

Introduction to the DNS

AfNOG 2009
Cairo, Egypt

Overview

- Goal of this session
- What is DNS ?
- How is DNS built and how does it work?
- How does a query work ?
- Record types
- Caching and Authoritative
- Delegation: domains vs zones
- Finding the error: where is it broken?

Goal of this session

- We will review the basics of DNS, including query mechanisms, delegation, and caching.
- The aim is to be able to understand enough of DNS to be able to configure a caching DNS server, and troubleshoot common DNS problems, both local and remote (on the Internet)

What is DNS ?

- System to convert names to IP addresses:

`www.ws.afnog.org` → `196.200.223.1`

`www.afrinic.net` → `2001:42d0::200:80:1`

- ... and back:

`196.200.223.1` → `noc.ws.afnog.org`

`1.0.0.0.0.8.0.0.0.0.2.0.0.0.0.0.0.0.0.0`

`.0.0.0.0.0.d.2.4.1.0.0.2.ip6.arpa.` →

`www.afrinic.net.`

What is DNS ?

- Other information can be found in DNS:
 - where to send mail for a domain
 - who is responsible for this system
 - geographical information
 - etc...
- How do we look this information up?

Basic DNS tools

- Using the host command:

```
# host noc.ws.afnog.org.
```

```
noc.ws.afnog.org has address 196.200.223.1
```

```
# host 196.200.223.1
```

```
1.223.200.196.in-addr.arpa domain name  
pointer noc.ws.afnog.org.
```


Basic DNS tools

- Try this yourself with other names – first lookup the names below, then do the same for the IP address returned:

