



# Introduction to the DNS system

---

Olaf M. Kolkman

Okolkman@ripe.net





# Purpose of naming

---

- Addresses are used to locate objects
- Names are easier to remember than numbers
- You would like to get to the address or other objects using a name
- **DNS provides a mapping from names to resources of several types**



# Names and addresses in general

---

- An address is how you get to an endpoint
  - Typically, hierarchical (for scaling):
    - 950 Charter Street, Redwood City CA, 94063
    - 204.152.187.11, +1-650-381-6003
- A “name” is how an endpoint is referenced
  - Typically, no structurally significant hierarchy
    - “David”, “Tokyo”, “itu.int”



# Naming History

---

- 1970's ARPANET
  - ◆ Host.txt maintained by the SRI-NIC
  - ◆ pulled from a single machine
  - ◆ Problems
    - ◆ traffic and load
    - ◆ Name collisions
    - ◆ Consistency
- DNS created in 1983 by Paul Mockapetris (RFCs 1034 and 1035), modified, updated, and enhanced by a myriad of subsequent RFCs

# DNS

- A lookup mechanism for translating objects into other objects
- A globally distributed, loosely coherent, scalable, reliable, dynamic database
- Comprised of three components
  - A “name space”
  - Servers making that name space available
  - Resolvers (clients) which query the servers about the name space



# DNS Features: Global Distribution

---

- Data is maintained locally, but retrievable globally
  - ◆ No single computer has all DNS data
- DNS lookups can be performed by any device
- Remote DNS data is locally cachable to improve performance



# DNS Features: Loose Coherency

---

- The database is always internally consistent
  - ◆ Each version of a subset of the database (a zone) has a serial number
    - ◆ The serial number is incremented on each database change
- Changes to the master copy of the database are replicated according to timing set by the zone administrator
- Cached data expires according to timeout set by zone administrator



# DNS Features: Scalability

---

- No limit to the size of the database
  - ◆ One server has over 20,000,000 names
    - ◆ Not a particularly good idea
- No limit to the number of queries
  - ◆ 24,000 queries per second handled easily
- Queries distributed among masters, slaves, and caches





# DNS Features: Reliability

---

- Data is replicated
  - ◆ Data from master is copied to multiple slaves
- Clients can query
  - ◆ Master server
  - ◆ Any of the copies at slave servers
- Clients will typically query local caches
- DNS protocols can use either UDP or TCP
  - ◆ If UDP, DNS protocol handles retransmission, sequencing, etc.



# DNS Features: Dynamicity

---

- Database can be updated dynamically
  - ◆ Add/delete/modify of any record
- Modification of the master database triggers replication
  - ◆ Only master can be dynamically updated
    - ◆ Creates a single point of failure



# DNS Concepts

---

- Next slides are about concepts
- After this set of slides you should understand
  - ◆ How the DNS is built
  - ◆ Why it is built the way it is
  - ◆ The terminology used throughout the course



# Concept: DNS Names 1

---

- The namespace needs to be made hierarchical to be able to scale.
- The idea is to name objects based on
  - ◆ location (within country, set of organizations, set of companies, etc)
  - ◆ unit within that location (company within set of company, etc)
  - ◆ object within unit (name of person in company)

# Concept: DNS Names 2

## How names appear in the DNS

---

Fully Qualified Domain Name (FQDN)

**WWW.RIPE.NET.**

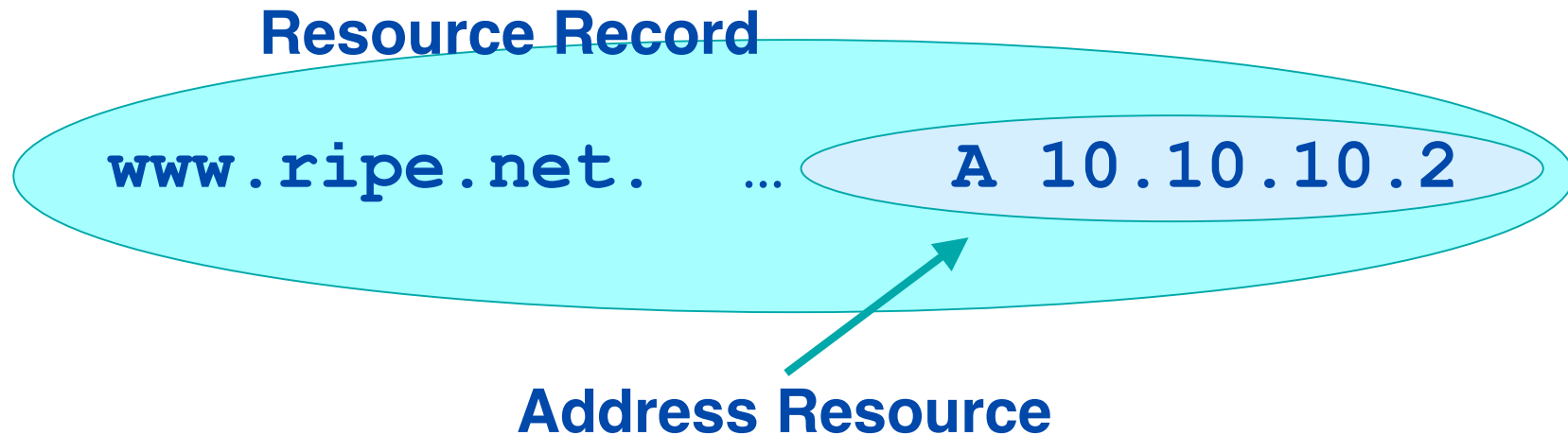
- labels separated by dots

**Note the trailing dot**

- DNS provides a mapping from FQDNs to resources of several types
- Names are used as a key when fetching data in the DNS

