DNS

- Généralités
- Historique
- Arborescence
- Notion de registrar
- Architecture
- Principaux RR

Licence:

- ce document est distribué sous la licence Gnu FDL
  - version originale anglaise: http://www.gnu.org/copyleft/fdl.html
  - traduction française: http://www.idealx.org/dossier/oss/gfdl.fr.html
- une partie des diapositives de ce document ont été reprises ou adaptées à partir du support de cours dns de l'afnic que l'on peut trouver sur la page : http://www.afnic.fr/doc/formations/supports.
- le support de cours dns de l'afnic a été écrit par: Erwan.Mas@nic.fr et Mohsen.Souissi@nic.fr
- la version utilisée est celle disponible en ligne le 26/11/2005

DNS: généralités

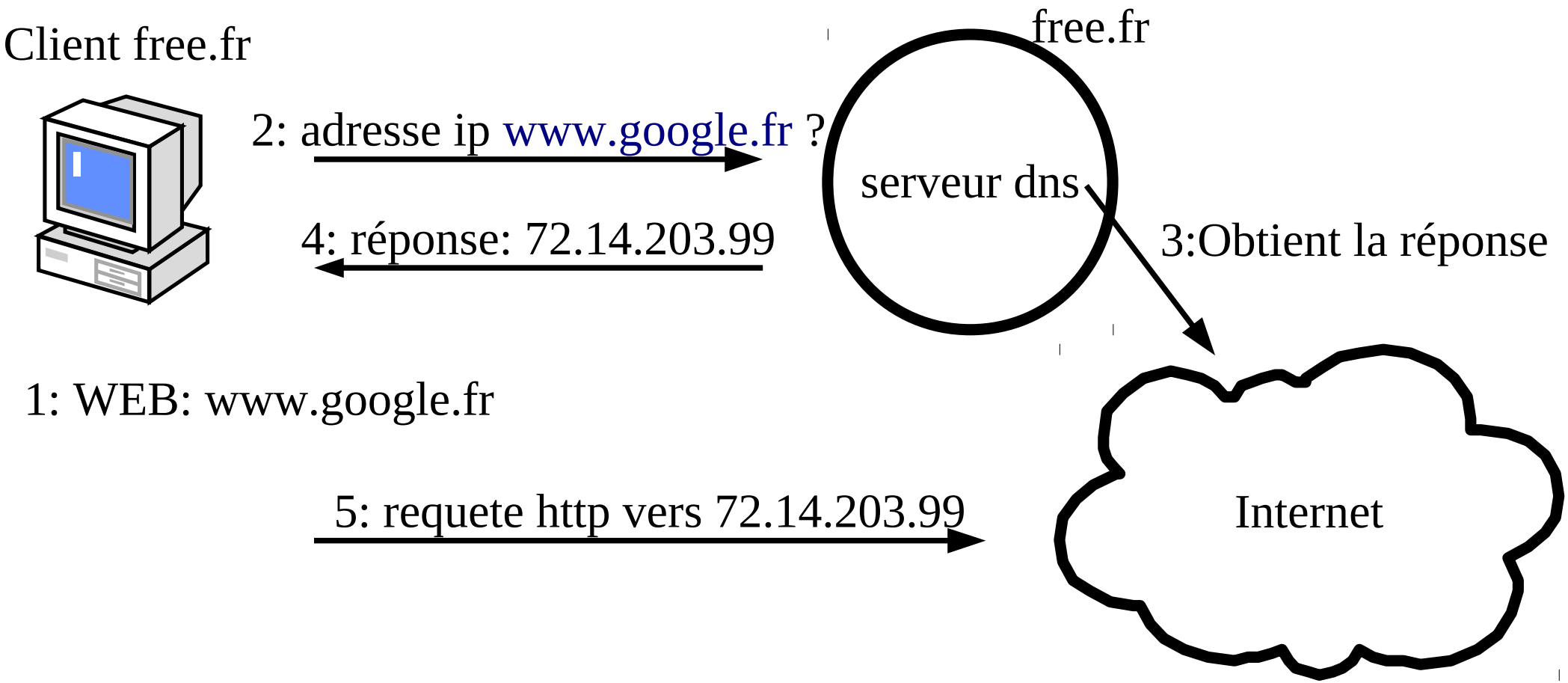
- Annuaire téléphonique:
  - utilisé par les centraux: No de tel. (01 69 47 70 00)
  - mémorisé par les humains : nom (P. Petit)
  - lien entre les deux: annuaire téléphonique
- DNS:
  - communication entre machines: adresse IP (ex.: 81.56.171.187)
  - mémorisé par les humains : nom (ex.: ns.shayol.org)

DNS : généralités

- Sans accès au dns, plus d'accès
  - au WeB (qui s'appuie beaucoup sur les noms),
  - L'accès aux autres services supposerait de connaître les adresses ip des serveurs concernés
- Dns:
  - Entrée directe : conversion nom -> adresse
  - Entrée inverse: conversion adresse -> nom
  - Sauf cas particulier rare, toute machine doit avoir une entrée directe et une entrée inverse

