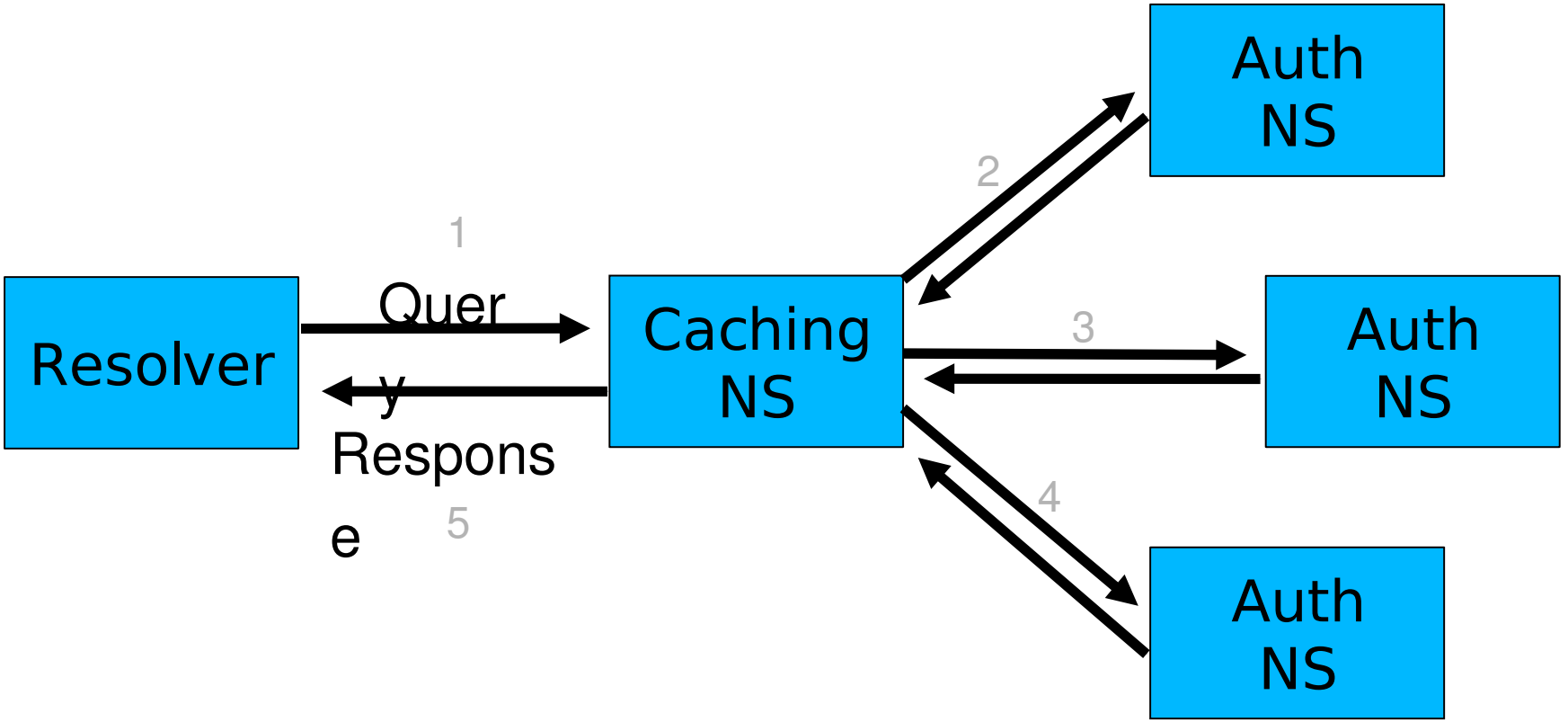


- If we've dealt with this query before recently, answer is already in the cache - easy!

What if the answer is not in the cache?

