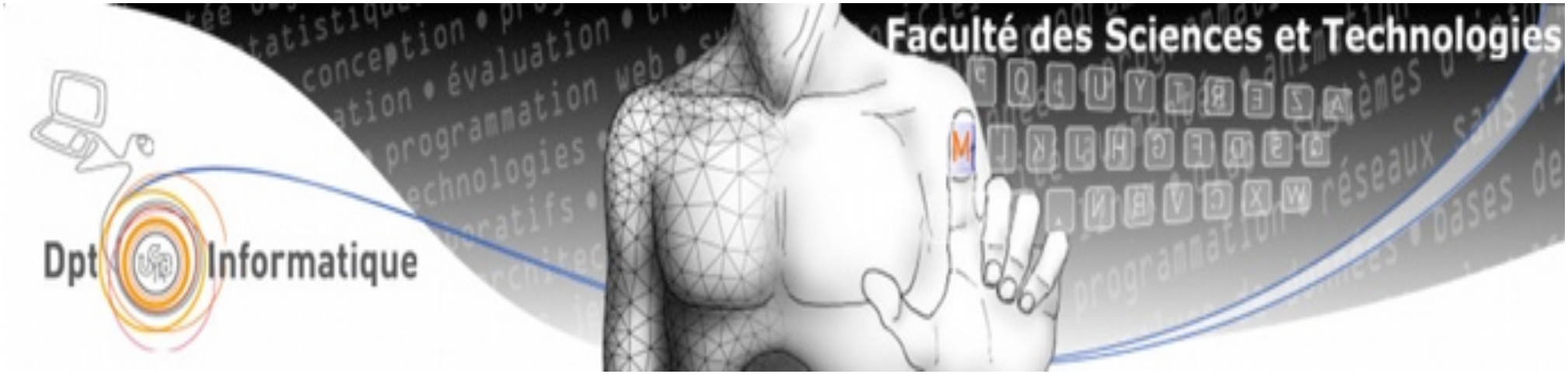


DIU-EIL : Réseaux, Internet et Web

UNIVERSITÉ DE LYON

Université Claude Bernard Lyon 1

juin 2020

Olivier Glück

Université LYON 1 / Département Informatique

Olivier.Gluck@univ-lyon1.fr

http://perso.univ-lyon1.fr/olivier.gluck

Copyright

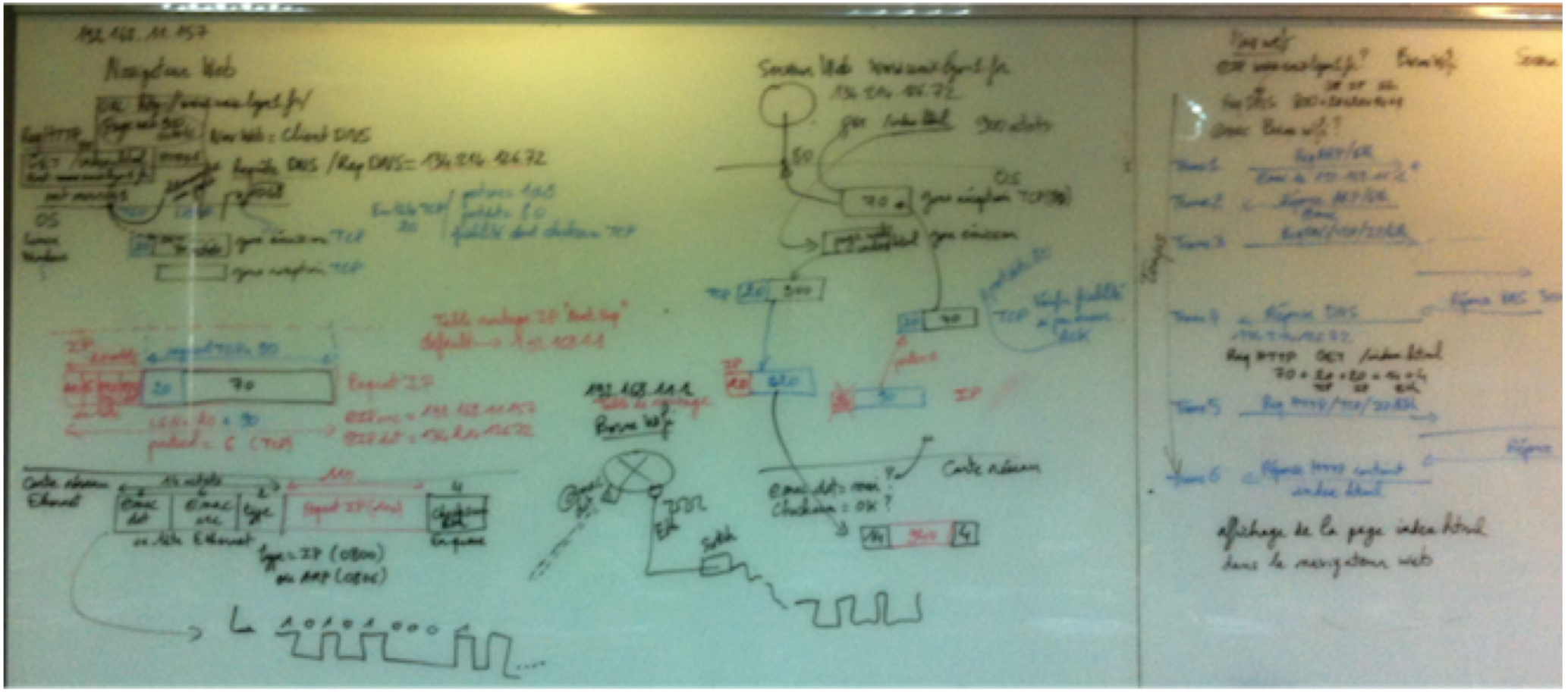
Plan du cours

- Internet, les réseaux et le web
- Le protocole HTTP, méthodes GET et POST
- Le modèle Client/Serveur
- Le protocole UDP et la rapidité
- Le protocole TCP et la fiabilité
- Le protocole IP
- Les protocoles Ethernet, ARP et ICMP
- Les principales applications d'Internet
- Pour aller plus loin sur IP…
- Pour aller plus loin sur HTTP

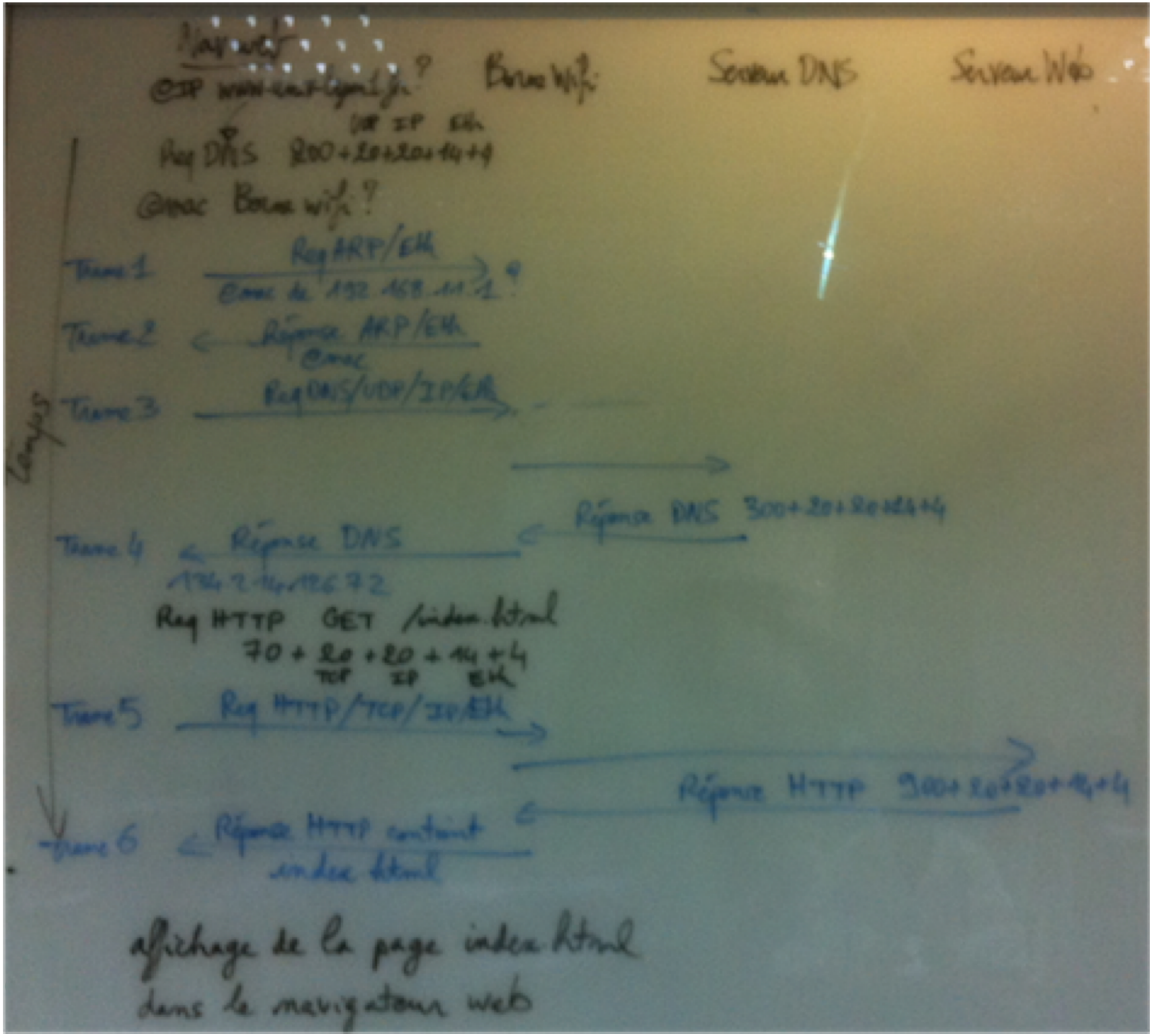
Objectifs du module

- Vous faire découvrir les réseaux informatiques et le fonctionnement d'Internet et ses applications
- Vous faire comprendre tout ce qui se passe quand un navigateur web demande une page web à un serveur web
- Vous faire comprendre comment fonctionne votre connexion à Internet chez vous
- Vous initier à la configuration d'un réseau informatique
- Vous donner des éléments pour résoudre des pannes simples du type « Internet ne marche pas ! »

A la fin du module !



Cette image et la suivante correspondent à la photo d'un tableau dont le contenu n'a pas d'importance à ce stade


Client
Borne Wifi
Serveur DNS
Serveur Web
Req DNS
Req ARP/EH
Trame 1
Trame 2
Réponse ARP/EH
Trame 3
Req DNS/UDP/IP/EH
Temps
Réponse DNS
Trame 4
Réponse DNS
Req HTTP GET /index.html
Trame 5
Req HTTP/TCP/IP/EH
Réponse HTTP
Trame 6
Réponse HTTP contient index.html
affichage de la page index.html dans le navigateur web

Contenus des TP

- TP1 et TP2 : initiation aux réseaux informatiques
  - Mettre en place une architecture réseau permettant des échanges entre un navigateur web et un serveur web
  - Concevoir dans un émulateur de réseaux une architecture composée d'un PC client avec navigateur web, d'une box, de deux routeurs intermédiaires, d'un serveur web et d'un serveur DNS
  - Configurer les équipements : adressage des cartes réseaux, routage, NAT, DNS
  - Tester la configuration et le bon fonctionnement de l'architecture
  - Observer les échanges HTTP, DNS, TCP, IP, ARP, Ethernet et être capable de les analyser

UNIVERSITÉ DE LYON
Université Claude Bernard Lyon 1
Faculté des Sciences et Technologies
Dpt Informatique

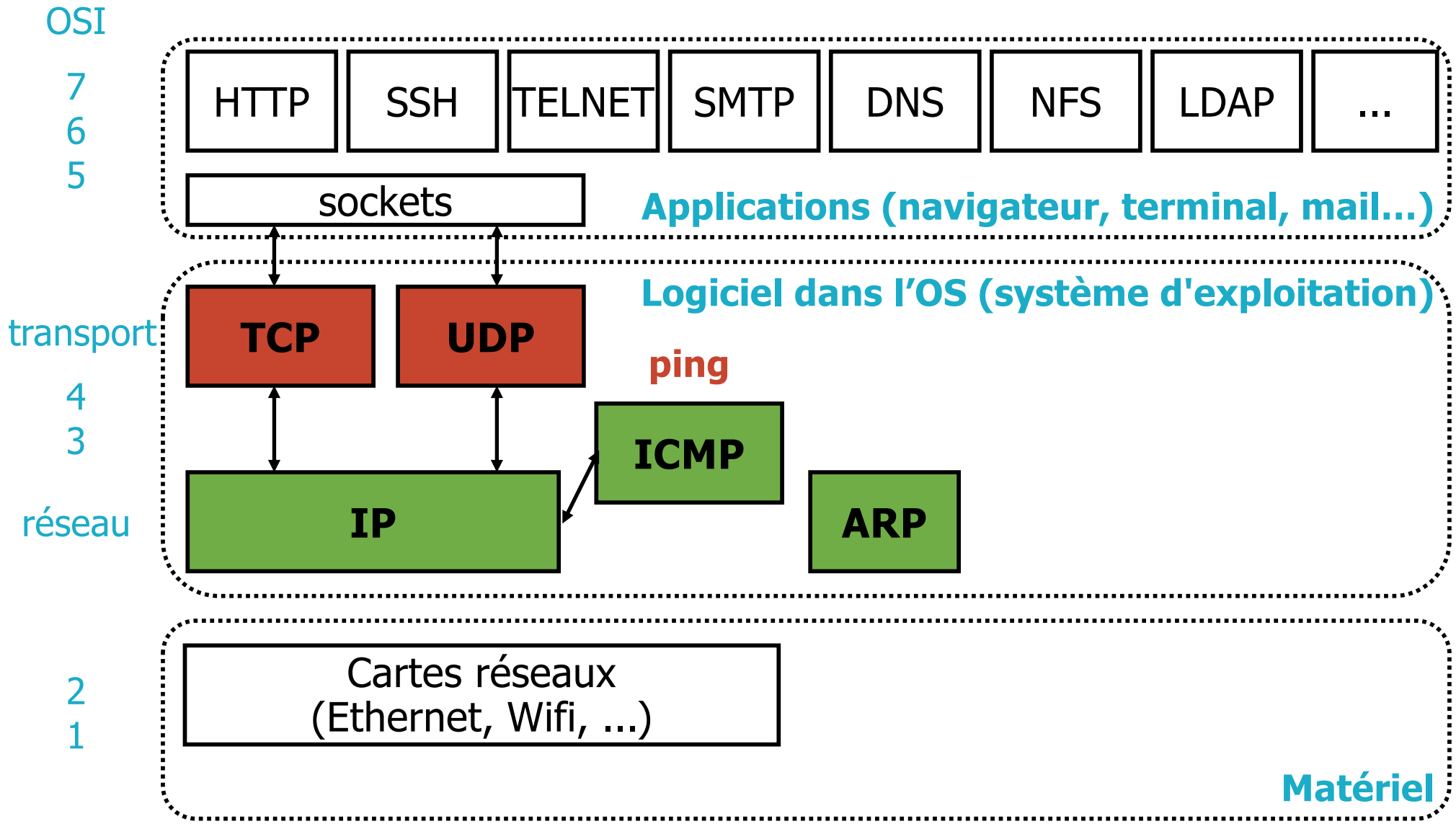
Internet, les réseaux et le Web

Internet, supports de transmissions, composants
Qu'est-ce que le web ?
Format simple des URL
Le navigateur et le serveur web
Pages statiques et dynamiques

Qu'est-ce qu'Internet ?

- Internetworking : un réseau de réseaux, le réseau public mondial, des équipements d'interconnexion
  - Des liaisons de toutes sortes (câbles, satellites…)
  - Répéteurs, commutateurs, routeurs
- Une architecture réseau appelée TCP/IP
  - Des protocoles de communication très variés
    - Réalise un service grâce à un algorithme
    - Définit le format des messages échangés (Requêtes/Réponses)
  - Des opérateurs
    - Possèdent les infrastructures du réseau (liaisons et équipements)
    - Gèrent et administrent le réseau
- Pour faire quoi ?
  - Permettre aux utilisateurs d'exécuter des applications
  - Grâce à des fournisseurs d'accès à Internet (FAI ou ISP)

L'architecture TCP/IP : protocoles d'Internet



Que se passe-t-il en 60 secondes sur Internet ?

Snapchat
527.760 photos partagées

Uber
1.389 courses

WhatsApp
20,8 millions de messages

YouTube
2,78 milllions de vidéos regardées

Tinder
972.222 swipes

Google
2,4 millions de recherches

Facebook
701.389 connexions

Netflix
69.444 d'heures passées sur Netflix

Mails
150 millions de mails envoyés

LinkedIn
120 nouvelles inscriptions

Twitter
347.222 nouveaux tweets

Instagram
38.194 nouveaux post

Spotify
38.052 heures de musique

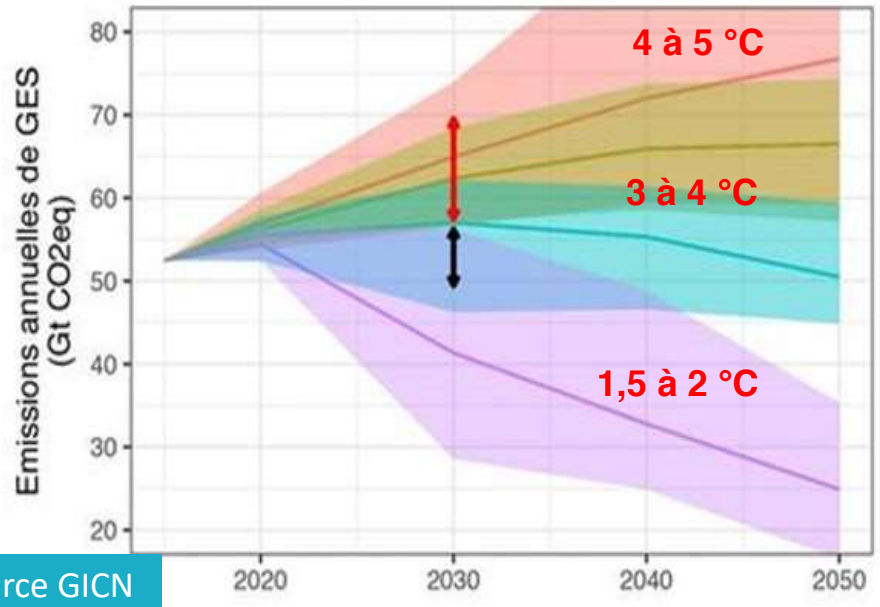
* données de 2016.

Source: Excelacom

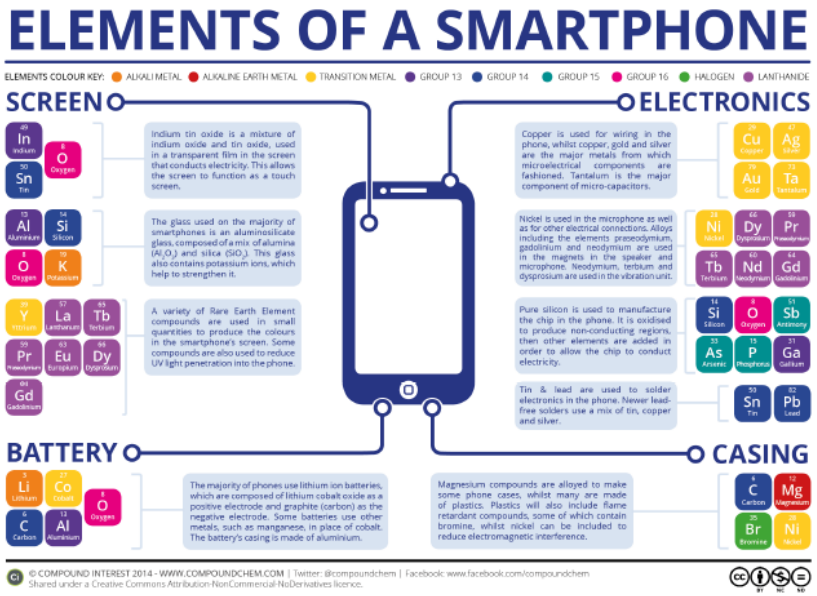
JDN statista

Internet et la planète (1)

- L'envoi d'un simple e-mail : 25Wh, 20g eq CO_2 (Ademe)
  - 100 000 milliards d'e-mail par an (1 smartphone = 80kg eq CO_2)
- Le numérique :
  - 10% de la consommation électrique mondiale, +9% par an (non soutenable car COP21 recommande -5% par an)
  - Eq CO_2 = aviation (8%) en 2013 = automobiles (20%) en 2025
  - 9000 milliards de $ = 60% PIB Europe (GAFAM = 4500 milliards)

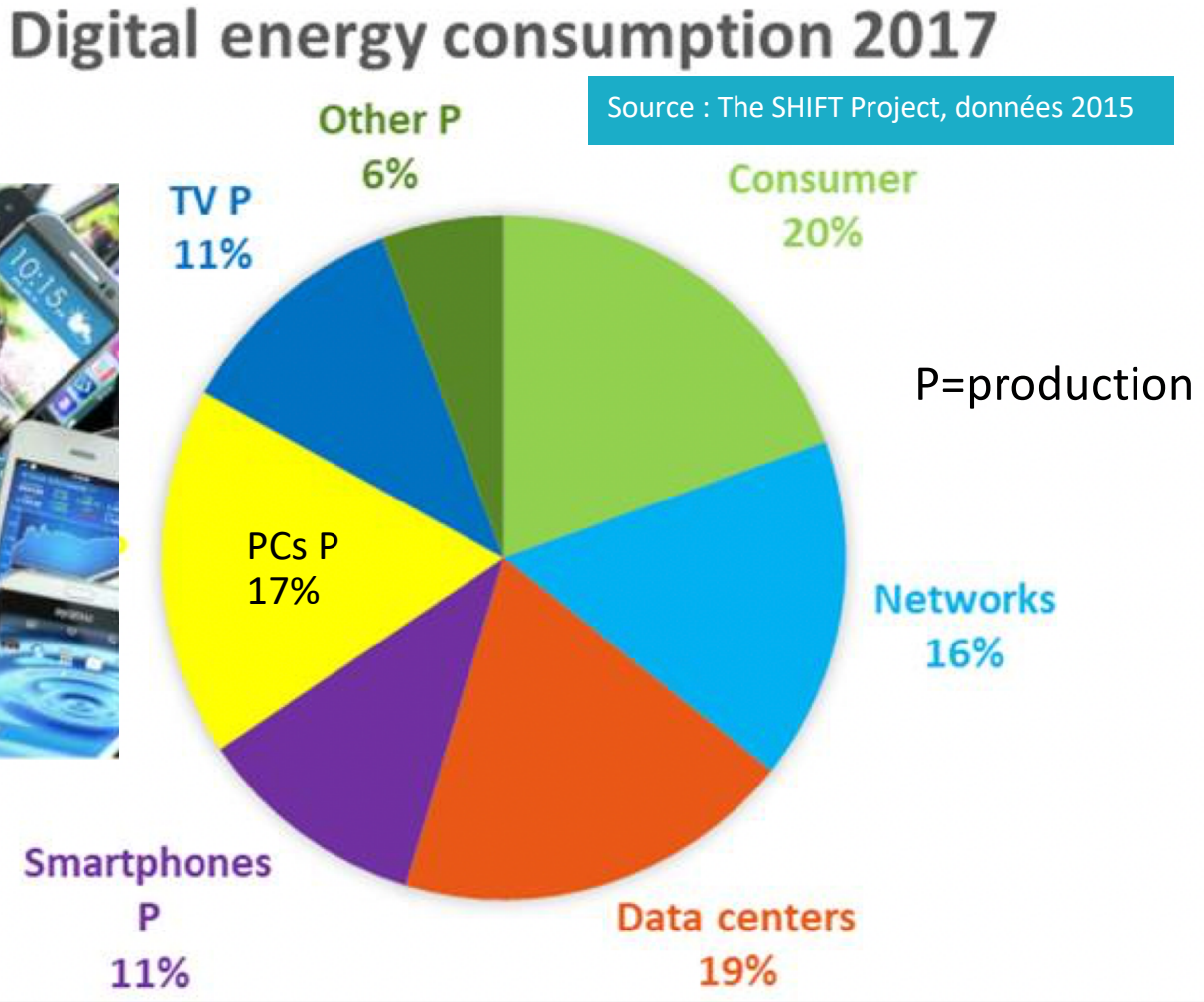


Source GICN



Internet et la planète (2)

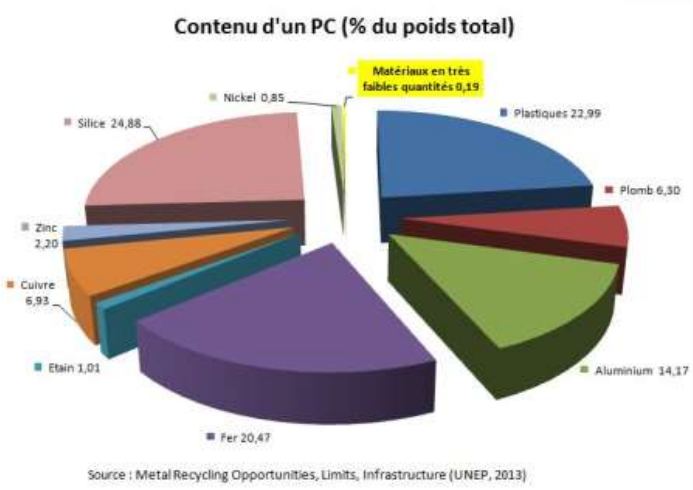
- 45% conso énergie = fabrication des équipements
  - Durée de vie : Smartphone = 18 mois, PC = 3 ans (obsolescence programmée), problématique de gestion des déchets + pollution

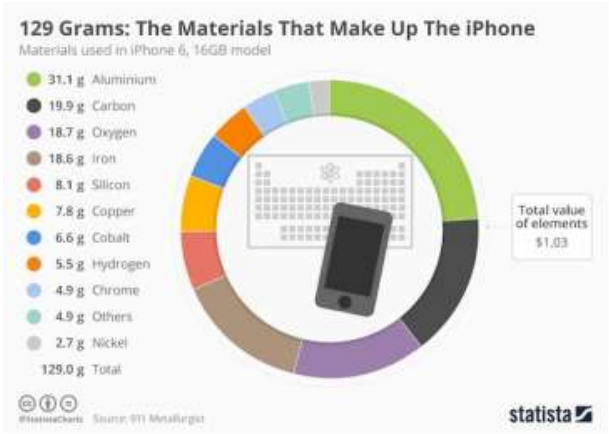


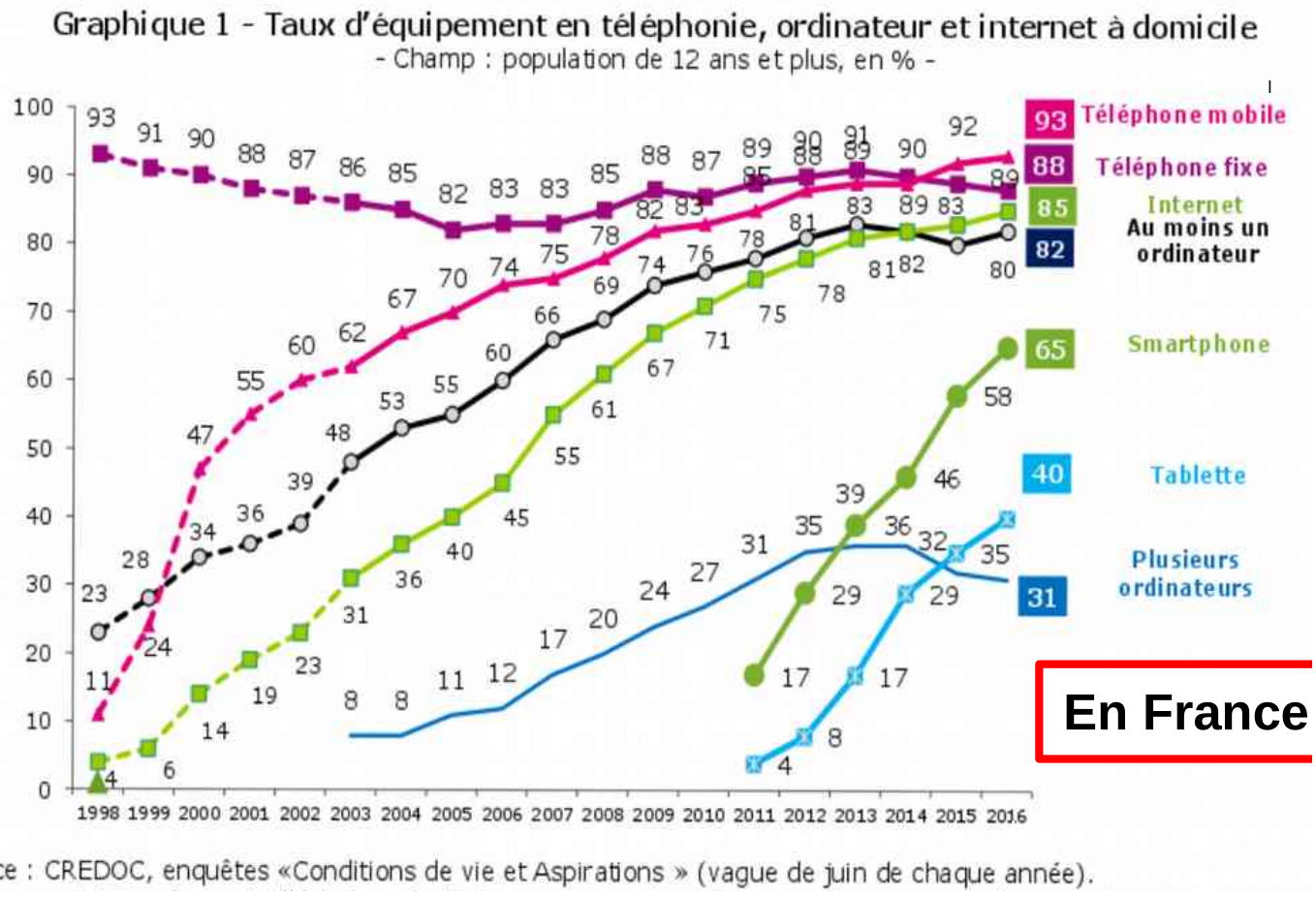
Internet et la planète (3)

- PCs en service dans le monde en 2015 : 2 milliards
- Vente de téléphones : 2 milliards/an soit 80/seconde

http://www.internetlivestats.com





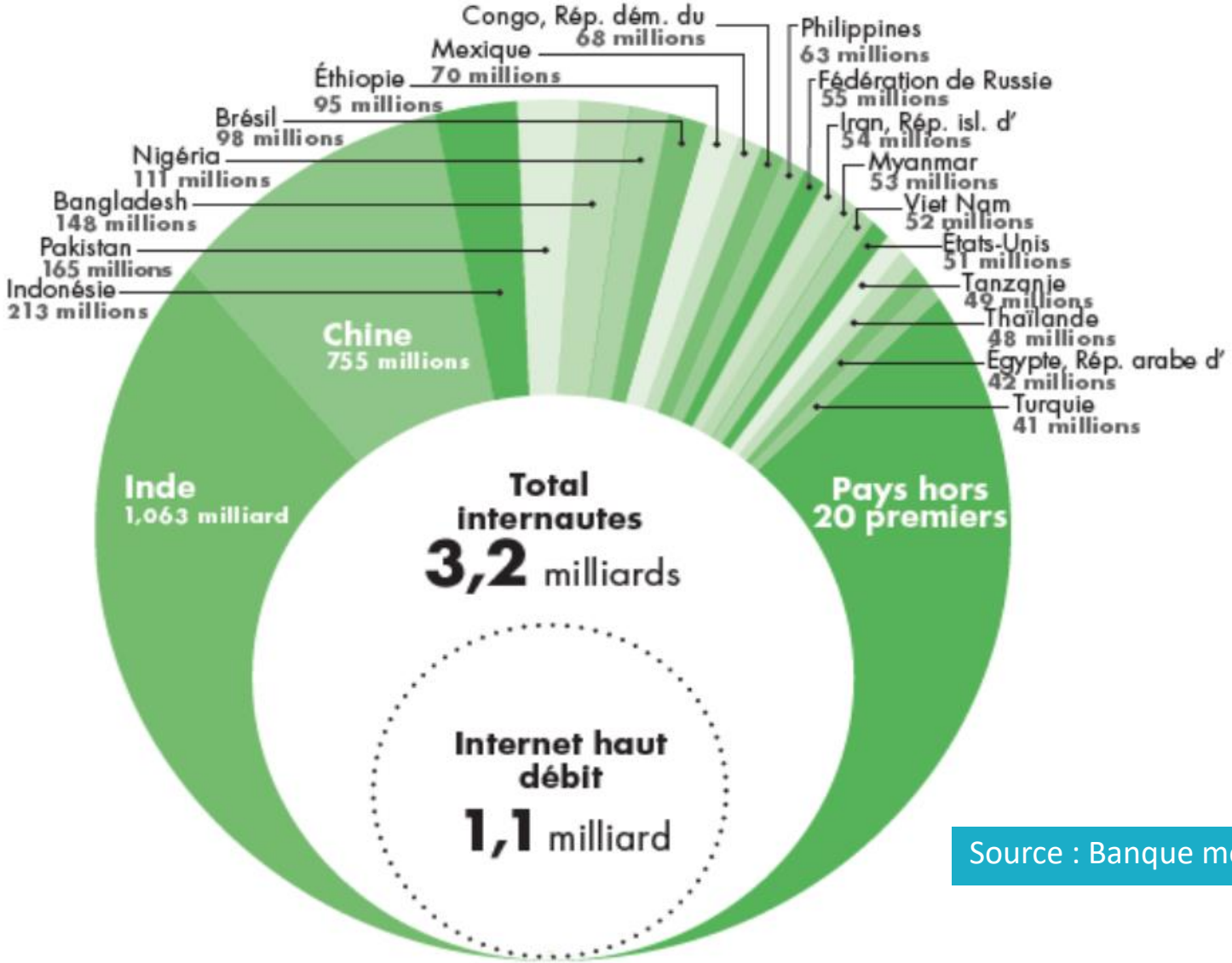


Une consommation numérique toujours plus polarisée

Nombre d'équipements connectés par personne	2016	2021	Croissance annuelle
Asie-Pacifique	1,9	2,9	8,3%
Europe centrale et orientale	2,5	3,8	9,1%
Amérique latine	2,1	2,9	7,0%
Moyen-Orient et Afrique	1,1	1,4	5,4%
Amérique du Nord	**7,7**	**12,9**	**11,0%**
Europe de l'Ouest	**5,3**	**8,9**	**10,9%**
Global	2,3	3,5	8,5%

Regional split 2016	Population (millions)	Devices per capita	Traffic per capita (GB/mth)	GES (MtCO2e)	GES per capita (kgCO2e)
USA	322	7,8	97,0	331	**1027**
Western Europe	415	5,3	34,0	201	**486**
Japan	126	6,3	35,0	60	**474**
China	1374	2,5	12,0	400	**291**
Developing countries	3700	1,1	*1,5*	238	**64**
World	7500	2,3	13,0	**1630**	**217**

Un constat de fracture numérique



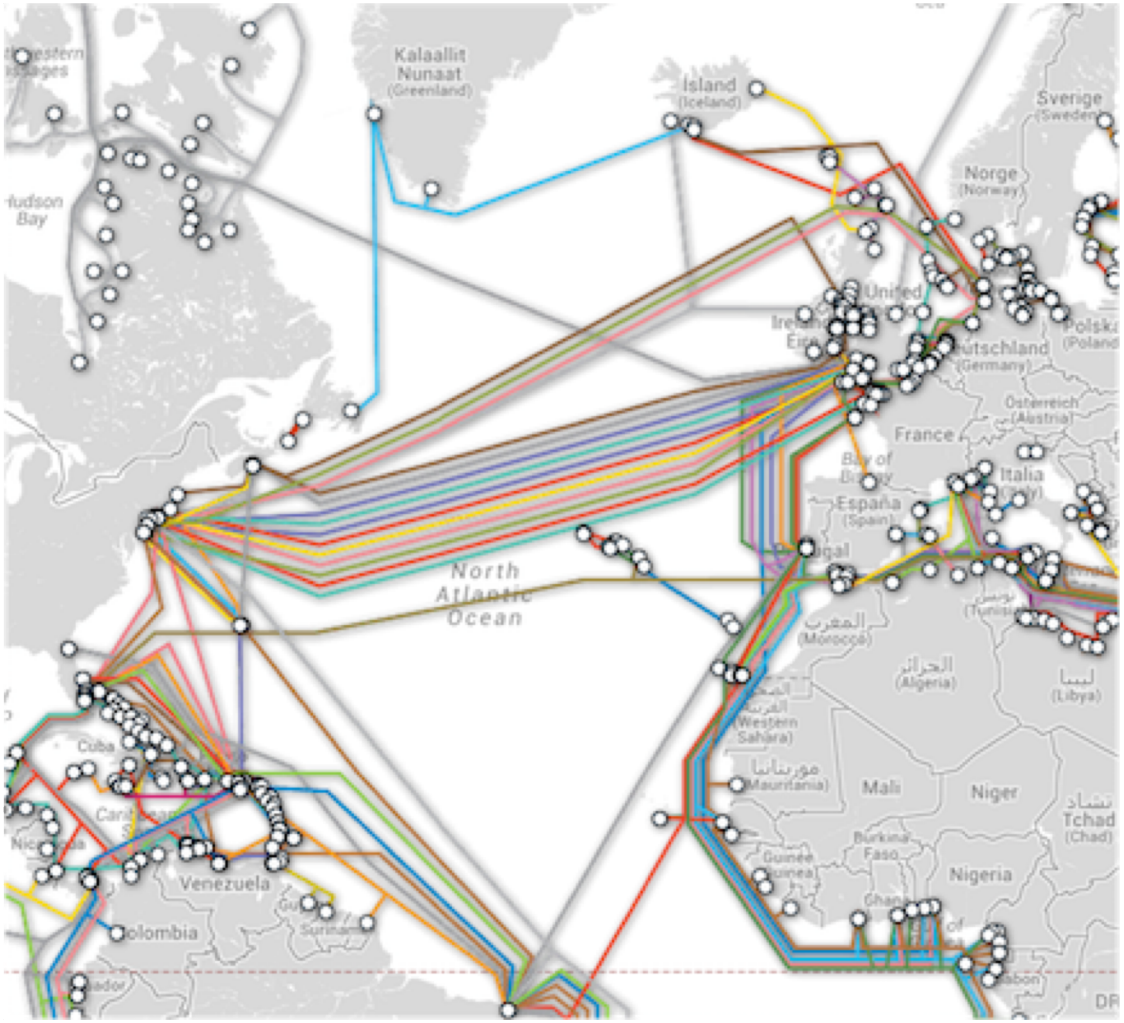
Source : Banque mondiale 2016

Population mondiale non connectée

60% (↗) des habitants de la planète n'ont pas accès à Internet

Les liaisons sous-marines

Source http://www.submarinecablemap.com/



Submarine Cable List

Apollo

Email link

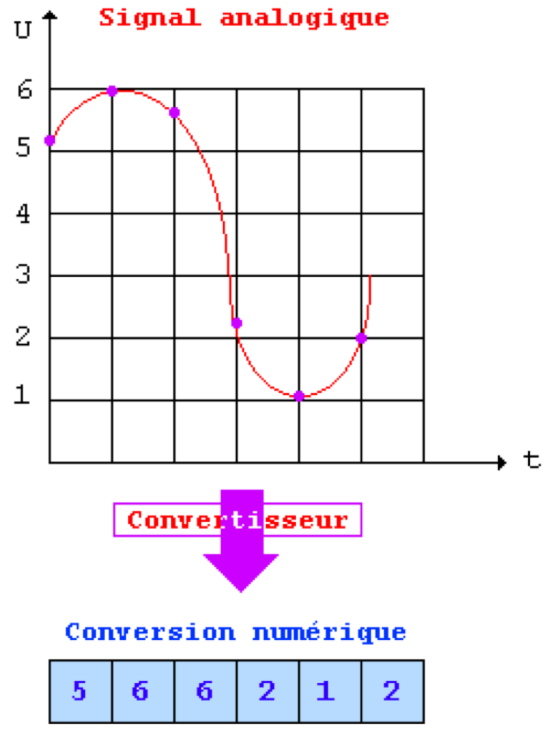
RFS: February 2003
Cable Length: 13,000 km
Owners: Alcatel-Lucent, Vodafone
URL: http://www.apollo-scs.com

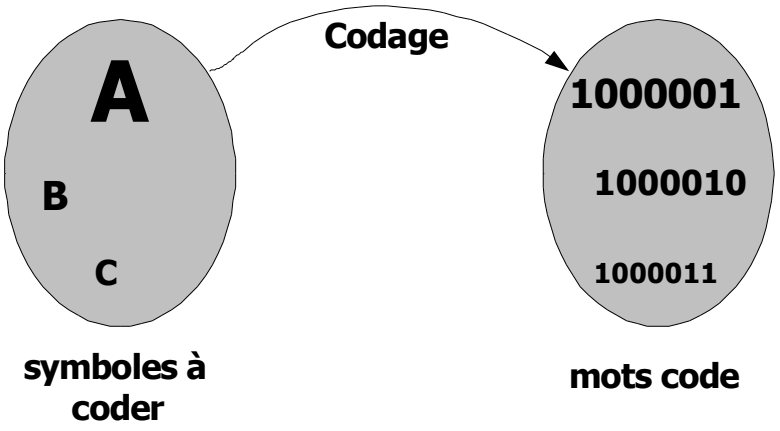
Landing Points

Bude, United Kingdom
Lannion, France
Manasquan, New Jersey, United States
Shirley, New York, United States