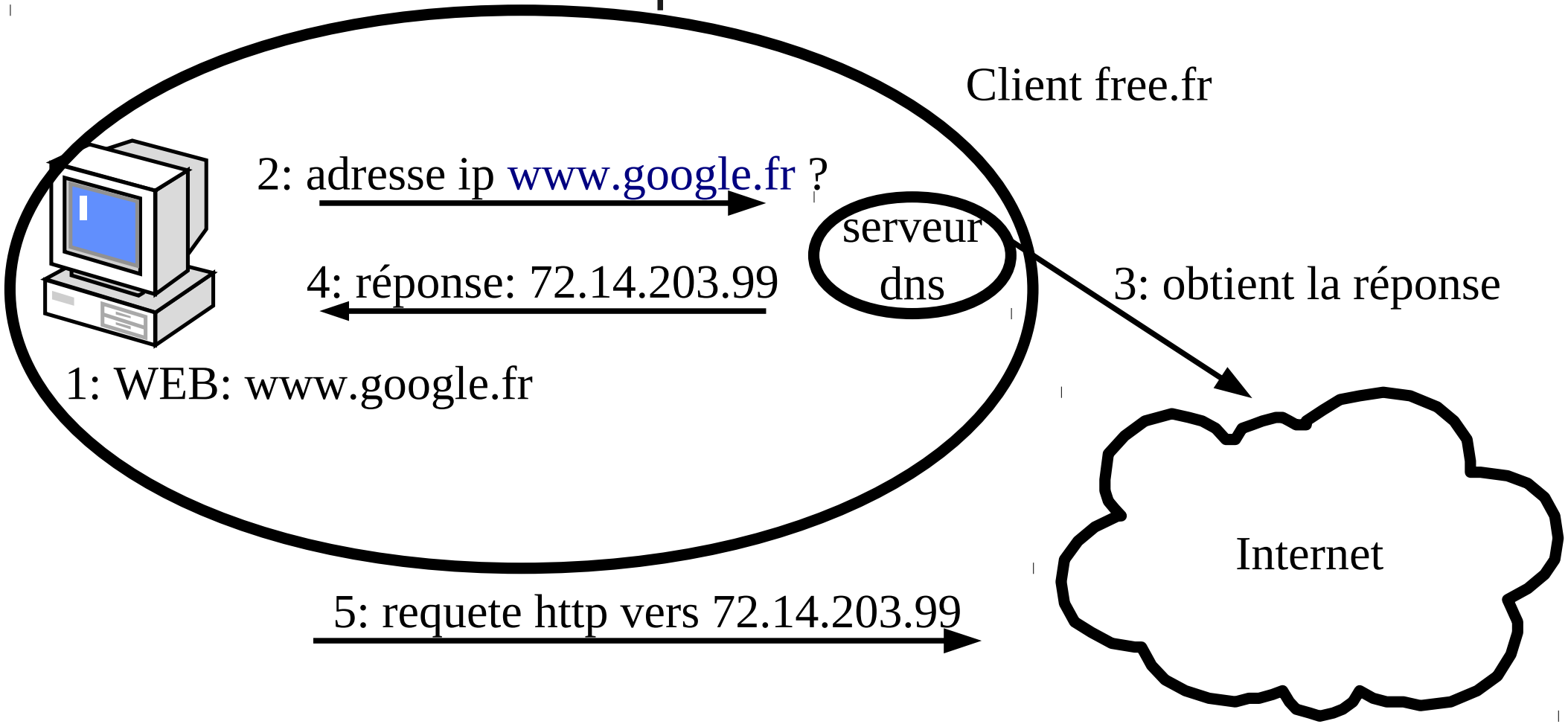
DNS: connexion domicile via un ISP

- on utilise le dns de son fournisseur d'accès



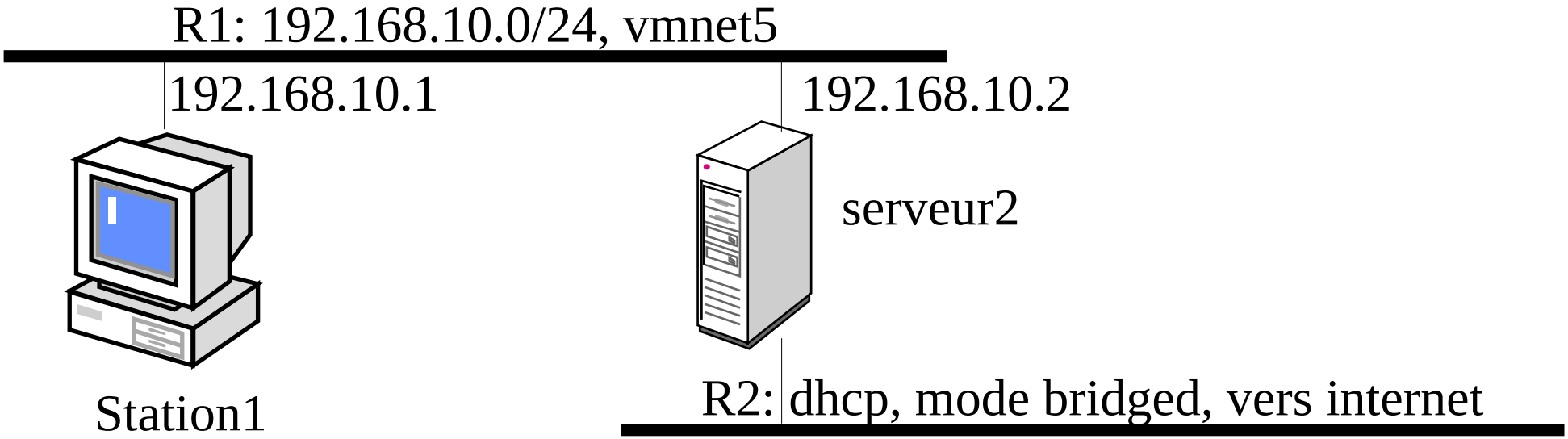
DNS: utiliser un dns local en cache

- on utilise un dns que l'on a installé chez soit



TD: installer un DNS cache

- réaliser la maquette suivante :
  - installer un logiciel serveur dns ne gérant aucune zone sur le serveur
  - la station de travail utilise le serveur comme serveur dns



DNS: avoir son propre domaine

- pourquoi:
  - pérennité: l'affichage est indépendant de la partie technique
    - pouvoir changer d'hébergeur de site WeB sans changer d'adresse
    - pouvoir changer de gestionnaire d'adresse mail sans changer d'adresse mèl
  - afficher une identité
  - simplifier la mémorisation
  - regrouper sur un même noms des ressources gérées de façon éparses

DNS: avoir son propre domaine

- Les adresses IP dépendent du fournisseur d'accès
  - Changer de fournisseur => changer d'adresses
- Un domaine ne dépend pas du fournisseur d'accès (ni de l'entité qui l'a « fourni »)
  - On peut changer de fournisseur et conserver ses domaines

DNS: avoir son propre domaine

- comment
  - en achetant son domaine à un « registrar » (bureau d'enregistrement)
    - le domaine peut être enregistré à n'importe quel bureau d'enregistrement gérant le TLD
    - on peut transférer un domaine d'un registrar à un autre
    - la faillite du registrar ne met donc pas en péril le domaine acheté
  - 2 solutions techniques ensuite:
    - gérer soit même ses propres dns, serveur mail, WeB (cas standard)
    - laisser le « registrar » gérer des redirections (offre en plus)
      - redirection pour le WeB: www.mondomaine.fr -> www.herbergeur.com/moi
      - redirection mail: toto@mondomaine.fr -> moi@isp.fr
  - Exemples de registar: www.gandi.net

TD:

- aller sur www.gandi.net
- voir les services offerts
- vérifier la disponibilité d'un domaine de votre choix

Introduction (4)

- Jusqu'en 1984 : fichier hosts.txt /etc/hosts
  - inadapté à grande échelle
    - temps de diffusion des infos (par ftp !)
    - système centralisé
    - quelques centaines de machines dans les années 70
    - plusieurs millions aujourd'hui
  - correspondance statique
  - ne contient que des infos réduites
  - noms enregistrés sous le domaine `arpa`
    ⇒ collision rapide de noms

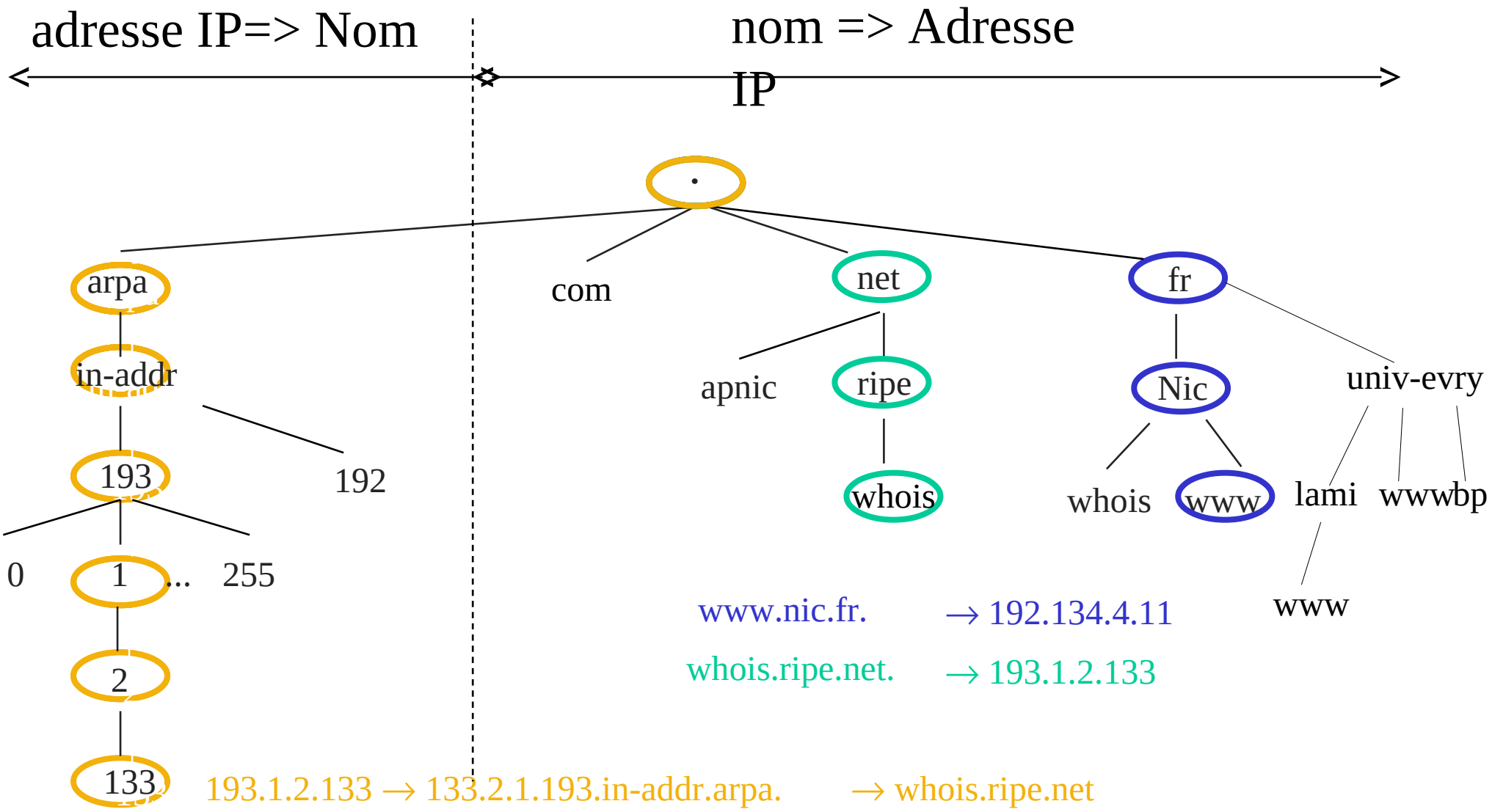
Introduction (5)

