Introduction to the DNS system

Bill Manning

bmanning@ep.net

Administrative Constraints

- RFC’s
  - 1219
  - 1591
  - 2010
- ICP-1

RFC's

- RFC 1219 – re-interates – two servers MINIMUM, separate power grids, different administration.
- RFC 1591 – Community service, Accurate contact data, Responsible, cooperative parties in the shared responsibility of the delegation
- RFC 2010 – The first pass at loading observations/documentation – Parents have a responsibility to teach their children.

ICP-1

- ICANNs codification of these principles for TLD operators.
- Not much new defined, save contractual language and escrow powers (contraversial)

Jump to first page

Notes DNS0/ICANN/CENTR

- Tendancy to run TLD as a standalone value proposition
- Contracts/SLA’s may seem more important than they are.
- Herding tendencies (small number of sites slaving many TLD zones)
- Hiding data. (It’s a public resource)

Preparing for the future

- New features
- Better data integrity
- Community cooperation builds trust
- Teaching your delegations their responsibilities.

Jump to first page

Introduction to the DNS system

Bill Manning

bmanning@ep.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**

Jump to first page

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
  - 2,000-4,000 qps is more “normal”
- 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 almost 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)

Jump to first page

Concept: DNS Names 2
How names appear in the DNS

Fully Qualified Domain Name (FQDN)

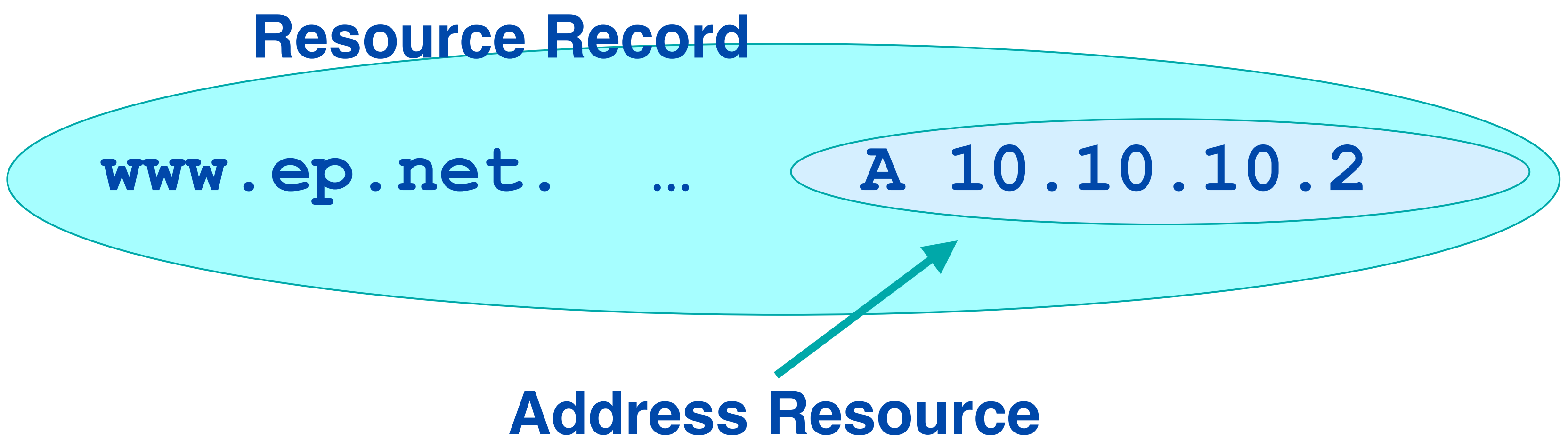
`WWW.EP.NET.`

Note the trailing dot

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

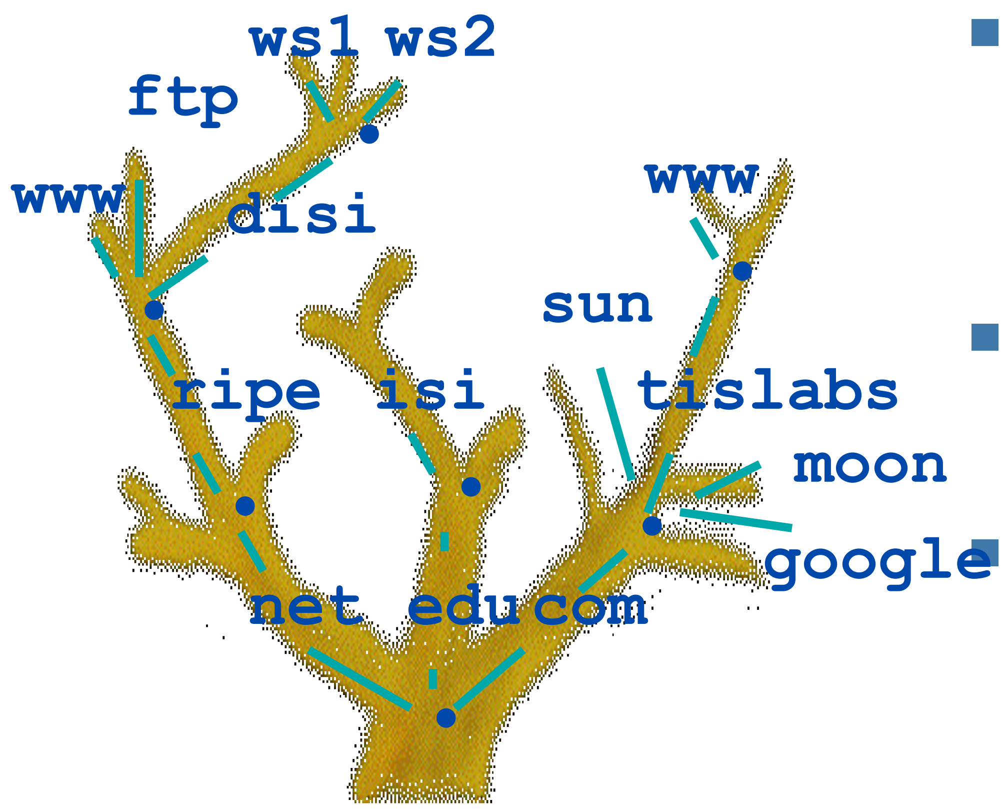
Concept: Resource Records

- The DNS maps names into data using Resource Records.



- More detail later

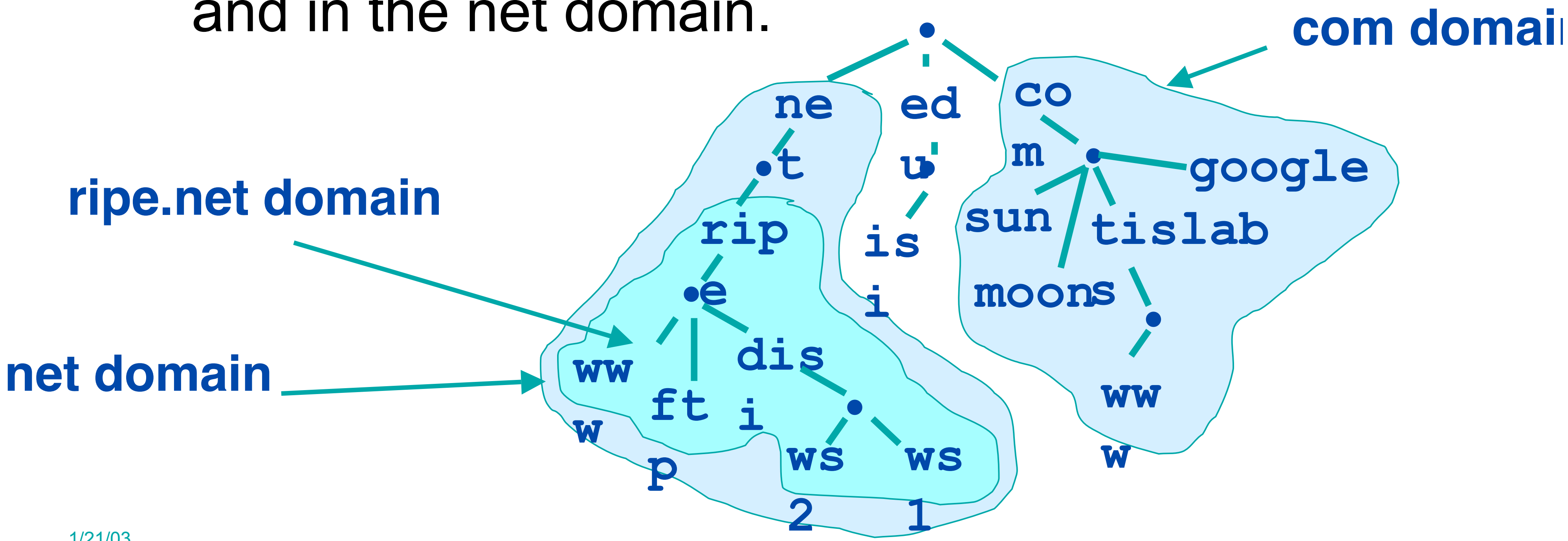
Concept: DNS Names 3



- 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.

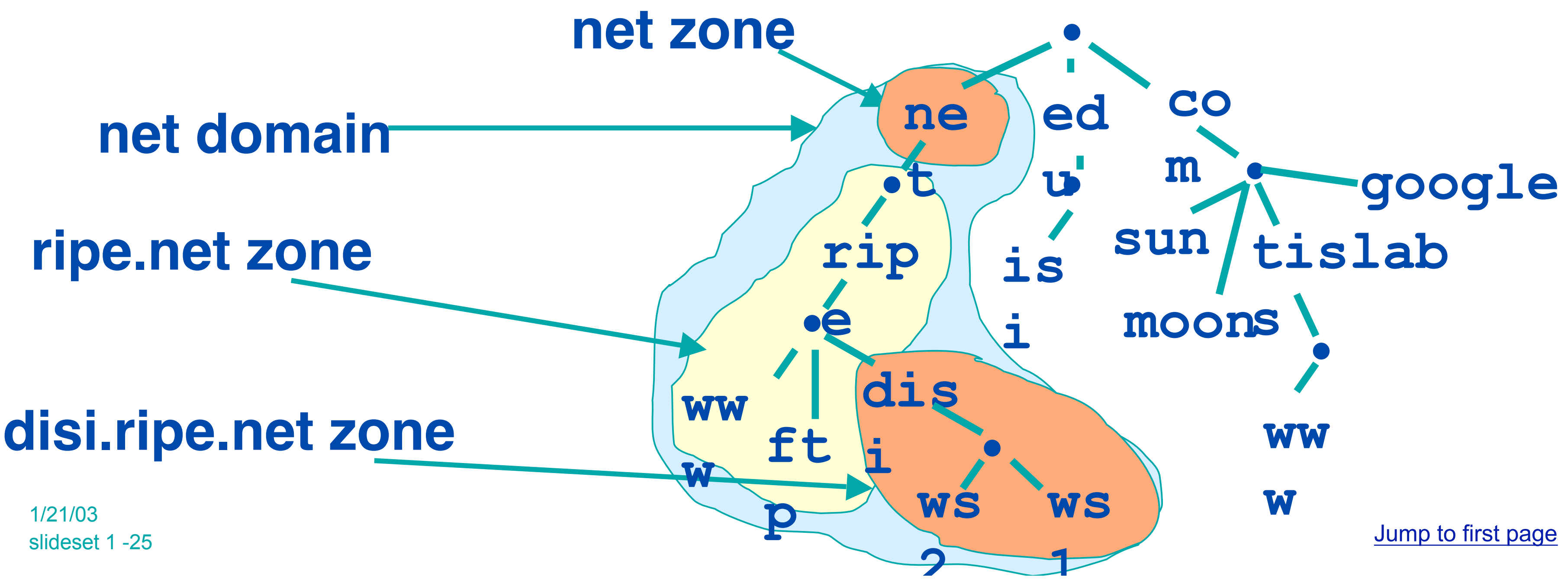


Delegation

- 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



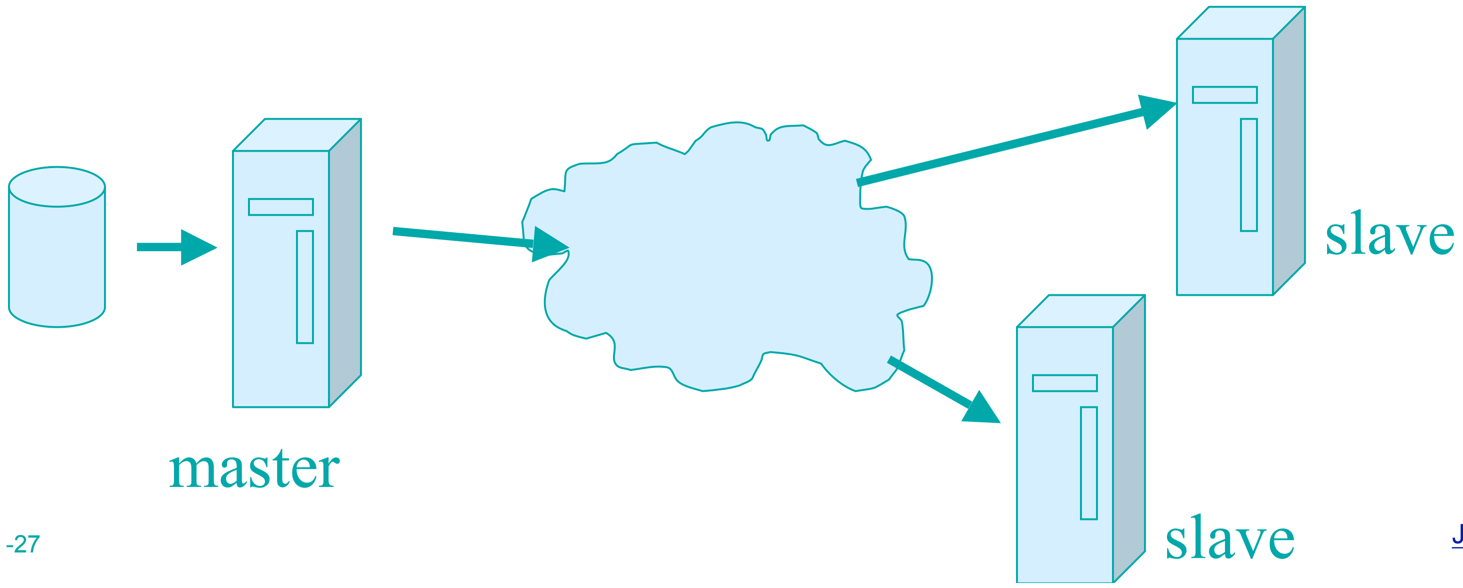
Concept: Name Servers

- 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

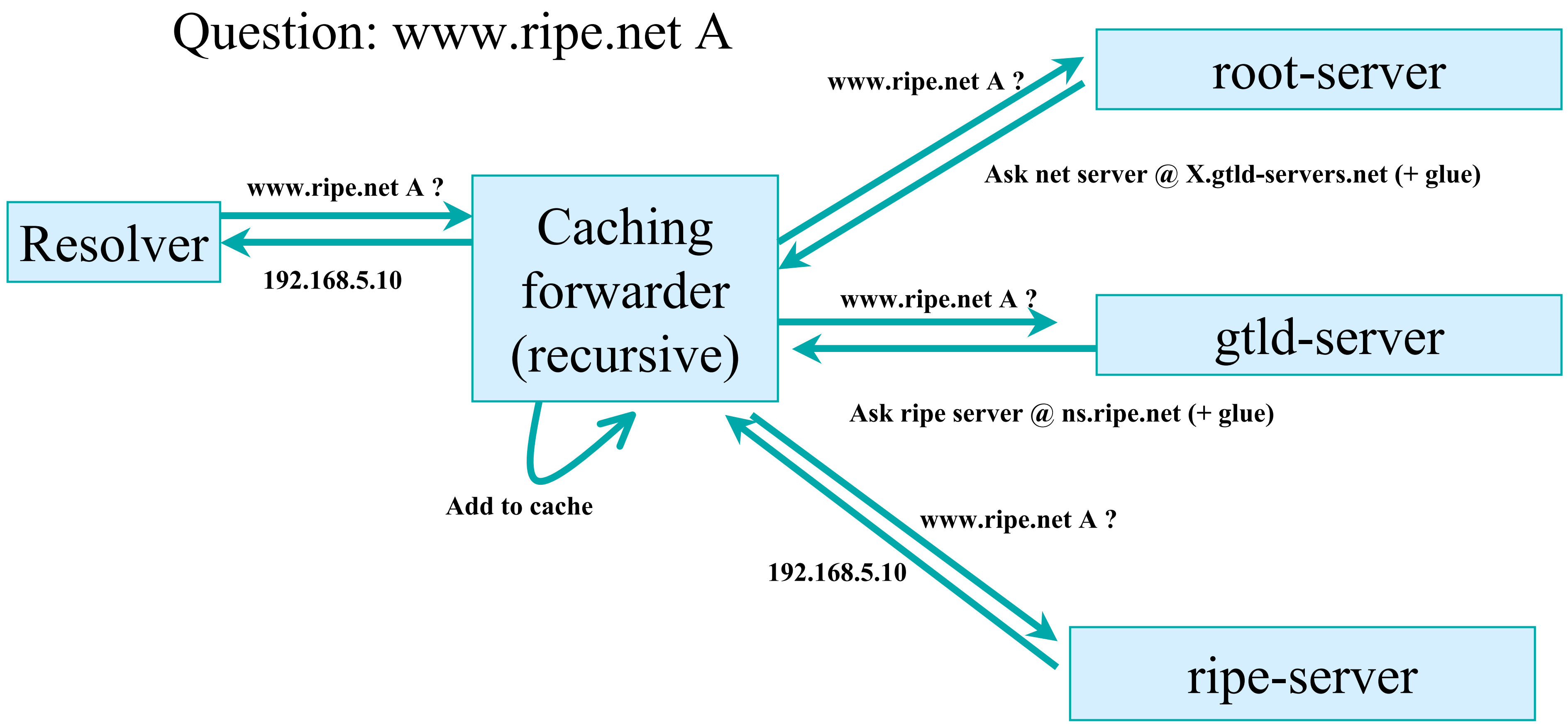
- Recursive servers do the actual lookups; they ask questions to the DNS on behalf of the clients.
- Answers are obtained from authoritative servers but the answers forwarded to the clients are marked as not authoritative
- Answers are stored for future reference in the cache