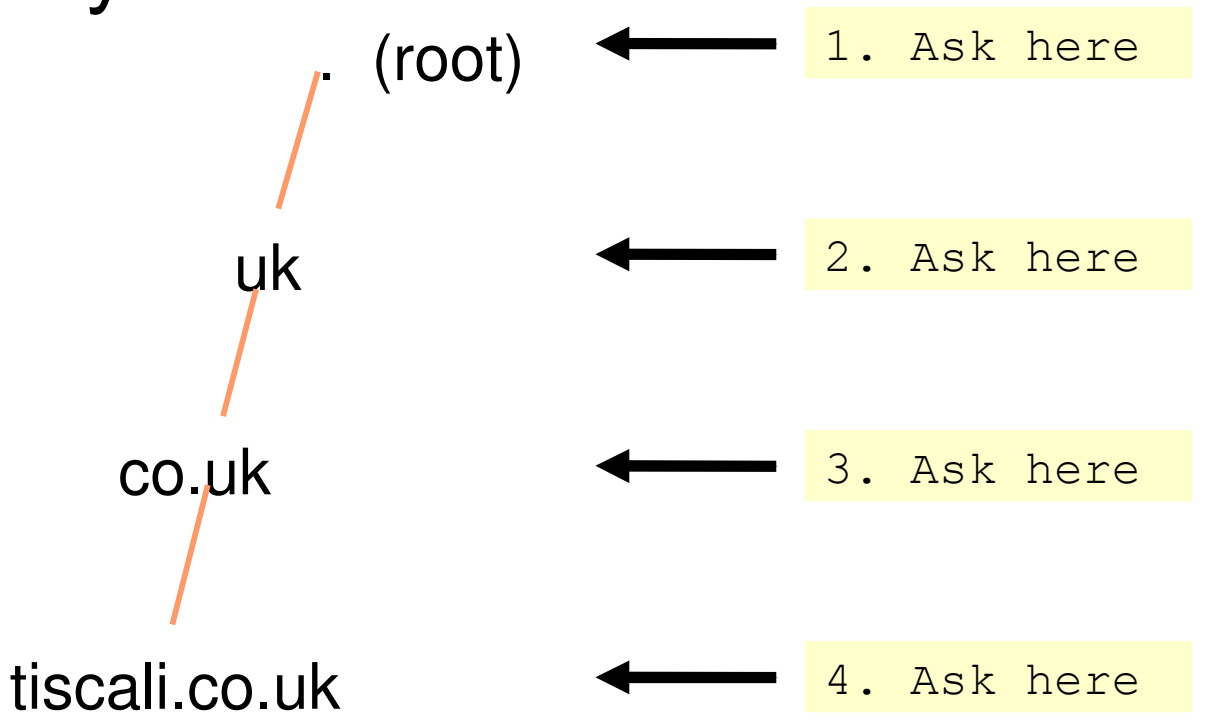
- DNS is a distributed database: parts of the tree (called "zones") are held in different servers
- They are called "authoritative" for their particular part of the tree
- It is the job of a caching nameserver to locate the right authoritative nameserver and get back the result
- It may have to ask other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

How caching NS works (2)



How does 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 caching nameserver 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 se

