TELECOM ParisTech

Application layer

Courtesy of a slight part of this course belongs to the RES240 teachers

Claude Chaudet
Claude.Chaudet@enst.fr

Applications example #1:

e-mail

Application protocol example: e-mail

- e-mail behaves like regular mail
  - Messages are routed according to the destination address
  - There is no authentication of the sender's address
  - Asynchronous behavior
    - Emitter sends the message whe it is ready
    - The network routes the message as fast as possible
    - The message waits for the client in a mailbox server / on its machine

- Relies on two kinds of applicative protocols
  - Mail sending and routing
    - SMTP (Send Mail Transfer Protocol — RFC 821, 822 ; 2821, 2822, 5321)
  - Mail recovery
    - POP3 (Post Office Protocol - RFC 1939)
    - IMAP (Internet Message Access Protocol - RFC 3501)

Sending e-mails: SMTP

- Involves: the client mailer program and its SMTP server
  - Usually located in the same network (IP-based authentication)
  - Otherwise, requires a login/password
- The client issues commands
- The server answers by issuing codes
  - 3 digits-based
  - 2xx — Positive answer
    - Message sent, service ready, ...
  - 3xx — Positive intermediate reply
    - Start mail input
  - 4xx — Transient Negative Completion Reply
    - Service / mailbox not available
    - Out of memory
  - 5xx — Permanent Negative Completion Reply
    - Syntax error
  -

Protocol behavior

- 3 phases protocol
  - 1) Handshake

    HELO / EHLO
    USER / PASS
  - 2) Message transfer

    MAIL FROM
    RCPT TO
    DATA
  - 3) Connection teardown
    -
    QUIT

Messages delivery

- The message is passed from the sending server to the destination server
  - Same protocol, repeated between the servers

- The message itself repeats most headers (RFC 822)
  - To: e-mail address(es) of the primary recipient(s)
  - Cc: e-mail address(es) of the secondary recipient(s)
  - Bcc: e-mail address(es) of the invisible recipient(s)
  - From: identity of the person who created the message
  - Sender: e-mail address of the sender
  - Received: identity of every computer involved in the message delivery process (1 header per server)
  - etc. (Date: ; Message-Id: ; Subject: ; ...)
- Why duplicate headers (protocol + message body ?)

Non-ASCII messages

- An e-mail body is composed of text only
  - Message delimiters, headers, ... are just particular character chains
  - How to send images, sounds, etc. ?

- MIME (*Multi-purpose Internet Mail Extensions*)
  - Convert everything into ASCII characters
  - Base64 encoding
  - Adds particular headers to identify the contents type

A typical e-mail message

```
From: testSender@enst.fr
To: testReceiver@gmail.com
Subject: Just a small e-mail message
MIME-Version: 1.0

Content-Type: multipart/mixed; boundary=StartOfNextPart
--StartOfNextPart
Please find a nice picture attached to this message
--StartOfNextPart
Content-Transfer-Encoding: base64
Content-Type: image/jpeg
base64 encoded data .....
.........................
......base64 encoded data
--StartOfNextPart
```

SMTP issues

- Mail storm
  - Mailing lists may contain circular references (mailing list A includes mailing list B's address and vice-versa)
  - Limitation of the number of Received: headers (equivalent to the TTL)
- Loose security (SPAM)
  - Most ISPs only filter users by IP addresses
  - Sometimes, filtering based on a user / password authentication or equivalent (stronger)
  - However, e-mail clients are easily infected
- Loose security 2 (e-mail address spoofing)
  - No real authentication of the sender's address
  - Servers trust users regarding the Sender: address
  - Nobody examines the Received: fields (hidden in most clients)

Acessing e-mail

- Two main protocols

- POP3 (state-less)
  - Downloads the messages from the server to a local client
  - Possible to leave messages on the server
  - Limited set of commands => a message needs to be fully transfered to read it (e.g. all parts of a Multipart message)

- IMAP (state-full)
  - Manipulate messages on the server (synchronization)
  - Messages have a status on the server (read, replied, ...)
  - Allows separate access to the message headers
  - Requires more complex server implementation (concurrent access, storage, transactions...)

Quick Comparison

Feature	POP3	IMAP
Protocol definition	RFC 1939	RFC 2060
TCP Port	110	143
e-mail storage	User's PC	Server
e-mail reading	Off-line	On-line + caching
Connection time	Little	Much
Multiple mailboxes	No	Yes
Backup responsibility	User	ISP
Mobility suitability	Low	High
Control over downloading	Little	Great
Partial messages	No	Yes
Simplicity	Yes	No
Support	Wide	Growing

Application example #2:

DNS

Why use names?