Concept: Resolvers

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

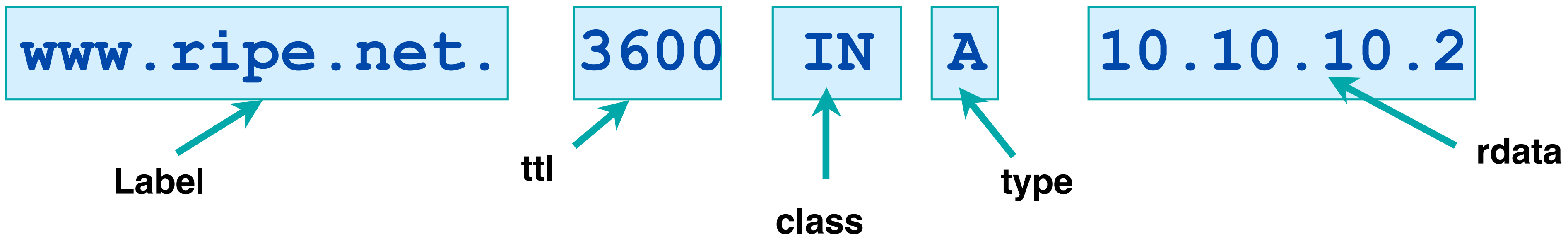
```
gethostbyname(char *name);
gethostbyaddr(char *addr, int len,
  type);
```

Concept: Resolving process & Cache



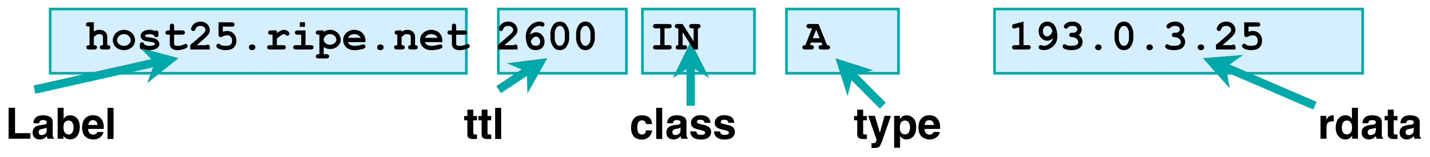
Concept: Resource Records (more detail)

- Resource records consist of it’s name, it’s TTL, it’s class, it’s type and it’s 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
```



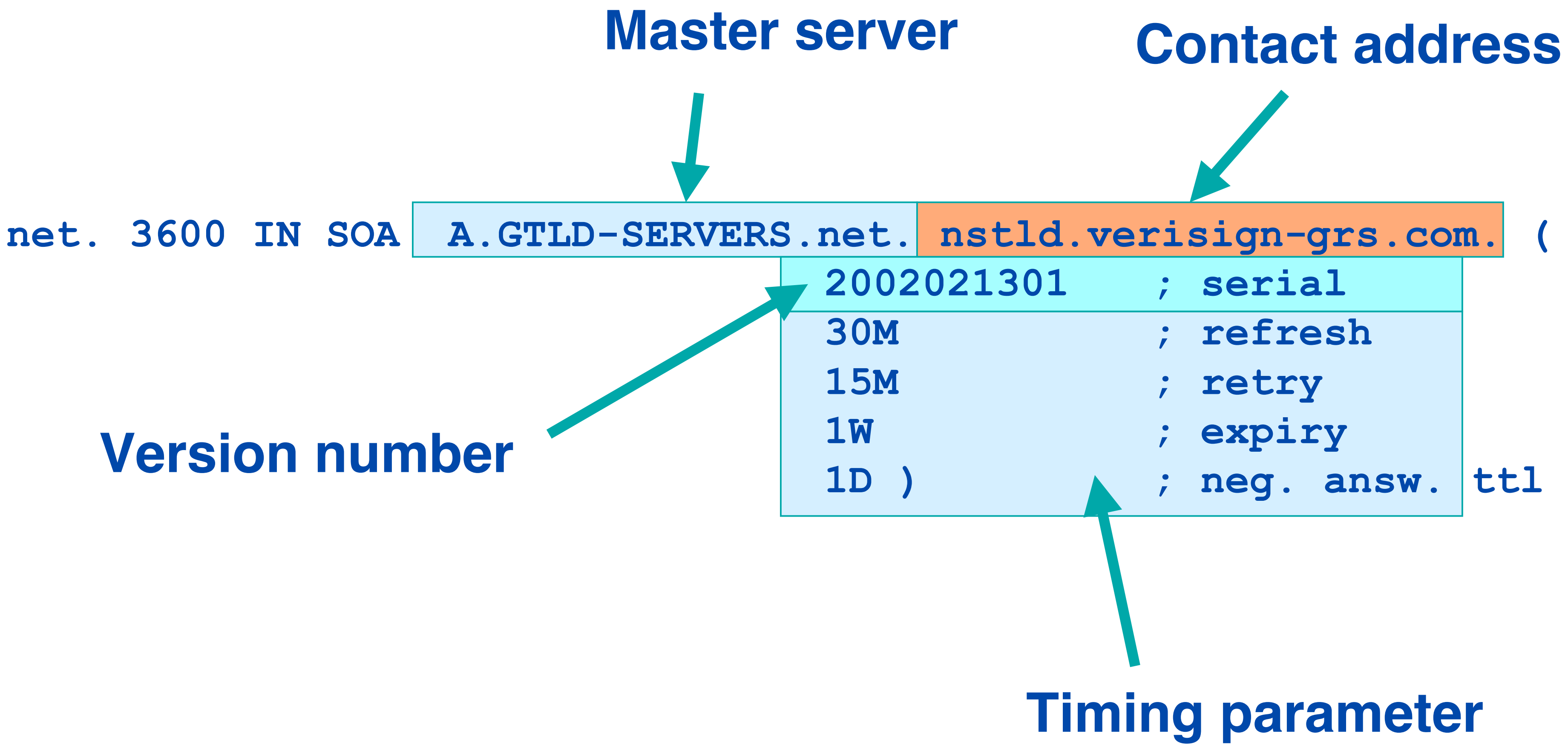
Resource Record: SOA and NS

- 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



```
net. 3600 IN SOA  A.GTLD-SERVERS.net. nstld.verisign-grs.com. (
                         2002021301     ; serial
                         30M            ; refresh
                         15M            ; retry
                         1W             ; expiry
                         1D )           ; neg. answ. ttl
```

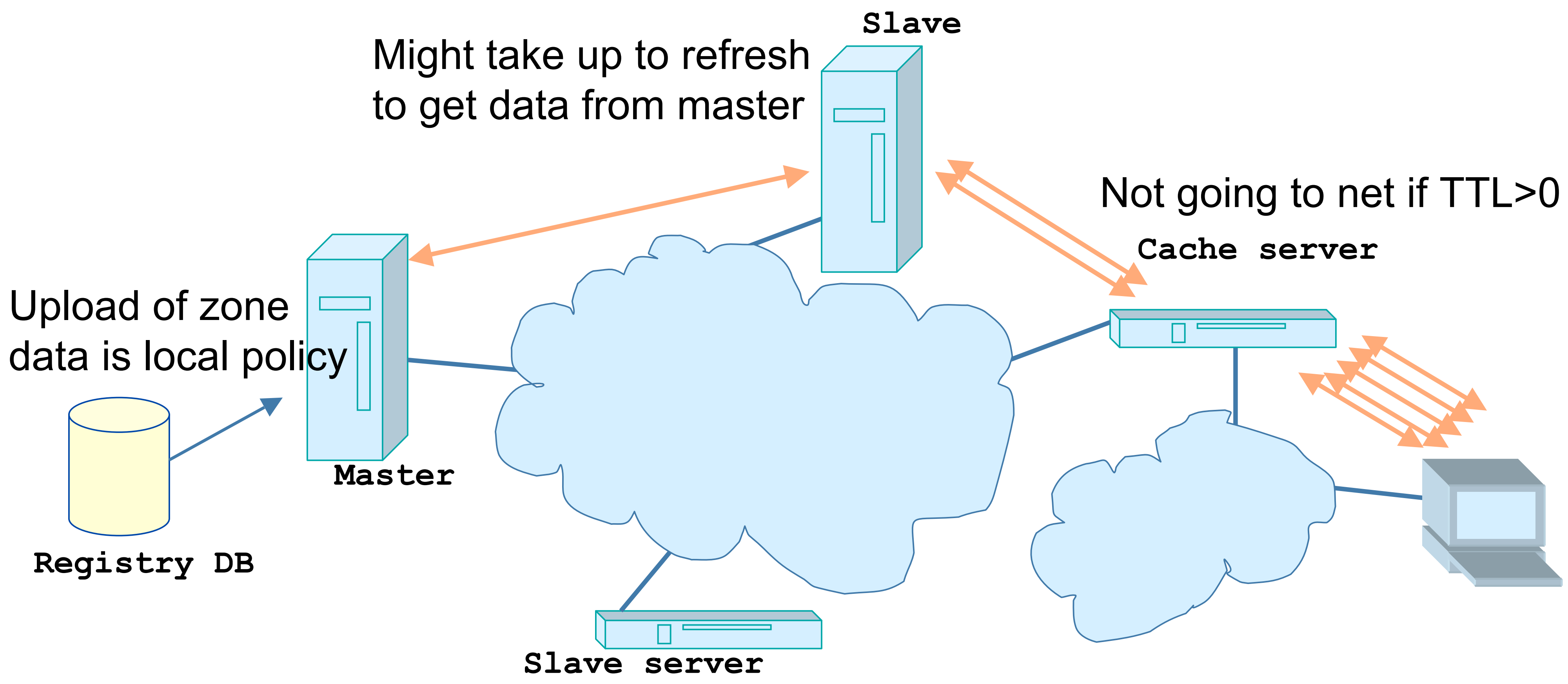
Concept: TTL and other Timers

- 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 administator
- 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**

```
ep.net. 3600 IN SOA flag.ep.net. (
                           bill\.manning.ep.net.
                           2002021301    ; serial
                           1h            ; refresh
                           30M           ; retry
                           1W            ; expiry
                           3600 )        ; neg. answ. ttl
```

- bill.manning@ep.net ➡ bill\.manning.ep.net
  - Should be the tech contact email
- Serial number: 32bit circular arithmetic
  - People often use date format
  - To be increased after editing
- The timers above qualify as reasonable (for some areas)

Authoritative NS records and related A records

```
secret-wg.org.                3600 IN NS  bert.secret-
wg.org.
secret-wg.org.                3600 IN NS  NS2.secret-wg.org.
bert.secret-wg.org.           3600 IN A  193.0.0.4
NS2.secret-wg.org.            3600 IN A  193.0.0.202
```

- 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 )
                            3600 IN TXT  "Demonstration and test zone"
bert.secret-wg.org.         3600 IN A    193.0.0.4
NS2.secret-wg.org.          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.
```

Jump to first page

Zone file format short cuts: 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 short cuts: ORIGIN

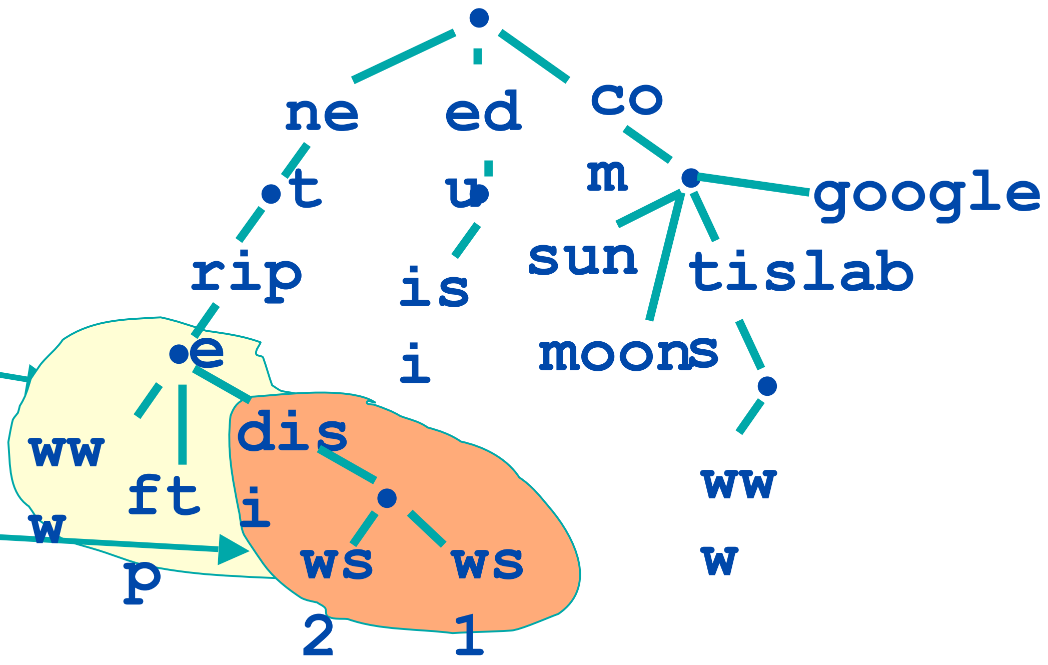
```
$TTL    3600 ; Default TTL directive
$ORIGIN secret-wg.org.
@              IN SOA bert (
                                       olaf\.kolkman.ripe.net.
                                       2002021301     ; serial
                                       1h             ; refresh
                                       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)

ripe.net zone

disi.ripe.net zone



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

Jump to first page

Troubleshooting

bill manning

bmanning@ep.net

Why Troubleshoot?

- What Can Go Wrong?
  - Misconfigured zone
  - Misconfigured server
  - Misconfigured host
  - Misconfigured network

Tools

- BIND Logging Facility
- named's built-in options
- ping and traceroute
- tcpdump and ethereal
- dig and nslookup

The Best Way To Handle Mistakes

- Assume You Will Make Them
- Prepare The Name Server via Logging
- Check the logs regularly

BIND Logging

- Telling named which messages to send
  - category specification
- Telling named where to send messages
  - channel specification

BIND Categories

- BIND has many categories
- Short descriptions of each can be found in the Administrator's Reference Manual (ARM)
  - Section 6.2.10.2, page 49
  - Example:

```
category dnssec {
 dnssec_log;
};
```

BIND channels

- BIND can use syslog
- BIND can direct output to other files
  - Example:

```
channel dnssec_log {
 file "seclog" versions 3 size 10m;
 print-time yes;
 print-category yes;
 print-severity yes;
 severity debug 3;
};
```

So You've Set Up A Server

- What testing should be done?
- From Basic liveness
  - Is the (right) server running?
  - Is the machine set up correctly?
- To data being served
  - Has the zone loaded?
  - Have zone transfers happened?

Checking the Configuration

- To see named start, use the -g flag
  - Keeps named process in the foreground
  - Prints some diagnostics
  - But does not execute logging
- When satisfied with named's start, kill the process and start without the flag
- Other option
  - named-checkconf
  - checks syntax only

Is the Server Running?

- Once the name server is thought to be running, make sure it is
  - `dig @localhost version.bind chaos txt`
- This makes the name server do the simplest lookup it can - its version string
- This also confirms which version you started
  - Common upgrade error: running the old version, forgetting to 'make install'

Is the Server Data Correct?

- Now that the server is the right one (executable)
  - `dig @localhost <zone> soa`
- Check the serial number to make sure the zone has loaded
- Also test changed data in case you forgot to update the serial number
- When we get to secondary servers, this check is made to see if the zone transfered

Is the Server Reachable?

- If the dig tests fail, its time to test the environment (machine, network)
  - `ping <server machine ip address>`
- This tests basic network flow, common errors
  - Network interface not UP
  - Routing to machine not correct
- Pinging 'locally' is useful, believe it or not
  - Confirms that the IP address is correctly configured

Is the Server Listening?

- If the server does not respond, but machine responds to ping
  - look at system log files
  - telnet server 53
- Server will run even if it can't open the network port
  - logs will show this
  - telnet opens a TCP connection, tests whether port was opened at all

Is the Server Logging the Right Stuff?

- Provoking and examining the logs
  - Log files only appear when needed
  - For example, dnssec logs will start only if 'trusted-keys' are configured and are used
  - Each category is triggered differently
    - Triggers may not be obvious

Using the Tools

- named itself
- dig/nslookup
- host diagnotics
- packet sniffers

Built in to named

- named -g to retain command line
  - named -g -c <conf file>
  - keeps named in foreground
- named -d <level>
  - sets the debug output volume
  - <level>'s aren't strictly defined
  - -d 3 is popular, -d 99 gives a lot of detail

dig

- domain internet groper
  - already used in examples
  - best tool for testing
  - shows query and response syntax
  - documentation
    - `man dig`
    - `dig -help`
- Included in named distribution

Non-BIND Tools

- Tools to make sure environment is right
  - Tools to look at server machine
  - Tools to test network
  - Tools to see what messages are on the network

ifconfig

- InterFace CONFIGuration
  - ifconfig -a
  - shows the status of interfaces
  - operating system utility
- Warning, during boot up, ifconfig may configure interfaces after named is started
  - named can't open delayed addresses
- Documentation
  - `man ifconfig`

ping

- Checks routing, machine health
  - Most useful if run from another host
  - Could be reason "no servers are reached"
  - Can be useful on local machine - to see if the interface is properly configured

traceroute

- If ping fails, traceroute can help pinpoint where trouble lies
  - the problem may be routing
  - if so - it's not named that needs fixing!
  - but is it important to know...

tcpdump and ethereal

- Once confident in the environment, problems with DNS set ups may exist
- To see what is happening in the protocol, use traffic sniffers
- These tools can help debug "forwarding" of queries
- ethereal can be retrived from
  - http://www.ethereal.com/

Jump to first page

Delegations in Forward DNS

bill manning

bmanning@ep.net

What is "Forward?"

- Generally, where the A records are
- "Domain Names" obtained from a parent zone
  - registrar if .com, .biz, .org., and some others
  - registry if a country code
  - another organization in other cases

Kinds of Delegations

- Contractural - outside organization
- Formal - another part of a large organization
- Informal - from yourself to yourself

Steps

- Negotiate with parent
- Set up a child zone
- Parent installs NS records
- Test

Get Domain from Parent

- TLD's have different approaches
- .com: ICANN-style
- ccTLDs may vary in application process

Child Name Servers

- Pick servers for your zone
  - In case a machine fails, two or more are recommended
  - In some situations, two are required
- Not on same network, etc.
  - In case of a network failure
  - Two machines behind one router...bad idea

Child Runs Standalone

- Child zones run without proper delegation
  - Make sure servers answer with the right information
  - Make sure zone transfers happen
  - Test your policies
- Test them by 'dig @' the server(s)

Insertion into Parent

- Parent adds the NS and glue records
- Test to see if records lead to right servers

Reverse DNS

bill manning
bmanning@ep.net

Outline

- General introduction
- IPv4 reverse DNS
  - Revere mapping and relation to address allocation
  - Problems and solutions for reverse mapping
- IPv6 reverse DNS

Addresses in the DNS

- Mapping from numbers to names
- It is just ordinary DNS
  - No different standards
  - No different operation
- But you might need a little background
  - There are some conventions
  - IPv6 is a moving/developing target
- First IPv4

Mapping of addresses to reverse

- Mapping from names to addresses is common:

```
flag.ep.net  A 198.32.4.13
```

- Sometimes one wants to know which name comes with a given address. If you can translate the address to a FQDN one can use the DNS

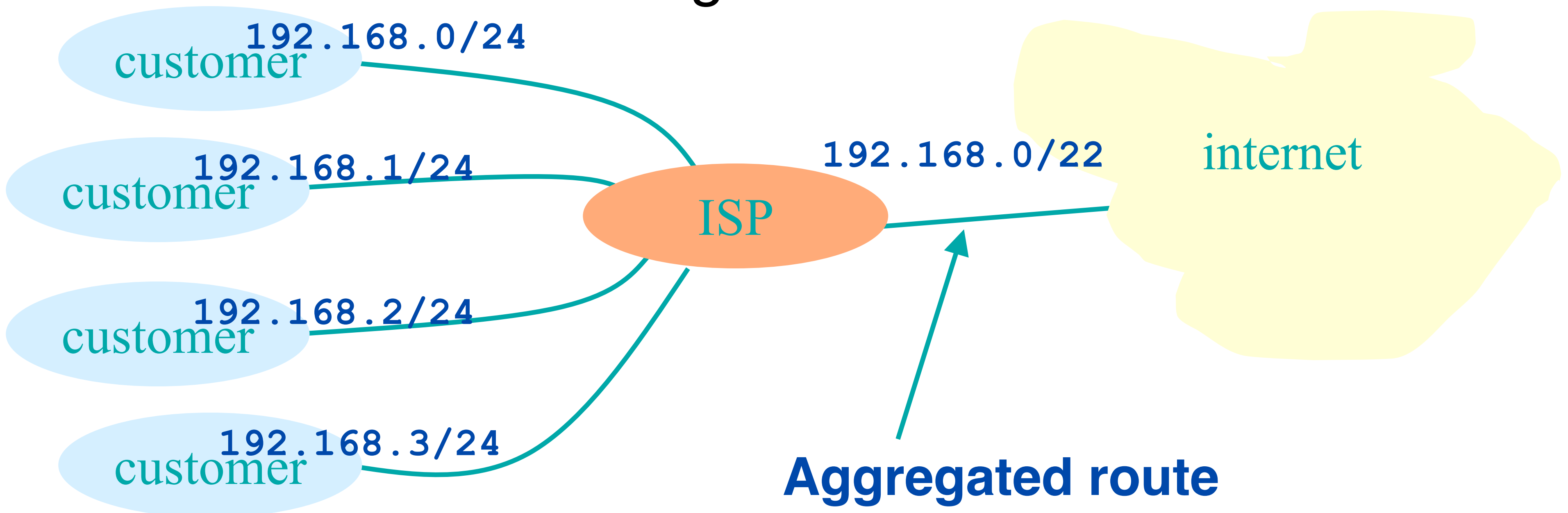
- Design goal: Delegate maintenance of the reverse DNS to the owner of the address block

Mapping the IPv4 address into the DNS: address allocation

- Address allocation is hierarchical:
  - blocks of addresses are allocated to ISPs
  - smaller blocks are allocated to client
  - clients will assign address blocks to end users
- Routing is based on destinations for given address blocks
  - Historically on 8 bit boundaries (Class A,B,C)
  - Classless Inter Domain Routing (CIDR)

Classless inter domain routing (CIDR)

- Routing table size (router memory) is a limited resource
- Goal of CIDR: aggregate many small address block into one larger block



Mapping the IPv4 address into the DNS: address blocks

- Address block notation:

  <address>/<number of significant bits>

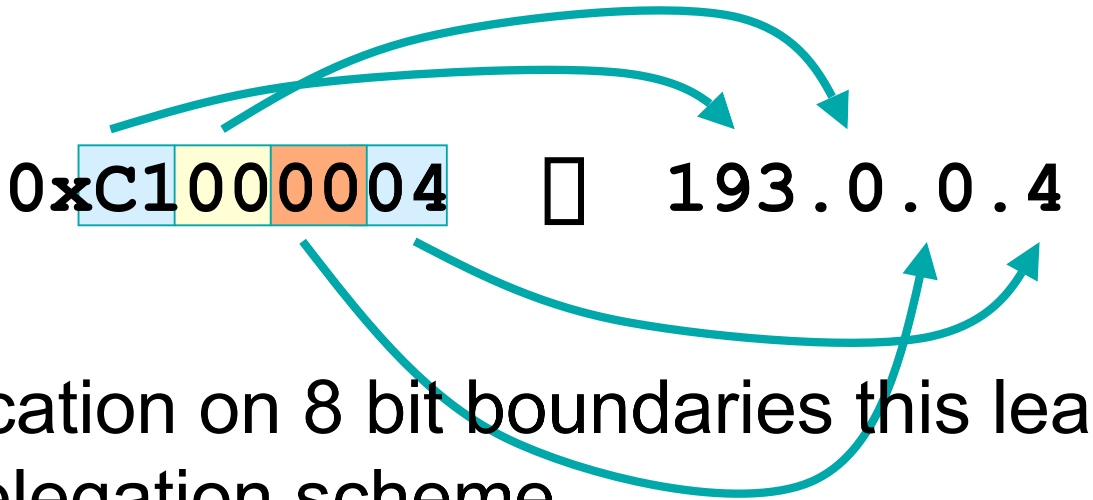
  For instance:

```
193.0.0.0/8 or short 193/8
193.165.64/19=
0xc1a54000/19 =
```

$$\underbrace{\texttt{1100 0001 1001 0101 001}}_{\text{19 bits}}\texttt{0 0000 0000 0000}$$

IPv4 address format

- An IP address is a 4 byte number normally represented by the decimal representation of the 4 bytes separated by dots



- With allocation on 8 bit boundaries this leads to a simple delegation scheme

Mapping the IPv4 address into the DNS

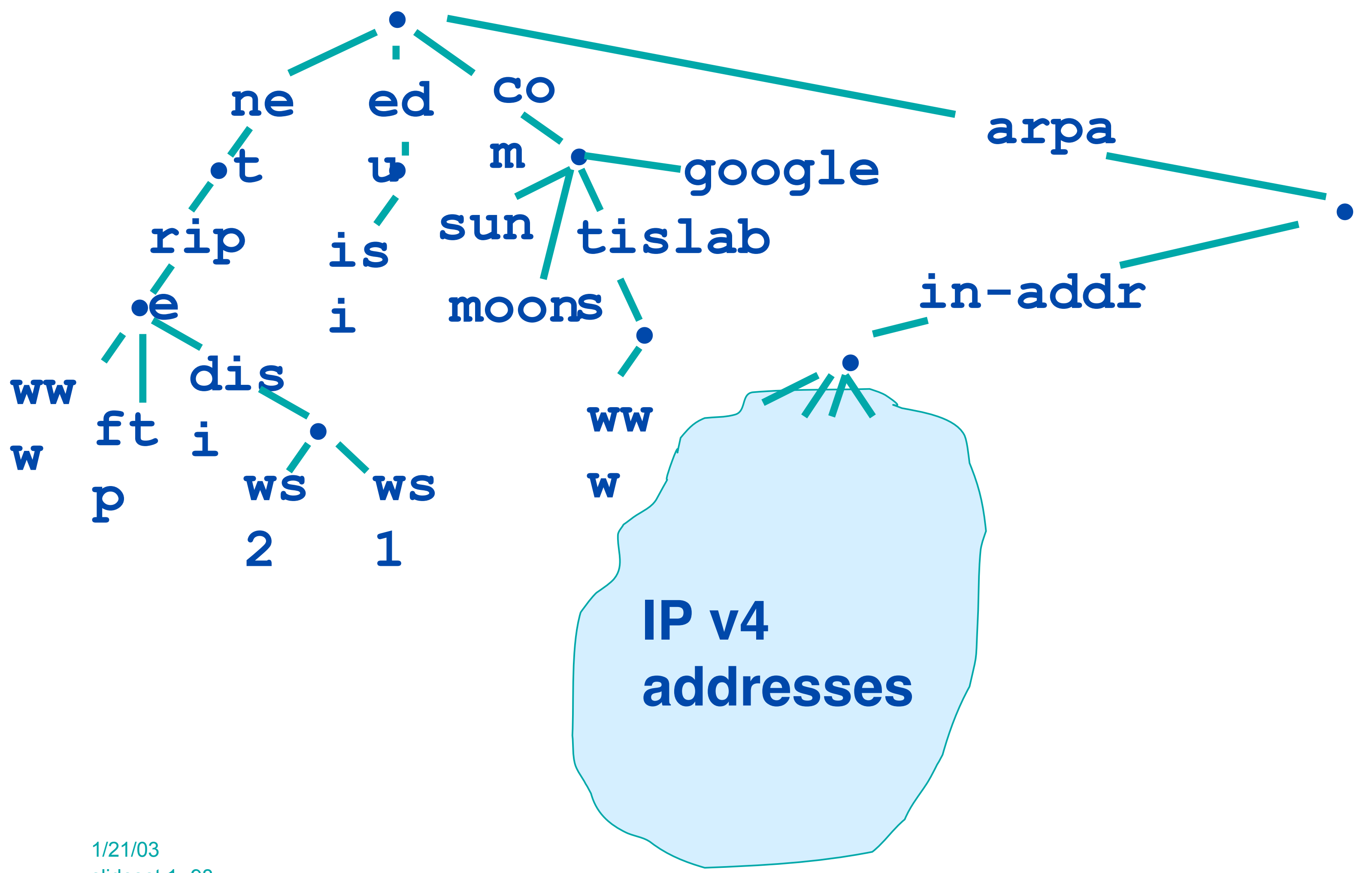
- Example 192.26.1.3
  - 192/8 is allocated to a RIR
  - 192.26/16 is allocated by RIR to LIR/ISP
  - 192.26.1/24 is assigned by ISP to a company.
- Delegation in the DNS:
  - root delegates 192 domain to RIR
  - RIR delegates “26” sub-zone to ISP
  - ISP delegates “1” sub-zone to company.
- Name that makes this possible: 1.26.192

Mapping addresses to names

- Revert the decimal representation:
  - 192.26.1.3 maps to 3.1.26.192 and put this under a top level domain.
  - For IPv4 this TLD is in-addr.arpa
- In the DNS one publishes PTR records to point back to the name:

```
4.0.0.193.in-addr.arpa 3600 IN PTR bert.secret-wg.org.
```

The reverse tree



Outline

- General introduction
- IPv4 reverse DNS
  - Revere mapping and relation to address allocation
  - → Problems and solutions for reverse mapping
- IPv6 reverse DNS

Mapping address to names: mapping problems

- In IPv4 the mapping is done on 8 bit boundaries (class full), address allocation is class less
- Zone administration does not always overlap address administration
- If you have a /19 of address space: divide it in /24s and request a delegation for each one of them as soon as you use the address space
- /25 and smaller we will cover later

Setting your reverse zones

- The reverse zone file is a regular zone file.
  - SOA and NS rrs in the APEX
  - Mostly PTR records in the zone itself
- Make sure the zone is served by the masters and slaves
- Bind9 has a $GENERATE directive that might be handy

A reverse zone example

```
$ORIGIN 1.168.192.in-addr.arpa.
@       3600   IN SOA bert.secret-wg.org.  (
                                  olaf\.kolkman.ripe.net.
                                  2002021301    ; serial
                                  1h            ; refresh
                                  30M           ; retry
                                  1W            ; expiry
                                  3600 )              ; neg. answ. ttl

        NS      ns.secret-wg.org.
        NS      ns2.secret-wg.org.

1       PTR     gw.secret-wg.org.
                router.secret-wg.org.
2       PTR     ns.secret-wg.org.
; BIND9 auto generate:  65 PTR host65.secret-wg.org
$GENERATE 65-127 $ PTR host$.secret-wg.org.
```

Note trailing dots

Getting a reverse delegation

- The procedure is registry dependent
  - For APNIC region read:

    http://www.apnic.net/db/revdel.html

    http://www.apnic.net/services/dns_guide.html

- Get a delegation from APNIC by filling adding a whois domain object:

  http://www.apnic.net/db/domain.html

- Only /16 and /24 delegations

Whois domain object

```
domain:      28.12.202.in-addr.arpa
descr:       in-addr.arpa zone for 28.12.202.in-addr.arpa
admin-c:     DNS3-AP
tech-c:      DNS3-AP
zone-c:      DNS3-AP
nserver:     ns.telstra.net
nserver:     rs.arin.net
nserver:     ns.myapnic.net
nserver:     svc00.apnic.net
nserver:     ns.apnic.net
mnt-by:      MAINT-APNIC-AP
mnt-lower:   MAINT-DNS-AP
changed:     inaddr@apnic.net 19990810
source:      APNIC
```

Jump to first page

Allocations smaller than /24

- Imagine a /25 address block delegated to a company by an ISP
- The company wants to maintain the reverse mapping of the address they use
- In the reverse DNS one can not delegate
- Use the 'classless inaddr' technique described in RFC 2317
- Based on the use of CNAME RRs
    - CNAME provide a means to alias names to another namespace

RFC2317 explained (1)

- 192.0.2.0/25 to organization A,
- 192.0.2.128/26 to organization B and
- 192.0.2.192/26 to organization C

```
$ORIGIN 2.0.192.in-addr.arpa.
  ;
  1       PTR     host1.organizationA.com.
  2       PTR     host2.organizationA.com.
  3       PTR     host3.organizationA.com.
  ;
  129     PTR     host1.organizationB.com.
  130     PTR     host2.organizationB.com.
  131     PTR     host3.organizationB.com.
  ;
  193      PTR     host1.organizationC.com.
  194      PTR     host2.organizationC.com.
  195      PTR     host3.organizationC.com.
```

RFC2317 explained (2)

- Generate a ‘sub domain’ for each address block and delegate these to the children
  - Name the sub domain after the address block
    - 0/25, 128/26, and 190/26
    - 0-127, 128-189, 190-255
    - orgA, orgB, orgC
- For each name in the zone create a CNAME that points into the delegated namespace e.g.:

```
1 CNAME   host1.orgA.2.0.193.inaddr-arpa.
```

RFC2317 explained(3)
Parent zone

