APNIC DNS/DNSSEC Workshop

Contact: training@apnic.net

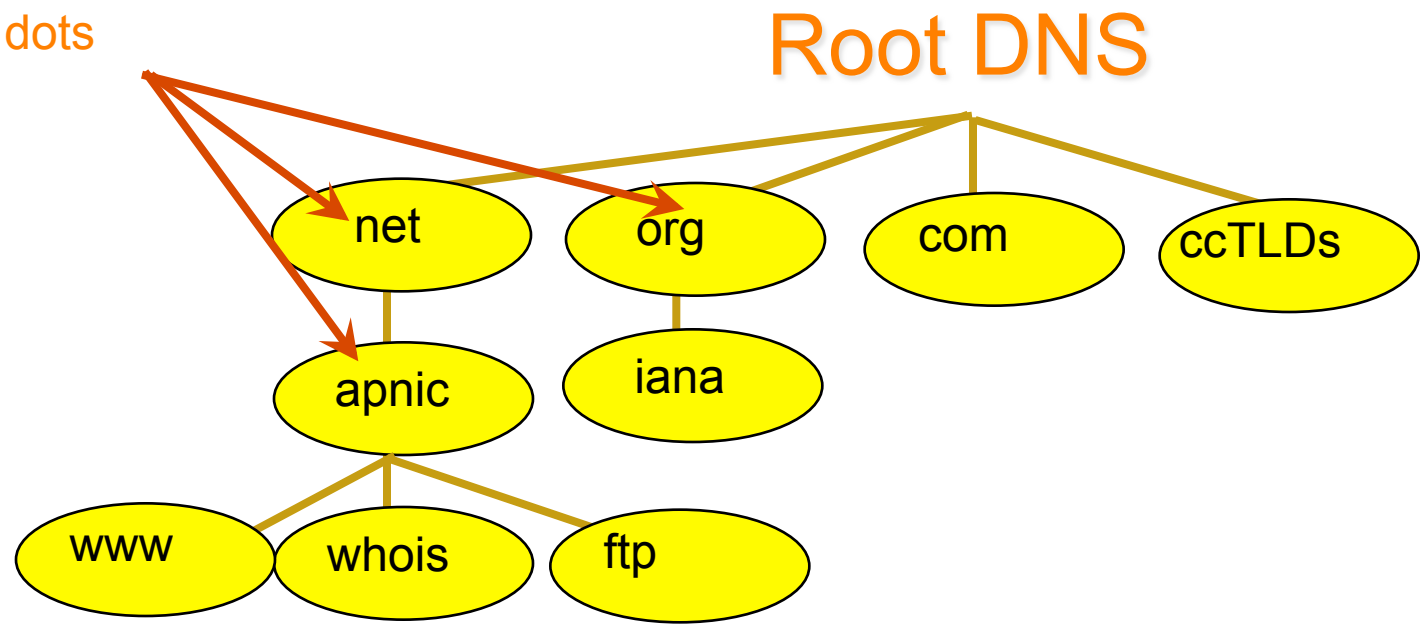
DNS

- 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
- Loose Coherency
- Scalability
- Reliability
- Dynamicity

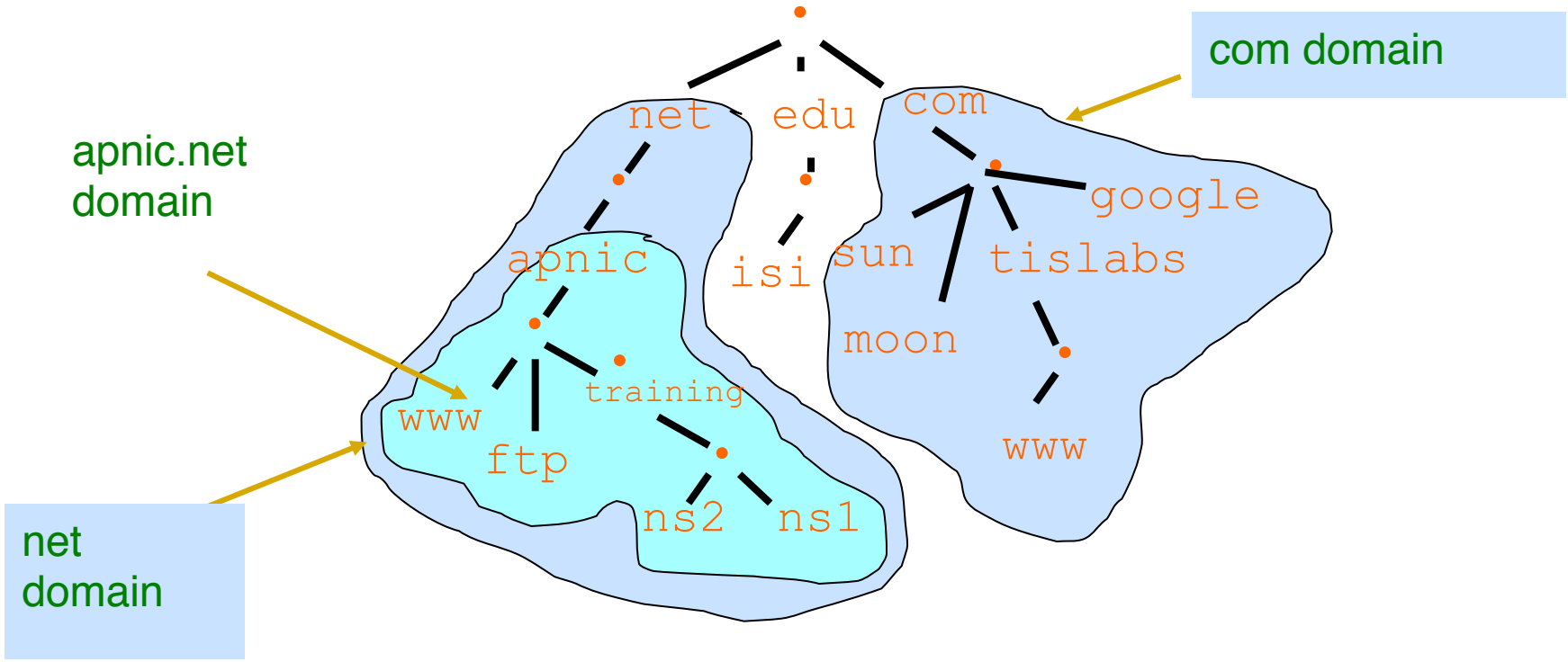
DNS Names - FQDNs



Domains

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

Domains


com domain
apnic.net
domain
net
edu
com
google
apnic
isi
sun
tislabs
moon
training
www
ftp
www
ns2
ns1
net
domain