/etc/bind/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
```

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

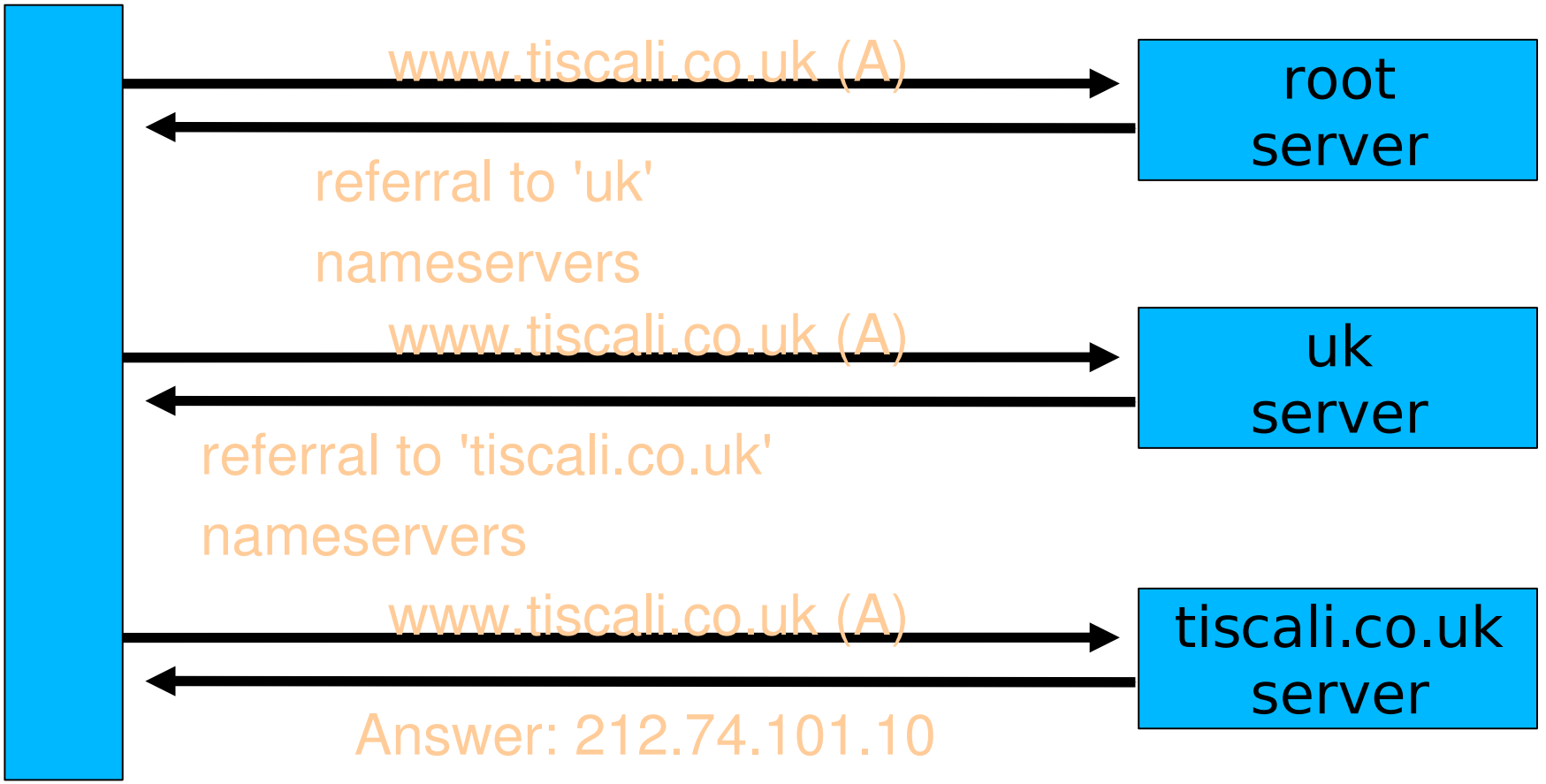
Distributed systems have many points of failure!

- 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 (each of which is a large cluster)

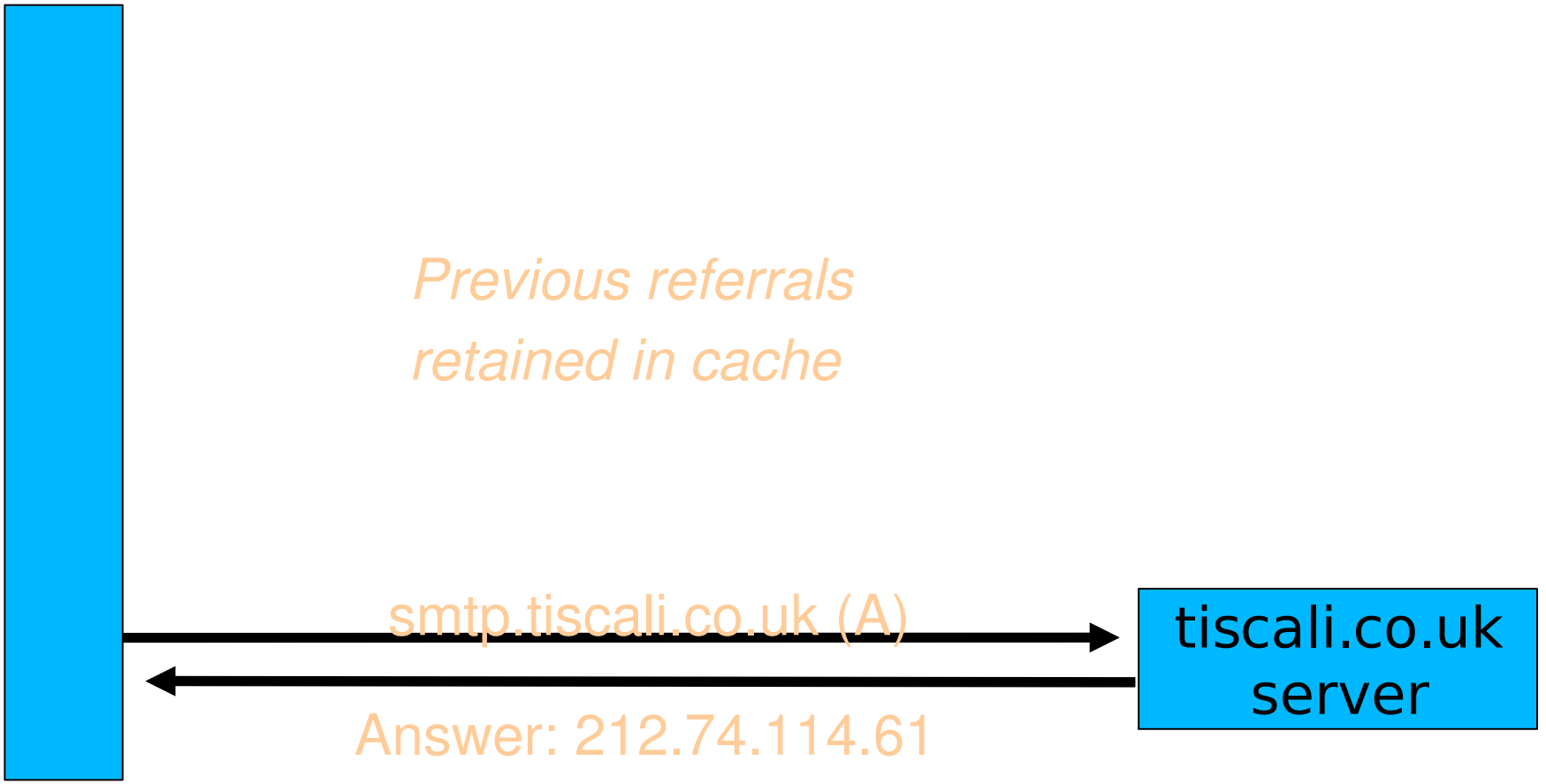
Caching reduces the load on auth nameservers

- 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?

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

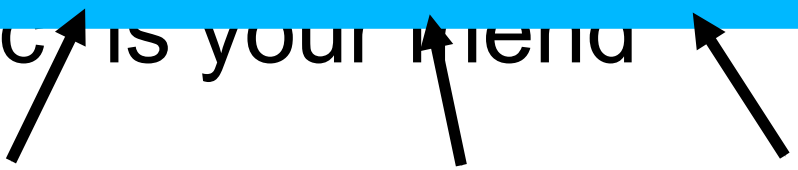
These problems are not the fault of the caching server!

- 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

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
- 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 domain does not exist). You should get back an SOA
- *"status: NOERROR" w*ith zero RRs