Traitement de l'information avant envoi

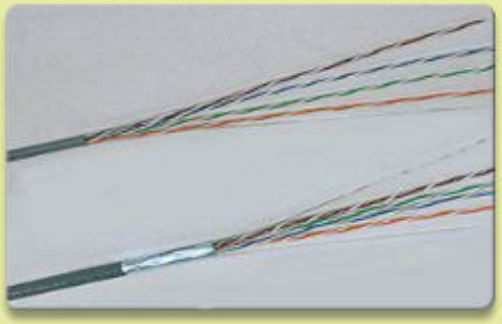
- Il faut associer une valeur binaire à chaque élément d'information
  - Numérisation de l'information pour des données continues (échantillonnage)
  - Codage de l'information pour des données discrètes (code Baudot, code ASCII…)





Les supports de transmission

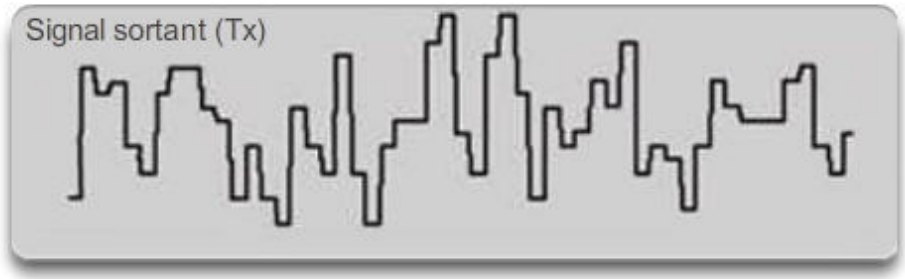
Cuivre



Fibre optique

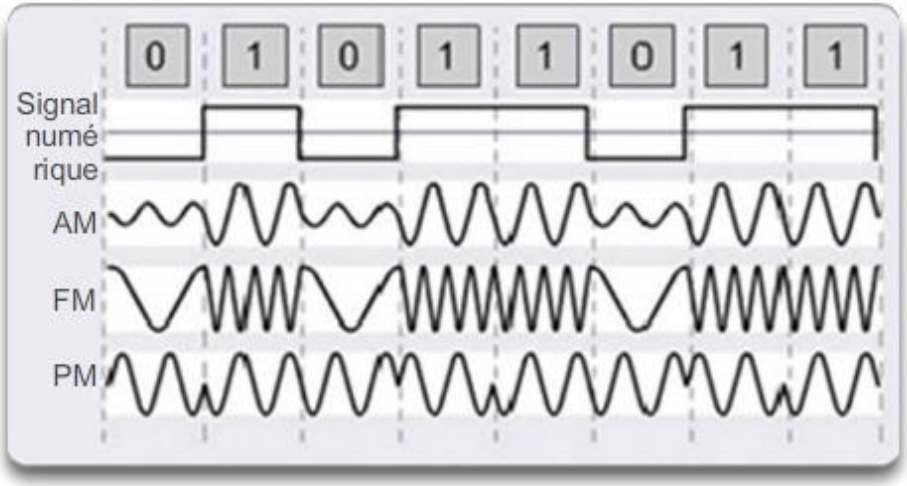
Sans fil

Les supports de transmission



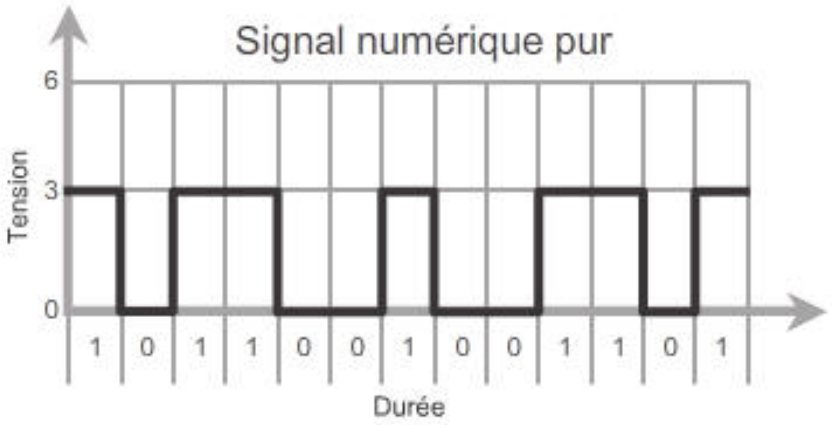
Signaux électriques -
câble en cuivre

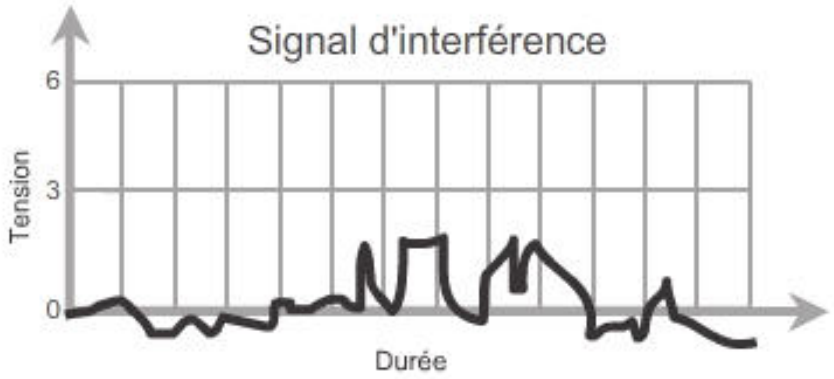
Impulsion lumineuse -
câble à fibre optique



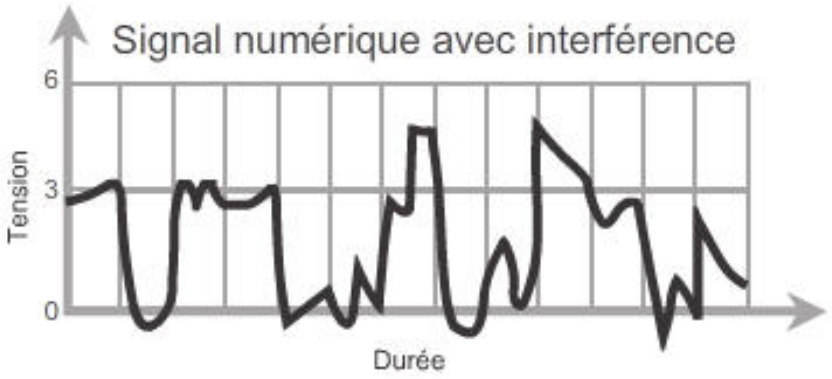
Signaux hyperfréquence -
sans fil

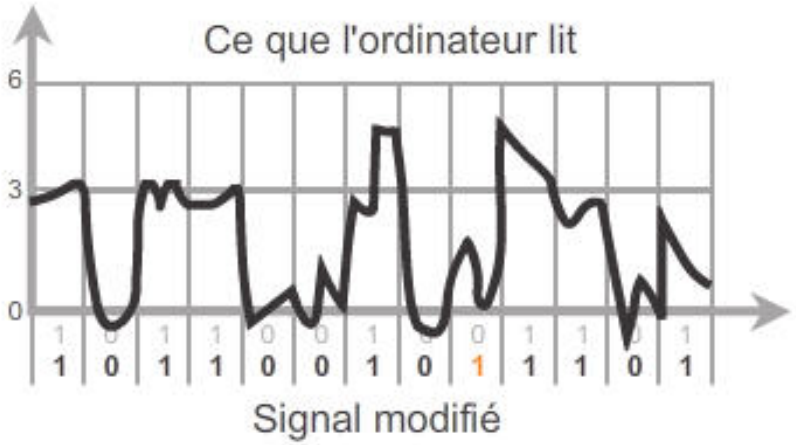
Les supports de transmission



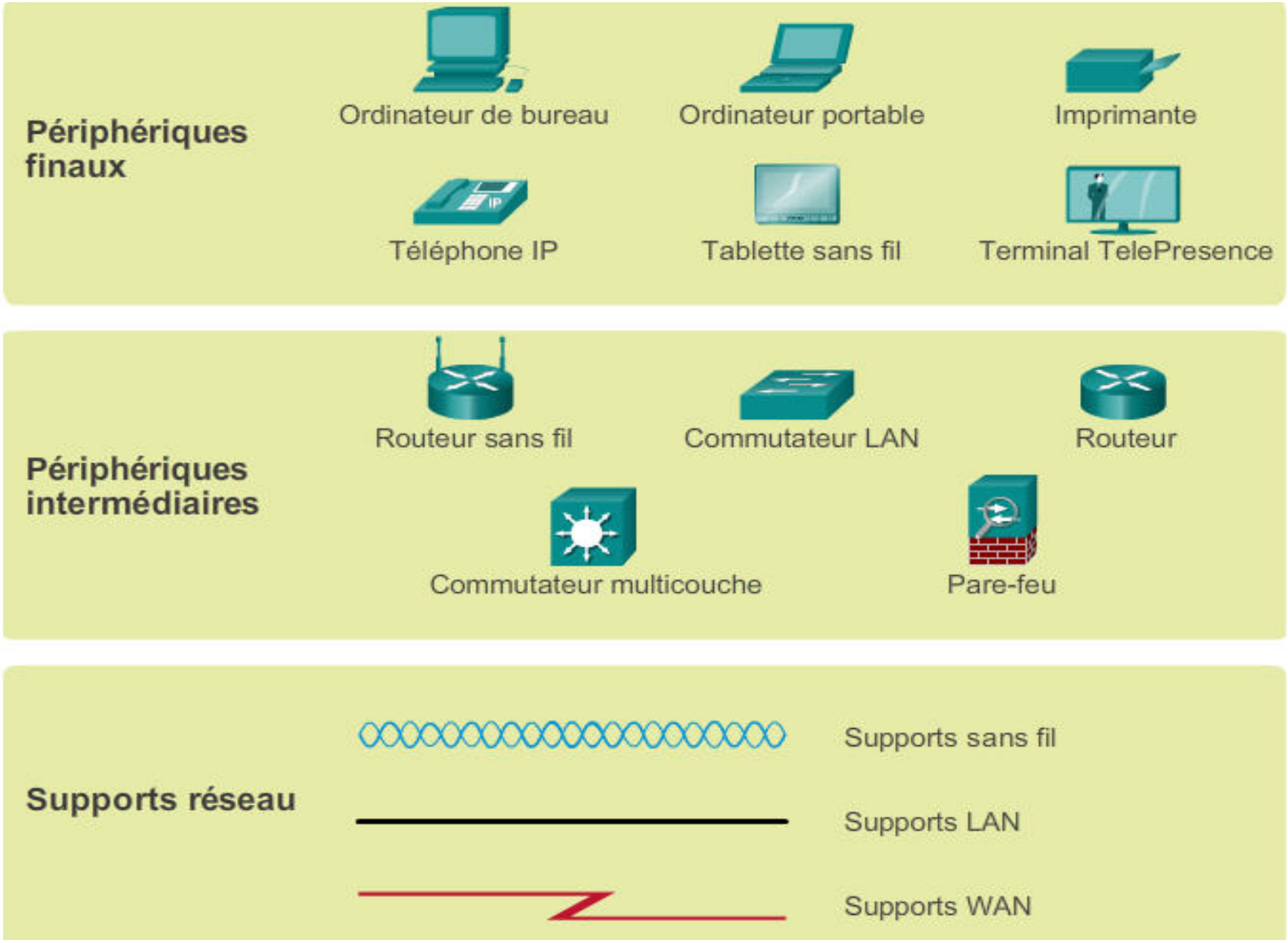


Exemple d'interférences sur un support cuivre

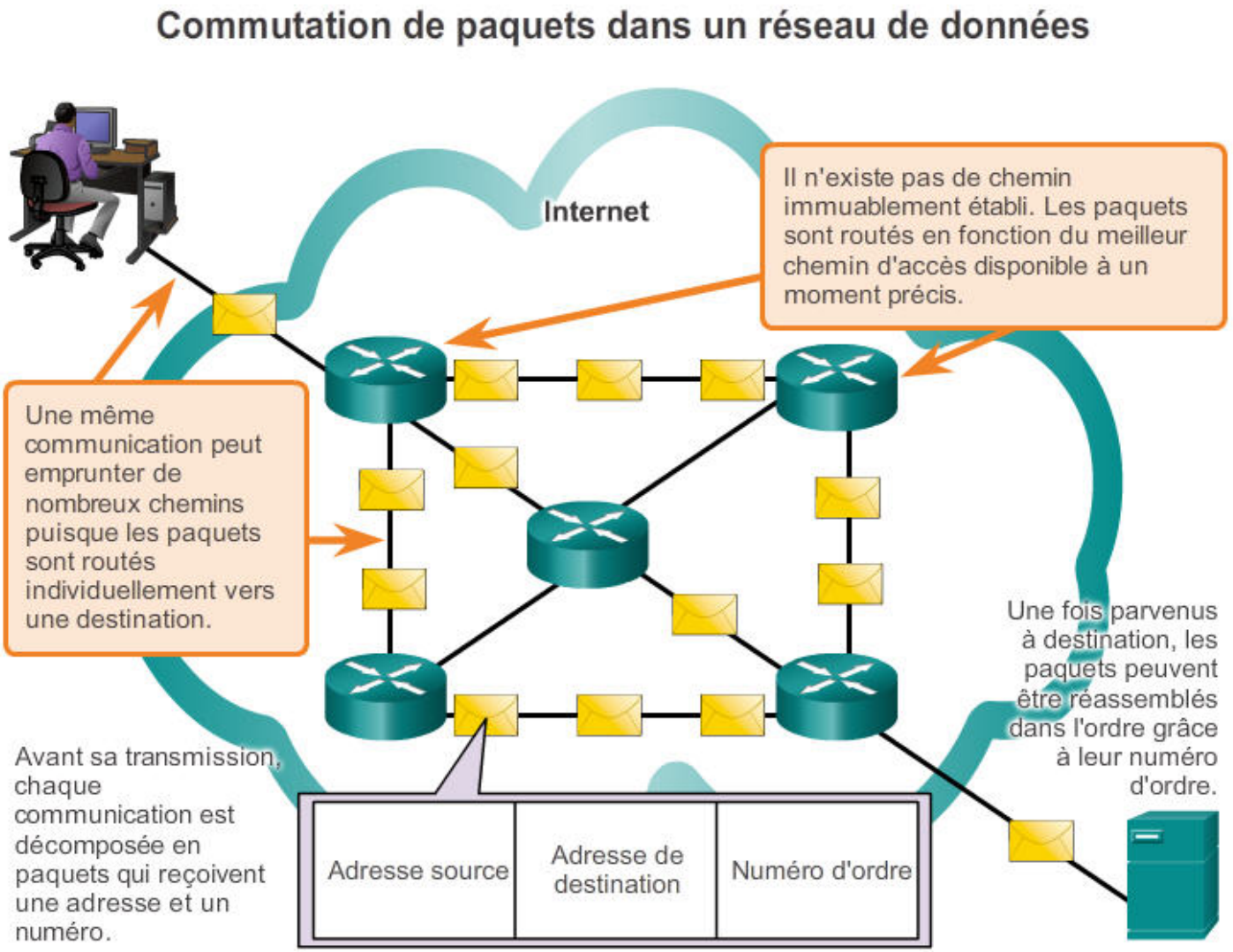




Les composants d'un réseau



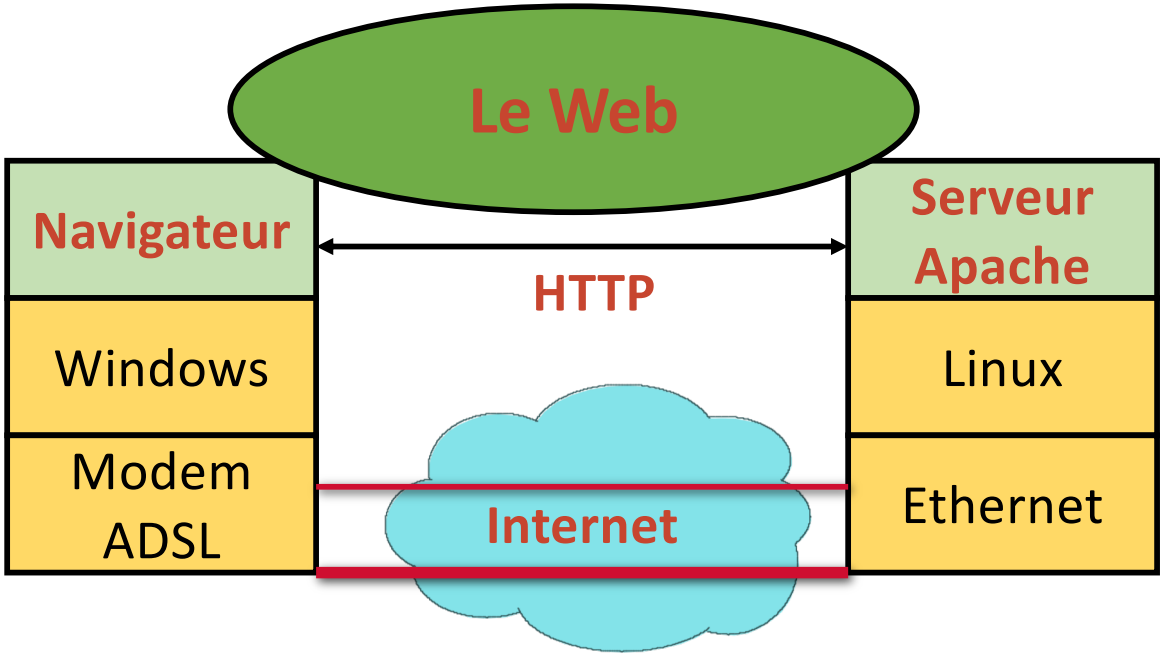
Les réseaux d'ordinateurs



Qu'est-ce que le web ? (1)

- Une application d'Internet qui permet le partage de documents liés entre eux et appelés "pages web"
- Une page web peut contenir du texte, des images, des programmes, des liens vers d'autres pages web…
- Fonctionne en mode Client/Serveur au dessus de l'architecture TCP/IP

L'application est répartie sur le client et le serveur qui dialoguent selon un protocole applicatif spécifique



Qu'est-ce que le web ? (2)

- Des clients : les navigateurs qui font l'interface avec l'humain (Firefox, Internet Exploreur, Chrome, Safari…)
- Des serveurs : ils hébergent les pages web et répondent aux demandes des clients (Apache, Microsoft IIS...)
- Le web est né en 1994 avec la création du W3C (**WWW Consortium**) par le CERN et le MIT (Tim Berners-Lee président) qui s'occupe de la normalisation et des développements du web
- Sa popularité est due à :
  - Des interfaces graphiques conviviales
  - Une très grande quantité d'informations très diverses

Qu'est-ce que le web ? (3)

- Le web repose sur 3 concepts :
  - **L'URL** : permet au client de désigner la page demandée
    - Uniform Ressource Locator : Comment ? Où ? Quoi ?

    Comment ? **Où ?** **Quoi ?**

    `http://etu.univ-lyon1.fr/licence/lifasr2.html`
  - **HTTP** : permet de définir le format et la signification des messages échangés entre le client et le serveur (protocole)
    - **Requête** HTTP : du client vers le serveur, pour demander une page web
    - **Réponse** HTTP : du serveur vers le client, pour répondre au navigateur
  - **HTML**, **CSS**, **PHP**, Javascript… : les langages du web
    - **HTML** : permet de décrire le contenu d'une page web, interprété par le navigateur web pour afficher la page et demander les objets incorporés
    - **CSS** : permet de définir les styles de la page (format, couleurs, positions...)
    - **PHP** : permet d'exécuter un programme par le serveur
    - **Javascript** : permet d'exécuter un programme par le navigateur

Format d'une URL

`proto://host_name:port/path?arguments`

- la racine "**`/`**" de **`path`** est définie par la configuration du serveur Web

  (**Attention :** à ne pas confondre avec la racine du système de fichiers sur le serveur)

- **`/path`** peut contenir une étiquette (point d'ancrage)

`http://www.monsite.fr/projet/doc.html`**`#label`**

- **`arguments`** permettent de passer des informations à des programmes s'exécutant sur le serveur

  Par exemple, **?action-joueur=gauche** dans le jeu 2048

Le navigateur web (1)

- Analyse l'URL demandée et récupère le nom du serveur
- Demande au DNS l'adresse IP de la machine serveur
- Etablit une connexion TCP vers le numéro de port de l'URL (80 par défaut)
- Fabrique la requête HTTP et l'envoie au serveur
- Réceptionne la réponse HTTP
- Interprète le code HTML reçu : commandes de formatage et de mise en forme (police, gras, couleurs...)
- Demande les objets incorporés au serveur et affiche la page correctement formatée
- Exécute les programmes Javascript s'il y en a

Le navigateur web (2)

- Pour faire l'affichage de la page, il se base sur
  - les valeurs par défaut du navigateur,
  - les préférences de l'utilisateur fixées dans le navigateur,
  - les valeurs fixées dans le document ou les feuilles de styles.
  - Exemples : couleur des liens (visités ou non), du texte, fond de la page, polices...

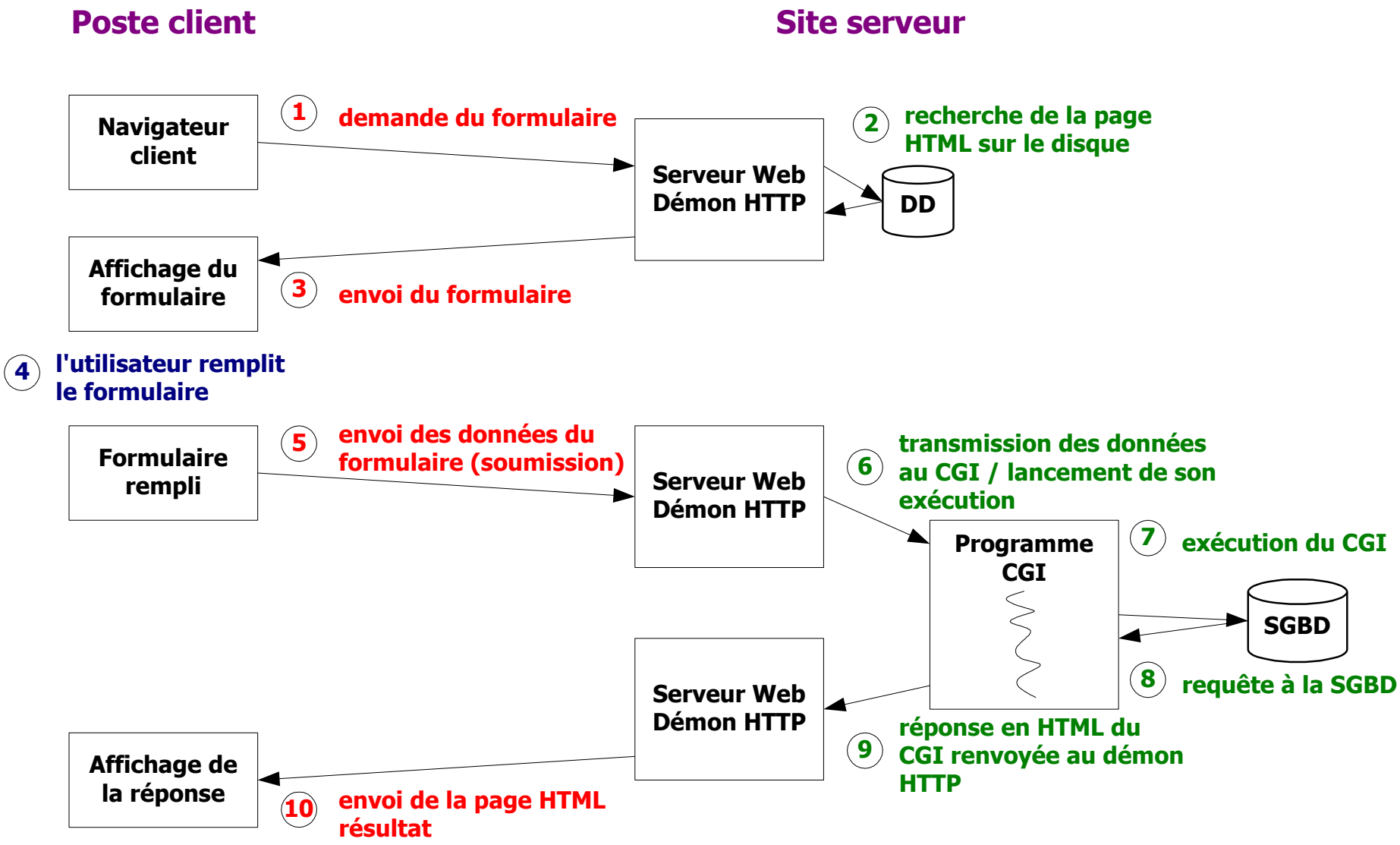
Le serveur web

- Il est en permanence à l'écoute des requêtes formulées par les clients (qui peuvent être très nombreux !)
- Il vérifie la validité de la requête…
  - Le document demandé peut ne pas exister
  - L'accès a un document peut être restreint (authentification possible)
- … et y répond si la requête est valide : envoi du texte, des images, de la feuille de styles, du code à exécuter sur le client (Javascript).
- Il peut renvoyer un message d'erreur, une demande d'authentification...
- Il peut exécuter un programme localement (PHP) qui va générer une réponse HTML (page **dynamique**) en fonction des arguments transmis par le navigateur.

Pages statiques et dynamiques

- Le HTML ne permet pas d'interactivité avec l'utilisateur
  - Les pages visualisées sont "statiques"
- Pages statiques
  - La page visualisée N fois sur le même navigateur donnera toujours le même résultat
- Pages dynamiques
  - La page visualisée dépend des manipulations de l'utilisateur
  - Elle est générée dynamiquement selon les actions de l'utilisateur dans la page
  - Nécessite de la programmation pour prendre en compte les actions
    - Programmation Web côté client : principalement Javascript
    - Programmation Web côté serveur : principalement PHP

Interactions navigateur/serveur web



UNIVERSITÉ DE LYON
Université Claude Bernard Lyon 1
Faculté des Sciences et Technologies
Dpt Informatique

Le protocole HTTP, méthodes GET et POST

Une transaction typique
Format des requêtes/réponses
Méthodes GET et POST
Champs cachés dans un formulaire et Cookies

Qu’est-ce que HTTP ?

- HTTP : Hyper Text Transfer Protocol
- Protocole régissant le dialogue entre des clients Web et un serveur (c'est le langage du Web !)
- Fonctionnement en mode Client/Serveur
- Une transaction HTTP contient
  - le type de la requête ou de la réponse (commande HTTP)
  - des en-têtes
  - une ligne vide
  - un contenu (parfois vide)
- Très peu de type de requêtes/réponses
- Port standard côté serveur : 80

Format des requêtes/réponses

- Format des requêtes
  - type de la requête (METHOD, URL, version HTTP)
  - en-têtes sur plusieurs lignes
  - une ligne vide
  - un contenu éventuel
- Format des réponses
  - un code de réponse (version HTTP, code, description)
  - en-têtes sur plusieurs lignes
  - une ligne vide
  - le contenu de la réponse

Une transaction typique (1)

- 1 - le client contacte le serveur pour demander le document index.html

GET /~ogluck/index2.html HTTP/1.1

- 2 - le client envoie des informations d'en-tête pour informer le serveur de sa configuration et des documents qu'il accepte

User-Agent: Mozilla/4.0 (compatible;MSIE 6.0;Windows NT 5.1)

Host: www710.univ-lyon1.fr

Accept: image/gif, image/jpeg, */*

- 3 - le client envoie une ligne vide (fin de l'en-tête) et un contenu vide dans cet exemple

Une transaction typique (2)

- 4 - le serveur répond en commençant par indiquer par un code, l'état de la requête

HTTP/1.1 200 OK

- 5 - le serveur envoie un en-tête qui donne des informations sur lui-même et le document demandé

Date: Sun, 23 May 2004 17:46:01 GMT

Server: Apache/1.3.28 (Debian GNU/Linux) PHP/3.0.18

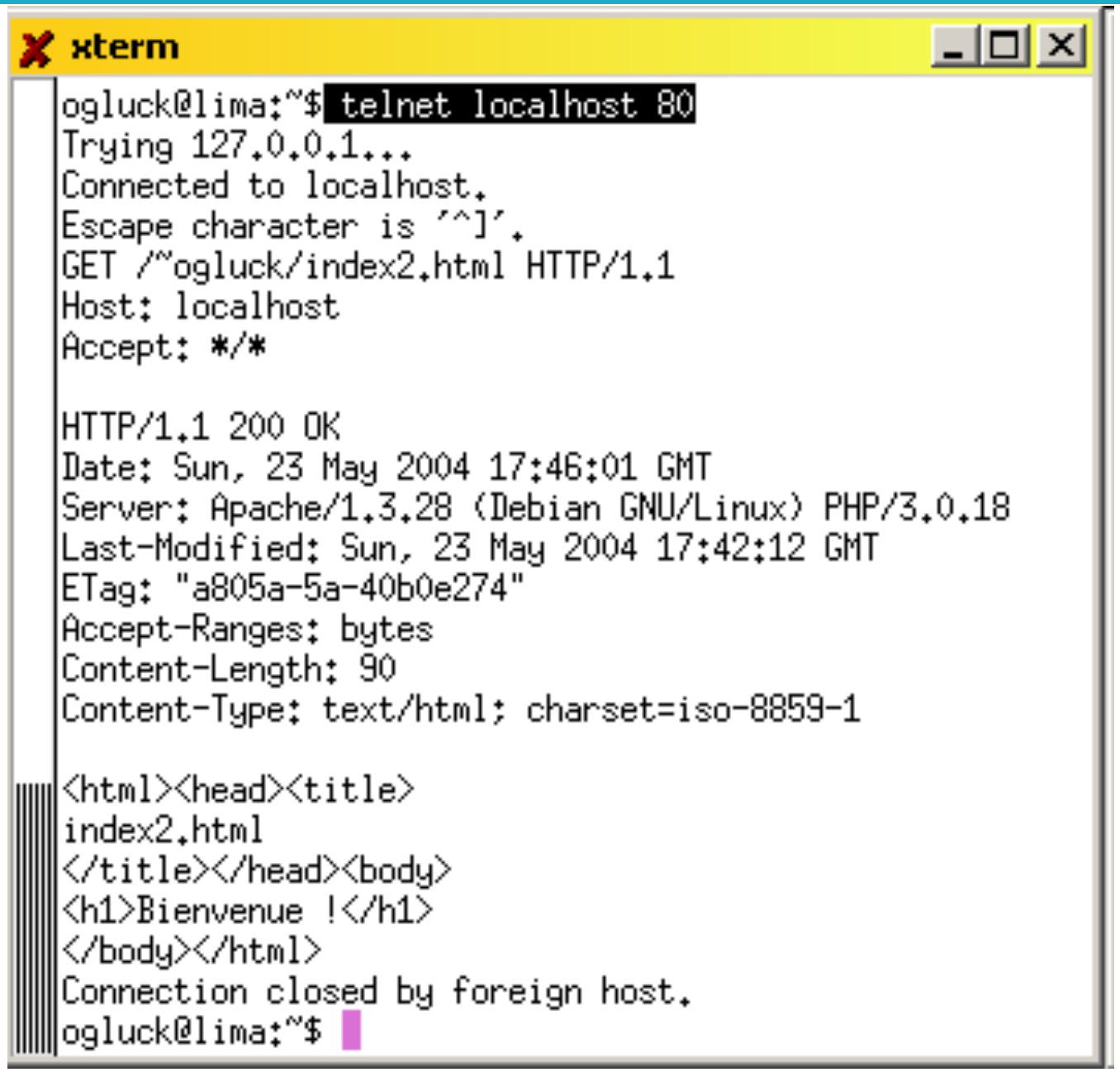
Last-Modified: Sun, 23 May 2004 17:42:12 GMT

Content-Length: 90

Content-Type: text/html; charset=iso-8859-1

- 6 - puis une ligne vide (fin de l'en-tête) et le contenu du document si la requête a réussi

Une transaction typique (3)



Quelques en-têtes de requêtes

- Identification du client

  From (adresse mail du client), Host (serveur, **obligatoire en HTTP1.1**), Referer (URL d'où l'on vient), User-Agent

- Préférences du client

  Accept (liste des types MIME acceptés), Accept-Encoding (compress|gzip|…), Accept-Langage, Accept-Charset

- Information pour le serveur

  Autorization (username:passwd encodé en base64), Cookie

- Conditions sur la réponse

  If-Modified-Since (utile pour les caches),

  If-Unmodified-Since, If-Match (Etag)

Quelques en-têtes de réponses

- Informations sur le contenu du document

  Content-Type (type MIME du document),
  Content-Length (barre de progression du chargement),
  Content-Encoding, Content-Location,
  Content-Langage

- Informations sur le document

  Last-Modified (date de dernière modification),
  Allow (méthodes autorisées pour ce document),
  Expires (date d'expiration du document)

- En-tête générales

  Date (date de la requête), Server (type du serveur)

Les réponses HTTP renvoyées par le serveur

- Une réponse contient trois éléments sur la 1ère ligne :

  version HTTP, code de statut, description textuelle du code

  HTTP/1.1 200 OK

  HTTP/1.1 404 Not Found

- Le code est un entier sur 3 chiffres classé selon des catégories

  100-199 : message d'information

  200-299 : succès de la requête cliente

  300-399 : la requête n'est pas directement serviable, le client doit préciser certaines choses

  400-499 : échec de la requête qui incombe au client

  500-599 : échec de la requête qui incombe au serveur (par ex. erreur d'exécution d'un programme sur le serveur)

La méthode GET

- La méthode standard de requête d'un document
  - Permet de récupérer un fichier HTML, une image, une feuille de styles, un fichier pdf…
  - Permet d'activer l'exécution d'un script PHP en lui transmettant des données en provenance d'un formulaire
- **Avec GET, le contenu de la requête est toujours vide**
- Le serveur répond avec une ligne décrivant l'état de la requête, un en-tête et le contenu demandé
- Si la requête échoue, le contenu de la réponse décrit la raison de l'échec (fichier non présent, non autorisé, …)

Transmission des paramètres avec GET

- Comme le contenu d'une requête GET est vide, les données du formulaire sont transmises via l'URL après un **?**
- Les champs sont séparés par un **&**

GET /cgi-bin/prog.php?email=toto@site.fr&pass=toto&s=login HTTP/1.1

- Ici, trois champs du formulaire sont transmis dans la requête : email, pass et s
- Attention : le mot de passe est transmis en clair !
- Permet de conserver l'URL dans un **favori** avec les données saisies dans le formulaire
- L'URL a une taille limitée qui dépend du navigateur et du serveur (conseillé de ne pas dépasser 2000 caractères)

La méthode POST

- Elle permet de transmettre des données au serveur dans le corps/contenu de la requête
- Exemple

POST /cgi-bin/prog.php HTTP/1.1

User-Agent: Mozilla/4.0 (compatible;MSIE 6.0;Windows NT 5.1)

Host: localhost

Accept: */*

Content-type: application/x-www-form-urlencoded

Content-length: 36

email=toto@site.fr&pass=toto&s=login

- Le mot de passe est toujours transmis en clair !

Qu'est ce qu'un programme CGI ?

- Un programme
  - qui s'exécute sur la machine hébergeant le serveur HTTP
  - en langage compilé (binaire) ou interprété (script)
  - qui permet de
    - récupérer les données du formulaire à l'aide d'un *parser* : **pour chaque champ, un couple NAME/VALUE est transmis au serveur**
    - effectuer des traitements sur le serveur
      - lecture/écriture dans une base de données
      - stockage d'une info (compteurs, identifiant de connexion, …)
      - recherche d'une info
      - pied de page automatique (ex : date de dernière modification)
    - générer un résultat qui est renvoyé au client
      - page HTML, image, document postcript, …

Récupérer les données du formulaire

- Format de la chaîne CGI

  nom_champ1=valeur1&nom_champ2=valeur2&…

- Cas des champs à valeurs multiples
  - exemple : listes à sélection multiples

  nom_liste=valeur1&nom_liste=valeur2&…

- La chaîne CGI
  - elle est construite par le client au format *URL-encoded* quand la requête est postée
  - elle est transmise au CGI tel quel via la variable d'environnement QUERY_STRING avec la méthode GET
  - elle est transmise au CGI tel quel via l'entrée standard avec la méthode POST
  - en PHP, **$_GET['nom_champ1']** ou **$_POST['nom_champ1']** contient la valeur du champ

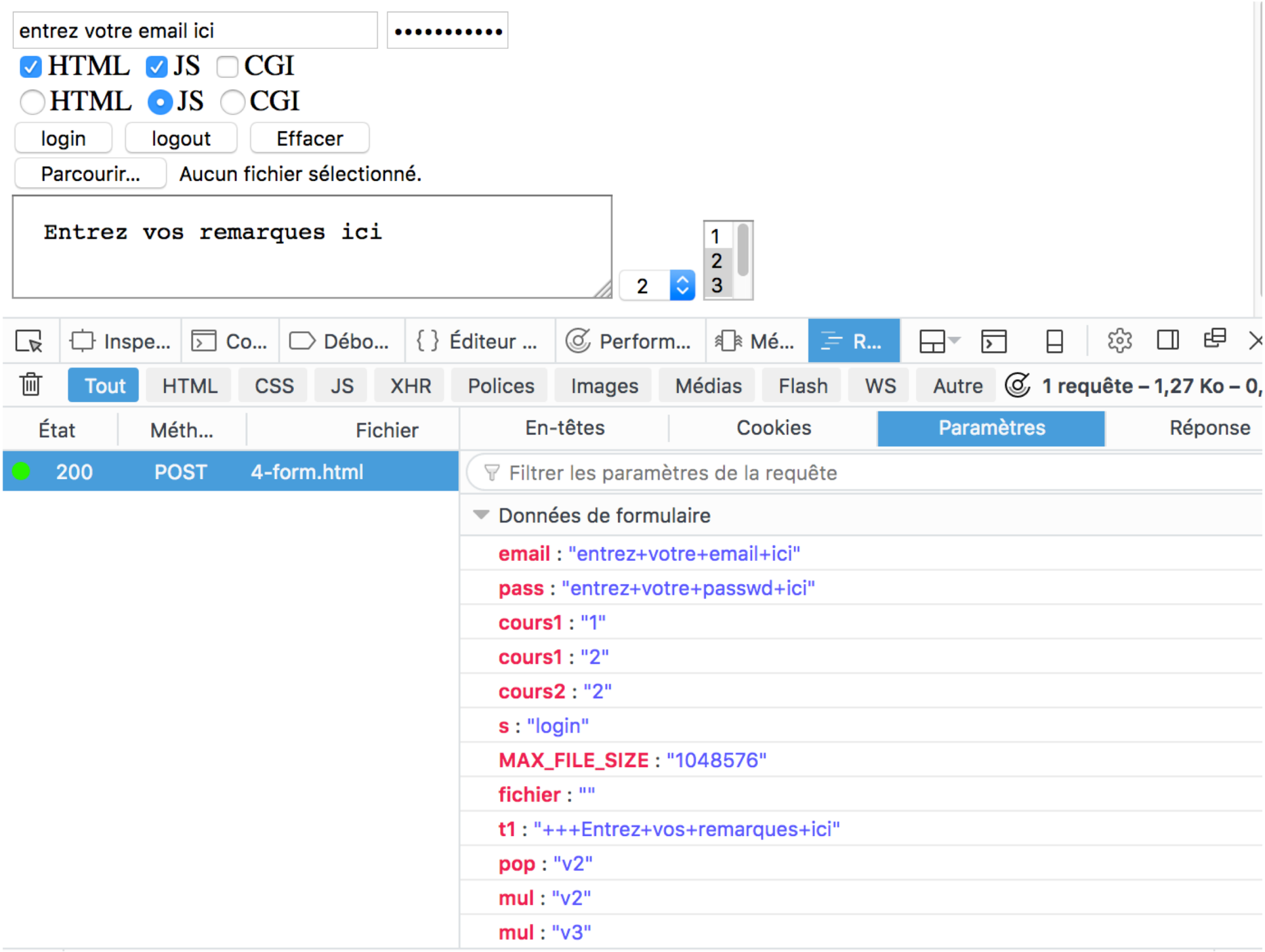
Méthodes GET/POST

- Avec la méthode GET
  - les données relatives aux champs du formulaire sont transmises via l'URL (dans le type de la requête)
  - le programme CGI les récupère dans la variable d'environnement **`QUERY_STRING`**
  - il est possible de cliquer sur "Actualiser" pour retransmettre les données et de définir un *bookmark*
- Avec la méthode POST
  - les données relatives aux champs du formulaire sont transmises dans le corps de la requête HTTP
  - `Content-type` et `Content-length` sont positionnés
  - le programme CGI les récupère sur l'entrée standard
  - "Actualiser" et *bookmark* impossibles, données du formulaire non visibles dans les logs du serveur

Exemple 4-form.html

```
<HTML><HEAD><TITLE>Eléments d'un formulaire</TITLE>
</HEAD><BODY>
<FORM name="f1" METHOD="POST" ACTION="4-form.html">
<INPUT NAME="email" VALUE="entrez votre email ici" SIZE="30" MAXLENGTH="50" />
<INPUT type="PASSWORD" NAME="pass" VALUE="entrez votre passwd ici" SIZE="8" MAXLENGTH="8" /><BR />
<INPUT type="CHECKBOX" NAME="cours1" VALUE="1" CHECKED />HTML
<INPUT type="CHECKBOX" NAME="cours1" VALUE="2" CHECKED />JS
<INPUT type="CHECKBOX" NAME="cours1" VALUE="3" />CGI<BR />
<INPUT type="RADIO" NAME="cours2" VALUE="1" />HTML <INPUT type="RADIO" NAME="cours2" VALUE="2" CHECKED />JS
<INPUT type="RADIO" NAME="cours2" VALUE="3" />CGI<BR />
<INPUT type="SUBMIT" NAME="s" VALUE="login" /> <INPUT type="SUBMIT" NAME="s" VALUE="logout" />
<INPUT type="RESET" NAME="raz" VALUE="Effacer" /><BR />
<INPUT type="hidden" name="MAX_FILE_SIZE" value="1048576" />
<INPUT type="FILE" NAME="fichier" VALUE="Parcourir" /><BR />
<TEXTAREA NAME="t1" ROWS="3" COLS="40"> Entrez vos remarques ici</TEXTAREA>
<SELECT NAME="pop"><OPTION VALUE="v1" />1 <OPTION VALUE="v2" SELECTED />2 <OPTION VALUE="v3" />3
</SELECT>
<SELECT NAME="mul" SIZE="3" MULTIPLE>
<OPTION VALUE="v1" />1 <OPTION VALUE="v2" SELECTED />2 <OPTION VALUE="v3" SELECTED />3 <OPTION VALUE="v4" />4
</SELECT> </FORM></BODY></HTML>
```

Regarder avec Firebug ce qui est transmis


entrez votre email ici
••••••••••
HTML JS CGI
HTML JS CGI
login
logout
Effacer
Parcourir...
Aucun fichier sélectionné.
Entrez vos remarques ici
2
1
2
3
Inspe...
Co...
Débo...
Éditeur ...
Perform...
Mé...
R...
Tout
HTML
CSS
JS
XHR
Polices
Images
Médias
Flash
WS
Autre
1 requête – 1,27 Ko – 0,
État
Méth...
Fichier
En-têtes
Cookies
Paramètres
Réponse
200
POST
4-form.html
Filtrer les paramètres de la requête
Données de formulaire
email : "entrez+votre+email+ici"
pass : "entrez+votre+passwd+ici"
cours1 : "1"
cours1 : "2"
cours2 : "2"
s : "login"
MAX_FILE_SIZE : "1048576"
fichier : ""
t1 : "+++Entrez+vos+remarques+ici"
pop : "v2"
mul : "v2"
mul : "v3"

HTTP est un protocole sans état

- HTTP 1.0 (RFC 1945)
  - dès que le serveur a répondu à une requête, il ferme la connexion TCP
- HTTP 1.1 (RFC 2068)
  - par défaut, la connexion est maintenue tant que le serveur ou le client ne décide pas de la fermer (`Connection: close`)
- HTTP est un protocole **sans état**
  - aucune information n'est conservée entre deux requêtes successives : les requêtes sont indépendantes les unes des autres
  - permet au serveur HTTP de servir plus de clients en un temps donné (gestion légère des transactions)
  - **pour conserver des informations entre deux requêtes, il faut utiliser un *cookie*, des champs cachés d'un formulaire, …**

Utilisation d'un champ HIDDEN

- Un champ de type HIDDEN permet de propager une variable d'une page à une autre.
- Souvent on propage un identifiant, un login, …
- Par exemple, on aurait pu utiliser

  <INPUT type="HIDDEN" NAME="score" VALUE="1024" />

  pour transmettre le score du jeu 2048 entre deux actions de jeu plutôt que de le stocker dans le fichier score.txt sur le serveur.