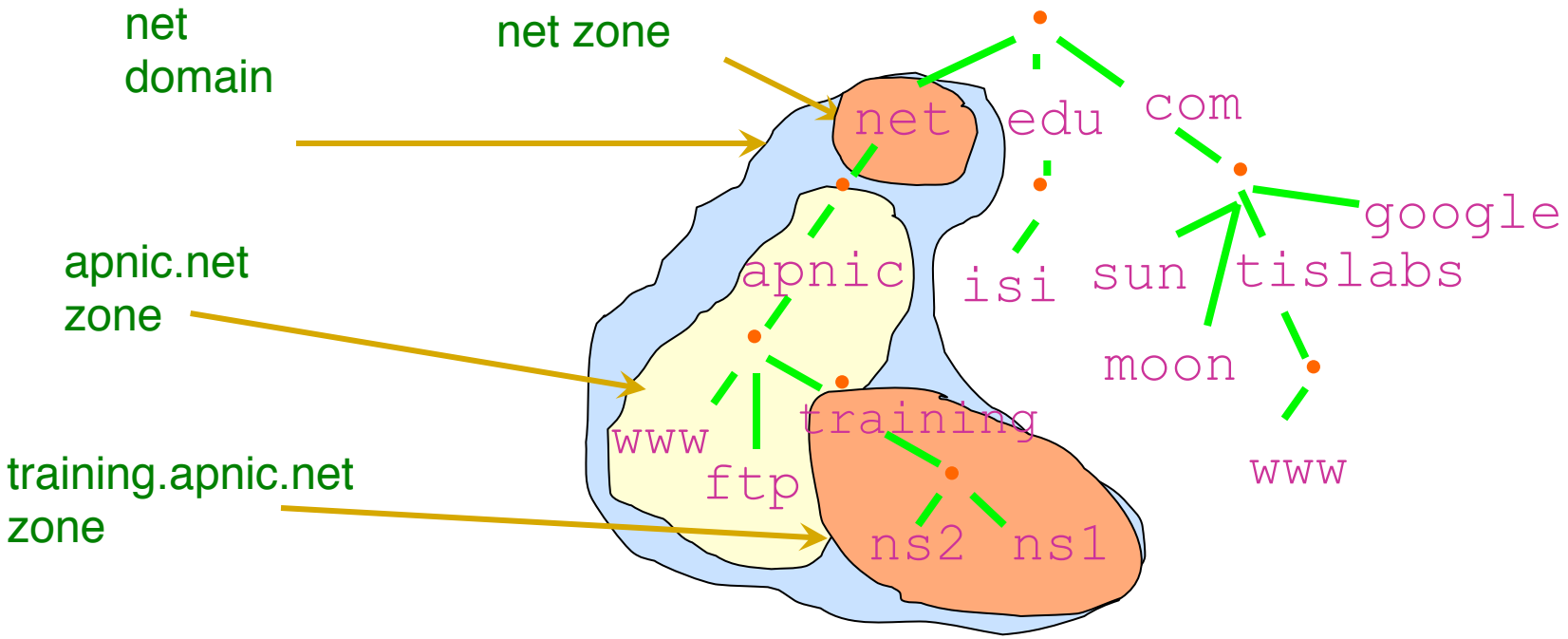
Delegation

- 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

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

Zones and Delegations

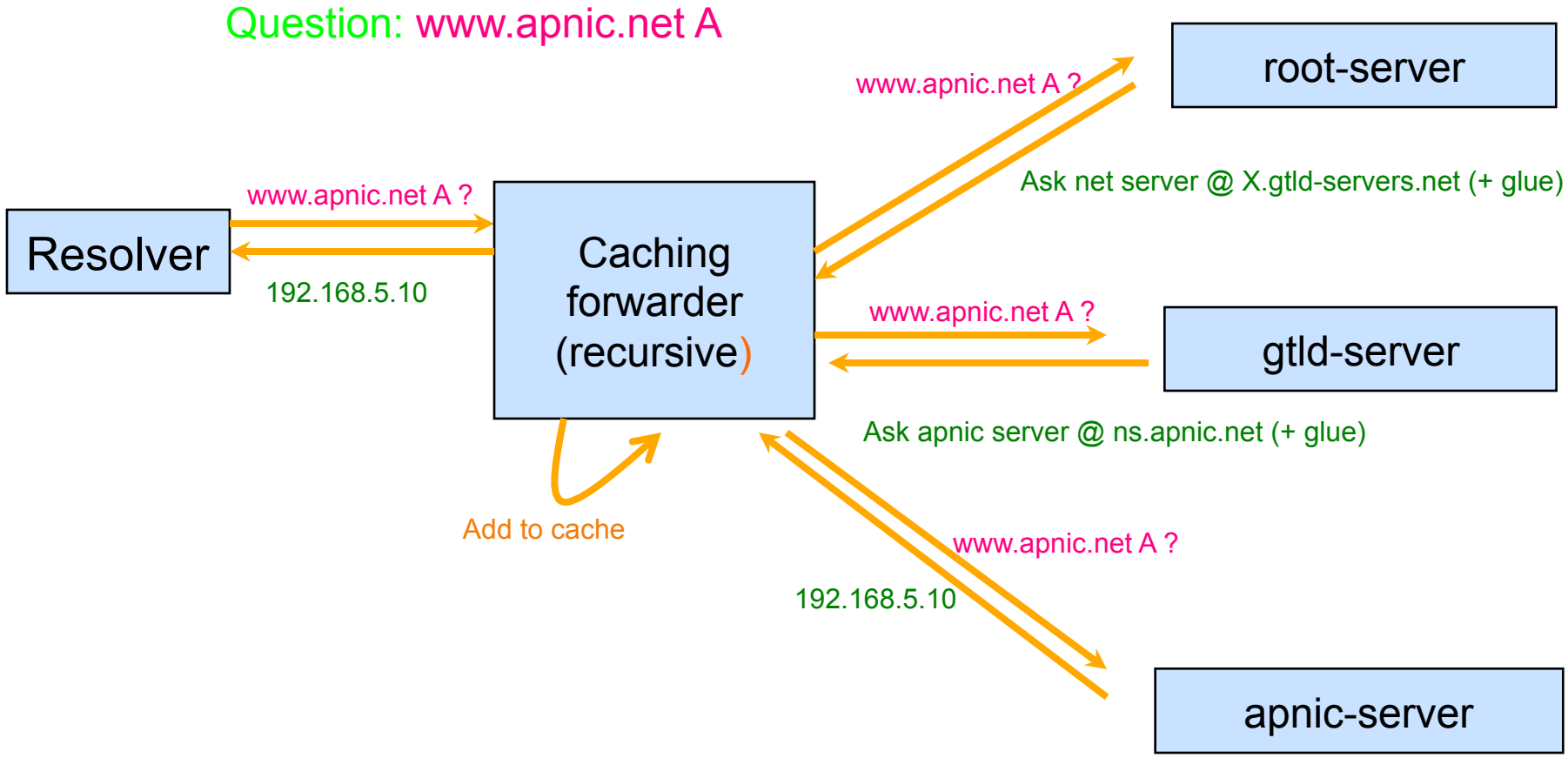


Name Servers

- Name servers answer ‘DNS’ questions

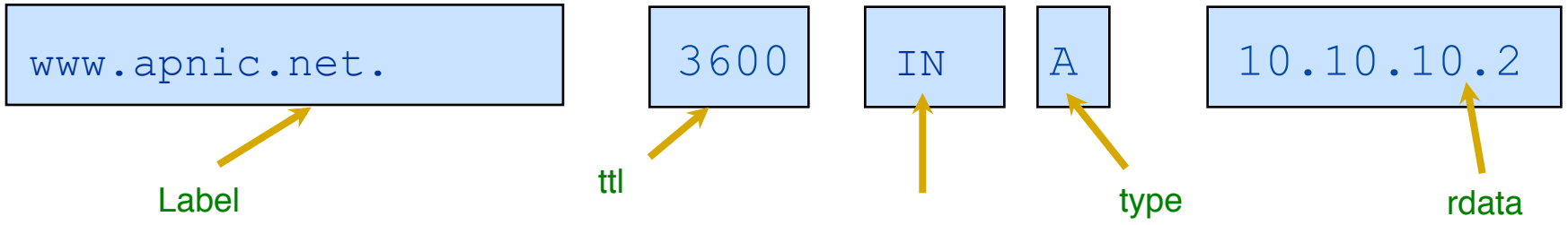
- Several types of name servers
  - Authoritative servers
    - master (primary)
    - slave (secondary)
  - (Caching) recursive servers
    - also caching forwarders
  - Mixture of functionality

Concept: Resolving process & Cache



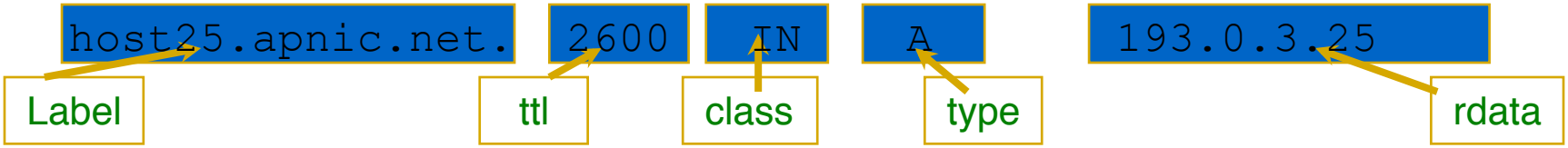
Concept: Resource Records

- 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



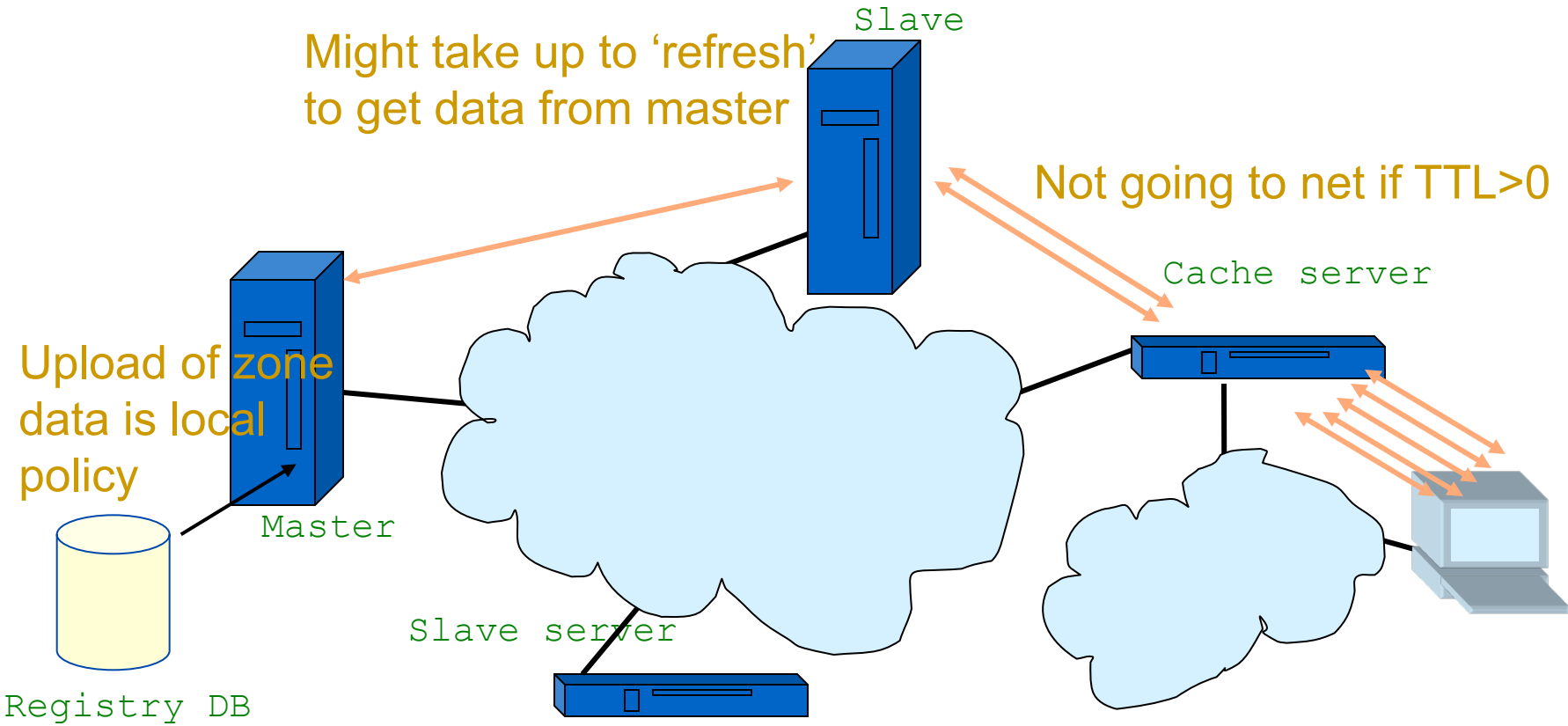
Example: RRs in a zone file

```
apnic.net. 7200 IN      SOA     ns.apnic.net. admin.apnic.net.
                        (
                            2009012001      ; Serial
                            12h   ; Refresh 12 hours
                            4h   ; Retry 4 hours
                            4d ; Expire 4 days
                            2h ; Negative cache 2 hours )

apnic.net.              7200   IN     NS       ns.apnic.net.
apnic.net.              7200   IN     NS       ns.ripe.net.

host25.apnic.net.       2600   IN     A        193.0.3.25
whois.apnic.net.        3600   IN     A        193.0.1.162
```



Places where DNS data lives

- Changes 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

Performance of DNS

- Server hardware requirements
- OS and the DNS server running
- How many DNS servers?
- How many zones expected to load?
- How large the zones are?
- Zone transfers
- Where the DNS servers are located?
- Bandwidth

Performance of DNS

- Are these servers Multihomed?
- How many interfaces are to be enabled for listening?
- How many queries are expected to receive?
- Recursion
- Dynamic updates?
- DNS notifications

Zone files

```
apnic.net.          3600  IN SOA NS1.apnic.net. admin
\.email.apnic.net. (
                                2002021301    ; serial
                                1h            ; refresh
                                30M           ; retry
                                1W            ; expiry
                                3600 ) ; neg. answ. Ttl

apnic.net.              3600 IN NS   NS1.apnic.net.
apnic.net.              3600 IN NS   NS2.apnic.net.
apnic.net.              3600 IN MX   50   mail.apnic.net.
apnic.net.              3600 IN MX   150  mailhost2.apnic.net.

apnic.net.              3600 IN TXT  "Demonstration and test zone"
NS1.apnic.net. 4500 IN A    203.0.0.4
NS2.apnic.net. 3600 IN A    193.0.0.202
localhost.apnic.net.   3600 IN A    127.0.0.1
www.apnic.net.          3600 IN CNAME IN.apnic.net.
```

Zone files

```
apnic.net.          3600  IN SOA NS1.apnic.net. admin
\.email.apnic.net. (
                                2002021301      ; serial
                                1h              ; refresh
                                30M             ; retry
                                1W              ; expiry
                                3600 )          ; neg. answ. Ttl
                        3600 IN NS   NS1.apnic.net.
                        3600 IN NS   NS2.apnic.net.
                        3600 IN MX   50   mail.apnic.net.
                        3600 IN MX   150  mailhost2.apnic.net.

                        3600 IN TXT  "Demonstration and test zone"
NS1.apnic.net. 3600 IN A    203.0.0.4
NS2.apnic.net. 3600 IN A    193.0.0.202

localhost.apnic.net.    4500 IN A    127.0.0.1

www.apnic.net.          3600 IN CNAME IN.apnic.net.
```