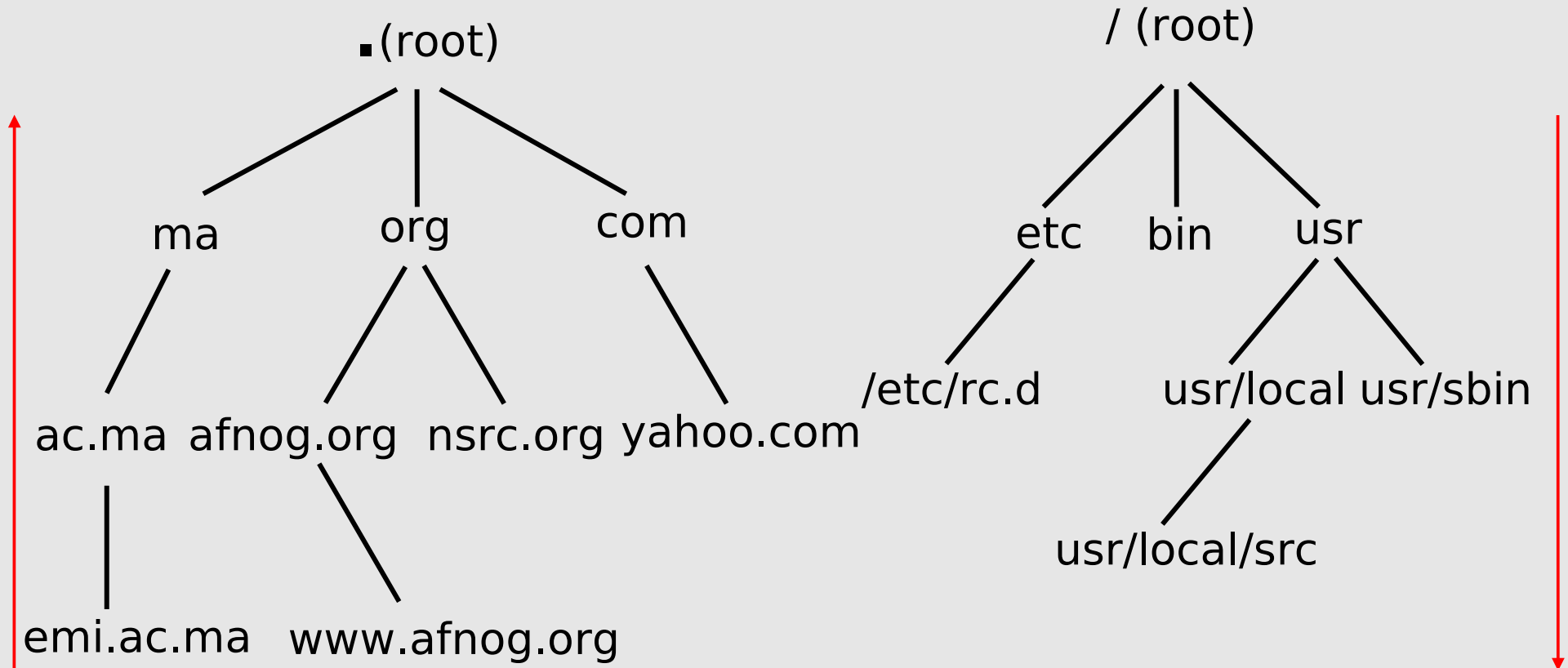
`www.yahoo.com`

`www.nsrc.org`

`ipv6.google.com`

- Does the lookup of the IP match the name ? Why ?
- Where did the 'host' command find the information ?

How is DNS built?



DNS Database

Unix Filesystem

... forms a tree structure

How is DNS built?

- DNS is hierarchical
- DNS administration is shared – no single central entity administrates all DNS data
- This distribution of the administration is called *delegation*

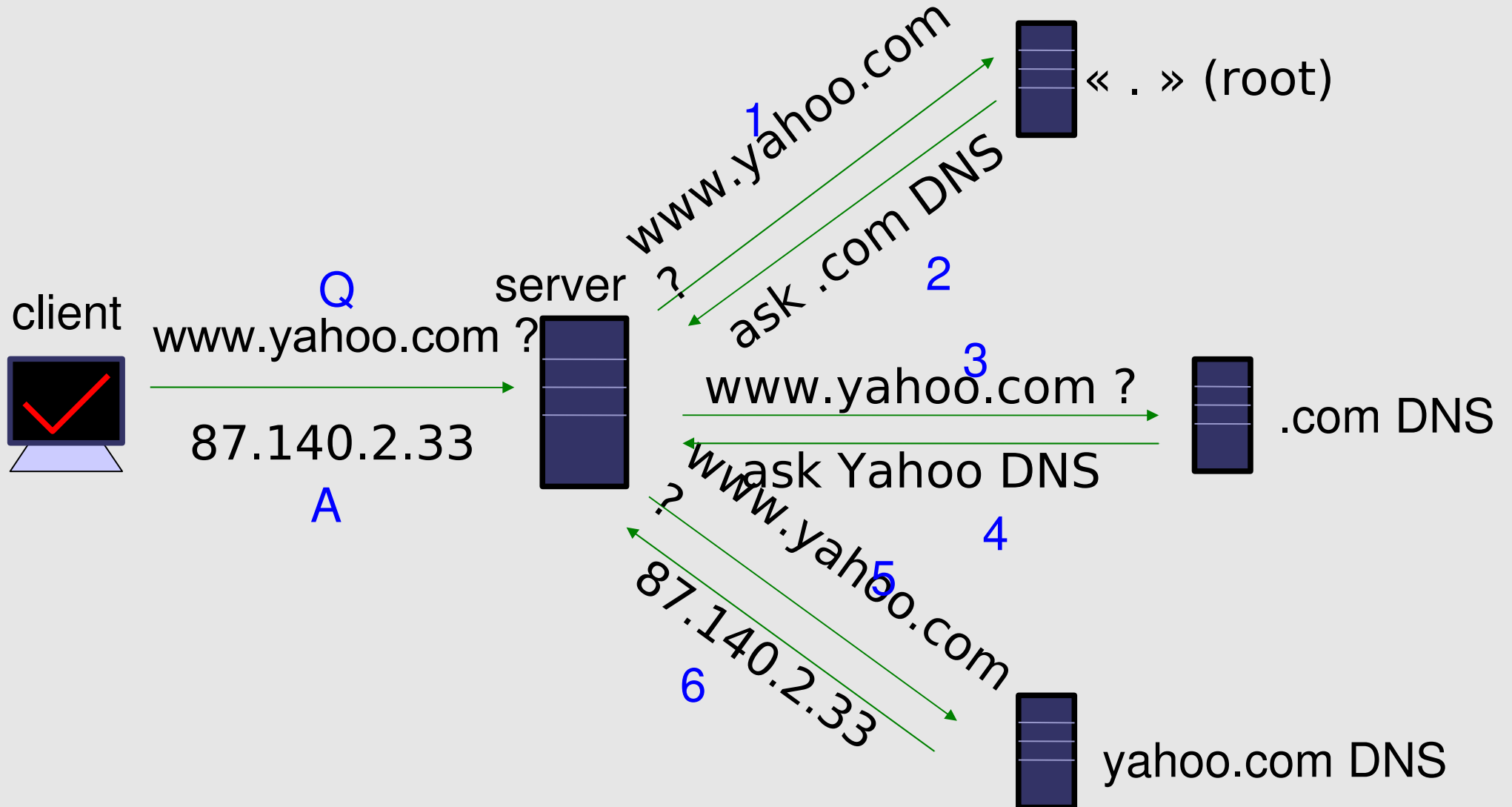
How does DNS work?