  Pour récupérer la valeur du score au début du programme PHP :

  $score = $_GET['score']; // $score contient 1024, l'ancien score

  $score = $score + 8; // nouveau score : deux 4 ont fusionné

  Pour propager la nouvelle valeur du score vers la page suivante :

  echo '<input type="hidden" name="score" value="' . $score . '"/>';

Qu’est-ce qu’un cookie ?

- C’est un moyen pour le serveur de stocker des informations dans le navigateur client pour palier au caractère sans état du protocole HTTP
- Un cookie est une chaîne de caractères url-encodée de 4ko maximum stockée sur le disque dur du client
- Un cookie stocke des informations associées à un ensemble d'URL qui sont envoyées lors de chaque requête vers l'une de ces URL
- Les *cookies* permettent de
  - donner un identifiant au client (permet par exemple au serveur de ne pas compter deux fois le même navigateur pour un comptage du nombre de visites sur le site)
  - propager un code d'accès (évite une authentification lors de chaque requête)
  - propager une identification dans une base de données
  - propager l’identifiant d’une session PHP

Différentes versions du protocole HTTP

- Version d'origine : HTTP 0.9
  - Une seule méthode : GET
  - Pas d'en-tête
  - Une requête = une connexion TCP
- Amélioration en deux étapes
  - HTTP 1.0 :
    - Introduction des en-têtes pour échanger des **métadonnées** entre le navigateur et le serveur web
    - Nouvelles fonctionnalités : utilisation de caches, méthodes d'authentification (`.htaccess`)…
  - HTTP 1.1 :
    - Mode **connexions persistantes** par défaut
    - Introduction des serveurs virtuels -> l'en-tête `Host` est obligatoire dans la requête

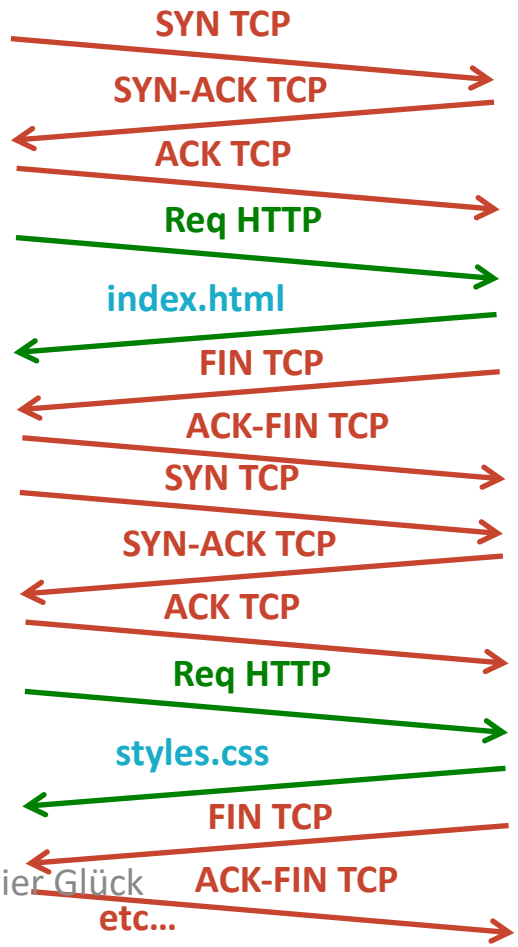
Intérêt des connexions persistantes

Exemple d'une page HTML contenant 1 feuille de styles et 2 images :

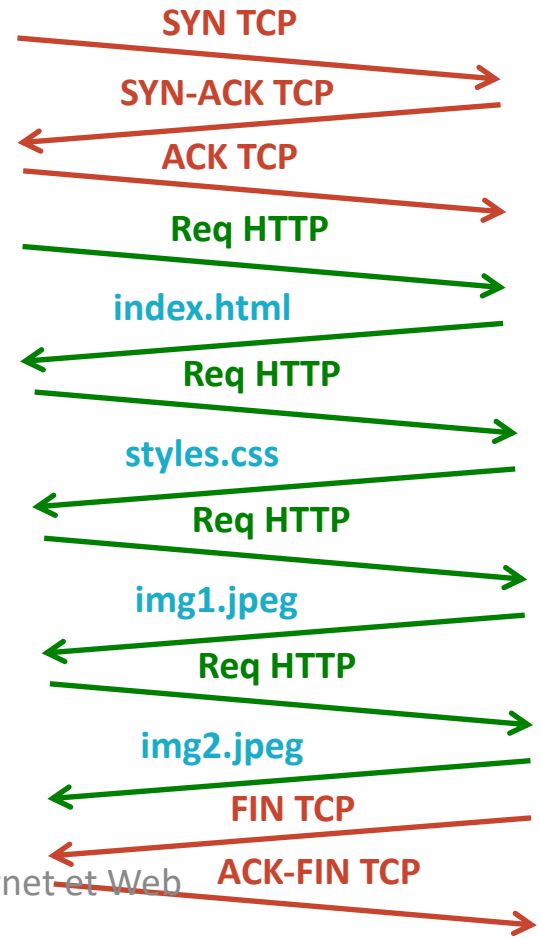
Avec HTTP 0.9, trois connexions/déconnexions TCP/IP

Avec HTTP 1.1, une seule connexion TCP/IP

Les connexions persistantes améliorent les performances.



Laquelle est persistante ?

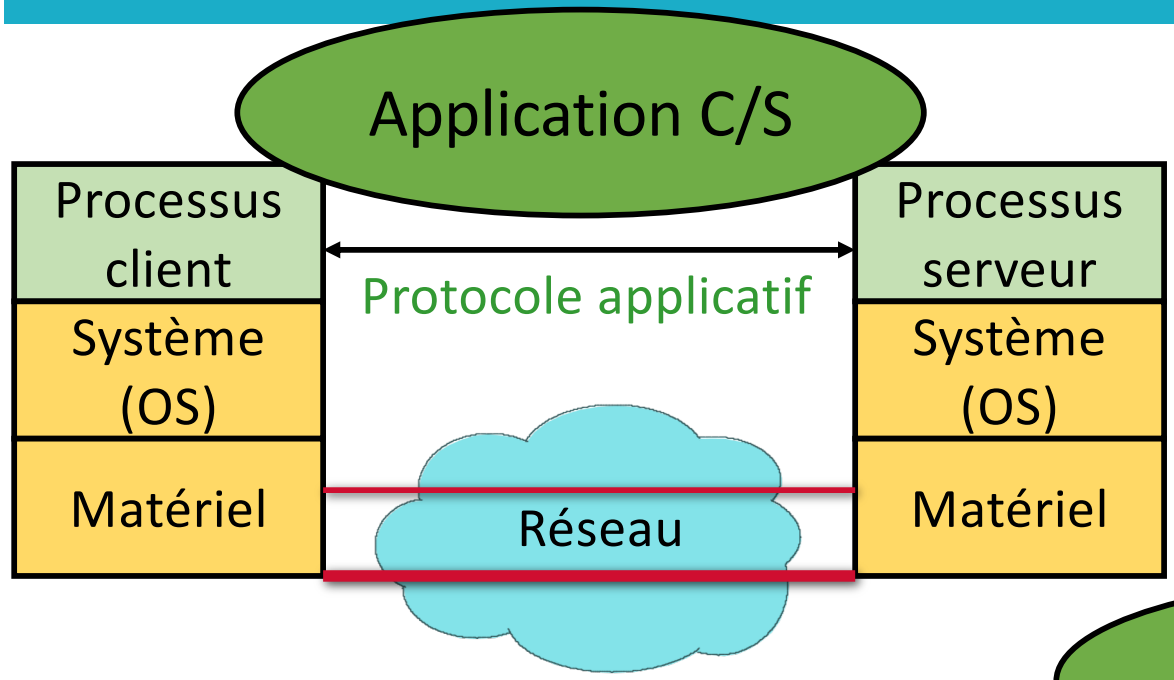


UNIVERSITÉ DE LYON
Université Claude Bernard Lyon 1
Faculté des Sciences et Technologies
Dpt Informatique

Le modèle Client/Serveur

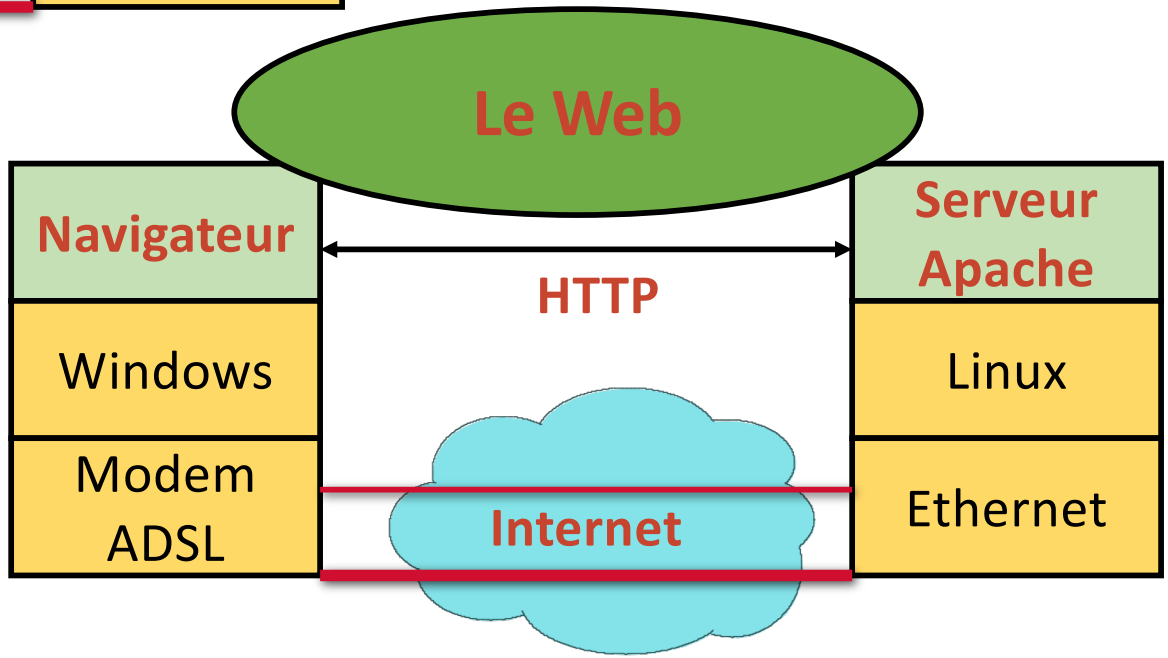
Qu'est-ce que le modèle client/serveur ?
Les sockets et les numéros de port
L'architecture TCP/IP

Le modèle Client / Serveur

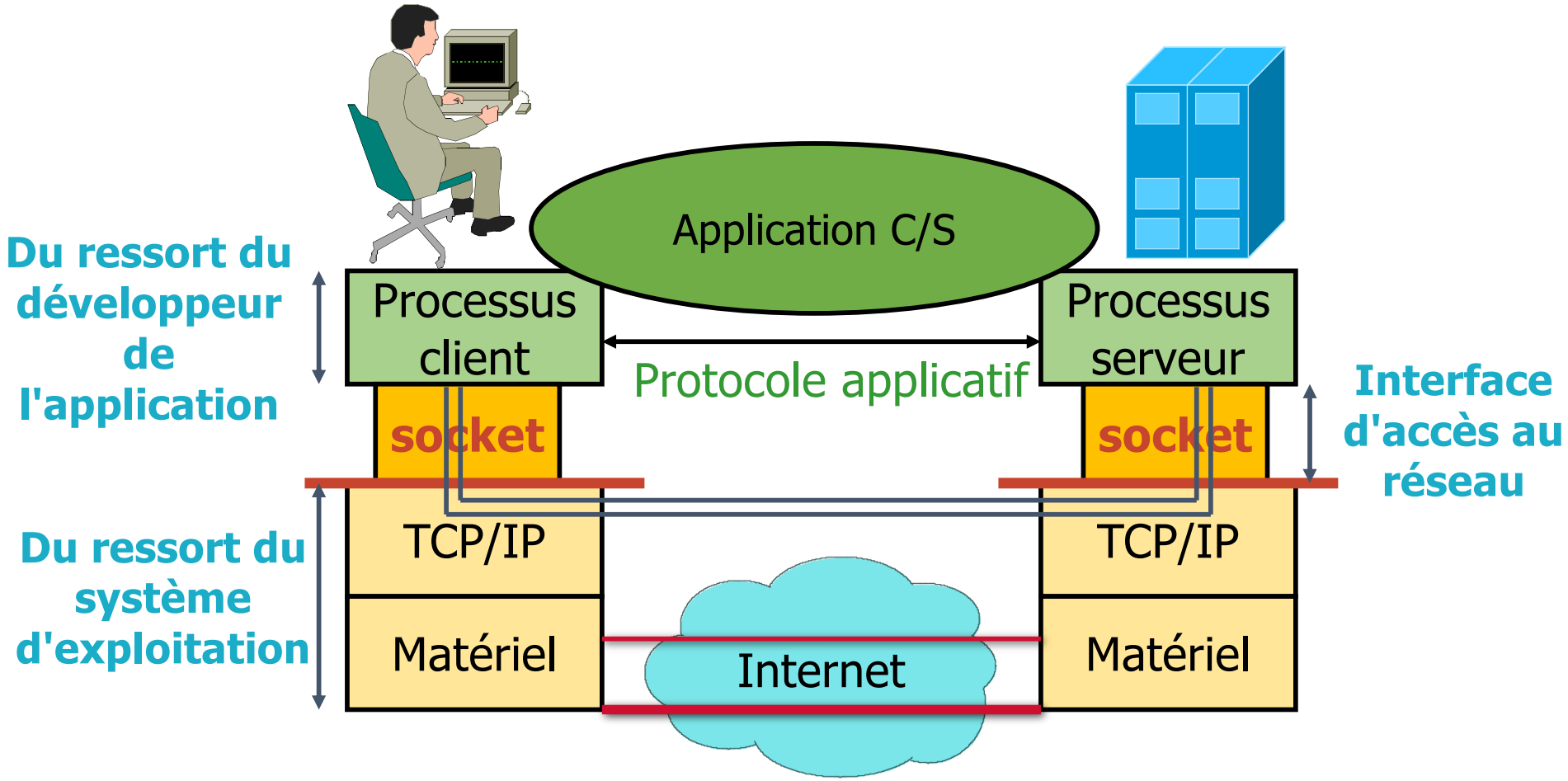


L'application est répartie sur le client et le serveur qui dialoguent selon un protocole applicatif spécifique

L'exemple du Web



Les sockets, interface d'accès au réseau

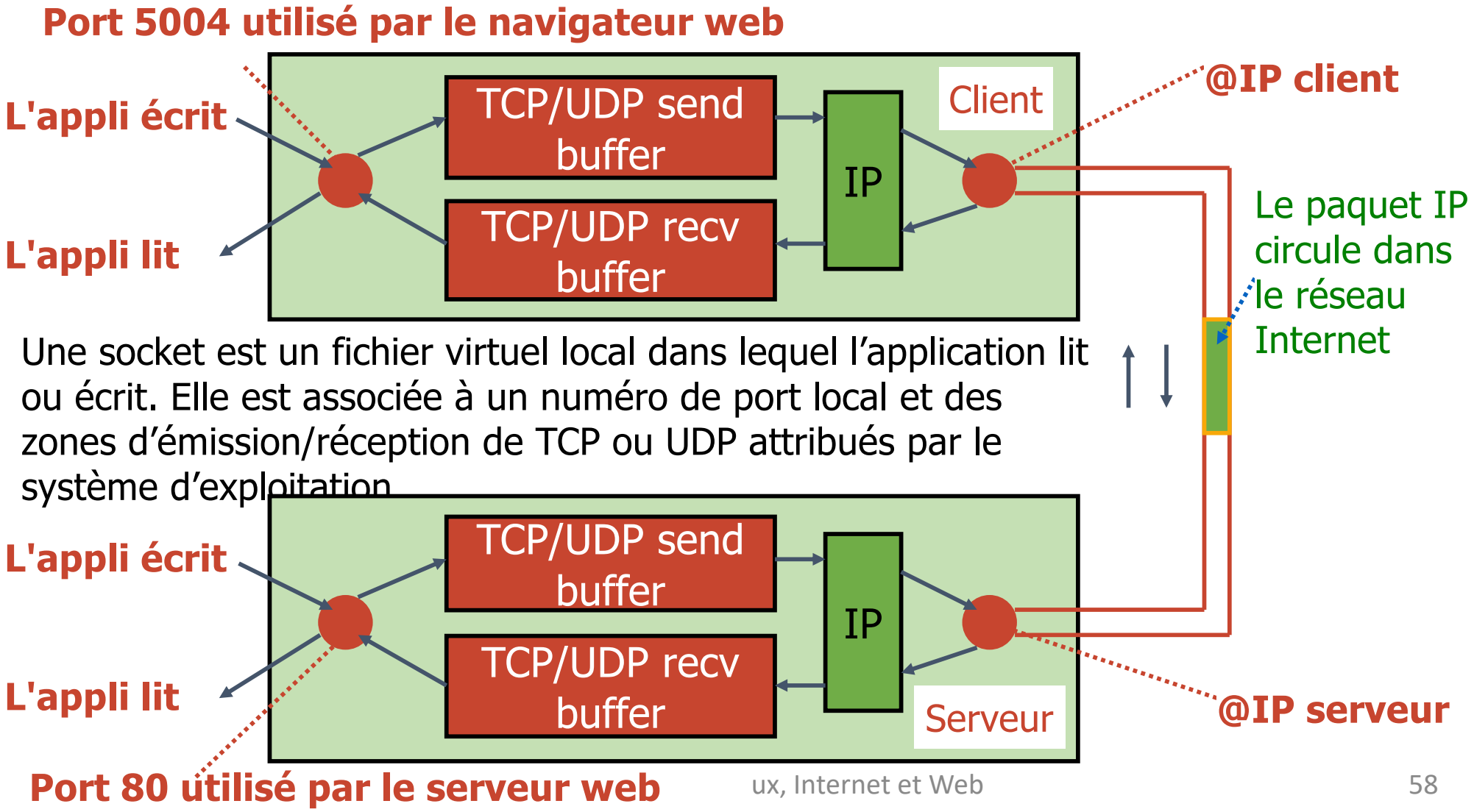


Une socket : interface locale à l'hôte, créée par l'application, contrôlée par l'OS

Porte de communication entre le processus client et le processus serveur

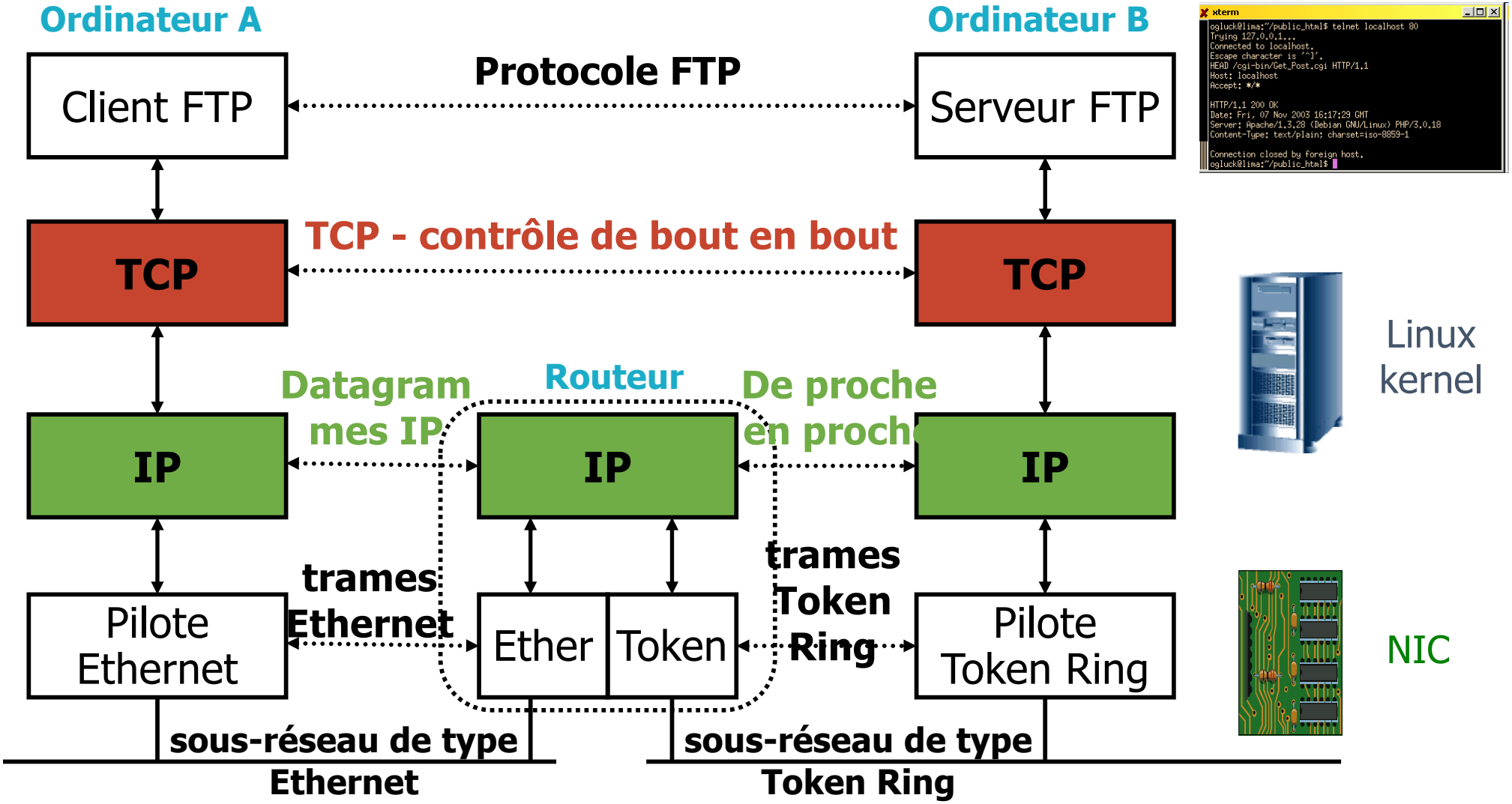
Les sockets et les numéros de port

- Un échange Client/Serveur = **(@IP_src,port_src,@IP_dest,port_dest)**

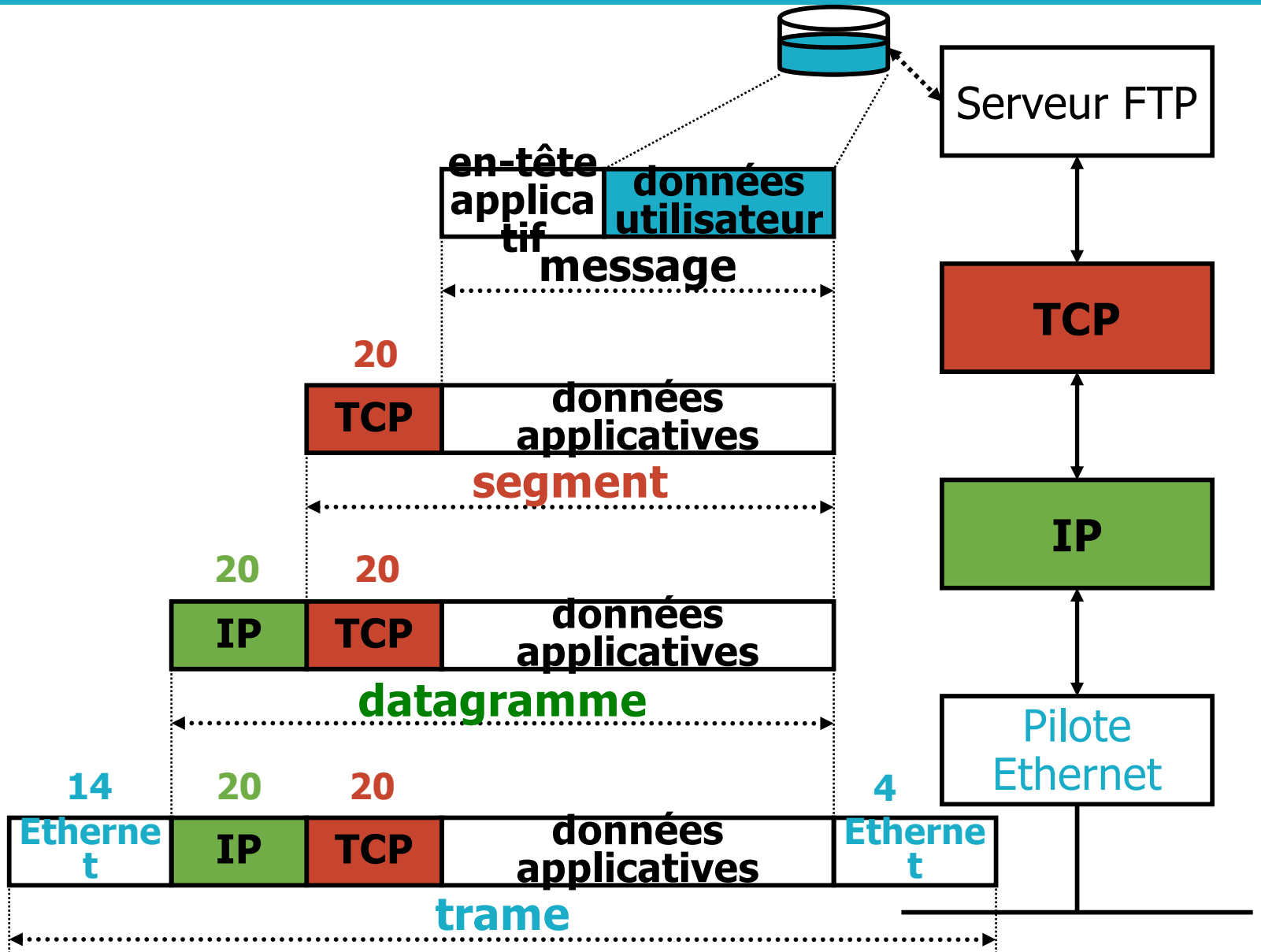


Communications avec routeur(s)

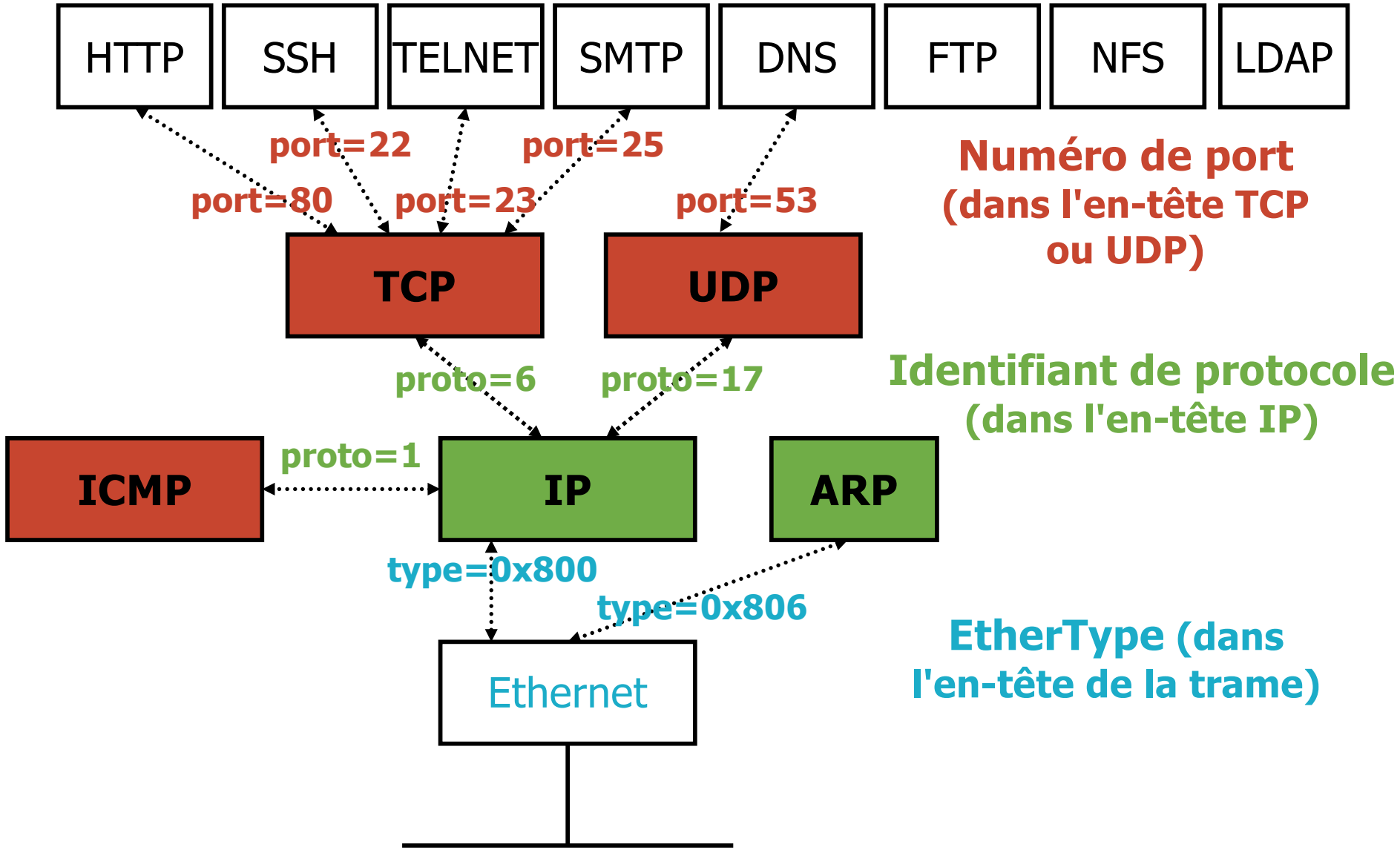
- Prise en compte de l'hétérogénéité



L'encapsulation : ajout des en-têtes



La désencapsulation : identifier la couche >



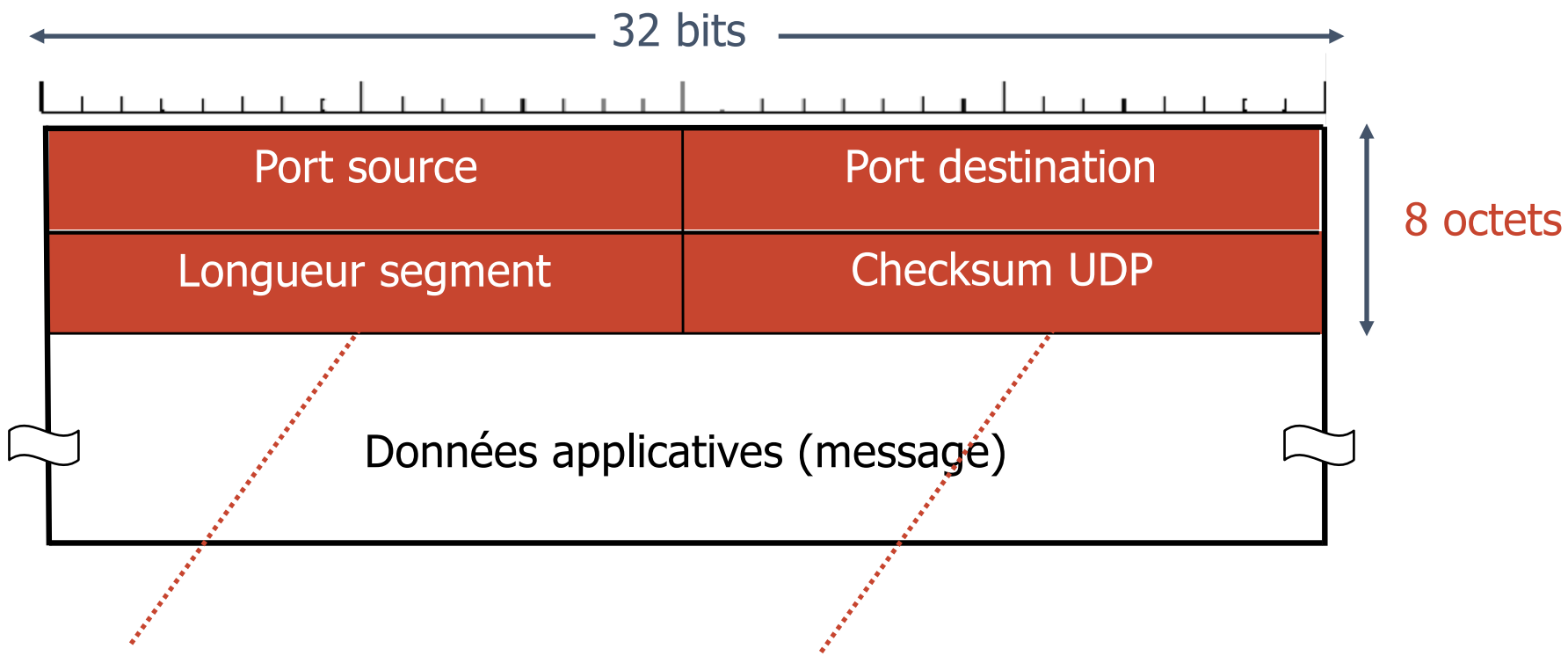
Le protocole UDP et la rapidité

Le protocole UDP et son en-tête
Le mode non connecté
Quelles applications utilisent UDP ?

Le protocole UDP

- UDP (RFC 768) - User Datagram Protocol
  - protocole de transport le plus simple
  - service de type best-effort (comme IP)
    - les datagrammes UDP peuvent être perdus
    - les datagrammes UDP peuvent arriver dans le désordre
  - mode non connecté : chaque datagramme UDP est traité indépendamment des autres
- Pourquoi un service **non fiable sans connexion** ?
  - simple donc **rapide** (pas de délai de connexion, pas d'état entre émetteur/récepteur)
  - petit en-tête donc économie de bande passante
  - UDP peut émettre aussi rapidement qu'il le souhaite : pas de limite à l'envoi contrairement à TCP (contrôle de congestion)

L'en-tête UDP : 8 octets



Taille totale du datagramme (en-tête+données)

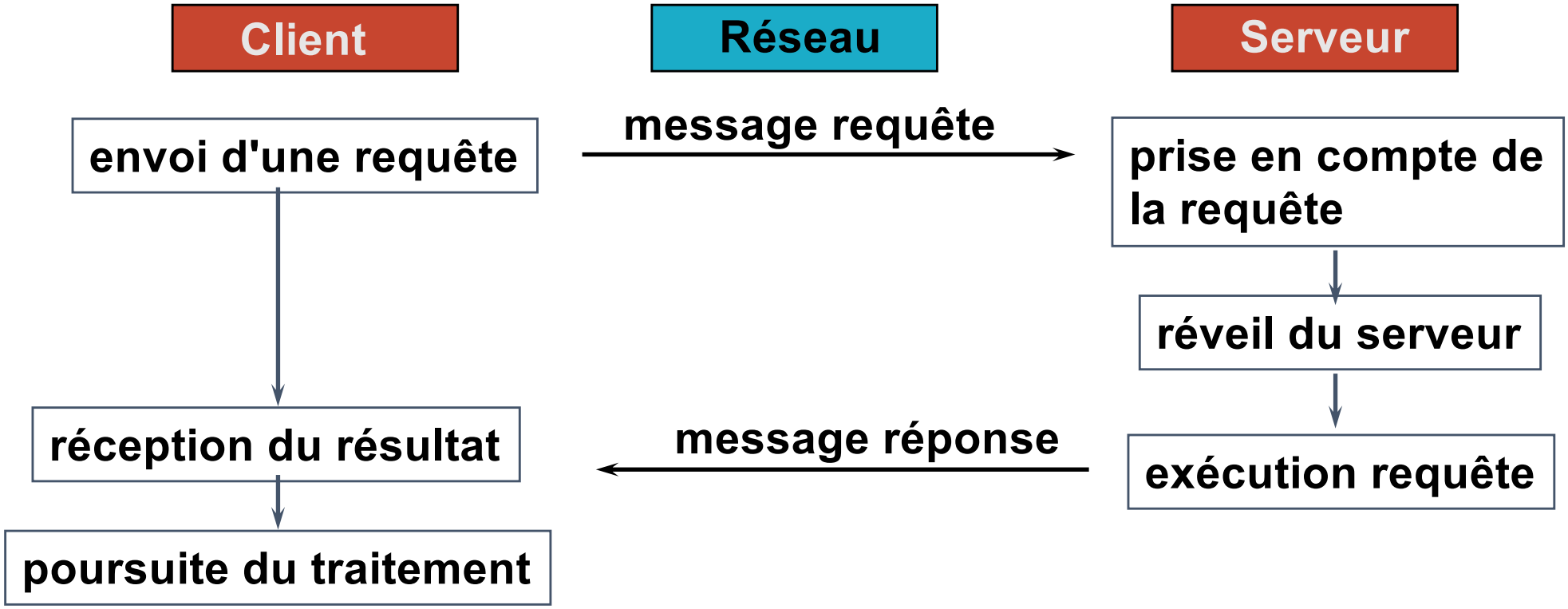
Somme de contrôle du datagramme (en-tête+données)

optionnel : peut être à 0

UDP = IP + numéros de port !!

UDP est en mode non connecté

Permet d'envoyer rapidement des messages mais sans l'autorisation du destinataire donc aucune garantie sur la réception du message



Exemple de mode non connecté : courrier postal, mail

Tous les protocoles d'Internet sont en mode non connecté SAUF TCP

Quelles applications utilisent UDP ?

- Pour les applications qui ont besoin d'envoyer rapidement
  - UDP permet des envois rapides sans limitation mais sans garantie donc sans fiabilité
- Souvent utilisé pour les **applications multimédias**
  - Vidéos, son, musique, streaming, visioconférence, voix sur IP
  - Ces applications sont tolérantes aux pertes/erreurs et sensibles au débit (les données doivent arriver à la bonne vitesse)
- Autres utilisations d'UDP
  - Applications qui envoient peu de données et qui ont besoin de rapidité
  - exemple : **DNS**
- Transfert fiable sur UDP
  - Comme UDP n'apporte aucune garantie, l'application peut ajouter des mécanismes pour réparer les pertes ou erreurs (acquittements…)

UNIVERSITÉ DE LYON
Université Claude Bernard Lyon 1
Faculté des Sciences et Technologies
Dpt Informatique

Le protocole TCP et la fiabilité

Qu'est-ce que la fiabilité ?
Les mécanismes de la fiabilité
Le mode connecté
L'en-tête TCP
Quelles applications utilisent TCP ?

