Ecole Centrale de Lyon – 3A – MOD 5.7
René CHALON

Réseaux informatiques
~
Applications sur TCP/IP

1- Introduction aux couches hautes
2- Le modèle client-serveur
3- le DNS
4- la messagerie électronique
5- FTP
6- le World Wide Web

Version du 05/11/2012

Couches hautes

- Rappel:
  2 grandes catégories de couches:
  - Couches basses: 1, 2, 3, 4
    - ➔ Gèrent les aspects de l'accès au médium physique de transmission et des règles de communication (couches 1 et 2)
    - ➔ S'occupent de la communication de bout en bout entre les machines en assurant la fiabilité (couches 3 et 4)
  - Couches hautes: 5, 6, 7
    - ➔ Règlent les problèmes d'hétérogénéité et de supervision de l'échange de données entre applications (couche 5 et 6)
    - ➔ Supportent la mise en œuvre d'applications déterminées (couche 7)

7	**Application**
6	**Présentation**
5	**Session**
4	Transport
3	*Réseau*
2	*Liaison de données*
1	*Physique*

Couche Session (couche 5)

- Fonction dans le modèle OSI :
  - assurer un dialogue ordonné entre les applications
- Gestion et synchronisation des échanges:
  - la session est découpée en unités de dialogue
  - permet la reprise du dialogue en cas de rupture de la connexion (grâce à des points de synchronisation)
  - possibilité de coordonner une activité sur plusieurs sessions
- Dans le modèle TCP/IP :
  - Pas de couche session
  - Les RPC sont cependant assimilables à un service de couche session mais ne sont pas systématiquement utilisés
  - La notion de session est introduite dans le protocole HTTP 1.1

Couche Présentation (couche 6)

- Fonctions dans le modèle OSI :
  - Échange de données structurées entre applications sur des systèmes hétérogènes:
    - représentation commune des données
    - permet aux applications de négocier une ou plusieurs syntaxes de transferts (contextes de présentation)
  - Services complémentaires (optionnels):
    - compression des données
    - chiffrement des données
  - Syntaxe de transfert ASN.1 [Abstract Syntax Notation 1] :
    - langage permettant de spécifier l'ensemble des données échangées entre applications
- Dans le modèle TCP/IP :
  - Pas de couche présentation explicite
  - XDR est assimilable à ASN.1 mais utilisé seulement par les RPC

Couche Application (couche 7)

- Fonction dans le modèle OSI :
  - assure la sémantique (ou signification) des informations à échanger entre les applications
- Structure complexe:
  - Association d'éléments de services d'application (ASE) :
    - éléments de services communs (ACSE, RTSE, ROSE, CCRSE)
    - éléments de services spécifiques (X400, FTAM, DTP, X500, etc.)
- Dans le modèle TCP/IP :
  - Aucune structure *a priori,* chaque application se « débrouille »
  - Evolution vers un modèle structuré autour du Web construit sur le protocole HTTP et le langage XML

Couche application du modèle TCP/IP

- Structure de la couche application
  - Par comparaison au modèle OSI:
    - pas de structure plus détaillée
    - or, elle couvre l'équivalent des couches 5, 6 et 7 du modèle OSI !
  - Evolution vers un modèle structuré autour du Web construit sur le protocole HTTP et le langage XML
- Très grande diversité d'applications:
  - Applications classiques:
    - messagerie, transfert de fichiers, etc…
  - De très nombreuses applications:
    - beaucoup d'applications expérimentales souvent concurrentes et ayant des fonctions similaires
    - état "normal": les standards sont proposés (statut= elective) et les utilisateurs choisissent ce qui leur semble le meilleur, les autres tombant dans l'oubli

Principales applications (1/2)

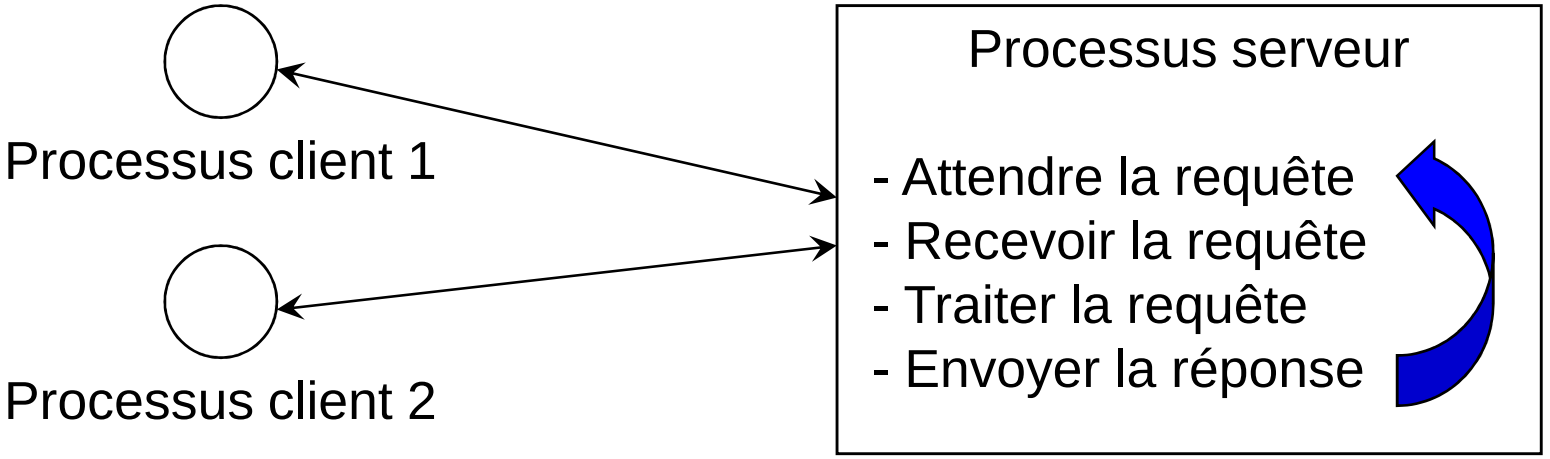
Nom	Norme	Fonction
SMTP	RFC 2821,2822	"Simple Mail Transfer Protocol" Service de messagerie électronique
DNS	RFC 1032 à 1035	"Domain Name System" Service d'annuaire (noms de domaines)
FTP	RFC 959	"File Transfer Protocol" Service de transfert de fichiers
TFTP	RFC 783	"Trivial File Transfer Protocol" Service de transfert de fichiers simplet
NFS	RFC 1813	"Network File System" Service de partage de fichiers
Telnet	RFC 854	Telnet Service de terminal à distance

Principales applications (2/2)

Nom	Norme	Fonction
HTTP (Web)	RFC 2616	"HyperText Transfer Protocol" Le World Wide Web
SNMP	RFC 1441,1452	"Simple Network Management Protocol" Administration de réseaux
MIME	RFC 2045 à 2049	"Multipurpose Internet Mail Extensions" Représentation de données en vue de leur transfert (par E-mail, par Web)
HTML		"HyperText Markup Language" Langage de représentation hypermédia
XML		"Extended Markup Language" Langage de représentation de données

Le modèle client-serveur

- Les différentes applications de la famille TCP/IP s'appuient sur le modèle client-serveur
- Deux catégories de processus d'application:
  - les processus serveurs qui fournissent des services,
  - les processus clients qui font des requêtes aux serveurs qui leur renvoient les réponses

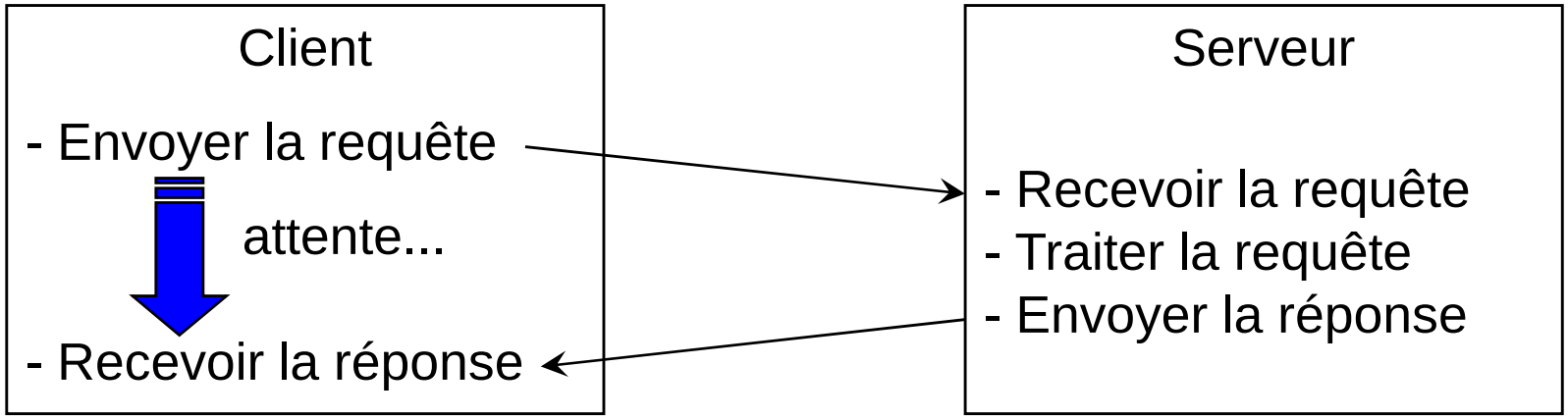


Clients et serveurs

- Les processus clients et serveurs sont des programmes
  - il s'exécutent sur des ordinateurs
  - un ordinateur peut exécuter plusieurs processus serveurs et/ou processus clients en même temps (systèmes multitâches)
  - si un ordinateur est plus particulièrement chargé de l'exécution d'un processus serveur particulier, alors on appelle l'ordinateur lui-même un "serveur": *"l'ordinateur A est notre serveur Web"*
- Processus serveur (ou serveur)
  - doit être démarré avant les clients
  - accepte des requêtes et renvoie des réponses
  - il ne s'arrête jamais (en principe)
- Processus client (ou client)
  - il est exécuté au moment où l'utilisateur en a besoin
  - après avoir fait une ou plusieurs requêtes, il se termine (en général)

Communication entre client et serveur

- La communication peut être du type synchrone:
  - le client est bloqué tant qu'il n'a pas reçu la réponse du serveur



- Dans ce modèle, le serveur traite une requête à la fois:
  - pour éviter la perte des requêtes venant d'autres clients:
    - ➔ utilisation d'une file d'attente des messages côté serveur
  - pour optimiser le temps de réponse:
    - ➔ un processus attend les requêtes ; quand une requête arrive un nouveau processus est lancé et traite la requête (il faut un serveur multitâche)

Notion de "port réservé"

- Port réservé ou Well-Known Port (WKP):
  - Pour chaque service, un n° de port est réservé (< 1024)
  - Un serveur "écoute" en permanence sur ce port réservé
  - Peut-être comparé à une adresse de niveau application
  - Pour éviter toute confusion, les clients utilisent des ports >= 1024
- Principaux ports réservés:

Protocole application	N° port réservé	Protocole transport
ECHO	7	UDP (TCP)
FTP données FTP contrôle	20 21	TCP TCP
TELNET	23	TCP
SMTP (E-mail)	25	TCP
DNS	53	UDP (TCP)

Protocole application	N° port réservé	Protocole transport
TFTP	69	UDP
HTTP (Web)	80	TCP
NNTP (news)	119	TCP
NTP (horloge)	123	UDP
SNMP get, put SNMP trap	161 162	UDP UDP

Exemple : le serveur d'écho

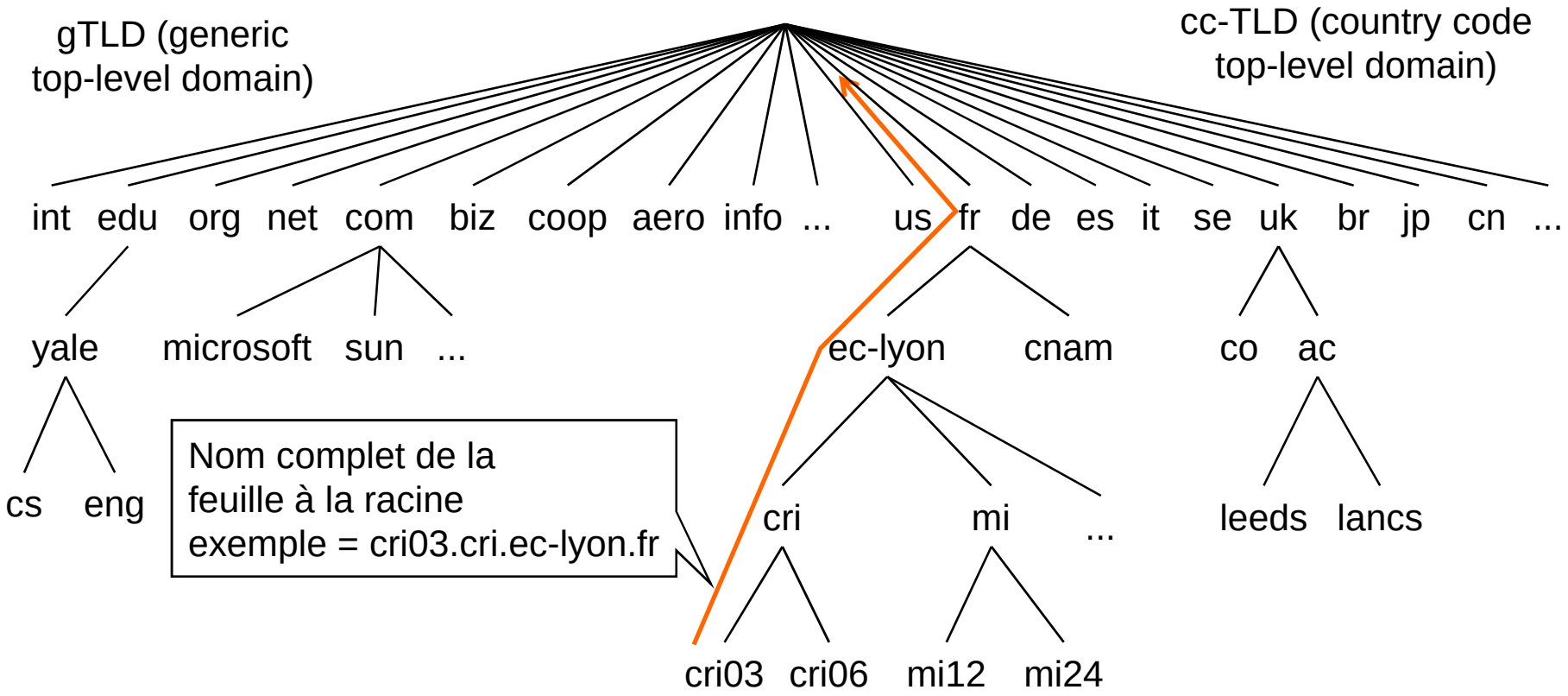
- Service d'echo (RFC 862) :
  - le serveur écoute sur le port UDP 7 (existe aussi sur TCP)
  - les clients envoient des paquets de données en prenant un port UDP libre (>1024)
  - A chaque fois que le serveur voit arriver un paquet d'un client:
    - il échange les adresses IP source et destination
    - il échange les n° de ports source et destination
    - il lui retourne les données inchangées (d'où le nom d'écho)
- Intérêt:
  - permet de tester la connectivité entre 2 machines et la pile de protocole TCP/IP
- Remarque :
  - Ne pas confondre avec les messages ICMP echo request (commande ping)