```
$ORIGIN 2.0.192.in-addr.arpa.
  @        IN       SOA     my-ns.my.domain. (
                           hostmaster.my.domain.
                            ...)
  ; ...
  orgA       NS ns1.organizationA.com.
             NS ns2.organizationA.com.
  1                 CNAME   1.orgA
  2                 CNAME   2.orgA
  ; ...
  orgB       NS ns1.organizationB.com.
             NS ns2.organizationB.com.
  129               CNAME   129.orgB
  130               CNAME   130.orgB

  ;
```

RFC2317 explained(4) Children's zone

```
$ORIGIN orgA.2.0.192.in-addr.arpa.
  @       IN      SOA     ns1.organizationA.com. (
                         hostmaster.organizationA.com.
                         ...)
  ; ...
  @          NS ns1.organizationA.com.
             NS ns2.organizationA.com.
  1          PTR    host1.organizationA.com.
  2          PTR    host2.organizationB.com.
```

RFC2317 explained(5)

- You could also delegate to a forward zone
  - Eases maintaining consistency in mapping

```
$ORIGIN 1.168.192.in-addr.arpa
;
;
24       CNAME   in24.foo.net.
25       CNAME   in25.foo.net.
26       CNAME   in26.foo.net.
27       CNAME   in27.foo.net.
28       CNAME   in28.foo.net.
;
; etc
```

```
$ORIGIN foo.net.
;
www    A    192.168.1.24
in24  PTR   www.foo.net.
ftp    A    192.168.1.25
in25  PTR   ftp.foo.net.
silver A    192.168.1.26
in26   PTR   silver.foo.net.
;
; etc
```

Outline

- General introduction
- IPv4 reverse DNS
- IPv6 reverse DNS
  - ◆ IPv6 addresses
  - ◆ IPv6 in the forward tree
  - ◆ IPv6 in the reverse tree

IPv6 addresses

- 128 bits
  - 64 low order bits “host” identifier
    - e.g. a mapping of the hosts’ Ethernet address
  - 64 high order bits “network” identifier
    - Further subdivision inside network id.
- Let’s look at notation first, then at further subdivision

IPv6 address Notation

- 16 bit integers (in Hex) separated by colons

  FEDC:BA98:7654:3210:FEDC:BA98:7654:3210

  1080:0000:0000:0000:0008:0800:200C:417A

- Leading zeros can be skipped

  1080:0:0:0:8:800:200C:417A

- Consecutive NULL 16-bit numbers □ "::"

  1080::8:800:200C:417A

IPv6 addresses

- For globally routable unicast addresses the 1st 3 bits are set to "001"
- Unicast addresses are further subdivided in "aggregates"
- More address classes available like:
  - Link local: fe80/10
  - Multicast: ff00/8
  - Mapped IPv4 address: 0::ffff:0:0:0:0/96

Globally routable unicast addresses

- 1st 3 bits are format prefix
- last 16 bits of network ID are used for ‘sites’
- RIRs currently allocate /32 blocks of address space (earlier delegations were smaller)
- The policy still moving target

```
|3 |           45            |  16|         64 bits          |
+--+-------------------------+----+--------------------------+
|FP|                         |SITE|      Interface ID        |
|  |                         | ID |                          |
+--+-------------------------+----+--------------------------+
```

Jump to first page

Outline

- General introduction
- IPv4 reverse DNS
- IPv6 reverse DNS
  - IPv6 addresses
  - IPv6 in the forward tree
  - → IPv6 in the reverse tree

IPv6 address representation in the DNS

- Multiple RR records for name to number
  - AAAA
  - A6 (depricated)

- Multiple ways to map address to DNS name
  - nibble notation
  - bit strings and nibbles (depricated)

AAAA RR

- Name to number mapping
- Similar to A RR for IPv4
- Uses the ‘common’ representation of the address

```
$ORIGIN ep.net.
flag      3600 IN   AAAA 3ffe:805:4::cafe:babe
```

Outline

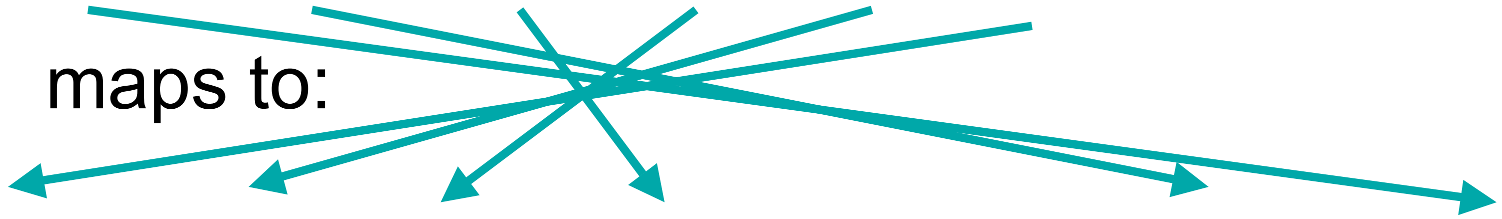
- General introduction
- IPv4 reverse DNS
- IPv6 reverse DNS
  - IPv6 addresses
  - IPv6 in the forward tree
  - IPv6 in the reverse tree
    - nibbles in ip6.arpa
    - DNAMEs and Bitlabels...

Reverse DNS

- Just as with IPv4 the responsibility for maintaining the reverse map can be delegated through the address hierarchy
- Number is translated into 4 bit nibbles under the ip6.arpa (ip6.int) TLD.

```
2001:0238::a00:46ff:fe06:1460
```

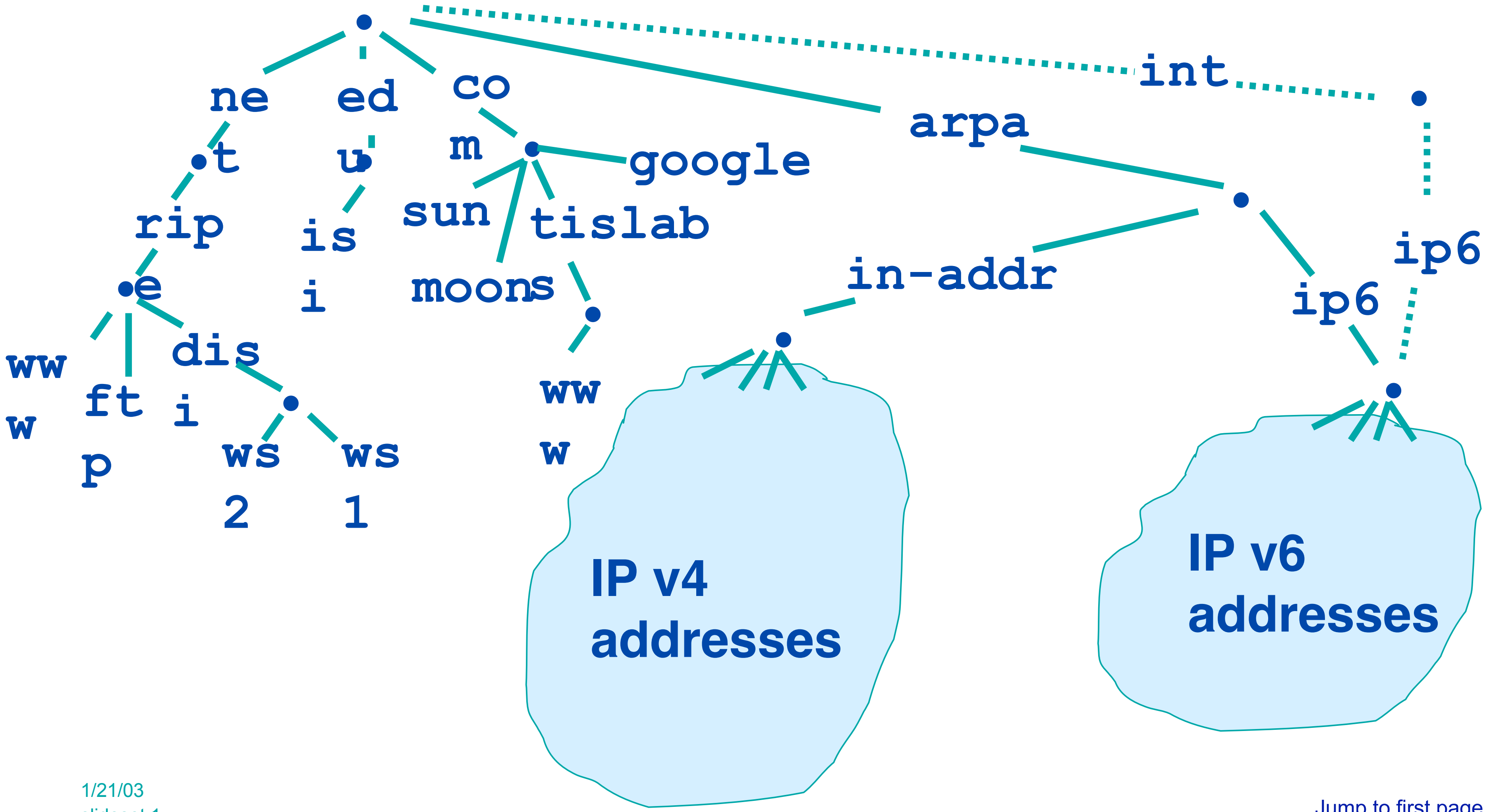
maps to:



```
0.6.4.1.6.0.e.f.f.f.6.4.0.0.a.0.0.0.0.0.0.0.0.0.8.3.2.0.1.0.0.2.ip6.arpa.
```

Jump to first page

The reverse tree



Setting up reverse for SUB TLA

- Remember the address format for initial allocation

```
FP | /20 regio    |    /35 allocations|
|3 |      20      |                   |
|--|-------------|-------------------|----......
00100000000000001000001000111000000?
------------ 35 bits --------------
   2   0   0   1:  0   2   3   8   0/35
```

Can be 1 or 0

- Delegation for two /36

```
0.8.3.2.0.1.0.0.2.ip6.arpa
1.8.3.2.0.1.0.0.2.ip6.arpa
```

DNAME RR
delegation name

- DNAME RRs resemble CNAME RRs
- The DNAME substitutes the suffix of a domain name with another
- Works as alias syntheses
- Example: all records for secret-wg.org in the ripe.net zone.

```
secret-wg.org          DNAME   secret.ripe.net.
```

- A query to host.secret-wg.org would return:

```
host.secret-wg.org CNAME host.secret.ripe.net.
```

DNAME exercise

- Take the other IP address for host.secret-wg.org and write out a DNAME chain.
- DNAME chains do not make non-4bit boundaries simpler.

DNS data and the transport layer

- In principle the transport layer does not have influence on DNS data;
  - Data can be published by servers running on IPv4 or IPv6, content should not differ
  - Transition problem: IPv6 client might not be able to see IPv6 servers and vice verse
  - Transition problems are by far not solved
- Exception to above: IPv4 mapped addresses
  - Mapping is depended on OS libraries

DNSSEC

Introduction to
Concepts

bill manning
bmanning@ep.net

Introduction
What and Why and ...

- WHAT:
  - DNS, DNSSEC and latest developments
- WHY:
  - Raise awareness on DNSSEC
  - Provide handles to start deployment
- FOR:
  - Folk that do DNS and want to know more about DNSSEC

OUTLINE

PART I - CONCEPTS

➡ **<u>Introduction</u>**
- DNSSEC mechanisms
  - …
  - ...
- Conclusions

PART II - OPERATIONS

Workshop

DNS
Known Concepts

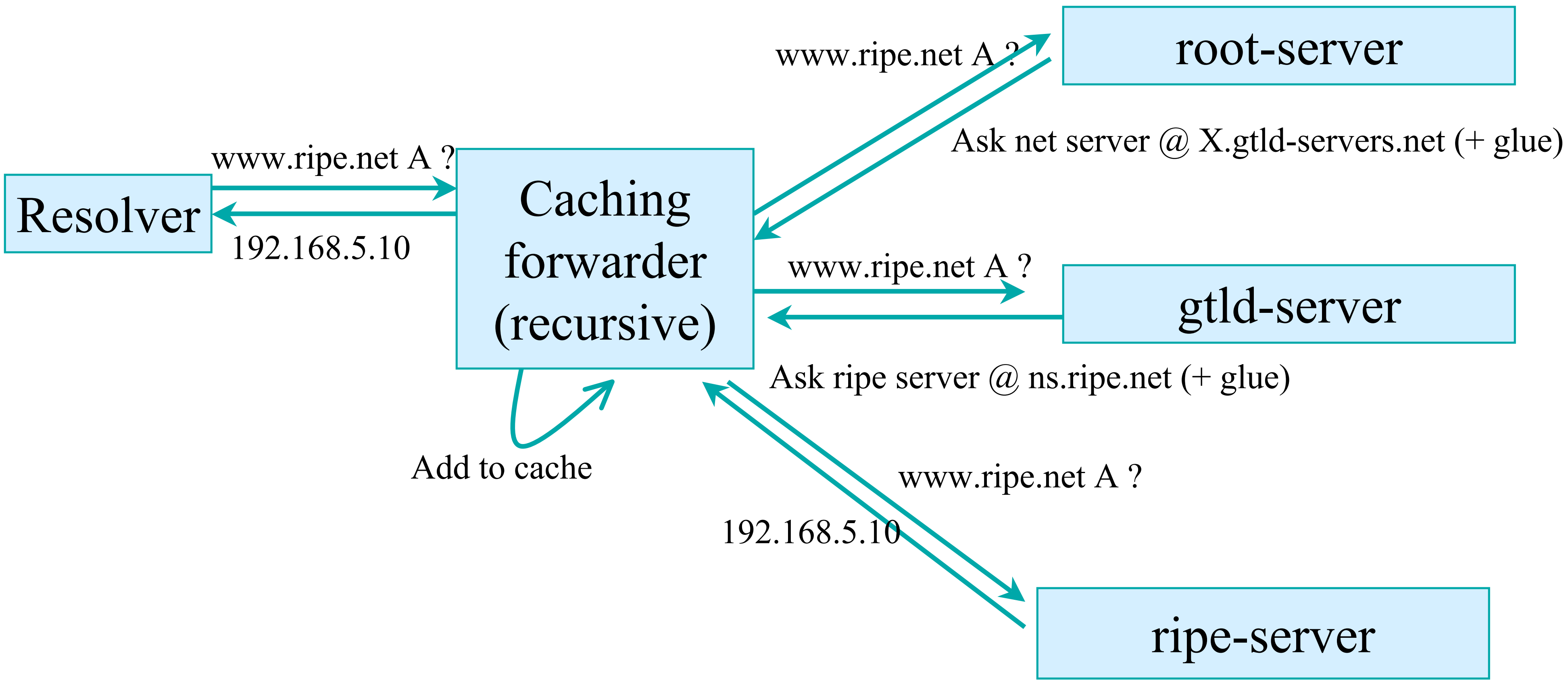
- Known DNS concepts:
  - Delegation, Referral, Zone, RRs, label, RDATA, Authoritative server, caching forwarder, resolver, SOA parameters

For part II:

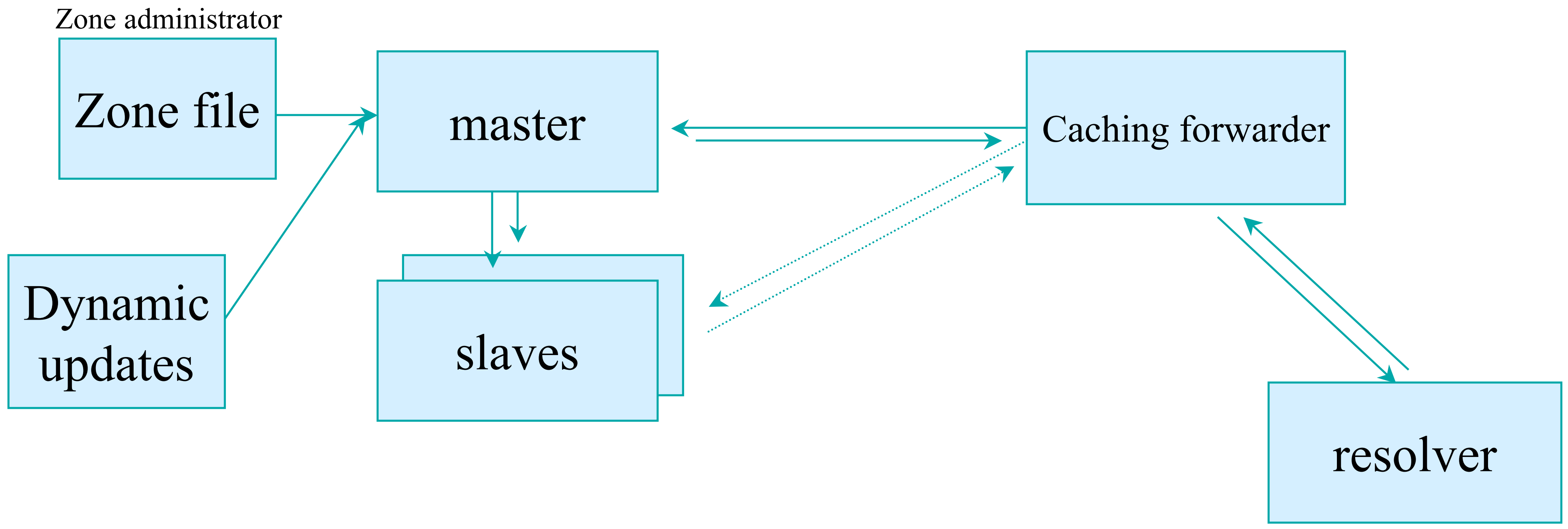
- You know how to operate a BIND server
  - Bind 8 or 9 named.conf, writing zone files

DNS resolving

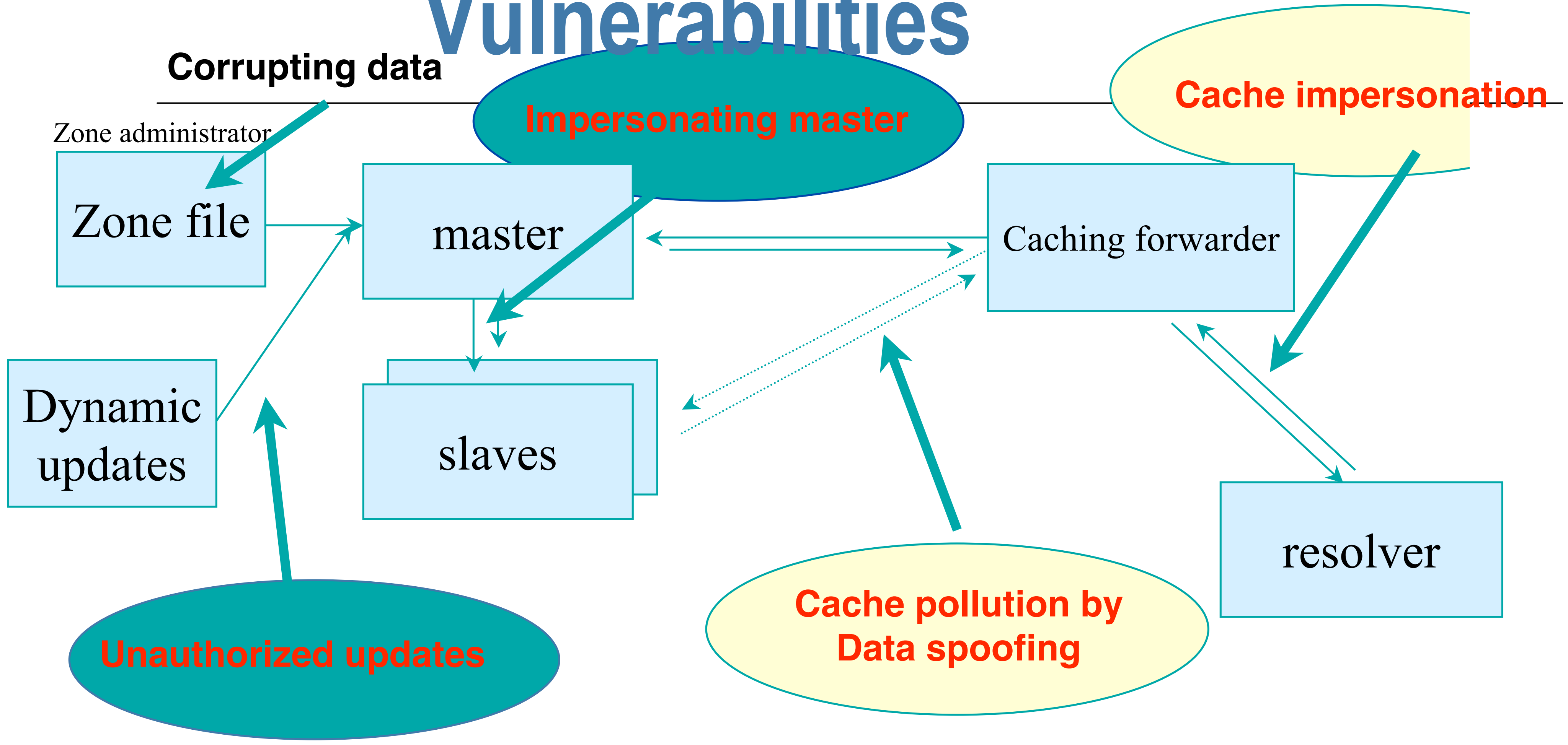
Question: www.ripe.net A



DNS
Data flow



DNS Vulnerabilities



Server Protection

DATA Protection

DNS protocol vulnerability

- DNS data can be spoofed and corrupted on its way between server and resolver or forwarder
- The DNS protocol does not allow you to check the validity of DNS data
  - Exploited by bugs in resolver implementation (predictable transaction ID)
  - Polluted caching forwarders can cause harm for quite some time (TTL)
  - Corrupted DNS data might end up in caches and stay there for a long time
- How does a slave (secondary) knows it is talking to the proper master (primary)?

Why: To protect the DNS system itself

DNSSEC protects against data spoofing and corruption

- DNSSEC also provides mechanisms to authenticate servers
- DNSSEC provides mechanisms to establish authenticity and integrity

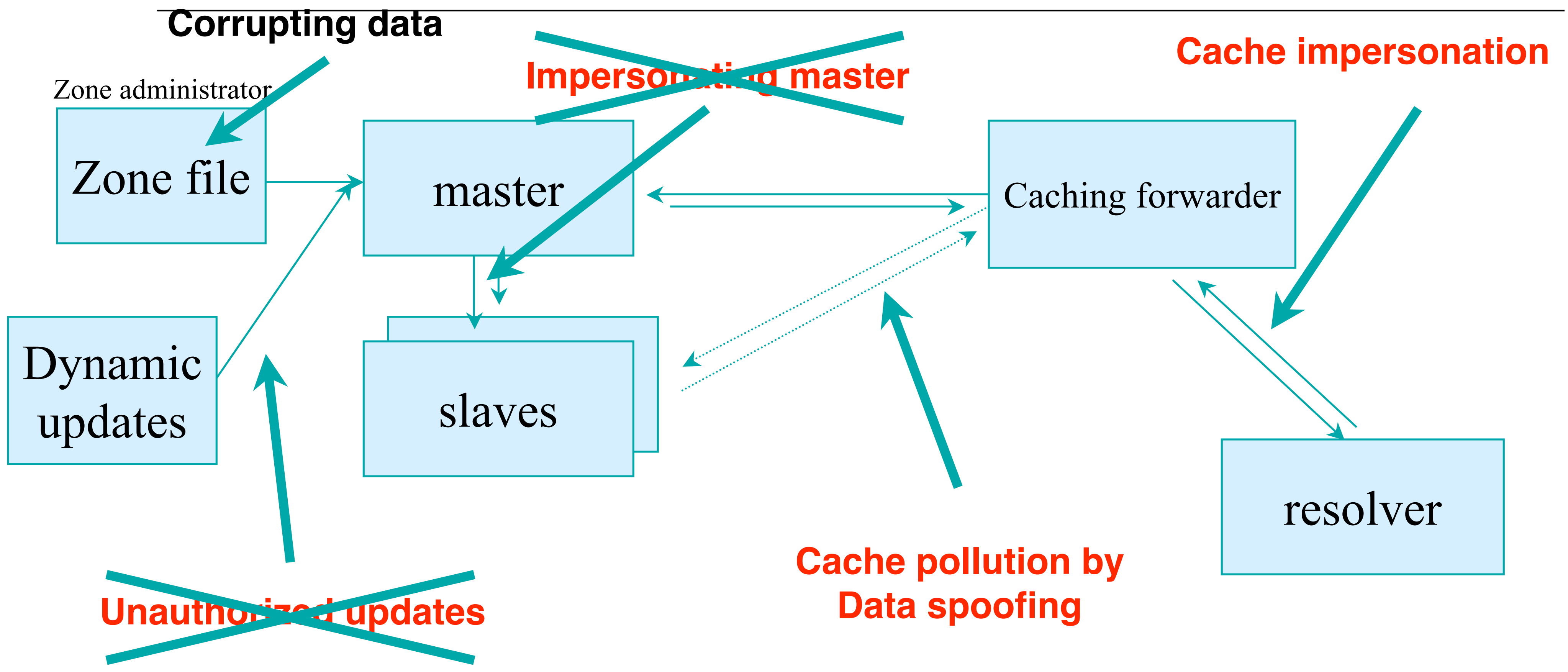
- A secure DNS will be used as a PKI
  - Even though it does not have all attributes of a PKI

OUTLINE
Part I - Concepts

- Introduction
- ➡ **DNSSEC mechanisms**
    - ➡ **DNSSEC mechanisms to authenticate servers**

      We will only discuss this shortly
    - DNSSEC mechanisms to establish authenticity and integrity
- Conclusions

Vulnerabilities protected by TSIG



DNSSEC mechanisms to authenticate servers

- Transaction Signature: TSIG (RFC 2845)
  - authorizing dynamic updates
  - zone transfers
  - authentication of caching forwarders
  - This feature can be used without deploying other features of DNSSEC
- In server configuration, not in zone file
- Based on shared secret
  - Usage limited due to key distribution limitations

DNSSEC mechanisms to authenticate servers (cont'd)

- Alternatively one can use SIG0
  - Not widely used yet
  - Works well in dynamic update environment
- Public key algorithm
  - Authentication against a public key published in the DNS
- TSIG/SIG0 signs a complete DNS request/response with time stamp
  - NTP synchronization!!!

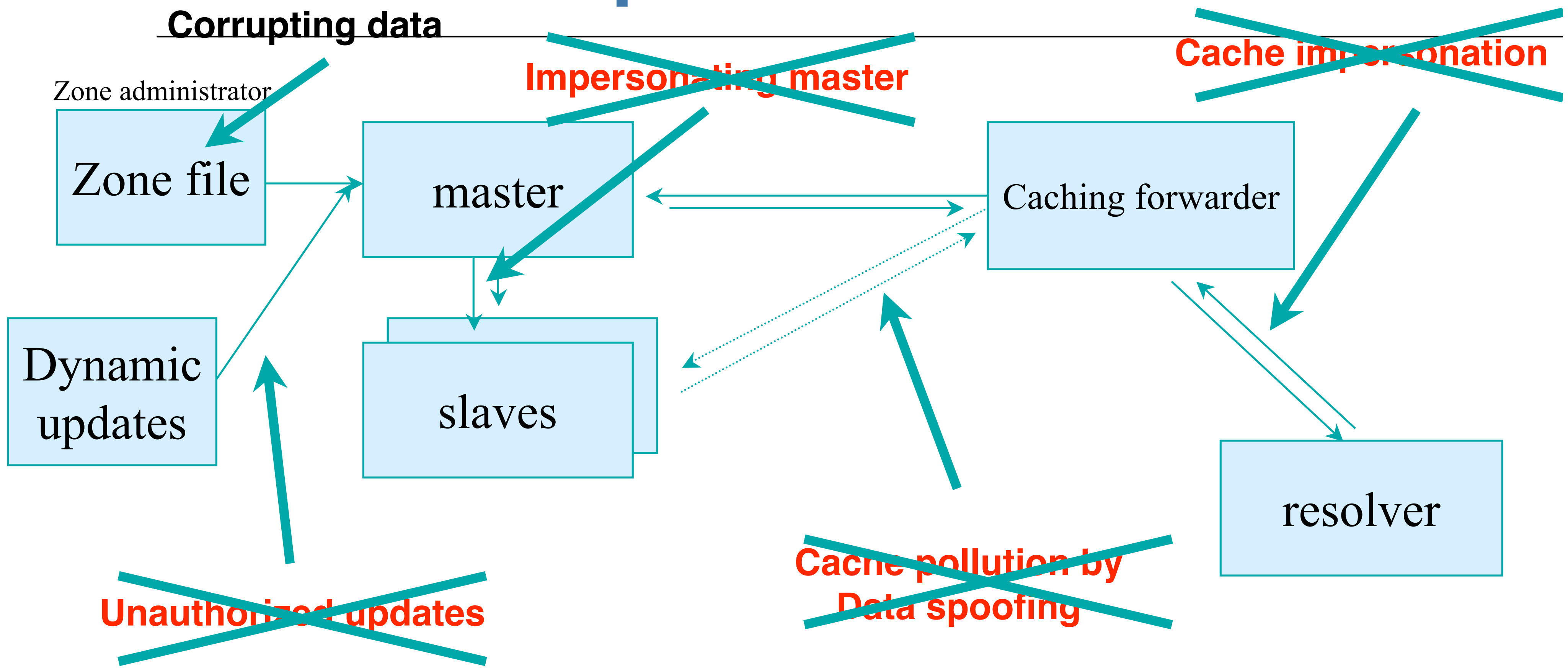
OUTLINE
Part I - Concepts

….

- **DNSSEC mechanisms**
  - DNSSEC mechanisms to authenticate servers
  - **DNSSEC mechanisms to establish authenticity and integrity**
    - **Quick overview**
    - New RRs
    - Using public key cryptography to sign a single zone
    - Delegating signing authority ; building chains of trust
    - Key exchange and rollovers

….

Vulnerabilities covered by data protection



DNSSEC on 1 page

- Data authenticity and integrity by SIGning the resource records
- Public KEYs can be used to verify the SIGs
- Children sign their zones with their private key. The authenticity of their KEY is established by a SIGnature over that key by the parent
- In the ideal case, only one public KEY needs to be distributed off-band

Authenticity and Integrity

- ~~We want to check authenticity and integrity of DNS~~ data
- Authenticity: Is the data published by the entity we think is authoritative?
- Integrity: Is the data received the same as what was published?
- Public Key cryptography helps to answer these questions
  - One can use signatures to check both integrity and authenticity of data
  - One can verify the authenticity of signatures

Public Key Crypto I

- Two keys available: a secret key and a public key
- Simplified:
  - If you know the public key, you can decrypt data encrypted with the secret key
    - Usually an encrypted hash value over a published piece of information; the owner is the only person who can construct the secret. Hence this a signature
  - If you know the secret key, you can decrypt data encrypted with the public key
    - Usually an encrypted key for symmetric cipher
- PGP uses both, DNSSEC only uses signatures

Public Key Crypto II

- The security of the cryptosystem is based on a set of mathematical problems for which guessing a solution requires scanning a huge solution space (*e.g.* factorization)
- Algorithms *e.g.*: DSA, RSA, elliptic curve
- Public keys need to be distributed. Secret keys need to be kept secret
  - Both key distribution and secrecy are not trivia
- Public key cryptography is ‘slow’

OUTLINE
Part I - Concepts

- Introduction
- **DNSSEC mechanisms**
  - DNSSEC mechanisms to authenticate servers
  - **DNSSEC mechanisms to establish authenticity and integrity**
    - Quick overview
    - ➡ **New RRs**
    - DNSSEC signing of an isolated zone
    - Delegating signing authority ; building chains of trust
    - Key exchange and rollovers
- Conclusions

DNSSEC new RRs

- 3 Public key related RRs
  - SIG signature over RRset made using private key
  - KEY public key, needed for verifying a SIG over a RRset
  - DS 'Pointer' for building chains of trust
- One RR for internal consistency (authenticated denial of data)
  - NXT RR to indicate which RRset is the next one in the zone
- For non DNSSEC public keys: CERT/APPKEY(?)

Recap: RRs and RRsets

- Resource Record:
  - label class ttl type rdata

```
www.ripe.net        IN     7200  A        192.168.10.3
```

- All RRs of a given label, class, type make up an RRset:

```
www.ripe.net        IN     7200  A        192.168.10.3
                           A        10.0.0.3
```

- In DNSSEC the RRsets are signed, not the individual RRs

NXT record

- Authenticated non-existence of TYPEs and labels

  Example ripe.net zone (leaving out the SIGs):

```
@          SOA    …..
           NS     NS.ripe.net.
           NXT    ns SOA NS NXT SIG
NS         A      192.168.10.1
           NXT    mailbox A NXT SIG
mailbox    A      192.168.10.2
           NXT    www A NXT SIG
WWW        A      192.168.10.3
           NXT    bla.foo A NXT SIG
```

query for popserver.ripe.net would return:

aa bit set   RCODE=NXDOMAIN

authority: mailbox NXT www A

The DNSSEC RR records

- Following slides show RR in detail
- The bits are as they appear ‘on the wire’
- The text as in a zone file

NXT RDATA

- next domain name
- N*32 bit type bit map

- Example:

www.ripe.net.   3600 IN **NXT**   ripe.net. A SIG NXT

NXT opt-in variant

- New variety of the NXT resource record
  - Introduced to cope with the problem that in a secure zone each name is accompanied by a NXT RR with a SIG
- Instead of authenticated denial of existence it indicates authenticated denial of security
- The change in semantic is indicated by leaving the NXT from the bitmap

Jump to first page

NXT opt-in variant

- Still under discussion in the IETF
  - No consensus yet
  - First implementations are being developed

```
a.com     ns       ns.a.com
a.com     NXT      SIG NS w.com
          SIG      NXT ....

b.com     NS       ns.b.com
c.com     NS       ns.c.com

w.com     NS       ns.w.com
          NXT      SIG NS z.com
          SIG      NXT ....
z.com     NS       ns.z.com
          NXT      SIG NS .com
          SIG      NXT
```

Question for non-existent ba.com will return:

NXDOMAIN
Auth: A.COM NXT   SIG NS w.com

One can not be sure ba.com does not exist.

KEY RDATA

- 16 bits FLAGS
- 8 bits protocol
- 8 bits algorithm
- N*32 bits public key

Examples:

```
ripe.net. 3600 IN KEY      256 3 3 (
                 AQOvhvXXU61Pr8sCwELcqqq1g4JJ
                 CALG4C9EtraBKVd+vGIF/unwigfLOA
                 O3nHp/cgGrG6gJYe8OWKYNgq3kDChN)
sub.ripe.net. 3600 IN KEY     49408 3 3
```

SIG RDATA

- 16 bits type covered
- 8 bits algorithm
- 8 bits labels covered
- 32 bit original TTL
- 32 bit signature expiration
- 32 bit signature inception
- 16 bit key tag
- signers name
- signature field

```
www.ripe.net.   3600 IN  SIG   A 1 3 3600 20010504144523 (
                20010404144523 3112 ripe.net.
                VJ+8ijXvbrTLeoAiEk/qMrdudRnYZM1VlqhN
                vhYuAcYKe2X/jqYfMfjfSUrmhPo+0/GOZjW
                66DJubZPmNSYXw== )
```

DS resource record

- Essentially a pointer to the next key in the chain of trust
- Still in draft but expected to become part of the standard
- Parent is authoritative, is NOT published in the child’s apex
- Solves complicated rollover problems

- More details on the working of DS will follow later

OUTLINE
Part I - Concepts

- Introduction
- **DNSSEC mechanisms**
  - DNSSEC mechanisms to authenticate servers
  - **DNSSEC mechanisms to establish authenticity and integrity**
    - Quick overview
    - New RRs
    - ➡ **DNSSEC signing of an isolated zone**
    - Delegating signing authority ; building chains of trust
    - Key exchange and rollovers
- Conclusions

DNSSEC signing of an isolated zone

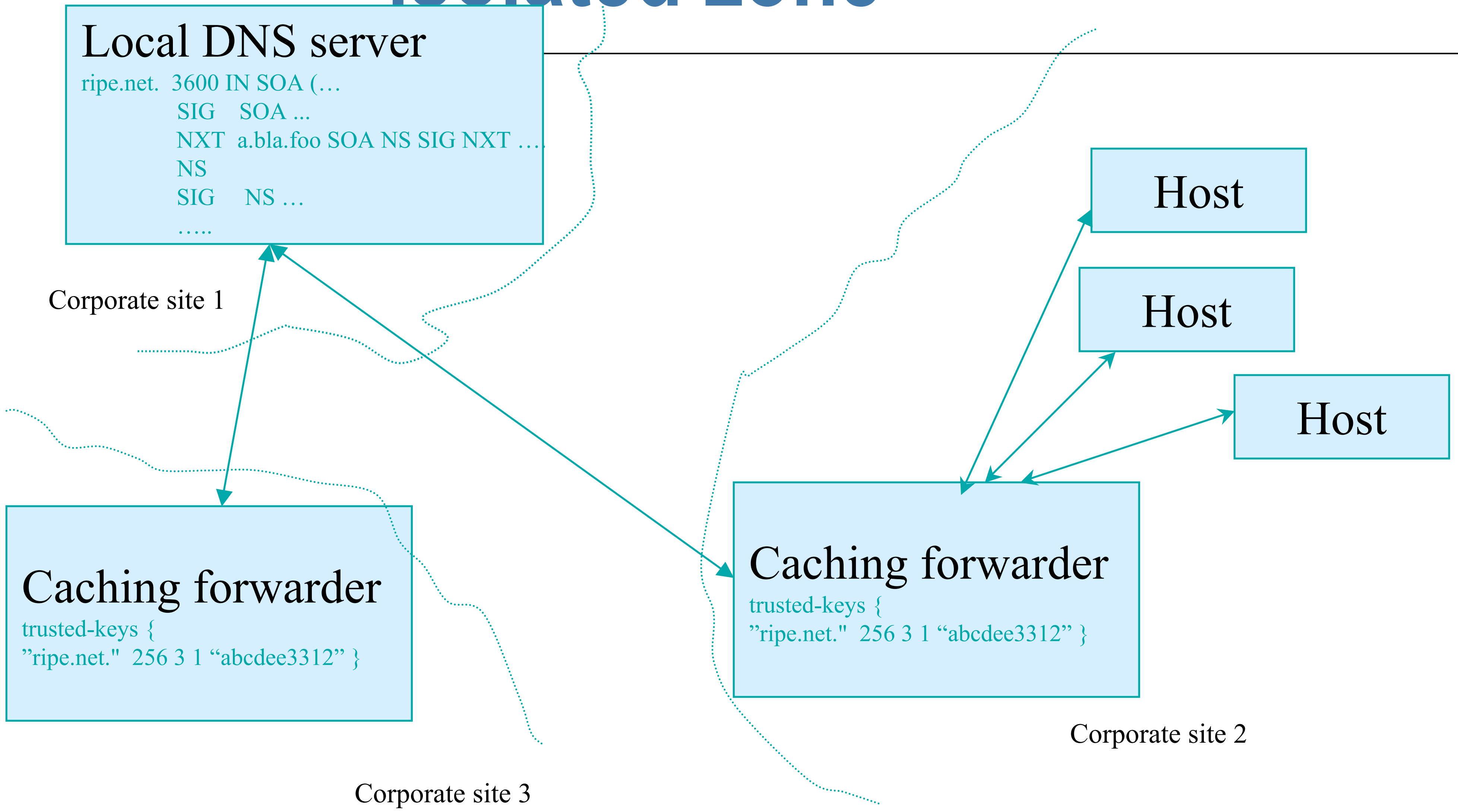
- The 2 steps to secure a zone for ‘corporate’ use
- Sign your zone

  signing will:

  - sort the zone
  - insert the NXT records
  - insert SIG containing a signature over each RRset

  The signature is made with your private key
- Distribute the Public KEY to those that need to be able to trust your zone
  - They configure the key in their resolver

DNSSEC signing of an isolated zone



Jump to first page

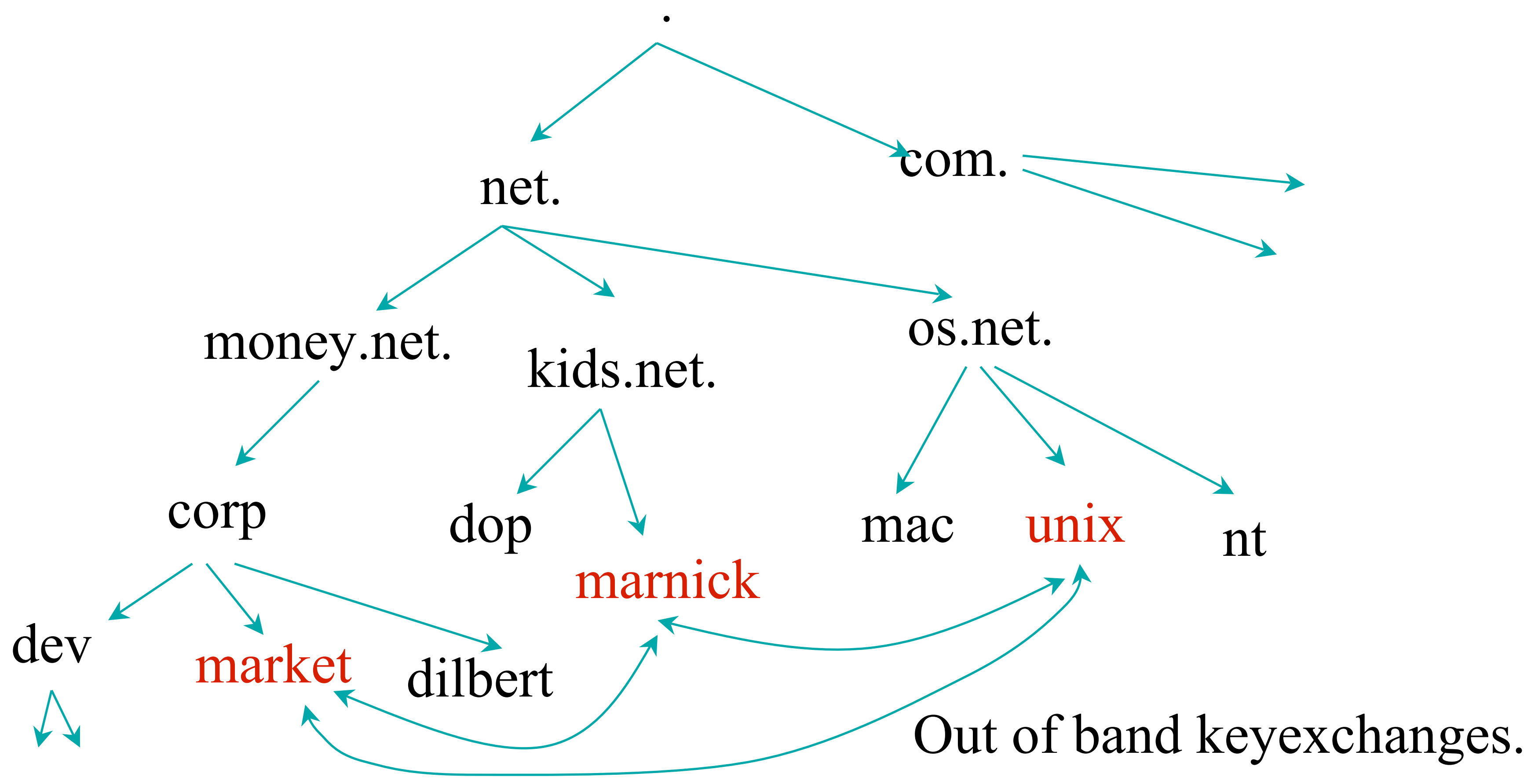
OUTLINE

- Introduction
- **DNSSEC mechanisms**
  - DNSSEC mechanisms to authenticate servers
  - **DNSSEC mechanisms to establish authenticity and integrity**
    - Quick overview
    - New RRs
    - DNSSEC signing of an isolated zone
    - ➡ **Delegating signing authority ; building chains of trust**
    - Key exchange and rollovers
- Conclusions

Jump to first page

Securing a DNS zone tree

- Key distribution problem



Using the DNS to distribute keys

- Building chains of trust from the root down the DNS tree
- Tools: KEY, SIG and DS records
- This material is based on new developments
  - on not yet implemented drafts
  - will be incompatible with RFC2535 .i.e. current tools !
  - Probable deployment in the coming months
  - No tools to try this at home

SIG RDATA

Recap for next slides

```
www.ripe.net.   3600 IN  SIG   A 1 3 3600
20010504144523 (   20010404144523 3112 ripe.net.
                   VJ+8ijXvbrTLeoAiEk/qMrdudRnYZM1VlqhN
                   vhYuAcYKe2X/jqYfMfjfSUrmhPo+0/GOZjW
                   66DJubZPmNSYXw== )
```

This field indicates the signer.

Delegation Signer (DS) in more detail

- Delegation Signer: The parent delegates authority to sign DNS RRs to the child using this RR
- Is a pointer to the next key in the chain of trust
  - You may trust data that is signed using a key that the DS points to

DS RDATA

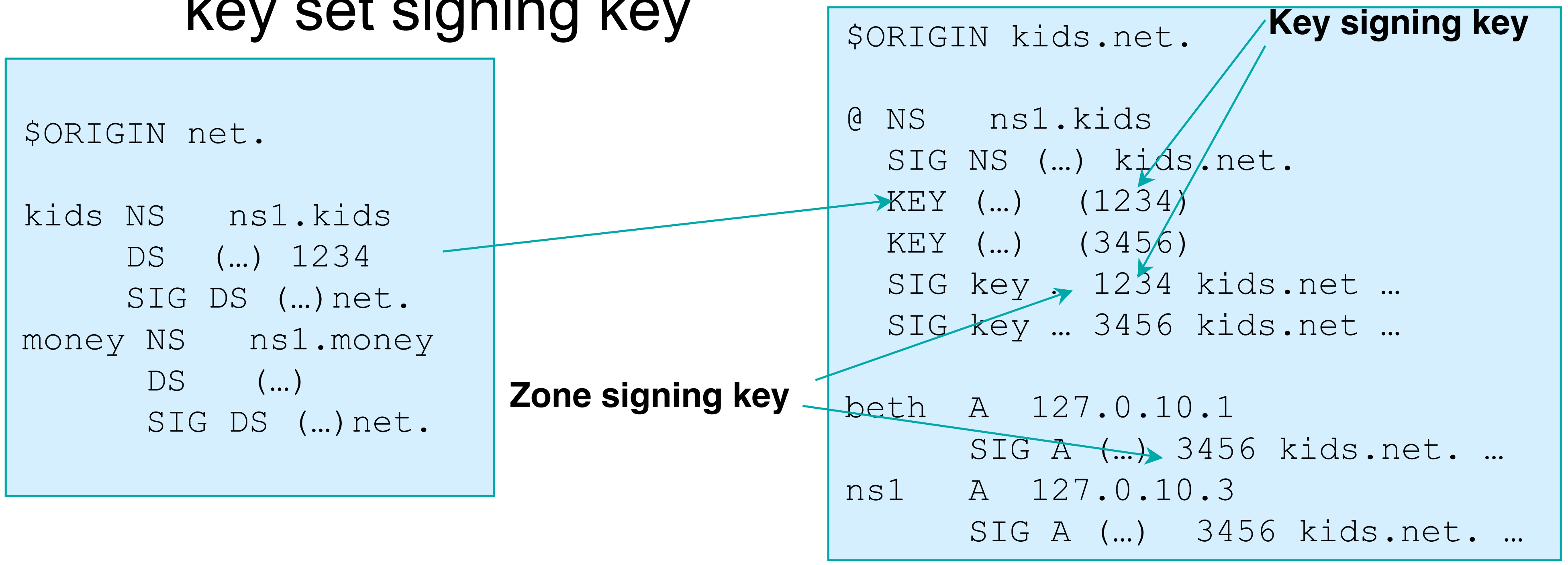
This field indicates which key is the next in the chain of trust

- 16 bit key tag
- 8 bit algorithm
- 20 bit SHA-1 Digest

```
$ORIGIN ripe.net.
disi.ripe.net     3600 IN   NS   ns.disi.ripe.net
disi.ripe.net.    3600 IN   DS   3112 1 (
                                 239af98b923c023371b52
                                 1g23b92da12f42162b1a9
                                 )
```

Delegating signing authority

- Parent signs the DS record pointing to the a key key set signing key



```
$ORIGIN net.

kids NS    ns1.kids
     DS   (…) 1234
     SIG DS (…)net.
money NS    ns1.money
      DS    (…)
      SIG DS (…)net.
```

```
$ORIGIN kids.net.

@ NS    ns1.kids
  SIG NS (…) kids.net.
  KEY (…)   (1234)
  KEY (…)   (3456)
  SIG key … 1234 kids.net …
  SIG key … 3456 kids.net …

beth  A  127.0.10.1
      SIG A (…) 3456 kids.net. …
ns1   A  127.0.10.3
      SIG A (…)  3456 kids.net. …
```

•The parent is authoritative for the DS RR.

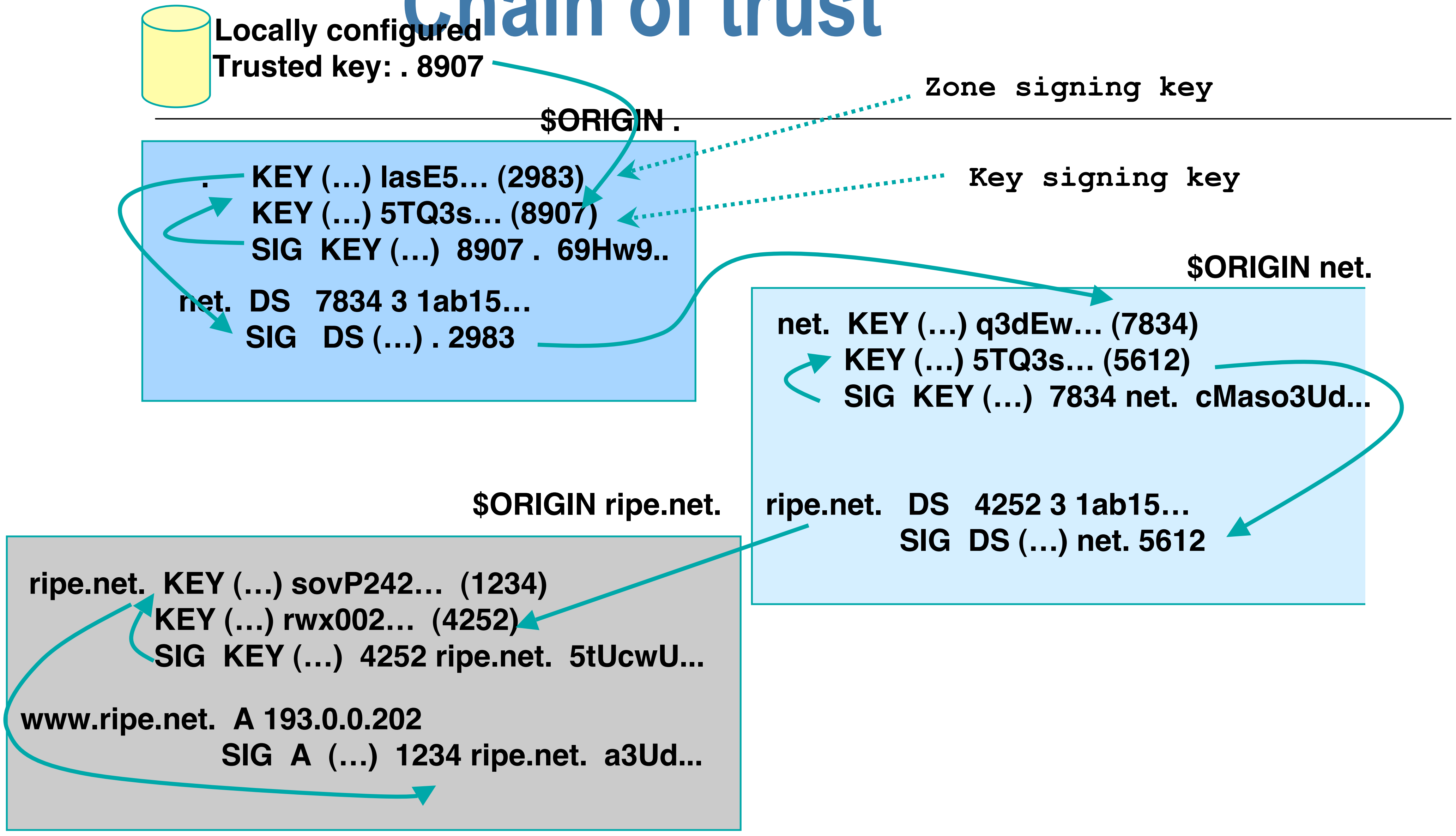
Jargon: key/zone signing keys

- DS points to a key signing key
- The zone is signed with a zone signing key
- These keys may be the same one
- Key signing key may be long lived
- Zone signing key may be short lived

Chain of trust

- Data in zone can be trusted if signed by a zone signing key
- Zone signing can be trusted if signed by a key signing key
- Key signing key can be trusted if pointed to by trusted DS record
- DS record can be trusted if signed by the parents zone signing key or
- DS record can be trusted if exchanged out of band (Trusted key)

Chain of trust



```
$ORIGIN .