- Après 1984 : **D**omain **N**ame **S**ystem
  Paul Mockapetris - RFC 882 883 puis 1034 1035
  - système hiérarchisé et distribué
    - modèle en arborescence (similaire à l'arborescence d'un système de fichiers avec ses répertoires)
    - gestion décentralisée des bases de données
  - chaque site est maître de ses données
  - informations complémentaires : relais de messagerie, ...
  - correspondance dynamique
  - limite les risques de collisions de noms

Arborescence (1)

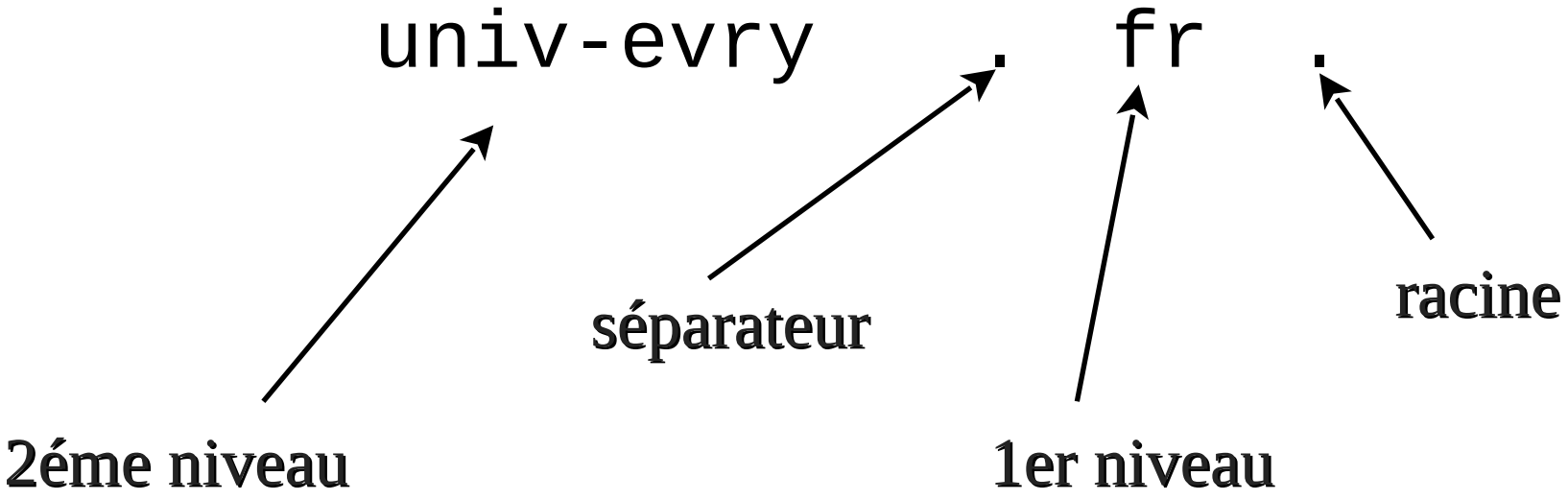
- Organisation générale
  - le système est organisé sous la forme d'une arborescence, composée par
    - la racine (`root`), sommet de l'arbre, qui est notée par un point « `.` »
    - des noeuds, identifiés par un label (`fr` , `com` , ...), dont les informations sont stockées dans une base de données propre à chacun des noeuds
  - base de données du système
    - **une** base de données par noeud
    - l'ensemble de ces bases de données constitue le système d'information hiérarchique et distribué du DNS

Arborescence (2)

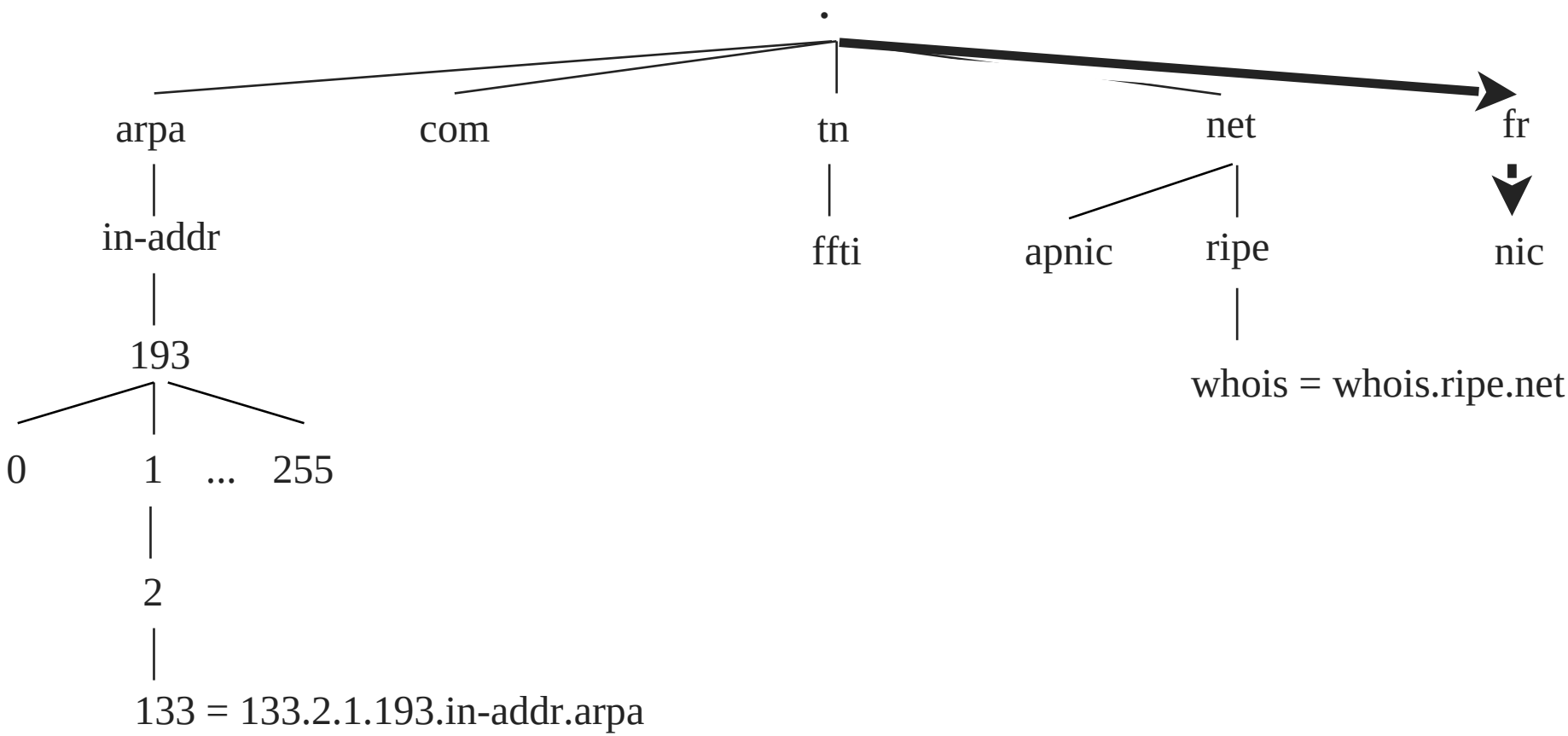


Arborescence (3)

- Parcours de l'arbre et nom de domaine
  - on va du général au particulier de droite à gauche
  - la descente dans l'arbre est représentée de la droite vers la gauche
  - chaque niveau de l'arborescence est séparé par un point



Arborescence (4)



Arborescence (5)

- Délégation d'un noeud père vers un noeud fils
  - un noeud peut être père de plusieurs noeuds fils
  - le lien est effectué en précisant au niveau du noeud père où trouver la base de donnée des noeuds fils
  - but
    - distribuer la gestion de chaque noeud à des entités différentes => une base de données pour chaque noeud, l'ensemble de ces bases étant géré de façon décentralisé
    - pour définir des domaines de responsabilités différentes
- créer un domaine:
  - action au niveau du domaine père
  - exemple: créer shayol.org.: action sur org.