Zone files

```
$TTL    3600 ; Default TTL directive
apnic.net.       IN SOA NS1.apnic.net. admin\.email.apnic.net. (
                                2002021301     ; serial
                                1h             ; refresh
                                30M            ; retry
                                1W             ; expiry
                                3600 ) ; neg. answ. Ttl
                          IN NS   NS1.apnic.net.
                          IN NS   NS2.apnic.net.
                          IN MX   50  mail.apnic.net.
                          IN MX   150 mailhost2.apnic.net.

                          IN TXT  "Demonstration and test zone"
NS1.apnic.net.  IN A    203.0.0.4
NS2.apnic.net.  IN A    193.0.0.202

localhost.apnic.net. 4500 IN A        127.0.0.1

www.apnic.net.           IN CNAME NS1.apnic.net.
```

Zone files

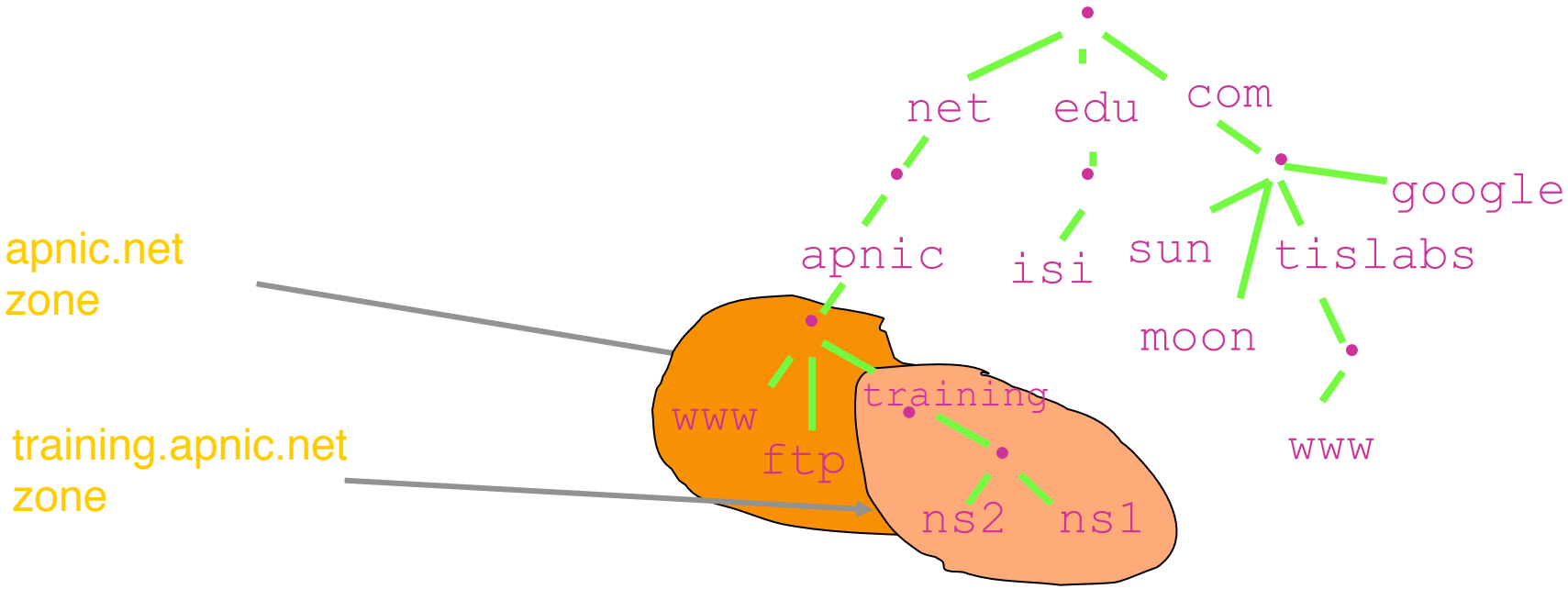
```
$TTL    3600 ; Default TTL directive
$ORIGIN apnic.net.
@               IN SOA NS1 admin\.email.apnic.net. (
                                2002021301    ; serial
                                1h            ; refresh
                                30M           ; retry
                                1W            ; expiry
                                3600 ) ; neg. answ. Ttl
                IN NS   NS1
                IN NS   NS2
                IN MX   50  mailhost
                IN MX   150 mailhost2


                IN TXT  "Demonstration and test zone"
NS1             IN A    203.0.0.4
NS2             IN A    193.0.0.202

localhost 4500 IN A     127.0.0.1

www             IN CNAME NS1
```

Zone files

```
$TTL    3600 ; Default TTL directive
$ORIGIN apnic.net.
@               SOA NS1 admin\.email.sanog.org. (
                                2002021301    ; serial
                                1h            ; refresh
                                30M           ; retry
                                1W            ; expiry
                                3600 ) ; neg. answ. Ttl
                NS    NS1
                NS    NS2
                MX    50  mailhost
                MX    150 mailhost2


                 TXT   "Demonstration and test zone"
NS1              A     203.0.0.4
NS2              A     193.0.0.202

localhost 4500   A     127.0.0.1
www              CNAME NS1
```

Delegating a zone (becoming a parent)

- Delegate authority for a sub domain to another party (splitting of *training.apnic.net* from *apnic.net*)



Glue

- Delegation is done by adding NS records:

```
training.apnic.net.      NS ns1.training.apnic.net.
training.apnic.net.      NS ns2.training.apnic.net.
training.apnic.net.      NS ns1.apnic.net.
training.apnic.net.      NS ns2.apnic.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.training.apnic.net. A 10.0.0.1
ns2.training.apnic.net. A 10.0.0.2
```

Glue

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

```
training.apnic.net.     NS      ns1.training.apnic.net.
training.apnic.net.     NS      ns2.training.apnic.net.

training.apnic.net.     NS      ns2.apnic.net.
training.apnic.net.     NS      ns1.apnic.net.

ns1.training.apnic.net. A   10.0.0.1
Ns2.training.apnic.net. A   10.0.0.2
```

Only this record needs glue

Delegating training.apnic.net. from apnic.net.

training.apnic.net

Setup minimum two servers

Create zone file with NS records

Add all training.apnic.net data

apnic.net

Add NS records and glue

Make sure there is no other data from the training.apnic.net. zone in the zone file

Questions ?

BIND

APNIC

Retrieving BIND

- HTTP, FTP
  - Internet Systems Consortium
    - http://www.isc.org
- Other packages
  - OpenSSL
    - Will be needed for DNSSEC

BIND

- Version 9
  - Current version (9.9.2)
    - Release
    - Release Candidate (Betas)
    - Snapshots (Alphas)
  - Never Use Snapshots on production servers

- Getting BIND
  - http://www.isc.org/software/bind/992/download/bind-992targz

Unpacking BIND9

- tar xvfz bind-9.9.2.tar.gz
  - Uncompresses and creates directory
  - bind-9.9.2
- What's in there?
  - A lot of stuff (dig, libraries etc)
  - ./configure (script)
  - ./doc/arm/Bv9ARM.html
    - Administrator's Reference Manual
    - Good source!!!

Building BIND9

- must be in the BIND 9.9.2 directory

```
>./configure –with-openssl
```

  – Determine the appropriate includes and compiler settings

```
> make
```

  – Build and compile

```
> make install
```

  – sudo (if not root)
  – Install BIND

What happens

- Executables
  - /usr/local/sbin
    - dnssec-keygen, dnssec-makekeyset, dnssec-signkey, dnssec-signzone
    - lwresd, named-checkconf, named-checkzone
    - rndc, rndc-confgen
    - named
  - /usr/local/bin
    - dig
    - host, isc-config.sh, nslookup
    - nsupdate
- And libraries included

Testing

- Make sure right version is now installed

  `> named -v`

  > BIND 9.9.2

Bind DNSSEC Tools

- Named
- dnssec-keygen
  - Generate keys of various types
- dnssec-signzone
  - Sign a zone
- dig
  - Troubleshoot: Usage: dig +dnssec @…
- named-checkzone & named-checkconf
  - syntax check for zonefiles and named.conf

Server/Named Configuration

- The configuration file is called “named.conf”
- Documentation in <src>/doc/arm/Bv9ARM.html
- Turn on logging for troubleshooting
  - Several categories
  - Categories are processed in one or more channels
  - Channels specify where the output goes

Questions ?

Recursive Server

Overview

- Recursive Service
- Root server list
- localhost
- 0.0.127.in-addr.arpa
- named.conf

Recursive Server

- Used to lookup data by applications
- Needs to know how to reach top of DNS
- Also should stop some queries
  - localhost, 127.0.0.1
- Files
  - named.conf
  - root.hints
  - localhost zone
  - 0.0.127.in-addr.arpa zone

Root server list

- List of the 13 root server records
- Where to get it
  - ftp rs.internic.net
    - anonymous login
    - cd domain
    - get one of these files (they are the same)
      - db.cache
      - named.root
      - named.cache

What it looks like

```
;   This file holds the information on root name servers needed to

;       initialize cache of Internet domain name servers (e.g. reference this file in the
   "cache  .  <file>"

;       configuration file of BIND domain name servers).

;  This file is made available by InterNIC under anonymous FTP as

;   file                /domain/named.root on server           FTP.INTERNIC.NET

;       -OR-                    RS.INTERNIC.NET

;       last update:    Feb 04, 2008 related version of root zone:   2008020400

; formerly NS.INTERNIC.NET

.                        3600000  IN  NS    A.ROOT-SERVERS.NET.

A.ROOT-SERVERS.NET.      3600000      A     198.41.0.4

A.ROOT-SERVERS.NET.      3600000      AAAA  2001:503:BA3E::2:30

; operated by WIDE

.                        3600000      NS    M.ROOT-SERVERS.NET.

M.ROOT-SERVERS.NET.      3600000      A     202.12.27.33

M.ROOT-SERVERS.NET.      3600000      AAAA  2001:dc3::35

; End of File
```

localhost

- Loopback name in operating systems
- Means 127.0.0.1
- Queries for this shouldn't use recursion
- So we will configure a file to define the localhost. zone
  - Note the "."

localhost file

```
$TTL 86400
@       IN      SOA localhost. root.localhost. (
                        1       ; serial
                        1800            ; refresh
                        900     ; retry
                        69120           ; expire
                        1080            ; negative cache ttl
                        )
                NS      localhost.
                A       127.0.0.1
```

Reverse for localhost

- Since we want "localhost -> 127.0.0.1" we want to have "127.0.0.1 -> localhost"

- We need a zone called 0.0.127.in-addr.arpa.