- Names are easier to remember than IP addresses
  - http://www.enst.fr or http://137.194.2.39 ?
  - If IPv6 is deployed, the need for name resolution will be even more important...
    http://fe80::211:24ff:fe92:d444
- Names allow changing the IP address of a computer transparently
  - Re-thinking the addressing map of an organization
  - Dialup connections (ADSL, ...) and dynamic IP addresses
- Names allow to deploy redundant servers
  - Fast migration in case of failure
  - Content Delivery Networks (CDN)

History : hosts file

- At the beginning of the Internet, all peer machines addresses / names associations were included in a file called *hosts*

- Today, too many machines connected to the Internet
  - Searching a text file is long
  - Update process takes too much time

- Host file still exists and has the priority over DNS
  - Stored in /etc/ for unix systems
  - Used for static, nearby and frequently accessed hosts

```
137.194.4.250   belenos-4.enst.fr       belenos-4       # foundry
137.194.4.246   infres2-4.enst.fr       infres2-4       # sun/e250
```

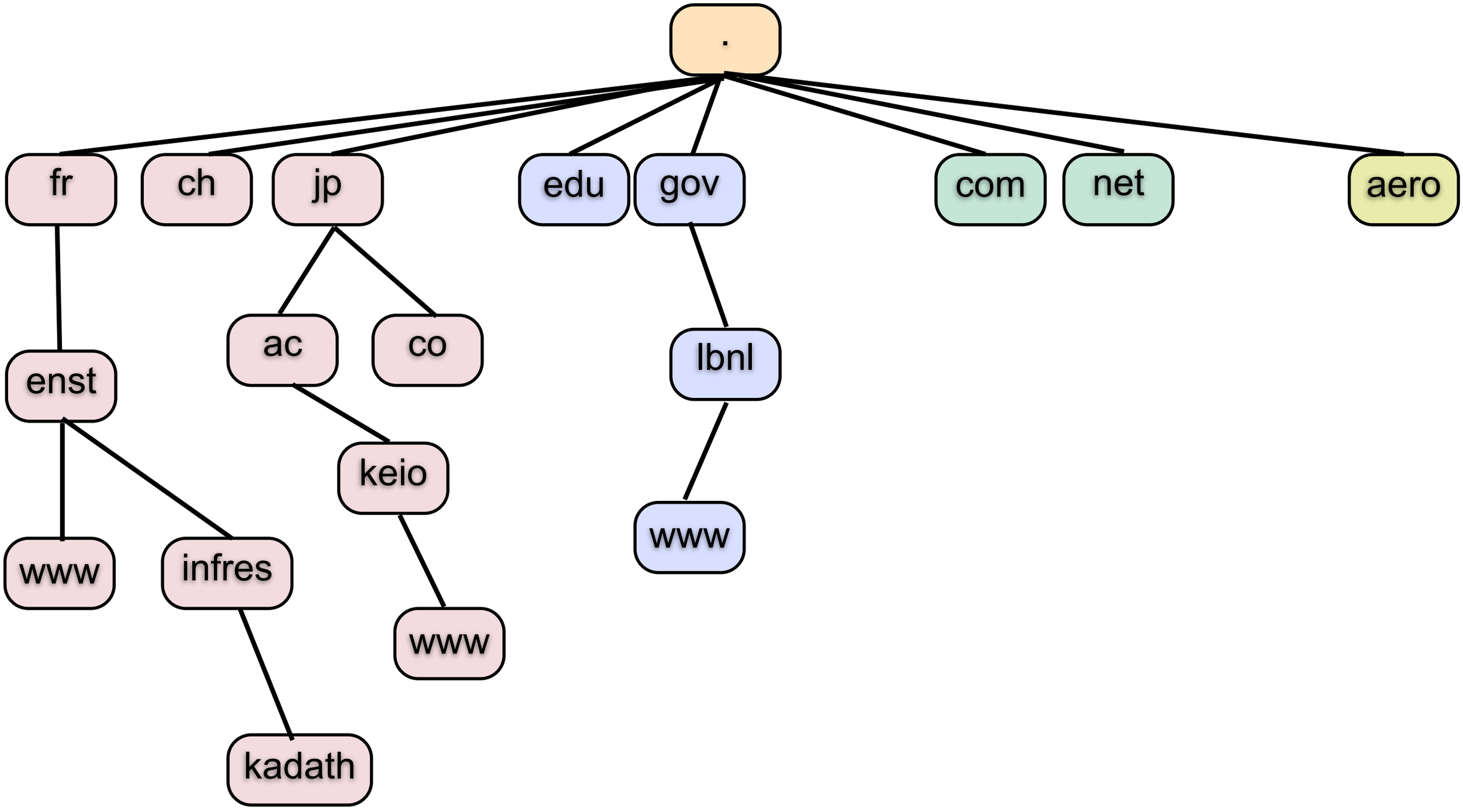
DNS Principles

- DNS is a distributed database system that can be queried by any host connected to the Internet
  - Distributed for robustness of the service
  - Contains the list of addresses, names and functions (types) of all nodes of the Internet
- DNS databases can be used for several tasks:
  - Associate an IP address to a name (classical usage)
  - Associate a name to an IP address (*Reverse DNS*)
  - Associate a name to a resource (mail server of a domain, ...)