Arborescence (6)

- Dénomination des domaines
  - caractères autorisés 'A...Z' 'a...z' '0...9' '-'
    pas de différences entre majuscule et minuscule
  - premier caractère: lettre ou chiffre (rfc 1123)
  - nom total limité à 255 caractères
  - label est unique au niveau d'un noeud
  - label au niveau d 'un noeud limité à 63 caractères

Arborescence (7)

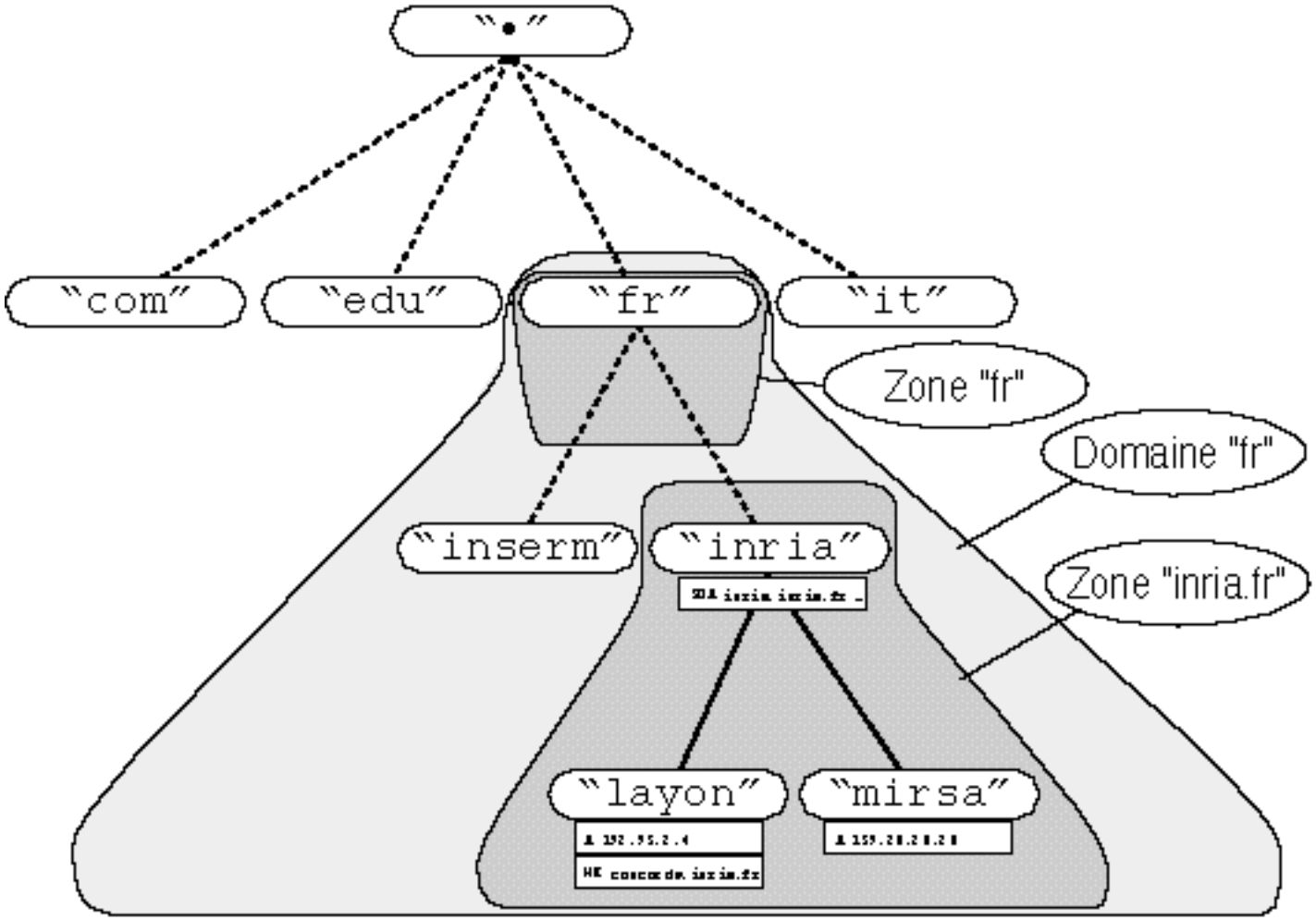
- IDN - International Domain Name
  - Prendre en compte les alphabets mondiaux
  - Compatible avec le protocole DNS existant
- 4 RFC
  - RFC 3454/RFC 3491/RFC 3490/RFC 3492
- Exemple:

```
http://france2.télévision-française.fr/
□
http://france2.xn-tlvision-franaise-msbzb.fr/
```

Arborescence (8)

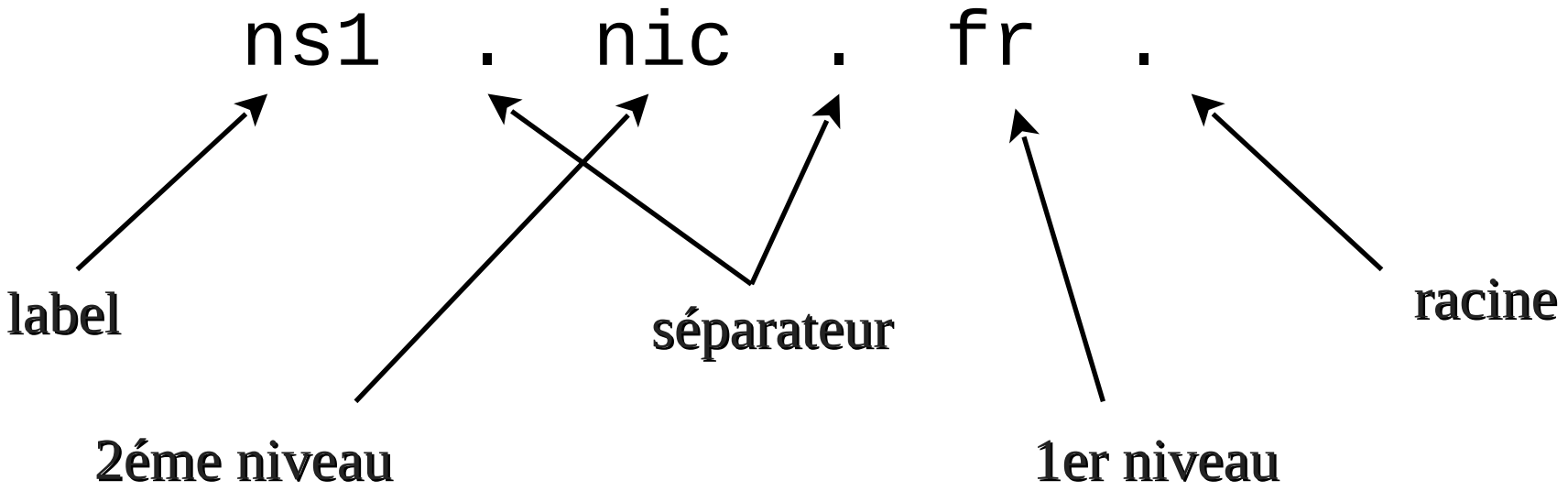
- Notion de domaine et de zone
  - le domaine est l'ensemble d'une sous arborescence
    exemple : le domaine univ-evry.fr . rassemble toute la sous arborescence à partir du noeud univ-evry.fr .
  - la zone est la partie décrite par la base de données d'un noeud.
    => un domaine contient :
    - la zone correspondante
    - les éléments délégués