.    KEY (…) lasE5… (2983)
     KEY (…) 5TQ3s… (8907)
     SIG  KEY (…)  8907 .  69Hw9..

net.  DS   7834 3 1ab15…
      SIG   DS (…) . 2983
```

```
$ORIGIN net.

net.  KEY (…) q3dEw… (7834)
      KEY (…) 5TQ3s… (5612)
      SIG  KEY (…)  7834 net.  cMaso3Ud...

ripe.net.   DS   4252 3 1ab15…
            SIG  DS (…) net. 5612
```

```
$ORIGIN ripe.net.

ripe.net.  KEY (…) sovP242…  (1234)
           KEY (…) rwx002…  (4252)
           SIG  KEY (…)  4252 ripe.net.  5tUcwU...

www.ripe.net.  A 193.0.0.202
               SIG  A  (…)  1234 ripe.net.  a3Ud...
```

Jargon

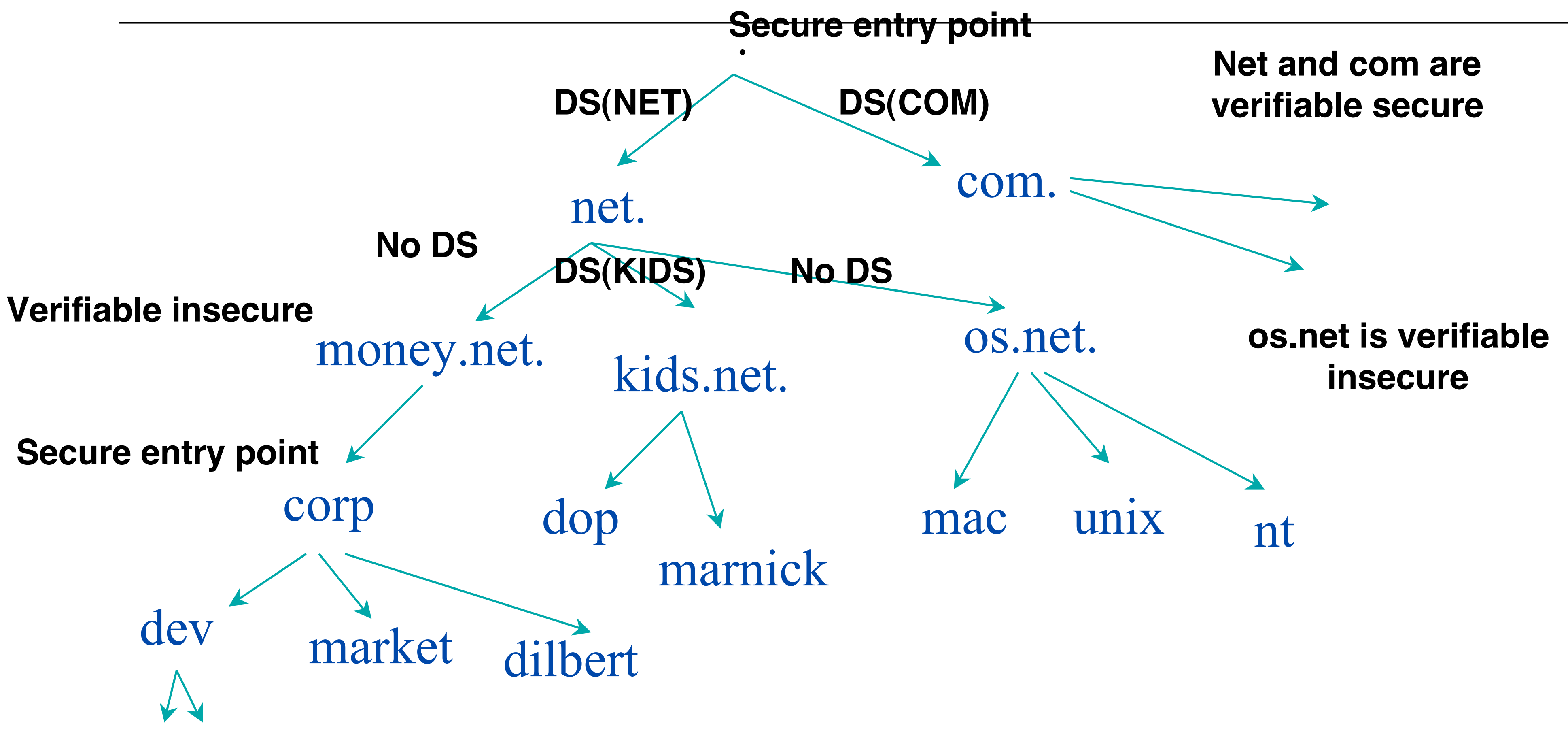
- Verifiable Secure
  - RRset and it’s SIG can be verified with a KEY that can be chased back to a trusted key, the parent has a DS record
- Verifiable Insecure
  - RRset sits in a zone that is not signed and for which the parent has no DS record
- BAD
  - RRset and its SIG can not be verified (somebody messed with the sig, the RRset, or the SIG expired)
  - A zone and it’s subzones are BAD when the parent’s SIG over the Child’s key is BAD

Jump to first page

Verifiably insecure zones

- Cryptographic evidence for the verifiably insecure zone status is given by parent
- If there is no DS record as proved by a NXT record with valid signature, the child is not secured
- In RFC2435 the parent has a “NULL” key with a signature
- A child may contain signatures but these will not be used when building a chain of trust

Jargon



Resolver has key of root and corp.money.net configured as secured entry points

Parental signature
adopting orphans carefully…

- Parents needs to check if the child KEY is really their child's… Did you get the KEY from the source authoritative for the child zone?
- This needs a out-of-DNS identification

Open operational issue:

- How do you identify the KEY comes from an authoritative source?
  - Billing information?
  - Phone call?
  - Secret token exchange via surface mail?

Jump to first page

The DNS is not a PKI

- All procedures on the previous slide are based on local policy i.e. policy set by the zone administrator
- A PKI is as strong as it’s weakest link, we do not know the strength of the weakest link
- If the domain is under one administrative control you might be able to enforce policy
- But it is closest to a globally secured distributed database
  - IPsec distribution of key material
    - opportunistic keys; if there is a key in the DNS and nothing better we’ll use it

OUTLINE

- Introduction
- **DNSSEC mechanisms**
  - DNSSEC's mechanisms to authenticate servers
  - **DNSSEC mechanisms to establish authenticity and integrity**
    - Quick overview
    - New RRs
    - DNSSEC signing of an isolated zone
    - Delegating signing authority following zone delegations or how to secure the DNS
    - **Key exchange and rollovers**
- Conclusions

Why key exchange and rollover?

- You have to keep your private key secret
- Private key can be stolen
  - Put the key on stand alone machines or on bastion hosts behind firewalls and strong access control
- Private key reconstruction (crypto analysis)
  - random number not random
  - Leakage of key material (DSA)
  - Brute force attacks
  - Slashdot headline: Prodigy discovers an analytical solution to NP problems

Why key exchange and rollover?

- Minimize impact of private key compromise
  - Short validity of signatures
  - Regular key-rollover

- Remember: KEYs do not have timestamps in them; The SIG over the KEY has the timestamp

Short Signature life time

- short parent signature over DS RR protects child
- Order 1 day possible

```
www.ripe.net.   3600 IN  SIG   A 1 3 3600 20010504144523 (
            20010404144523 3112 ripe.net.
            VJ+8ijXvbrTLeoAiEk/qMrdudRnYZM1VlqhN
            vhYuAcYKe2X/jqYfMfjfSUrmhPo+0/GOZjW
            66DJubZPmNSYXw== )
```

Signature expiration

Scheduled Key rollover

- Child starts using a new key and wants parent to sign it

```
$ORIGIN net.

kids NS    ns1.kids
     DS   (…) 2
     SIG KEY (…)net.
money NS   ns1.money
      DS    (…) 1
      SIG KEY (…)net.
```

```
$ORIGIN kids.net.

@ NS    ns1.kids
  KEY (…) 2
  KEY (…) 5
  SIG KEY (…) kids.net. 2

ns1    A  127.0.1.10.3
       SIG A (…) kids.net. 2
```

Create key 2 and sign keyset with key 1 and 2.
Send key 2 to parent, parent signs DS record.
Sign with key 2 only once parent updated.

Jump to first page

Unscheduled rollover problems

- Needs out of band communication with the parent and to pre-configured resolvers
- The parent needs to establish your identity out of band again
- Your children need protection How to protect them best? Leaving them unsecured?
- There will be a period that the stolen key can be used to generate data useful on the Internet
- There is no ‘revoke key’ mechanism
- Emergency procedure must be on the shelf

OUTLINE

- Introduction
- **DNSSEC mechanisms**
- ➡ Conclusions

Open issues
(the where-shall-I-put-it slide)

DNSSEC is still a moving target…

- RFC 2535 rewrite
- KEY restrict/APPKEY
- NXT/OPT-IN
- Delegation Signer (DS)
- BIND development
- Operational issues

What did we learn

- DNSSEC provides a mechanism to protect DNS
- DNSSEC implementation:
  - TSIG for servers
  - SIG, KEY and NXT for data
- DNSSEC main difficulties:
  - keeping private key safe
  - distributing keys

End of Part I...
Questions???

- NLnet labs maintains a list of DNSSEC resources http://www.nlnetlabs.nl/dnssec/

PART II

DNSSEC Operations

Description of tools

Outline Part II - DNSSEC Operations

- Configuration
- Securing host-host communication
- Securing zones
- Building a secure tree
- Miscellaneous

Setup of the OS

- When using DNSSEC (Transaction signatures or Data verification) the time needs to be in sync
- Use:
  - ntpdate -b
  - xntpd
- Common error: time zone definition wrong

- You will also need openssl libraries

Getting and compiling the latest version of bind9

- Obtain the source and compile it
  - `ftp://ftp.isc.org/isc/bind9/9.2.0/bind-9.2.0.tar.gz`
  - Alternatively 9.3 snapshot from `ftp://ftp.isc.org/isc/bind9/snapshots/`
  - Compile the source using openssl libraries: `./configure --prefix=/usr/local --with-openssl`
- (We assume: `--prefix=/usr/local`)

Server/Named configuration

- The configuration file lives in `/usr/local/etc/named.conf`
- Documentation in <src>/doc/arm/Bv9ARM.html
- Turn on logging
  - Several categories
  - Categories are processed in one or more channels
  - Channels specify where the output goes

Logging example

```
Logging {
       channel update_debug_channel {
                   file “log/update_debug.log”;
                   severity debug 5;
                   };
      channel update_info_channel {
                   file “log/update_info.log”;
                   severity info;
                   };

      category update { update_debug_channel;
                        update_info_channel;};
};
```

Logging Categories

Only those relevant to DNSSEC

- dnssec
  - Processing DNSSEC signed responses
- security
  - Request that are approved or not
- notify
  - Zone change notification (relevant for dynamic update environments)
- update
  - Dynamic update events

Toolbag
The complete set

- NAMED
- DNSSEC tools:
  - dnssec-keygen Generate keys of various types
  - dnssec-signzone Sign a zone
  - dnssec-makekeyset Generate a keyset from a key
  - dnssec-signkey Sign a keyset

  - dig +dnssec Use dig to troubleshoot (or host, nslookup not supported)

  - named-checkzone & named-checkconf

Toolbag: dig For trouble shooting

- **`dig`** is your friend, learn to use it!
  - nslookup will not be supported, host will
  - dig is low level *i.e.* not user friendly
- All pieces of information are relevant
  - Status, flags, answer section, authority section, additional section
  - Not all servers are BIND servers and implementations may be broken

Jump to first page

Dig example

dig bert.secret-wg.org

```
; <<>> DiG 9.3.0s20020122 <<>> bert.secret-wg.org
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40334
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2,
   ADDITIONAL: 2

;; QUESTION SECTION:
;bert.secret-wg.org.            IN      A

;; ANSWER SECTION:
bert.secret-wg.org.     36293   IN      A       193.0.0.4

;; AUTHORITY SECTION:
secret-wg.org.          170827  IN      NS      bert.secret-wg.org.
secret-wg.org.          170827  IN      NS      NS2.secret-wg.org.

;; ADDITIONAL SECTION:
bert.secret-wg.org.     36293   IN      A       193.0.0.4
NS2.secret-wg.org.      170827  IN      A       193.0.0.202

;; Query time: 89 msec
;; SERVER: 193.0.1.96#53(193.0.1.96)
;; WHEN: Wed Feb 13 11:08:36 2002
;; MSG SIZE  rcvd: 116
```

Jump to first page

Outline Part II - DNSSEC Operations

- Configuration
- Securing host-host communication
- Securing zones
- Building a secure tree
- Miscellaneous

TSIG configuration Outline

- Generate a key
- Configuring secure transfers between servers with TSIG
- Testing
- Other types of host-host communication

TSIG Toolbag: dnssec-keygen

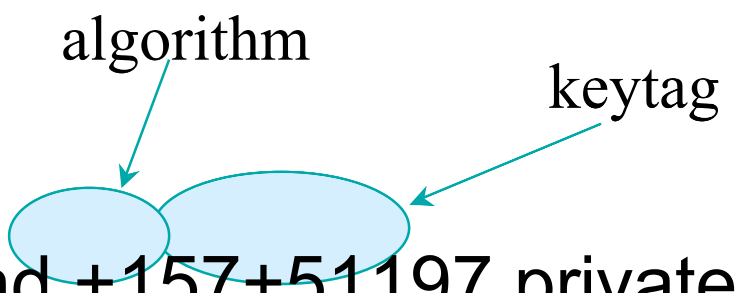
- **Use dnssec-keygen to Generate TSIG keys**

```
Usage:
   dnssec-keygen -a alg -b bits -n type [options] name
```

- Use HMAC-MD5 as algorithm
- type is host
- Bitsize: 256 or larger
- Name: unique indintifyer
  - Suggested: **`host-host.domain.foo`**
  - We use: **`me-friend`** because of formatting constraints

TSIG Toolbag: dnssec-keygen output

```
dnssec-keygen -a HMAC-MD5 -b 256 -n host me-friend
```



- Kme-friend.+157+51197.private
- Kme-friend.+157+51197.key
- Private and Public Key content both the same
- TSIG should never be put in zone files!!!

(This might be confusing because of the content of Kme-friend+157+51197.key)

```
me-friend. IN KEY 512 3 157 nEfRX9…bbPn7lyQtE=
```

TSIG configuration steps 1

- Create key using DNSSEC-keygen:

```
dnssec-keygen -a HMAC-MD5 -b 256 -n HOST me-
  friend
Kme-friend.+157+51197
```

- Cut-n-paste key material into named.conf

```
key ”me-friend." {
        algorithm hmac-md5;
        secret
  “nEfRX9jxOmzsby8VKRgDWEJorhyNbjt1ebbPn7lyQtE=
  ";
};
```

- Communicate this with your partner (off band, PGP…)

TSIG configuration step 2

- Configure your server to require the key for zone transfers
  - Use the key statement to configure the key
  - Use the allow-transfer statement in the zone statement to indicate which keys are allowed transfer