3 le DNS (Domain Name System)

- Dans l'Internet
  - Une machine n'est connue que par l'adresse IP de la forme: **156.18.19.220**
    - Elle est indispensable pour pouvoir se connecter à cette machine
  - Un n° est difficile à mémoriser pour un utilisateur final
    - il est plus simple de retenir **www.ec-lyon.fr** que **156.18.19.220**
  - L'adresse dépend de l'architecture du réseau
    - En cas de changement de fournisseur d'accès, la machine change d'adresse
- Solution 1 :
  - Stocker les correspondances dans un fichier sur chaque machine:
    - c'est ce qui était fait au début : fichier ***/etc/hosts*** sur UNIX
- Solution 2 : DNS [Domain Name System] - Système de noms de domaine
  - Annuaire réparti (base de données répartie)
    - Les applications interrogent le DNS pour convertir le nom donné par l'utilisateur en une adresse IP indispensable pour la connexion au serveur
  - Avantage: le nom de la machine devient indépendant de son adresse IP
    - Le nom est librement choisi par le propriétaire de la machine
    - L'adresse IP est imposée par le fournisseur d'accès → nécessité pour le routage

Espace de noms du DNS (1/2)

- Espace de noms hiérarchique:
  - éviter les homonymes ➔ unicité des noms à l 'échelle mondiale
  - simplifier la gestion: délégation de zones



Espace de noms du DNS (2/2)

- Au premier niveau:
  - 7 domaines génériques traditionnels [gTLD (generic top-level domain)]
    - **edu** [education], **gov** [governmental], **mil** [military] réservés aux USA
    - **com** [commercial], **net** [networks], **org** [organisation], **int** [international] utilisables dans le monde entier
  - 1 domaine par pays [cc-TLD (country code top-level domain)]
    - utilisation du code pays à 2 lettres (ISO 3166)
    - exemples: **fr** (France), **uk** (Royaume-Uni), **it** (Italie), etc...
  - De nouveaux domaines génériques
    - Ajoutés régulièrement depuis 2001
    - **aero, biz, coop, info, museum, name, pro, jobs, mobi, travel**
- Au deuxième niveau, c'est très variable:
  - **uk** et **jp** subdivisaient en **co** [commercial] et **ac** [academic],
  - **fr** ne subdivise pas, mais il a existé **asso** (associations), **tm** (marques déposés), etc...

Gestion des domaines (1/3)

- Gestion administrative
  - La gestion globale des domaines est coordonnée par l'IANA (avec transition progressive à l'ICANN)
  - Les TLD sont gérés par des organismes ayant reçu un délégation de l'ICANN :
    - Pour la plupart des gTLD il est possible d'enregistrer un nom de domaine via un mécanisme de « registrar » (env. 400 dans le monde)
    - Pour les cc-TLD, il existe un NIC local qui gère le domaine du pays
      - AFNIC [Association Française pour le Nommage Internet en Coopération]
  - Pour créer un sous-domaine
    - il faut avoir l'autorisation du domaine de niveau supérieur qui va réaliser l 'enregistrement
    - Exemples:
      - pour créer ec-lyon.fr il faut l'autorisation de l 'AFNIC
      - pour créer ictt.ec-lyon.fr il faut l'autorisation de l 'Ecole Centrale

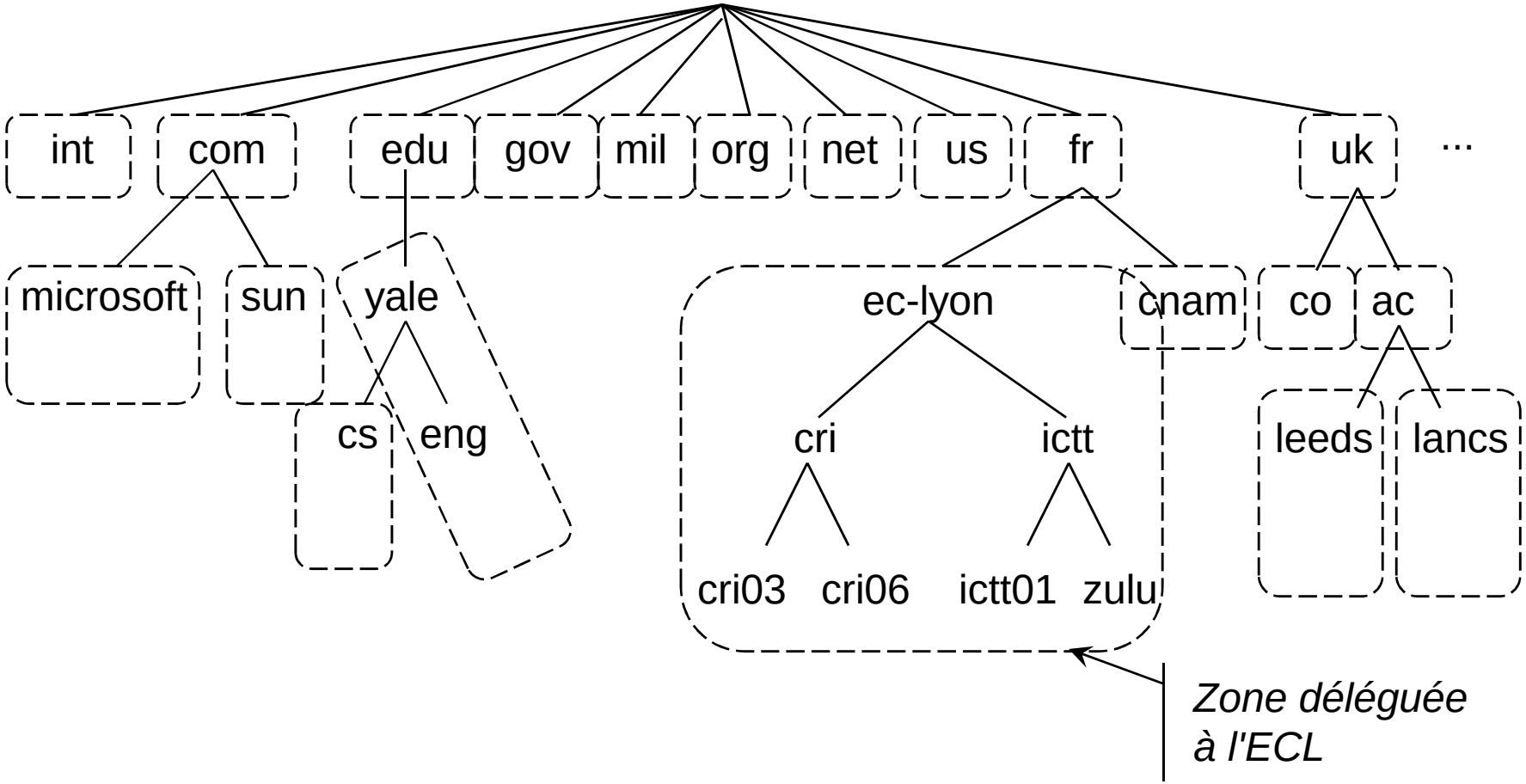
Gestion des domaines (2/3)

- Le nommage est basé sur des organisations
  - Indépendance par rapport à l ’architecture physique des réseaux
  - Exemples :
    - ardemi.fr partageait la même classe B que ec-lyon.fr (156.18.0.0)
    - urec.cnrs.fr a deux réseaux : un a l ’Université de Jussieu (Paris) et l ’autre à l ’Université de Grenoble 1

- Gestion technique
  - L’arbre est divisé en zones ; chaque zone
    - Gère une sous-partie de l’arbre
    - Possède un serveur primaire et plusieurs serveurs secondaires
  - Chaque zone peut être divisée en sous-zones
  - les frontières d ’une zone ne dépendent que de son administrateur et des délégations de gestion qu’il a accordé

Gestion des domaines (3/3)

- Exemple de découpage en zones:



Serveurs de noms

- Chaque zone possède plusieurs serveurs:
  - un serveur primaire qui contient la base de donnée originale
  - un ou plusieurs serveurs secondaires qui contiennent des copies de la base et qui peuvent répondre en cas de défaillance du serveur primaire (ou du réseau d 'accès au serveur)
- Chaque serveur peut servir une ou plusieurs zones, comme serveur primaire ou secondaire
- Chaque serveur de zone primaire contient une base de donnée de description de la zone gérée:
  - description de la zone
  - liste d'enregistrements de ressources
  - pointeur vers d 'autres serveurs de noms [glue records]

Enregistrement de ressource

- Faire correspondre à chaque nom de domaine un enregistrement de ressource:
  - l 'adresse IP d 'une machine (ou hôte)
  - un relais de messagerie
  - un serveur de nom de domaines
  - un alias d 'une machine, etc…
- Structure d 'un enregistrement:

  `Nom_domaine  durée_de_vie  classe  type  valeur`

  - **`Nom_domaine`**: nom du domaine (complet ou relatif)
  - **`durée_de_vie`**: stabilité de cet enregistrement; utilisé par le système de cache lors de l 'interrogation du DNS
  - **`classe`**: aujourd'hui seule la valeur **IN** (pour Internet) est définie
  - **`type`**: type de l 'enregistrement
  - **`valeur`**: valeur de l 'enregistrement; dépend du type

Types d ’enregistrement (1/2)

- Enregistrement SOA [Start of Authority] :
  - décrit la zone gérée par un serveur de nom
  - contient l ’adresse de messagerie électronique du responsable
  - un numéro de série unique et divers paramètres de temporisation
- Enregistrement NS [Name server] :
  - permet à de spécifier les serveurs de noms qui gèrent un domaine afin de faire les liens entre les serveurs DNS
- Enregistrement A [Address] :
  - permet de faire correspondre un nom de machine à une adresse IP
  - exemple:
    ```
    cri03.cri.ec-lyon.fr.  86400   IN   A   156.18.22.3
    ```
  - un même hôte peut avoir plusieurs adresses IP (cas d ’une machine ayant plusieurs cartes réseaux sur différents réseaux)
  - une même adresse IP peut être attribuée à des noms différents

Types d ’enregistrement (2/2)

- Enregistrement CNAME [canonical name] :
  - permet de créer des alias sur des hôtes
  - exemple:

```
www.ec-lyon.fr. 86400  IN CNAME  cri06.cri.ec-lyon.fr.
```

  - permet de donner des noms plus « explicites » à des hôtes
  - permet de changer plus facilement la machine; exemple: migration sur trotek05.trotek.ec-lyon.fr transparente pour les utilisateurs du serveur Web
- Enregistrement MX [Mail exchanger] :
  - permet de spécifier la machine qui va servir de relais de messagerie pour un domaine
  - exemple:

```
ec-lyon.fr.     86400  IN  MX  10  mail1.ec-lyon.fr.
```

- Enregistrement PTR [pointer] :
  - enregistrement spécial utilisé pour le DNS inverse (voir plus loin)

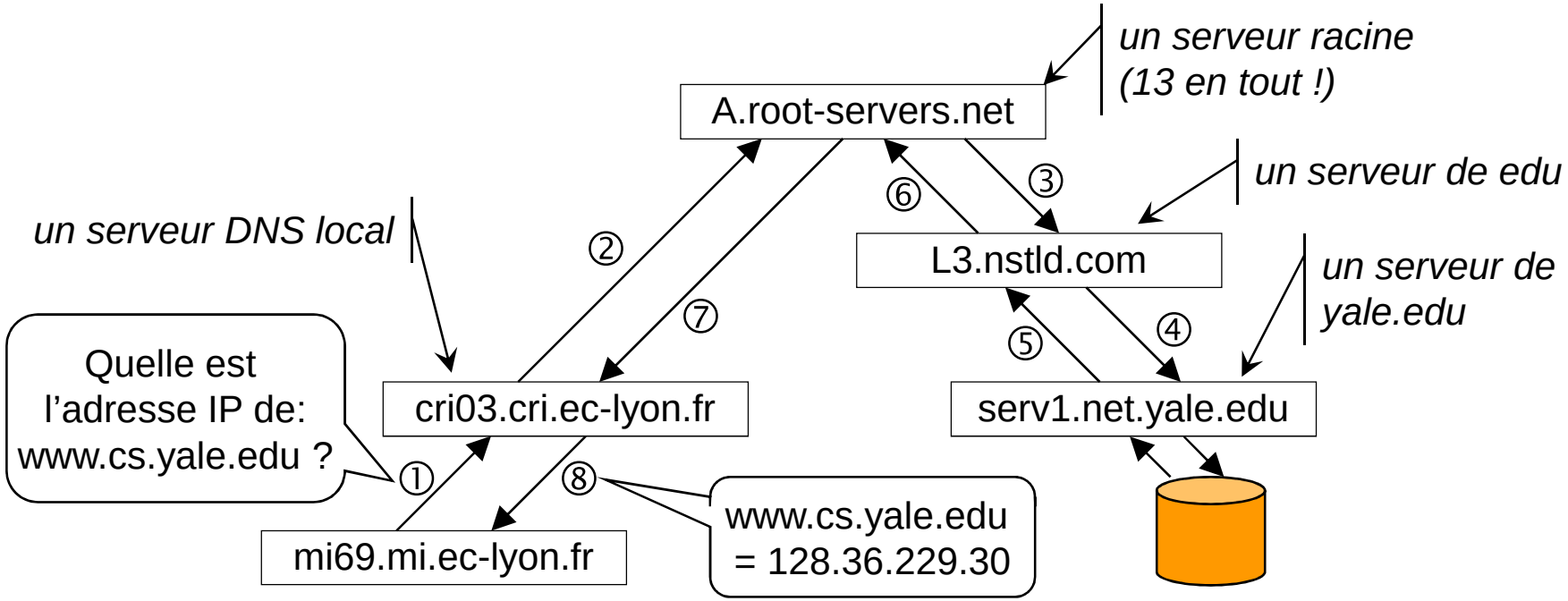
Exemple d ’enregistrements

```
ec-lyon.fr.        86400 IN  SOA     cri03.cri.ec-lyon.fr. dnsmaster.ec-lyon.fr.
                                        (2008101801 21600 3600 604800 86400)
                   86400 IN  NS      cri03.cri.ec-lyon.fr.
                   86400 IN  NS      ecl05.servers.ec-lyon.fr.
                   86400 IN  NS      calypso.urec.cnrs.fr.
                   86400 IN  MX      10  mail2b.ec-lyon.fr.
                   86400 IN  MX      999  mail3.ec-lyon.fr.
                   86400 IN  MX      10  mail1.ec-lyon.fr.
                   86400 IN  MX      10  mail2.ec-lyon.fr.
   ...
cri03.cri          86400 IN  A       156.18.22.3
                   86400 IN  MX      10  mail1.ec-lyon.fr.
                   86400 IN  MX      10  mail2.ec-lyon.fr.
                   86400 IN  MX      999  mail3.ec-lyon.fr.
   ...
ecl05.servers      86400 IN  A       156.18.19.5
   ...
www                86400 IN  A       156.18.19.220
   ...
```