Arborescence (9)



Arborescence (10)

- Résolution nom => numéro IP
  - le nom de machine est formé en ajoutant le label choisi suffixé avec « . » avec le domaine auquel cette machine appartient



Arborescence (12)

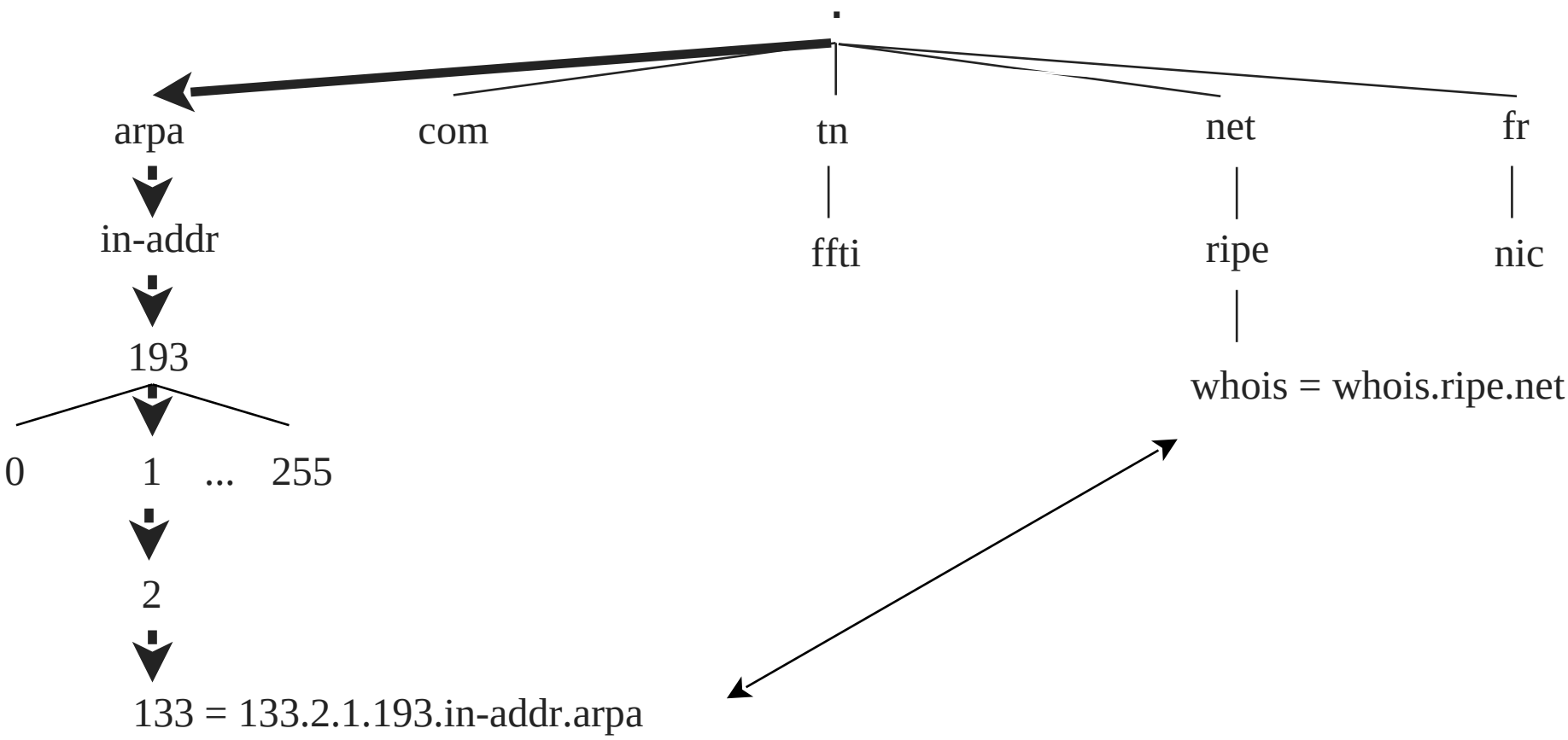
- Résolution inverse: numéro IP => nom
  - retrouver à partir d'un numéro IP le nom d'une machine associée
  - l'arborescence se trouve sous le domaine in-addr.arpa (sous ip6.int pour ipv6)
  - l'arborescence est subdivisée à partir de la notation classique sur 4 octets des numéros IPv4

Arborescence (13)

- Parcours de l'arbre et résolution inverse
  - Problème:
    - noms : le général à droite, le particulier à gauche
    - adresses IP: le général à gauche, le particulier à droite
  - Solution : le nom de domaine est inversé par apport au numéro IP

    domaine : 133.2.1.193.in-addr.arpa.
    pour le numéro IP : 193.1.2.133
  - la descente dans l'arbre est représentée de la droite vers la gauche
  - chaque niveau de l'arborescence est séparé par un point

Arborescence (14)

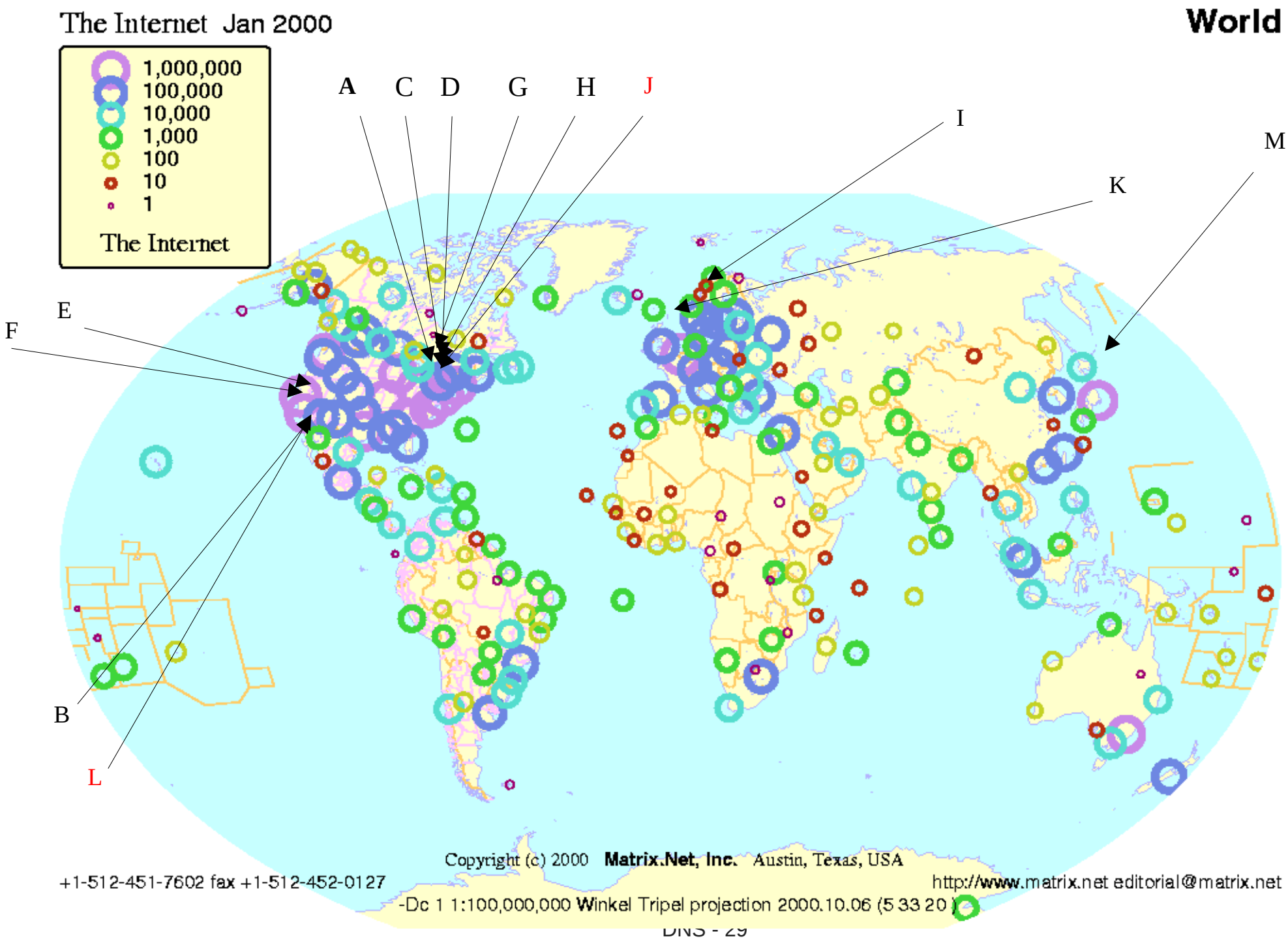


Arborescence (15)