Le protocole TCP et la fiabilité

- Transport Control Protocol (RFC 793, 1122, 1323, 2018, 2581)
- Transport **fiable** en **mode connecté**
  - entre un client et un serveur : (@IP src, port src) --> (@IP dest, port dest)
  - transporte un flot d'octets (ou flux)

    L'application lit/écrit des octets dans un tampon, TCP assure la fiabilité
- Fiabilité : faire en sorte que tout ce qui arrive est exactement ce qui a été envoyé
  - Les données ne doivent pas être perdues : **sans perte**
  - Les données ne doivent pas subir d'erreurs : **sans erreur**

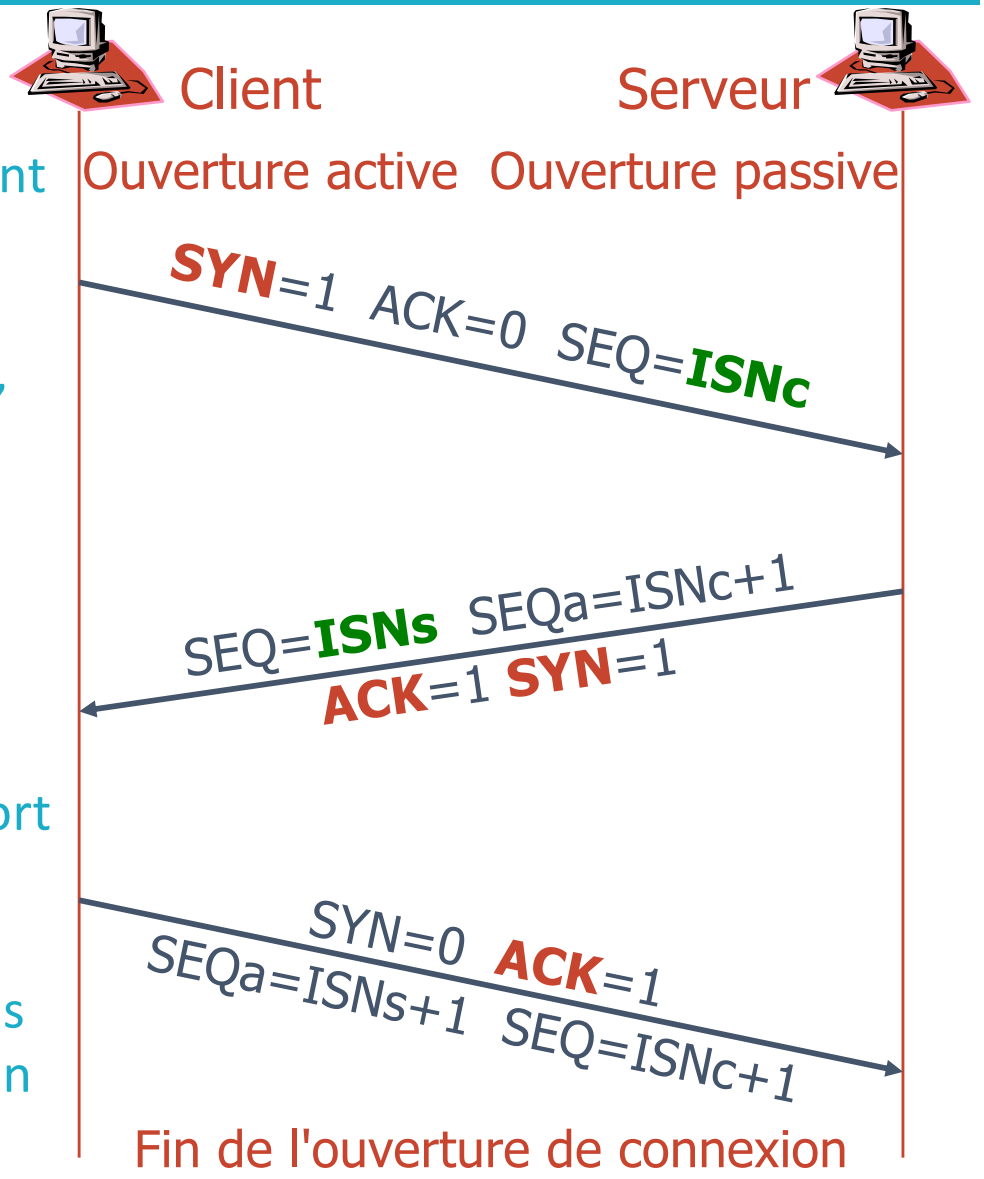
    Une erreur = un bit qui change de valeur pendant le transfert
  - Les données doivent arriver **dans l'ordre**
  - Les données ne doivent pas arriver en double : **sans duplication**

Les mécanismes de la fiabilité

- Fiabilité : **sans perte, sans erreur, dans l'ordre, sans duplication**
- En cas de perte ou erreur, il faut retransmettre si on n'a pas reçu d'acquittement (ACK) au bout d'un certain temps
- Pour détecter une perte ou une duplication, il faut des ACK et numéroter les messages et les ACK
- Pour détecter les erreurs, on utilise les checksum
- Pour retransmettre, il faut conserver les messages envoyés qui n'ont pas encore été acquittés
- Mécanismes : **retransmissions, timeout, ACK, stockage des messages non acquittés, checksum, numérotation des messages et des acquittements**

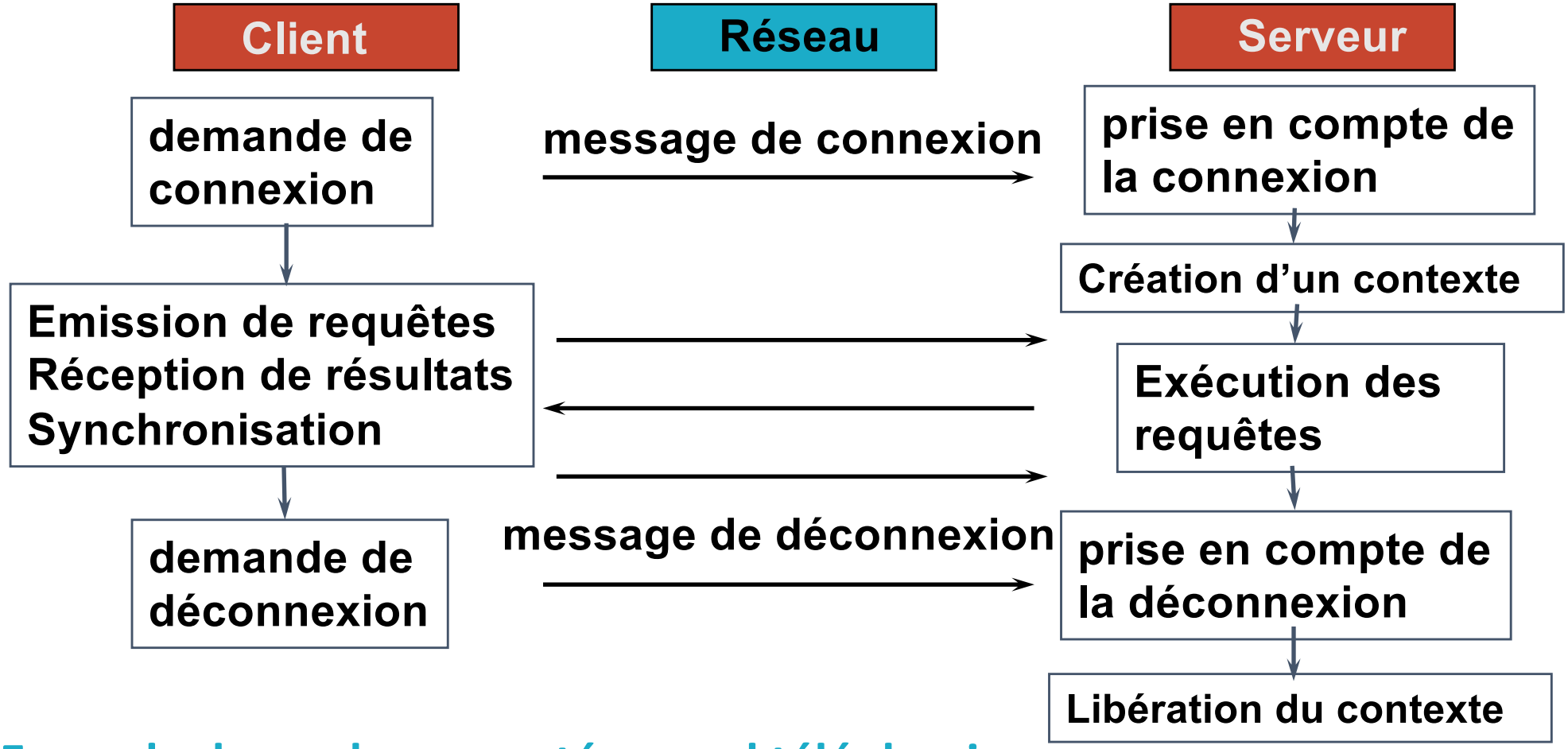
Etablissement d'une connexion TCP

- Connexion en trois phases
  - 1 - demande d'ouverture par le client (SYN), choix ISNc
  - 2 - acceptation par le serveur (SYN+ACK), allocation des tampons, choix ISNs
  - 3 - le client acquitte l'acceptation (ACK)
- Mode Client/Serveur
  - Le TCP du client fait une demande d'ouverture de connexion vers le port du serveur qui doit être connu à l'avance
  - Le TCP du serveur est en attente des demandes d'ouverture de connexion en provenance des clients



TCP est en mode connecté

Permet d'envoyer des messages avec fiabilité mais limitation du débit à l'envoi : contrôle de flux et contrôle de congestion

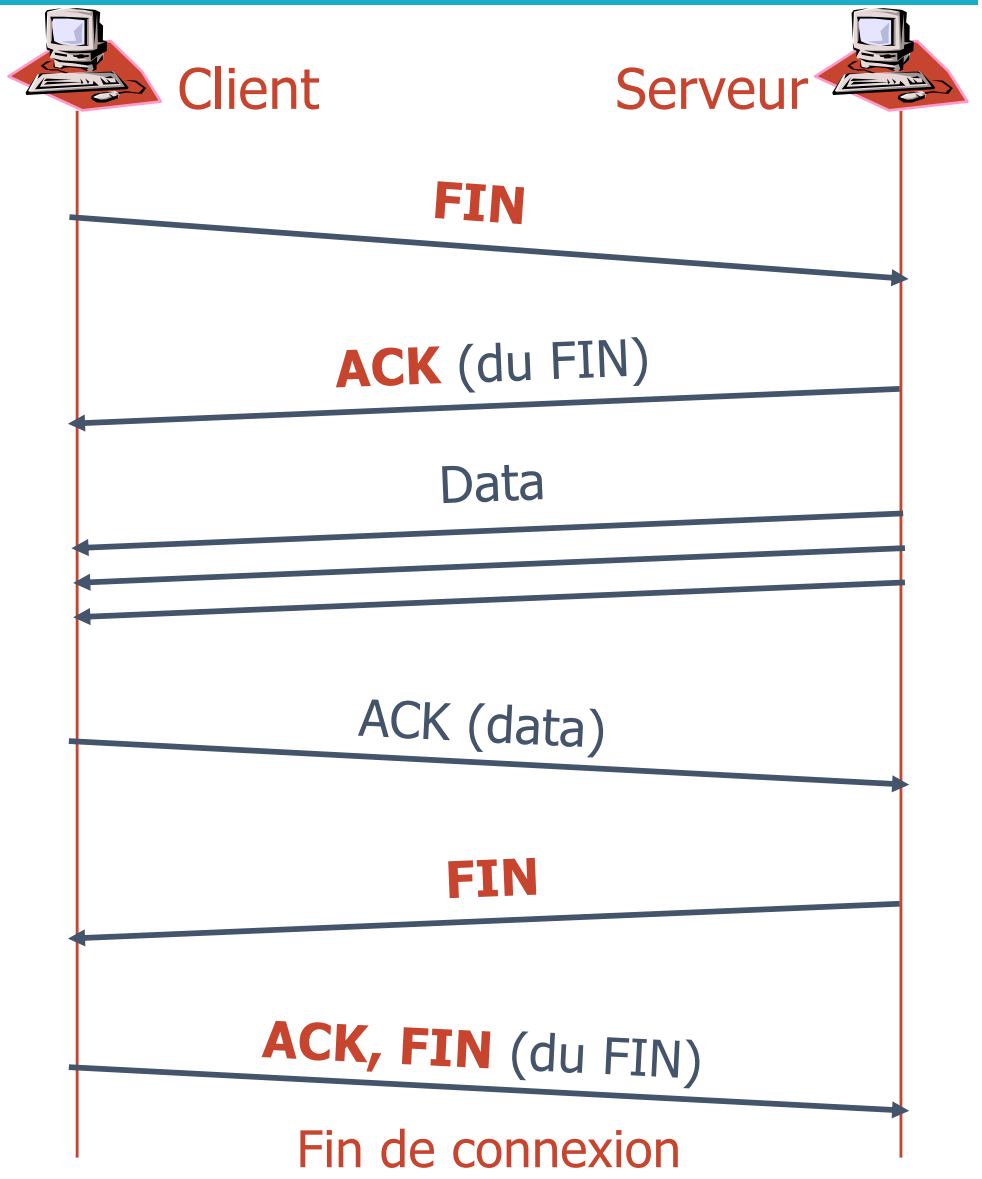


Exemple de mode connecté : appel téléphonique

TCP est le seul protocole d'Internet en mode connecté

Fermeture d'une connexion TCP

- Fermeture négociée
  - 1 - demande de fin de connexion (FIN) par une des extrémités
  - 2 - acquittement du FIN (ACK) mais mise en attente de la demande (Le serveur a encore des données non transmises)
  - 3 - envoi des données en attente
  - 4 - acquittement des données (ACK)
  - 5 - acceptation de la fin de connexion par le serveur (FIN)
  - 6 - acquittement de la fin de connexion (ACK)



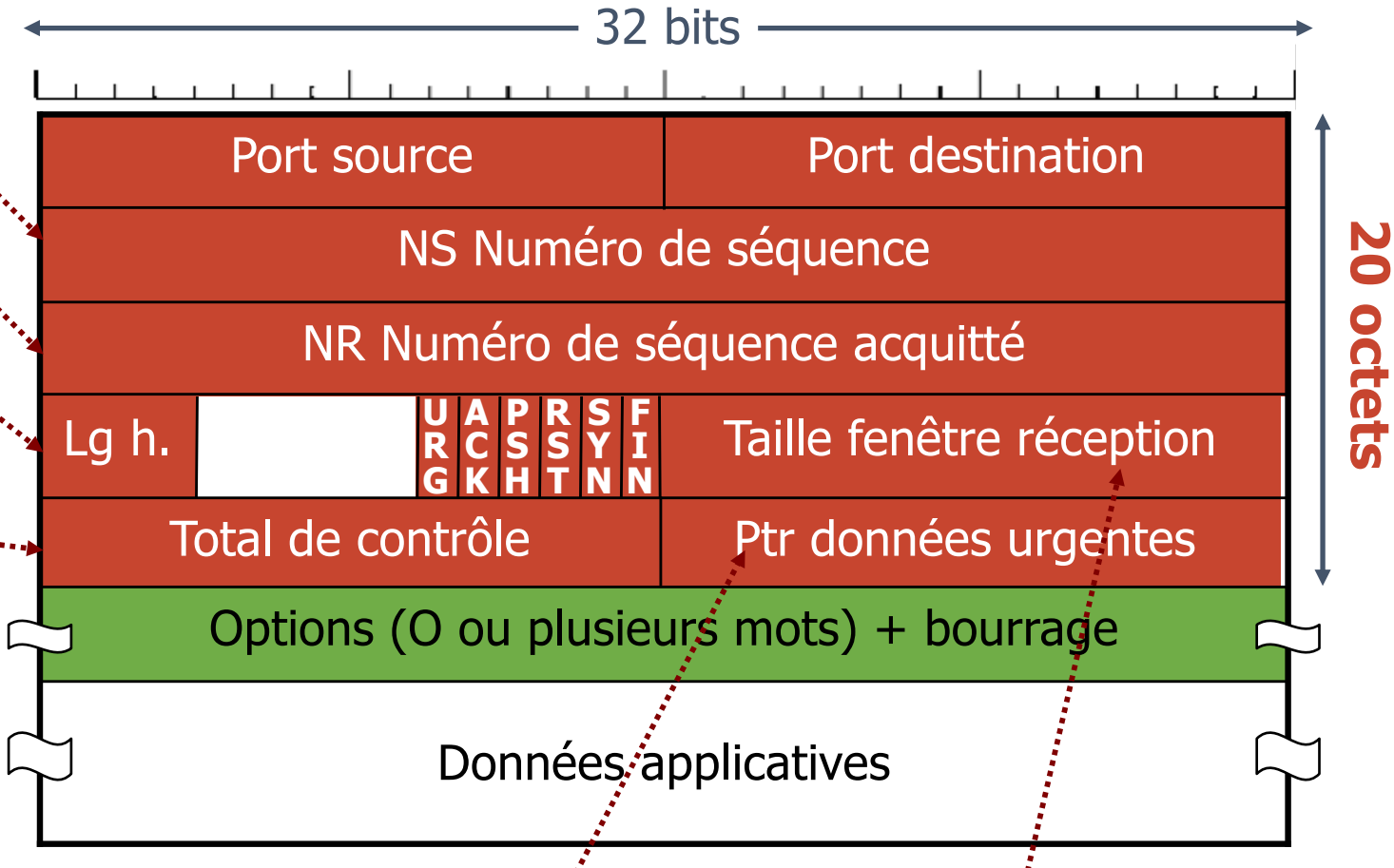
L'en-tête TCP : 20 octets

Numéro du premier octet du segment

Numéro du prochain octet attendu

Longueur en-tête en multiple de 4 octets

Checksum sur tout le segment (cf. UDP)

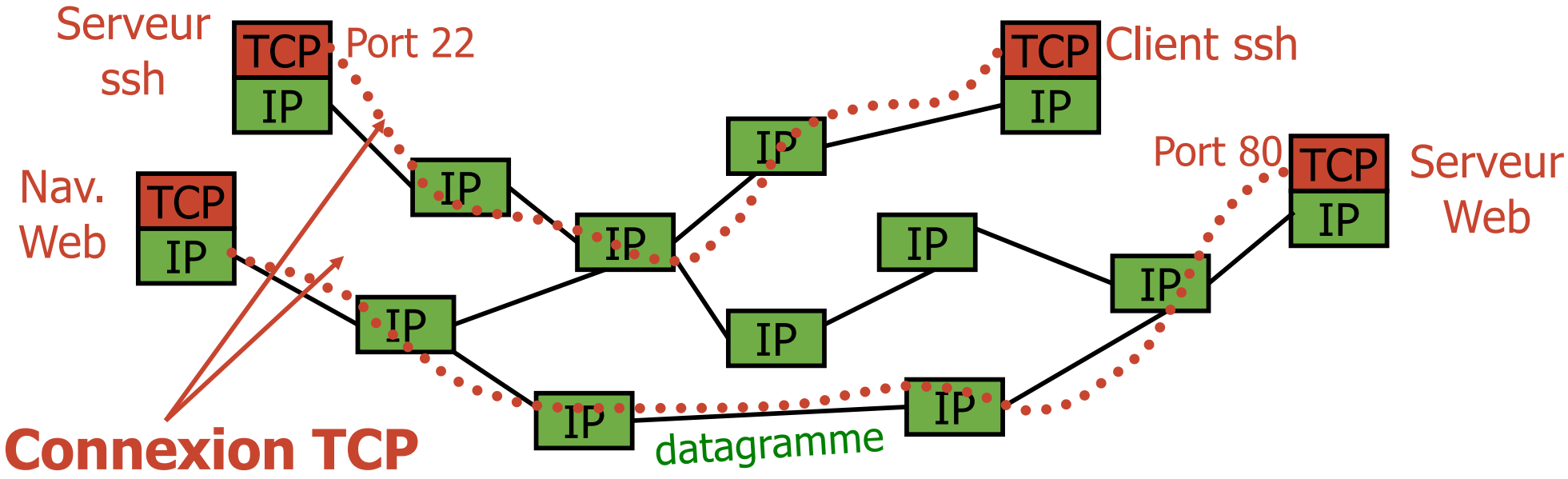


Les données comprises entre le premier octet DATA et la valeur du Ptr sont urgentes : TCP interrompt l'application pour forcer la lecture

Nb d'octets que le récepteur peut recevoir

Caractéristiques du protocole TCP

Couche transport : communications entre applis



- TCP - protocole de transport **de bout en bout**
  - uniquement présent **aux extrémités**
  - transport **fiable** de **segments** (mode **connecté**)
  - protocole complexe (retransmission, gestion des erreurs, séquencement, …)

Quelles applications utilisent TCP ?

Toutes celles qui ne peuvent pas se passer de la fiabilité c'est à dire presque toutes !

- Le web (**HTTP**)
- La connexion à distance (**telnet**, **ssh** et **X**)
- Le courrier électronique (**SMTP**, **POP**, **IMAP**, **Webmail**)
- Le transfert de fichiers (**FTP**)
- L'accès aux fichiers distants (**NFS**, **SMB**)
- L'annuaire fédérateur (**LDAP**)

Les applications multimédias et le DNS n'utilisent pas TCP

UNIVERSITÉ DE LYON
Université Claude Bernard Lyon 1
Faculté des Sciences et Technologies
Dpt Informatique

Le protocole IP

Les adresses IPv4

Les sous-réseaux

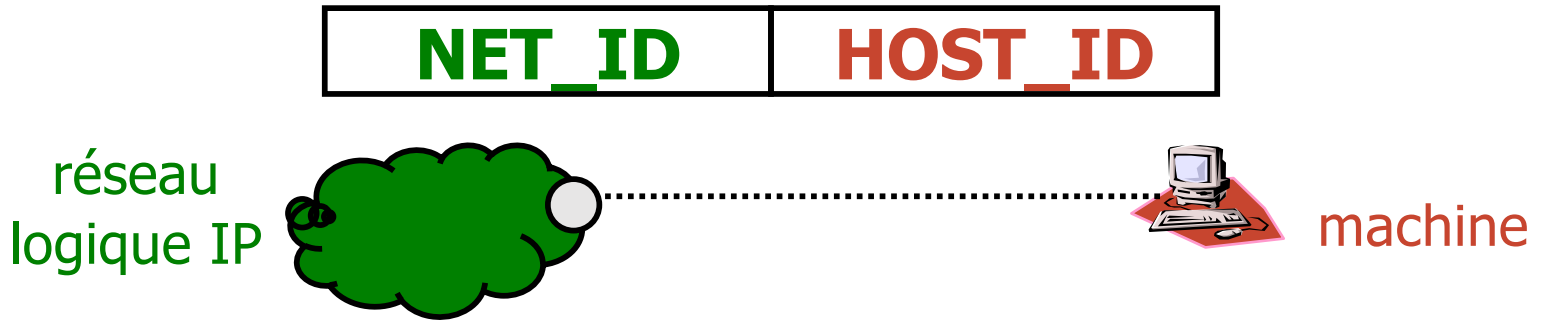
Le routage

Les adresses privées et le NAT

Format de l'en-tête, fragmentation

Format de l'adresse IPv4

- L'internet se décompose en plusieurs réseaux logiques IP
- L'adresse IP est composée de deux champs
  - NET_ID : identifiant du réseau IP (utilisé pour le routage)
  - HOST_ID : identifiant de la machine dans le réseau IP



- Adresse IP = 32 bits = 4 octets (représentée par 4 valeurs décimales [0-255] séparées par un .)

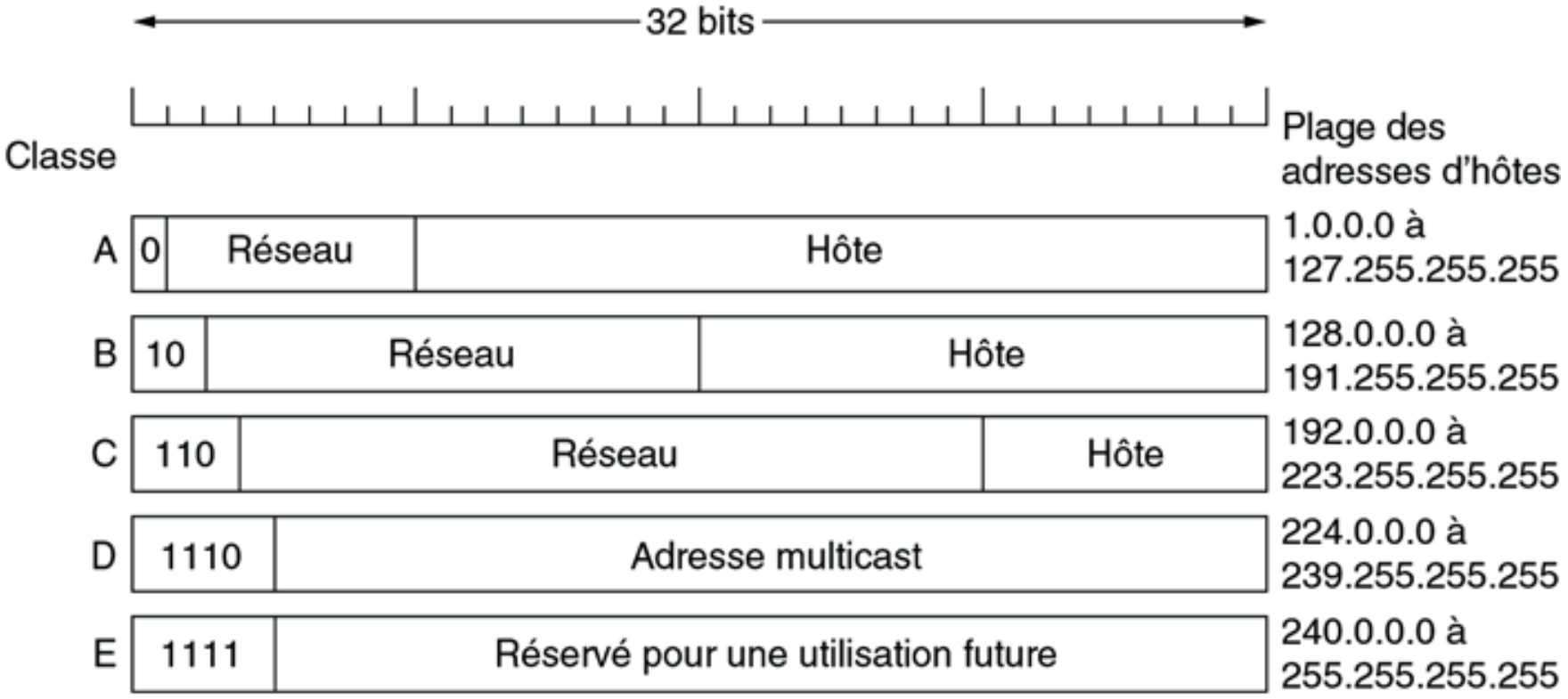
  Exemples : 8.8.8.8 134.214.92.8
  0.0.0.0 255.255.255.255

Héxadécimal / Décimal / Binaire

Conversions au format hexadécimal des octets binaires

Hexadécimal	Décimal	Binaire
00	0	0000 0000
01	1	0000 0001
02	2	0000 0010
03	3	0000 0011
04	4	0000 0100
05	5	0000 0101
06	6	0000 0110
07	7	0000 0111
08	8	0000 1000
0A	10	0000 1010
0F	15	0000 1111
10	16	0001 0000
20	32	0010 0000
40	64	0100 0000
80	128	1000 0000
C0	192	1100 0000
CA	202	1100 1010
F0	240	1111 0000
FF	255	1111 1111

Les classes d'adresses IPv4



- Les adresses réseaux sont distribuées par un organisme international à but non lucratif : ICANN (Internet Corporation for Assigned Names and Numbers) puis décentralisé au niveau de chaque pays

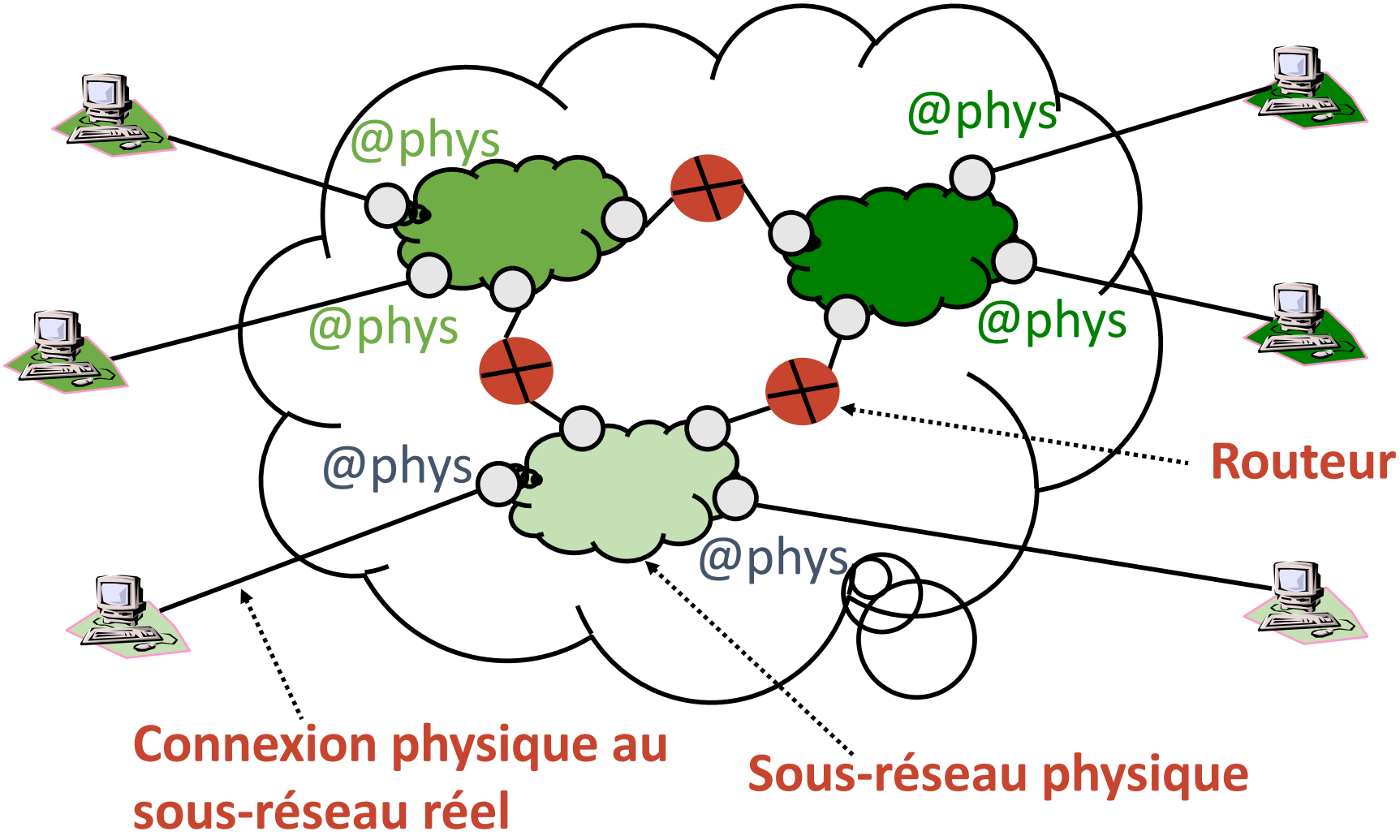
Adresses IPv4 particulières

- Diffusions locale et distante
  - 255.255.255.255 : adresse de broadcast sur le réseau IP local (ne traverse pas le routeur, traduit en broadcast ARP)
  - <NET_ID><111…111> : adresse de broadcast dirigée vers le réseau de numéro NET_ID (exemple : 132.227.255.255 = diffusion dans le réseau 132.227.0.0/16 traduit en broadcast ARP par le routeur destination)
- Rebouclage local (*loopback*) : 127.x.y.z
  - généralement 127.0.0.1 (*localhost*)
  - permet de tester la pile TCP/IP locale sans passer par une interface matérielle
- l'adresse 0.0.0.0
  - attribuée à une machine qui n'a pas encore d'adresse
  - adresse de la route par défaut qui englobe tout l'Internet

Les adresses privées IPv4

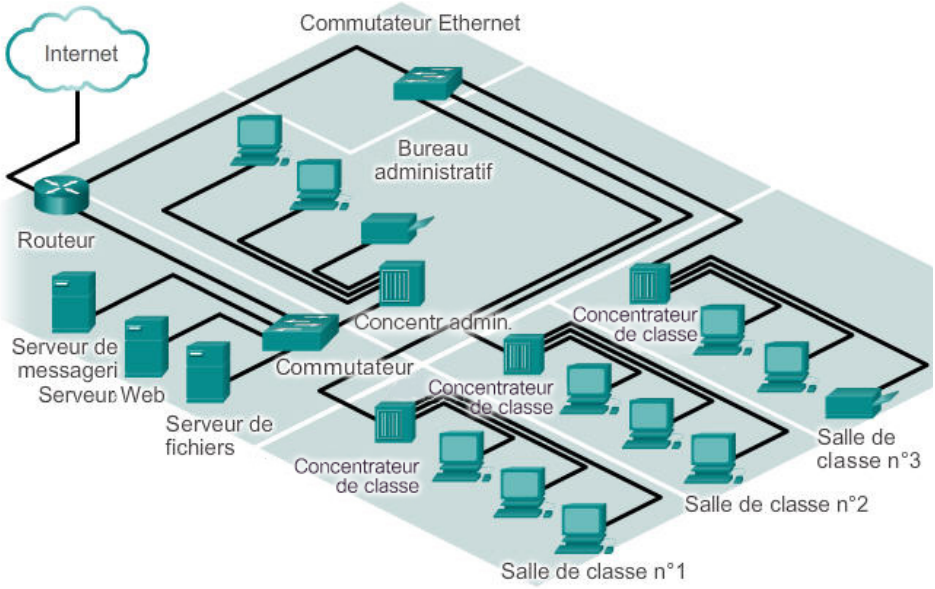
- Adresses privées (RFC 1918)
  - des adresses qui ne seront jamais attribuées (usage strictement privé) et qui ne sont pas routées sur l'Internet
  - classe A : **de 10.0.0.0 à 10.255.255.255**
  - classe B : **de 172.16.0.0 à 172.31.255.255**
  - classe C : **de 192.168.0.0 à 192.168.255.255**
- Si une entreprise qui utilise des adresses privées souhaitent tout de même disposer d'une connexion à l'Internet, il faut
  - demander une adresse publique
  - faire des conversions adresse privée <--> adresse publique (Network Address Translation)

Internet du point de vue réel

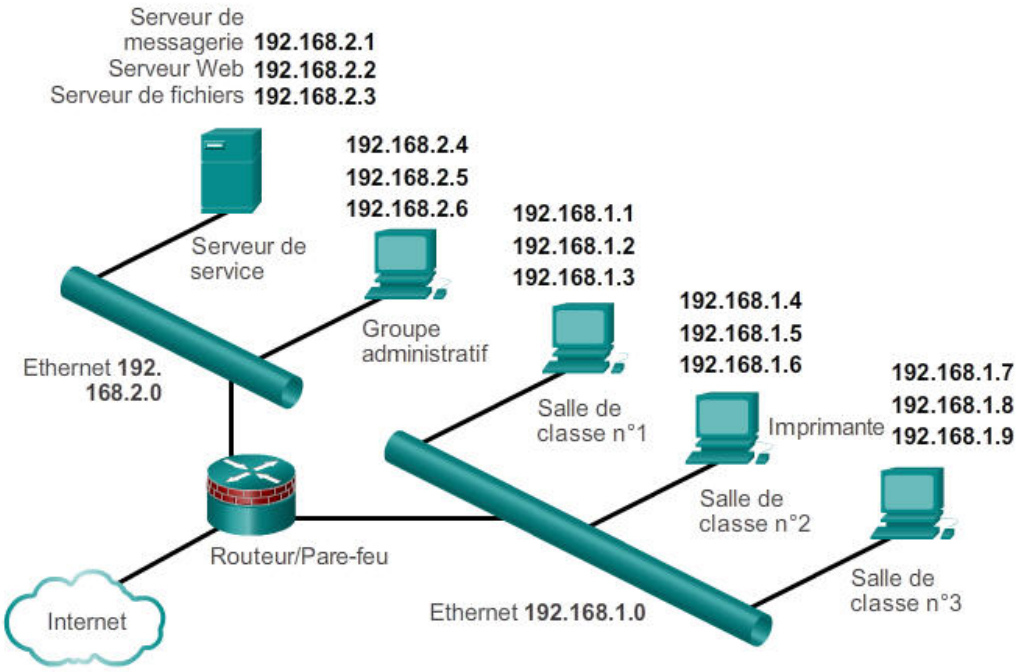


Topologie physique/logique

Topologie physique



Topologie logique

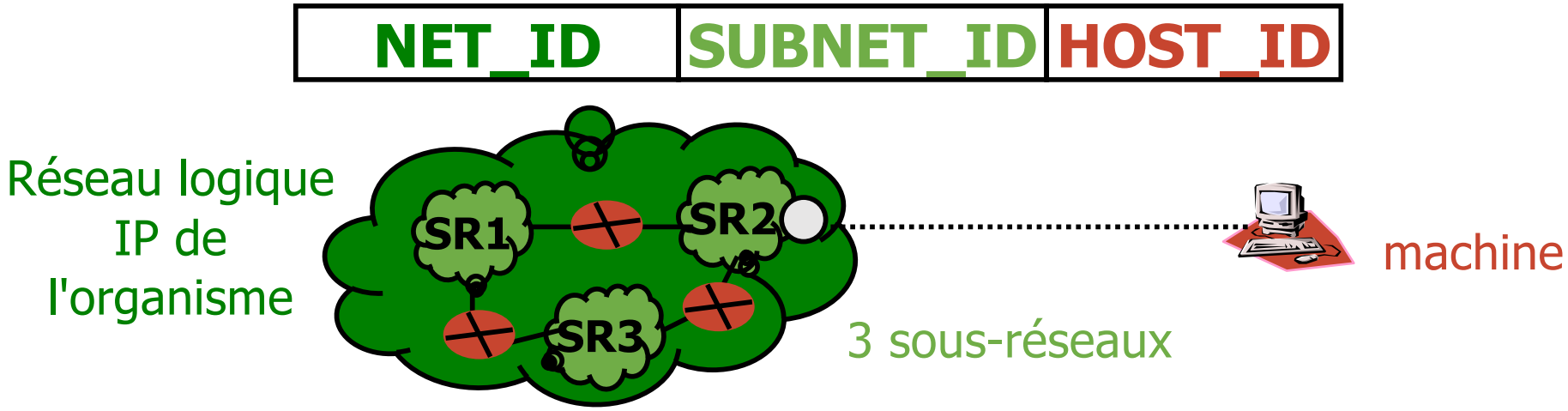


Les sous-réseaux

- Une organisation dispose généralement d'une seule adresse de réseau IP mais est composée de plusieurs sites/départements
  - Il est possible de diviser un réseau IP en plusieurs sous-réseaux

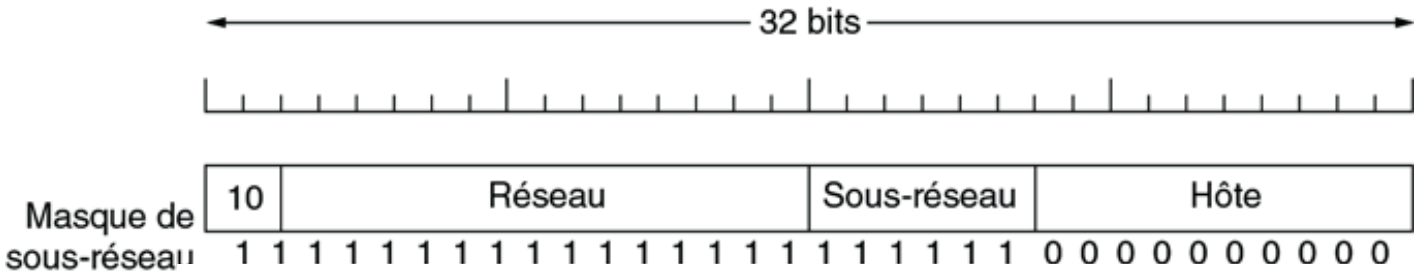
  -> prendre quelques bits dans la partie <HOST_ID> de l'adresse IP pour distinguer les sous-réseaux

  -> transparent vis à vis de l'extérieur



Masque de sous-réseau

- Masque de sous-réseau (*Netmask*)
  - Permet de donner la taille du sous-réseau
  - Se note comme une adresse IP avec tous les bits à 1 dans <NET_ID><SUBNET_ID>
- Exemple : 134.214.0.0 attribuée à l'UCBL
  - divisée en 64 sous-réseaux : 134.214.0.0, 134.214.4.0, 134.214.8.0, …, 134.214.248.0, 134.214.252.0
  - netmask = **255.255.252.0** ou **/22** (22 bits pour désigner le sous-réseau, il reste 10 bits pour les machines)



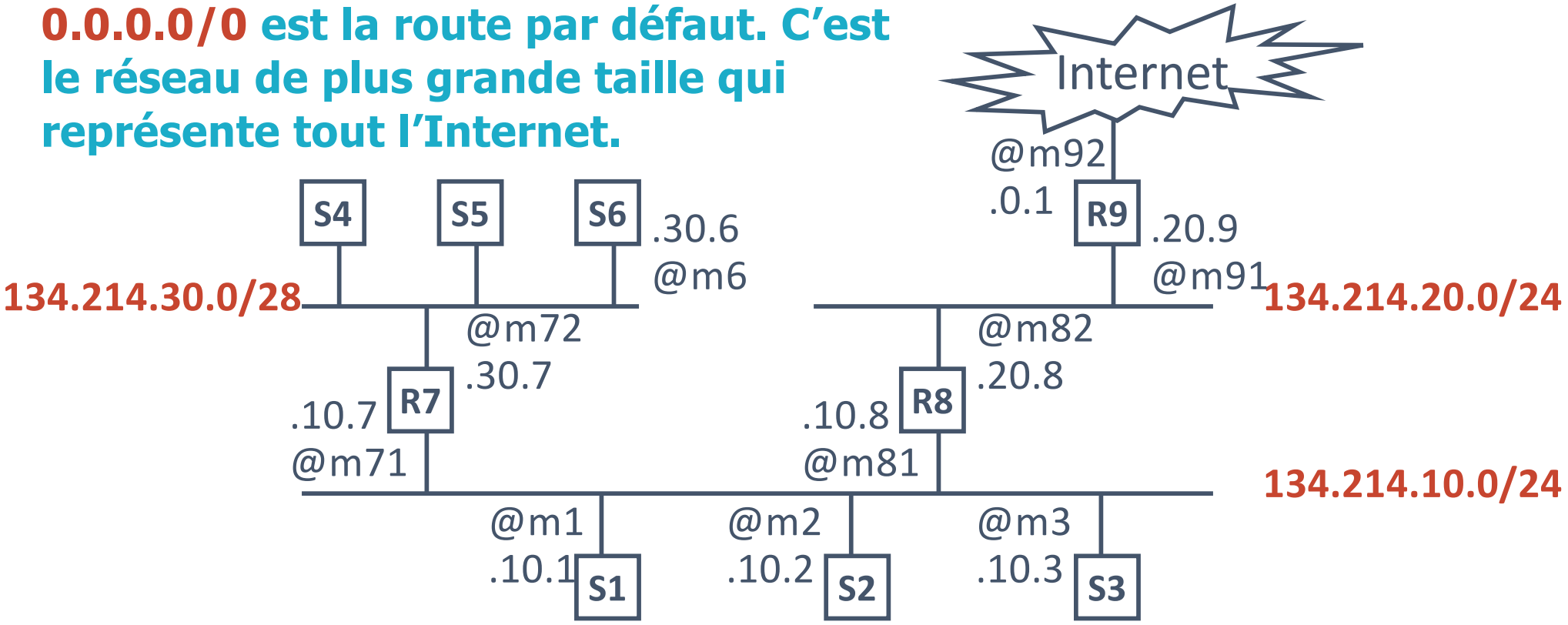
Adresse de classe B dont 6 bits sont réservés à la numérotation des sous-réseaux

Exemple de routage IPv4

Table de routage de S2

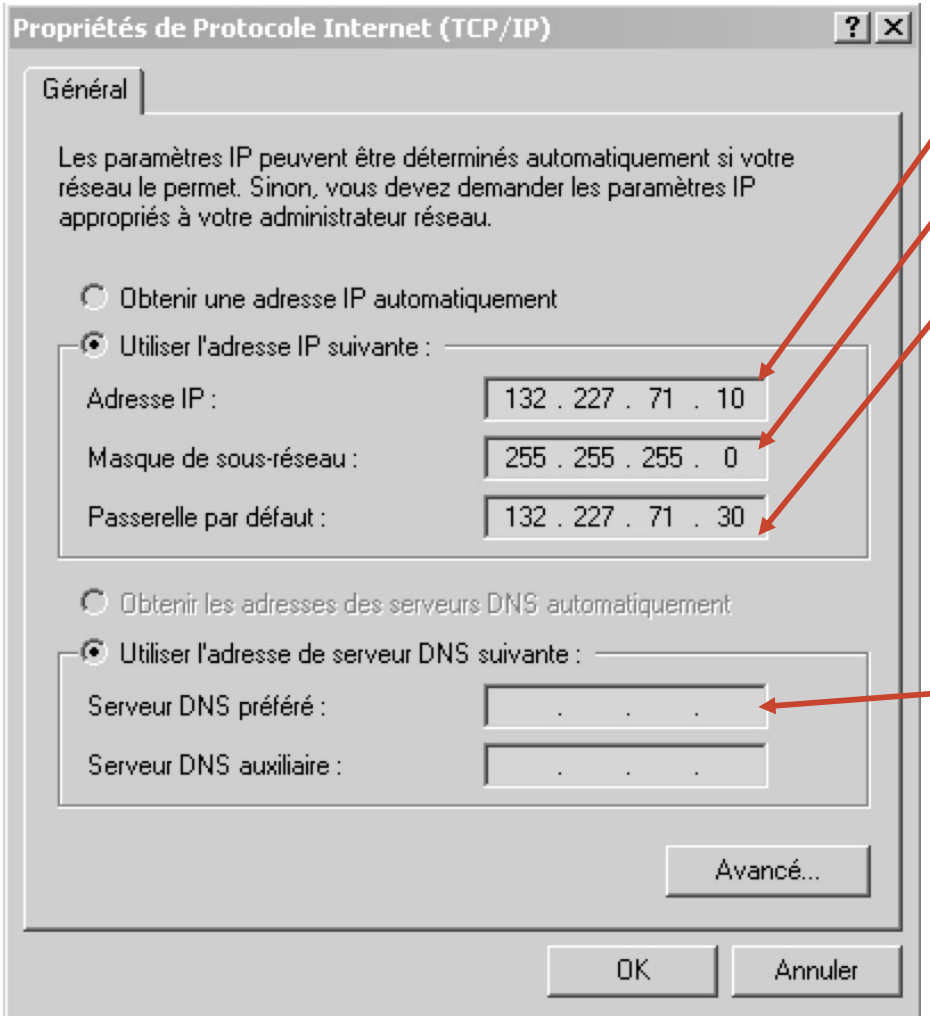
destination	netmask	gateway	int	cost
134.214.10.0	255.255.255.0	-	eth0	0
134.214.30.0	255.255.255.240	134.214.10.7	-	1
default	0.0.0.0 ou /0	134.214.10.8	-	-

0.0.0.0/0 est la route par défaut. C'est le réseau de plus grande taille qui représente tout l'Internet.