Interrogation du DNS (1/2)

- Principe (méthode récursive) :
  - Le client s 'adresse au serveur de nom local :
    - ➔ si celui-ci a la réponse, il retourne directement la valeur
    - ➔ sinon il transmet la requête à un serveur de nom racine qui fait redescendre la demande et ainsi de suite
  - la réponse remonte la chaîne jusqu'au client initial

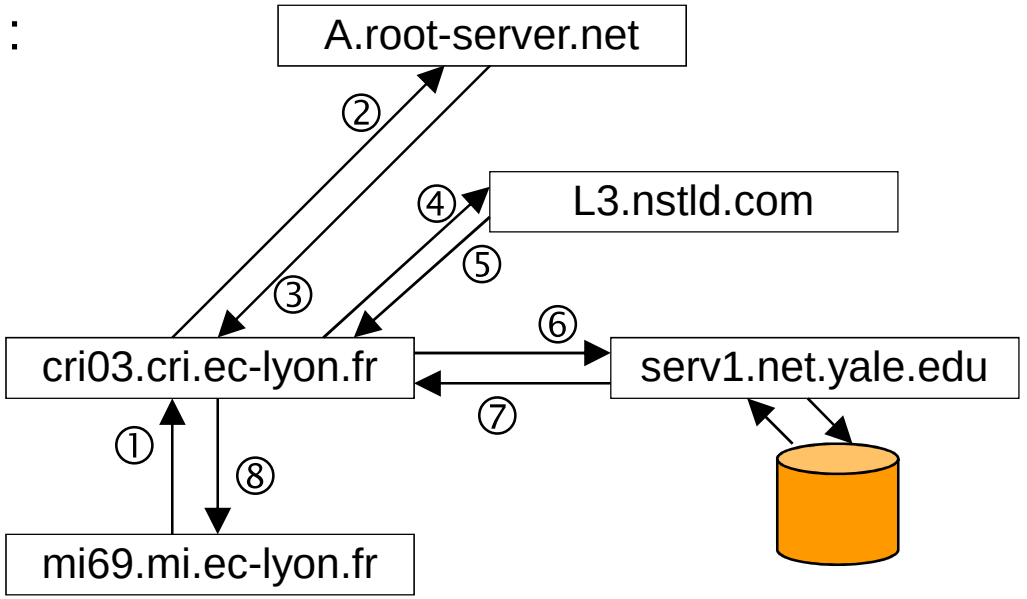
Pour explorer le DNS : **nslookup**



Interrogation du DNS (2/2)

- Méthode itérative :
  - le client s ’adresse au serveur DNS local
  - s ’il n ’a pas la réponse il lui retourne le nom et l ’adresse d'un autre serveur à contacter qui pourrait avoir la réponse
  - le client interroge ce serveur et ainsi de suite jusqu’à ce qu’il ait la réponse

- En pratique, méthode mixte :
  - le client interroge le serveur DNS local
  - s'il n'a pas la réponse, il propage la requête par la méthode itérative aux autres serveurs
  - il retourne la réponse au client



Notion de cache (1/2)

- Problème de performance de l'interrogation:
  - il n ’est pas performant d ’aller interroger à chaque fois un (ou plusieurs) serveur(s) de nom à l ’autre bout du monde
  - en particulier très pénalisant pour les serveurs racines (13 en tout dans le monde !!)
- Solution:
  - mettre en place des "mémoires caches"
  - conserver les réponses des requêtes précédentes
  - si la réponse est dans le cache, alors on la retourne directement sans faire de requête
  - d'autant plus efficace :
    - ➔ que ce sont des requêtes fréquentes (adresses des serveurs racine et des serveurs de 1er niveau)
    - ➔ qu'il y a beaucoup d'utilisateurs dans le réseau local

Notion de cache (2/2)

- L ’information conservée dans les caches peut devenir obsolète suite à un changement dans l ’architecture du réseau:
  - disparition d ’une machine
  - changement d ’adresse d ’une machine, etc…
- l ’information n ’est donc conservée dans un cache que pendant la « durée_de_vie » de l ’enregistrement:
  - donné par l ’enregistrement
  - typiquement 86400 s = 24 h
- Malgré tout, comme l ’information peut être caduque:
  - le client est informé que la réponse vient d ’un cache (non-authoritative answer)
  - le serveur donne le nom et l ’adresse du serveur de nom capable de donner une réponse de « première main » et que le client peut interroger directement s ’il souhaite privilégier la fiabilité à la performance

Protocole d ’accès au DNS

- Protocole basé sur UDP:
  - les questions et les réponses ont tous le même format:
    - faciliter pour retransmettre le message de serveur en serveur
    - la réponse est insérée dans le message qui est retourné avec la question

0	16	31
Identification	paramètre	
Nombre de questions	Nombre de réponses	
Nombre d ’« authority »	Nombre de compléments	
Section des questions ...		
Section des réponses ...		
Section des « authority » ...		
Section des informations additionnelles ...		

Correspondances inverses (1/2)

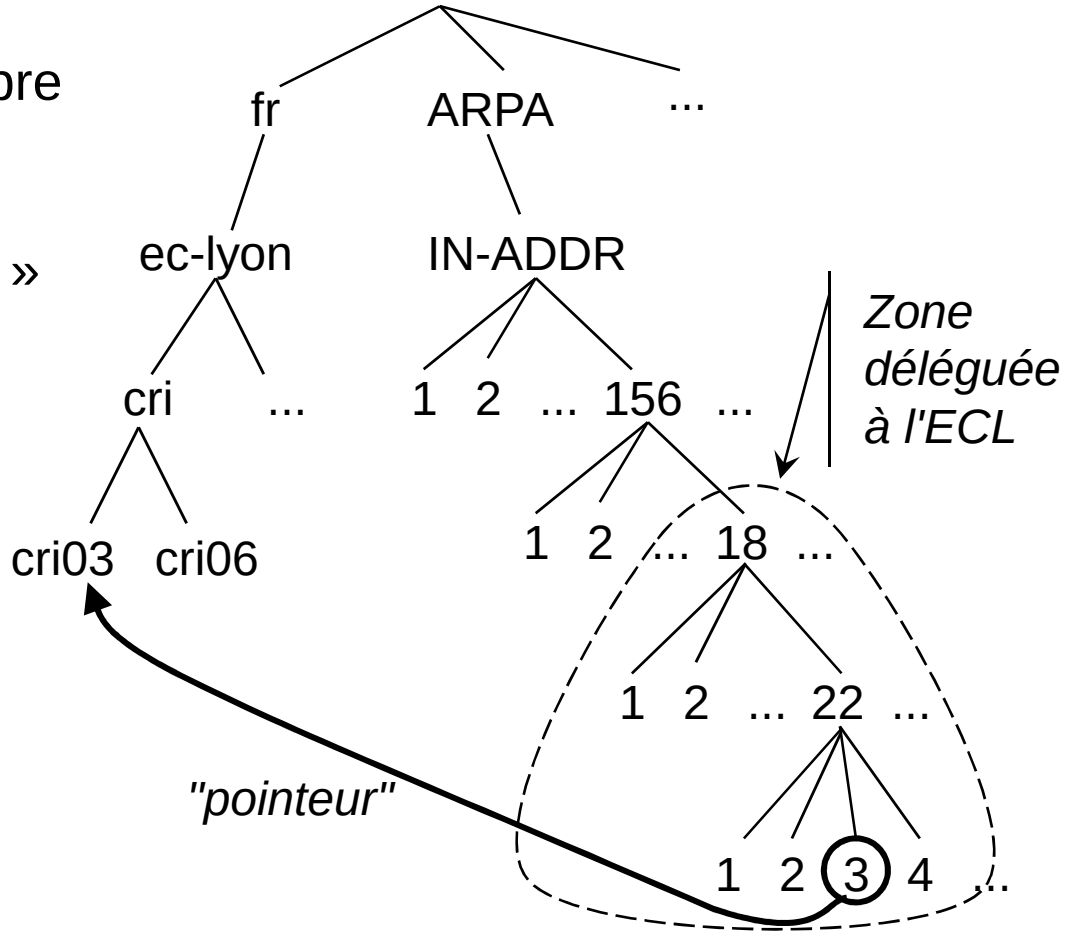
- Problème:
  Comment obtenir le nom de l'hôte à partir d'une adresse IP ?
- Solutions:
  - faire des « requêtes inverses » ==> trop complexe car pas de rapport simple entre nom de domaines et adresses IP
  - rajouter dans le DNS des enregistrements spéciaux faisant la correspondance entre l'adresse IP et le nom de domaine
- Mise en œuvre:
  - il existe un pseudo-domaine **`IN-ADDR.ARPA`** dans lequel les adresses IP sont enregistrées de manière hiérarchique
  - Utilisation des enregistrements PTR pour faire correspondre une entrée de cette arbre avec un nom de domaine (pointeur)
  - Ce sous-arbre est également divisé en zones qui sont déléguées aux institutions qui possèdent les adresses IP correspondantes

Correspondances inverses (2/2)

- Exemple:

```
3.22.18.156.in-addr.arpa.   IN  PTR  cri03.cri.ec-lyon.fr.
```

- Remarques:
  - Comme on lit dans l'arbre de bas en haut, les adresses IP sont donc spécifiées « à l 'envers »
  - la cohérence entre les correspondances directes et inverses doit être assurée à la main!



Ajouts pour IP v6

- Pour les enregistrements directs :
  - Nouvel enregistrement AAAA pour la correspondre entre un nom de machine et une adresse IP
  - Exemple :
    cri03.cri.ec-lyon.fr. 86400 IN AAAA 2001:0660:5004:0016:0006:8D00:00C5:88FC

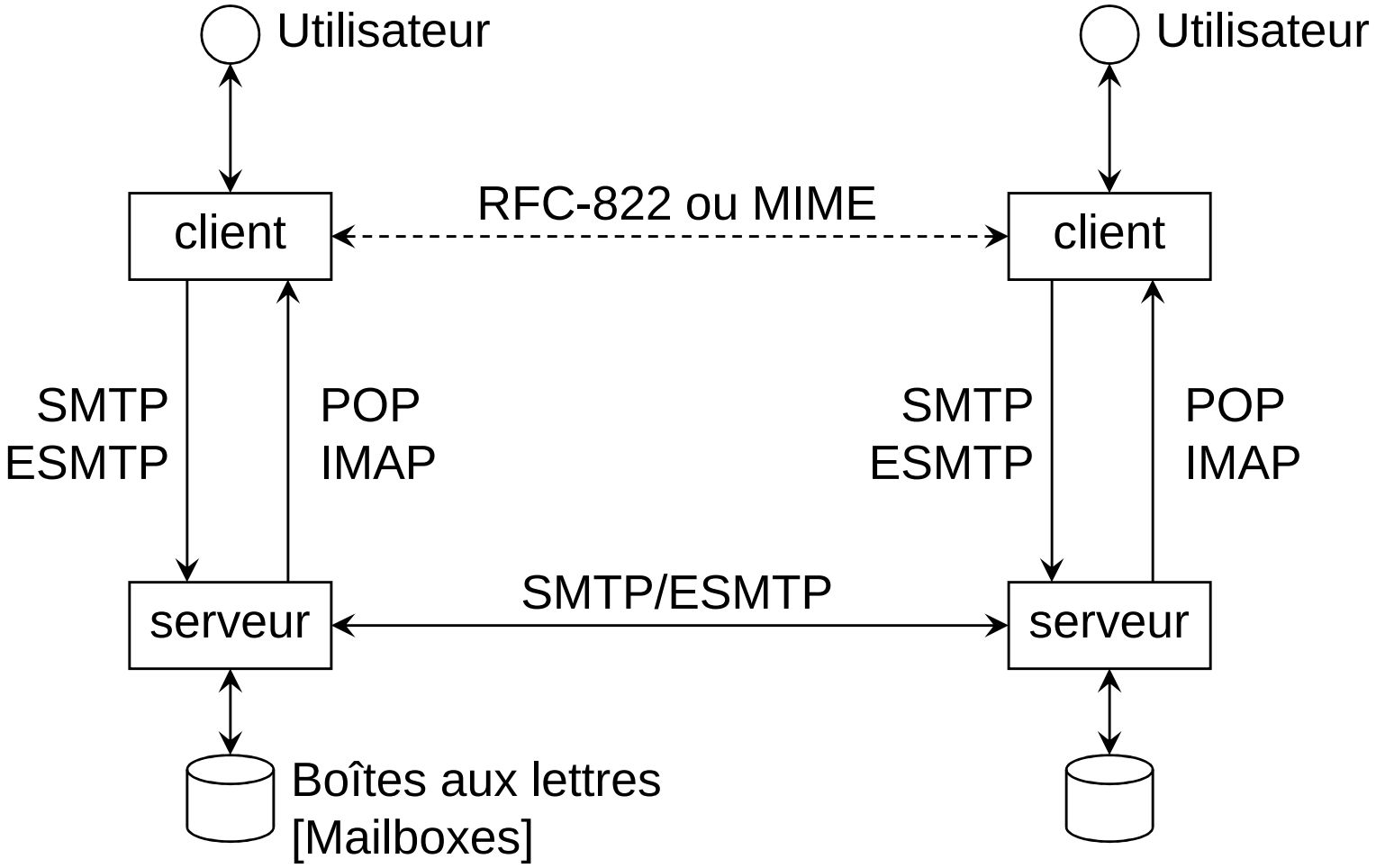
- Pour le DNS inverse :
  - Nouveau pseudo-domaine ip6.arpa
  - Codage des adresses à l'envers comme avec IPv4 mais en utilisant chaque chiffre héxadécimal comme nœud de l'arbre (soit 32 niveaux de 4 bits)
  - Exemple :
    C.F.8.8.5.C.0.0.0.0.D.8.6.0.0.0.6.1.0.0.4.0.0.5.0.6.6.0.1.0.0.2.ip6.arpa. IN PTR cri03.cri.ec-lyon.fr.