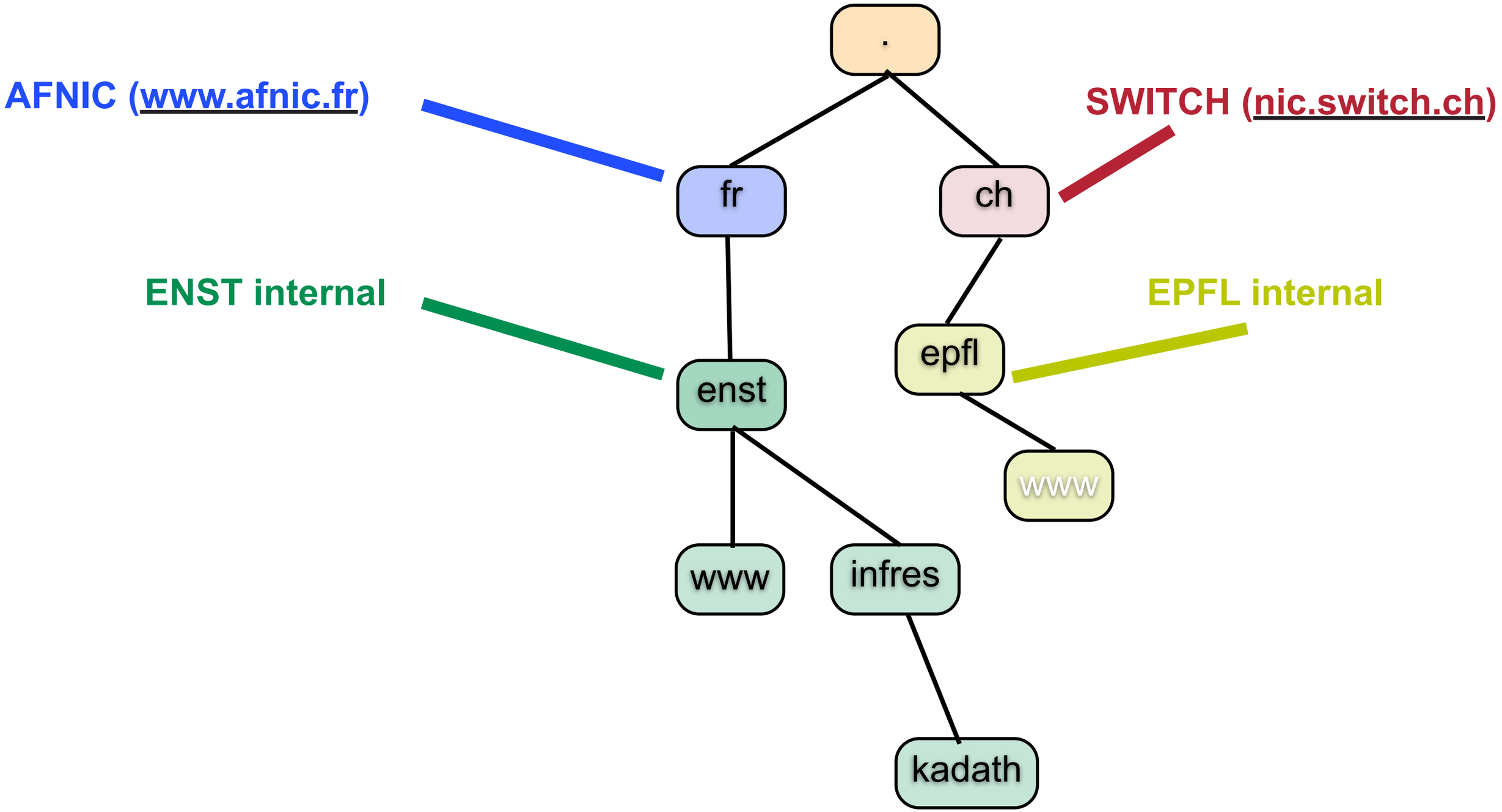
Naming space

- The naming space is organized in a hierarchical manner
  - Top Level Domains (TLD) — RFC 1591
    - Countries names (.fr ; .ch ; .jp ; .nl ; ...)
      - http://www.iana.org/root-whois/index.html
      - Conforms to the ISO 3166-1 standard
    - Restricted names (.arpa ; .edu ; .gov ; .int ; mil)
    - Generic names (.biz ; .com ; .info ; .name ; .net ; org ; .pro)
    - Sponsored (.aero ; .coop ; .jobs ; .museum ; .mobi ; .travel ; ...)
    - See http://fr.wikipedia.org/wiki/Internet_TLD for a complete list (in french)
    - New names are created by ICANN (Internet Corporation for Assigned Names and Numbers)
  - Second Level Domains
- Generic rules
  - Each level must contain less than 63 characters
  - The whole domain name (FQDN) must contain less than 255 characters
  - Case insensitive

Names hierarchy



Who allocates the (sub)names?