0.0.127.in-addr.arpa file

```
$TTL 86400
@         IN      SOA localhost. root.localhost. (
                        1         ; serial
                        1800      ;refresh
                        900       ;retry
                        69120     ;expire
                        1080      ;negative cache ttl
                        )
                  NS       localhost.
        1         PTR      localhost.
```

Assembling the files

- Here's my directory:

```
[/var/named/recursive] % ls
0.0.127.in-addr.arpa   localhost  named.root
```

- The directory name and file names will be in named.conf

- Now create a named.conf file in the same directory

named.conf

```
options {
     directory "/var/named/recursive";
       recursion yes;    // by default recursion is on
};
zone "." {
        type hint;
        file "named.root";
};
zone "localhost." {
        type master;
        file "localhost";
};
zone "0.0.127.in-addr.arpa." {
        type master;
        file "0.0.127.in-addr.arpa";
};
```

Running the server

- From the directory

```
% named -g -c named.conf
```

Testing the server

- Just to show it is alive

```
% dig @127.0.0.1 www.arin.net
; <<>> DiG 9.2.2rc1 <<>> @127.0.0.1 www.arin.net
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16580
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 10, ADDITIONAL: 0
;; QUESTION SECTION:
;www.arin.net.                  IN      A
;; ANSWER SECTION:
www.arin.net.           10800   IN      A       192.149.252.17
www.arin.net.           10800   IN      A       192.149.252.16
;; AUTHORITY SECTION:
arin.net.               10800   IN      NS      arrowroot.arin.net.
(and so on)
;; Query time: 3066 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Feb 19 11:07:05 2003
;; MSG SIZE  rcvd: 251
```

Questions ?

Reverse DNS

APNIC

Overview

- Principles
- Creating reverse zones
- Setting up nameservers
- Reverse delegation procedures

What is ‘Reverse DNS’?

- ‘Forward DNS’ maps names to numbers
  - svc00.apnic.net -> 202.12.28.131

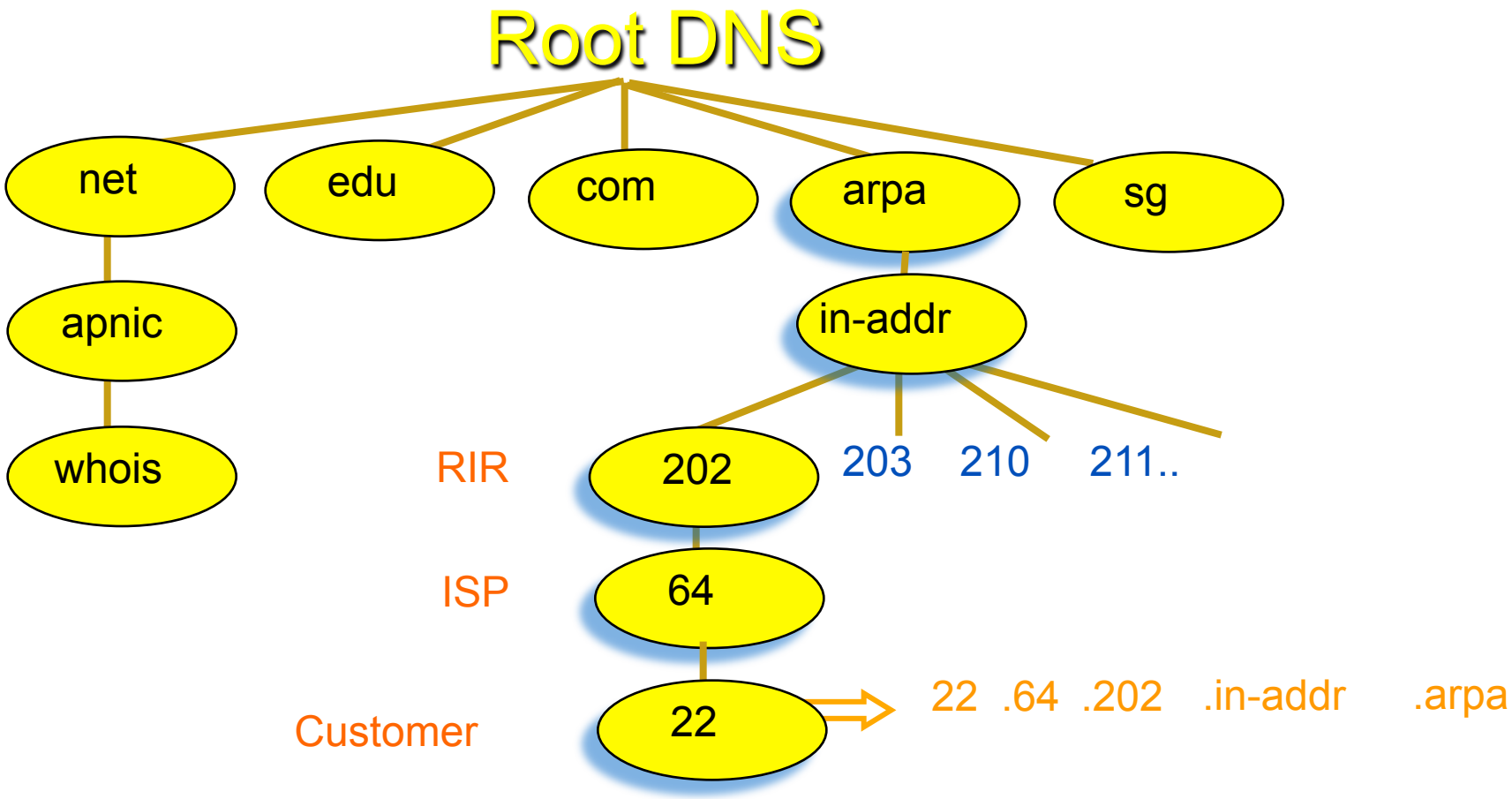
- ‘Reverse DNS’ maps numbers to names
  - 202.12.28.131 -> svc00.apnic.net

Reverse DNS - why bother?

- Service denial
  - That only allow access when fully reverse delegated eg. anonymous ftp
- Diagnostics
  - Assisting in trace routes etc
- SPAM identifications
- Registration responsibilities

Principles – DNS tree

- Mapping numbers to names - 'reverse DNS'



Creating reverse zones

- Same as creating a forward zone file
  - SOA and initial NS records are the same as normal zone
  - Main difference
    - need to create additional PTR records

- Can use BIND or other DNS software to create and manage reverse zones
  - Details can be different

Creating reverse zones - contd

- Files involved
  - Zone files
    - Forward zone file
      - e.g. db.domain.net
    - Reverse zone file
      - e.g. db.192.168.254
  - Config files
    - <named.conf>
  - Other
    - Hints files etc.
      - Root.hints

Start of Authority (SOA) record

```
<domain.name.>          CLASS   SOA     <hostname.domain.name.>
<mailbox.domain.name> (
                                                <serial-number>
                                                <refresh>
                                                <retry>
                                                <expire>
                                                <negative-caching> )
```

253.253.192.in-addr.arpa.

Pointer (PTR) records

- Create pointer (PTR) records for each IP address

```
131.28.12.202.in-addr.arpa. IN PTR svc00.apnic.net.
```

or

```
131             IN      PTR             svc00.apnic.net.
```

A reverse zone example

```
$ORIGIN 1.168.192.in-addr.arpa.
@       3600   IN SOA test.company.org. (
                      sys\.admin.company.org.
                      2002021301    ; serial
                      1h            ; refresh
                      30M           ; retry
                      1W            ; expiry
                      3600 )        ; neg. answ. ttl

        NS     ns.company.org.
        NS     ns2.company.org.

1       PTR    gw.company.org.
               router.company.org.

2       PTR    ns.company.org.
;auto generate:  65 PTR host65.company.org
$GENERATE 65-127 $ PTR host$.company.org.
```

Setting up the primary nameserver

- Add an entry specifying the primary server to the ***named.conf*** file

```
zone "<domain-name>" in {
type master;
file "<path-name>"; };
```

- <domain-name>
  - Ex: 28.12.202.in-addr.arpa.
- <type master>
  - Define the name server as the primary
- <path-name>
  - location of the file that contains the zone records

Setting up the secondary nameserver

- Add an entry specifying the primary server to the ***named.conf*** file

```
zone "<domain-name>" in {
type slave;
file "<path-name>";
Masters { <IP address> ; }; };
```

- <type slave> defines the name server as the secondary
- <ip address> is the IP address of the primary name server
- <domain-name> is same as before
- <path-name> is where the back-up file is

Reverse delegation requirements

- /24 Delegations
  - Address blocks should be assigned/allocated
  - At least two name servers
- /16 Delegations
  - Same as /24 delegations
  - APNIC delegates entire zone to member
  - Recommend APNIC secondary zone
- < /24 Delegations
  - Read “classless in-addr.arpa delegation”

APNIC & ISPs responsibilities

- APNIC
  - Manage reverse delegations of address block distributed by APNIC
  - Process organisations requests for reverse delegations of network allocations
- Organisations
  - Be familiar with APNIC procedures
  - Ensure that addresses are reverse-mapped
  - Maintain nameservers for allocations
    - Minimise pollution of DNS

Subdomains of in-addr.arpa domain

- Example: an organisation given a /16
  - 192.168.0.0/16 (one zone file and further delegations to downstreams)
  - 168.192.in-addr.arpa zone file should have:

```
0.168.192.in-addr.arpa.  NS ns1.organisation0.com.
0.168.192.in-addr.arpa.  NS ns2.organisation0.com.
1.168.192.in-addr.arpa.  NS ns1.organisation1.com.
1.168.192.in-addr.arpa.  NS ns2.organisation1.com.
2.168.192.in-addr.arpa.  NS ns1.organisation2.com.
2.168.192.in-addr.arpa.  NS ns2.organisation2.com.
        :
```