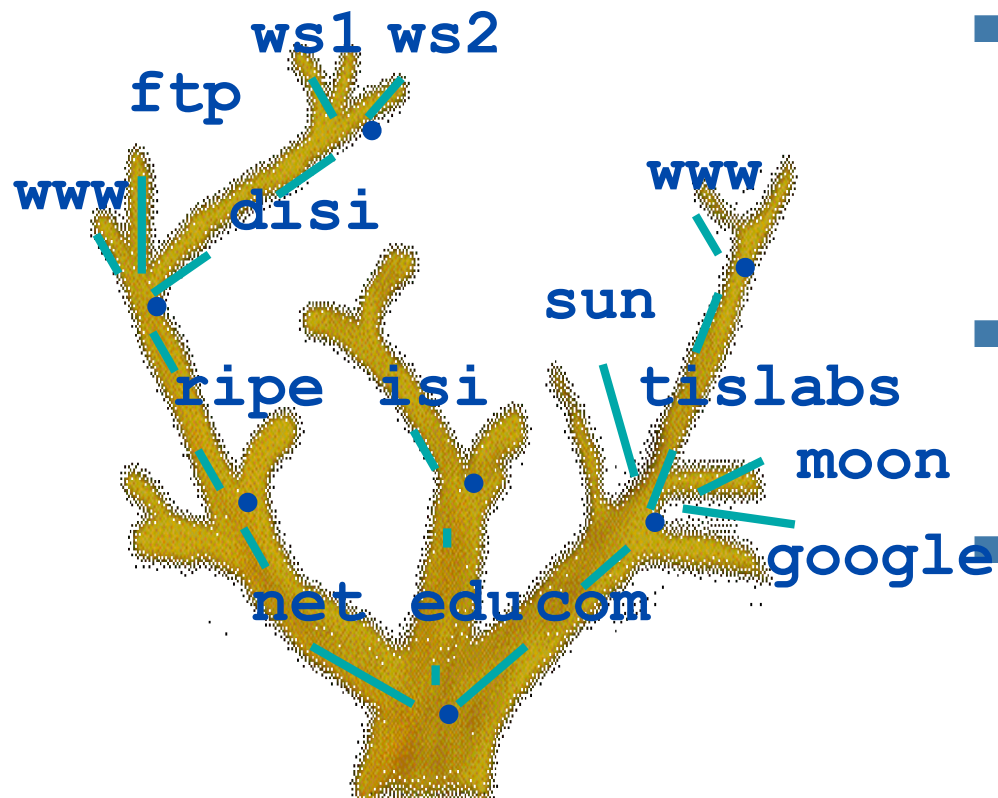
# Concept: Resource Records

- The DNS maps names into data using Resource Records.



- More detail later

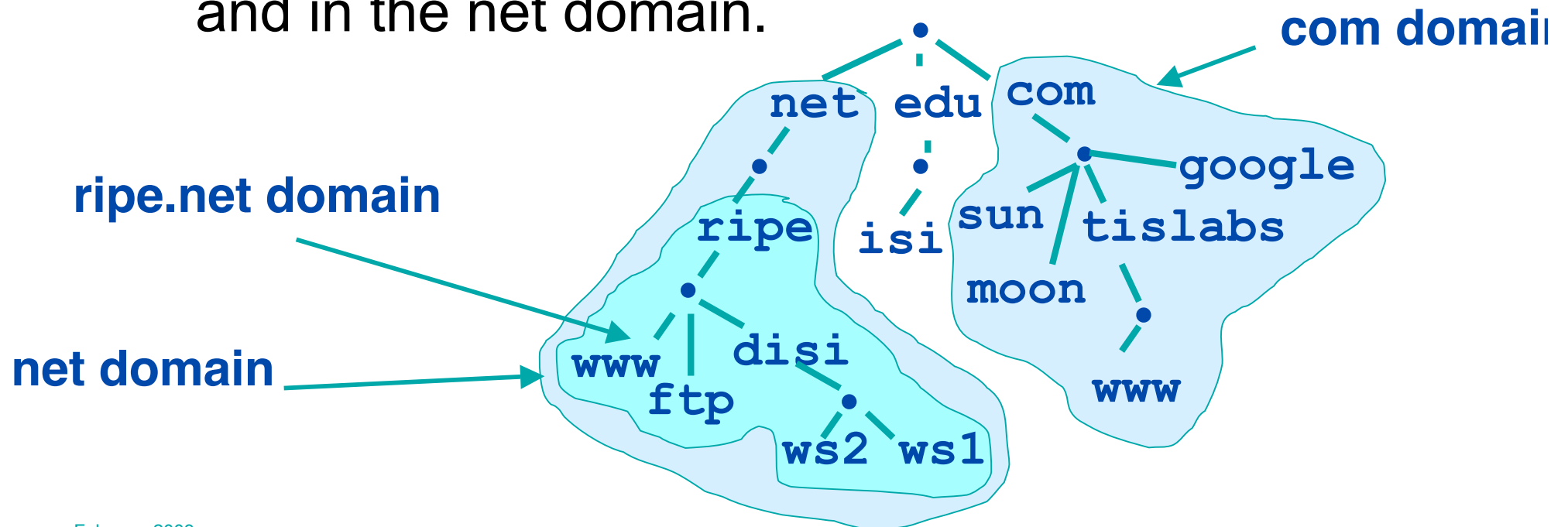
# Concept: DNS Names 3



- Domain names can be mapped to a tree.
- New branches at the 'dots'
- No restriction to the amount of branches.