4

Messagerie électronique

- Messagerie électronique [Electronic Mail ou E-mail]:
  - communication interpersonnelle en temps différé
  - message écrit: persistance des messages
  - mode « store and forward »: stockage temporaire des messages avant retransmission au nœud suivant
  - avis en cas de non-remise
  - transfert de messages textuels sans accents (ASCII)
  - extensions permettant de transmettre tout type de données
- Historique:
  - 1982: RFC 821 (SMTP) et RFC 822
  - 1992: RFC 1341 (MIME 1ère version)
  - 1994: RFC 1651 (ESMTP 1ère version)
  - 1996: RFC 2045 à 2049 (MIME 3ème version)
  - 2001: RFC 2821 et 2822 rendent obsolètes RFC 821 et 822

Architecture fonctionnelle



Définitions

- SMTP [Simple Mail Transfer Protocol] et ESMTP [Enhanced SMTP]:
  - Protocoles de transfert de messages
- POP [Post Office Protocol]:
  - protocole permettant la relève de la boîte aux lettres à distance
- IMAP [Interactive Mail Access Protocol]:
  - protocole de gestion de la boîte aux lettres d'un serveur à distance
- RFC-822 et MIME [Multipurpose Internet Mail Extensions]:
  - protocoles de codage des messages électroniques
- Remarques:
  - les termes MTA (pour serveur) et UA (pour client) sont parfois utilisés en référence à l 'OSI
  - la traduction officielle en français de E-mail est « mél »; certains utilisateurs utilisent également « courriel »

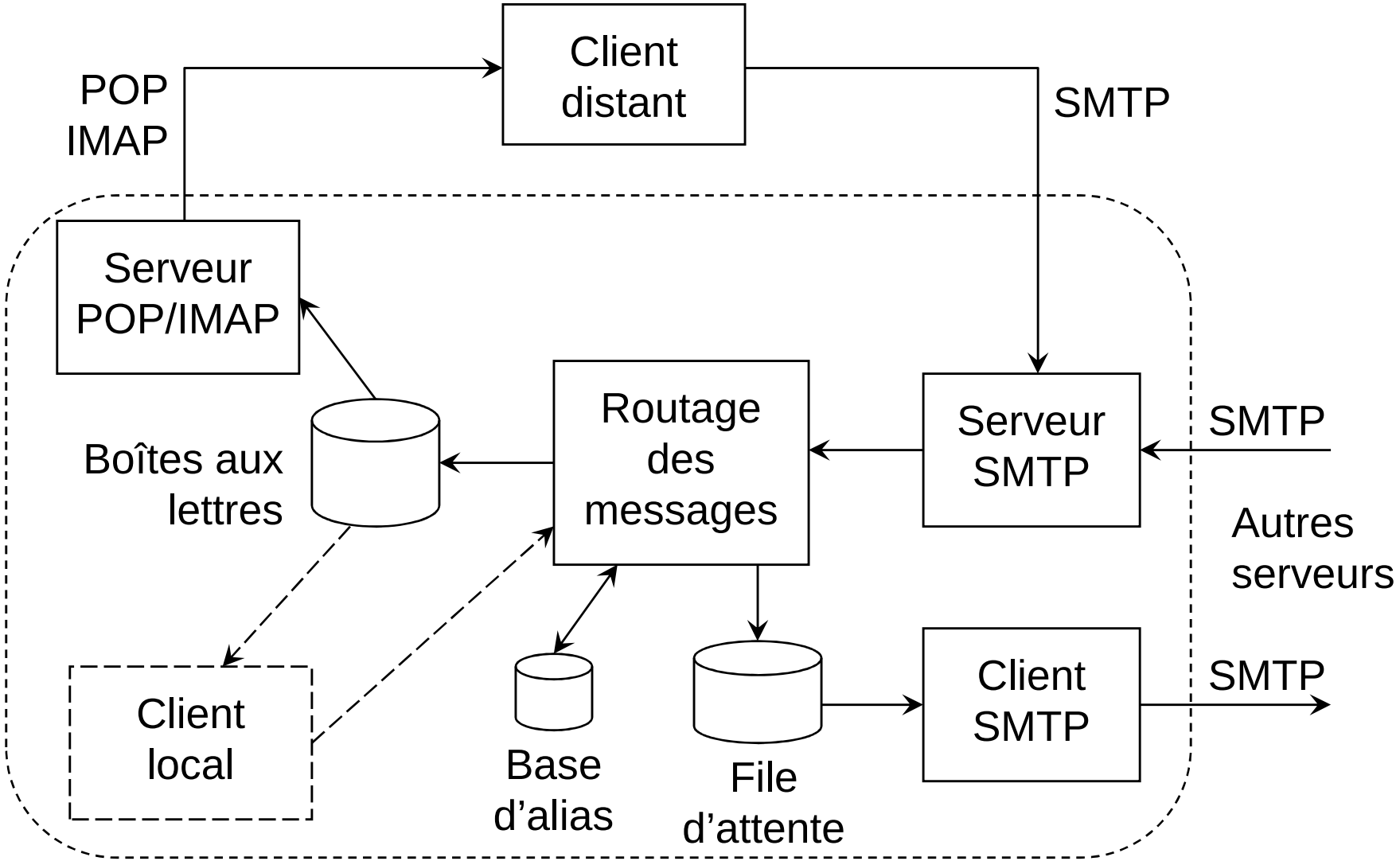
Adresses de messagerie

- Format des adresses:
  - **`boite@domaine`**
- Domaine de messagerie (partie droite):
  - Correspont à un (ou plusieurs) serveur(s) qui va(vont) recevoir les messages grâce aux enregistrements MX du DNS:
    - s'il y a plusieurs serveurs, ils sont utilisés par ordre de priorité croissant
    - permet de donner des serveurs de secours en cas de panne
- Boite (partie gauche):
  - identification de la boîte aux lettres
  - peut correspondre à un utilisateur, à un groupe (liste de diffusion) ou à un automate
- Adresse obligatoire:
  - **`postmaster@domaine`** (utilisateur humain auquel on peut s 'adresser en cas de problèmes)

Adresses de messagerie (compléments)

- Exemples:
  - **`Rene.Chalon@ec-lyon.fr`**
  - **`rchalon@mrash.fr`**
- Pour faciliter la mémorisation des adresses, Il est souvent utilisé des alias:
  - exemple: Rene.chalon@mrash.fr est un alias de rchalon@mrash.fr
- Pour faciliter l ’administration, pour tout domaine de messagerie existant, l ’adresse **`postmaster@domaine`** doit exister et correspondre à un utilisateur humain auquel on peut s ’adresser en cas de problèmes

Structure d ’un serveur



Protocole SMTP

- Protocole SMTP:
  - modèle client/serveur
  - construit sur TCP (le serveur écoute sur le port 25)
  - le dialogue se fait en clair (commandes en ASCII)
- Connexion directe du serveur initial (éventuellement du client!) vers le serveur du destinataire:
  - pas de relais intermédiaire (sauf pour raisons de sécurité)
- Transfert de messages ASCII non accentués seulement
- Commandes:
  - HELO: identification du client sur le serveur
  - MAIL FROM: expéditeur
  - RCPT TO: destinataire (répété si plusieurs destinataires)
  - DATA: débute l 'envoi du message (la fin est signalé par un point)
  - QUIT: fin du dialogue

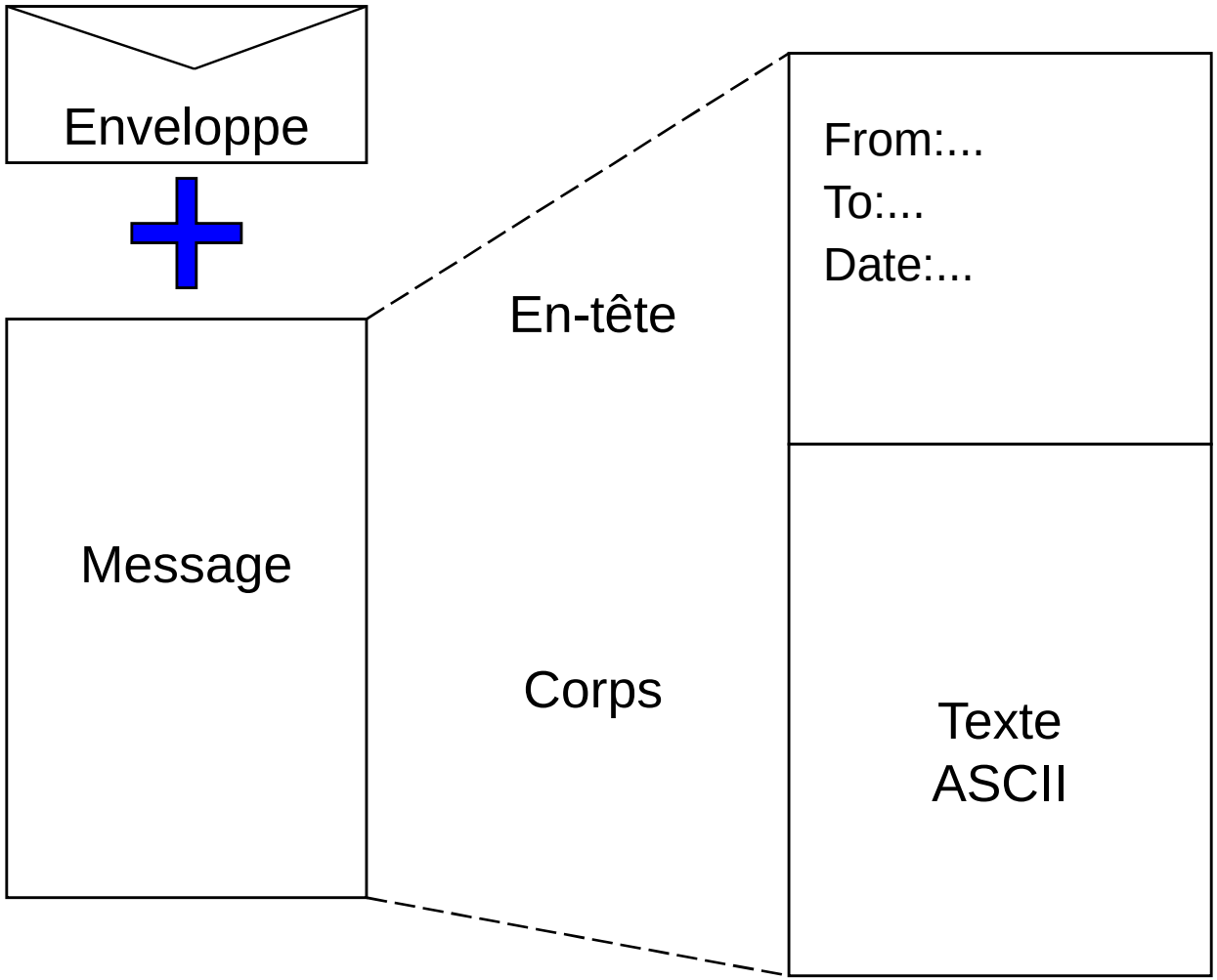
Exemple de connexion SMTP

```
Client
        Serveur
-------------------------------------------------------
        220 SMTP ready
HELO zulu.ictt.ec-lyon.fr
        250 cc03.cc.ec-lyon.fr
MAIL FROM: Rene.chalon@ictt.ec-lyon.fr
        250 ok
RCPT TO: postmaster@ec-lyon.fr
        250 ok
DATA
        354 start mail input; end with .
message ...
message ...
.
        250 ok
QUIT
        221 cc03.cc.ec-lyon.fr closing
```

Protocole ESMTP

- Extension du protocole SMTP
- supporte les transferts de messages codés sur 8 bits
- Nombreuses améliorations:
  - permet de préciser la taille des messages avant l ’envoi, ce qui permet au serveur destinataire de refuser des messages trop grands
  - meilleure gestion de la sécurité
- le dialogue est initié par EHLO:
  - si le serveur répond 250, alors il supporte ESMTP
  - s ’il répond 500 (commande inconnue), alors revenir à SMTP en envoyant HELO

Format des messages (RFC-822)