Subdomains of in-addr.arpa domain

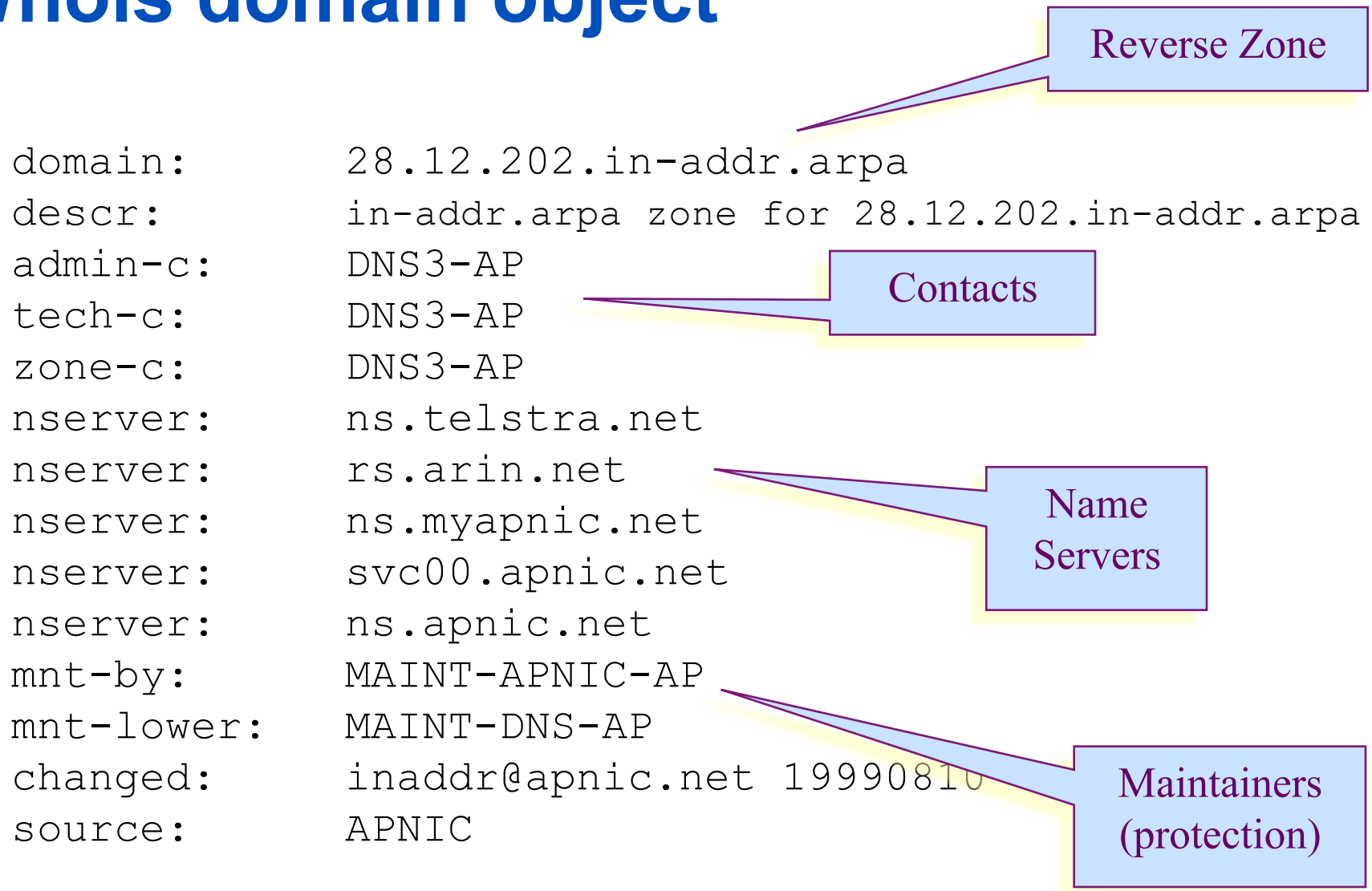
- Example: an organisation given a /20
  - 192.168.0.0/20 (a lot of zone files!) – have to do it per /24)
  - Zone files

0.168.192.in-addr.arpa.
1.168.192.in-addr.arpa.
2.168.192.in-addr.arpa.
:
:
15.168.192.in-addr.arpa.

Reverse delegation procedures

- Standard APNIC database object,
  - can be updated through myAPNIC.
- Nameserver/domain set up verified before being submitted to the database.
- Protection by maintainer object
  - (current auths: CRYPT-PW, PGP)
- Any queries
  - Contact <helpdesk@apnic.net>

Whois domain object



Removing lame delegations

- Objective
  - To repair or remove persistently lame DNS delegations
- DNS delegations are lame if:
  - Some or all of the registered DNS nameservers are unreachable or badly configured
- APNIC has formal implementation of the lame DNS reverse delegation procedures

Questions ?

DNS and IPv6

APNIC

IPv6 Representation in the DNS

- Forward lookup support: Multiple RR records for name to number
  - AAAA (Similar to A RR for IPv4)
- Reverse lookup support:
  - Reverse nibble format for zone ip6.arpa
- Multiple addresses are possible for any given name
  - Ex: in a multi-homed situation
- Can assign A records and AAAA records to a given name/ domain
- Can also assign separate domains for IPv6 and IPv4

Sample Forward Lookup File

```
apnic.net. 7200 IN      SOA     ns.apnic.net. admin.apnic.net.
                       (
                        2010020901     ; Serial
                        12h  ; Refresh 12 hours
                        4h  ; Retry 4 hours
                        4d ; Expire 4 days
                        2h ; Negative cache 2 hours )


apnic.net.                              7200  IN    NS
ns.apnic.net.
server1.apnic.net.     3600  IN    A       193.0.1.162
                                           3600  IN    AAAA
2001:0db8:1230::ABC:1
```

IPv6 Reverse Lookups – PTR records

- Similar to the IPv4 reverse record

```
b.a.9.8.7.6.5.0.4.0.0.0.3.0.0.0.2.0.0.0.1.0.0.0.0.0.0.0.1.2.3.4.ip6.arpa.
        IN      PTR   test.ip6.example.com.
```

- Example: reverse name lookup for a host with address

  3ffe:8050:201:1860:42::1

```
$ORIGIN 0.6.8.1.1.0.2.0.0.5.0.8.e.f.f.3.ip6.arpa.
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.4.0.0   14400   IN PTR host.example.com.
```

Sample Reverse Lookup File

```
$ORIGIN 0.0.0.0.4.3.2.1.8.B.D.0.1.0.0.2
apnic.net. 7200 IN      SOA     ns.apnic.net. admin.apnic.net.
                        (
                         2010020901     ; Serial
                         12h  ; Refresh 12 hours
                         4h   ; Retry 4 hours
                         4d ; Expire 4 days
                         2h ; Negative cache 2 hours )

apnic.net.              7200  IN    NS      ns.apnic.net.
1.C.B.A.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0 3600  IN    PTR server1.apnic.net.
```

IPv6 in the Root Servers

- http://www.internic.net/zones/named.root
- 9 of 13 root servers have IPv6 AAAA records
  - C, E, G root servers don't have IPv6 capability yet
  - root.hints file contains the IP address of the root servers

IPv6 in TLDs

- Total number of TLDs: 313
- TLDs with IPv6: 266
- Registered domains with AAAA records
  - COM: 760,678 of 101,872,424 domains
  - NET: 170,062 of 14,624,650 domains

Source: Global IPv6 Deployment Progress Report
http://bgp.he.net/ipv6-progress-report.cgi

Using BIND with IPv6

- BIND options for IPv6
  - `Listen-on-v6 { };`
  - `Query-source-v6 { };`
  - `Use-v6-udp-ports or avoid-v6-udp-ports`
  - `Transfer-source-v6`
- AAAA records
- PTR records
  - In named.conf

```
Zone "1.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa" {
        Type master;
        File "ipv6ptr.zone";
};
```

  - In zone file

```
4.3.2.1.0.0.0.1.0.0.0.0. IN PTR www.example.com
```

Questions?

Transaction Signatures (TSIG)

APNIC

What is TSIG - Transaction Signature?

- A mechanism for protecting a message from a primary to secondary and vice versa

- A keyed-hash is applied (like a digital signature) so recipient can verify message
  - DNS question or answer
  - & the timestamp

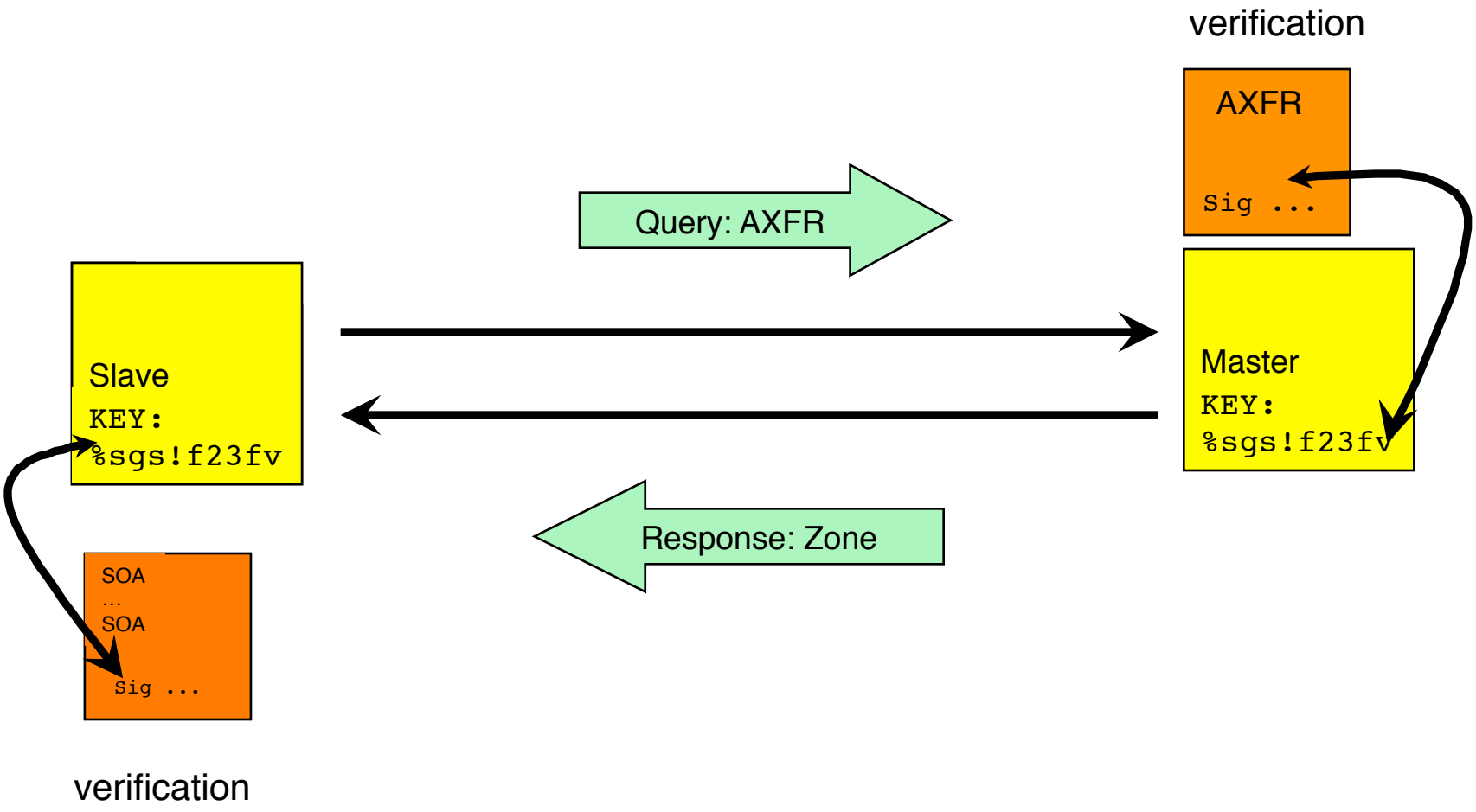
- Based on a shared secret - both sender and receiver are configured with it

What is TSIG - Transaction Signature?