  - OK, negative (domain 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 +norec" (1)

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

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

Remember the trailing dots!

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

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://dnsecheck.se/
  - These tools all have limitations, none is perfect

Practical

Debugging domain with dig

checking domain with

http://www.squish.net/dnscheck/

http://dnsecheck.se/

DNS Session 3: Configuration of Authoritative Nameservice

Based on Brian Candler's materials

ISOC CCTLD workshop

Recap

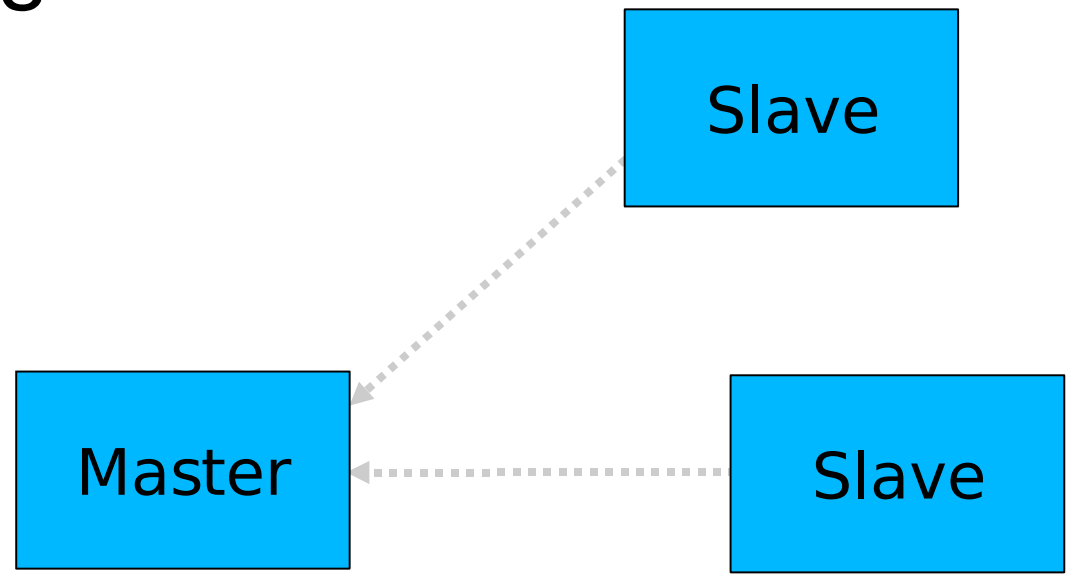
- DNS is a distributed database
- Resolver asks Cache for information
- Cache traverses the DNS delegation tree to find Authoritative nameserver which has the information requested
- Bad configuration of authoritative servers can result in broken domains

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.

Serial Numbers: Danger 1

- If you ever *decrease the serial number, the slaves will never update again until the serial number goes above its previous value*
- *RFC1912 section 3.1 explains a method to fix this problem*
- *At worst, you can contact all your slaves and get them to delete their copy of the zone data*

Serial Numbers: Danger 2

- Serial no. is a 32-bit unsigned number
- Range: 0 to 4,294,967,295
- Any value larger than this is silently truncated
- e.g. 20040305000 (note extra digit)

  = 4AA7EC968 (hex)

  = AA7EC968 (32 bits)

  = 2860435816
- If you make this mistake, then later correct it, the serial number will have decreased

Configuration of Master

- /usr/local/etc/named.conf points to zone file (manually created) containing your RRs
- Choose a logical place to keep them