Configuration d'une interface réseau (1)

- Pour une machine d'extrémité, il suffit d'indiquer



Son adresse IP

Le masque de sous-réseau

L'adresse IP du routeur par défaut (tous les paquets qui ne sont pas à destination du sous-réseau de la machine sont envoyés vers ce routeur)

Eventuellement, l'adresse IP d'un serveur de noms pour faire les requêtes DNS

Configuration d'une interface réseau (2)

- Le fait de configurer une interface réseau en lui assignant une adresse IP et un masque ajoute une ligne dans la table de routage qui permet de joindre toutes les machines qui sont dans le même réseau qu'elle
- Par exemple,

```
ip addr add 192.168.9.3/22 dev eth0
ip link set up dev eth0
```

ajoute la ligne suivante dans le table de routage :

Destination	Masque	Passerelle	Interface
192.168.8.0	**255.255.252.0**	**-**	**eth0**

La carte `eth0` est dans le réseau `192.168.8.0/22` et permet de joindre toutes les machines de `192.168.8.1` à `192.168.11.254` (dernière machine du réseau)

Petits calculs sur les adresses réseaux (1)

- Soit le réseau, `192.168.8.0/23`

Le masque `/23` est équivalent à `255.255.254.0`

Il donne la taille du réseau c'est-à-dire le nombre de machines qu'il contient : `2^(32-23)-2 = 510`

En effet, il reste 9 bits pour numéroter les machines du réseau soit 512 adresses mais il faut retirer l'adresse du réseau et l'adresse de diffusion que l'on ne peut pas attribuer à une machine

Adresse du réseau : `192.168.8.0/23`

Adresse de diffusion : `192.168.9.255`

Première machine du réseau : `192.168.8.1`

Dernière machine du réseau : `192.168.9.254`

Petits calculs sur les adresses réseaux (2)

- Soit le réseau, `192.168.8.64/27`

La partie réseau de l'adresse est constituée des 3 premiers octets (24 bits) + les bits 2^7, 2^6 et 2^5 du 4ème octet.

Comme 2^5=32, les réseaux de taille `/27` sont les multiples de 32 dans le dernier octet. `192.168.8.64/27` est donc bien l'adresse d'un réseau car 64 est un multiple de 32.

L'adresse du réseau suivant est le multiple de 32 suivant soit

Adresse du réseau suivant : `192.168.8.96/27`

Adresse du réseau précédant : `192.168.8.32/27`

Adresse du réseau : `192.168.8.64/27`

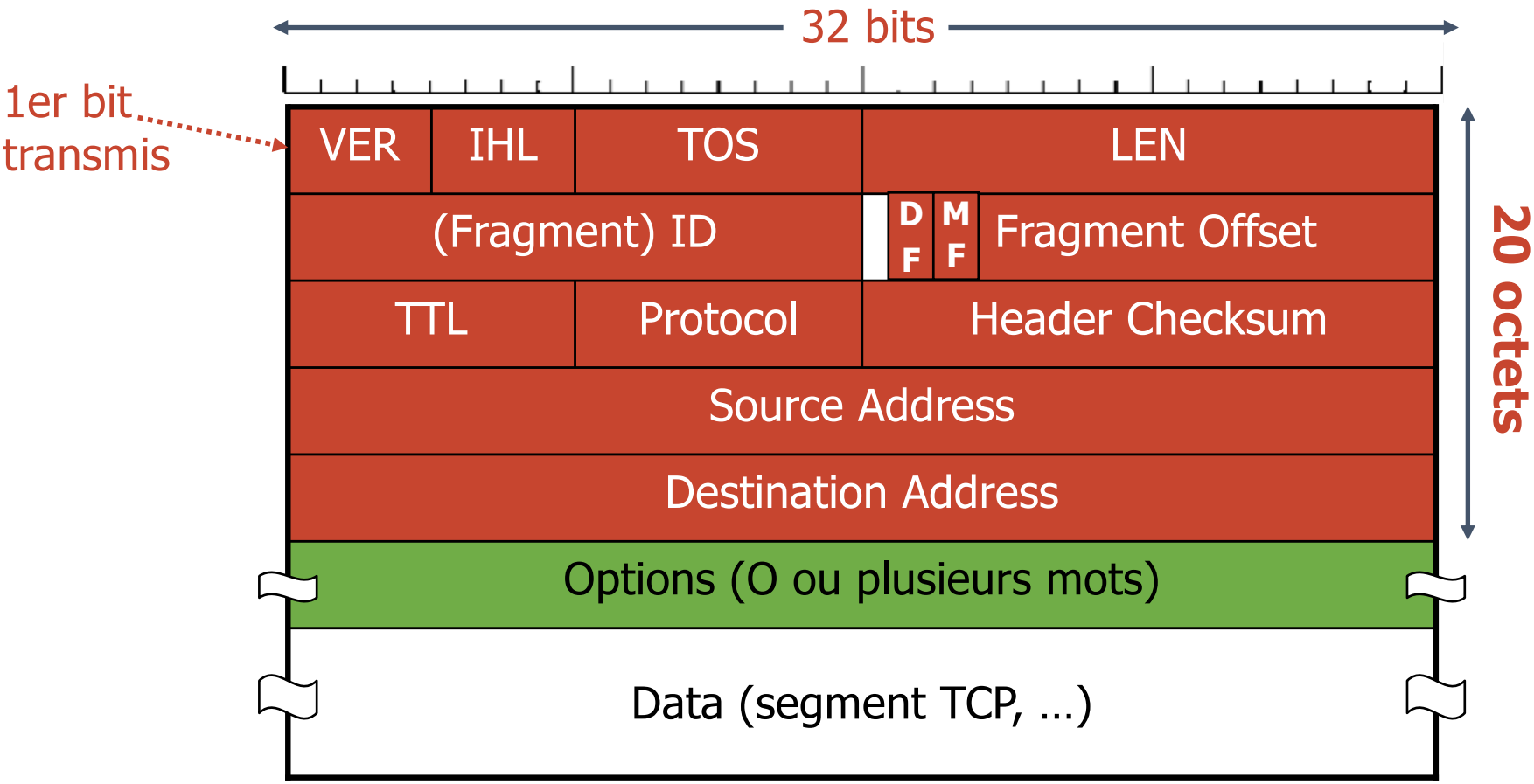
Première machine = @réseau+1 : `192.168.8.65`

Dernière machine =@diffusion-1: `192.168.8.94`

Adresse de diffusion = @réseau_suivant-1 : `192.168.8.95`

Format de l'en-tête IPv4

- Un en-tête de 20 octets + une partie facultative de longueur variable (options)

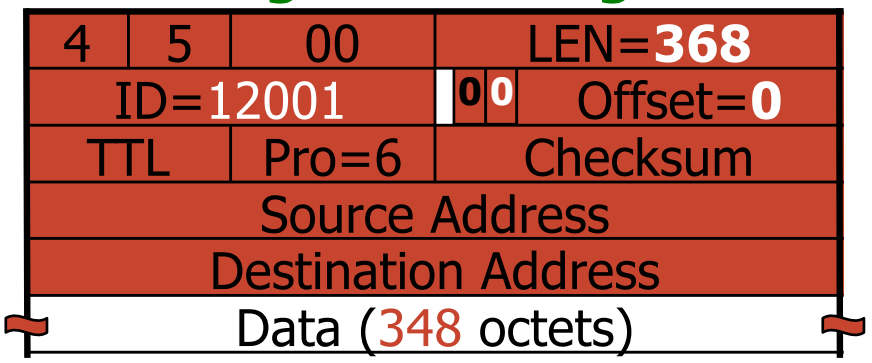


La fragmentation des datagrammes IPv4

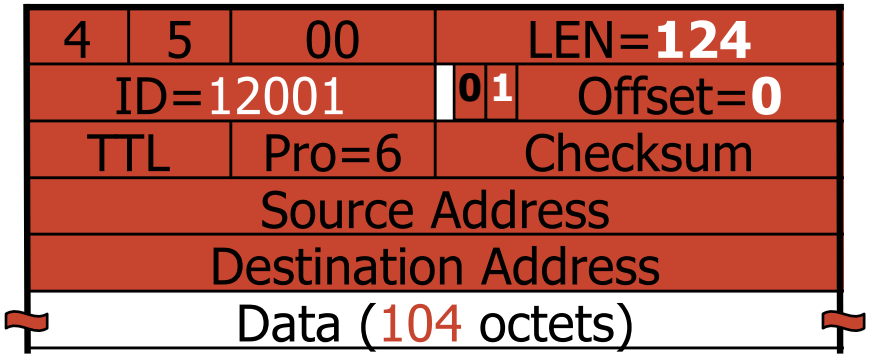
Exemple (valeurs en décimal) :

MTU de 128 octets (soit 108 octets de données IP par fragment), l'offset devant être un multiple de 8 octets -> 13*8=104 octets

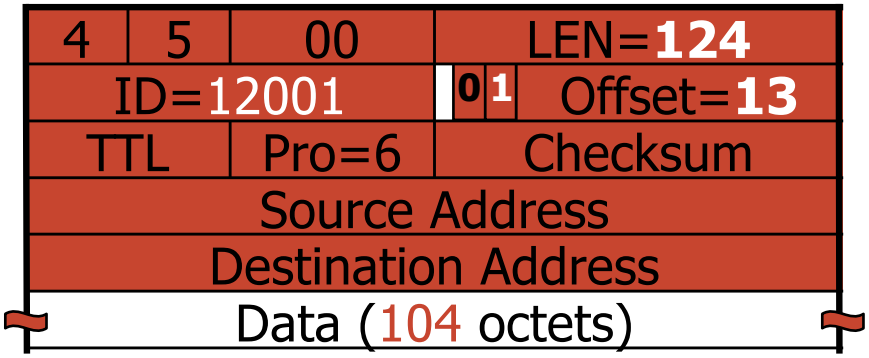
Datagramme origine



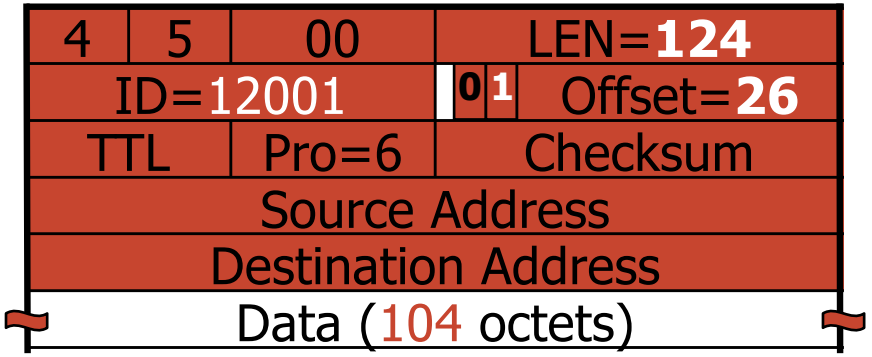
F1



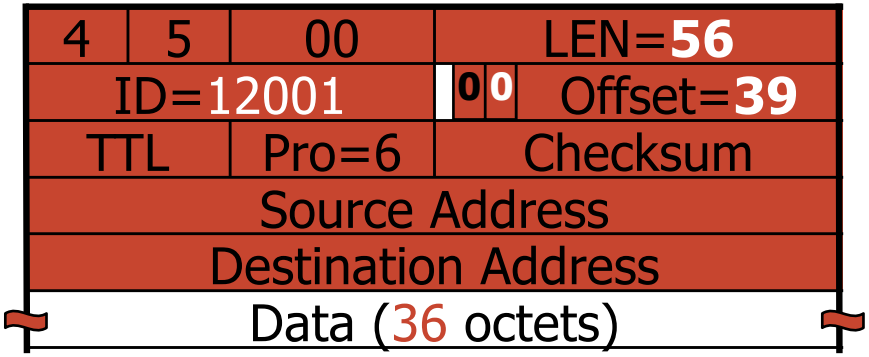
F2



F3

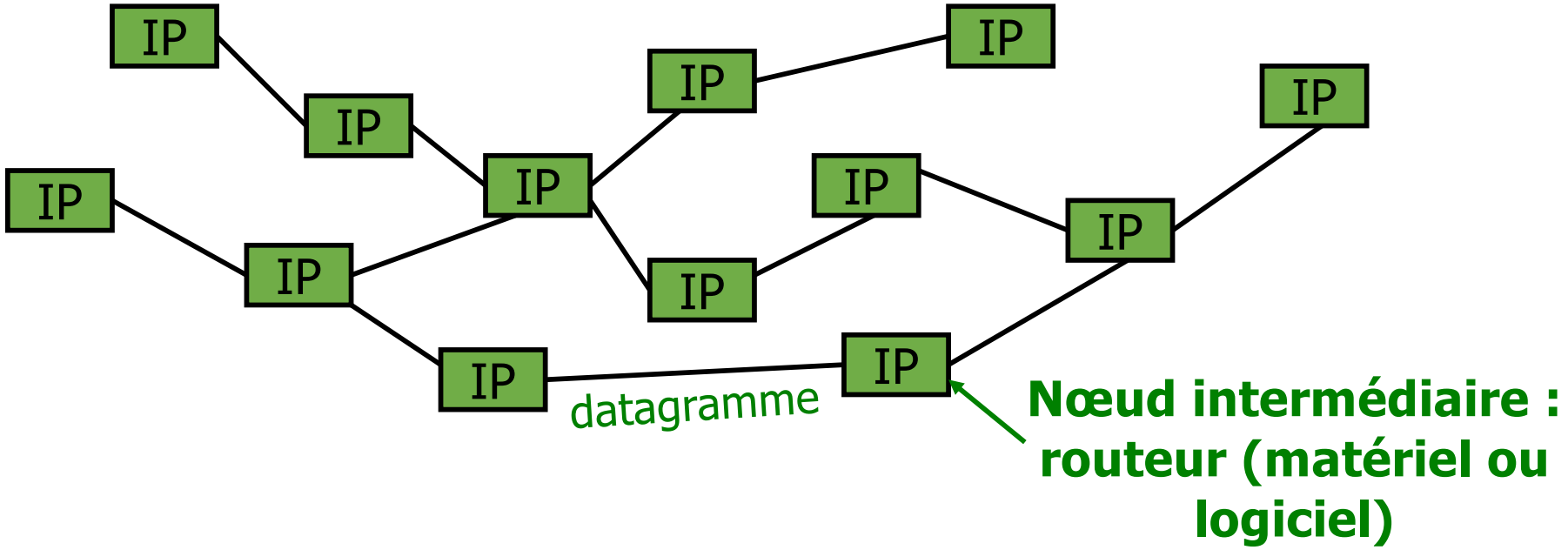


F4



Caractéristiques du protocole IP

Couche réseau : communications entre machines



- IP - protocole d'interconnexion, best-effort
  - acheminement de **datagrammes** (mode **non connecté**)
  - peu de fonctionnalités, pas de garanties
  - simple mais robuste (à la défaillance d'un nœud intermédiaire)

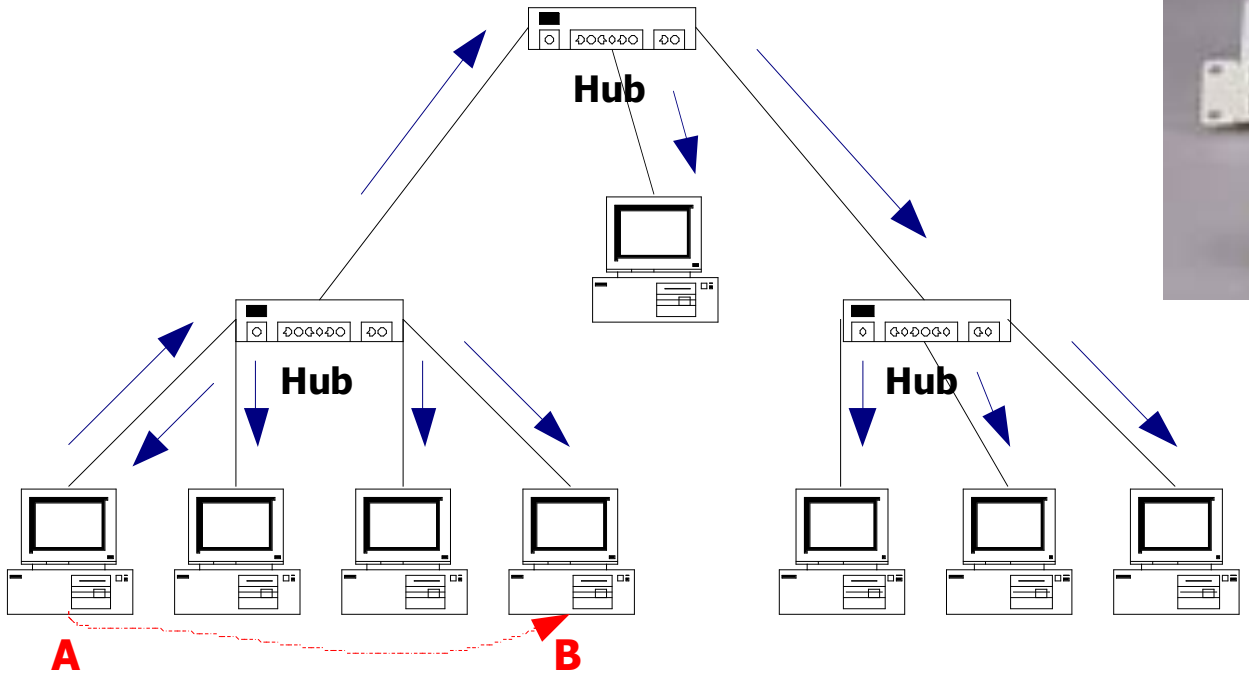
UNIVERSITÉ DE LYON
Université Claude Bernard Lyon 1
Faculté des Sciences et Technologies
Dpt Informatique

Les protocoles Ethernet, ARP et ICMP

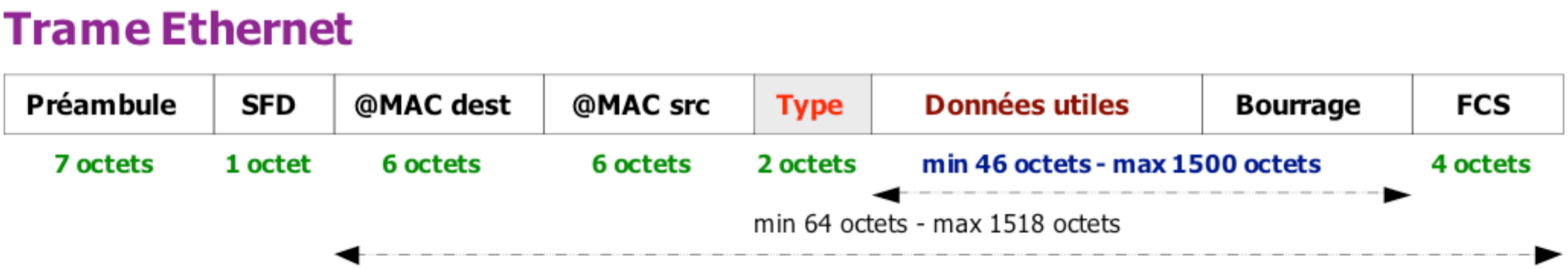
Format de la trame Ethernet
Les protocoles ARP et ICMP

Ethernet, le protocole des cartes réseaux

- Une carte réseau Ethernet fabrique un signal qui est envoyée à une autre carte réseau Ethernet
- Les répéteurs (Hub) ou commutateurs (Switch) interconnectent les ordinateurs entre eux
- Exemple avec des Hub



Format de la trame Ethernet

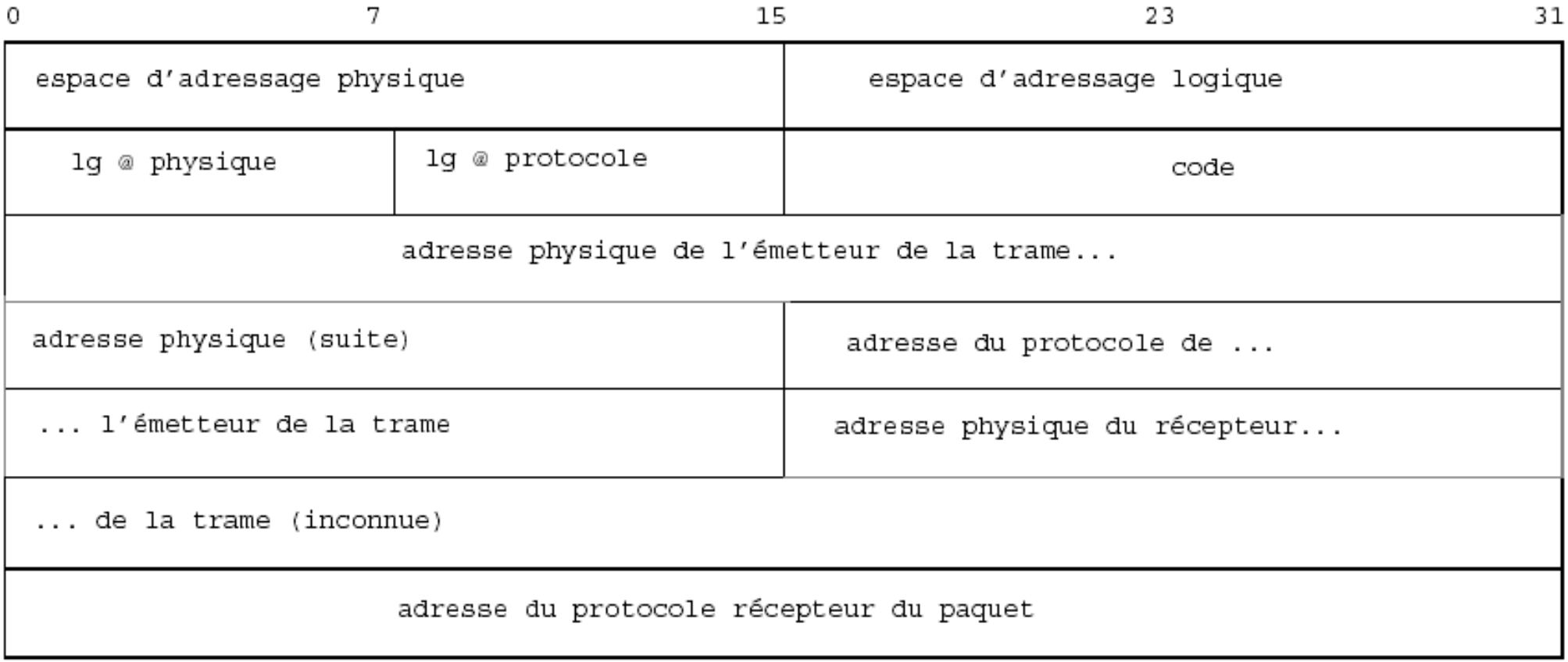


- Préambule : 7 fois 10101010 pour la synchronisation bit
- SFD (*Start Frame Delimitor*) : 10101011 pour la synchronisation octet
- Bourrage si Lg < 46 octets pour détection collision
- FCS sur 4 octets pour la détection d'erreur
- Différence IEEE 802.3/Ethernet : champ Lg/Type

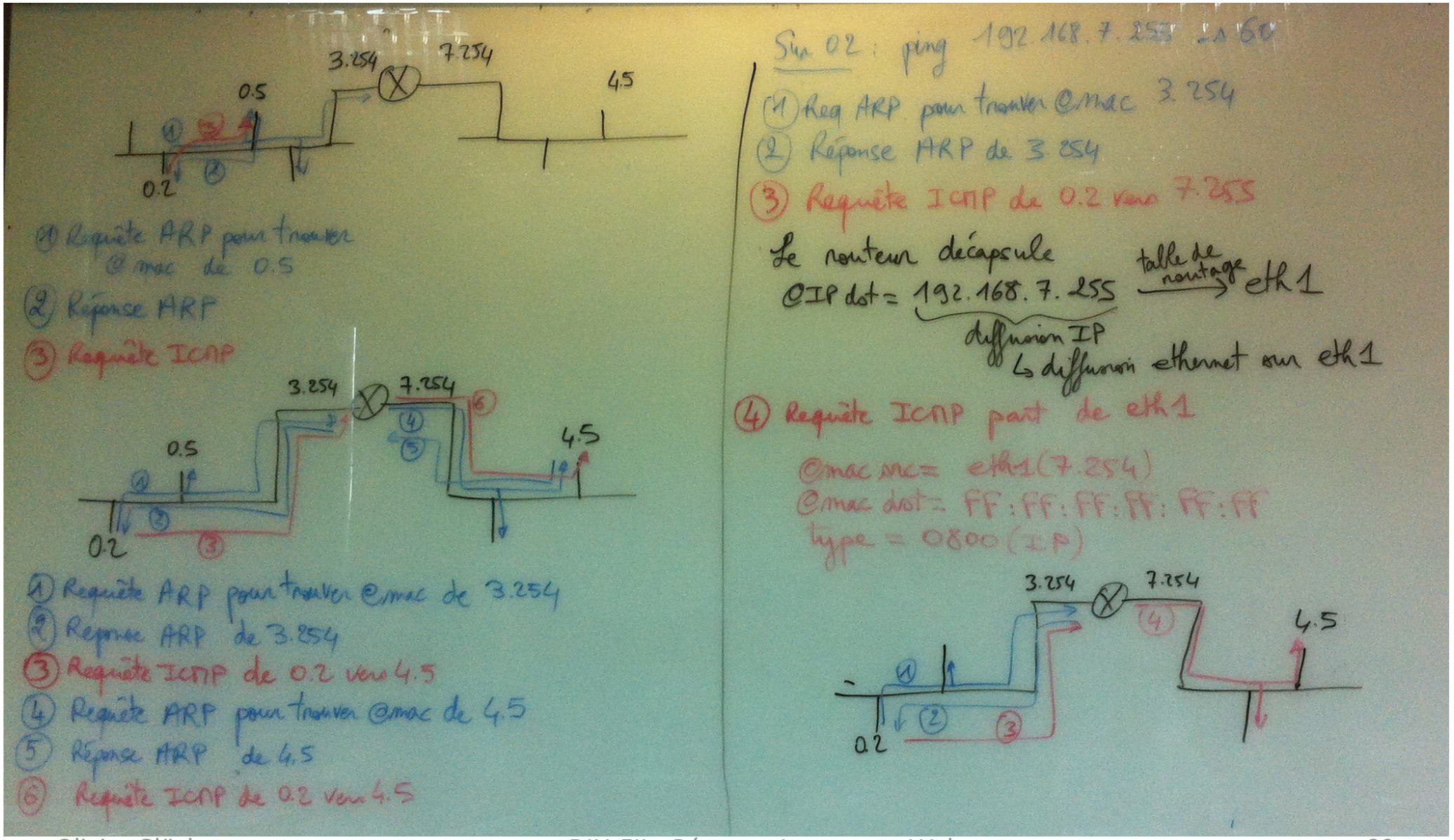
ARP – Rôle et fonctionnement

- ARP (Address Resolution Protocol) sert à trouver l'adresse MAC d'une carte réseau alors que l'on connaît son adresse IP
- Si la machine source et destinataire sont sur le même réseau
  - 1 - requête ARP (broadcast MAC)
  - 2 - réponse ARP (le destinataire a reçu le broadcast et s'est reconnu, il envoie son @MAC)
  - 3 - la source peut envoyer ses données vers le destinataire (adresse MAC destination connue)
- Si elles ne sont pas sur le même réseau
  - la diffusion ne passe pas le routeur
  - résolution de proche en proche : le client envoie ses données au prochain saut dont l'adresse MAC est trouvée par ARP, puis le prochain envoie au suivant en utilisant sa table de routage et ARP…

ARP - Format du paquet



ARP - ICMP


3.254
7.254
0.5
4.5
0.2
① Requête ARP pour trouver @mac de 0.5
② Réponse ARP
③ Requête ICMP
3.254
7.254
0.5
4.5
0.2
① Requête ARP pour trouver @mac de 3.254
② Réponse ARP de 3.254
③ Requête ICMP de 0.2 vers 4.5
④ Requête ARP pour trouver @mac de 4.5
⑤ Réponse ARP de 4.5
⑥ Requête ICMP de 0.2 vers 4.5
Sur 0.2 : ping 192.168.7.255 -c 60
① Req ARP pour trouver @mac 3.254
② Réponse ARP de 3.254
③ Requête ICMP de 0.2 vers 7.255
Le routeur décapsule
@IP dst = 192.168.7.255
table de routage
eth1
diffusion IP
↳ diffusion ethernet sur eth1
④ Requête ICMP part de eth1
@mac src = eth1(7.254)
@mac dst = FF:FF:FF:FF:FF:FF
type = 0800 (IP)
3.254
7.254
4.5
0.2

Sur le routeur : décapsulation IP. Comment aller vers 4.5 ?

entête IP → @IP dst = 192.168.4.5 → table de routage du routeur

interface eth0 192.168.3.254/22 — 192.168.0.0/22 → eth0

eth1 192.168.7.254/22 — 192.168.4.0/22 → eth1

Pour aller vers 4.5, on passe par eth1

↳ @mac src = eth1 (7.254)

type = 0800 (IP)

@mac dst = ? celle de 4.5 que le routeur ne connait pas

④ Requête ARP

ARP [@mac src = eth1 (7.254)
@IP src = 192.168.7.254
@mac dst = 0 (?)
@IP dst = 192.168.4.5]

Trame 4

@mac dst = FF:FF:FF:FF:FF:FF
@mac src = eth1 (7.254)
type = 0806 (ARP)
] EH

⑤ Réponse ARP

Trame

ARP [@mac src = eth1 (7.254)
@IP src = 192.168.7.254
@mac dst = eth0 (4.5)
@IP dst = 192.168.4.5]

@mac dst = eth1 (7.254)
@mac src = eth0 (4.5)
type = 0806 (ARP)
] EH

Trame ⑥ Requête ICMP

@mac dst = eth0 (4.5)

20 entête IP [@IP dst = 192.168.4.5
@IP src = 192.168.0.2
proto = 1 (ICMP)]

UNIVERSITÉ DE LYON
Université Claude Bernard Lyon 1
Faculté des Sciences et Technologies
Dpt Informatique

Les principales applications d'Internet

Le web (HTTP)
La résolution des noms (DNS)
Le courrier électronique (SMTP, POP, IMAP, Webmail)
La connexion à distance (telnet et ssh)
Les autres applications (FTP, NFS, LDAP…)

UNIVERSITÉ DE LYON
Université Claude Bernard Lyon 1
Faculté des Sciences et Technologies
Dpt Informatique

La résolution des noms (DNS)

Les services fournis par le DNS
Un système distribué
Qu'est-ce qu'un domaine DNS ?
Les serveurs racine
Les messages DNS
La commande host

DNS : Domain Name System

- Les personnes ont plusieurs identifiants
  - Le nom mais aussi #sécu, #Passeport, #téléphone
- Les machines et les routeurs aussi
  - L'adresse IP (32 bits ou 128 bits) est l'équivalent du numéro de téléphone
  - Le nom de la machine est ce que l'on utilise dans l'URL par exemple :

    `www.univ-lyon1.fr`

    `www.education.gouv.fr`
- Le DNS fait le lien entre les adresses IP utilisées pour acheminer les paquets et les noms de machine utilisés par les utilisateurs ou les applications

DNS : Domain Name System

- C'est une base de données **distribuée**
  - Il y a plein de serveurs de noms dans le monde. Chaque serveur stocke les noms et les adresses IP dont il est responsable.
- C'est un protocole applicatif comme HTTP, SMTP…
  - Les machines clientes et les serveurs de noms communiquent pour effectuer la traduction d'un nom en adresse IP.
  - Le DNS est utilisé par les applications clientes pour trouver les adresses IP des serveurs mais n'est pas utilisé directement par l'application comme SMTP...
  - Le DNS fonctionne selon le modèle Client/Serveur comme les autres applications.
  - Le serveur utilise le port 53/UDP (ou 53/TCP mises à jour)
  - RFC 1034, 1035, 2181, …

Les services fournis par le DNS

- Le service principal : obtenir l'adresse IP d'un serveur

Requête DNS : Quelle est l'adresse IP de **`www.univ-lyon1.fr`** ?

Réponse DNS : **`134.214.126.72`**

```
olivier.gluck@lifasr2:~$ host www.univ-lyon1.fr
www.univ-lyon1.fr is an alias for ksup.univ-lyon1.fr.
ksup.univ-lyon1.fr has address 134.214.126.72
```

- Autres exemples de services fournis par le DNS
  - Donner plusieurs noms à une machine (Alias)
  - Donner plusieurs adresses IP à un serveur (Répartition de la charge)
  - Trouver le nom d'un serveur mail (*Mail server aliasing*)

```
olivier.gluck@lifasr2:~$ host univ-lyon1.fr | grep mail
univ-lyon1.fr mail is handled by 5 smtpbv.univ-lyon1.fr.
olivier.gluck@lifasr2:~$ host smtpbv.univ-lyon1.fr
smtpbv.univ-lyon1.fr has address 134.214.126.92
```

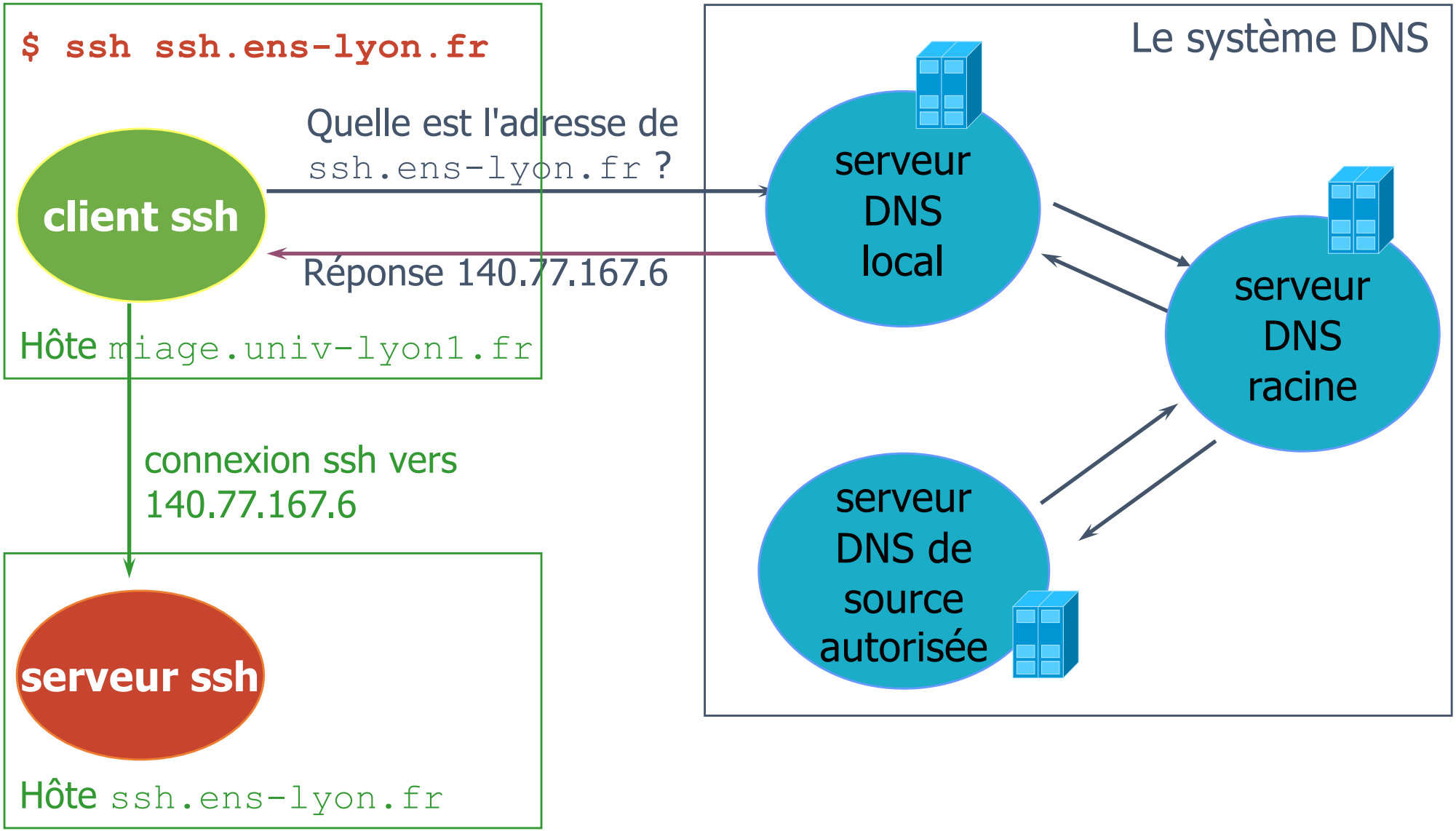
Pourquoi le DNS est un système distribué ?