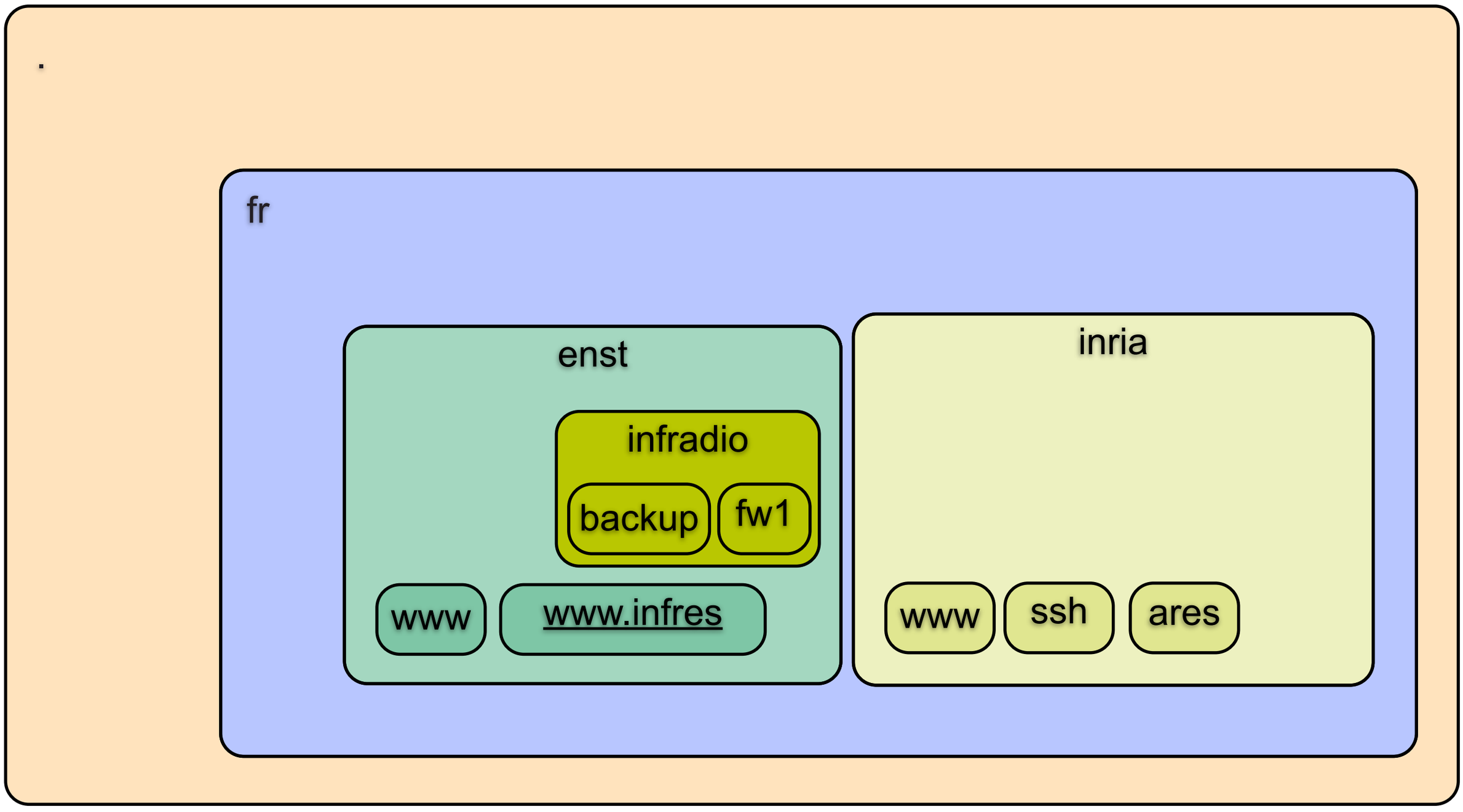


Database organization

- Some companies have the responsibility for a level of the hierarchy
  - .com & .net : Verisign ; .info : Afilias ; ...
- Organizations are able to delegate the responsibility of a sub-space (called ***zone***) to another organization
  - The father node must always know its delegations status and the IDs of its sons

- Each zone is characterized by :
  - The name of the zone and its scope
  - The address of the main database server of the zone
  - A secondary server that copies the database of the primary

Zones hierarchy



Database records details

- A database entry is composed of several fields
  - The registered domain name
  - The lifetime of the record - of the order of a minute to a day
  - The class of the record (usually a single value is used : IN)
  - The type of record (IP address, SOA, ...) => see next slides
  - The value of the record (for an IP address record type : the address)

- The database is distributed across all the servers hierarchy
  - The global database Interrogated by a mechanism of Query / Answer
  - Can be seen as an SQL request to a "black box" networked database