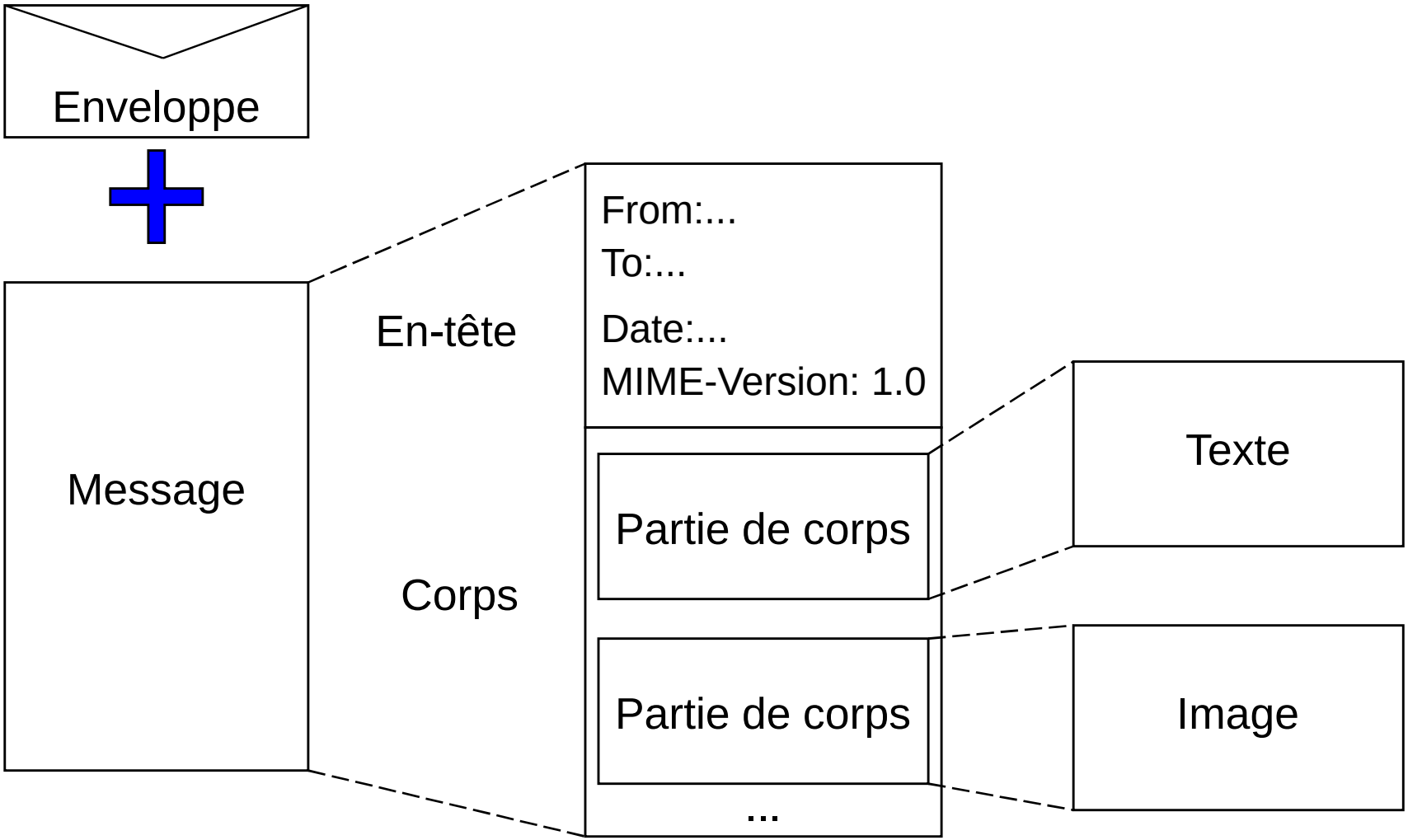
Champs d ’en-tête RFC-822

- Quelques champs d ’en-tête RFC-822 :
  - From: expéditeur du message
  - Sender: adresse de celui qui a envoyé le message
  - To: destinataires
  - Cc: destinataires en copie
  - Bcc: destinataires en copie muette
  - Received: trace ajoutée par chaque serveur SMTP qui a relayé le message
  - Reply-to: adresse de la personne à qui il faut répondre (ajouté par l ’expéditeur du message)
  - Message-id: identificateur unique pour le message
  - Date: date et heure d ’émission du message
  - Subject: objet du message

Corps du message RFC-822

- Le corps est séparé de l ’en-tête par une ligne vide
- le corps ne peut contenir que du texte ASCII non-accentué sous forme de ligne de 80 caractères
- Pour envoyer des contenus d ’autre nature, il faut coder les données:
  - par exemple: uuencode, BinHex
  - ces codages transforment les données binaires en « texte » de manière à être transportées de manière transparente
  - inconvénient: le destinataire doit disposer du décodeur pour récupérer le contenu initial.
  - Le destinataire n ’a aucune information sur le contenu du fichier
- Meilleure solution: MIME

MIME


Enveloppe
Message
En-tête
Corps
From:...
To:...
Date:...
MIME-Version: 1.0
Partie de corps
Partie de corps
...
Texte
Image

Extensions MIME

- Extensions à la RFC-822: RFC 2045 à 2047 + 2048 et 2049
- Ajout de nouveaux champs d 'en-tête:
  - MIME-Version: identification de la version (aujourd 'hui 1.0)
  - Content-Description: description en clair du contenu
  - Content-Id: identificateur unique (format idem Message-id)
  - Content-Transfer-Encoding: façon dont le contenu est codé pour être transporté
  - Content-type: spécifie le contenu du message:
    - format: `<type>/<sous-type>; <paramètres éventuels>`
- Nouvelle structuration du corps des messages:
  - le corps du message peut être d 'un seul type
  - le corps du message contient plusieurs parties --> messages multiparties

Types de contenus

Type	Sous-type	Description
text	plain	Texte non formaté
	html	Texte avec des commandes de formatage
image	gif	Image au format gif
	jpeg	Image au format JPEG
audio	basic	Son non compressé
video	mpeg	Vidéo au format MPEG
application	octet-stream	Suite d ’octets non-interprétés
	postcript	Fichier imprimable au format PostScript
message	rfc822	Message MIME RFC 822
	partial	Partie de message (découpé pour la transmission
	external-body	Le corps contient une référence externe

Messages multiparties

Type	Sous-type	Description
multipart	mixed	Parties indépendantes
	alternative	Même message en différents formats
	parallel	Parties à voir simultanément
	digest	Chaque partie est un message RFC 822

- Un paramètre supplémentaire précise le texte qui sera utilisé pour séparer chaque partie:
  - Content-type: multipart/mixed; boudary=azertyuiop
- Chaque partie contient:
  - un en-tête avec les champs:
    - Content-Type (si non présent, alors text/plain; charset=us-ascii)
    - Content-Transfer-Encoding
  - une ligne vide séparatrice
  - le contenu de la partie

Codages pour le transfert (1/2)

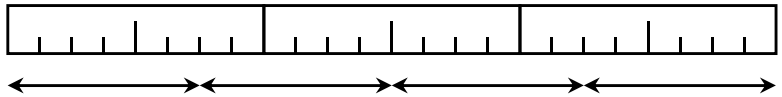
- Pour être compatible avec le système de transport SMTP (caractères ASCII seulement), il faut encoder les messages
  - Utilisation de l'en-tête "Content-Transfer-Encoding" :
    - quoted-printable
    - base64
- Codage **Quoted-printable** (QP)
  - pour les textes simples contenant des caractères accentués
  - codage du code du caractère en hexadécimal en préfixant par =
  - exemples:
    si code ISO 8859-1 utilisé, →
    - Déjà se code `D=E9j=E0`
    - çà se code `=E7=E0`

poids forts →

poids faibles ↓	A	B	C	D	E	F
0		°	à	Ð	à	ð
1	¡	±	Á	Ñ	á	ñ
2	¢	²	Â	Ò	â	ò
3	£	³	Ã	Ó	ã	ó
4	¤	´	Ä	Ô	ä	ô
5	¥	µ	Å	Õ	å	õ
6	¦	¶	Æ	Ö	æ	ö
7	§	·	Ç	×	ç	÷
8	¨	¸	È	Ø	è	ø
9	©	¹	É	Ù	é	ù
A	ª	º	Ê	Ú	ê	ú
B	«	»	Ë	Û	ë	û
C	¬	¼	Ì	Ü	ì	ü
D		½	Í	Ý	í	ý
E	®	¾	Î	Þ	î	þ
F	¯	¿	Ï	ß	ï	ÿ

Codages pour le transfert (2/2)

- Codage **Base64** ou **blindage ASCII**
  - fonctionne pour tout type de données
  - on groupe 3 octets = 24 bits qu'on divise en 4 groupes de 6 bits



  - à chaque combinaison de 6 bits on associe un caractère ASCII selon ce codage →
    - ➔ si pas assez d'octets alors "=" et "==" servent au bourrage à la fin
    - ➔ des CR et LF sont ajoutés pour faire des lignes de 76 caractères

poids forts →

poids faibles ↓	00	01	10	11
0000	A	Q	g	w
0001	B	R	h	x
0010	C	S	i	y
0011	D	T	j	z
0100	E	U	k	0
0101	F	V	l	1
0110	G	W	m	2
0111	H	X	n	3
1000	I	Y	o	4
1001	J	Z	p	5
1010	K	a	q	6
1011	L	b	r	7
1100	M	c	s	8
1101	N	d	t	9
1110	O	e	u	+
1111	P	f	v	/

  - Exemple:

3F	52	A3
0011 1111	0101 0010	1010 0011
P	1	K	j

Exemple de message MIME

```
From: Rene.Chalon@ictt.ec-lyon.fr
To: postmaster@ec-lyon.fr
MIME-Version:1.0
Message-Id: <24782407487488@ictt.ec-lyon.fr>
Content-Type: multipart/mixed; boundary=azertyuiop
Subject: Essai

--azertyuiop
Content-Type: text/plain; charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

Essai de courrier =E9lectronique
--azertyuiop
Content-Type: application/octet-stream
Content-Transfer-Encoding: base64

Ejfh/HuGp4+qz6A
--azertyuiop--
```

Remise finale du courrier à l'utilisateur

- SMTP (ou ESMTP) est prévu pour délivrer directement le message sur la machine de l 'utilisateur:
  - il faut que la machine soit toujours en écoute et connectée au réseau, ce qui n 'est pas le cas des PC et surtout des ordinateurs portables
- Le courrier est donc délivré sur un serveur central de l 'organisation et chaque client vient relever sa boîte aux lettres périodiquement pour voir si du courrier est arrivé:
  - soit directement sur le serveur de messagerie (protocole propriétaire)
  - soit en utilisant un protocole de relève de boîte
- Trois protocoles sont définis:
  - POP [Post Office Protocol]: RFC 1225, version 3
  - IMAP [Interactive Mail Access Protocol]: RFC 1064
  - DMSP [Distributed Mail System Protocol]: RFC 1056

Protocole POP3

- Permet de transférer les messages depuis le serveur de messagerie sur le poste de l 'utilisateur
- Protocole POP3:
  - modèle client/serveur
  - construit sur TCP (le serveur écoute sur le port 110)
  - le dialogue se fait en clair (commandes en ASCII)
- Commandes principales:
  - USER: identification du client sur le serveur
  - PASS: mot de passe pour authentification
  - LIST: liste des messages en attente
  - TOP: lire le début d 'un message
  - RETR: récupération d 'un message
  - DELE: effacement d 'un message
  - QUIT: fin du dialogue

Exemple de connexion POP3

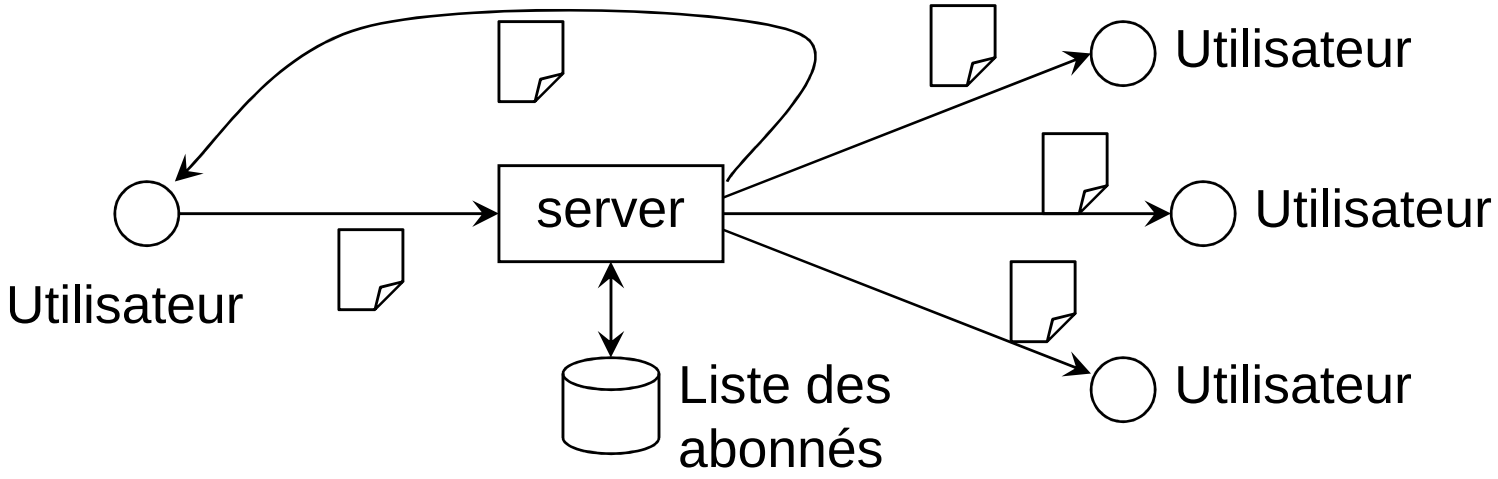
```
Client
        Serveur
        +OK cc03.cc.ec-lyon.fr POP3 server ready
USER chalon
        +OK I know you, chalon
PASS *****
        +OK Welcome! 1 messages (18184 bytes)
LIST
        +OK 1 messages (18184 bytes)
        1 18184
        .
RETR 1
        message ...
        message ...
        .
DELE 1
        +OK message deleted
QUIT
        +OK cc03.cc.ec-lyon.fr server closing down
```