# Concept: Domains

- Domains are “namespaces”
- Everything below .com is in the com domain.
- Everything below ripe.net is in the ripe.net domain and in the net domain.







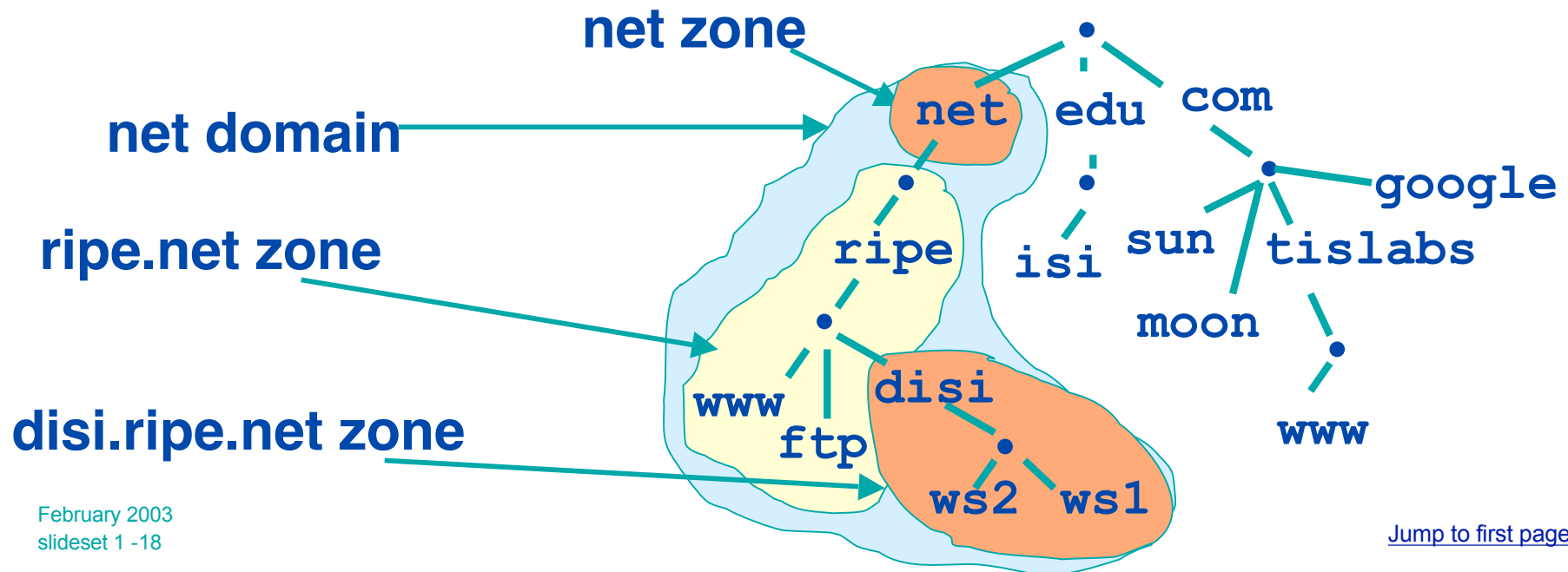
# Delegation

---

- Administrators can create subdomains to group hosts
  - ◆ According to geography, organizational affiliation or any other criterion
- An administrator of a domain can delegate responsibility for managing a subdomain to someone else
  - ◆ But this isn't required
- The parent domain retains links to the delegated subdomain
  - ◆ The parent domain “remembers” who it delegated the subdomain to

# Concept: Zones and Delegations

- Zones are “administrative spaces”
- Zone administrators are responsible for portion of a domain’s name space
- Authority is delegated from a parent and to a child