- TSIG (RFC 2845)
  - authorizing dynamic updates & zone transfers
  - authentication of caching forwarders

- Used in server configuration, not in zone file

TSIG example



TSIG steps

1. Generate secret
2. Communicate secret
3. Configure servers
4. Test

TSIG - 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

TSIG – Generating a Secret

- dnssec-keygen
  - Simple tool to generate keys
  - Used here to generate TSIG keys

```
> dnssec-keygen -a <algorithm> -b <bits> -n host
  <name of the key>
```

TSIG – Generating a Secret

- Example

```
> dnssec-keygen -a HMAC-MD5 -b 128 -n HOST ns1-
  ns2.pcx.net

This will generate the key
> Kns1-ns2.pcx.net.+157+15921

>ls
```

- Kns1-ns2.pcx.net.+157+15921.key
- Kns1-ns2.pcx.net.+157+15921.private

TSIG – Generating a Secret

- TSIG should never be put in zone files!!!
  - might be confusing because it looks like RR:

```
ns1-ns2.pcx.net. IN KEY 128 3 157 nEfRX9…bbPn7lyQtE=
```

TSIG – Configuring Servers

- 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 – named.conf

Primary server 10.33.40.46

```
key ns1-ns2.pcx. net {
        algorithm hmac-md5;
        secret "APlaceToBe";
};
server 10.33.50.35 {
        keys {ns1-ns2.pcx.net;};
};
zone "my.zone.test." {
        type master;
        file “db.myzone”;
        allow-transfer {
        key ns1-ns2..pcx.net ;}; };
};
```

Secondary server 10.33.50.35

```
key ns1-ns2.pcx.net {
        algorithm hmac-md5;
        secret "APlaceToBe";
};
server 10.33.40.46 {
  keys {ns1-ns2.pcx.net;};
};
zone "my.zone.test." {
        type slave;
        file “myzone.backup”;
        masters {10.33.40.46;};
```

You can save this in a file and refer to it in the named.conf using ‘include’ statement:

```
include “/var/named/master/tsig-key-ns1-ns2”;
```

TSIG Testing : dig

- You can use dig to check TSIG configuration
  - `dig @<server> <zone> AXFR -k <TSIG keyfile>`

```
$ dig @127.0.0.1 example.net AXFR \
    -k Kns1-ns2.pcx.net.+157+15921.key
```

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

TSIG Testing - 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

Questions?

DNS Security : DNSSEC Deployment

APNIC

Overview

- Introduction
  - DNSSEC support in BIND
  - Why DNSSEC?
- DNSSEC mechanisms
  - To authenticate servers (TSIG)
  - To establish authenticity and integrity of data
    - Quick overview
    - New RRs
    - Using public key cryptography to sign a single zone
    - Delegating signing authority ; building chains of trust
    - Key exchange and rollovers
- Steps

Background

- The original DNS protocol wasn't designed with security in mind
- It has very few built-in security mechanism
- As the Internet grew wilder & wollier, IETF realized this would be a problem
  – For example DNS spoofing was to easy
- DNSSEC and TSIG were develop to help address this problem

DNS Protocol Vulnerability

- DNS data can be spoofed and corrupted between master 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 DNSSEC?

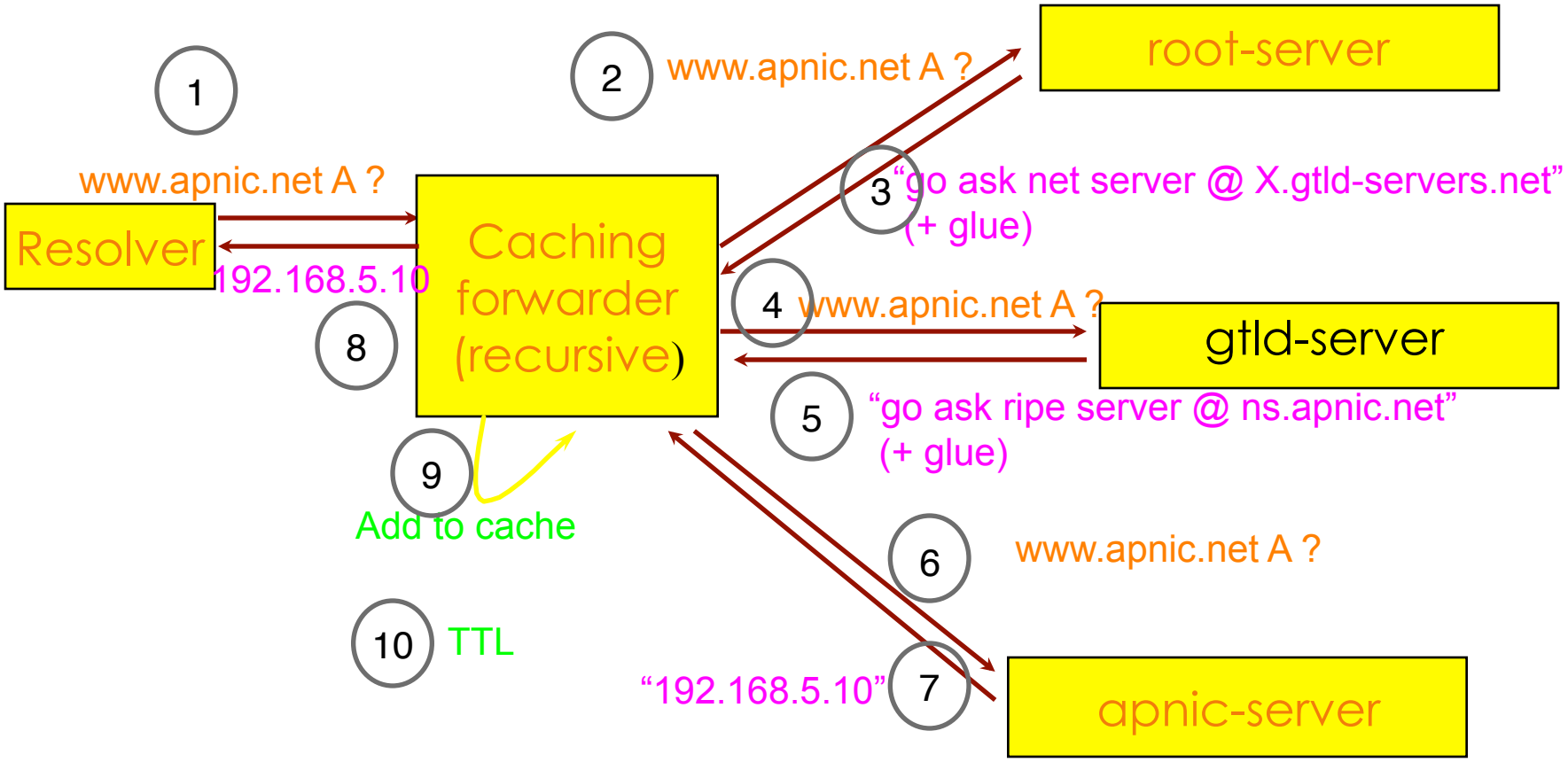
- DNS is not secure
  - Applications depend on DNS
    - Known vulnerabilities

- DNSSEC protects against data spoofing and corruption

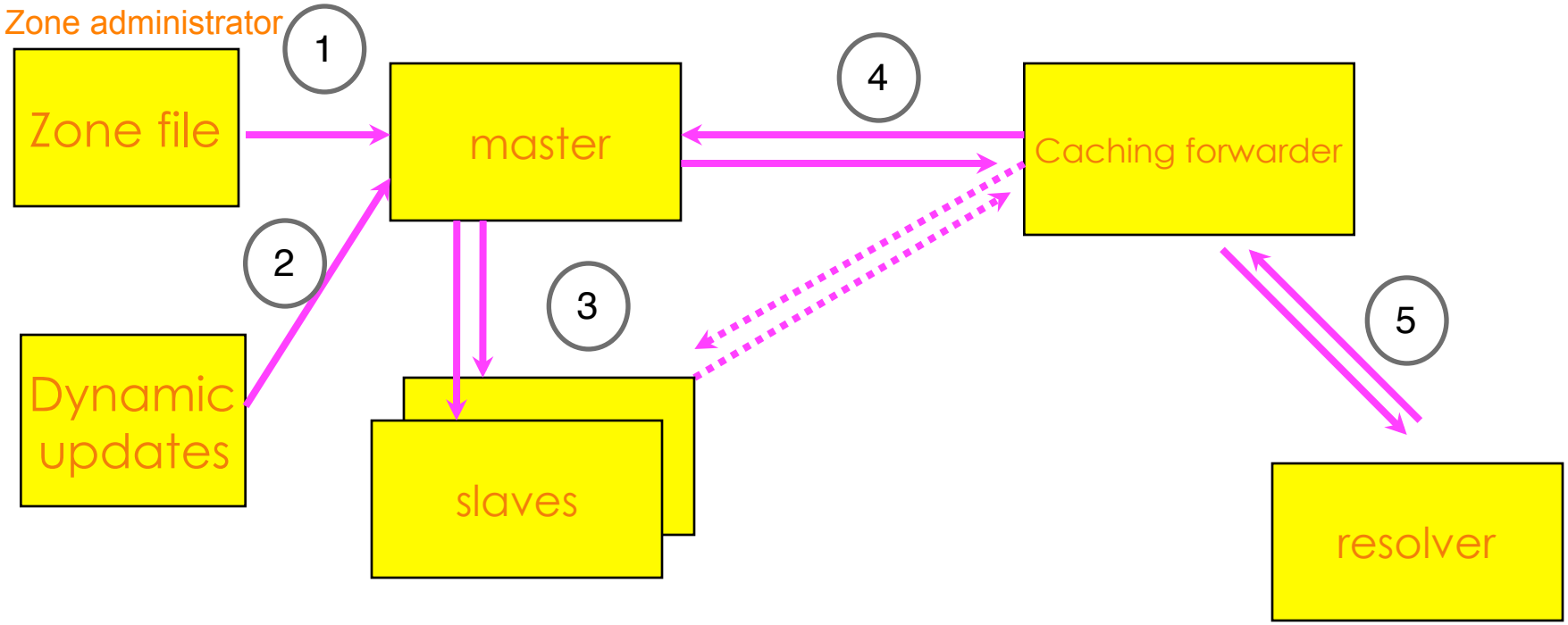
Reminder: DNS Resolving

Question:

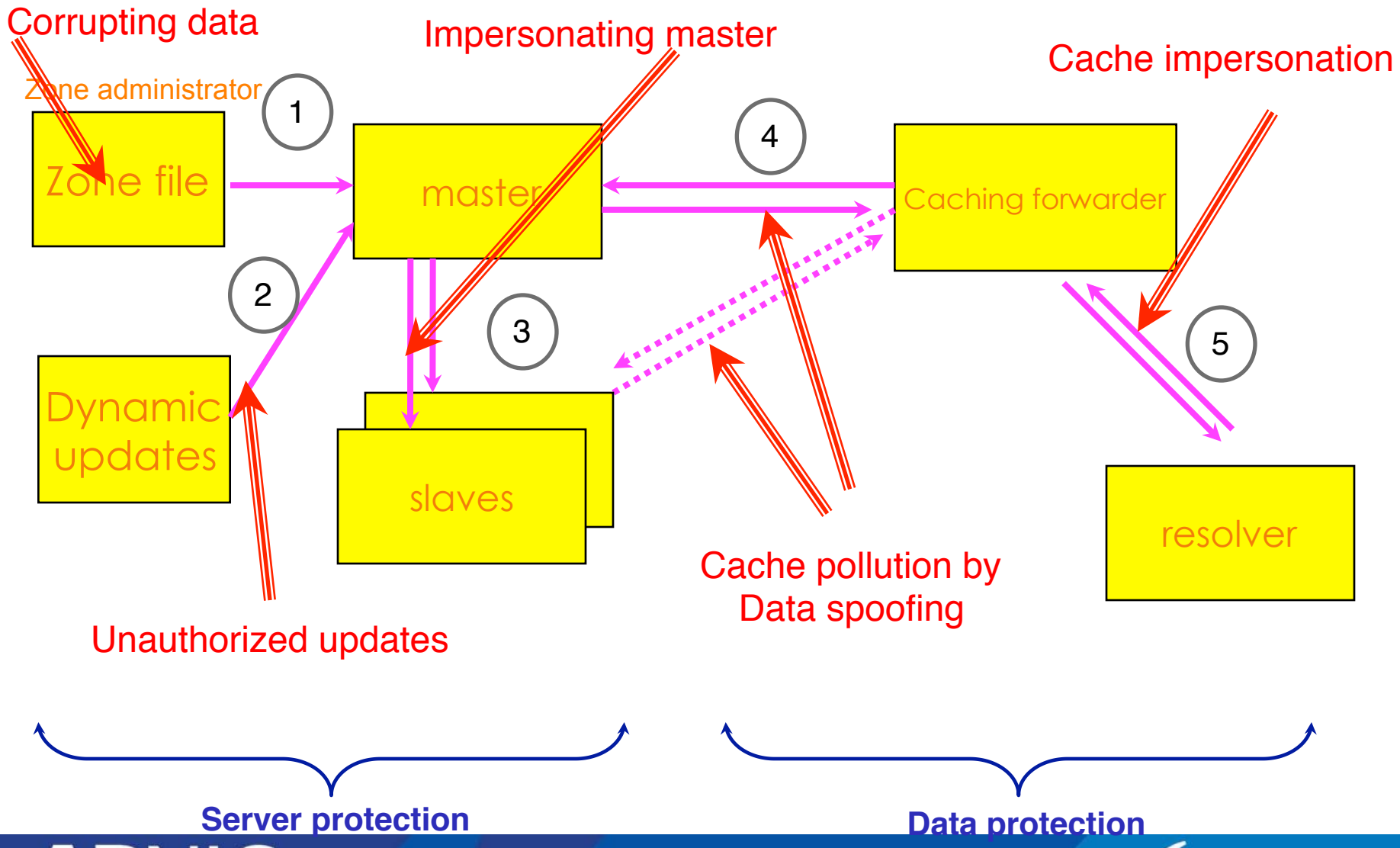
www.apnic.net A



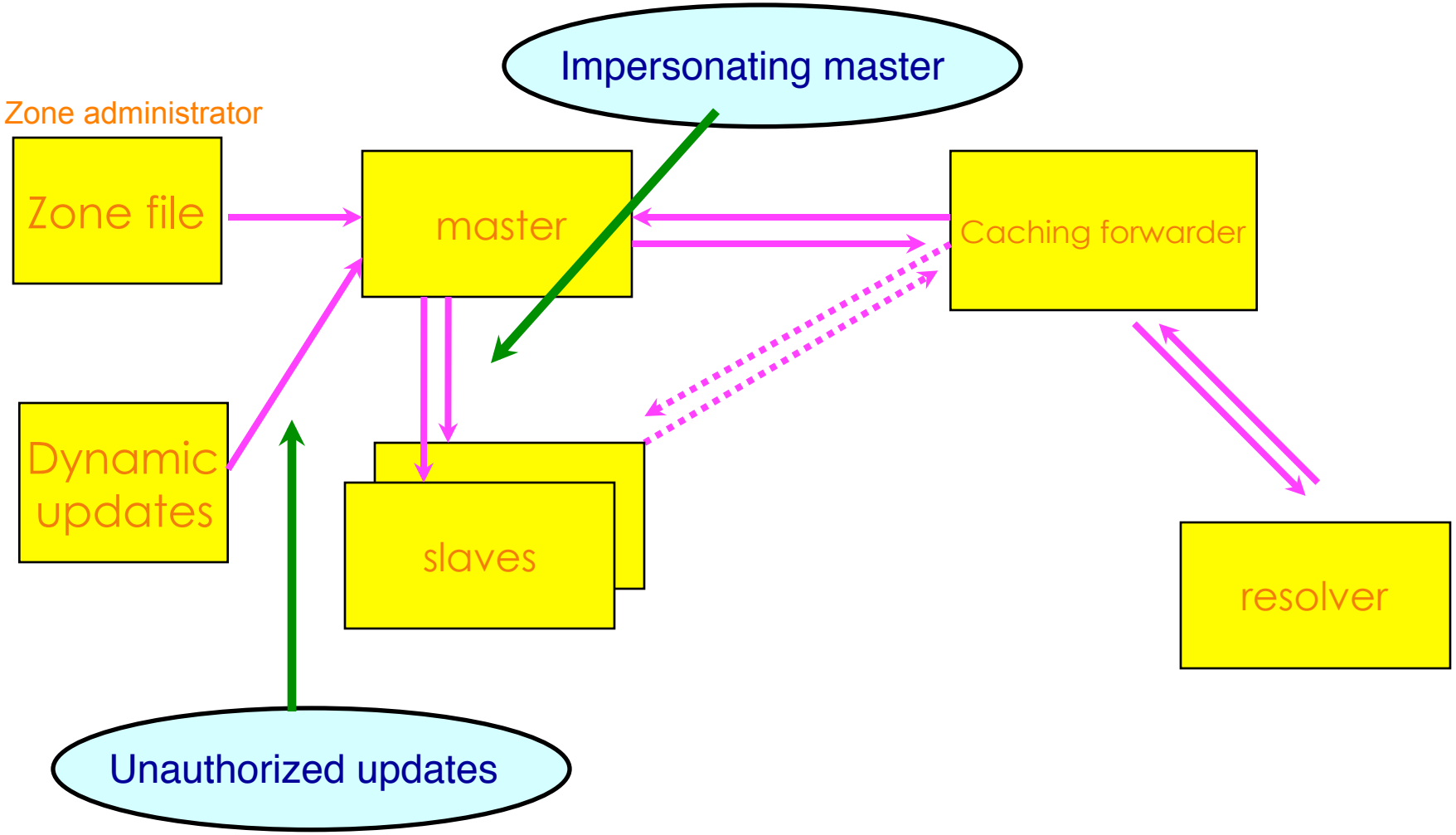
DNS: Data Flow



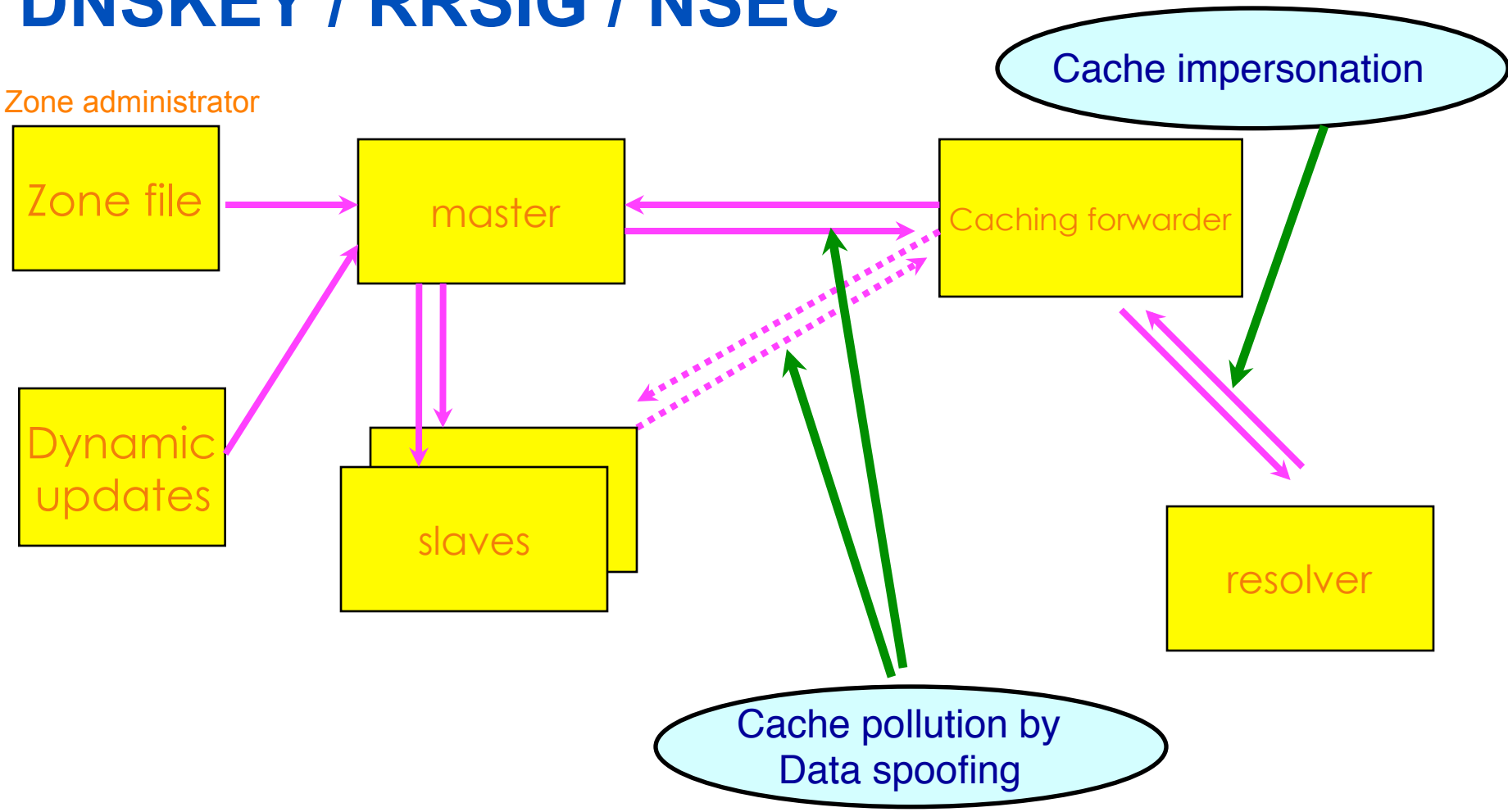
DNS Vulnerabilities



TSIG Protected Vulnerabilities



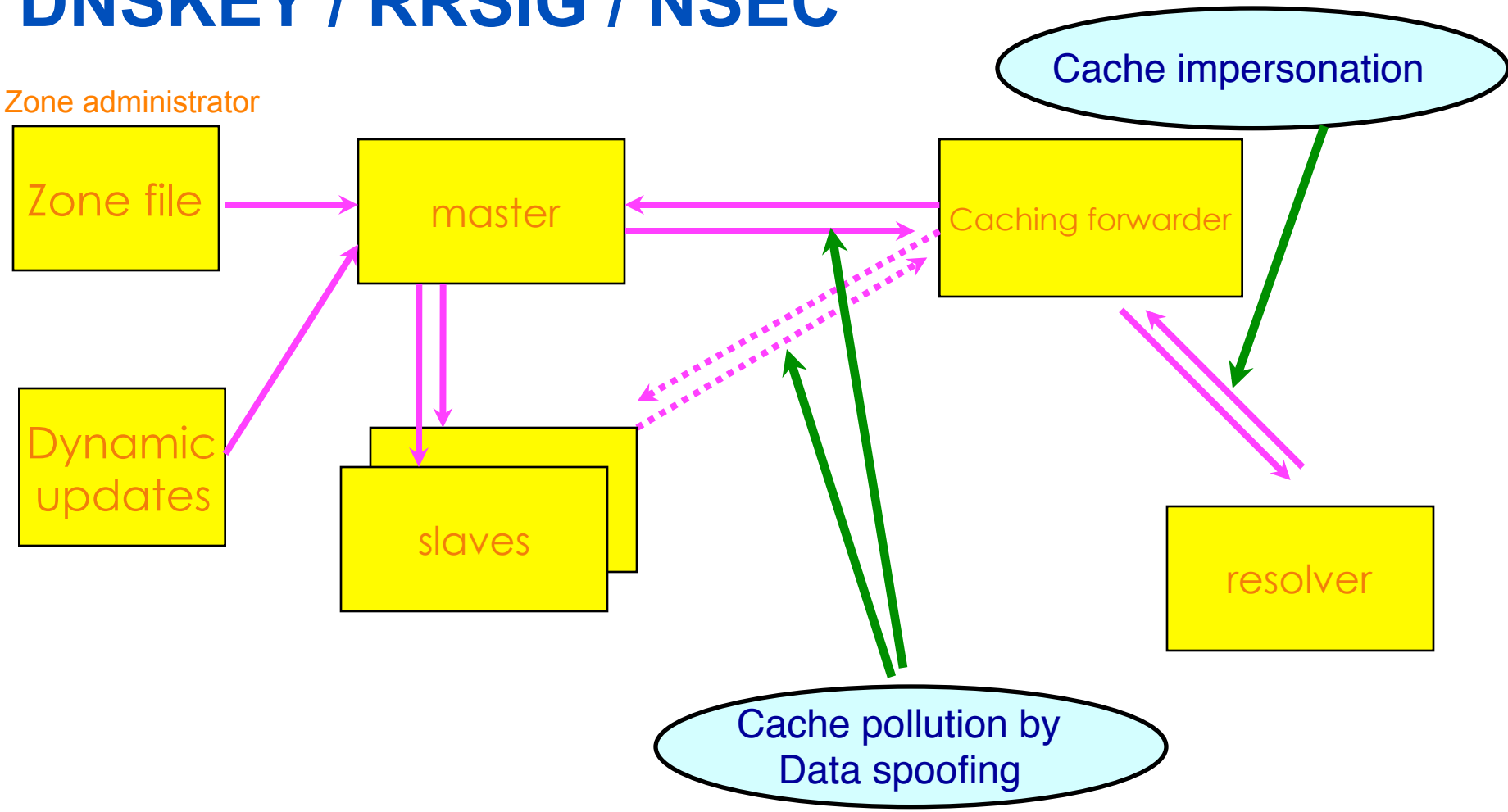
Vulnerabilities protected by DNSKEY / RRSIG / NSEC



DNSSEC mechanisms

- TSIG: provides mechanisms to authenticate communication between servers
- DNSKEY/RRSIG/NSEC: provides mechanisms to establish authenticity and integrity of data
- DS: provides a mechanism to delegate trust to public keys of third parties
- A secure DNS will be used as an infrastructure with public keys
  - However it is **NOT** a PKI

Vulnerabilities protected by DNSKEY / RRSIG / NSEC



DNSSEC RRs

- Data authenticity and integrity by signing the Resource Records Sets with private key
- Public DNSKEYs used to verify the RRSIGs
- Children sign their zones with their private key
  - Authenticity of that key established by signature/checksum by the parent (DS)
- Ideal case: one public DNSKEY distributed

New Resource Records

- 3 Public key crypto related RRs
  - RRSIG
    - Signature over RRset made using private key
  - DNSKEY
    - Public key, needed for verifying a RRSIG
  - DS
    - Delegation Signer; ‘Pointer’ for building chains of authentication
- One RR for internal consistency
  - NSEC
    - Indicates which name is the next one in the zone and which typecodes are available for the current name
    - authenticated non-existence of data

RR's and RRsets

- Resource Record:
  - Name TTL class type rdata

```
www.example.net.   7200   IN      A       192.168.1.1
```

- RRset: RRs with same name, class **and** type:

```
www.example.net.        7200   IN     A      192.168.1.1
                               A      10.0.0.3
                               A      172.10.1.1
```

- RRsets are signed, not the individual RRs

DNSKEY RDATA

```
Example:
example.net. 3600 IN DNSKEY   256  3  5   (
            AQOvhvXXU61Pr8sCwELcqqq1g4JJ
            CALG4C9EtraBKVd+vGIF/unwigfLOA
            O3nHp/cgGrG6gJYe8OWKYNgq3kDChN)
```

RRSIG RDATA

```
example.net.  3600 IN  RRSIG   A  5  2  3600  (
  20081104144523 20081004144523  3112  example.net.  VJ
  +8ijXvbrTLeoAiEk/qMrdudRnYZM1VlqhNvhYuAcYKe2X/
  jqYfMfjfSUrmhPo+0/GOZjW66DJubZPmNSYXw== )
```

Delegation Signer (DS)

- Delegation Signer (DS) RR indicates that:
  - delegated zone is digitally signed
  - indicated key is used for the delegated zone

- Parent is authorative for the DS of the childs zone
  - Not for the NS record delegating the childs zone!
  - DS **should not** be in the childs zone

DS RDATA

```
$ORIGIN .net.

example.net.          3600 IN   NS   ns.example.net

ns.example.net.       3600 IN   DS   3112  5 1 (
                                     239af98b923c023371b52

                                   1g23b92da12f42162b1a9

                                   )
```

NSEC RDATA

- Points to the next domain name in the zone
  - also lists what are all the existing RRs for “name”
  - NSEC record for last name “wraps around” to first name in zone

- Used for authenticated denial-of-existence of data
  - authenticated non-existence of TYPEs and labels

NSEC Record example

```
$ORIGIN example.net.
@ SOA        …
       NS     NS.example.net.
       DNSKEY …
       NSEC    mailbox.example.net. SOA NS NSEC DNSKEY        RRSIG

mailbox      A      192.168.10.2
                    NSEC  www.example.net.  A NSEC RRSIG
 WWW         A      192.168.10.3
                    TXT    Public webserver
                    NSEC  example.net. A NSEC RRSIG TXT
```

Setting up a secure zone

Enable dnssec

- In the named.conf,

```
Options {
        directory “….”
        dnssec-enable yes;