Autres protocoles de remise

- IMAP:
  - protocole plus sophistiqué
  - le courrier reste sur le serveur: gestion d ’une base de donnée de courrier
  - permet de consulter ses messages depuis différents ordinateurs: au bureau, à la maison, en voyage, etc…
  - nombreuses commandes de recherche de courrier par ses attributs
  - permet de gérer des filtres de classement et d ’effacement automatique du courrier
- DMSP:
  - le courrier peut se trouver sur plusieurs serveurs et clients
  - permet de travailler sur plusieurs machines et de les synchroniser périodiquement
  - à la différence d ’IMAP, il ne nécessite pas d ’être connecté au serveur pour travailler

Listes de messagerie

- Utilisation de la messagerie, qui est a priori un moyen de communication interpersonnel, pour diffuser des messages au sein d ’un groupe
- Une adresse spécifique est déclarée sur un serveur qui correspond à la liste (exemple: tous@ec-lyon.fr)
- Tout message envoyé à cette adresse est renvoyé automatiquement par le serveur à tous les abonnés de la liste



Caractéristiques des listes de messagerie (1/2)

- Les listes peuvent être:
  - publiques: quiconque peut s 'abonner
  - non-publiques: seuls les gestionnaires de la liste peuvent ajouter des abonnés
- Listes restreintes (ou fermées):
  - seuls les abonnés (et les modérateurs) peuvent envoyer des messages
- Listes modérées:
  - le message est lu par un (ou plusieurs) modérateurs qui juge de l 'opportunité de diffuser le message à la liste toute entière
- Mode « digest »:
  - Au lieu d 'envoyer immédiatement à tous les abonnés les messages qui arrivent à la liste, certains abonnés peuvent choisir de recevoir tous les messages des dernières 24h (ou autre période) dans un seul message journalier

Caractéristiques des listes de messagerie (2/2)

- Listes anonymes:
  - toute référence liée à l ’expéditeur est supprimée du message
- Archivage des messages:
  - Tous les messages envoyés à la liste peuvent être archivées dans un fichier ou une base de donnée sur le serveur
  - certains systèmes proposent de pouvoir consulter cette archive sur un serveur FTP ou Web
- Autres options:
  - la réponse (champ Reply-To) peut être positionné pour répondre à la liste ou à l ’expéditeur du message
  - on peut autoriser ou interdire la consultation de la liste des membres abonnés

Serveurs de listes de messagerie (1/2)

- De nombreux programmes (souvent gratuits) existent pour mettre en place des listes: Listserv, majordomo, TULP, Sympa
- Le serveur dispose d 'une adresse spéciale permettant de lui envoyer des commandes; exemple: sympa@sympa.ec-lyon.fr
  - Il suffit d 'envoyer un message à cette adresse avec comme contenu la liste des commandes à exécuter
  - Un message est automatiquement retourné quelques instants après contenant le résultat de l 'exécution des commandes
- Exemple de quelques commandes de Sympa :
  - **HELp**: obtenir un fichier d 'aide
  - **LISts**: obtenir la liste de toutes les listes du serveur
  - **SUBscribe <liste> <prénom> <nom>**: s 'inscrire à la liste (c 'est l 'adresse de l 'expéditeur du message qui est utilisée); l 'inscription n 'est possible que pour les listes publiques
  - **SIGnoff <liste>**: se désabonner à la liste

Serveurs de listes de messagerie (2/2)

- **REView <liste>**: obtenir la liste des abonnés à une liste (si l 'utilisateur est autorisé)
- **ADD <liste> <user@host> <prénom> <nom>**: ajoute un utilisateur a une liste (utilisé par les propriétaires de liste pour inscrire des personnes sur une liste non-publique ou fermée)
- **DEL <liste> <user@host>**: retirer un utilisateur d 'une liste, etc...
- **DISTribute <liste> <clef>**: pour distribuer un message (réservé aux modérateurs de la liste)
- **REJect <liste> <clef>**: pour refuser un message à modérer

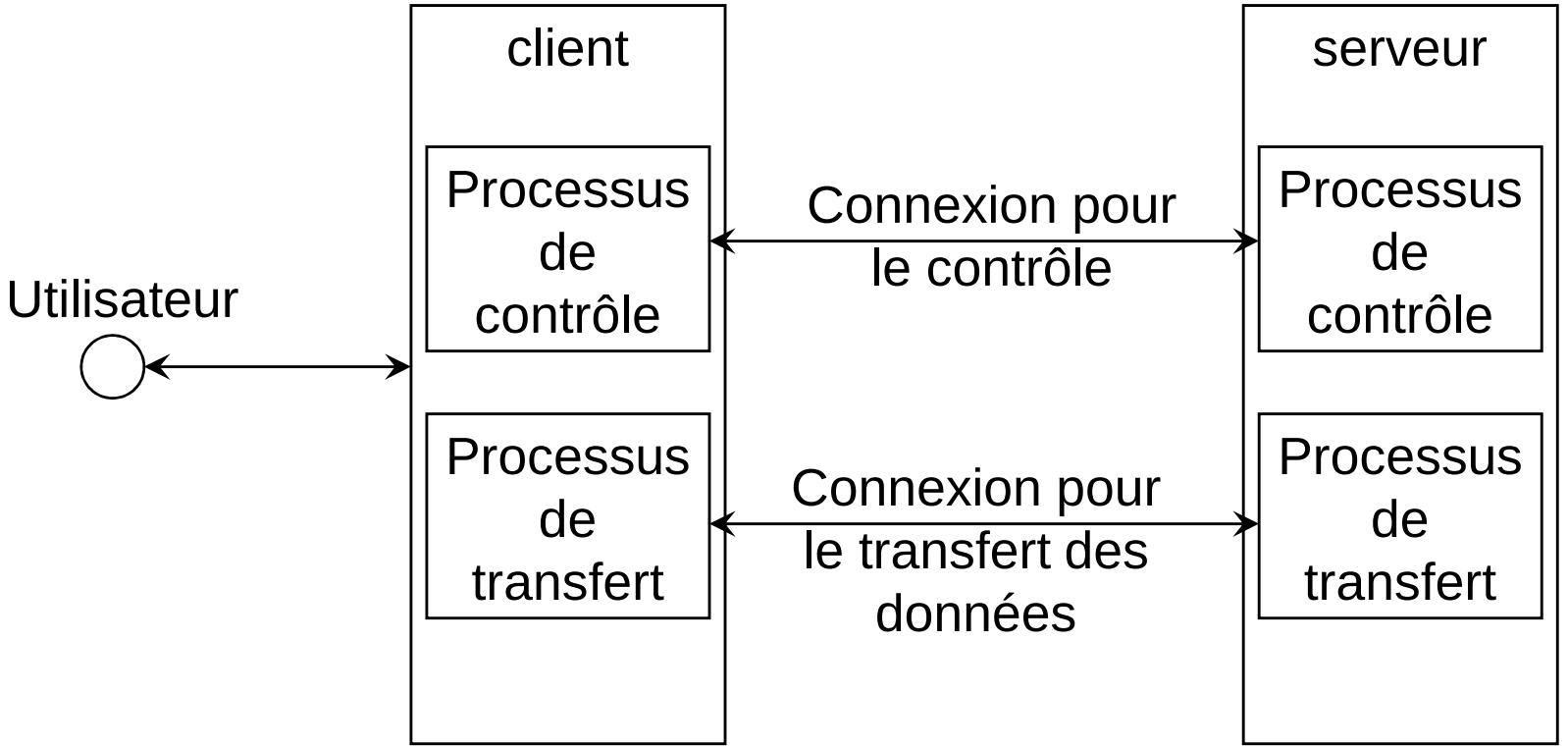
- Exemples à l 'Ecole Centrale:
  - **tous-les-personnels@sympa.ec-lyon.fr**: liste fermée du personnel (~ 450 personnes)
  - **promotion-2008@sympa.ec-lyon.fr** : liste fermée de la promo 2008
  - etc…

FTP

- FTP [File Transfer Protocol] définit un protocole de transfert fiable de fichiers entre un client et un serveur (RFC 959)
- Les transferts sont à l 'initiative du client mais il peuvent tout aussi bien être:
  - des récupérations [dowloading]: récupérer un fichier sur le serveur
  - des chargements [uploading]: déposer un fichier sur le serveur
- La connexion est authentifiée par un nom d 'utilisateur et un mot de passe
- Les droits d 'accès aux fichiers dépendent de l'utilisateur et du système d 'exploitation utilisé sur le serveur
- On distingue 2 types de transferts:
  - les transferts de fichiers textes: le client et/ou le serveur convertissent les représentations des textes
  - les transferts de fichiers binaires: transfert des données brutes

Architecture fonctionnelle de FTP

- Protocole basé sur TCP:
  - utilisation du port 21 pour les commandes
  - utilisation du port 20 pour le transfert de données



Protocole FTP

- Le client établit une connexion sur le serveur sur le port 21 en s ’authentifiant: c ’est la **connexion de contrôle**
- Toutes les commandes sont envoyés sur cette connexion
- A chaque transfert de données une nouvelle connexion est établie entre le client et le serveur et refermée après le transfert:
  - la connexion est ouverte par le serveur vers le client qui a communiqué au préalable par la connexion de contrôle le numéro du port à utiliser
  - le client peut également établir cette connexion (mode passif) mais tous les serveurs n ’acceptent pas ce mode
- Avantages de 2 connexions:
  - plusieurs transferts peuvent être lancés en parallèle
  - le canal de contrôle reste disponible pendant les transferts (par exemple pour les interrompre)
  - on peut transférer des fichiers binaires sans risque de les mélanger avec des commandes

Autre protocole de transfert de fichiers

- TFTP [Trivial File Transfer Protocol] (RFC 783):
  - fonctionne sur UDP: il gère donc lui-même les erreurs de transmission
  - basé sur l ’envoi de blocks numérotés de 512 octets qui doivent être acquittés avant la transmission du bloc suivant
  - protocole très simple lorsqu’on a pas besoin de la sophistication de FTP ou que le système d ’exploitation ne peut le supporter
  - Un usage typique: chargement de l ’image d ’un système d ’exploitation au démarrage d ’une machine sans disque dur (« bootstraping » réseau)

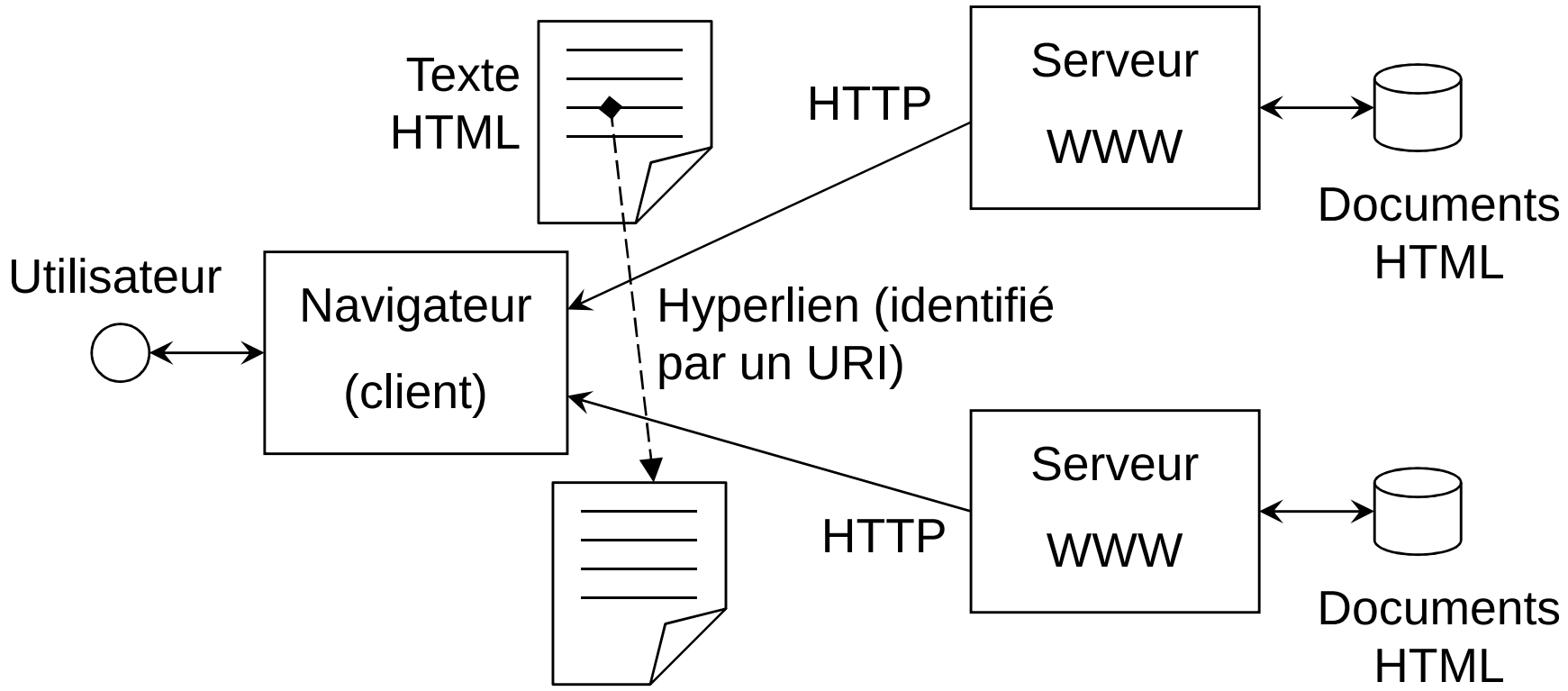
6

Le World Wide Web

- Le World Wide Web (WWW, W3 ou Web — en français, la Toile) désigne l ’ensemble des documents multimédia accessibles par l ’Internet et reliés entre eux par des **hyperliens**
- C ’est aujourd’hui l ’application la plus représentative de l ’Internet au point que les utilisateurs confondent les deux
- Historique:
  - le WWW est né au CERN en 1989:
    - ➔ le but était de faciliter l ’accès aux publications de recherche de la communauté de la physique des particules en les plaçant sur l ’Internet et en pouvant facilement trouver les documents associés
  - Le W3C [World Wide Web Consortium] est créé en 1995 pour coordonner les développements liés au Web.
    Il est conjointement hébergé par le MIT aux USA, l’ERCIM en Europe et l'Université de Keio (Japon)