# Concept: Name Servers

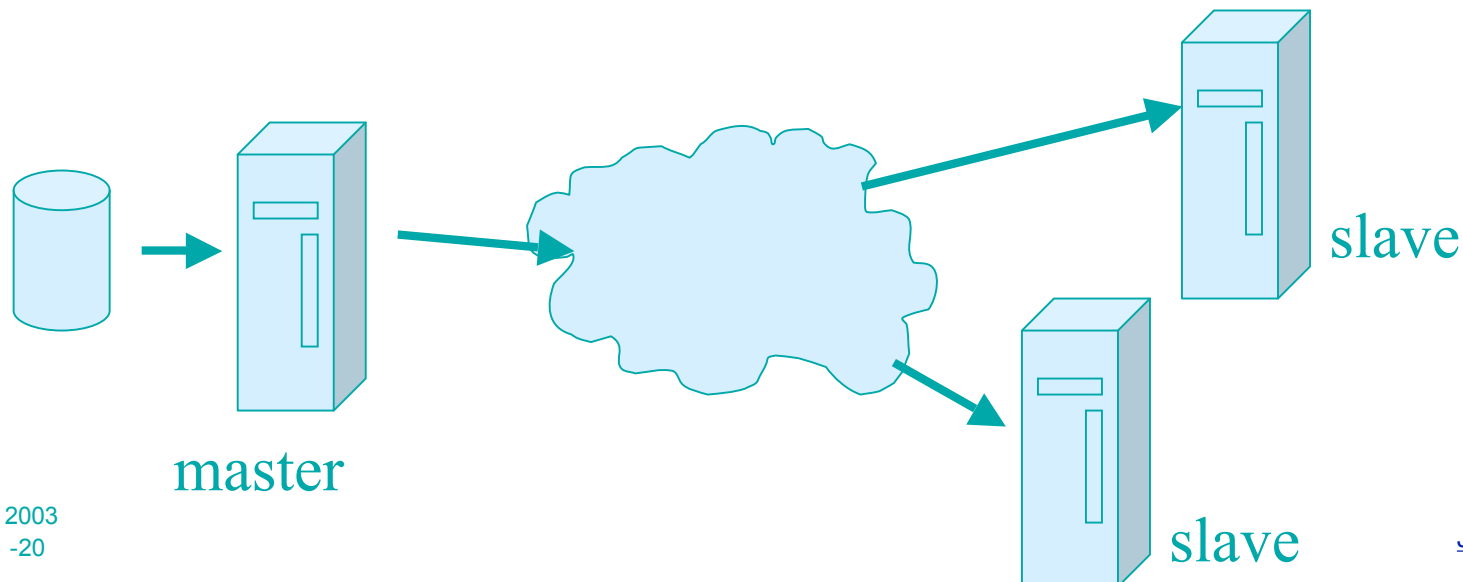
---

- Name servers answer 'DNS' questions.
- Several types of name servers
  - ◆ Authoritative servers
    - ◆ master (primary)
    - ◆ slave (secondary)
  - ◆ (Caching) recursive servers
    - ◆ also caching forwarders
  - ◆ Mixture of functionality

# Concept: Name Servers

## authoritative name server

- Give authoritative answers for one or more zones.
- The master server normally loads the data from a zone file
- A slave server normally replicates the data from the master via a zone transfer





# Concept: Name Servers recursive server

---

- Recursive servers do the actual lookups; they ask questions to the DNS on behalf of the clients.
- Answers are obtained from authoritative servers but the answers forwarded to the clients are marked as not authoritative
- Answers are stored for future reference in the cache



# Concept: Resolvers

---

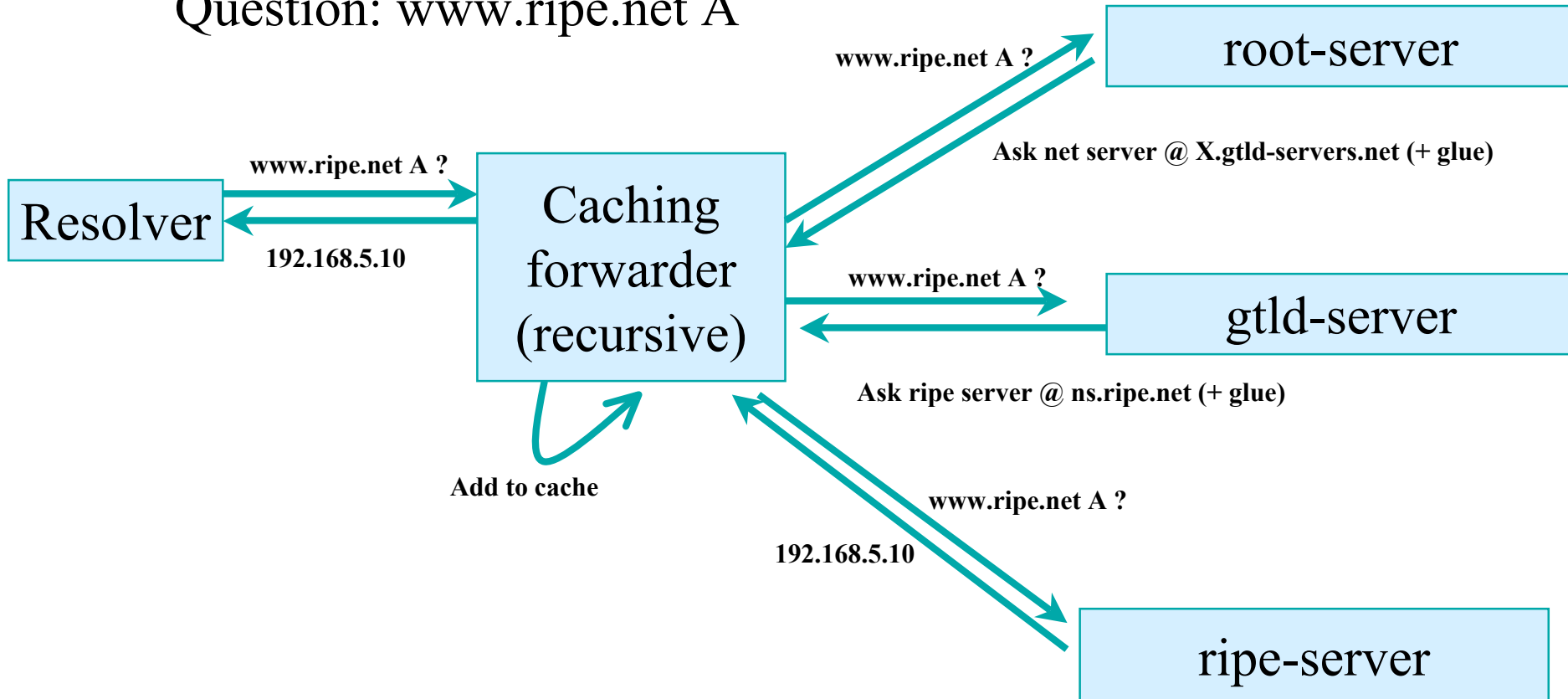
- Resolvers ask the questions to the DNS system on behalf of the application.
- Normally implemented in a system library (e.g, libc)

```
gethostbyname(char *name) ;
```

```
gethostbyaddr(char *addr, int len,  
type) ;
```