- **Clients** use a mechanism called a **resolver** and ask **servers** – this is called a **query**
- The server being queried will try to find the answer on behalf of the client
- The server functions recursively, from top (the root) to bottom, until it finds the answer, asking other servers along the way - the server is referred to other servers

How does DNS work?

- The client (web browser, mail program, ...) use the OS's resolver to find the IP address.
- For example, if we go to the webpage `www.yahoo.com`:
 - the web browser asks the OS « I need the IP for `www.yahoo.com` »
 - the OS looks in the resolver configuration which server to ask, and sends the query
- On UNIX, `/etc/resolv.conf` is where the resolver is configured.

A DNS query



Query detail with tcpdump

- Let's lookup 'h1-web.hosting.catpipe.net'

- On the server, we do:

```
# tcpdump -n udp and port 53
```

- In another window/screen do:

```
# host <something>
```

Query detail - output

- 1: 18:40:38.62 IP 192.168.1.1.57811 > 192.112.36.4.53:29030 [1au] A? h1-web.hosting.catpipe.net. (55)
- 2: 18:40:39.24 IP 192.112.36.4.53 > 192.168.1.1.57811:29030- 0/13/16 (540)
- 3: 18:40:39.24 IP 192.168.1.1.57811 > 192.43.172.30.53:7286 [1au] A? h1-web.hosting.catpipe.net. (55)
- 4: 18:40:39.93 IP 192.43.172.30.53 > 192.168.1.1.57811:7286 FormErr- [0q] 0/0/0 (12)
- 5: 18:40:39.93 IP 192.168.1.1.57811 > 192.43.172.30.53:50994 A? h1-web.hosting.catpipe.net. (44)
- 6: 18:40:40.60 IP 192.43.172.30.53 > 192.168.1.1.57811:50994- 0/3/3 (152)
- 7: 18:40:40.60 IP 192.168.1.1.57811 > 83.221.131.7.53:58265 [1au] A? h1-web.hosting.catpipe.net. (55)
- 8: 18:40:41.26 IP 83.221.131.7.53 > 192.168.1.1.57811:58265* 1/2/3 A 83.221.131.6 (139)

Query detail - analysis

- We use a packet analyser (wireshark / ethereal) to view the contents of the query...

Resolver configuration

- So how does your computer know which server to ask to get answers to DNS queries ?
- On UNIX, look in `/etc/resolv.conf`
- Look now in the file, and verify that you have a 'nameserver' statement of the form:
`nameserver a.b.c.d`
or
`nameserver ip:v6:ad:dr:es:ss`
... where a.b.c.d is the IP/IPv6 of a functioning DNS server (it should).