Architecture fonctionnelle du Web

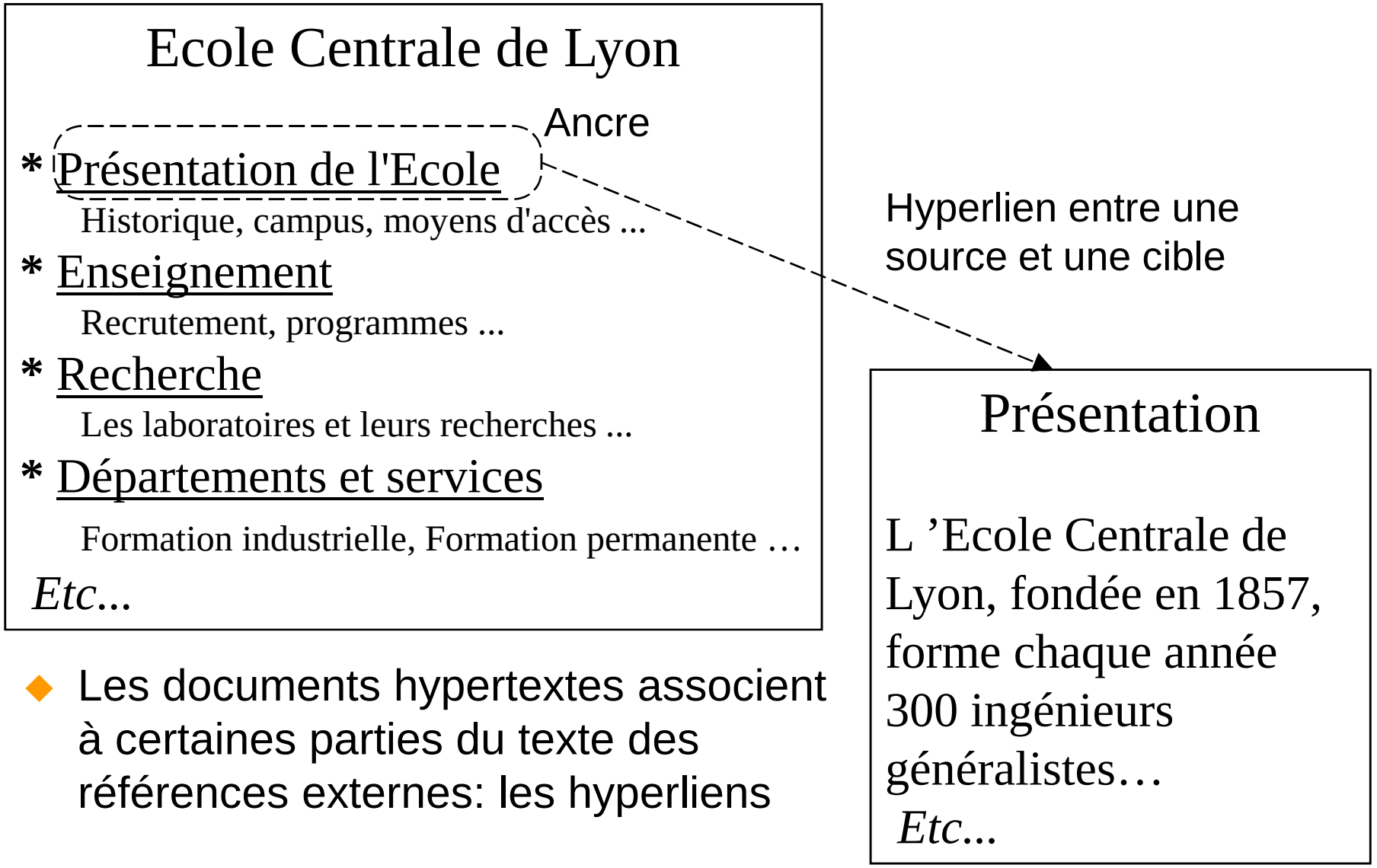
- Les principaux éléments du Web sont:
  - un protocole client/serveur : HTTP
  - un langage d ’écriture des pages : HTML
  - un moyen de référencer et de lier les pages : les URI



Définitions

- HTTP [HyperText Transfer Protocol]: protocole de transfert
- HTML [HyperText Markup Language]: langage d ’écriture des pages Web
- URI [Uniform Resource Identifier]: référence d’une ressource:
  - URL [Uniform Resource Locator]: référence par la localisation du document sur un serveur
  - URN [Uniform Resource Name]: réfénce unique et persistante mais si la ressource n’est plus accessible sur un serveur
- Navigateur: client Web permettant de récupérer les pages et de les afficher; il permet également de suivre les hyperliens. Il est parfois appeler butineur ou fureteur (en anglais, browser)
- Hypertexte: désigne une technique dans laquelle certains éléments d ’un texte « pointent » sur d ’autres éléments du même texte ou sur d ’autres documents.

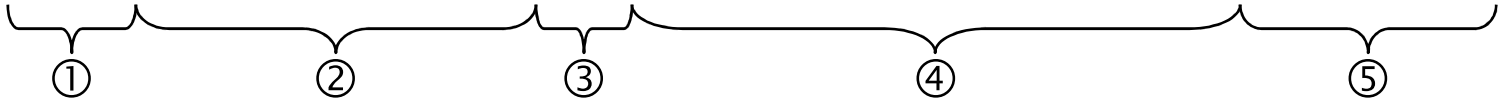
Hypertexte et hyperliens



- Les documents hypertextes associent à certaines parties du texte des références externes: les hyperliens

Les URI (RFC2396)

- A chaque hyperlien est associé une URL qui forme une référence unique à tout document sur le Web
- URL http: elle est formée de 5 parties:
  - le protocole à utiliser (pour le Web = http) ①
  - le nom du serveur où se trouve le document à charger ②
  - le numéro du port à utiliser (facultatif, par défaut = 80) ③
  - un chemin d 'accès au document sur le serveur ④
  - une référence (label) à l 'intérieur du document (facultatif) ⑤
- Exemple:
  - **`http://www.ec-lyon.fr:80/presentation/index.html#signature`**



- Références relatives:
  - référence locale au document: **`"#signature"`**
  - référence à un document sur le même serveur: **`"../intro.html"`**

Serveur Web

- Chaque serveur Web écoute le port 80 (par défaut)
- Liste des étapes pour récupérer un document (dont l 'URL est http://*nom_serveur*/*nom_document*) sur un serveur:
  - à partir de *nom_serveur* (contenu dans l 'URL), le client demande au DNS l 'adresse IP du serveur
  - le client établit une connexion TCP sur le port 80 du serveur
  - il envoie la commande: GET *nom_document* HTTP/1.0
  - le serveur envoie un en-tête et le document
  - le serveur ferme la connexion
  - le client affiche le document
- Si le client doit demander un autre document au même serveur, il doit ouvrir une nouvelle connexion
  - peu efficace, les évolutions du protocole cherchent à améliorer cela

Protocole HTTP

- HTTP [HyperText Transfer Protocol]:
  - Protocole client/serveur
  - Fonctionne sur TCP (mais UDP pourrait aussi être utilisé)
  - Port par défaut : 80
- Versions :
  - HTTP v 0.9, non normalisé (1990) - obsolète
  - HTTP v 1.0, RFC 1945 (1996)
  - HTTP v 1.1, RFC 2616 (1999)
- Les transferts des données se font en utilisant MIME:
  - utilisé pour l'envoi de la page HTML
  - utilisé aussi pour la requête si il y a des données à envoyer vers le serveur
- Par la suite, nous parlerons principalement de la version 1.0

Format d'une requête HTTP

- 1 ligne pour la requête proprement dite :
  - méthode (GET, HEAD, …)
  - URL (relative ou absolue) de la ressource
  - version du protocole utilisé (HTTP/1.0 ou HTTP/1.1)
- plusieurs lignes d'en-tête (optionnelles) :
  - en-tête générique
  - en-tête de la requête (paramètres de la requête)
  - en-tête de l'entité (en-tête MIME des données)
- 1 ligne vide séparatrice
- plusieurs lignes de données = corps de l'entité (optionnelles):
  - n'est utile que pour les méthodes POST ou PUT
- Exemple:

```
GET http://www.ec-lyon.fr/index.html HTTP/1.0
```

Méthodes HTTP

- GET: récupérer un document
  - il est possible de demander le document que s ’il a été modifié depuis une certaine date (lié aux caches)
- HEAD: récupérer l’en-tête d’un document sans le contenu
  - utile pour récupérer la date de dernière modification ou pour faire des tests de connectivité
- POST: permet d ’envoyer des informations au serveur
  - lié aux formulaires
- PUT, DELETE: permettent d ’ajouter des documents ou d ’en supprimer
  - en général, désactivé
- LINK, UNLINK: permettent d ’établir et enlever des liens
  - dans la norme mais implémenté ?

En-têtes dans la requête

- En-tête générique:
  - **Date:** Date de la requête, peu utilisé
  - Exemple:

```
Date: Mon, 30 Aug 1999 12:54:39 GMT
```

- En-tête de la requête:
  - **From:** adresse mél de l'utilisateur
  - **Referer:** URL de la ressource à l'origine de la requête, c-à-d URL de la page Web qui contient le lien suivi
  - **User-Agent:** identifie le navigateur
  - Exemple:

```
From: rene.chalon@ec-lyon.fr
Referer: http://www.ec-lyon.fr/index.html
User-Agent: Mozilla/4.03 [en] (Win98; I)
```

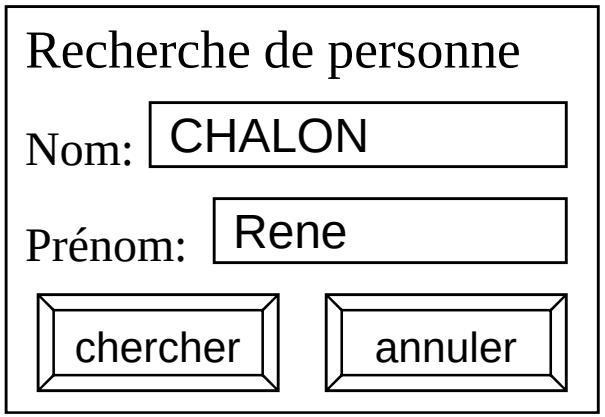
En-tête et corps de l'entité (requête)

- En-tête de l'entité :
  - **Content-Type:** identifie le type de l'information (Cf. types MIME)
  - **Content-Length:** spécifie la taille du corps de l'entité
  - **Content-Encoding:** spécifie un codage éventuel du contenu
  - Exemple :

```
Content-Type: text/html
Content-Length: 2564
Content-Encoding: x-gzip
```

- Corps de l'entité :
  - n'est utilisé qu'avec les méthodes POST ou PUT
  - Exemple:

```
POST http://www.ec-lyon.fr/cgi-bin/search HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 22

NOM=Chalon&PRENOM=Rene
```



Format de la réponse

- 1 ligne pour le statut de la réponse :
  - version du protocole utilisé par le serveur (HTTP/1.0 ou HTTP/1.1)
  - code numérique du statut de la réponse (3 chiffres)
  - statut de la réponse en clair
- plusieurs lignes d'en-tête :
  - en-tête générique
  - en-tête de la réponse
  - en-tête de l'entité (en-tête MIME des données)
- 1 ligne vide séparatrice
- plusieurs lignes de données = corps de l'entité (optionnelles)

- Exemple:

```
HTTP/1.0 200 Ok
```

Statuts de la réponse

- 1xx (Informational) :
  - non utilisé
- 2xx (Success) :
  - requête correctement reçue, traitée
- 3xx (Redirection):
  - il faut une autre requête pour accéder à la ressource
- 4xx (Client Error)
  - requête incorrecte ou mal comprise
- 5xx (Server Error):
  - requête correcte mais non satisfaite

- Exemples :

```
HTTP/1.0 200 Ok
HTTP/1.0 204 No Content

HTTP/1.0 300 Multiple Choices
HTTP/1.0 301 Moved Permanently
HTTP/1.0 302 Moved Temporarily

HTTP/1.0 400 Bad Request
HTTP/1.0 401 Unauthorized
HTTP/1.0 403 Forbidden
HTTP/1.0 404 Not Found

HTTP/1.0 500 Internal Server Error
HTTP/1.0 501 Not Implemented
HTTP/1.0 503 Service Unavailabe
```

En-têtes dans la réponse (1/2)

- En-tête générique:
  - **Date:** date de la réponse du serveur
  - **Pragma: no-cache** ; la réponse ne doit pas être "cachée"
  - Exemple:

```
Date: Mon, 30 Aug 1999 12:54:39 GMT
Pragma: no-cache
```

- En-tête de la réponse:
  - **Location:** URL absolue d'une ressource ; utilisé par codes 3xx
  - **Serveur:** identifie le logiciel du serveur
  - **WWW-Authenticate:** demande d'authentification (Cf. code 401)
  - Exemple:

```
Location: http://www.ec-lyon.fr/index.html.fr
Server: Apache/1.3.6 LV/LM-1.3 (Unix) PHP/3.0.9
WWW-Authenticate: Basic realm="Acces a l'intranet ECL"
```

En-têtes dans la réponse (2/2)

- En-tête de l'entité :
  - **Content-Type:** identifie le type de l'information (Cf. types MIME)
  - **Content-Length:** spécifie la taille du corps de l'entité
  - **Content-Encoding:** spécifie un codage éventuel du contenu
  - **Expires:** limite de validité du document
  - **Last-Modified:** date et heure de dernière modification du document
  - Exemple :