```
zone "example.com" {
        type master;
        file "/etc/bind/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

```
zone "example.com" {
        type slave;
        masters { 192.188.58.126; };
        file "/etc/bind/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
  - /etc/bind/master/
  - /etc/bind/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

```
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

Format of a Resource Record

```
www        3600   IN        A        212.74.112.80
Domain     TTL    Class     Type     Data
```

- One per line (except SOA can extend over several lines)
- If you omit the Domain Name, it is the same as the previous line
- TTL shortcuts: e.g. 60s, 30m, 4h, 1w2d
- If you omit the TTL, uses the $TTL default value
- If you omit the Class, it defaults to IN
- Type and Data cannot be omitted
- Comments start with SEMICOLON (;)

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.`

If you write this...

```
$TTL 1d
@                               SOA ( ... )
                                NS  ns0
                                NS  ns0.as9105.net.

; Main webserver
www                             A   212.74.112.80
                                MX  10 mail
```

... it becomes this

```
example.com.        86400   IN  SOA ( ... )
example.com.        86400   IN  NS  ns0.example.com.
example.com.        86400   IN  NS  ns0.as9105.net.
www.example.com.    86400   IN  A   212.74.112.80
www.example.com.    86400   IN  MX  10 mail.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"`

When you have added or changed a zone file:

- Remember to increase the serial number!
- `named-checkzone example.com \ /etc/bind/master/example.com`
  - bind 9 feature
  - reports zone file syntax errors; correct them!
- `named-checkconf`
  - reports errors in named.conf
- `rndc reload`
  - or: `rndc reload example.com`
- `tail /var/log/messages`

These checks are ESSENTIAL

- If you have an error in named.conf or a zone file, named may continue to run but will not be authoritative for the bad zone(s)
- You will be lame for the zone without realising it
- Slaves will not be able to contact the master
- Eventually (e.g. 4 weeks later) the slaves will expire the zone
- Your domain will stop working

Other checks you can do

- **`dig +norec @x.x.x.x example.com. soa`**
  - Check the AA flag
  - Repeat for the master and all the slaves
  - Check the serial numbers match
- **`dig @x.x.x.x example.com. axfr`**
  - "Authority Transfer"
  - Requests a full copy of the zone contents over TCP, as slaves do to master
  - This will only work from IP addresses listed in the allow-transfer {...} section

So now you have working authoritative nameservers!

- But none of this will work until you have delegation from the domain above
- That is, they put in NS records for your domain, pointing at your nameservers
- You have also put NS records within the zone file
- The two sets should match

Any questions?

?

TOP TEN ERRORS in authoritative nameservers

- All operators of auth nameservers should read RFC 1912
  - Common DNS Operational and Configuration Errors
- And also RFC 2182
  - Selection and Operation of Secondary DNS servers

1. Serial number errors

- Forgot to increment serial number
- Incremented serial number, then decremented it
- Used serial number greater than 2^{32}
- Impact:
  - Slaves do not update
  - Master and slaves have inconsistent data
  - Caches will sometimes get the new data and sometimes old - intermittent problem

2. Comments in zone files starting '#' instead of ';'

- Syntax error in zone file
- Master is no longer authoritative for the zone
- Slaves cannot check SOA
- Slaves eventually expire the zone, and your domain stops working entirely
- Use "named-checkzone"
- Use "tail /var/log/messages"

3. Other syntax errors in zone files

- e.g. omitting the preference value from MX records
- Same impact

4. Missing the trailing dot