- Le même mécanisme s'applique pour la sous arborescence in-addr.arpa comme pour les domaines « classiques » (nic.fr) : par exemple le domaine 11.193.in-addr.arpa est un sous domaine du 193.in-addr.arpa, le noeud 11.193.in-addr.arpa étant défini par sa base de données
- Tout numéro officiellement attribué à une machine doit être déclaré dans cette arborescence

Arborescence (16)

- Racine : 13 bases de données (serveurs de nom) répartis dans le monde connaissant tous les serveurs des domaines de 1er niveau (`.fr .arpa .com ...`)
  - serveur origine géré par l 'IANA / ICANN
    `A.ROOT-SERVERS.NET`
  - serveurs miroirs
    de
    `B.ROOT-SERVERS.NET`
    à
    `M.ROOT-SERVERS.NET`


The Internet Jan 2000
World
1,000,000
100,000
10,000
1,000
100
10
1
The Internet
A
C
D
G
H
J
I
M
K
E
F
B
L
Copyright (c) 2000 Matrix.Net, Inc. Austin, Texas, USA
+1-512-451-7602 fax +1-512-452-0127
http://www.matrix.net editorial@matrix.net
-Dc 1 1:100,000,000 Winkel Tripel projection 2000.10.06 (5 33 20)

Attaque d'octobre 2002: DDOS

- Les 21 et 22 octobre 2002, une attaque contre les serveurs dns racine a eu lieu
- Sans serveur racines, seules les requètes s'appuyant sur des informations en cache aboutiront
- DDOS: déni de service distribué
  - Grâce à un parc énorme de machine piratées
  - A généré 40 fois le trafic normalement géré par les serveurs racine
  - Seuls 7 serveurs sur 13 ont été paramysés
- Cette attaque a entraîné l'utilisation d'anycast

RFC 3258: dns et anycast

- Anycast : mécanisme s'appuyant sur le routage pour permettre à une même adresse ip de correspondre à plusieurs machines
- En 2008, Anycast est utilisé
  - pour les serveurs racines C,F, I, J, K, L et M (on a donc maintenant une majorité de serveurs hors USA)
  - pour la gestion des zones .fr et .ch.
- Avantages :
  - Répartition de charge, Tolérance de panne
  - Diminution du trafic (la cible choisie est la plus proche du point de vue de l'algo. de routage)

RFC 3258: dns et anycast

- Défaut:
  - Le choix de la destination réelle se fait paquet par paquet
  - En cas d'égalité de poids de route, les paquets peuvent atteindre des serveurs différents
  - Aucun problème en UDP (non connecté)
  - De gros problèmes en TCP

RFC 3258: dns et anycast

- Résolution du problème:
  - Le trafic dns est quasi exclusivement constitué de trafic UDP
  - Le cas où deux serveurs réels correspondant à la même IP sont à la même distance d'un client est un cas rare voire inexistant si le placement des serveurs est fait correctement
  - Pour l'éviter on fera en sorte qu'un dns de la zone soit associé à une autre ip
    - Ainsi, toute implémentation du dns qui analyse les performances des serveurs dns et se trouvant dans le cas posant problème finira par préférer le serveur le plus fiable qui est celui n'utilisant anycast.

Arborescence (17)

- Top-level domain (TLD) : Domaine de 1er niveau - RFC 1591
  - à 2 lettres : code ISO-3166 de chaque pays
  - à 3 lettres : `.com`, `.net`, `.org`, `.edu`, `.gov`, `.mil`, `.int`
  - à 4 lettres : `.arpa`
- De nouveaux TLD sont apparus en 2001 :
  - `.biz, .info, .name, .aero, .coop, .museum`
  - Le `.sex` un temps envisagé semble ne pas devoir exister :-)

Architecture (1)

- Système client/serveur
  - client
    - resolver : interface cliente permettant d'interroger un serveur
    - les machines clientes pointent généralement vers un serveur par défaut (/etc/resolv.conf sur Unix, dns du panneau de configuration IP sous windows)
    - Le resolver d'une machine serveur dns est, en général, paramètré pour utiliser le serveur dns local
  - serveur
    - chaque serveur gère sa propre base de données
    - optimisation par des systèmes de cache et de réplication

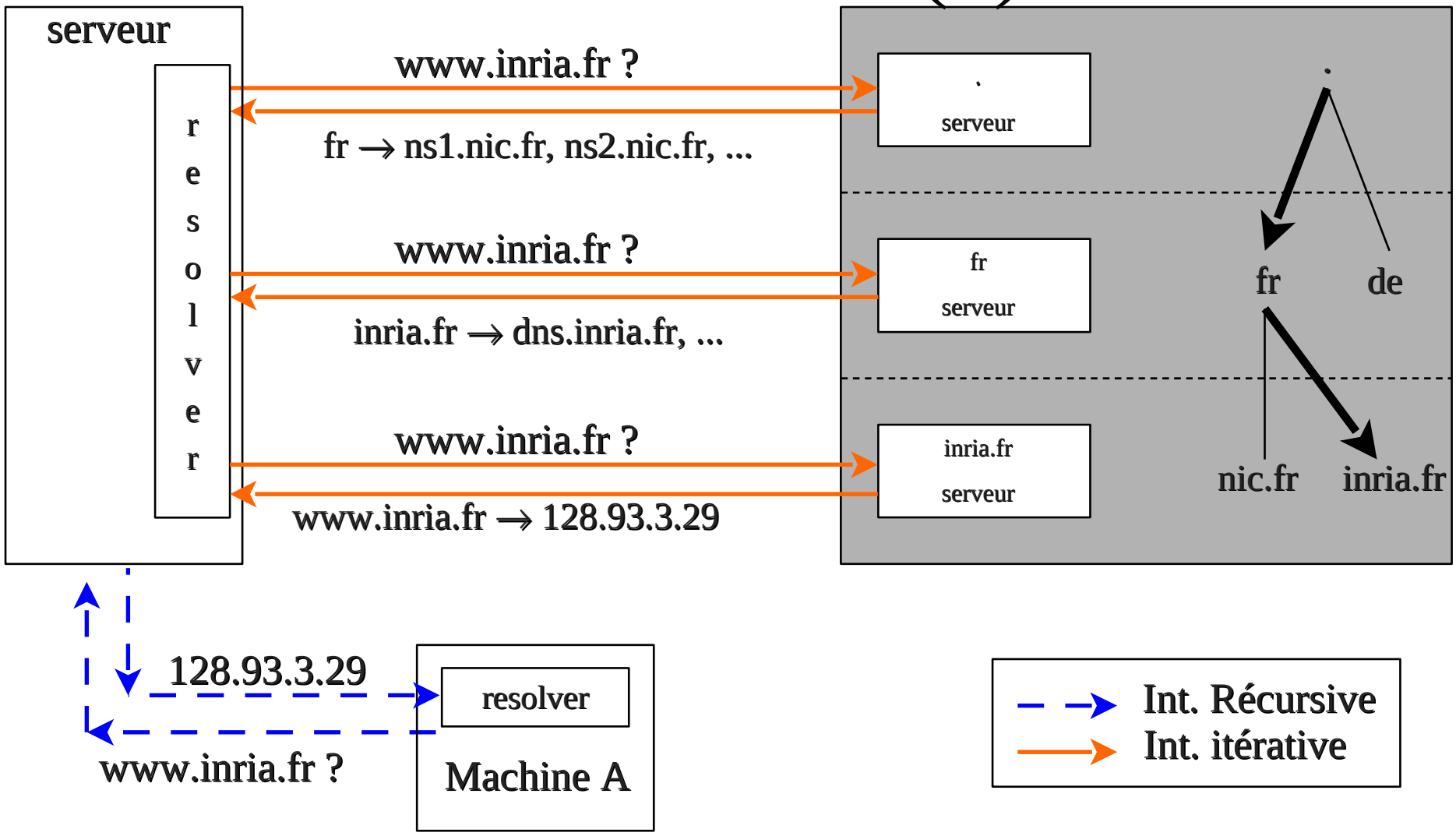
Architecture (2)