Finding the root...

- The first query is directed to:

192.112.36.4 (G.ROOT-SERVERS.NET.)

- How does the server know where to reach the root servers ?
- Chicken-and-egg problem
- Each nameserver has a list of the root nameservers (A – M.ROOT-SERVERS.NET) and their IP address
- In BIND, `named.root`

Using 'dig' to get more details

- the 'host' command is limited in its output – good for lookups, but not enough for debugging.
- we use the 'dig' command to obtain more details
- dig shows a lot of interesting stuff...

Using 'dig' to get more details

```
ns# dig @147.28.0.39 www.nsrc.org. a

; <<>> DiG 9.3.2 <<>> @147.28.0.39 www.afnog.org
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4620
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 4,
ADDITIONAL: 2

;; QUESTION SECTION:
;www.afnog.org.                IN      A

;; ANSWER SECTION:
www.afnog.org.                14400   IN      A      128.223.162.29

;; AUTHORITY SECTION:
afnog.org.                    14400   IN      NS     rip.psg.com.
afnog.org.                    14400   IN      NS     arizona.edu.

;; ADDITIONAL SECTION:
rip.psg.com.                  77044   IN      A      147.28.0.39
arizona.edu.                 2301    IN      A      128.196.128.233

;; Query time: 708 msec
;; SERVER: 147.28.0.39#53(147.28.0.39)
;; WHEN: Wed May 10 15:05:55 2007
;; MSG SIZE rcvd: 128
```

```
noc# dig www.afrinic.net any
```

```
; <<>> DiG 9.4.2 <<>> any www.afrinic.net
```

```
;; global options: printcmd
```

```
;; Got answer:
```