Querying the database

- When one host wants to find the IP address of another one:
  - It sends a message on TCP or UDP port 53 to its local DNS server
    - Address of the server can be entered manually or announced / set up by DHCP
  - If the server is the one managing both machines, it sends back an **authoritative answer**
  - Otherwise, it queries its root in the tree.
    - The root may not know the exact address of the destination host, but it knows its father and all its children.
    - *Name-based routing in the tree*
    - When the host is found a **non-authoritative** answer is sent to the machine that asked

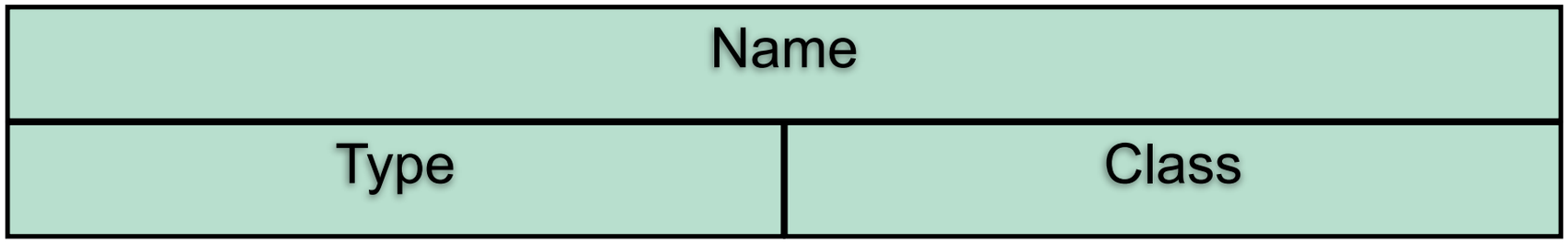
Query & Answer format

- Same message for both directions

Identification	Flags
Number of questions	Number of answers
Nb of authoritative RRs	Nb of additionnal RRs
Questions	
Answers	
Authority	
Additional information	

- ID : associates a query to the corresponding answer
- Flags : some parameters (is it a question or an answer, truncated answer, ...)

Questions



- Name = 6kadath4enst2fr0
  - Always FQDN (trailing dot)
  - Field length used as separator

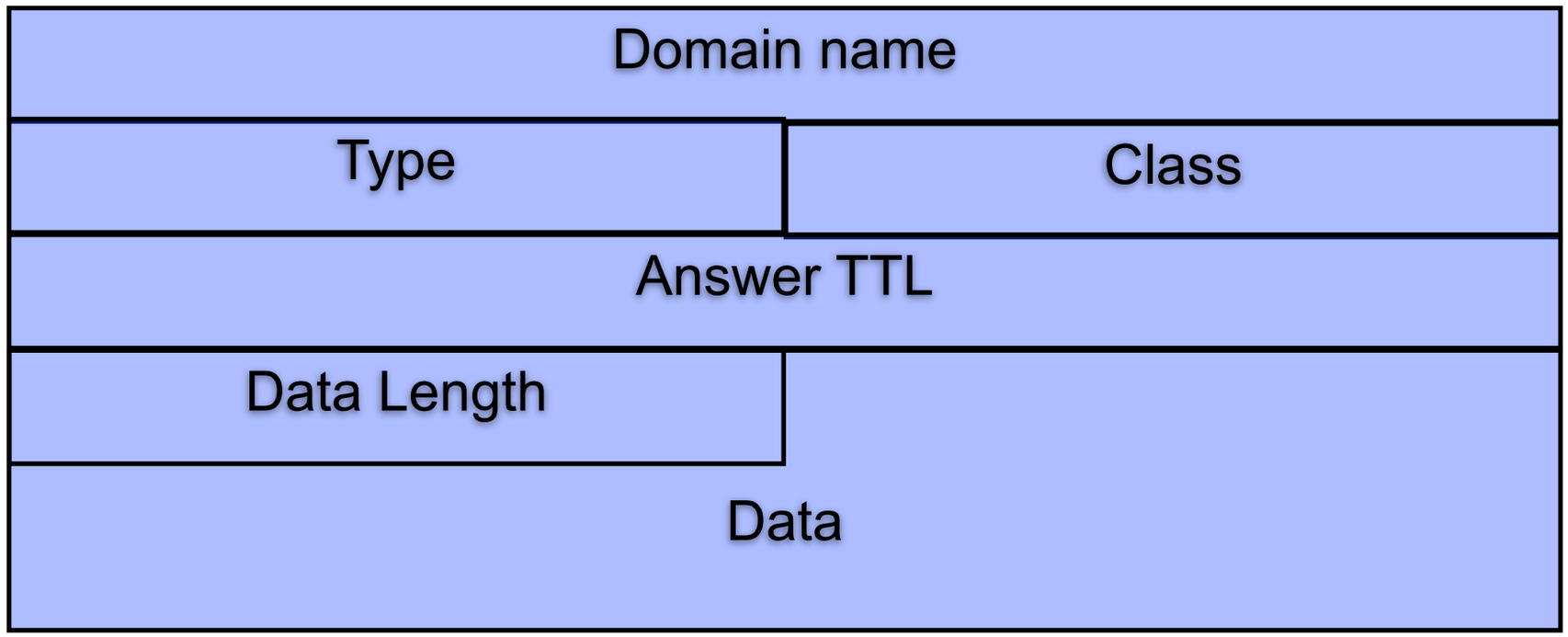
- Type = A ; NS ; CNAME ; ... (see next slide)