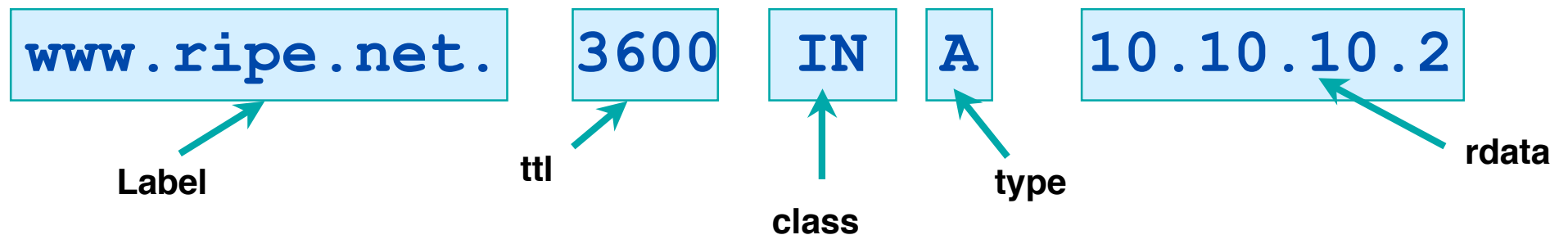
# Concept: Resolving process & Cache

Question: www.ripe.net A



# Concept: Resource Records (more detail)

- Resource records consist of its name, its TTL, its class, its type and its RDATA
- TTL is a timing parameter
- IN class is widest used
- There are multiple types of RR records
- Everything behind the type identifier is called rdata







# Example: RRs in a zone file

```
ripe.net. 7200 IN      SOA      ns.ripe.net.      olaf.ripe.net. (  
    2001061501      ; Serial  
    43200      ; Refresh 12 hours  
    14400      ; Retry 4 hours  
    345600     ; Expire 4 days  
    7200      ; Negative cache 2 hours  
    )
```

```
ripe.net. 7200 IN      NS      ns.ripe.net.  
ripe.net. 7200 IN      NS      ns.eu.net.
```

```
pinkje.ripe.net. 3600 IN      A      193.0.1.162
```

```
host25.ripe.net. 2600 IN      A      193.0.3.25
```

Label

ttd

class

type

rdata



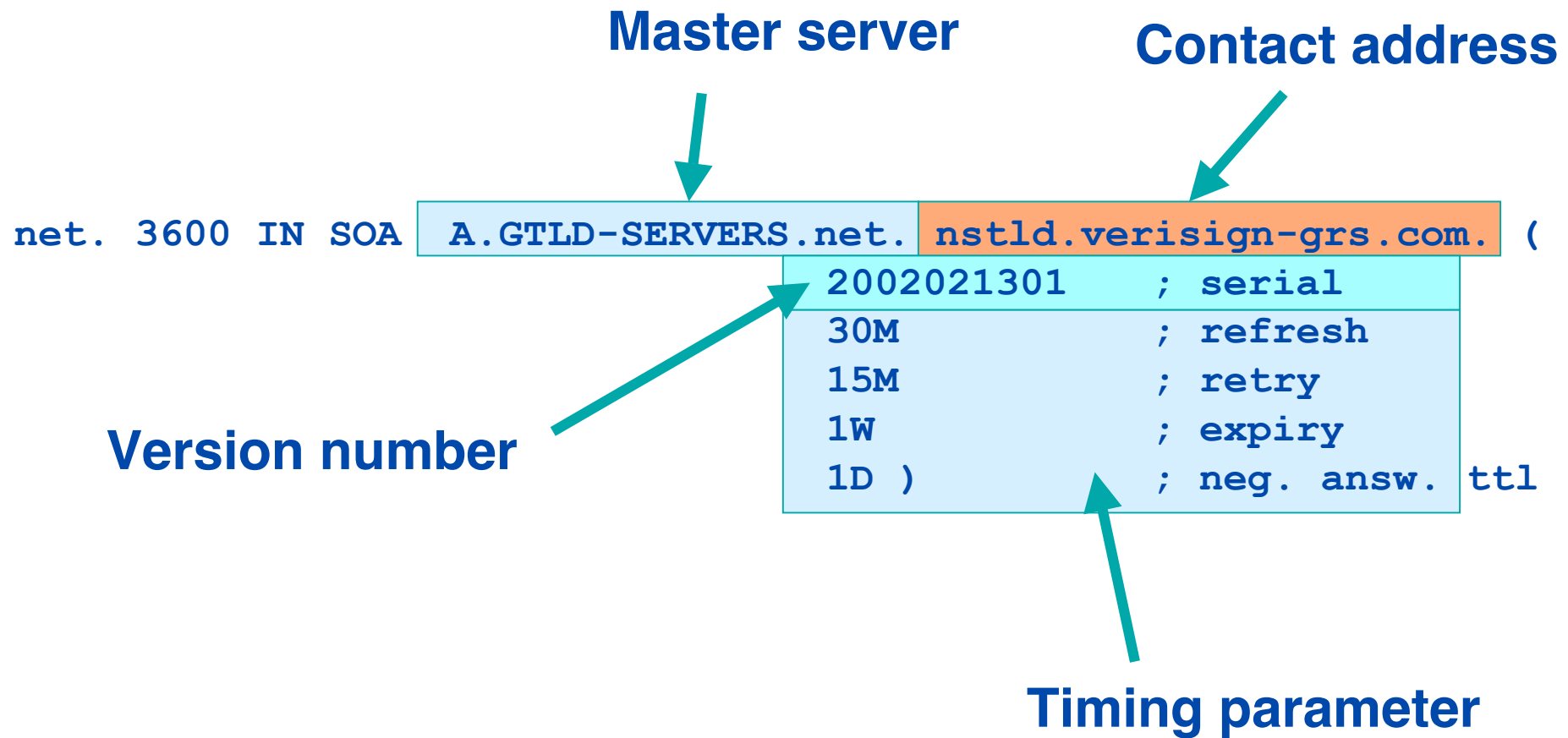
# Resource Record: SOA and NS

- The SOA and NS records are used to provide information about the DNS itself.
- The NS indicates where information about a given zone can be found:

```
ripe.net. 7200 IN NS ns.ripe.net.  
ripe.net. 7200 IN NS ns.eu.net.
```

- The SOA record provides information about the start of authority, i.e. the top of the zone, also called the APEX.