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36019
```

```
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 6, ADDITIONAL: 10
```

```
;; QUESTION SECTION:
```

```
;www.afrinic.net.          IN  ANY
```

```
;; ANSWER SECTION:
```

```
www.afrinic.net. 477      IN  AAAA  2001:42d0::200:80:1
```

```
www.afrinic.net. 65423   IN  A      196.216.2.1
```

```
;; AUTHORITY SECTION:
```

```
afrinic.net.      65324   IN  NS     sec1.apnic.net.
```

```
afrinic.net.      65324   IN  NS     sec3.apnic.net.
```

```
afrinic.net.      65324   IN  NS     ns1.afrinic.net.
```

```
afrinic.net.      65324   IN  NS     tinnie.arin.net.
```

```
afrinic.net.      65324   IN  NS     ns.lacnic.net.
```

```
afrinic.net.      65324   IN  NS     ns-sec.ripe.net.
```

```
;; ADDITIONAL SECTION:
```

```
ns.lacnic.net.    151715  IN  A      200.160.0.7
```

```
ns.lacnic.net.    65315   IN  AAAA   2001:12ff::7
```

```
ns-sec.ripe.net. 136865  IN  A      193.0.0.196
```

```
ns-sec.ripe.net. 136865  IN  AAAA   2001:610:240:0:53::4
```

```
ns1.afrinic.net. 65315   IN  A      196.216.2.1
```

```
tinnie.arin.net. 151715  IN  A      168.143.101.18
```

```
sec1.apnic.net.   151715  IN  A      202.12.29.59
```

```
sec1.apnic.net.   151715  IN  AAAA   2001:dc0:2001:a:4608::59
```

```
sec3.apnic.net.   151715  IN  A      202.12.28.140
```

```
sec3.apnic.net.   151715  IN  AAAA   2001:dc0:1:0:4777::140
```

```
;; Query time: 1 msec
```

```
;; SERVER: 196.200.218.1#53(196.200.218.1)
```

```
;; WHEN: Tue May 27 08:48:13 2008
```

```
;; MSG SIZE rcvd: 423
```



dig output

- Some interesting fields:
 - flags section: qr aa rd
 - status
 - answer section
 - authority section
 - TTL (numbers in the left column)
 - query time
 - server
- Notice the 'A' and 'AAAA' record type in the output.

Record types

- Basic record types:
 - A, AAAA: IPv4, IPv6 address
 - NS: NameServer
 - MX: Mail eXchanger
 - CNAME: Canonical name (alias)
 - PTR: Reverse information

Caching vs Authoritative

- In the dig output, and in subsequent outputs, we noticed a decrease in query time if we repeated the query.
- Answers are being **cached** by the querying nameserver, to speed up requests and save network resources
- The TTL value controls the time an answer can be cached
- DNS servers can be put in two categories: **caching** and **authoritative**.

Caching vs Authoritative: authoritative

- Authoritative servers typically only answer queries for data over which they have authority, i.e.: data of which they have an external copy, i.e. from disk (file or database)
- If they do not know the answer, they will point to a source of authority, but will not process the query recursively.

Caching vs Authoritative: caching

- Caching nameservers act as query forwarders on behalf of clients, and cache answers for later.
- Can be the same software (often is), but mixing functionality (recursive/caching and authoritative) is discouraged (security risks + confusing)
- The TTL of the answer is used to determine how long it may be cached without re-querying.

TTL values

- TTL values decrement and expire
- Try repeatedly asking for the A record for `www.yahoo.com`:

```
# dig www.yahoo.com
```

- What do you observe about the query time and the TTL ?

SOA

- Let's query the SOA for a domain:

```
# dig SOA <domain>
...
;; AUTHORITY SECTION:
<domain>. 860 IN SOA ns.<domain>. root.<domain>.
                200702270 ; serial
                28800    ; refresh
                14400    ; retry
                3600000   ; expire
                86400    ; neg ttl
...
```

SOA

- The first two fields highlighted are:
 - the SOA (Start Of Authority), which the administrator sets to the name of the « source » server for the domain data (this is not always the case)
 - the RP (Responsible Person), which is the email address (with the first @ replaced by a '.') to contact in case of technical problems.

SOA

- The other fields are:
 - serial: the serial number of the zone: this is used for replication between two nameservers
 - refresh: how often a replica server should check the master to see if there is new data
 - retry: how often to retry if the master server fails to answer after refresh.
 - expire: when the master server has failed to answer for too long, stop answering clients about this data.
- Why is expire necessary ?

Running a caching nameserver

- Running a caching nameserver locally can be very useful
- Easy to setup, for example on FreeBSD:
 - add `named_enable="YES"` to `/etc/rc.conf`
 - start named:

```
/etc/rc.d/named start
```

- **What is a good test to verify that named is running ?**

Running a caching nameserver

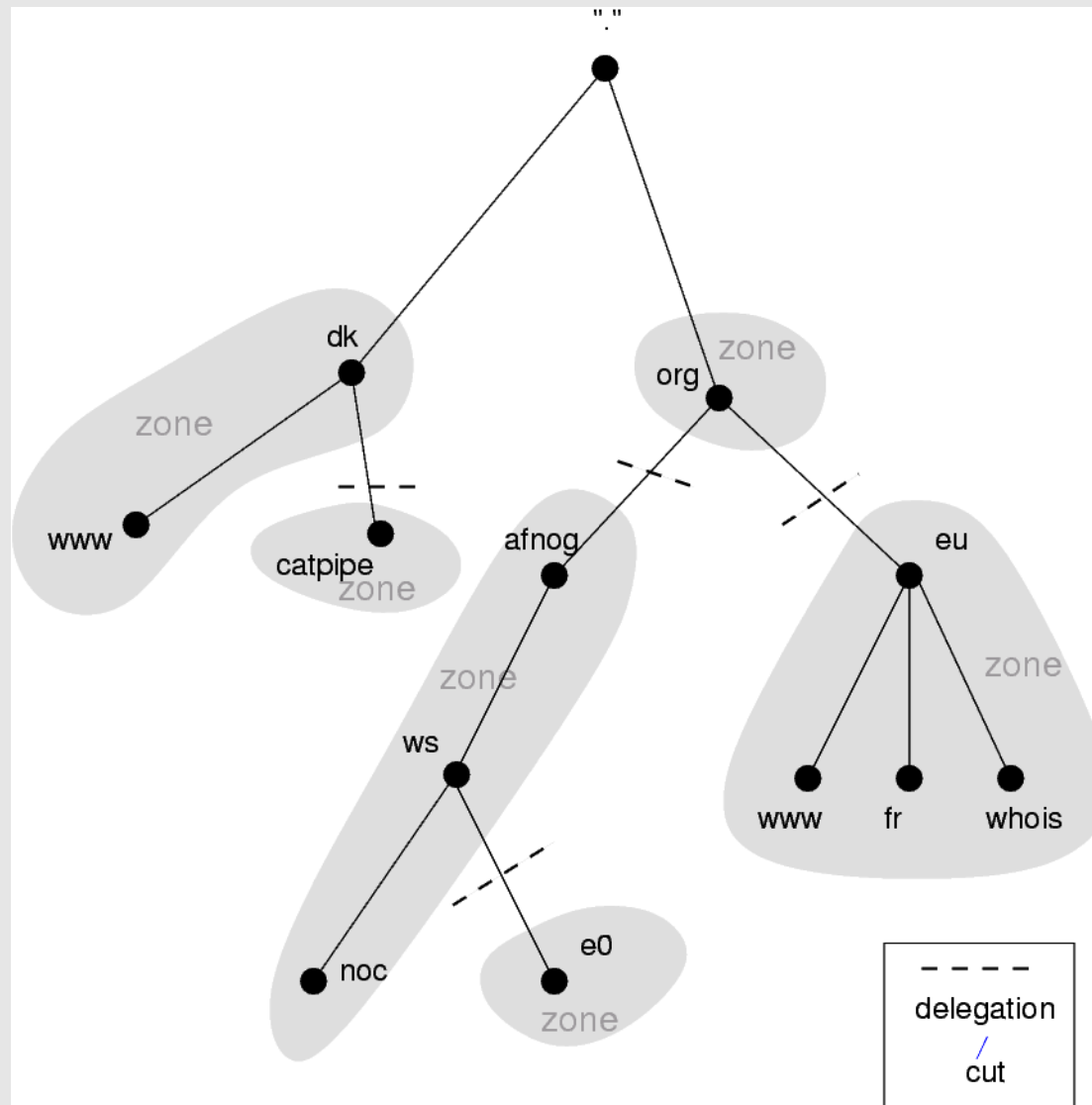
- When you are confident that your caching nameserver is working, enable it in your local resolver configuration (`/etc/resolv.conf`):