- Class = IN (Internet) or other networks (MIT essentially)
  - Almost always : IN

Main Types of records

- SOA (*Start of Authority*):
  - Who has the authority (administrative responsibility) of the domain?
  - Contains:
    - The address of the machine responsible for the domain
    - A serial number
    - Expiration date and refresh timeouts
- NS (*Name Server*):
  - Gives the names of the name servers in the domain
- A (*Address*):
  - Associates a name to an IP address
- MX (*Mail eXchanger*):
  - Gives the address of the machine hosting the SMTP server
- CNAME (*Cannonical Name*):
  - Gives the real name of a machine alongside its aliases (other names)

Answers



Database query example (SOA)

```
; <<>> DiG 9.3.4 <<>> -t soa +all enst.fr
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 53368
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 5, ADDITIONAL: 6

;; QUESTION SECTION:
;enst.fr.                       IN      SOA

;; ANSWER SECTION:
enst.fr.                172800  IN      SOA     minos.enst.fr. hostmaster.enst.fr. 2007111500 3600 3600 3600000
259200

;; AUTHORITY SECTION:
enst.fr.                172800  IN      NS      ns3.enst.fr.
enst.fr.                172800  IN      NS      enst.enst.fr.
enst.fr.                172800  IN      NS      minos.enst.fr.
enst.fr.                172800  IN      NS      infres.enst.fr.
enst.fr.                172800  IN      NS      phoenix.uneec.eurocontrol.fr.

;; ADDITIONAL SECTION:
ns3.enst.fr.            172800  IN      AAAA    2001:660:330f:20::54
ns3.enst.fr.            172800  IN      A       137.194.32.84
enst.enst.fr.           172800  IN      A       137.194.2.16
minos.enst.fr.          600     IN      A       137.194.2.34
infres.enst.fr.         172800  IN      A       137.194.160.3
infres.enst.fr.         172800  IN      A       137.194.192.1

;; Query time: 5 msec
;; SERVER: 137.194.164.4#53(137.194.164.4)
;; WHEN: Thu Nov 15 10:27:50 2007
;; MSG SIZE  rcvd: 298
```

Domain's authority server

Domain's Name Servers

IP Addresses of the name servers

Database query example (MX)

```
; <<>> DiG 9.3.4 <<>> -t mx +all +multiline enst.fr
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 20872
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 5, ADDITIONAL: 7

;; QUESTION SECTION:
;enst.fr.                IN MX

;; ANSWER SECTION:
enst.fr.                172800 IN MX 10 smtp2.enst.fr.

;; AUTHORITY SECTION:
enst.fr.                172800 IN NS ns3.enst.fr.
enst.fr.                172800 IN NS enst.enst.fr.
enst.fr.                172800 IN NS minos.enst.fr.
enst.fr.                172800 IN NS infres.enst.fr.
enst.fr.                172800 IN NS phoenix.uneec.eurocontrol.fr.

;; ADDITIONAL SECTION:
smtp2.enst.fr.          3600 IN A 137.194.2.14
ns3.enst.fr.            172800 IN AAAA 2001:660:330f:20::54
ns3.enst.fr.            172800 IN A 137.194.32.84
enst.enst.fr.           172800 IN A 137.194.2.16
minos.enst.fr.          600 IN A 137.194.2.34
infres.enst.fr.         172800 IN A 137.194.160.3
infres.enst.fr.         172800 IN A 137.194.192.1

;; Query time: 2 msec
;; SERVER: 137.194.164.4#53(137.194.164.4)
;; WHEN: Thu Nov 15 10:55:38 2007
;; MSG SIZE  rcvd: 289
```

Domain's MX server

Domain's Name Servers

IP Addresses of the name and MX servers

Database query example (host)