- Au-dessus d 'IP
  - service s'exécutant sur le port 53
    droits de super utilisateur (unix)
  - UDP et TCP
    (TCP n'est pas réservé qu'au transfert de zone
    et est utilisé si la taille de la réponse est supérieure à la limite d 'un paquet UDP de 512 octets)
- RFC 1035

Architecture (3)

- Fonctionnement du client : le resolver
  - permet de communiquer avec les serveurs DNS
  - 2 modes d'interrogation
    - récursif : le client envoie une requête à un serveur, ce dernier devant interroger tous les autres serveurs nécessaires pour renvoyer la réponse complète au client (mode utilisé par les machines clientes en général)
    - itératif : le client envoie une requête à un serveur, ce dernier renvoyant la réponse si il la connaît, ou le nom d'un autre serveur qu'il suppose plus renseigné pour résoudre cette question (mode utilisé par le resolver des serveurs en général)

Architecture (4)



Architecture

- Serveur autoritaire :contient les informations fournie par le propriétaire d'une zone sur la zone
- types de serveurs autoritaires d'une zone :
  - serveur primaire: accès en lecture/écriture à la BD
  - serveurs secondaires: accès en lecture seule à la BD
  - Serveur secondaire esclave: récupère ses données d'un serveur maître
  - Serveur primaire ou secondaire maître: fourni ses données à un ou plusieurs esclaves
  - Un serveur secondaire peut être esclave (pour récupérer les données) et maître vis à vis d'autres serveurs (pour leur fournir les données récupérées)

Architecture

- Tout serveur doit supporter le mode de fonctionnement maître/esclave
- Rien n'empeche de fournir mieux
  - w2k le fait pour les zones intégrées à Active Directory) :
  - Tous les serveurs ont un accès en lecture/écriture aux zones
- Plusieurs serveurs dns autoritaires :
  - Pour la tolérance de panne
  - Pour la répartition de charge

Architecture (15)

- Rafraîchissement des données entres serveurs autoritaires
  - système classique
    - un serveur secondaire d'une zone interroge à intervalles réguliers le serveur primaire de cette zone pour voir si une modification a eu lieu. La fréquence est indiquée par la valeur `refresh` défini dans le `SOA`
    - ce type de mise à jour peut se faire entre un primaire et un secondaire ou entre secondaires

Architecture (16)

- Rafraîchissement des données entre serveurs autoritaires
  - la version d 'une zone est identifiée par son numéro de série (`serial`) ; à chaque modification celui-ci **doit** être augmenté
  - le transfert de zone
    - le serveur secondaire transfère d'abord le SOA de la zone et vérifie si le numéro de série a augmenté
    - si c'est le cas toute la zone est transférée (transfert total - AXFR) ou seules les nouvelles modifications entre les 2 versions sont transférées (transfert incrémental - IXFR - RFC 1995)

Architecture (17)

- Rafraîchissement des données entre serveurs autoritaires
  - reprise en cas d 'échec
    - en cas d 'échec de cette interrogation, le secondaire recommence toutes les `retry` secondes jusqu'à atteindre le temps d'expiration (`expire`), ces valeurs étant fixés également dans le `SOA`

Architecture (18)

- Rafraîchissement des données entres serveurs ayant autorités sur une zone
  - DNS Change Notification (RFC 1996)
    - après la prise en compte de modifications de la part du serveur primaire, ce dernier notifie les serveurs secondaires qu'une nouvelle version de la zone a été générée
    - ce système de mise à jour ne peut se faire qu'entre un primaire et un secondaire
    - accélère le rafraîchissement des données par rapport au système classique

Architecture (19)

- Remarques
  - un serveur peut être à la fois serveur cache et autoritaire pour des zones : le cache possède alors des informations locales et non locales
  - un serveur peut être à la fois primaire pour des zones et secondaire pour d'autres zones

Architecture (20)

- Remarques
  - mode de fonctionnement d'un serveur
    - récursif
      - le serveur résout les requêtes récursives des clients et garde les informations obtenues dans son cache
      - □ le cache stocke des informations pour lesquelles le serveur n 'a pas autorité
      - serveurs cache de campus par exemple

Architecture (21)

- Remarques
  - mode de fonctionnement d'un serveur
    - itératif
      - il répond toujours en fonction des données qu'il possède localement
      - ⇒ ne construit pas de cache pour des données non locales .
      - ⇒ une machine cliente (d'utilisateur final) ne doit jamais pointer sur un serveur de ce type comme serveur par défaut .
      - mode permettant de limiter la charge d'un serveur (il ne résout pas toute la requête)
        ⇒ serveurs de la racine, serveurs ayant autorité pour un grand nombre de zones

configuration pratique d'un serveur

- Un serveur dns
  - peut gérer zéro, une ou plusieurs zones
  - peut être primaire pour certaines zones et secondaires pour d'autres
- Les serveurs dns d'une zone
  - peuvent être situés n'importe où,
  - Peuvent avoir des ip/nom qui ne sont pas dans la zone gérée
- Mise en service: 2 étapes
  - Créer la zone sur le serveur dns, configurer ses paramètres
  - Saisir, mettre à jour ses données (RR)

configuration pratique d'un serveur

- Configurer un logiciel serveur dns :
  - Préciser de quels clients il accepte les requêtes récursives
  - Préciser le comportement à adopter pour les requêtes concernant des zones qu'il ne gère pas (interrogation directe d'internet, passage par un redirecteur (serveur dns en cascade), ...
  - Définir les zones hébergées par le serveur et leur type
    - De nombreux paramètres se décident zone par zone :
      - Type de zone : Primaire, secondaire, stub, forward, ...
      - Serveurs esclaves
    - Un serveur peut être secondaire pour certaines zones, primaire pour d'autres, ...
    - Saisir les données des zones

Configuration: types de zones

- L'entité de base est la zone
- Certaines options (redirecteur, ...) peuvent s'appliquer au serveur entier ou à des zones particulières
- Types de zones:
  - Zone primaire :
    - accès lecture/écriture aux données
    - Le serveur est autoritaire pour la zone
  - Zone secondaire:
    - Récupérée depuis un serveur maître
    - Accès en lecture seule à la zone
    - Le serveur est autoritaire pour la zone

Configuration: types de zones

- Zone en redirection (forward)
  - Permet de définir un serveur dns à interroger en cas de requêtes sur la zone
  - Permet de spécifier le rôle de redirecteur zone par zone et pas globalement pour tout le serveur
  - Notion spécifique à bind
  - À partir de W2K3+: possibilité de définir des redirections conditionnelles qui fournissent des fonctionnalités similaires au serveur DNS microsoft
- Zone stub:
  - Similaire à une zone esclave mais ne réplique que les enregistrements NS de la zone. Permet d'éviter d'avoir à rechercher ces NS via des requêtes sur internet.