```
zone ”ripe.net" {
        type master;
        file "zones/ripe.net.";
        allow-transfer { key me-friend ; };
        notify yes;
};
```

TSIG configuration step 3

- Have your partners configure their servers to use the key when talking to you
  - Use the key statement to configure the key
  - Use the server statement to indicate which key is needed for communication with that server

```
server 192.168.10.1 {
        keys {me-friend; };
};
zone ”ripe.net" {
        type slave;
        masters { 192.168.10.1;};
        file “slaves/ripe.net”;
};
```

Jump to first page

TSIG Troubleshooting: dig

- You can use dig to check TSIG configuration
- `dig  @<server> <zone> AXFR -y name:key`

```
$ dig @193.0.0.202 ripe.net AXFR \
  -y me-friend:nE1Gw6fpW...tE=
```

- Wrong key will give you “Transfer failed” and on the server the security-category will log:

```
security: error: client 193.0.0.182#1228: zone transfer 'ripe.net/IN' denied
```

Using TSIG to protect dynamic updates

- You can use TSIG or SIG0 to protect your dynamic updates
  - Detailed howto at: Secure dynamic update HOWTO on ops.ietf.org
- Steps for TSIG dynamic update of forward tree:
  - Configure your TSIG key into /etc/dhclient.conf and specify the FQDN
  - Configure named.conf to allow updates using the key

Outline Part II - DNSSEC Operations

- Configuration
- Securing host-host communication
- Securing zones ←
- Building a secure tree
- Miscellaneous

Setting up a secure zone Outline

We now focus on setting up a secure island e.g. for use in a corporate environment

- Resolver issues
- Generating key
- Signing the zone

Resolving in a secured DNS environment

A few remarks

- DNSSEC is not in POSIX yet (e.g. `gethostbyname()`)
- SIG verification is (only) done by caching forwarders
- To test DNSSEC setups, you have to work with dig, or use the BIND lwresolver library
- Alternatively: write some tools in PERL (Net::DNS with security extensions)

Setting up a verifying caching forwarder

- You want to verify the content of a zone:
  - Get the public key and check, out of band, that his key belongs to the zone owner
  - Configure the key in your forwarder
- We will configure the key of secret-wg.org in our verifying forwarder

Configuring verifying forwarders

- In the forwarder you configure the keys you trust as a secure entrypoint

```
 trusted-keys {
                 “.” 256 3 1 ”Abc12..zZ";
                 “SECRET-WG.ORG” 256 3 1 “AQ…QQ=="
      };
```

- Sys-admins of resolvers should verify authenticity of **`trusted-keys`** before putting them in zone files

Testing a verifying forwarder dig

`dig +dnssec [@server] record [TYPE]`

- Answer Flags are relevant
- Example query to a authoritative nameserver

```
; <<>> DiG 9.1.1 <<>> +dnssec @193.0.0.202
  www.ripencc.dnssec.nl.nl
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1947
;; flags: qr aa rd; QUERY: 1, ANSWER: 4, AUTHORITY: 3,
  ADDITIONAL: 4
```

Recursion desired (but not available, RA is not set)

authoritative answer

Testing a verifying forwarder dig: an example

```
; <<>> DiG 9.3.0s20020122 <<>> +dnssec @127.0.0.1 secret-wg.org
  NS
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31630
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 3, AUTHORITY: 0,
  ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version:    0, udp=   4096
;; QUESTION SECTION:
;secret-wg.org.         IN NS

;; ANSWER SECTION:
secret-wg.org.        600 IN NS ns2.secret-wg.org.
secret-wg.org.        600 IN NS bert.secret-wg.org.
secret-wg.org.        600 IN SIG NS 1 2 600 20020314134313
                         20020212134313 47783 secret-wg.org.
      DVC/ACejHtZylifpS6VSSqLa15xPH6p33HHmr3hC7eE6/QodM6fBi5z3
      fsLhbQuuJ3pCEdi2bu+A0duuQlQMiHPvrkYia4bKmoyyvWHwB3jcyFhW
      lV4YOzX/fgkLUmu8ysGOiD9C0CkSvNSE6rBCzUa3hfkksHt4FBsuA1oQ
      yoc=
```

Jump to first page

Troubleshooting client side

- Dig returns status: SERVFAIL
- First try without **`+dnssec`**
- Also try with **`+dnssec +cdflag`**
  - Checking is disabled. Data directly forwarded
- Be ready for some nice trouble shooting

Troubleshooting Server side

- Turn on logging. Category “dnssec” with severity debug 3 gives you appropriate hints

- Debug output is a little detailed
  - On the next page is an example where we corrupted the trusted-key
  - It is not directly obvious what the problem is
  - We edited the output a little so that it fits on a slide

Example debugging output

```
validating secret-wg.org KEY: starting
validating secret-wg.org KEY: attempting positive response validation
validating secret-wg.org KEY: keyset was self-signed but not
                              preconfigured
validating secret-wg.org KEY: no valid signature found
validating secret-wg.org KEY: falling back to insecurity proof
validating secret-wg.org KEY: insecurity proof failed
validator @0x81e6900: dns_validator_destroy
validating secret-wg.org NS: in fetch_callback_validator
validating secret-wg.org NS: fetch_callback_validator: got no valid SIG
validator @0x81e1d00: dns_validator_destroy
```

Setting up a secure zone Outline

- Resolver issues
- Generating key ←
- Signing the zone

Toolbag: dnssec-keygen

- **Use dnssec-keygen to Generate zone keys**

```
Usage:
   dnssec-keygen -a alg -b bits -n type [options] name
```

- Use RSA/SHA1 as algorithm
- type is zone
- Bitsize: depends...
- Name: the name of the zone you want to sign

Toolbag: dnssec-signzone

```
Usage:
      dnssec-signzone [options] zonefile [keys]
```

- If the name of your zonefile is not the name of the zone then use the –o <origin> option
- You might need the '-r /dev/urandom' option on your OS

Signing a Zone 1
Creating the KEY

- You need to regenerate keys for a zone periodically (order months, years)
- `dnssec-keygen -a RSA -b 1024 -n zone secret-wg.org`

  `Ksecret-wg.org.+001+20704`
- `Ksecret-wg.org.+001+20704.key` contains the public key. It may be “cut ‘n pasted” into your zonefile
- **`Ksecret-wg.org.+001+20704.private` should be kept secret!**

Setting up a secure zone
Outline

- Resolver issues
- Generating key
- Signing the zone

Signing a Zone 2
Signing the Zone

- Include the key RR for the apex:

  -
    ```
    cat Ksecret-wg.org.+001+20704.key >>\
        secret-wg.org.
    ```
  - You may want to change the TTL field.

- Increase the SOA serial number
  - **Always increase the SOA serial before signing!**

- Sign the zone:

```
dnssec-signzone secret-wg.org. \
      Ksecret-wg.org.+001+20704
```

Signing a Zone 2 Publishing Zone

- Publish the zone:

```
zone ”secret-wg.org" {
        type master;
        file "zones/secret-wg.org.signed";
        allow-transfer { 10.1.2.3 ;
                         key mstr-slave.secret-wg.org.; };
        notify yes;    // Don’t forget to increase serial
  before signing
};
```

- RNDC reload and test.

Notes on secured zones

- Only those records for which the server is authoritative for are signed
  - NS records in the APEX are signed
  - Delegating NS records and GLUE are not signed

For 2535 signer:

- Zone is sorted and NXT records are introduced
- NULL keys are introduced at each delegation
- 4 extra RRs for each name in the zone

Outline Part II - DNSSEC Operations

- Configuration
- Securing host-host communication
- Securing zones
- Building a secure tree
- Miscellaneous

Task 3
Parent-Child interaction

- With secured islands there is a problem with key distribution

- Use the DNS itself to distribute keys; once authenticity is established for one key you can use that key to establish authenticity of other keys
- In an ideal world:
  - You would only configure one key (the root key)
  - Delegate trust from parent to child

Intermezzo Delegating zones and key distribution

- Delegation of Zones and Key distribution are closely linked
- The problems occurring in this area are delaying deployment
- In this intermezzo we discuss the issues

DNS: Delegation and glue

- NS records are used for delegation
- NS records are in the apex of the zone and appear in both the ‘parent’ and ‘child’ zones
  - Parent is not authoritative (answer is a referral)
  - Child is authoritative
- Glue is needed if nameserver’s names in the zone which is being delegated
- dnssec-signzone only generates SIGs over RRs for which the zone is authoritative

Delegated Zone security

- Security is delegated at zone delegation points
  - public key cryptography
- org. delegates zone signing authority to secret-wg.org.

Delegation Points

According to RFC 2535

- Parent signs the key of the child
- Data at the parent is less authoritative than child
  - NS and Glue records are not signed at the parent
- Only when the child is unsecured, a null key must appear in the parent

Consequence: Difficult key exchange procedure

- Child key must be uploaded to parent
- Verified, signed and returned to child
- This is time consuming, error prone and can lead to unsigned delegations

RFC 2535 example

- Child sends self-signed key to parent
  - use dnssec-makekeyset
- Parent signs the keyset
  - use dnssec-signkey
- Child publishes the data from the signed key set in it’s zone

```
secret-wg.org.      IN  KEY  256 3 1 AB123…RDf
                    IN  SIG  1  2 3600 2002… 2002… 14815 org. Usdf…/s21
```

Parental signature

DS Record

- Parent is authoritative for the DS record
  - It may not appear in the child's apex
- Simplifies KEY exchange
- Eases resigning
  - parent can sign often ➡short signature lifetime ➡ shorter impact time when key gets compromised

Delegation of authority Near future…

- Expect the DS record for delegation of security
- No tools yet so you cannot play with this
- You may want to experiment with RFC2535 style delegation of authority
  - Out of this tutorial's scope
- DS will be backwards incompatible with 2535

Outline Part II - DNSSEC Operations

- Configuration
- Securing host-host communication
- Securing zones
- Building a secure tree
- Miscellaneous (and conclusions) ←

Key exchange and Key rollover

- Upload your key to parent (first key exchange)
  - procedure is registry dependent
- Key rollover Task
  - Generate a new key
  - Publish new key in your zone file and sign with old and new key
  - Don’t forget to inform those resolvers that need you as a secure island (**`trusted-keys`** configuration)
  - Trigger the registry (push or pull)
  - Check availability of SIG over new DS record at parent
  - Remove old key

Back at the ranch

- Design a secure architecture
- Design a key exchange procedure
- Resign your zone regularly
- Automate the process (cron and Makefiles)
- Have an emergency procedure in place

Feedback

- Give feedback on DNSSEC operations
  - Which tools would make your life easier
  - Why are you (un)successful with deployment
  - What kind of information would ease your tasks

Jump to first page

RNDC and TSIG

bill manning

bmanning@ep.net

What is RNDC?

- Remote Name Daemon Controller
- Command-line control of named daemon
- Usually on same host, can be across hosts

Configuring RNDC

- "rndc-conf" generates lines to be added to two files
  - named.conf
  - rndc.conf

Enabling RNDC in the server

- key definition

```
key rndc_key {
  secret "dY7/uIiR0fKGvi5z50+Q=="; algorithm
  hmac-md5;
};
```

  - Warning: example secret looks good but is invalid (don't copy it!)

- controls statement

```
controls {
  inet 127.0.0.1 port 953
       allow { 127.0.0.1; }
       keys { "rndc-key"; };
 };
```

Using an rndc.conf file

- /etc/rndc.conf specifies defaults for rndc
- E.g.,

```
key "rndc-key" {
        algorithm hmac-md5;
        secret "dY7/uIiR0fKGvi5z50+Q==";
};

options {
        default-key "rndc-key";
        default-server 127.0.0.1;
        default-port 953;
};
```

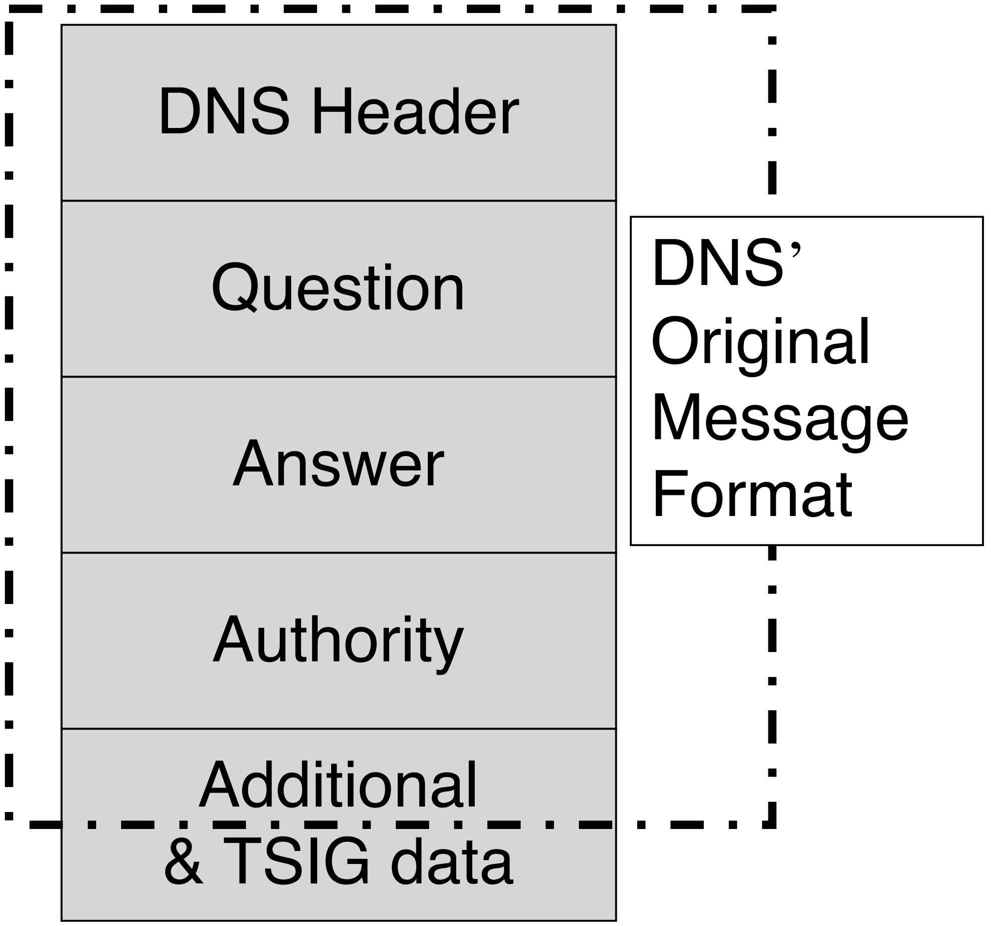
What can be done with RNDC

- rndc stop - kills server
- rndc status - prints some information
- rndc stats - generates stat file (named.stats)
- rndc reload - refresh zone(s), w/variations
- rndc trace - increases debug level
- rndc flush - removes cached data
- other commands in the ARM

What is TSIG?

- A mechanism for protecting a message from a resolver to server and vice versa
- A keyed-hash is applied (like a digital signature) so recipient can verify message
- Based on a shared secret - both sender and reciever are configured with it

TSIG and Message Format



Names and Secrets

- TSIG name
  - A name is given to the key, the name is what is transmitted in the message (so receiver knows what key the sender used)
- TSIG secret value
  - A value determined during key generation
  - Usually seen in Base64 encoding
- 'Looks' like the rndc key
  - BIND uses same interface for TSIG and RNDC keys

Using TSIG to protect AXFR

- Deriving a secret
  - `dnssec-keygen -a ... -b ... -n... name`
- Configuring the key
  - in named.conf file, same syntax as for rndc
  - `key { algorithm ...; secret ...;}`
- Making use of the key
  - in named.conf file
  - `server x { key ...; }`
  - where 'x' is an IP number of the other server

Configuration Example

- Primary server
  - 10.33.40.46

```
key ns1-ns2.zone. {
   algorithm hmac-md5;
   secret "APlaceToBe";
};
server 10.33.40.35 {
   keys {ns1-ns2.zone.;};
};
zone "my.zone.test." {
   type master;
   file...;
   allow-transfer {
        key ns1-ns2.zone.;
        key ns1-ns3.zone.;};
};
```

- Secondary server
  - 10.33.40.35

```
key ns1-ns2.zone. {
   algorithm hmac-md5;
   secret "APlaceToBe";
};
server 10.33.40.46 {
  keys {ns1-ns2.zone.;};
};
zone "my.zone.test." {
   type slave;
   file...;
   masters {10.33.40.46;};
   allow-transfer {
        key ns1-ns2.zone.;};
};
```

Again, the secret looks okay, but is purposely invalid

TIME!!!

- TSIG is time sensitive - to stop replays
  - Message protection expires in 5 minutes
  - Make sure time is synchronized
  - For testing, set the time
  - In operations, (secure) NTP is needed

Other uses of TSIG

- TSIG was designed for other purposes
  - Protecting sensitive stub resolvers
    - This has proven hard to accomplish
  - Dynamic Update
    - Discussed later, securing this relies on TSIG

Alternatives to TSIG

- SIG (0)
  - Public key approach to same services
  - Has potential, but not much experience yet
- TKEY
  - Means to start with SIG(0) and wind up with TSIG
  - Also, Microsoft uses this with Kerberos via GSSAPI

Questions