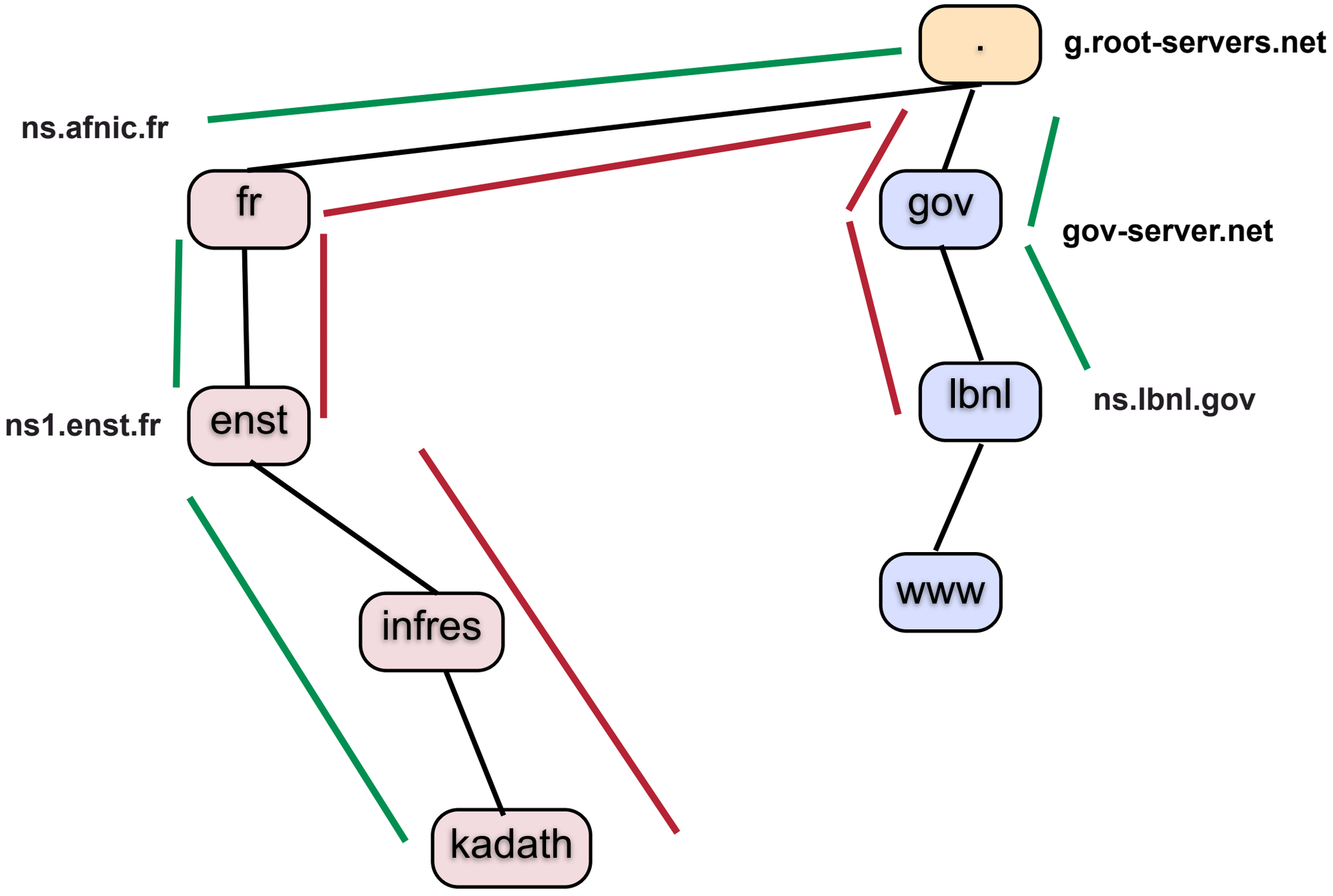
```
; <<>> DiG 9.3.4 <<>> +all ssh.enst.fr
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 25845
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 5, ADDITIONAL: 6

;; QUESTION SECTION:
;ssh.enst.fr.                   IN      A

;; ANSWER SECTION:
ssh.enst.fr.            172800  IN      CNAME   ares.enst.fr.
ares.enst.fr.           172800  IN      A       137.194.34.9

;; AUTHORITY SECTION:
enst.fr.                172800  IN      NS      ns3.enst.fr.
enst.fr.                172800  IN      NS      enst.enst.fr.
enst.fr.                172800  IN      NS      minos.enst.fr.
enst.fr.                172800  IN      NS      infres.enst.fr.
enst.fr.                172800  IN      NS      phoenix.uneec.eurocontrol.fr.

;; ADDITIONAL SECTION:
ns3.enst.fr.            172800  IN      AAAA    2001:660:330f:20::54
ns3.enst.fr.            172800  IN      A       137.194.32.84
enst.enst.fr.           172800  IN      A       137.194.2.16
minos.enst.fr.          600     IN      A       137.194.2.34
infres.enst.fr.         172800  IN      A       137.194.160.3
infres.enst.fr.         172800  IN      A       137.194.192.1

;; Query time: 2 msec
;; SERVER: 137.194.164.4#53(137.194.164.4)
;; WHEN: Thu Nov 15 10:57:37 2007
;; MSG SIZE  rcvd: 290
```

Host address and aliases

Domain's Name Servers

IP Addresses of the name servers

Query-answer example



Some notes

- A query is not sent through all the Internet for every search
  - Use of Caches at first
  - Use of an iterative search : if the server does not know the answer (not in the cache), he sends the next server's ID instead of querying it himself

- Possible round-robin algorithm
  - The server may store several addresses for a single name
  - It answers successively the first, then the second, ...