Configuration: création d'une zone avec bind:

- Dans le fichier de configuration de bind named.conf (ou dans un fichier importé. Par ex. named.conf.local sur une distribution linux debian etch
- Une entrée par zone de la forme:

```
zone "ibisc.fr" {
        type master;
        file "pz/ibisc.fr";
};
```

Configuration: création d'une zone avec le serveur dns W2K3:

- La console mmc de gestion du DNS présente:
  - Les zones directes
  - Les zones de recherche inversée
- Créer une zone directe: clic droit sur zone directe -> nouvelle zone
- Créer une zone inversée: clic droit sur zone inversée -> nouvelle zone
- Un assistant permet ensuite de saisir les paramètres de la zone

W2k3+: paramètres de la zone

- Cette section sera complétée dans la prochaine version de ce document (ràf)

Base de données (12)

- Principaux RR - SOA (Start Of Autority)

  caractéristiques techniques de la zone :

```
zone    IN    SOA  primaire.  email.    (
                                 serial
                                 refresh
                                 retry
                                 expire
                                 ttl                 )
```

Base de données (13)

- Principaux RR - SOA

email : contact technique de la zone

remplacer le @ par le premier point non protégé (\)

l'email doit être suivi d 'un point

francis\.dupont.inria.fr.

pour

francis.dupont@inria.fr

Base de données (14)

- Principaux RR - SOA

  numéro de série : spécifie la version des données de la zone

  incrémenter ce numéro à chaque modification (entier sur 32 bits)

  format conseillé : `YYYYMMDDxx`

  `1997052702`

Base de données (15)

- Principaux RR - SOA

  refresh : intervalle, en secondes, entre 2 vérifications du numéro de série par les secondaires (24H - 86400s ; à ajuster si la zone est souvent modifiée)

  retry : intervalle en seconde entre 2 vérifications du numéro de série par les secondaires si la 1ere vérification a échoué (6H - 21400s ; à ajuster en fonction de sa connectivité)

  expire : durée d 'expiration de la zone sur un secondaire (41 jours -3600000s)

  retry<<refresh<<expire

Base de données (16)

- Principaux RR - SOA

  ttl (time to live) - RFC 2308 - Negative caching

  spécifie le TTL pour le « negative caching », soit le temps que doit rester dans les caches une réponse négative suite à une question sur ce domaine (valeur recommandée de 1 à 3 heure).
  Il existe 2 types de réponses négatives :

  - NXDOMAIN : aucun enregistrements ayant le nom demandé dans la classe (IN) n'existe dans cette zone
  - NODATA : aucune donnée pour le triplet (nom, type, classe) demandé n'existe ; il existe d'autre enregistrements possédant ce nom, mais de type différent

Base de données (17)

- Principaux RR - NS (Name Server)

  indique un serveur de nom pour le nom spécifié (ce nom devient une zone dont la délégation est donnée au serveur en partie droite)

```
zone    IN    NS    serveur-nom1.domaine.
        IN    NS    serveur-nom2.domaine.
```

  Il faut spécifier les serveurs de noms de la zone que l'on décrit (associée au SOA)

  exemple: host -t NS univ-evry.fr

Base de données (18)

```
$ORIGIN     nic.fr.
$TTL        86400
@           IN    SOA   ns1.nic.fr.
                        hostmaster.nic.fr. (
                        1997052704  ;serial
                        86400       ;refresh
                        21600       ;retry
                        3600000     ;expire
                        3600  ; negative caching ttl  )
            IN    NS          ns1.nic.fr.
            IN    NS          ns2.nic.fr.
www         IN    CNAME ns2.nic.fr.
ftp  21600  IN    A           192.34.4.45
```

Base de données (19)

- Principaux RR - A (Adresse IPv4)

  indique l 'adresse IP associée à un nom

  machine.domaine. IN A 193.10.20.30

  AAAA : Adresse IPv6

Base de données (20)

- Principaux RR - PTR (Pointeur) : entrée dans la zone inversée

  indique le nom associé à un numéro IP dans l 'arborescence in-addr.arpa (ip6.arpa)

  10.20.30.192.in-addr.arpa. IN PTR machine.domaine.

Base de données (21)

- Principaux RR - CNAME (Canonical Name)

  indique que le nom est un alias vers un autre nom (le nom canonique)

  alias IN CNAME nom.canonique.

  Nota

  - un nom en partie droite d'un enregistrement (<données>) ne doit pas pointer vers un alias
  - quand un nom a déjà un CNAME il est interdit de faire figurer d'autres enregistrements pour ce nom

Base de données (22)

- Principaux RR - CNAME (Canonical Name) - faux

```
zone.          IN    MX    10    alias.zone.
alias.zone.    IN    CNAME       relais.zone.

zone.          IN    NS          alias.zone.
alias.zone.    IN    CNAME       serveur.zone.

zone.          IN    SOA         xxxx. yyy. ( zzzz ... )
zone.          IN    CNAME       www.zone.
```

Base de données (23)

- Principaux RR - CNAME (Canonical Name) - correct

```
zone.          IN    MX    10    relais.zone.
relais.zone.   IN    A           193.1.2.3

alias2.zone.   IN    CNAME       alias1.zone.
alias1.zone.   IN    CNAME       canonical.zone.

zone.          IN    SOA         xxxx. yyy. ( zzzz ... )
zone.          IN    A           193.2.3.4
```

Et l'enregistrement PTR correspondant dans les reverses

DNS et SMTP (1)

- Principaux RR - MX (Mail eXchanger)

  email à quelqu-un@nom (ex.: petit@shayol.org)

  On cherche dans le DNS un MX indiquant la machine sur laquelle il faut envoyer le courrier pour `nom`.

  Un paramètre précise le poids relatif de l 'enregistrement MX : si plusieurs MX existent, le courrier est envoyé en 1er à la machine ayant le poids le plus bas, puis dans l 'ordre croissant des poids en cas d'échec

nom	IN	MX	10	nom.relais1.
	IN	MX	20	nom.relais2.
	IN	MX	30	nom.relais3.

Exemple: host -t MX free.fr

DNS et SMTP (2)

- Principaux RR - MX (Mail eXchanger)

  Envoi d 'un message à `nom - RFC 974`

  - (1) tri les MX par ordre croissant et contacte les machines dans cet ordre ; si une connexion est établie transfert ; sinon mail mis en file d'attente
  - (2) transfert sur `nom.relais1` : le mail est traité localement
  - (3) transfert sur l'une des autres machines: on trie de nouveau les MX en supprimant les entrées de préférence supérieure on égale à celle associée à cette machine ; si la liste est vide erreur de configuration ; sinon on tente de contacter les machines de la même manière qu'en (1)

DNS et SMTP (3)

- Principaux RR - MX (Mail eXchanger)
  - wilcard MX

    nic.fr. IN MX 10 relais.nic.fr.
    *.nic.fr. IN MX 10 relais.nic.fr.

    - associe le MX à tout nom inconnu dans le domaine, **il n'est utilisé qu'en l'absence de tout autre RR associé à un nom.**

    Exemple :
    `nom.nic.fr. IN A IP`

    - pas de MX hérité des wilcards pour `nom`
    - associer systématiquement un MX à chaque fois que l 'on définit un A RR et éviter les wilcards

DNS et SMTP (4)

- Principaux RR - MX (Mail eXchanger)

  Si il n 'y a pas de MX associé à nom :

  - SMTP utilise l 'adresse IP associé à ce nom (A RR)
    nom IN A IP
  - si il n 'y a pas de RR, SMTP utilise les enregistrement wildcard MX
  - si il n 'y a pas de wildcard MX => erreur

Base de données (PP1)

- Principaux RR - SRV (localisateur de service)

indique les serveurs proposant un service donné

Syntaxe :

service.protocole.nom ttl classe SRV préférence poids port cible

Exemples :

_ldap._tcp.ms-dcs SRV 0 0 389 dc1.exemple.microsoft.com

très utilisé par windows 2k+, notamment pour la localisation des contrôleurs de domaine

Base de données (PP1)

- Principaux RR - SRV (localisateur de service)

indique les serveurs proposant un service donné

Syntaxe :

service.protocole.nom ttl classe SRV préférence poids port cible

Exemples :

_ldap._tcp.ms-dcs SRV 0 0 389 dc1.exemple.microsoft.com

très utilisé par windows 2k+, notamment pour la localisation des contrôleurs de domaine