# Resource Record: SOA





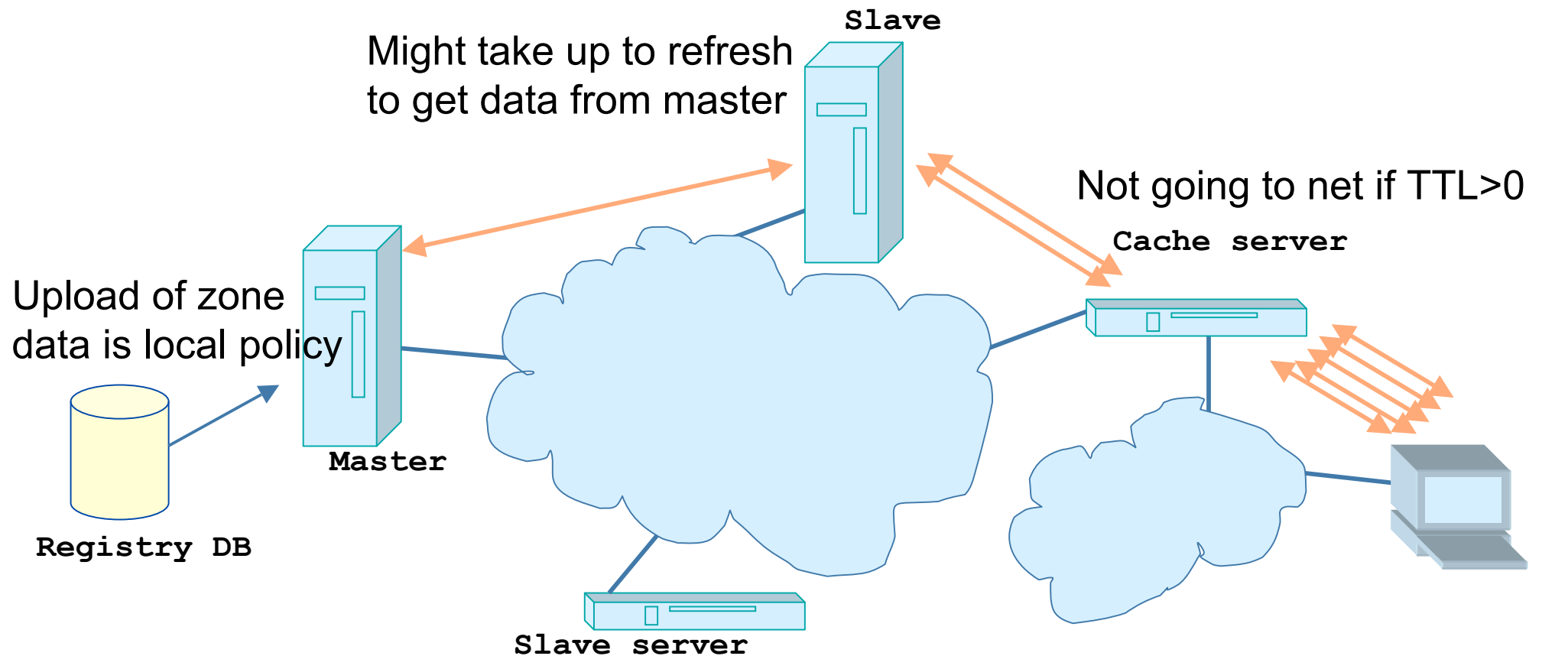
# Concept: TTL and other Timers

---

- TTL is a timer used in caches
  - ◆ An indication for how long the data may be reused
  - ◆ Data that is expected to be 'stable' can have high TTLs
- SOA timers are used for maintaining consistency between primary and secondary servers

# Places where DNS data lives

Changes in DNS do not propagate instantly!





# To remember...

- 
- Multiple authoritative servers to distribute load and risk:
    - ◆ Put your name servers apart from each other
  - Caches to reduce load to authoritative servers and reduce response times
  - SOA timers and TTL need to be tuned to needs of zone. Stable data: higher numbers



# What have we learned What are we about to learn

---

- We learned about the architecture:
  - ◆ resolvers,
  - ◆ caching forwarders,
  - ◆ authoritative servers,
  - ◆ timing parameters
- We continue writing a zone file



# Writing a zone file.

---

- Zone file is written by the zone administrator
- Zone file is read by the master server and it's content is replicated to slave servers
- What is in the zone file will end up in the database
- Because of timing issues it might take some time before the data is actually visible at the client side.





# First attempt

---

- The 'header' of the zone file
  - ◆ Start with a SOA record
  - ◆ Include authoritative name servers and, if needed, glue
  - ◆ Add other information
  
- Add other RRs
  
- Delegate to other zones

# The SOA record

Line break

Comments

---

```
secret-wg.org. 3600 IN SOA bert.secret-wg.org. (  
    olaf\.kolkman.ripe.net.  
    2002021301      ; serial  
    1h              ; refresh  
    30M             ; retry  
    1W              ; expiry  
    3600 )          ; neg. answ. ttl
```

- Olaf.Kolkman@ripe.net → olaf\.kolkman.ripe.net
- Serial number: 32bit circular arithmetic
  - ◆ People often use date format
  - ◆ To be increased after editing
- The timers above qualify as reasonable



# Authoritative NS records and related A records

---

<code>secret-wg.org.</code>	<code>3600 IN NS bert.secret-wg.org.</code>
<code>secret-wg.org.</code>	<code>3600 IN NS NS2.secret-wg.org.</code>
<code>bert.secret-wg.org.</code>	<code>3600 IN A 193.0.0.4</code>
<code>NS2.secret-wg.org.</code>	<code>3600 IN A 193.0.0.202</code>

- NS record for all the authoritative servers.
  - ◆ They need to carry the zone at the moment you publish
- A records only for “in-zone” name servers.
  - ◆ Delegating NS records might have glue associated.