- DNS is an application-level protocol
  - It is implemented in the TCP/IP stack though : *resolver*

About the naming system

- DNS stores data on a modified US-ASCII character set
  - Not case sensitive
  - Usually only alphanumeric characters and hyphens are used
  - Internationalization process is difficult
    - 10 years of work without much result
- Trailing periods
  - They can be omitted for FQDN :
    - www.enst.fr = www.enst.fr.
  - We need to interpret names within their context
    - When there is no period in the name, append the context
      - eole => eole.enst.fr
    - When there are periods, don't
      - www.google.fr
    - What about ssh.infres ?
      - append context after querying the DNS server ?

About the RR storage

- RR consistency
  - RR describing delegation should be present in a parent's database
  - It should also be present in the son's database
  - They should be identical

  - What if they are not?
    - This happens quite often

- Truncation
  - Sometimes RR cannot be exchanged completely.
  - The transmitting node indicates truncation (Flags field)
  - The **receiver** decides whether he needs the rest of the information or not
  - Usually leads to a new transmission using TCP instead of UDP (more expensive)

Query-response behavior

- Several questions can be asked in the same query
  - Not implemented !
  - Convention : the answer "question" section contains the question that the response answers to
  - This is a convention and not specified !

- Answers may contain additional data if found useful by the server:
  - If one.com 's DNS is ns.two.com and If two.com 's DNS is ns.one.com
  - The answers from a query for one.com includes ns.two.com
  - It also includes an additional, unrequested, answer for ns.two.com which will be ignored
  - Subsequent queries...

Why all these implementation glitches?

- DNS is a "loose" specification
  - Functional interoperability
  - Ease of implementation

- Initially not designed for a planetary use
  - At the beginning, experimental software that is now used constantly throughout the world

- Strangely, this imprecise specification eases the evolutions process
  - Updating the DNS system has been around for more than 10 years

Future DNS evolutions

- Internationalization
  - Definition of IDN (Internationalized Domain Names)
  - Unicode support not directly compatible with current case insensible DNS
  - Need translation mechanisms from and to US-ASCII
- Performance enhancements
  - IXFR (Incremental Zone Transfer)
  - Allows to only communicate differences for large zones
- Secondary and primary servers coherence
  - Today, the secondary server polls the primary server regularly
  - Too fast or too slow process, depending on the cases
  - Possibility to trigger updates on changes ?
- Security concerns...
  - Authenticating database updates
  - Nobody approves messages triggering cache updates

DNS conclusion...

- On the other hand, the combination of things that were left unspecified in the protocol, things that were loosely specified in the protocol, and things that were unenforceably specified in the protocol - and implementations in the field that interpret the protocol specifications in all of their myriad ways - describes a rich and multidimensioned space where it's almost deliberately impossible to know exactly what's happening or exactly what would happen under describable circumstances. This holds true for the Internet's routing system as well, but in DNS there are more variables in every axis than in any other distributed system I've studied. We who work in the field think of DNS a little bit as though it were alive, and I hope that after reading this article, you can see why

➡ *P.Vixie, 2007*

Application Example #3:

DHCP

DHCP Introduction

- DHCP = Dynamic Host Configuration Protocol
  - Protocol destined to allocate IP addresses to terminals on a network
  - Evolution of BootP

- Main goal: send to connecting hosts the whole set of network parameters
  - Allocate an IP address and a network mask
  - Propagate the addresses of useful servers (SMTP, DNS, ...)

Protocol principles

- Simple behavior :
  - When requesting an address, the client broadcasts a DHCPDISCOVER request
  - The server replies with a broadcast DHCPOFFER packet
  - The client sends back a DHCPREQUEST unicast message to validate the IP address choice
  - The server finally answers with an unicast DHCPACK confirmation

- The address is *leased* to the client for a certain duration
  - The client may prolongate this duration by issuing another DHCPREQUEST message

Detailed behavior

- What happens when the server and the client are not on the same segment?
  - Broadcasts do not cross routers
  - Use DHCP relays to catch broadcast and retransmit it towards the real DHCP server
- Why is DHCP an application layer protocol?
  - It only concerns layer-3 addresses
  - However, it relies on a client-server architecture and may need other services provided by upper layers (reliable transmission, security, ...)
  - In the absence of a pure layer-3 solution, it has been deployed as an application layer protocol.

Conclusion on the application layer

- Application layer gathers several protocols
  - real application-specific protocols
  - general purpose helper protocols (SMTP, DNS, DHCP) that may be used for several other tasks

- This duality is due to the lack of separation, in the OSI model, of the control and user planes (see ATM model)
  - DNS, DHCP, ... work over TCP / UDP on a specified port

- It is a general tendency to conceive the TCP/IP stack as an inert block and build everything into the application layer
  - Peer-to-peer overlays
  - Applicative VPN