```
; zone example.com.
@   IN   MX 10   mailhost.example.com
```

becomes

```
@   IN   MX 10   mailhost.example.com.example.com.
```

```
; zone 2.0.192.in-addr.arpa.
1   IN   PTR     host.example.com
```

becomes

```
1   IN   PTR     host.example.com.2.0.192.in-addr.arpa.
```

5. NS or MX records pointing to IP addresses

- They must point to hostnames, not IP addresses
- Unfortunately, a few mail servers *do accept IP addresses in MX records, so you may not see a problem with all remote sites*

6. Slave cannot transfer zone from master

- Access restricted by allow-transfer {...} and slave not listed
- Or IP filters not configured correctly
- Slave will be lame (non-authoritative)

7. Lame delegation

- You cannot just list any nameserver in NS records for your domain
- You must get agreement from the nameserver operator, and they must configure it as a slave for your zone
- At best: slower DNS resolution and lack of resilience
- At worst: intermittent failures to resolve your domain

8. No delegation at all

- You can configure "example.com" on your nameservers but the outside world will not send requests to them until you have delegation
- The problem is hidden if your nameserver is acting both as your cache and as authoritative nameserver
- Your own clients can resolve www.example.com, but the rest of the world cannot

9. Out-of-date glue records

- See later

10. Not managing TTL correctly during changes

- e.g. if you have a 24 hour TTL, and you swing www.example.com to point to a new server, then there will be an extended period when some users hit one machine and some hit the other
- Follow the procedure:
  - Reduce TTL to 10 minutes
  - Wait at least 24 hours
  - Make the change
  - Put the TTL back to 24 hours

Practical

- Create a new domain
- Set up master and slave nameservice
- Obtain delegation from the domain above
- Test it

DNS Session 4: Delegation

Based on Brian Candler's materials

ISOC CCTLD workshop

How do you delegate a subdomain?

- In principle straightforward: just insert NS records for the subdomain, pointing at someone else's servers
- If you are being careful, you should first *check that those servers are authoritative for the subdomain*
  - *by using "dig +norec" on all the servers*
- *If the subdomain is managed badly, it reflects badly on you!*
  - *and you don't want to be fielding problem reports when the problem is somewhere else*

Zone file for "example.com"

```
$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.

; My own zone data
       IN  MX  10  mailhost.example.net.
www    IN  A   212.74.112.80

; A delegated subdomain
subdom IN  NS  ns1.othernet.net.
       IN  NS  ns2.othernet.net.
```

There is one problem here:

- NS records point to names, not IPs
- What if zone "example.com" is delegated to "ns.example.com"?
- Someone who is in the process of resolving (say) www.example.com first has to resolve ns.example.com
- But in order to resolve ns.example.com they must first resolve ns.example.com !!

In this case you need "glue"

- A "glue record" is an A record for the nameserver, held higher in the tree
- Example: consider the .com nameservers, and a delegation for example.com

```
; this is the com. zone

example             NS   ns.example.com.
                    NS   ns.othernet.net.

ns.example.com.     A    192.0.2.1      ; GLUE RECORD
```

Don't put in glue records except where necessary

- In the previous example, "ns.othernet.net" is not a subdomain of "example.com". Therefore no glue is needed.
- Out-of-date glue records are a big source of problems
  - e.g. after renumbering a nameserver
  - Results in intermittent problems, difficult to debug

Example where a glue record IS needed

```
; My own zone data
            IN  MX  10  mailhost.example.net.
www         IN  A   212.74.112.80

; A delegated subdomain
subdom      IN  NS  ns1.subdom          ; needs glue
            IN  NS  ns2.othernet.net.   ; doesn't
ns1.subdom  IN  A   192.0.2.4
```

Checking for glue records

- dig +norec *... and repeat several times*
- Look for A records in the "Additional" section whose TTL does not count down

```
$ dig +norec @a.gtld-servers.net. www.as9105.net. a
 ...
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 2, ADDITIONAL: 1
;; QUERY SECTION:
;;      www.as9105.net, type = A, class = IN

;; AUTHORITY SECTION:
as9105.net.          172800  IN    NS        ns0.as9105.com.
as9105.net.          172800  IN    NS        ns0.tiscali.co.uk.

;; ADDITIONAL SECTION:
ns0.as9105.com.      172800  IN    A         212.139.129.130
```

Practical

- Delegating a subdomain

Further reading

- "DNS and BIND" (O'Reilly)
- BIND 9 Administrator Reference Manual
  - /usr/share/doc/bind9/arm/Bv9ARM.html
- http://www.isc.org/sw/bind/
  - includes FAQ, security alerts
- RFC 1912, RFC 2182
  - http://www.rfc-editor.org/