# Other 'APEX' data

---

```
secret-wg.org. 3600 IN MX 50 mailhost.secret-wg.org.  
secret-wg.org. 3600 IN MX 150 mailhost2.secret-wg.org.
```

```
secret-wg.org. 3600 IN LOC (  
                    52 21 23.0 N 04 57 05.5 E 0m 100m 100m 100m )  
secret-wg.org. 3600 IN TXT "Demonstration and test zone"
```

## Examples:

- MX records for mail  
(see next slide)
- Location records

TXT records

A records

KEY records for dnssec



# Intermezzo: MX record

---

- SMTP (simple mail transfer protocol) uses MX records to find the destination mail server.
- If a mail is sent to olaf@ripe.net the sending mail agent looks up 'ripe.net MX'
- MX record contains mail relays with priority.
  - ◆ The lower the number the higher the priority.
- Don't add MX records without having a mail relay configured.



# Other data in the zone

---

`localhost.secret-wg.org. 3600 IN A 127.0.0.1`

`bert.secret-wg.org. 4500 IN A 193.0.0.4`

`www.secret-wg.org. 3600 IN CNAME bert.secret-wg.org.`

- Add all the other data to your zone file.
- Some notes on notation.
  - ◆ Note the fully qualified domain name including trailing dot.
  - ◆ Note TTL and CLASS



# Zone file format short cuts nice formatting

```
secret-wg.org.          3600  IN  SOA  bert.secret-wg.org. (
                        olaf\.kolkman.ripe.net.
                        2002021301      ; serial
                        1h               ; refresh
                        30M              ; retry
                        1W               ; expiry
                        3600 )          ; neg. answ. Ttl

secret-wg.org.          3600  IN  NS   bert.secret-wg.org.
secret-wg.org.          3600  IN  NS   NS2.secret-wg.org.
secret-wg.org.          3600  IN  MX   50 mailhost.secret-wg.org.
secret-wg.org.          3600  IN  MX   150 mailhost2.secret-wg.org.

secret-wg.org.          3600  IN  LOC  ( 52 21 23.0 N 04 57 05.5 E
                        0m 100m 100m 100m )

secret-wg.org.          3600  IN  TXT  "Demonstration and test zone"
bert.secret-wg.org.     4500  IN  A    193.0.0.4
NS2.secret-wg.org.      3600  IN  A    193.0.0.202
localhost.secret-wg.org. 3600  IN  A    127.0.0.1

bert.secret-wg.org.     3600  IN  A      193.0.0.4
www.secret-wg.org.      3600  IN  CNAME  bert.secret-wg.org.
```



# Zone file format short cuts: repeating last name

```
secret-wg.org.          3600  IN  SOA  bert.secret-wg.org. (
                        olaf\.kolkman.ripe.net.
                        2002021301      ; serial
                        1h               ; refresh
                        30M              ; retry
                        1W               ; expiry
                        3600 )           ; neg. answ. Ttl
                        3600 IN NS      bert.secret-wg.org.
                        3600 IN NS      NS2.secret-wg.org.
                        3600 IN MX      50 mailhost.secret-wg.org.
                        3600 IN MX      150 mailhost2.secret-wg.org.

                        3600 IN LOC     ( 52 21 23.0 N 04 57 05.5 E
                        0m 100m 100m 100m )
bert.secret-wg.org.      3600 IN TXT    "Demonstration and test zone"
NS2.secret-wg.org.      3600 IN A      193.0.0.4
                        3600 IN A      193.0.0.202

localhost.secret-wg.org. 4500 IN A      127.0.0.1

bert.secret-wg.org.      3600 IN A      193.0.0.4
www.secret-wg.org.      3600 IN CNAME  bert.secret-wg.org.
```





# Zone file format shortcuts: default TTL

```
$TTL      3600 ; Default TTL directive
secret-wg.org.      IN SOA bert.secret-wg.org. (
                                olaf\.kolkman.ripe.net.
                                2002021301      ; serial
                                1h                ; refresh
                                30M              ; retry
                                1W                ; expiry
                                3600 )           ; neg. answ. Ttl

                                IN NS      bert.secret-wg.org.
                                IN NS      NS2.secret-wg.org.
                                IN MX      50 mailhost.secret-wg.org.
                                IN MX      150 mailhost2.secret-wg.org.

                                IN LOC      ( 52 21 23.0 N 04 57 05.5 E
                                                0m 100m 100m 100m )

                                IN TXT      "Demonstration and test zone"
bert.secret-wg.org.      IN A      193.0.0.4
NS2.secret-wg.org.      IN A      193.0.0.202

localhost.secret-wg.org. IN A      127.0.0.1

bert.secret-wg.org. 4500 IN A      193.0.0.4
www.secret-wg.org.  IN CNAME bert.secret-wg.org.
```