- Pour l'utilisateur, le DNS n'est qu'une boîte noire mais en réalité très compliquée
  - Une requête DNS peut impliquer plusieurs serveurs de noms répartis dans le monde entier
- Pourquoi pas de DNS centralisé ? Un seul serveur contiendrait toutes les correspondances requises par les applications de l'Internet
  - Dimension de l'Internet : trop de correspondances à gérer, nombre de requêtes au serveur trop important
  - Tolérance aux pannes : si le serveur DNS tombe, tout l'Internet aussi !
  - Volume de trafic impossible à supporter par un seul serveur
  - Délais de réponse : il faut faire en sorte que la réponse soit la plus proche possible du demandeur
  - Problème lié à la maintenance et aux mises à jour perpétuelles de la base

Un système distribué

- Aucun serveur ne peut connaître toutes les correspondances nom <--> adresse IP
  - Si un serveur ne connaît pas une correspondance, il interroge un autre serveur jusqu'à atteindre le serveur détenant l'information souhaitée
- Trois types de serveur DNS
  - **Les serveurs de noms locaux :** c'est au serveur local que les applications clientes envoient toutes leurs requêtes
  - **Les serveurs de noms racine :** si un serveur local n'a pas la réponse, il transmet la requête à un serveur racine ; un serveur de noms racine connaît au moins les serveurs de source autorisée du premier niveau (`.fr.`, …)
  - **Les serveurs de noms de source autorisée :** un serveur de source autorisée contient les informations "officielles" de sa zone DNS ; il a autorité sur sa zone

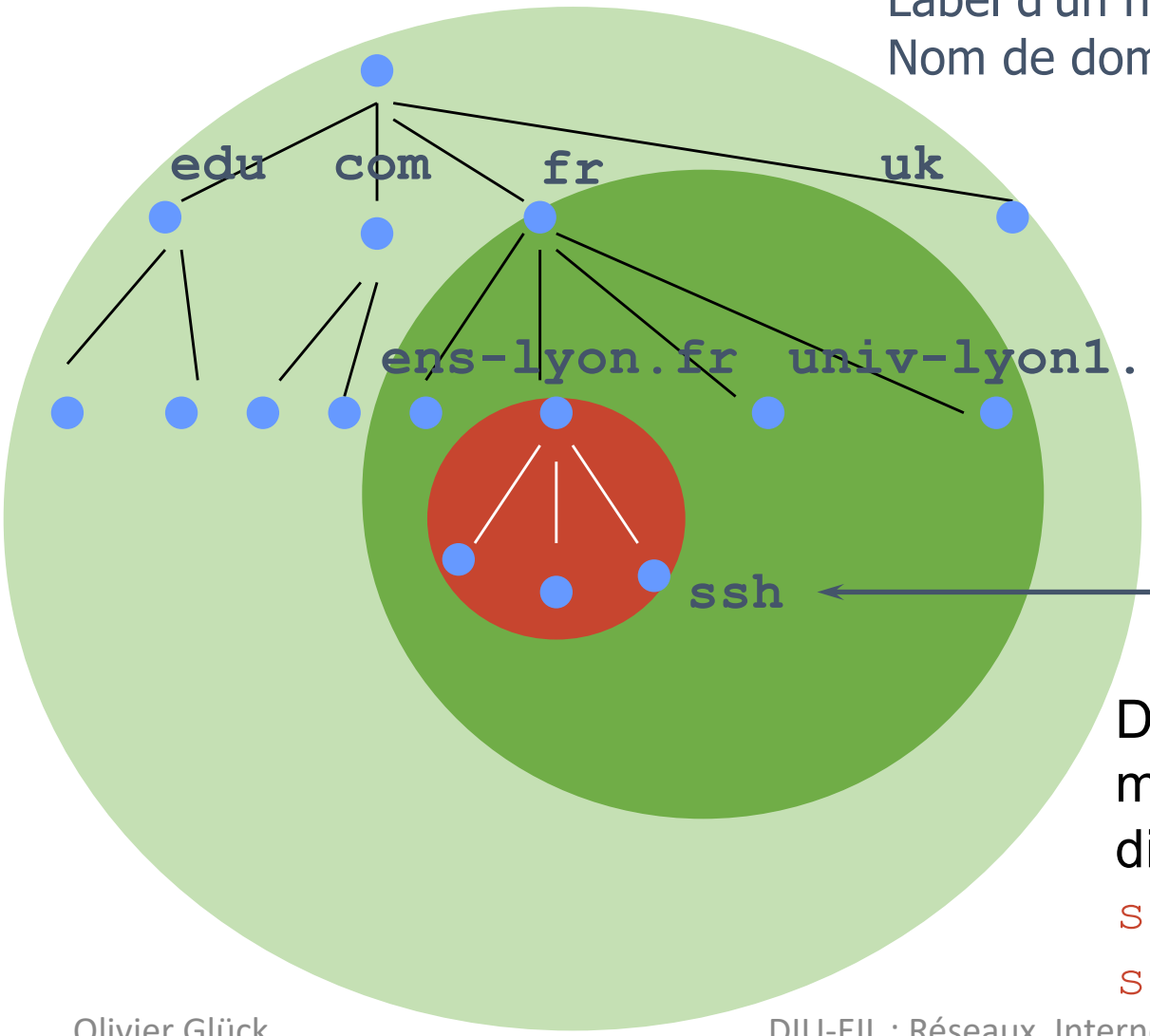
Un système distribué



Qu’est-ce qu’un domaine DNS ?

Un domaine est un sous-arbre entier de l’espace de nommage

Label d'un nœud : 63 caractères max.
Nom de domaine ou hôte : 255 caractères max.



Domaine complet

Domaine `fr`

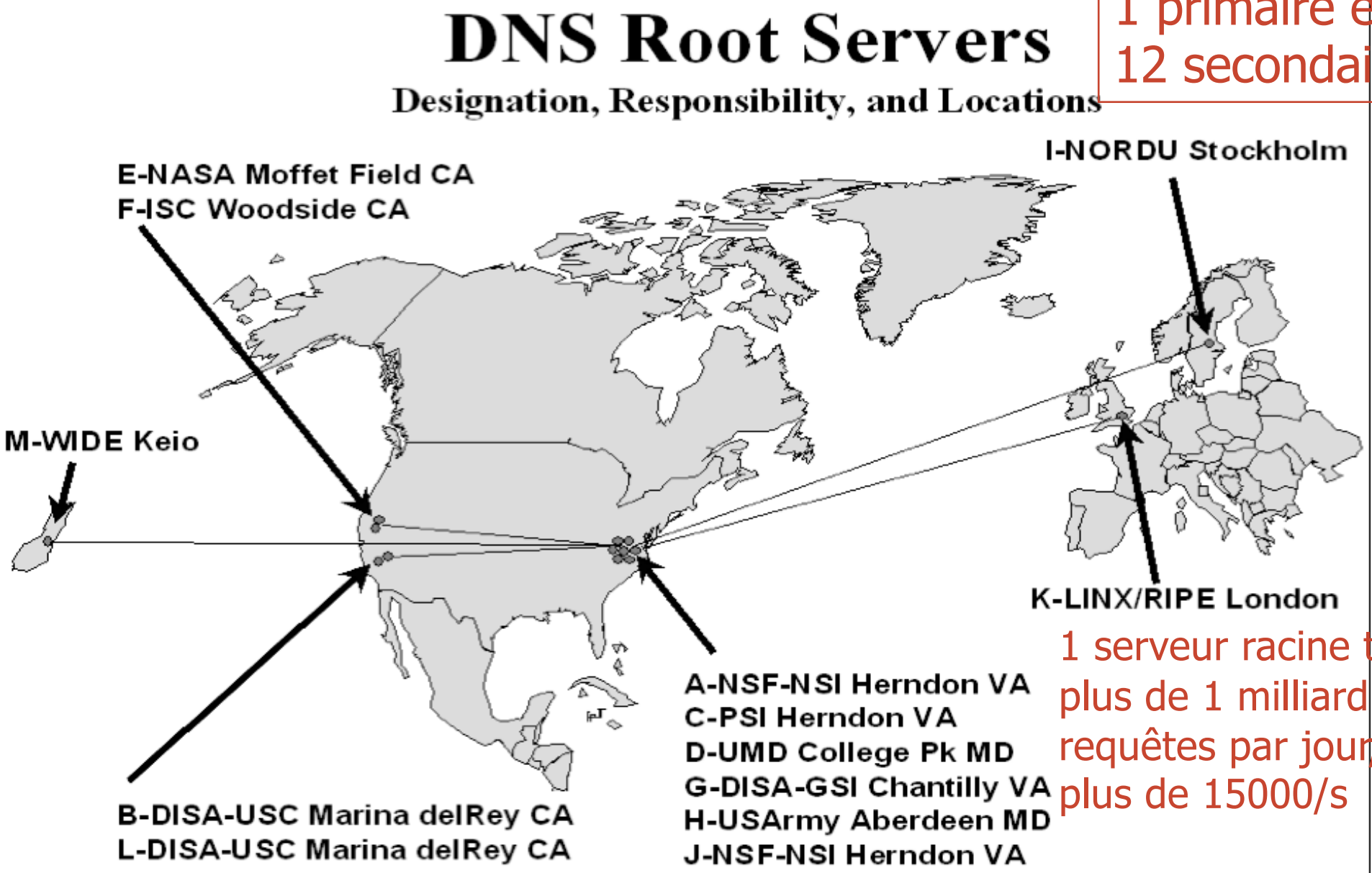
Domaine `ens-lyon.fr`

hôte **`ssh.ens-lyon.fr`**

Deux noeuds peuvent avoir le même nom dans des domaines différents :
`ssh.ens-lyon.fr` et
`ssh.univ-lyon1.fr`

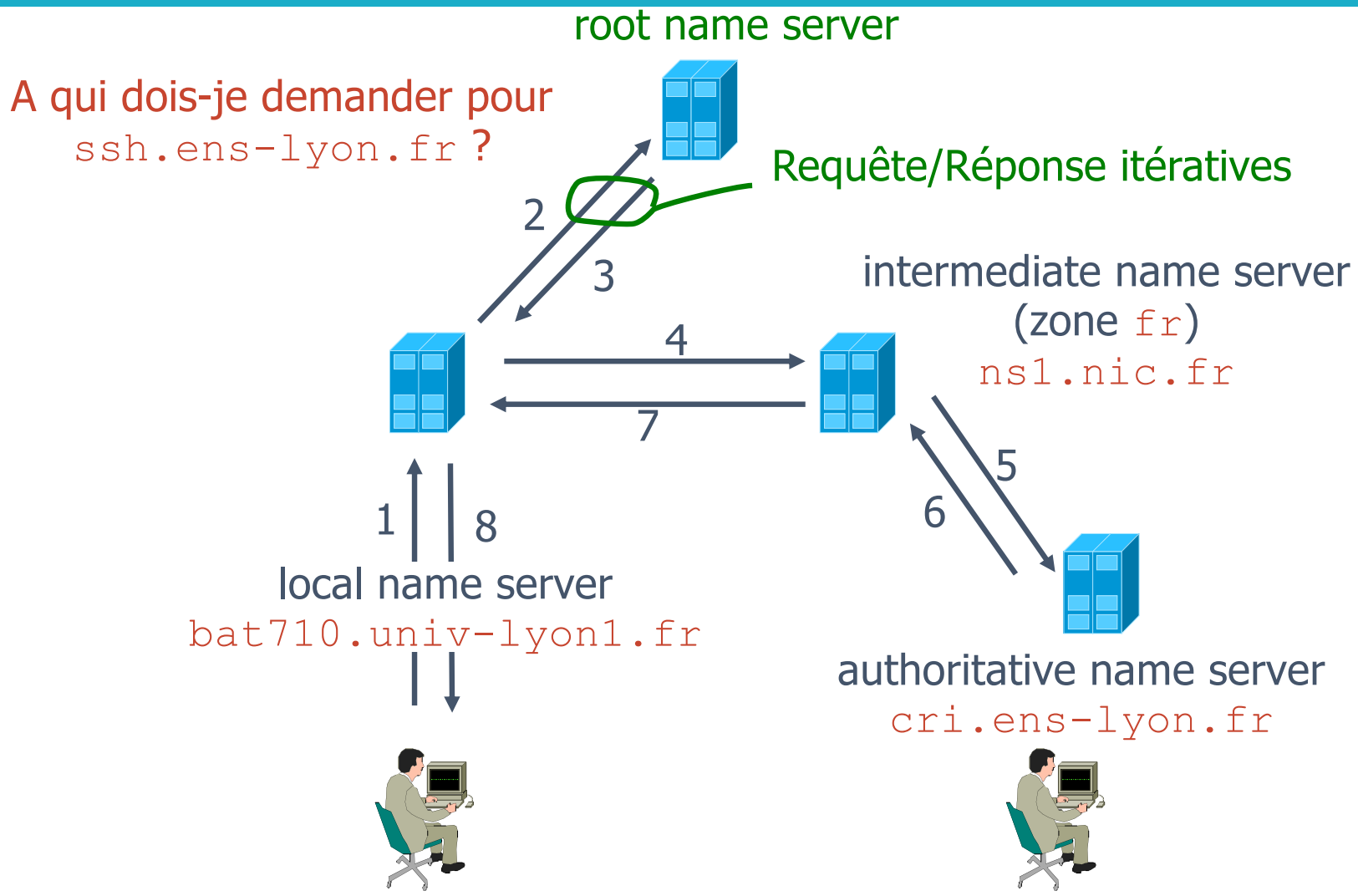
Les serveurs racine

1 primaire et
12 secondaires

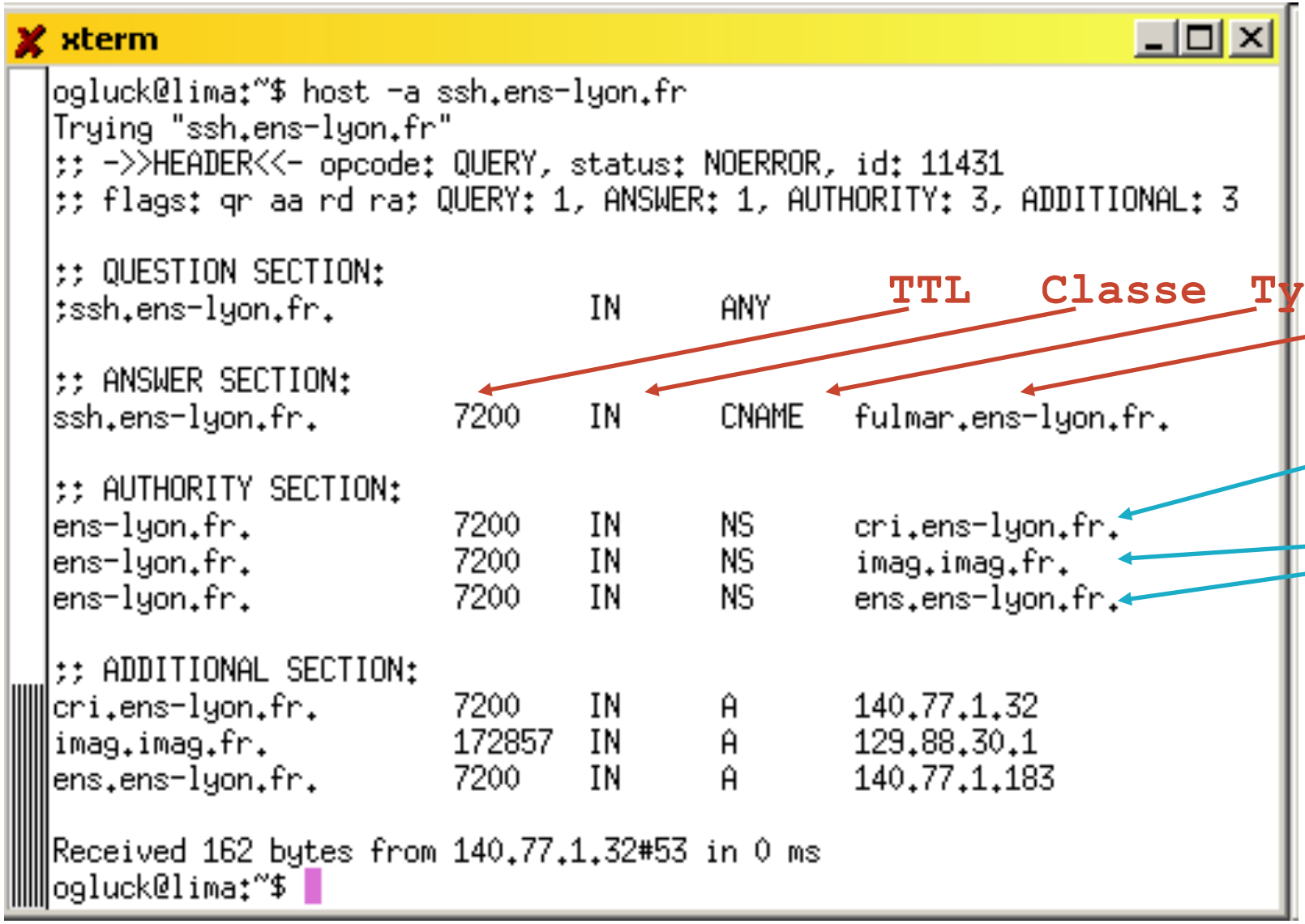


1 serveur racine traite plus de 1 milliard de requêtes par jour, soit plus de 15000/s

Principe d'une résolution de nom



Les messages DNS [RFC 1034, 1035]



Les enregistrements stockés par les serveurs

- **`Type=A`** (val=1) : sert à décrire une correspondance

  `Nom`=nom d'hôte (canonique), `Value`=@IPv4
- **`Type=AAAA`** (val=28, RFC 1886) : idem mais adresse IPv6

  `Nom`=nom d'hôte, `Value`=@IPv6
- **`Type=PTR`** (val=12) : sert à la résolution inverse

  `Nom`=un nom de la zone `arpa`, `Value`=nom canonique (valeur pointée)
- **`Type=NS`** (val=2) : sert à associer un nom de domaine à un serveur de noms de source autorisée

  `Nom`=domaine, `Value`=nom du serveur de noms
- **`Type=CNAME`** (val=5) : sert à définir un alias pour un hôte

  `Nom`=un alias, `Value`=nom canonique (le vrai nom)

Les enregistrements stockés par les serveurs

- **Type=MX** (val=15) : alias réservés aux serveurs mail permettant d'associer plusieurs serveurs de mail avec différentes priorités à une même adresse (RFC 974)

  `Nom`=un alias, `Value`=nom canonique d'un serveur de mail

- **Type=SOA** (val=6) : sert à donner des infos sur la zone

  `Nom`=nom d'une zone, `Value`=informations sur la zone

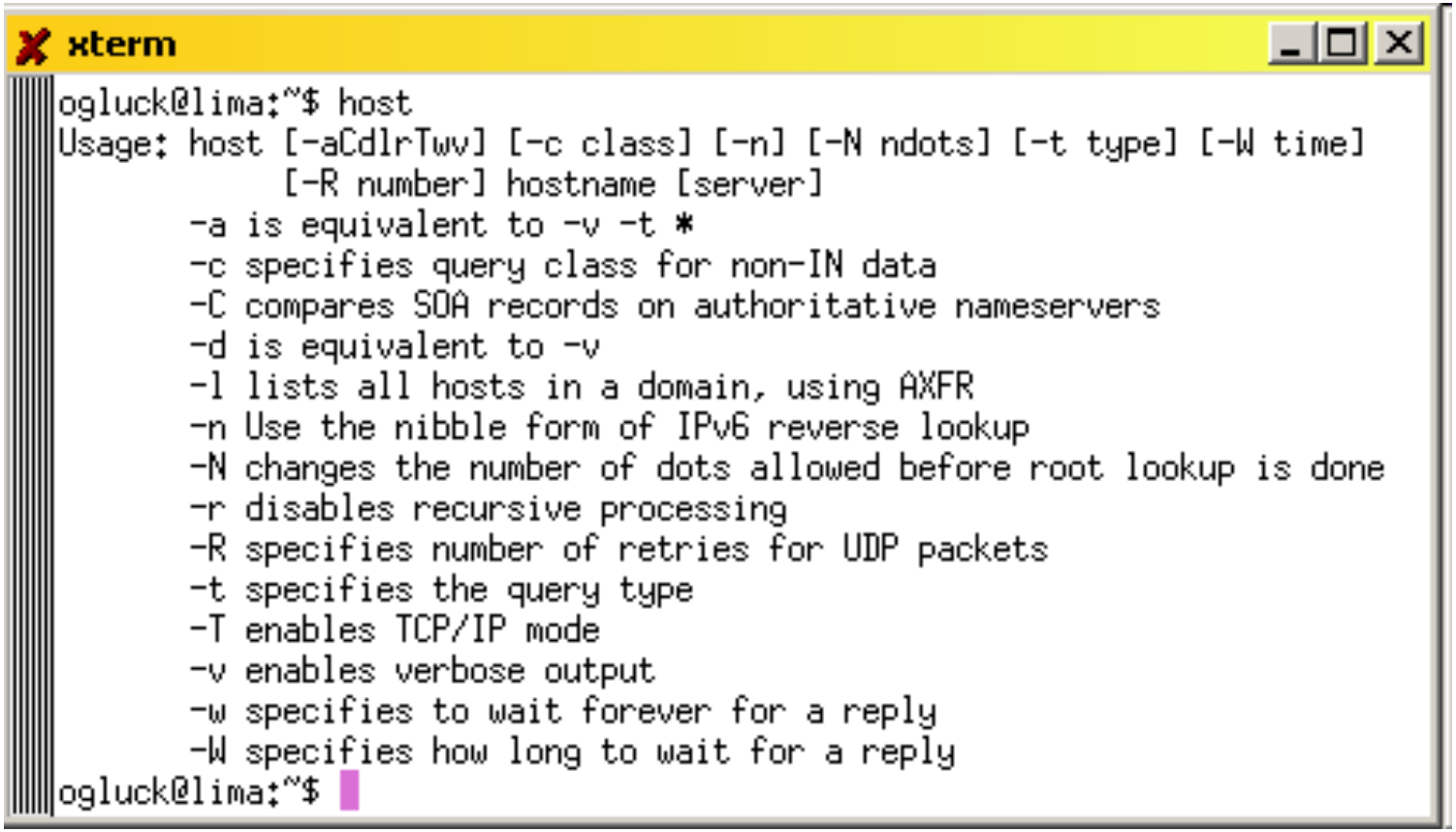
- **Type=ANY** (val=255) : utilisé dans les requêtes pour indiquer n'importe quel type (*)
- **Type=AXFR** (val=252) : utilisé dans les requêtes pour demander le transfert d'une zone entière (mise à jour d'un serveur secondaire…)
- **Type=HINFO** (val=13) : sert à indiquer les CPU et OS du serveur interrogé

Les enregistrements stockés par les serveurs

Exemples :

```
ssh.ens-lyon.fr.              CNAME   fulmar.ens-lyon.fr.
ens-lyon.fr.                  NS      cri.ens-lyon.fr.
ens-lyon.fr.                  NS      ens.ens-lyon.fr.
cri.ens-lyon.fr.              A       140.77.1.32
relaissmtp.ens-lyon.fr.       CNAME   pluvier.ens-lyon.fr.
ens-lyon.fr.                  MX      20 pluvier.ens-lyon.fr.
ens-lyon.fr.                  MX      30 pluvier2.ens-lyon.fr.
listes.ens-lyon.fr.           MX      20 pluvier.ens-lyon.fr.
fulmar.ens-lyon.fr.           A       140.77.167.6
6.167.77.140.in-addr.arpa.    PTR     fulmar.ens-lyon.fr
```

La commande `host`

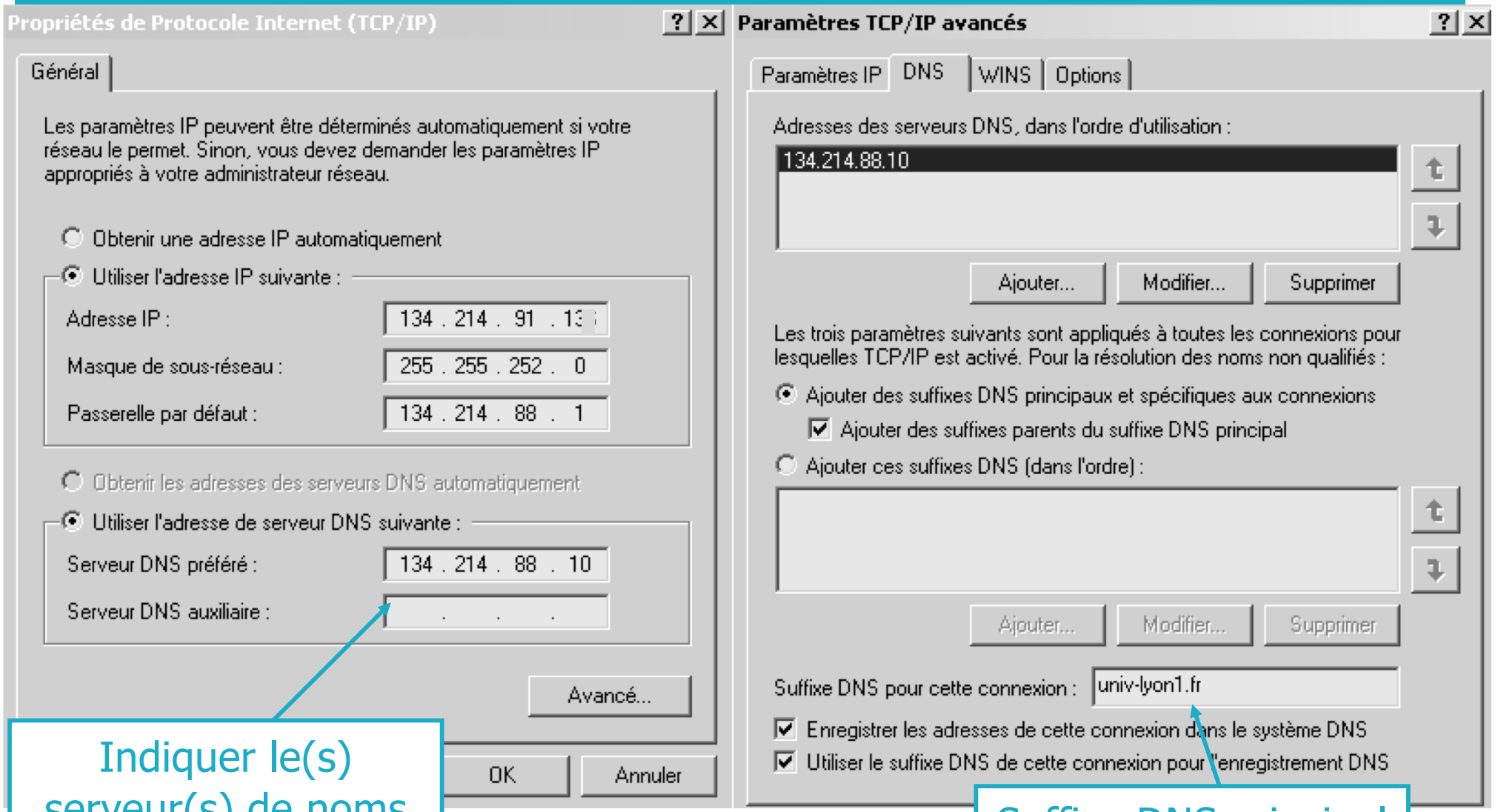



```
ogluck@lima:~$ host ssh.ens-lyon.fr
ssh.ens-lyon.fr is an alias for fulmar.ens-lyon.fr.
fulmar.ens-lyon.fr has address 140.77.167.6
```

ogluck — ssh -X lifasr2.univ-lyon1.fr -l olivier.gluck — 81×28

```
olivier.gluck@lifasr2:~$ host -a etu.univ-lyon1.fr
Trying "etu.univ-lyon1.fr"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8179
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 3, ADDITIONAL: 7

;; QUESTION SECTION:
;etu.univ-lyon1.fr.             IN      ANY

;; ANSWER SECTION:
etu.univ-lyon1.fr.      432000  IN      A       134.214.126.72
etu.univ-lyon1.fr.      3600    IN      MX      5 smtpbv.univ-lyon1.fr.

;; AUTHORITY SECTION:
univ-lyon1.fr.          432000  IN      NS      dnsi.univ-lyon1.fr.
univ-lyon1.fr.          432000  IN      NS      dns2.univ-lyon1.fr.
univ-lyon1.fr.          432000  IN      NS      dns.univ-lyon1.fr.

;; ADDITIONAL SECTION:
smtpbv.univ-lyon1.fr.   1800    IN      A       134.214.126.92
dns.univ-lyon1.fr.      432000  IN      A       134.214.100.6
dns.univ-lyon1.fr.      432000  IN      AAAA    2001:660:5001:100::6
dns2.univ-lyon1.fr.     432000  IN      A       134.214.100.245
dns2.univ-lyon1.fr.     1800    IN      AAAA    2001:660:5001:100::245
dnsi.univ-lyon1.fr.     432000  IN      A       134.214.100.9
dnsi.univ-lyon1.fr.     432000  IN      AAAA    2001:660:5001:100::9

Received 278 bytes from 10.10.10.10#53 in 2 ms
olivier.gluck@lifasr2:~$
```

Configuration d'un poste de travail



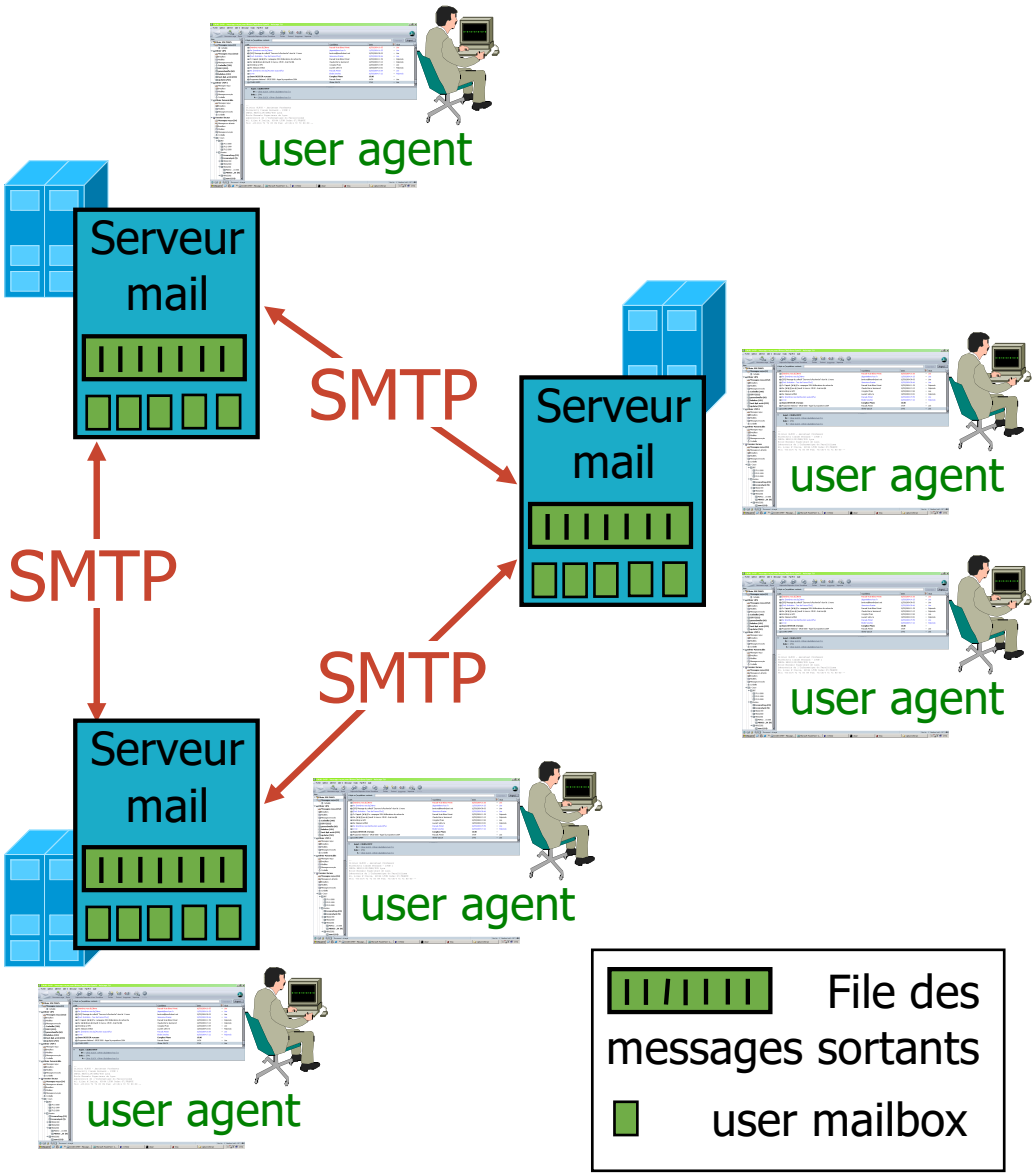
UNIVERSITÉ DE LYON
Université Claude Bernard Lyon 1
Faculté des Sciences et Technologies
Dpt Informatique

Le courrier électronique (SMTP, POP, IMAP, webmail)

Les composants du courrier électronique
La transmission d'un courriel
Configuration d'un client mail
Les types MIME
Les protocoles SMTP, POP et IMAP
Qu'est-ce qu'un Webmail ?
Format d'une adresse mail

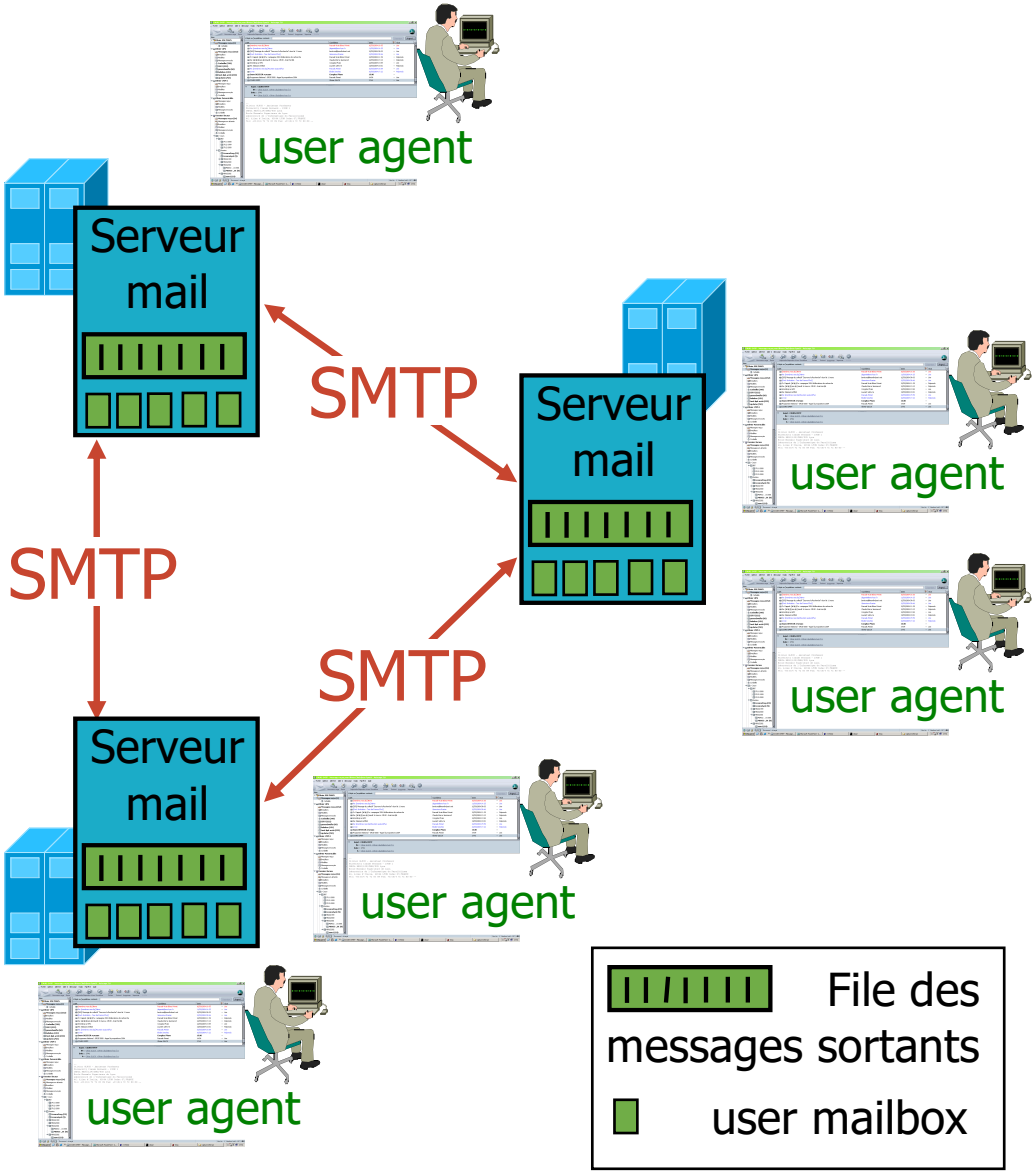
Les composants du courrier électronique

- 4 composants principaux :
  - des agents utilisateurs
  - des serveurs de mail
  - un protocole de transfert de mail : *Simple Mail Transfer Protocol* (SMTP)
  - un protocole d'accès à la boîte aux lettres (POP, IMAP, …)
- Les agents utilisateurs :
  - composition, édition, lecture du courrier électronique
  - ex : Eudora, Outlook, elm, pine, Thunderbird
  - un agent utilisateur dialogue avec un serveur pour émettre/recevoir des messages

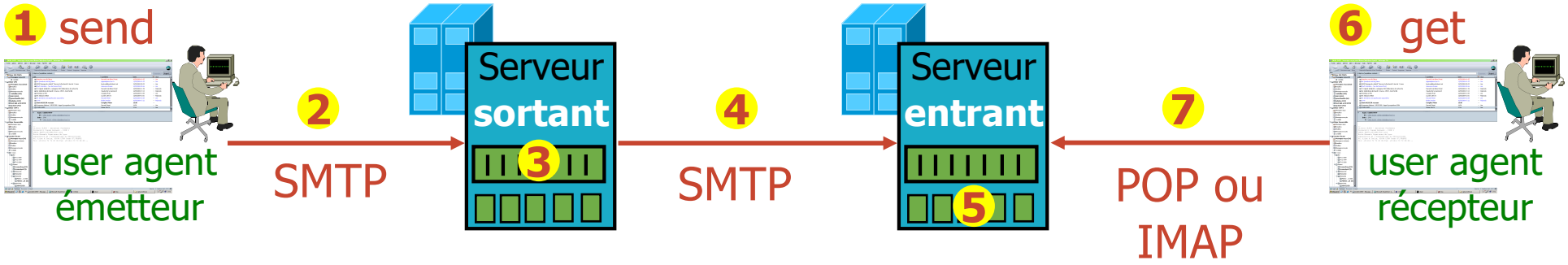


Les composants du courrier électronique

- Les messages entrants et sortants sont stockés sur le serveur
- La boîte aux lettres de chaque utilisateur contient les messages entrants (à lire)
- File d'attente des messages mail sortants (à envoyer)
- Protocole SMTP entre les serveurs de mail pour l'envoi des messages
  - modèle C/S : Client (serveur de mail émetteur) - Serveur (serveur de mail récepteur)
  - le client se connecte sur le port 25/TCP du serveur pour transférer son message



La transmission d'un courrier électronique

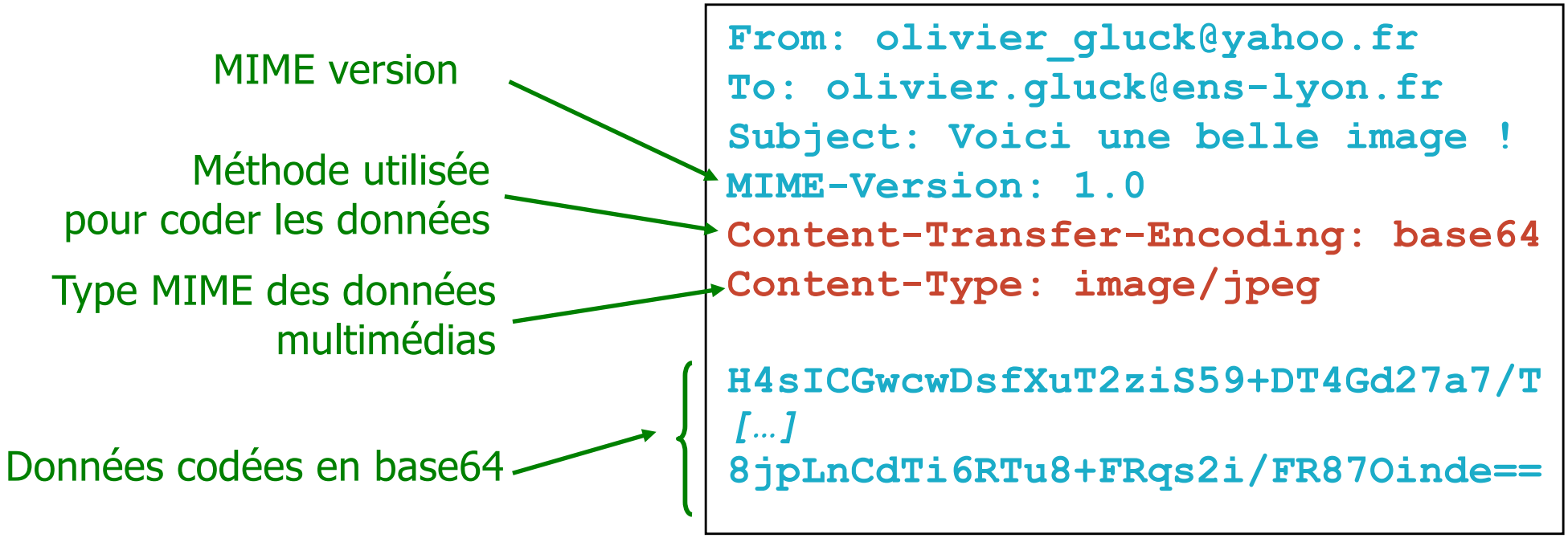


- Les protocoles d'accès : consultation de sa boîte aux lettres (après authentification)
  - POP3 : *Post Office Protocol v3* [RFC 1939]
    - autorisation (agent <--> server) et téléchargement
  - IMAP4 : *Internet Message Access Protocol* v4 [RFC 3501]
    - plus de caractéristiques, plus complexe, plus récent
    - manipulation de messages stockés sur le serveur
  - HTTP (*Webmail*) : Hotmail , Yahoo! Mail, …

Les types MIME [RFC 2045, 2056]

`Content-Type: type/subtype; parameters`

- Lignes supplémentaires dans l'en-tête du message pour déclarer un type MIME et un encodage
- Content-type est généralement positionné à partir de l'extension du document demandé (`/etc/mime.types`)



Un mail avec pièce jointe : type Multipart

```
From: olivier_gluck@yahoo.fr
To: olivier.gluck@ens-lyon.fr
Subject: Voici une belle image mais avec du texte !
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=98766789

--98766789
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain

Cher Olivier,
Voici une photo de nos dernieres vacances !
--98766789
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

H4sICGYRMTQAA3NsaWRlcy5wcwDsfXuT2ziS59+DT4Gd275a
56o7LlgSJbFNiWpSqsfw6rvLxPgSxIlVnk64i54ftRKi67/T
[…]
8jpLnCdTi6RTu8+FRqs2i/RTuy56plYbYVsa1fdvUjHrtV6g
RTf4/hy67fgIIVDfeR+rtYuNFR87Oinde==
--98766789--
```

Le protocole POP3 [RFC 1939]

- POP3 est extrêmement simple
  - permet uniquement de télécharger des messages depuis le serveur en laissant éventuellement une copie de ceux-ci dans la BAL de l'utilisateur
  - pas adapté aux utilisateurs nomades
    - impossible de gérer des répertoires sur le serveur
    - impossible de gérer les messages en les laissant sur le serveur

IMAP répond à cette problématique au prix d'un protocole beaucoup plus complexe

Le protocole IMAP [RFC 3501]

- IMAP permet la gestion distante des messages
  - Associe un message à un répertoire distant sur le serveur
  - Permet à l'utilisateur de faire une recherche dans les messages sur le serveur
  - Permet de ne consulter que des extraits de messages (par exemple que l'en-tête ou que la partie texte d'un message *multipart*…)
  - Contrairement à POP3, IMAP conserve des informations d'état sur chaque utilisateur (noms des répertoires, listes des messages qu'ils contiennent…)

Plus d'infos : `https://tools.ietf.org/html/rfc3501`

Qu'est-ce qu'un Webmail ?