    dnssec-validation yes;
        };
```

Creation of keys

- Zones are digitally signed using the private key
- Can use RSA-SHA-1, DSA-SHA-1 and RSA-MD5 digital signatures
- The public key corresponding to the private key used to sign the zone is published using a DNSKEY RR

Keys

- Two types of keys
  - Zone Signing Key (ZSK)
    - Sign the RRsets within the zone
    - Public key of ZSK is defined by a DNSKEY RR
  - Key Signing Key (KSK)
    - Signed the keys which includes ZSK and KSK and may also be used outside the zone
      - Trusted anchor in a security aware server
      - Part of the chain of trust by a parent name server
  - Using a single key or both keys is an operational choice (RFC allows both methods)

Creating key pairs

- To create ZSK

  > dnssec-keygen -a rsasha1 -b 1024 -n zone champika.net

- To create KSK

  > dnssec-keygen -a rsasha1 -b 1400 -f KSK -n zone champika.net

Publishing your public key

- Using $INCLUDE you can call the public key (DNSKEY RR) inside the zone file
  - $INCLUDE /path/Kchampika.net.+005+33633.key ; ZSK
  - $INCLUDE /path/Kchampika.net.+005+00478.key ; KSK

- You can also manually enter the DNSKEY RR in the zone file

Signing the zone

> dnssec-signzone –o champika.net -t -k Kchampika.net.+005+00478 db.champika.net Kchampika.net.+005+33633

- Once you sign the zone a file with a .signed extension will be created
  - db.champika.net.signed

Testing the server

- Ask a dnssec enabled question from the server and see whether the answer contains dnssec-enabled data
  - Basically the answers are signed

> dig @localhost www.champika.net +dnssec +multiline

Testing with dig: an example

Terminal — bash — 144×46

```
bash-3.2# dig @localhost www.champika.net +dnssec +multiline

; <<>> DiG 9.6.0-APPLE-P2 <<>> @localhost www.champika.net +dnssec +multiline
; (3 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37425
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 3

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;www.champika.net.      IN A

;; ANSWER SECTION:
www.champika.net.       86400 IN A 192.168.1.2
www.champika.net.       86400 IN RRSIG A 5 3 86400 20091123163643 (
                                20091024163643 22827 champika.net.
                                Eyp1IVyQyYBLKOX2u/LT1+40xjBomXzLrcdwSErgioMb
                                pGyDWDLzP+FTbE3QCfBMLNDt2AGoYcty1cfY4li9sHkw
                                fue6hTQTSm0LhisBkVKQBy6ZD5oGiJQgaIkBGmLtVkPh
                                jGJ8Z1UhbwKcGGK13doAa+5X8mx6MXNCudiNWeg= )

;; AUTHORITY SECTION:
champika.net.           86400 IN NS ns.champika.net.
champika.net.           86400 IN RRSIG NS 5 2 86400 20091123163643 (
                                20091024163643 22827 champika.net.
                                CZsPewlhPWpYTl8wPh09QhD6pWt0If2mLVshviGKq4no
                                ISNVoijmX0LyIns+o3DZz/2+TtwoQCRFLbfI99YMS3fx
                                BHGYqFDeGItyVx3oBpmTuAtMu2+od5WFS+LClsJsEP/N
                                QvUDgtWrj8+Z0wVVj8aLe+I51h29ek7Mzk7+P4E= )

;; ADDITIONAL SECTION:
ns.champika.net.        86400 IN A 192.168.1.1
ns.champika.net.        86400 IN RRSIG A 5 3 86400 20091123163643 (
                                20091024163643 22827 champika.net.
                                eTPO5c4GscnoC9V5sR6vgDo02WgCr1T5arU7YZhWctXI
                                vkmU1ni+wguwqW6xezfB/Eu4J69bMnpQoX2zWUDtLUCM
                                +FVLsFx4Bbt+BjPEJKW03g9vv6IdKkR/pxyE1kJWJWmI
                                tR49P2dywlzqqTyvnj3F1yuFRTLHhJvfcVc+n8w= )

;; Query time: 3 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Oct 25 03:40:38 2009
;; MSG SIZE  rcvd: 610
```

Questions?

Thank You

APNIC