```
nameserver 127.0.0.1
```


Delegation

- We mentioned that one of the advantages of DNS was that of distribution through shared administration. This is called delegation.
- We delegate when there is an administrative boundary and we want to turn over control of a subdomain to:
 - a department of a larger organization
 - an organization in a country
 - an entity representing a country's domain

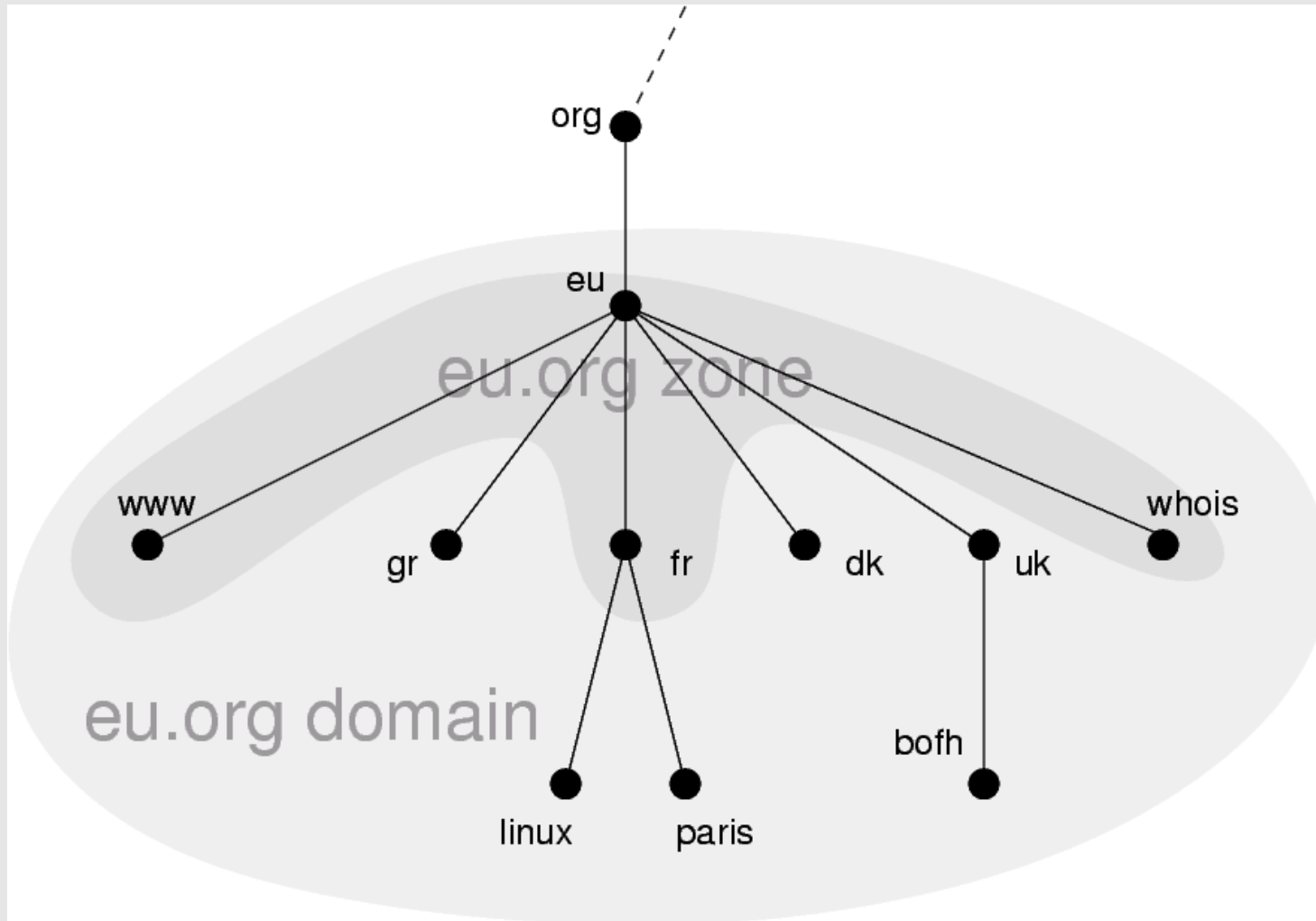
Delegation



Delegation: Domains vs Zones

- When we talk about the entire subtree, we talk about *domains*
- When we talk about part of a domain that is administered by an entity, we talk about *zones*

Delegation: Domains vs Zones



Finding the error: using doc

- When you encounter problems with your network, web service or email, you don't always suspect DNS.
- When you do, it's not always obvious what the problem is – DNS is tricky.
- A great tool for quickly spotting configuration problems is 'doc'
- `/usr/ports/dns/doc` – install it now!
- Let's do a few tests on screen with doc...

Conclusion

- DNS is a vast subject
- It takes a lot of practice to pinpoint problems accurately the first time – caching and recursion are especially confusing
- Remember that there are several servers for the same data, and you don't always talk to the same one
- Practice, practice, practice!
- Don't be afraid to ask questions...

Questions ?

?