# Zone file format shortcuts: ORIGIN

```
$TTL      3600 ; Default TTL directive
$ORIGIN   secret-wg.org.
@          IN SOA  bert (
                                olaf\.kolkman.ripe.net.
                                2002021301      ; serial
                                1h                ; refresh
                                30M               ; retry
                                1W                ; expiry
                                3600 )           ; neg. answ. Ttl

          IN NS   bert
          IN NS   NS2
          IN MX   50 mailhost
          IN MX   150 mailhost2

          IN LOC  ( 52 21 23.0 N 04 57 05.5 E
                    0m 100m 100m 100m )
          IN TXT  "Demonstration and test zone"

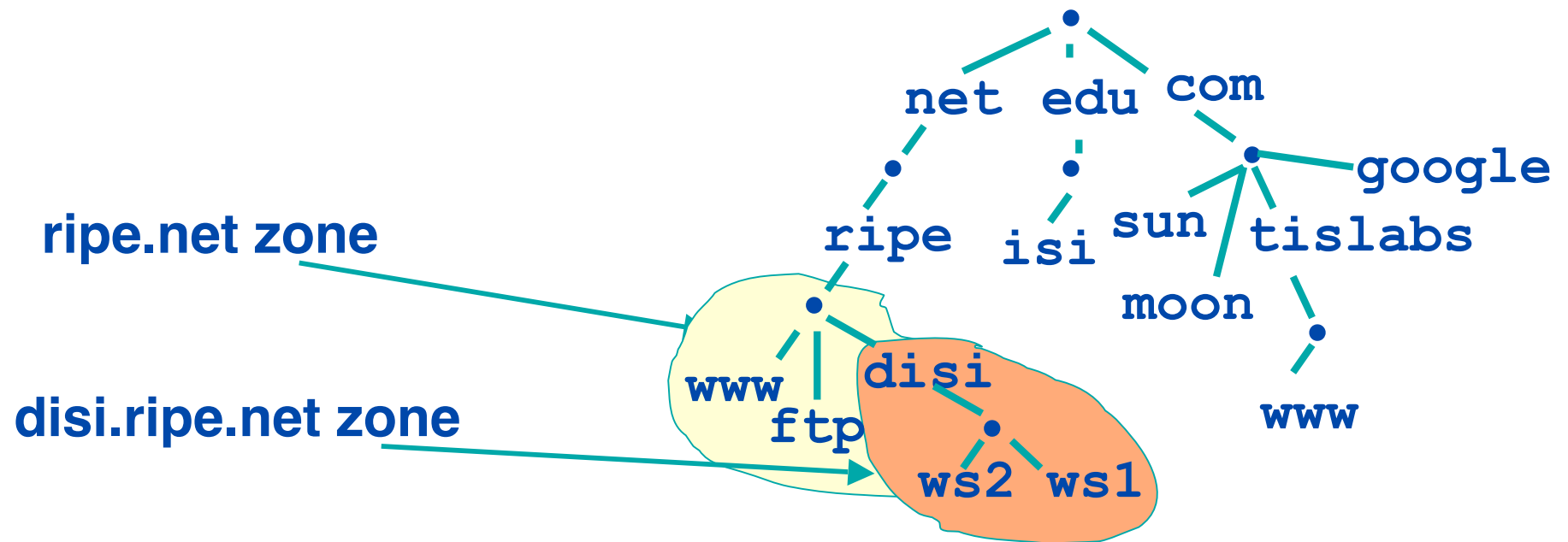
bert      IN A    193.0.0.4
NS2       IN A    193.0.0.202

localhost IN A    127.0.0.1

bert      4500 IN A    193.0.0.4
www       IN CNAME bert
```

# Delegating a zone (becoming a parent)

- Delegate authority for a sub domain to another party (splitting of disi.ripe.net from ripe.net)





# Concept: Glue

- Delegation is done by adding NS records:

```
disi.ripe.net.      NS      ns1.disi.ripe.net.
```

```
disi.ripe.net.      NS      ns2.disi.ripe.net.
```

- How to get to ns1 and ns2... We need the addresses.
- Add glue records to so that resolvers can reach ns1 and ns2.

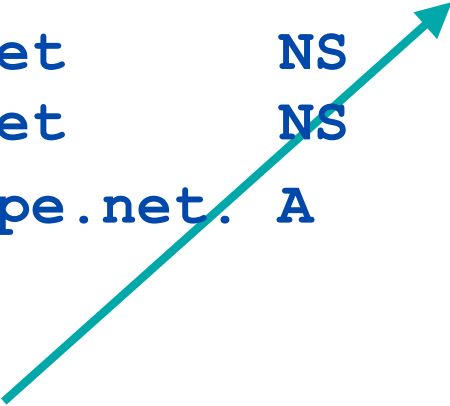
```
ns1.disi.ripe.net.  A  10.0.0.1
```

```
ns2.disi.ripe.net.  A  10.0.0.2
```

# Concept: Glue (continued)

- Glue is 'non-authoritative' data
- Don't include glue for servers that are not in sub zones

disi.ripe.net.	NS	ns1.disi.ripe.net.
disi.ripe.net	NS	ns2.ripe.net.
disi.ripe.net	NS	ns.bert.secret-wg.org.
ns1.disi.ripe.net.	A	10.0.0.1



**Only this record needs glue**



# Delegating disi.ripe.net. from ripe.net.

---

## disi.ripe.net

- Setup minimum two servers
- Create zone file with NS records
- Add all disi.ripe.net data

## ripe.net

- Add NS records and glue
- Make sure there is no other data from the disi.ripe.net. zone in the zone file.`



# Becoming a child In general

---

- Buy your domain at favorite registry
- Set up your name servers
- Register the name servers: your registry will communicate the name servers to the registrar who will make sure the name servers are published.
  - ◆ This process might take hours-days.
- Registrars may require a sensible setup