- L'utilisateur utilise un **navigateur Web** comme agent utilisateur pour consulter/envoyer ses courriers
- Le navigateur fait des requêtes **HTTP** (ou HTTPS) vers un serveur Web qui s'interface avec les serveurs SMTP/IMAP
  - Le serveur HTTP exécute des scripts qui font des requêtes
    - IMAP pour communiquer avec le serveur IMAP qui stockent les messages reçus par l'utilisateur
    - SMTP pour envoyer les messages de l'utilisateur
- Avantages du Webmail
  - adapté aux utilisateurs itinérants
  - pas besoin d'un agent utilisateur particulier, seule une connexion Internet avec Navigateur Web est nécessaire

Format d'une adresse mail

- Adresse d'un destinataire : **`bal@nom_domaine`**
- Problème :
  - **`bal`** n'est pas forcément le login de l'utilisateur : souvent de la forme `prenom.nom` qui est un alias vers le login
  - **`nom_domaine`** n'est pas forcément le nom du serveur de mail contenant la boîte aux lettres pour avoir des adresses plus courtes et plus faciles à retenir
  - **`bal`** peut représenter plusieurs destinataires (liste de mail)
- Exemple : **`Olivier.Gluck@ens-lyon.fr`**

  `Olivier.Gluck` est un alias vers `/var/mail/ogluck`

  `ens-lyon.fr` pointe vers `mailhost.ens-lyon.fr`

  (enregistrement de type MX dans le DNS)

UNIVERSITÉ DE LYON
Université Claude Bernard Lyon 1
Faculté des Sciences et Technologies
Dpt Informatique

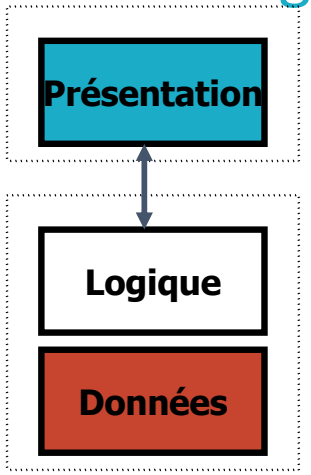
La connexion à distance (telnet, ssh et X)

Connexion locale et distante
L'application telnet
L'application ssh
Principe du chiffrement

Qu'est-ce qu'une connexion à distance ?

- Application permettant à un utilisateur de se connecter à une machine distante pour en prendre partiellement le contrôle c'est à dire **exécuter des commandes** autorisées
  - à partir d'un terminal local et à condition que cet utilisateur dispose d'un accès autorisé à cette machine (login, mot de passe…)
- Les commandes saisies localement au clavier s'exécutent sur la machine distante
  - Les environnements local et distant peuvent être hétérogènes (windows-->unix, …).

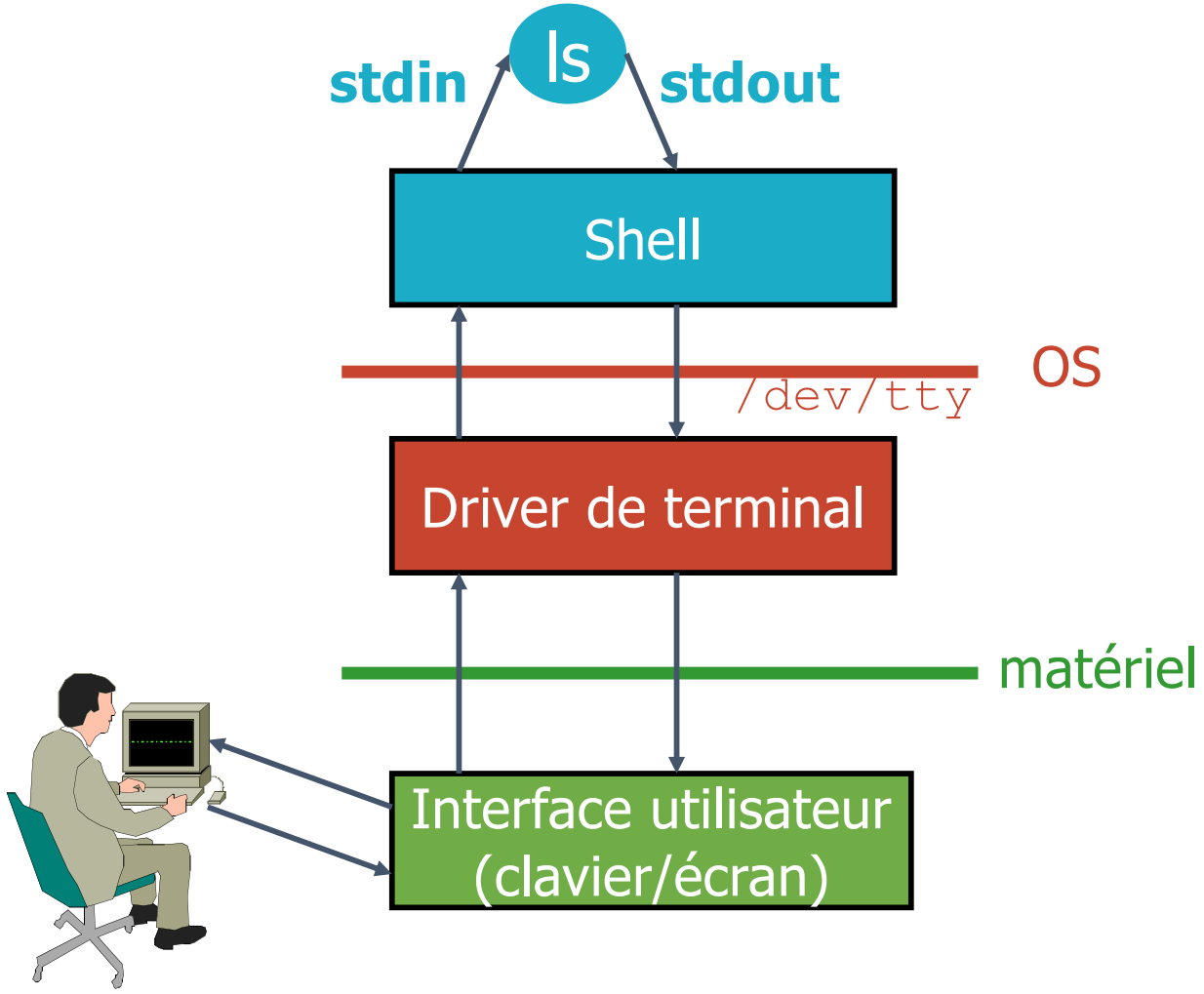
Connexion à distance :



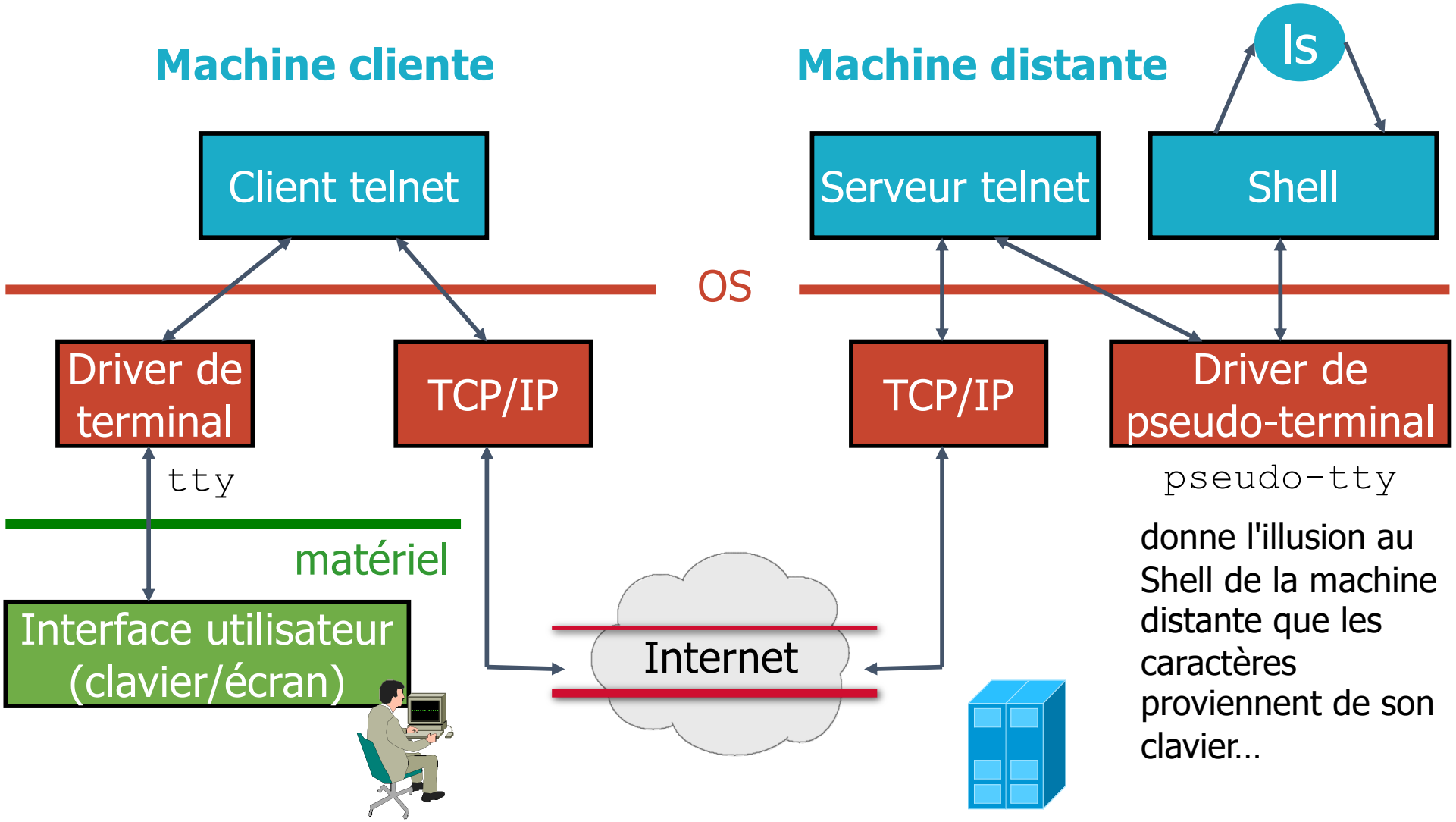
La connexion à distance

- Plusieurs protocoles
  - **`telnet`** : le standard (existe sur de nombreuses plate-formes)
  - **`rlogin`** : uniquement entre machines unix
  - **`ssh`** : sécurisé (authentification + chiffrement), peut transporter le DISPLAY c'est à dire gérer des fenêtres distantes
- La connexion à distance a besoin d'interactivité
  - Tout ce qui est tapé au clavier sur le client est envoyé au serveur à travers la connexion puis exécuté par lui.
  - Tout ce qui est envoyé par le serveur au client s'affiche dans le terminal sur l'écran de la machine cliente.

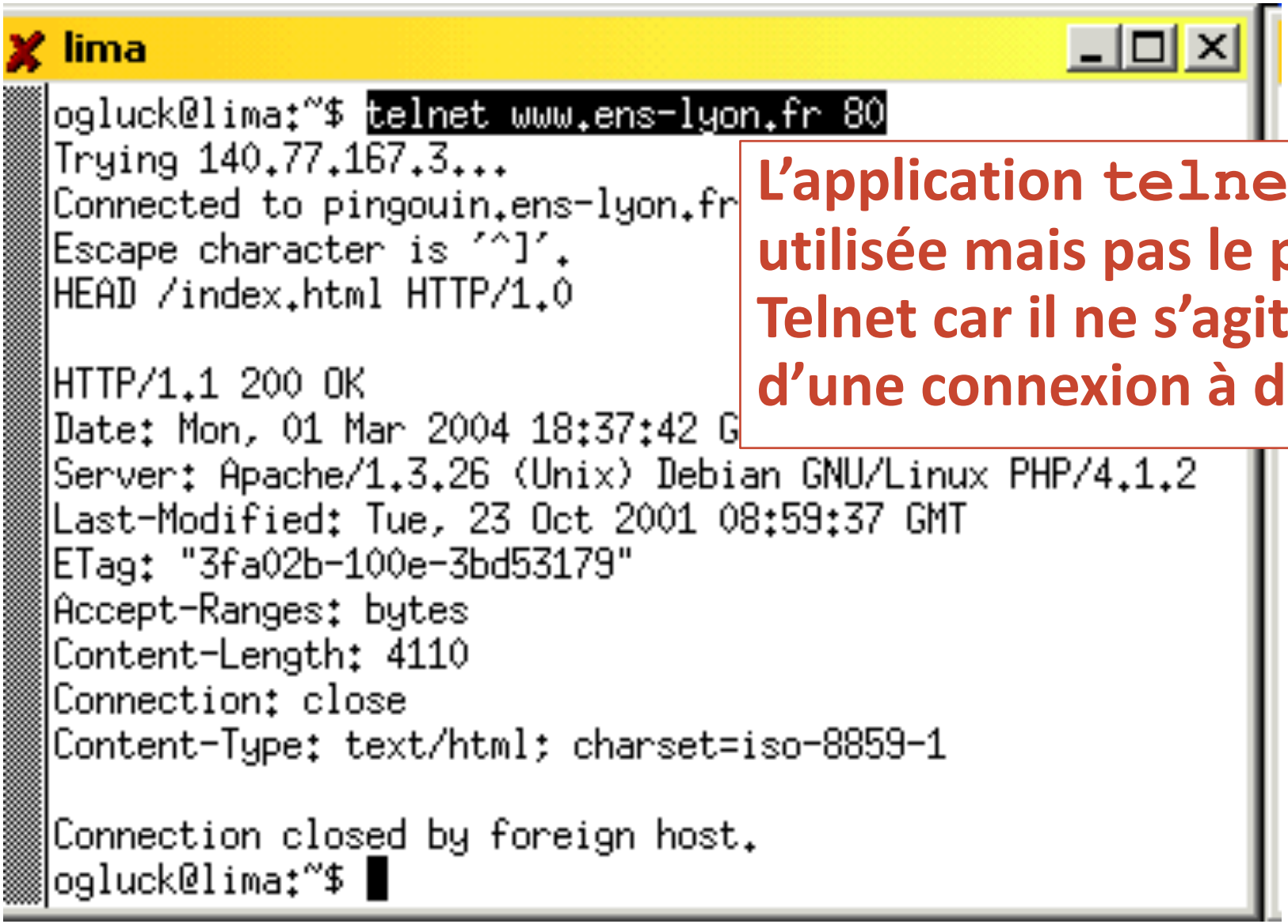
Fonctionnement d'une connexion locale



Fonctionnement d'une connexion distante



`telnet` utilisé comme client web



L'application `telnet` est utilisée mais pas le protocole Telnet car il ne s'agit pas d'une connexion à distance

SSH : un shell distant sécurisé

Secure SHell

- Les communications sont cryptées
- Authentification à base de clés
- Un des seuls protocoles de connexion à distance qui passe les pare-feux de nos jours
- Permet de transporter des fenêtres graphiques via le tunnel SSH avec `ssh -X`
- Le serveur attend sur son port 22 les demandes de connexion TCP qui arrivent des clients
- Pas encore de RFC (ietf-internet-draft)

Les commandes `ssh` et `scp`

- Connexions à distance

```
ssh -l user hostname
ssh user@hostname
```

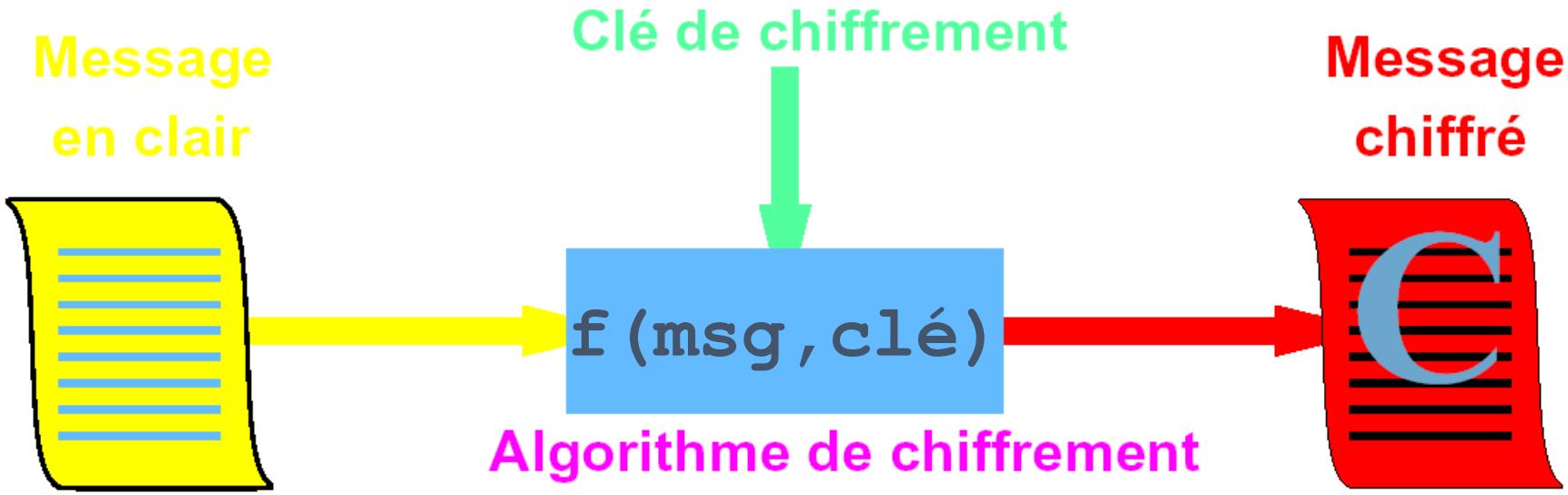
- Exécution de commande à distance

```
ssh -l user hostname cmd
ssh user@hostname cmd
```

- Copie de fichiers à distance

```
scp file1 file2 user@hostname:
scp -r dir user@hostname:/tmp
```

Principe du chiffrement



- La qualité de la sécurité dépend
  - du secret de la clé
  - de la longueur de la clé (plus il y a de bits, plus il est difficile d'essayer toutes les clés)
  - de la difficulté d'inversion de l'algorithme de chiffrement

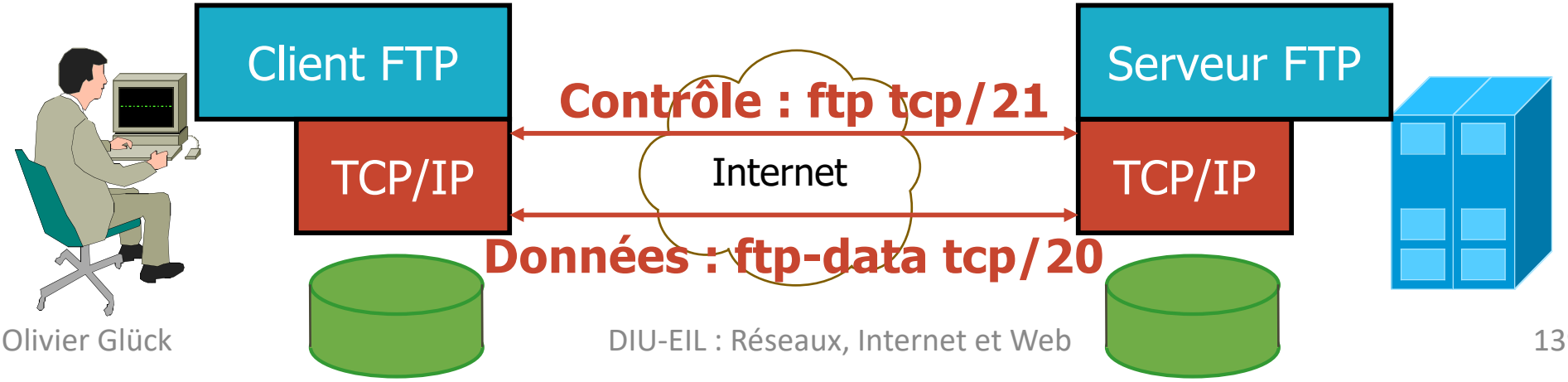
UNIVERSITÉ DE LYON
Université Claude Bernard Lyon 1
Faculté des Sciences et Technologies
Dpt Informatique

Les autres applications (FTP, NFS, LDAP…)

Le transfert de fichiers (FTP)
L'accès aux fichiers distants (NFS, SMB)
LDAP : un annuaire fédérateur

Le transfert de fichiers

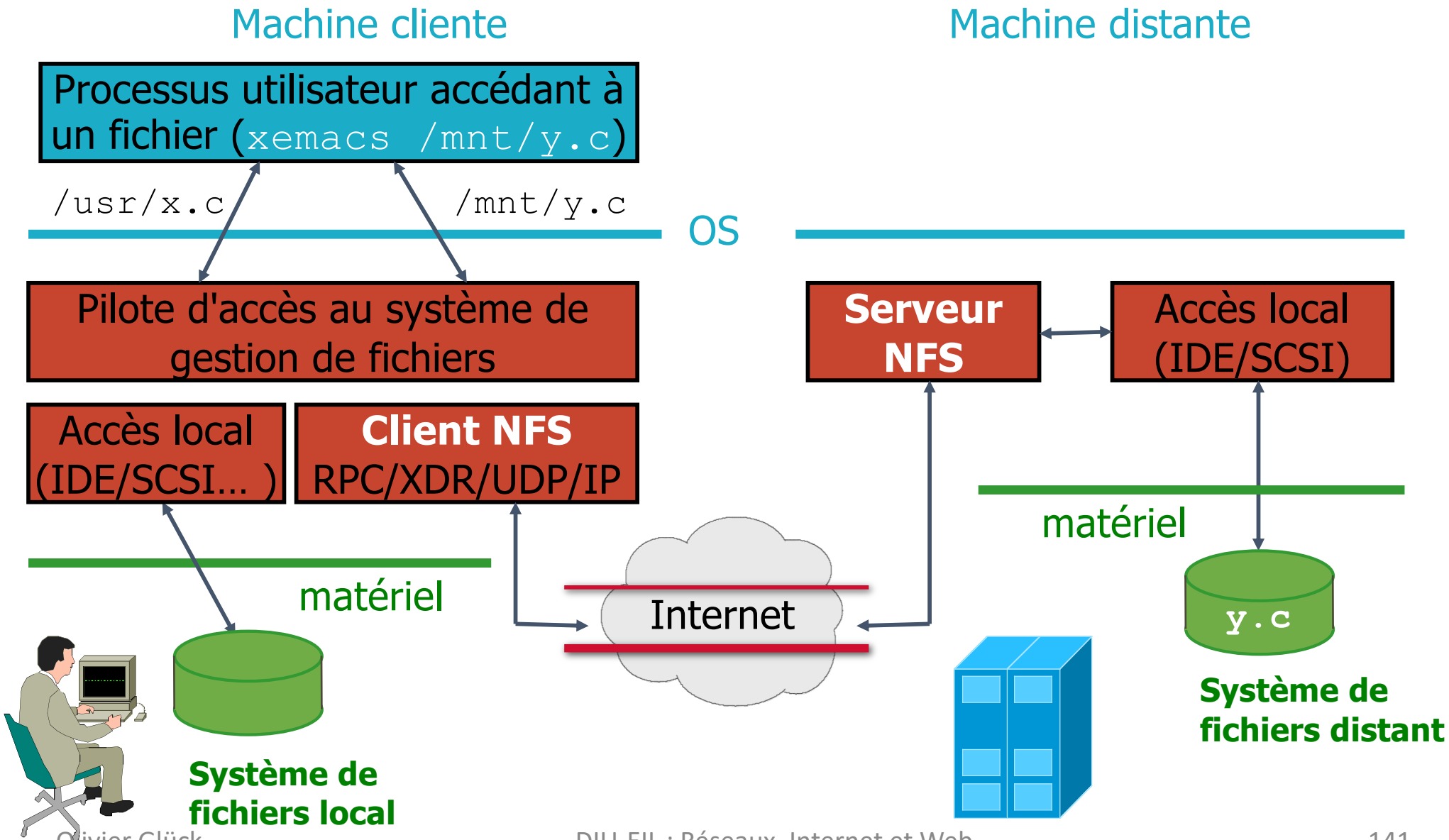
- Copie intégrale d'un fichier d'un système de fichiers vers un autre en environnement hétérogène
  - copie de fichiers à distance : **`rcp`**, **`scp`**
  - protocole de transfert de fichiers avec accès aux systèmes de fichiers local et distant : **`ftp`**, **`tftp`**, **`sftp`**
- Ne pas confondre avec les protocoles d'accès aux fichiers distants : NFS (RPC), SMB (Microsoft)
- Le serveur FTP maintient un "état" : répertoires courants local et distant, username



L’accès aux fichiers distants

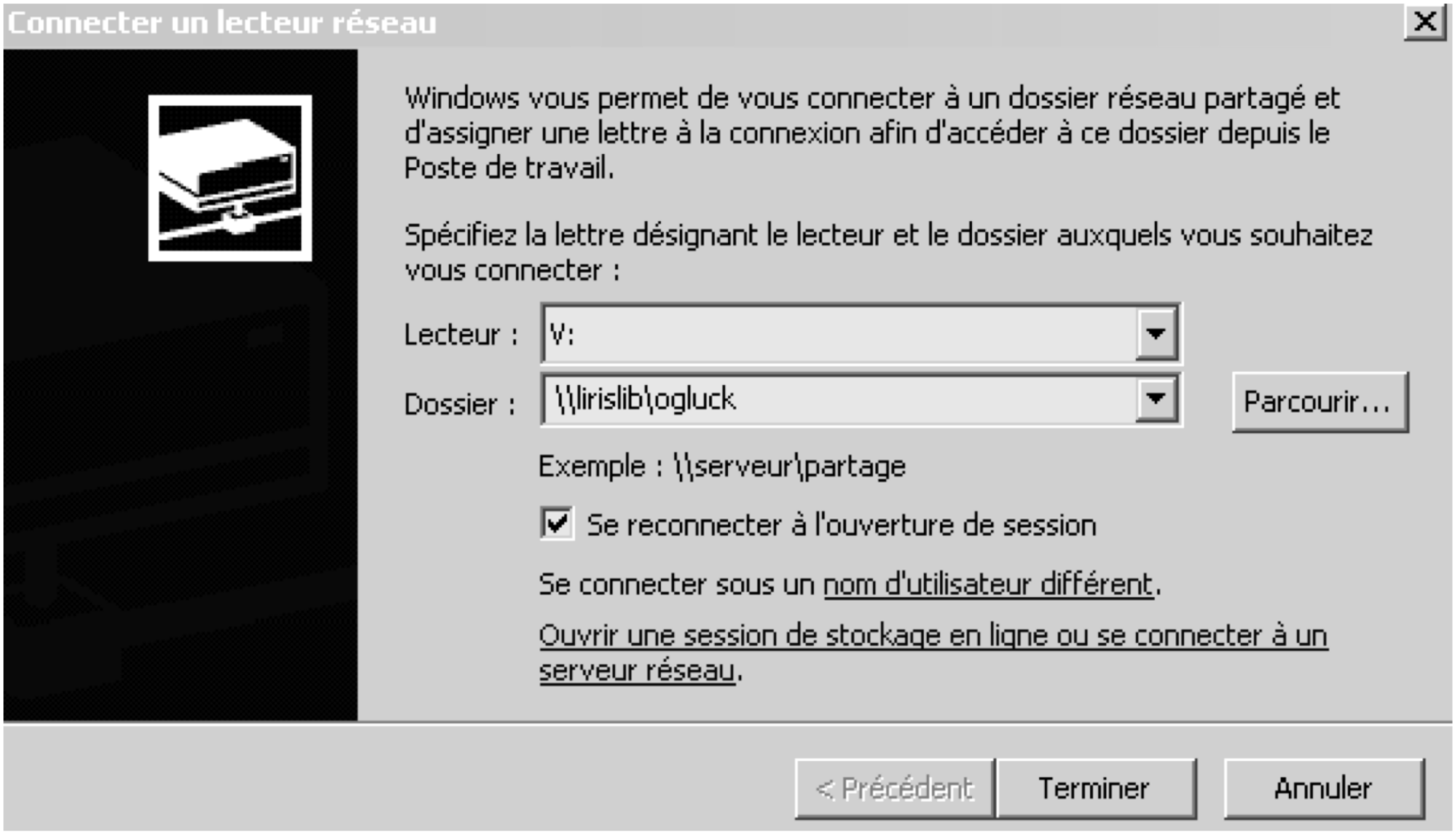
- Différences avec le transfert de fichiers
  - L'accès aux fichiers distants est complètement transparent pour l'utilisateur
  - Tout se passe comme si le système de fichiers distant était local
  - L'utilisateur peut éditer le fichier, le modifier, … ; les modifications seront répercutées sur le système de fichiers distant
- Les deux principaux protocoles
  - NFS : *Network File System* (Unix/Sun-RPC)
  - SMB : *Server Message Block* (issu du monde Microsoft)

NFS : principe de fonctionnement



SMB : Server Message Block

- Protocole de Microsoft et Intel permettant le partage de ressources (disques, imprimantes…) à travers un réseau (1987)



LDAP : un annuaire fédérateur

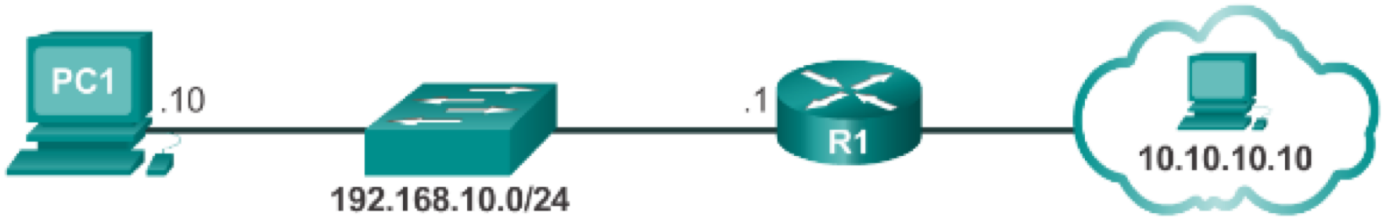
- Permettre la fusion de multiples BD dans un unique annuaire informatique
  - base Microsoft Excel du personnel administratif
  - base Microsoft Access du personnel enseignant
  - base Microsoft Excel des numéros de téléphone
  - base `/etc/passwd` des comptes Unix des utilisateurs
  - base `/etc/aliases` (ou Sympa) de listes de Mail
  - base Samba des utilisateurs Windows
  - autres bases MySQL, Oracle, maps NIS,…
- Comment envoyer un mail à l'ensemble du personnel administratif sachant que l'administrateur système recevra uniquement une liste de (Nom, Prénom) ?

UNIVERSITÉ DE LYON
Université Claude Bernard Lyon 1
Faculté des Sciences et Technologies
Dpt Informatique

Pour aller plus loin sur IP…

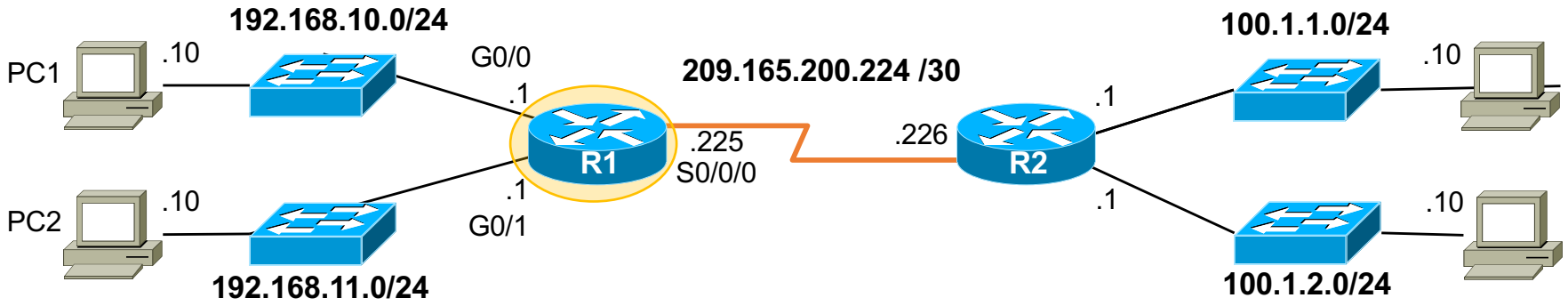
Configuration des interfaces
Les adresses privées et le NAT
Les adresses IPv6

Exemple de table de routage d'hôte IPv4



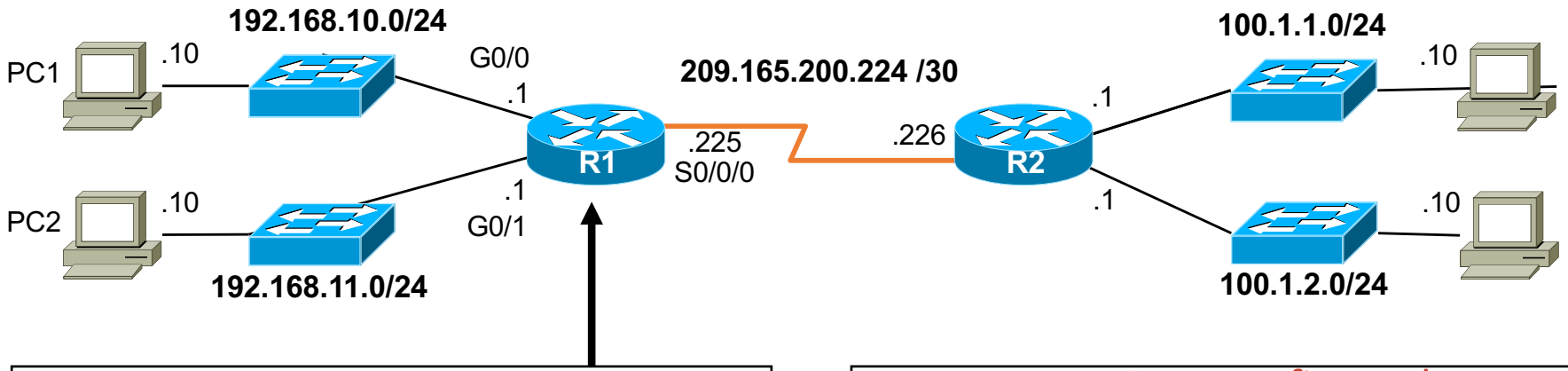

```
C:\Users\PC1> netstat -r

<Output omitted>

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface Metric
          0.0.0.0          0.0.0.0    192.168.10.1  192.168.10.10     25
        127.0.0.0        255.0.0.0         On-link       127.0.0.1    306
        127.0.0.1  255.255.255.255         On-link       127.0.0.1    306
  127.255.255.255  255.255.255.255         On-link       127.0.0.1    306
    192.168.10.0    255.255.255.0         On-link  192.168.10.10    281
   192.168.10.10  255.255.255.255         On-link  192.168.10.10    281
  192.168.10.255  255.255.255.255         On-link  192.168.10.10    281
        224.0.0.0        240.0.0.0         On-link       127.0.0.1    306
        224.0.0.0        240.0.0.0         On-link  192.168.10.10    281
  255.255.255.255  255.255.255.255         On-link       127.0.0.1    306
  255.255.255.255  255.255.255.255         On-link  192.168.10.10    281
===========================================================================

<Output omitted>
```

Table de routage d'un routeur IPv4




```
R1#show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     100.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
D       100.1.1.0/24 [90/2170112] via 209.165.200.226, 00:00:05, Serial0/0/0
D       100.1.2.0/24 [90/2170112] via 209.165.200.226, 00:00:05, Serial0/0/0
     192.168.10.0/24 is variably subnetted, 2 subnets, 3 masks
C       192.168.10.0/24 is directly connected, GigabitEthernet0/0
L       192.168.10.1/32 is directly connected, GigabitEthernet0/0
     192.168.11.0/24 is variably subnetted, 2 subnets, 3 masks
C       192.168.11.0/24 is directly connected, GigabitEthernet0/1
L       192.168.11.1/32 is directly connected, GigabitEthernet0/1
     209.165.200.0/24 is variably subnetted, 2 subnets, 3 masks
C       209.165.200.224/30 is directly connected, Serial0/0/0
L       209.165.200.225/32 is directly connected, Serial0/0/0
```

Etapes de la configuration d'un routeur




```
Router> enable
Router# configure terminal
Enter configuration commands, one per line.
Terminez par CNTL/Z.
Router(config)# hostname R1
R1(config)#
```

OU

```
Router> en
Router# conf t
Enter configuration commands, one per line.
Terminez par CNTL/Z.
Router(config)# ho R1
R2(config)#
```

Pour configurer le routeur

Donne un nom au routeur

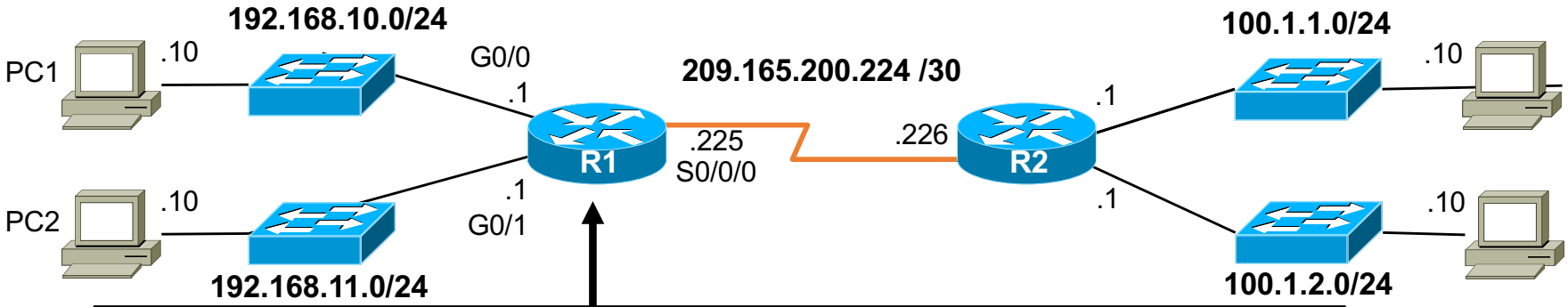
```
R1(config)# ip route 100.1.1.0/24 209.165.200.226
R1(config)# ip route 100.1.2.0/24 209.165.200.226
R1(config)#
```

Ajout des routes vers les 2 réseaux distants

```
R1# copy running-config startup-config
Destination filename [startup-config]?
Building configuration...
[OK]
```