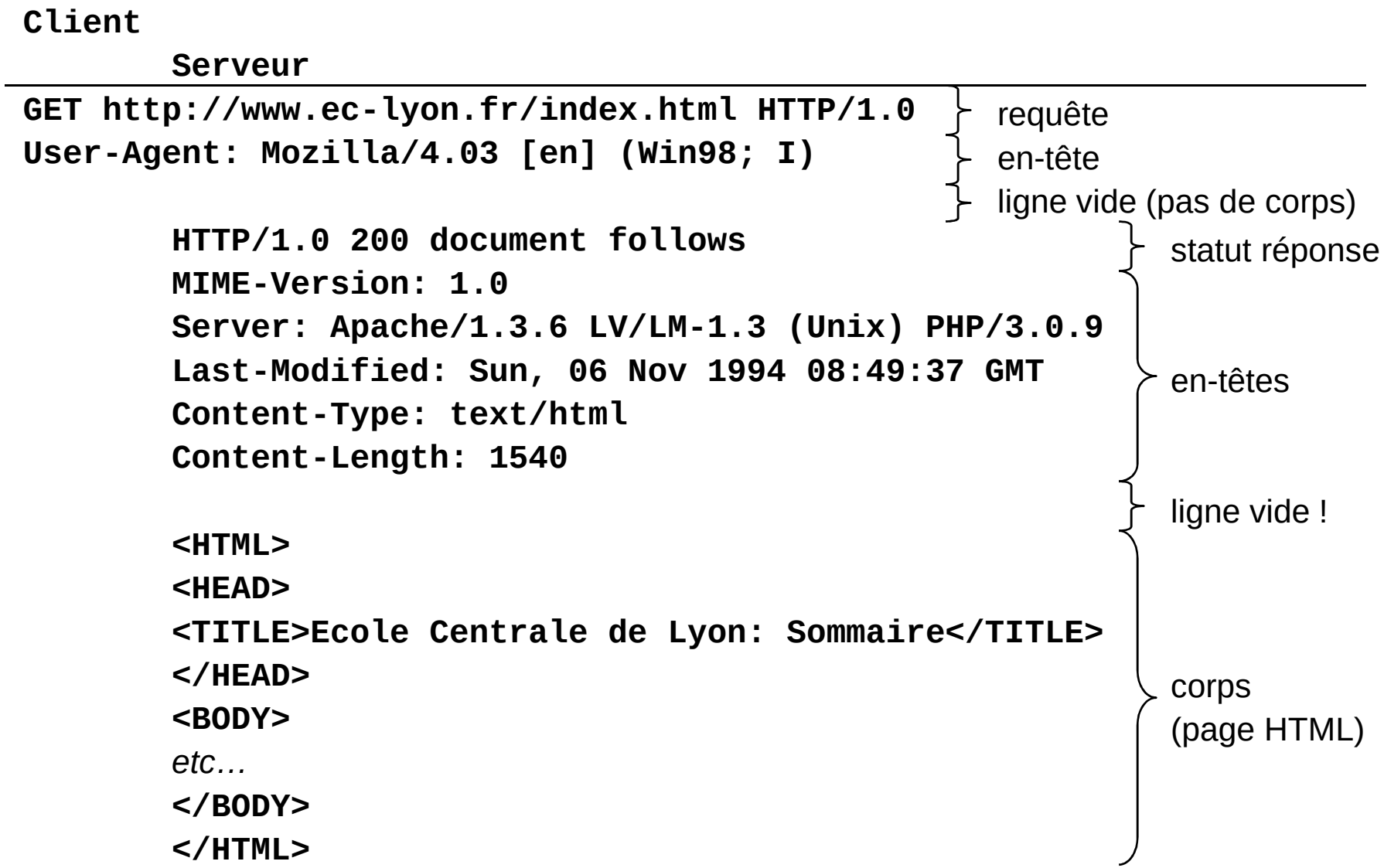
```
Content-Type: text/html
Content-Length: 2564
Content-Encoding: x-gzip
Expires: Sat, 01 Jan 2002 00:00:01 GMT
Last-Modified: Sun, 06 Nov 1994 08:49:37 GMT
```

Corps de l'entité de la réponse

- Une réponse comporte en général une entité avec un corps
  - dans ce cas l'en-tête Content-Length est obligatoire
- Seules les requêtes avec la méthode HEAD retournent une entité dépourvue de corps (mais avec un Content-Length ≠ 0 !)
- Certaines réponses avec des statuts particuliers ne retournent pas de corps (exemple: 401)
- Exemple:

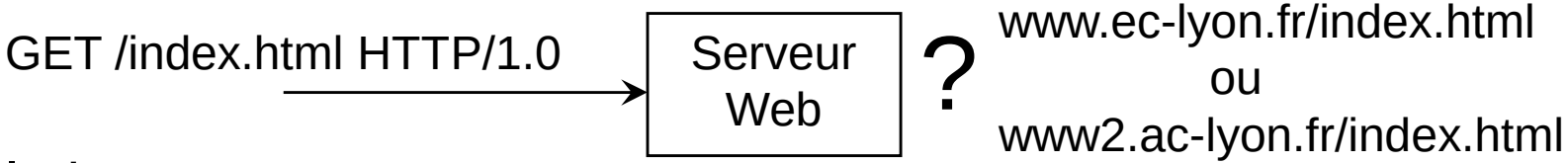
```
HTTP/1.0 200 Ok
Content-Type: text/html
Content-Length: 90

<HTML>
<HEAD> <TITLE>Premier essai</TITLE> </HEAD>
<BODY>
Bonjour !
</BODY>
</HTML>
```

Exemple de connexion HTTP



Problème des URL relatives

- Dans les requêtes il est possible d'utiliser des URL relatives
  - cependant cela pose un problème dans le cas de serveurs virtuels (c-à-d plusieurs serveurs Web sur une même machine physique)
  - Exemple: connexion à 156.18.19.5 et requête



- Solution:
  - n'utiliser que des URL absolues
- Cas de HTTP 1.1 :
  - usage obligatoire de l'en-tête **Host:** qui permet de préciser le serveur auquel on adresse la requête; exemple:

```
GET /index.html HTTP/1.1
Host: www.ec-lyon.fr
```

  - si on essaye: **`GET http://www.ec-lyon.fr/index.html HTTP/1.1`** le serveur retourne le statut: **`400 Bad Request`**

Problème des connexions multiples (1/2)

- la connexion est systématiquement coupée par le serveur lorsqu'il retourne la réponse :
  - peu performant ; si une page comporte 5 images incluses il faut en tout 6 connexions TCP (1 pour la page et 5 pour les images)
  - pour améliorer Netscape ouvre plusieurs connexions simultanées (jusqu'à 4) mais cela augmente la charge du réseau et du serveur
- Solution :
  - il est possible d'utiliser un en-tête **Connection: Keep-Alive** dans la requête pour demander au serveur de garder la connexion ouverte
  - la réponse comporte un en-tête **Keep-Alive:** avec 2 paramètres:
    - ➔ un time-out exprimé en secondes ; sans nouvelle requête dans ce laps de temps, la connexion est coupée par le serveur
    - ➔ un nombre maximum de requêtes au-delà duquel le serveur coupe la connexion

Problème des connexions multiples (2/2)

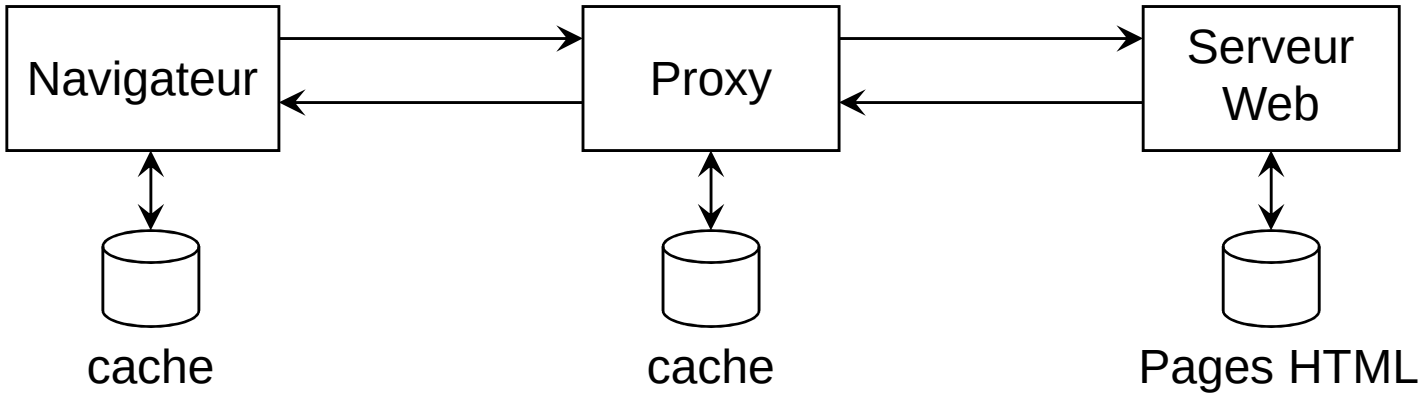
- Exemple:

```
HEAD http://www.ec-lyon.fr/index.html HTTP/1.0
Connection: Keep-Alive

    HTTP/1.0 200 document follows
    MIME-Version: 1.0
    Keep-Alive: time-out=15, max=100
    Connection: Keep-Alive
    etc...
```

- Cas de HTTP 1.1:
  - la connexion reste ouverte par défaut
  - plusieurs requêtes peuvent être envoyées avant que la première réponse ne soit parvenue (mode pipe-line)
  - Pour fermer la connexion:
    - utilisation de l'en-tête **Connection: Close**
    - time-out

Gestion des caches (1/4)

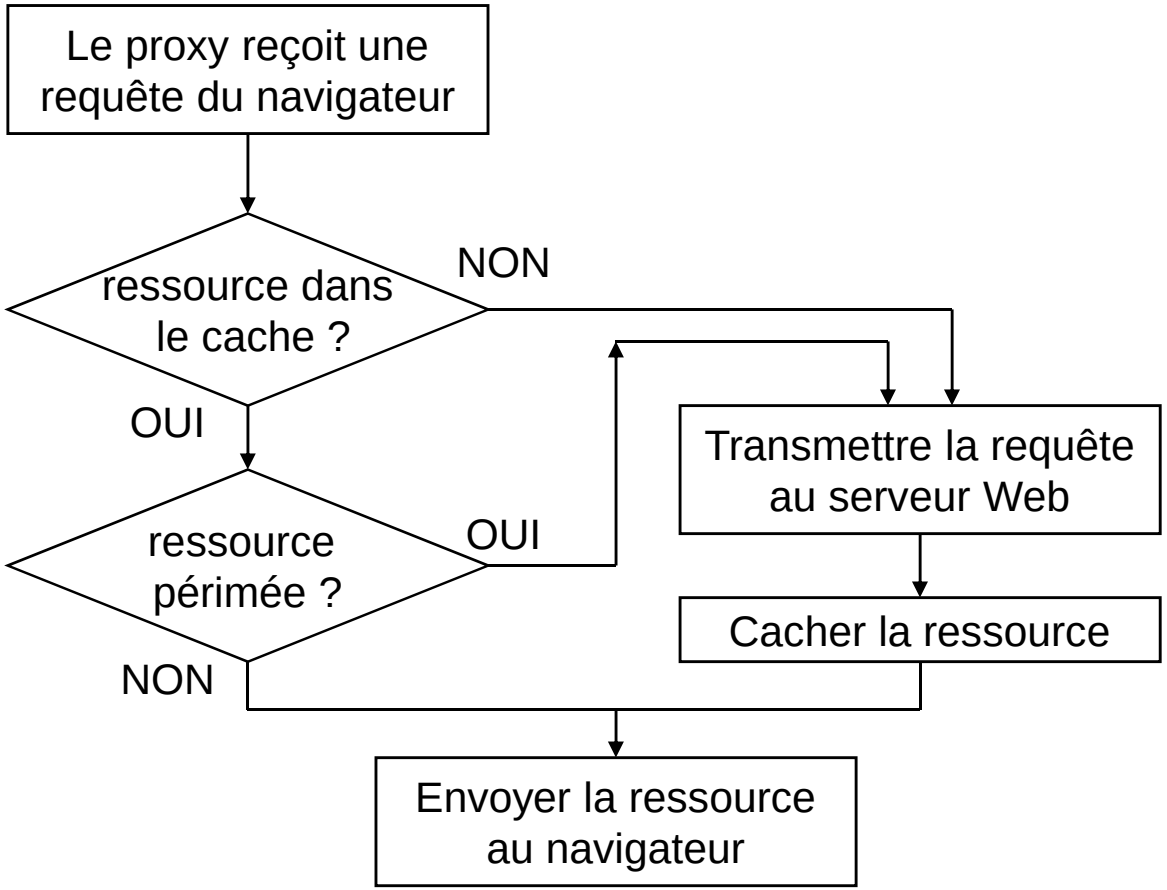
- Principe des caches :



  - avant de demander une ressource à un serveur on regarde si on a pas déjà une copie plus rapidement accessible dans un cache
  - les navigateurs ont des caches (en mémoire et sur disque)
  - on peut utiliser des serveurs mandataires (proxy serveur) qui comportent des caches en général assez gros et partagés par de nombreux utilisateurs ce qui augmente les chances de trouver la ressource dans le cache
- Problème: la ressource cachée peut être obsolète

Gestion des caches (2/4)

- Procédure d'utilisation d'un cache (sur un proxy):



Gestion des caches (3/4)

- Déterminer si une ressource dans un cache est périmée :
  - utilisation de l'en-tête **Expires:** renvoyé avec la ressource et stocké dans le cache
  - redemander l'en-tête de la ressource au serveur avec la méthode HEAD et comparer la date (en-tête Last-Modified:) ou la taille (en-tête Content-Lenth:) avec les valeurs dans le cache ; en cas de différence il faut refaire une requête GET
- GET conditionnel:
  - combine les effets d'un HEAD suivi d'un GET
  - utilise l'en-tête **If-Modified-Since:**
  - Exemple:

```
GET http://www.ec-lyon.fr/index.html HTTP/1.0
If-Modified-Since: Wed, 07 Jul 1999 00:00:01 GMT

        HTTP/1.0 304 Not Modified
        Server: Apache/1.3.6 LV/LM-1.3 (Unix) PHP/3.0.9
```

Gestion des caches (4/4)

- Entité non cachable:
  - Si le serveur ne désire pas qu'une ressource soit cachée:
    - ➔ inclure un en-tête **Pragma: no-cache** dans la réponse
    - ➔ inclure un en-tête **Expires:** avec un date antérieure à la date courante
- Cas de HTTP 1.1:
  - la gestion par date peut s'avérer imprécise ou insuffisante
  - ajout d'un en-tête **ETag:** dans la réponse donnant un identifiant unique (modifié quand la ressource change)
  - pour vérifier si la ressource a changée, le proxy utilise un GET conditionnel avec un en-tête **If-None-Match:** en passant la valeur du ETag qu'il a dans son cache
  - ajout d'un en-tête **Cache-Control:** permettant de gérer finement le cache (nombreuses possibilités)