Sauvegarde de la configuration courante pour le prochaine démarrage du routeur

Configuration des interfaces d'un routeur



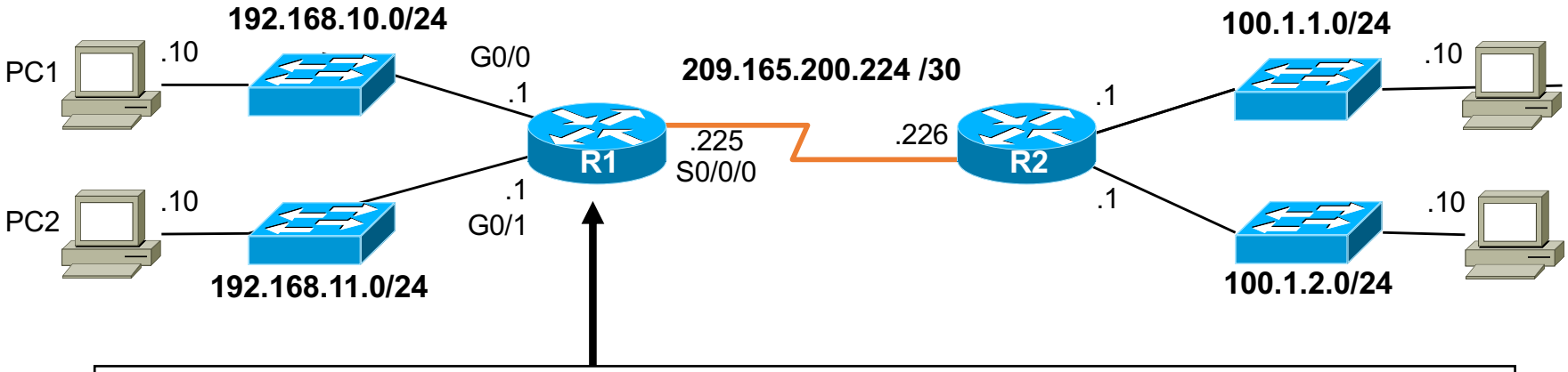

```
R1# conf t
Enter configuration commands, one per line.  Terminez par CNTL/Z.
R1(config)#
R1(config)# interface gigabitethernet 0/0
R1(config-if)# ip address 192.168.10.1 255.255.255.0
R1(config-if)# description Link to LAN-10
R1(config-if)# no shutdown
%LINK-5-CHANGED: Interface GigabitEthernet0/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/0,
changed state to up
R1(config-if)# exit
R1(config)#
R1(config)# int g0/1
R1(config-if)# ip addr 192.168.11.1 255.255.255.0
R1(config-if)# des Link to LAN-11
R1(config-if)# no shut
%LINK-5-CHANGED: Interface GigabitEthernet0/1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/1,
changed state to up
R1(config-if)# exit
R1(config)#
```

Configuration de l'interface G0/0

Configuration de l'interface G0/1

Il manque la configuration de S0/0/0

Vérification de la config. des interfaces



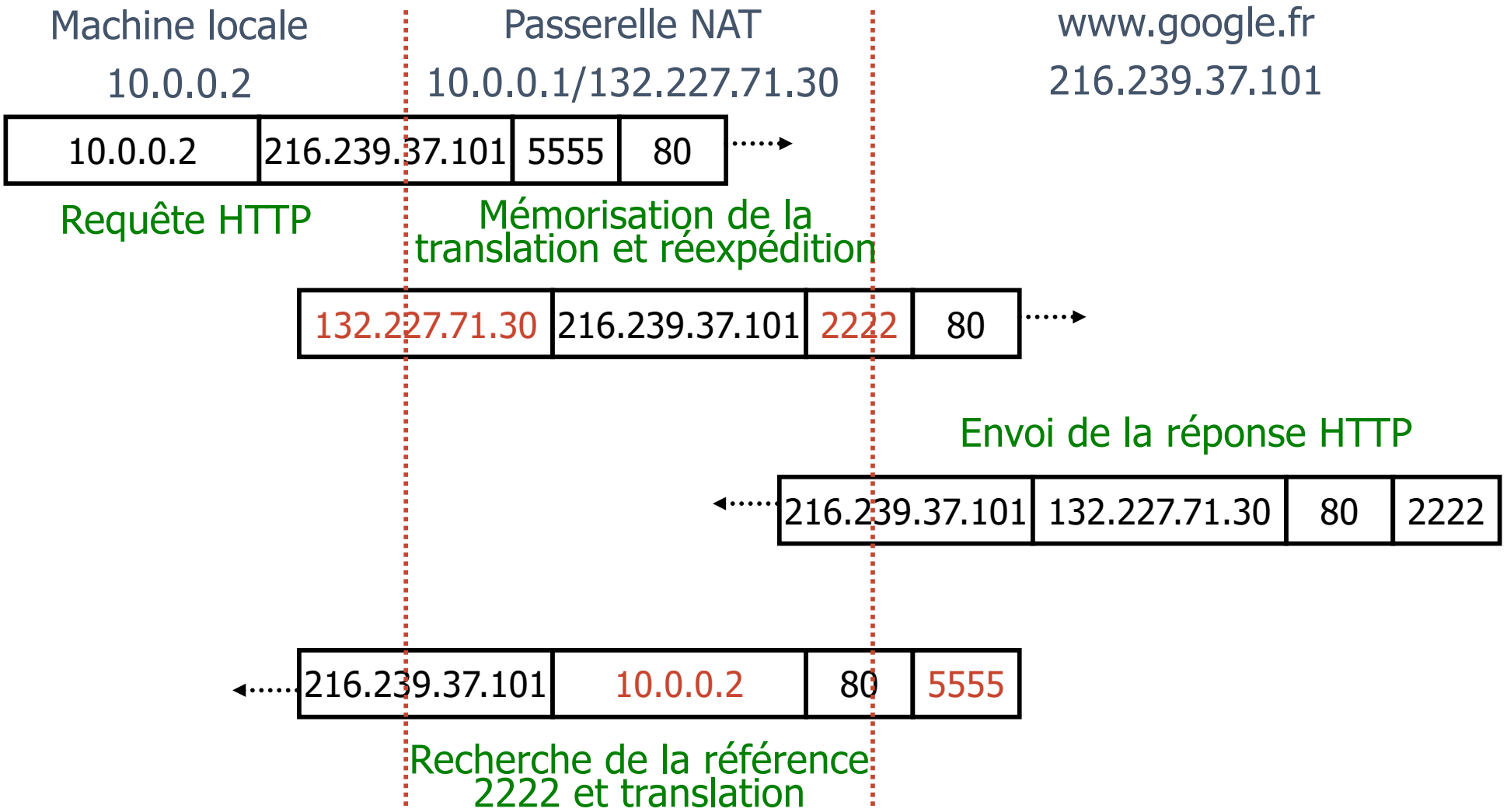

```
R1# show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol

GigabitEthernet0/0     192.168.10.1    YES manual up                    up
GigabitEthernet0/1     192.168.11.1    YES manual up                    up
Serial0/0/0            209.165.200.225 YES manual up                    up
Serial0/0/1            unassigned      YES NVRAM  administratively down down
Vlan1                  unassigned      YES NVRAM  administratively down down
R1#
R1# ping 209.165.200.226

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 209 165 200 226, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/9 ms

R1#
```

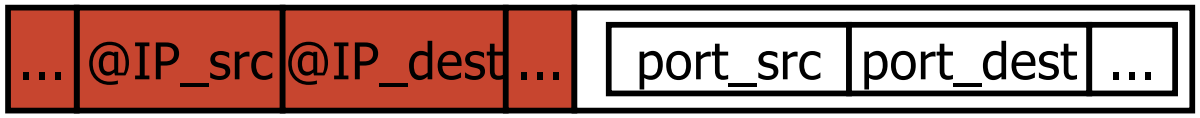
La traduction d'adresses privées (NAT)

- NAT (RFC 3022) - Network Address Translator
  - mise en correspondance d'une adresse privée et d'une adresse publique
  - traduction statique ou dynamique (lors de la connexion)
  - une solution au manque d'adresses IP publiques : quelques adresses IP publiques pour beaucoup d'adresses IP privées mais le NAT est coûteux en perf.
- Fonctionnement du NAT
  - une table stockée dans le NAT fait la correspondance entre (@IP_src privée, port_src) et une @IP_publique
  - quand le paquet part : @IP_src devient @IP_publique, port_src devient la référence de l'entrée dans la table
  - quand la réponse revient : port_dest du paquet permet de retrouver dans la table @IP et port_src

NAT - IP masquerading

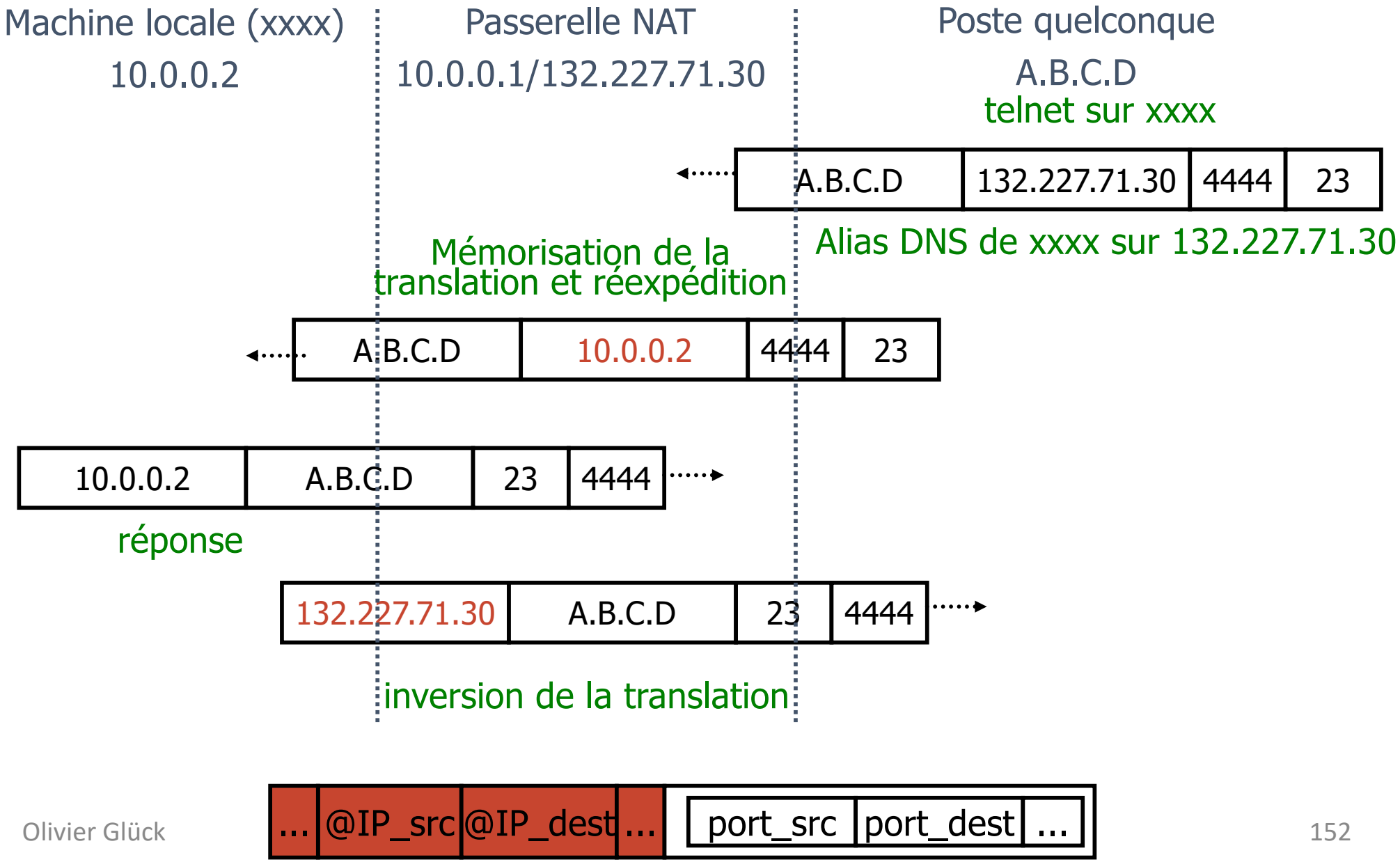
Exemple de requête sortante





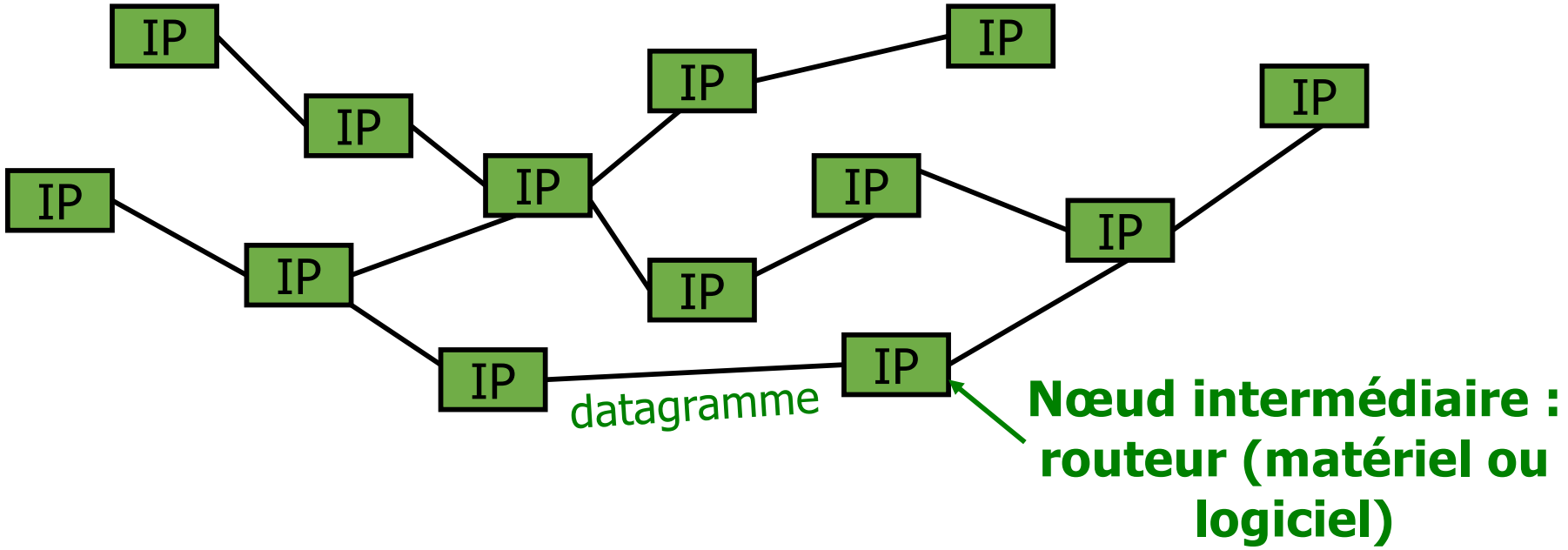
NAT - port forwarding

Exemple de requête entrante



Caractéristiques du protocole IP

Couche réseau : communications entre machines



- IP - protocole d'interconnexion, best-effort
  - acheminement de **datagrammes** (mode **non connecté**)
  - peu de fonctionnalités, pas de garanties
  - simple mais robuste (à la défaillance d'un nœud intermédiaire)

Pourquoi l'IPv6 ?

- Espace d'adressage plus important
- Amélioration du traitement des paquets
- Élimination du besoin d'adresses réseau (NAT)

- 4 milliards d'adresses IPv4
  4 000 000 000
- 340 undécillions d'adresses IPv6
  340 000 000 000 000 000 000 000 000 000 000 000 000

Les adresses IPv6

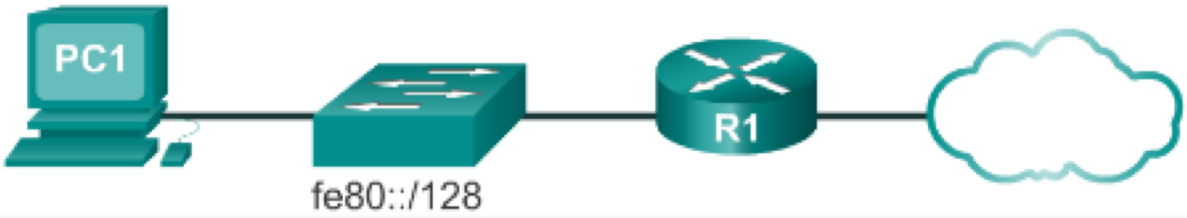
- Comportent 128 bits, sous la forme d'une chaîne de valeurs hexadécimales
- Dans l'adressage IPv6, 4 bits représentent un seul chiffre hexadécimal, 32 valeurs hexadécimales = adresse IPv6

```
2001:0DB8:0000:1111:0000:0000:0000:0200
FE80:0000:0000:0000:0123:4567:89AB:CDEF
```

- Un hextet fait référence à un segment de 16 bits ou quatre hexadécimales
- Peuvent être écrites en minuscules ou en majuscules

Exemple de table de routage d'hôte IPv6

fe80::2c30:3071:e718:a926/128
2001:db8:9d38:953c:2c30:3071:e718:a926/128




```
C:\Users\PC1> netstat -r

<Output omitted>

IPv6 Route Table
===========================================================
Active Routes:
 If Metric Network Destination      Gateway
 16     58 ::/0                     On-link
  1    306 ::1/128                  On-link
 16     58 2001::/32                On-link
 16    306 2001:0:9d38:953c:2c30:3071:e718:a926/128
                                    On-link
 15    281 fe80::/64                On-link
 16    306 fe80::/64                On-link
 16    306 fe80::2c30:3071:e718:a926/128
                                    On-link
 15    281 fe80::b1ee:c4ae:a117:271f/128
                                    On-link
  1    306 ff00::/8                 On-link
 16    306 ff00::/8                 On-link
 15    281 ff00::/8                 On-link
```

Pour aller plus loin sur HTTP…

Les types MIME

- MIME : *Multi-purpose Internet Mail Extensions*
- Permet l'échange de fichiers multimédias entre machines quelconques en spécifiant le type du fichier
- Les commandes MIME ont été intégrées dans HTTP1.0
- Un type MIME est composé
  - d'un type général (text, image, audio, video, application…)
  - et d'un sous-type (image/gif, image/jpeg, application/pdf, application/rtf, text/plain, text/html)
- En perpétuelle évolution
- La machine cliente doit ensuite associer l'exécution d'une application à chaque type MIME
- Le serveur positionne Content-type à partir de l'extension du document demandé (`/etc/mime.types`)

Format URL encodé

- Nécessité de coder les données de l'URL (méthode GET) sur le client pour construire la chaîne CGI pour respecter la RFC 2396 qui spécifie la syntaxe des URL
- Les caractères non-alphanumériques sont remplacés par %xx (xx=code ASCII du caractère en hexadécimal)
- Les caractères `; / ? : @ & = + $` et `,` sont réservés pour une signification particulière dans l'URL
  - `?` : début de QUERY_STRING
  - `&` : séparateur de champ
  - `=` : séparation entre le nom du champ et sa valeur
- Les espaces sont remplacés par des `+`

Exemple 1-index.html

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8" />
        <title>
                1-index.html
        </title>
        <!-- style.css n'existe pas -->
        <link rel="stylesheet" type="text/css" href="style.css"/>
</head>
<body>
        Bonjour Olivier !
</body>
</html>
```

Regarder avec Firebug « Réseaux »

- le 404 Not found pour style.css
- le format des deux requêtes/réponses HTTP

```
$ telnet localhost 80
GET /Sites/Exemples/CM4-HTTP/1-index.html HTTP/1.0
```

Exemple 2-index.php

```
<!DOCTYPE html>
<html> <head>
        <meta charset="utf-8" />
        <title> 2-index.php </title>
</head>
<body>
        <?php
                $url = $_SERVER['REQUEST_URI'];
                echo $url;
        ?>
        <br />
        <?php
                $tab = explode("/",$url);
                echo $tab[1];
        ?>
</body>
</html>
```

Regarder la réponse avec Firebug « Réseaux » pour voir que le code PHP a disparu

```
$ telnet localhost 80
GET /Sites/Exemples/CM4-HTTP/2-index.php HTTP/1.0
```

Exemple 3-compteur-get.php

```
<!DOCTYPE html>
<html> <head> <meta charset="utf-8" />
        <title>3-compteur-get.php </title>
</head>
<body> <form name="compteur" method="get" action="3-compteur-get.php">
        <?php
                if (isset($_GET['compteur']))
                        $cpt = $_GET['compteur'];
                else
                        $cpt = 0;
                $cpt = $cpt + 1;
                // Pour propager la nouvelle valeur de cpt vers la page suivante
                echo '<input type="hidden" name="compteur" value="'.$cpt.'"/>';
                echo "Le compteur vaut $cpt.\n";
        ?>
        <input type="submit" name="action" value="+1" />
</form></body>
</html>
```

Regarder la réponse avec Firebug
Changer la valeur dans l'URL

```
$ telnet localhost 80
GET /Sites/Exemples/CM4-HTTP/3-compteur-get.php?compteur=3&action=%2B1 HTTP/1.0
```

Exemple 3-compteur-post.php

```
<!DOCTYPE html>
<html> <head> <meta charset="utf-8" />
        <title>3-compteur-get.php </title>
</head>
<body> <form name="compteur" method=”post" action="3-compteur-post.php">
        <?php
                if (isset($_POST['compteur']))
                        $cpt = $_POST['compteur'];
                else
                        $cpt = 0;
                $cpt = $cpt + 1;

        ?>
        <input t
</form></body>
</html>
```

Regarder avec Firebug les en-têtes et la réponse après +1

```
$ telnet localhost 80
POST /Sites/Exemples/CM4-HTTP/3-compteur-post.php HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 22

compteur=3&action=%2B1
```

Pour afficher tous les champs du formulaire

```
// afficher la chaîne CGI en méthode GET
echo $_SERVER['QUERY_STRING'];
echo "<br />";
// afficher tous les champs transmis en méthode GET
foreach ($_GET as $name => $value) {
        echo "$name = $value<br />\n";
}
// afficher tous les champs transmis en méthode POST
foreach ($_POST as $name => $value) {
        echo "$name = $value<br />\n";
}
```

Voir 4-form.php

Installation d'un cookie sur le client

- Directive Set-Cookie **dans l'en-tête de la réponse HTTP** (envoyé lors de la première connexion)

Set-Cookie: nom=valeur; expires=date; path=chemin_accès; domain=nom_domaine; secure

  - le couple nom/valeur est le contenu du cookie (seul champ obligatoire), sans espace ; et ,
  - le cookie devient invalide après la date indiquée
  - `path=/pub` signifie que le cookie est valable pour toutes les requêtes dont l'URL contient `/pub`
  - `domain` indique le nom de domaine (associé au serveur) pour lequel le cookie est valable
  - `secure` : le cookie n'est valable que lors d'une connexion sécurisée

Utilisation d'un cookie par le client

- Chaque fois qu'un client va effectuer une requête, il vérifie dans sa liste de *cookies* s'il y en a un qui est associé à cette requête
- Si c'est le cas, le client utilise la directive Cookie **dans l'en-tête de la requête HTTP**

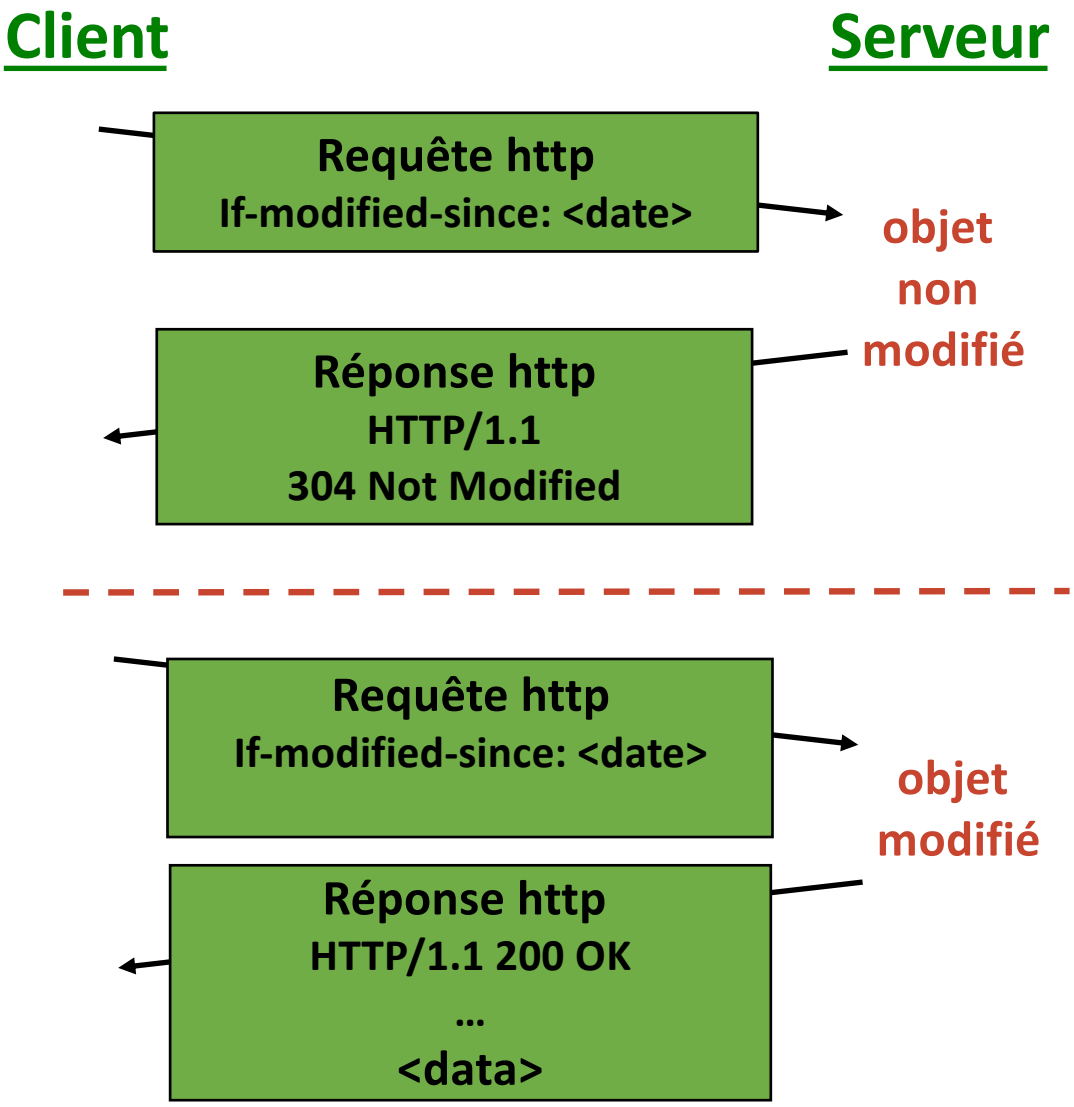
Cookie: nom1=valeur1; nom2=valeur2; …

- Le serveur peut insérer plusieurs directives Set-Cookie
- Dans la première spécification des *cookies* :
  - un client peut stocker un maximum de 300 *cookies*
  - un maximum de 20 *cookies* par domaine est permis
  - la taille d'un *cookie* est limitée à 4Ko

L'en-tête If-Modified-Since

- Objectif : ne pas envoyer un objet que le client a déjà dans son cache
- Problème : les objets contenus dans le cache peuvent être obsolètes
- Le client spécifie la date de la copie cachée dans la requête http

  If-modified-since: <date>
- La réponse du serveur est vide si la copie du cache est à jour



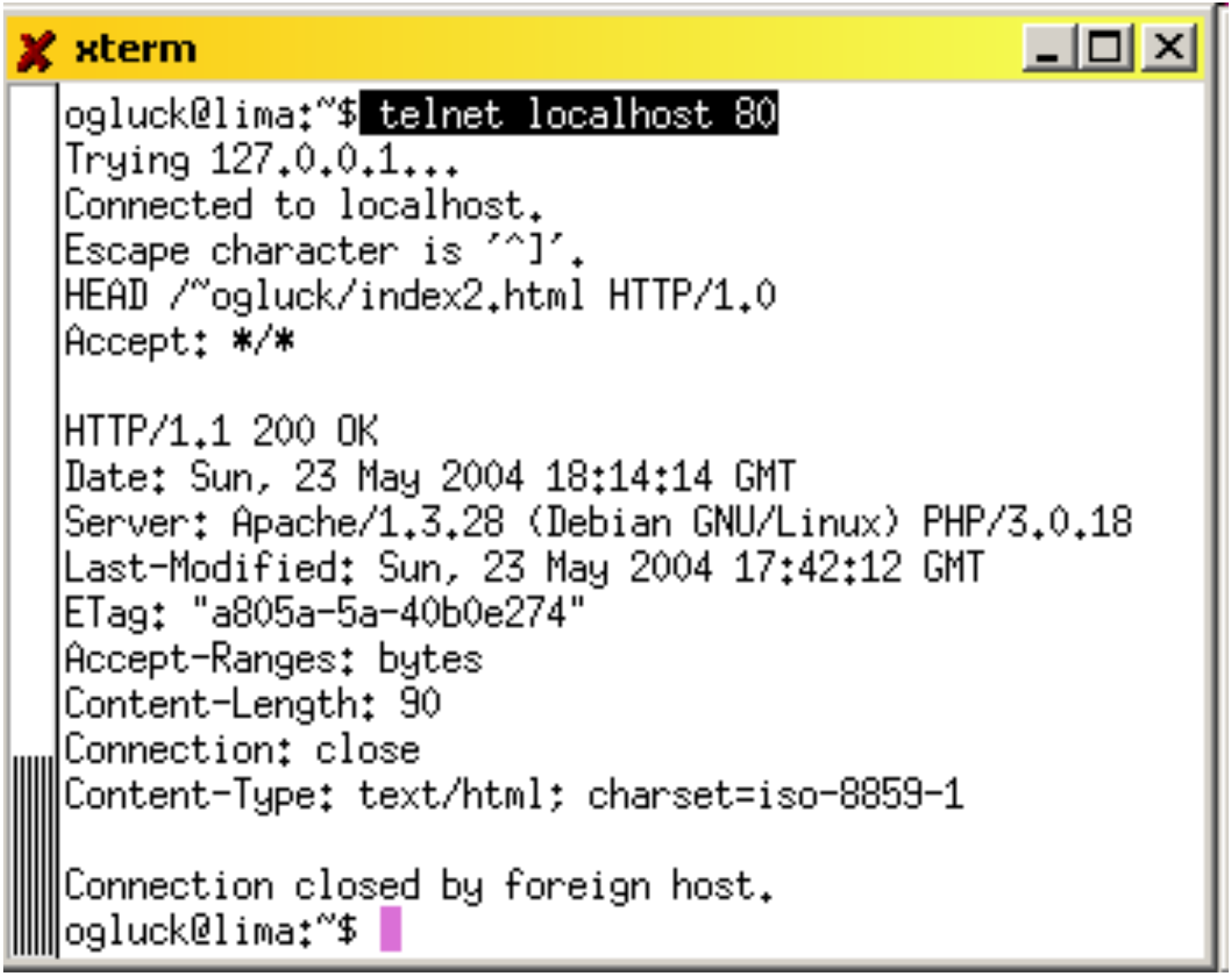
Les requêtes HTTP faites par le client

- Format d'une requête HTTP (du client vers le serveur)
  - une commande HTTP (METHOD), une URL qui identifie la ressource demandée, la version de HTTP
  - l'en-tête et une ligne vide
  - éventuellement un contenu (corps de la requête)
- Méthode GET
- Méthode POST
- Méthode HEAD
- D'autres méthodes qui ne sont pas souvent supportées par les serveurs

La méthode HEAD (1)

- Identique à GET mais permet uniquement de récupérer les en-têtes relatifs à un document
- Permet de récupérer
  - la date de dernière modification du document (important pour les caches, JavaScript)
  - la taille du document (estimation du temps d'arrivée du document)
  - le type du document (le client peut sélectionner le type de documents qu'il accepte)
  - le type du serveur (permet de faire des requêtes spécifiques selon le type du serveur)
- Remarque : le serveur ne fournit pas nécessairement toutes ces informations !

La méthode HEAD (2)



Autres requêtes HTTP

- PUT : permet de stocker le corps de la requête sur le serveur à l'URL spécifiée
- DELETE : suppression du document spécifié par l'URL
- OPTIONS : renvoie la liste des méthodes autorisées par le serveur
- TRACE : la corps de la requête entrante est renvoyée au client (utilisé pour faire du débug)
- …

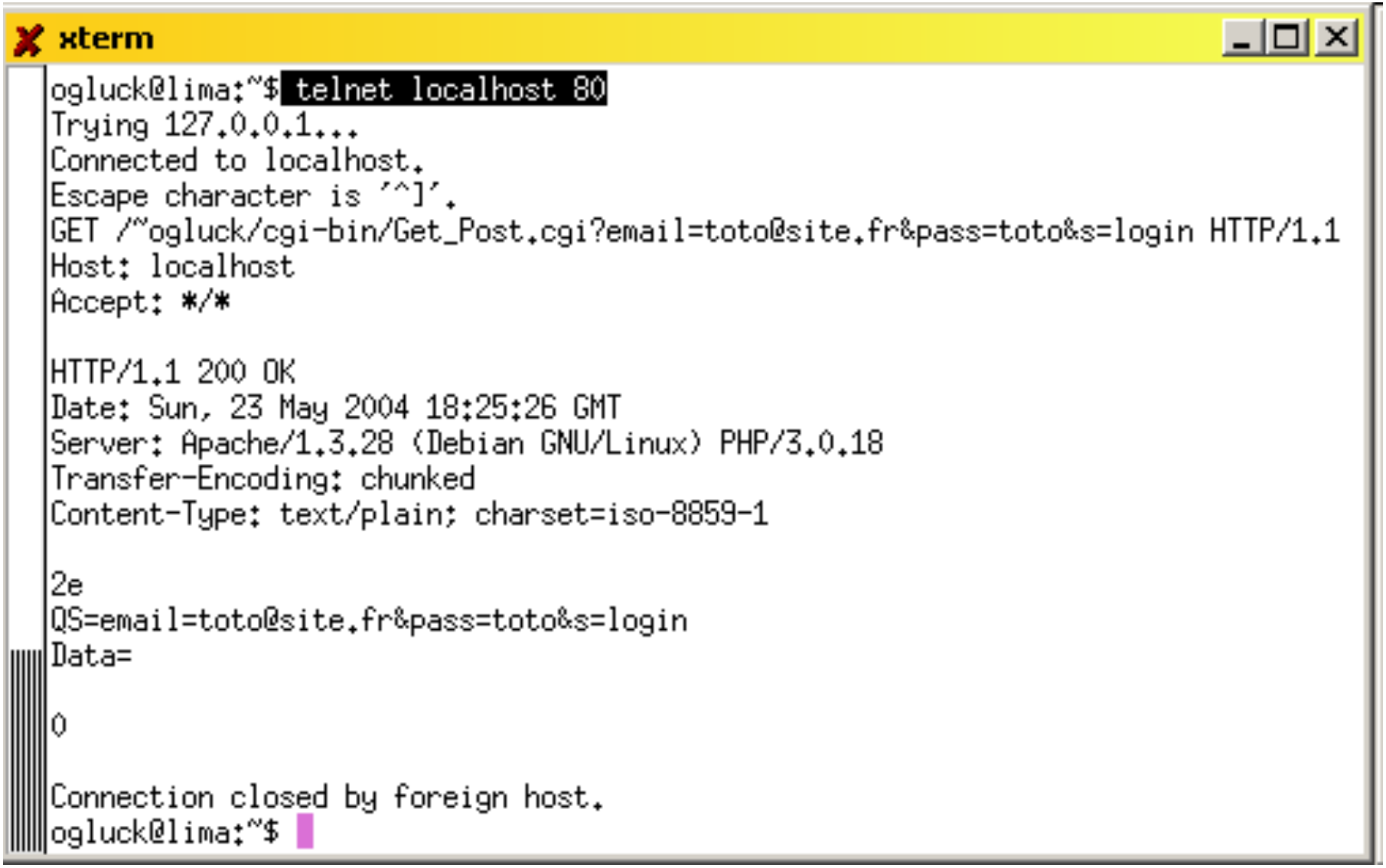
Méthodes GET/POST (1)

- Voici le code d'un petit script CGI en shell

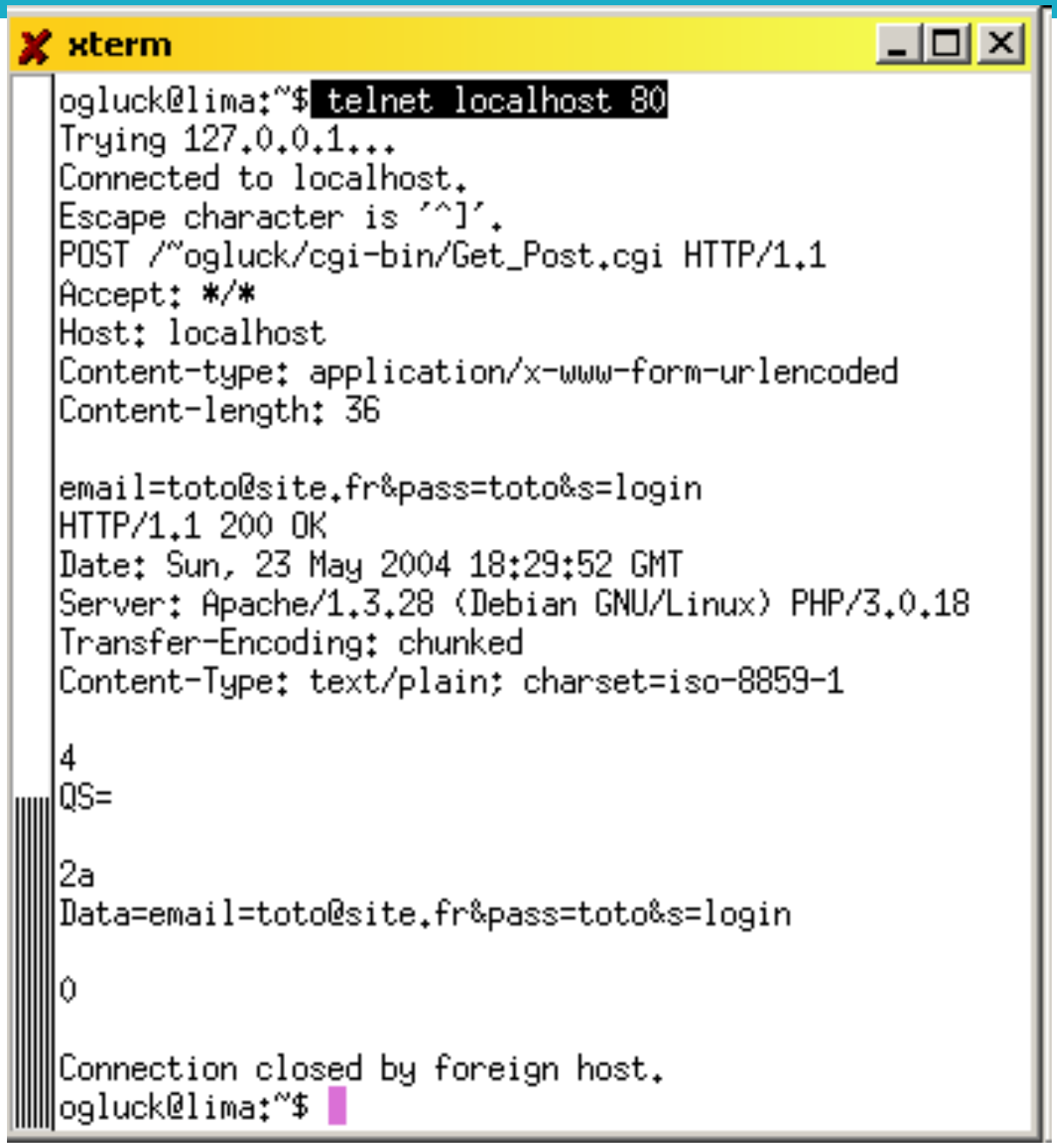
```
#! /bin/sh
# Get_Post.cgi
echo 'Content-type: text/plain'
echo ' '
echo "QS=$QUERY_STRING"
read DATA
echo "Data=$DATA"
```

- Les résultats de l'exécution avec la méthode GET puis POST sont montrés dans les deux transparents suivants

Méthodes GET/POST (2)



Méthodes GET/POST (3)

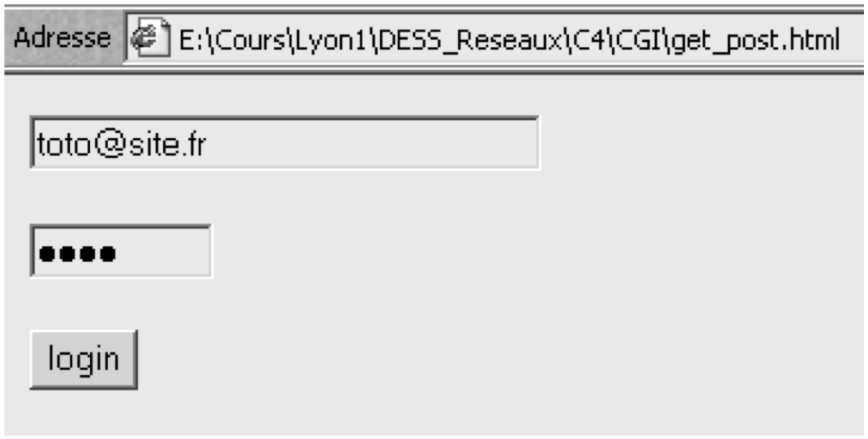


Méthodes GET/POST (4)

- Avec la méthode GET
  - les données relatives aux champs du formulaire sont transmises via l'URL (dans le type de la requête)
  - le programme CGI les récupère dans la variable d'environnement **`QUERY_STRING`**
  - il est possible de cliquer sur "Actualiser" pour retransmettre les données et de définir un *bookmark*
- Avec la méthode POST
  - les données relatives aux champs du formulaire sont transmises dans le corps de la requête HTTP
  - `Content-type` et `Content-length` sont positionnés
  - le programme CGI les récupère sur l'entrée standard
  - "Actualiser" et *bookmark* impossibles, données du formulaire non visibles dans les logs du serveur

Méthodes GET/POST (5)

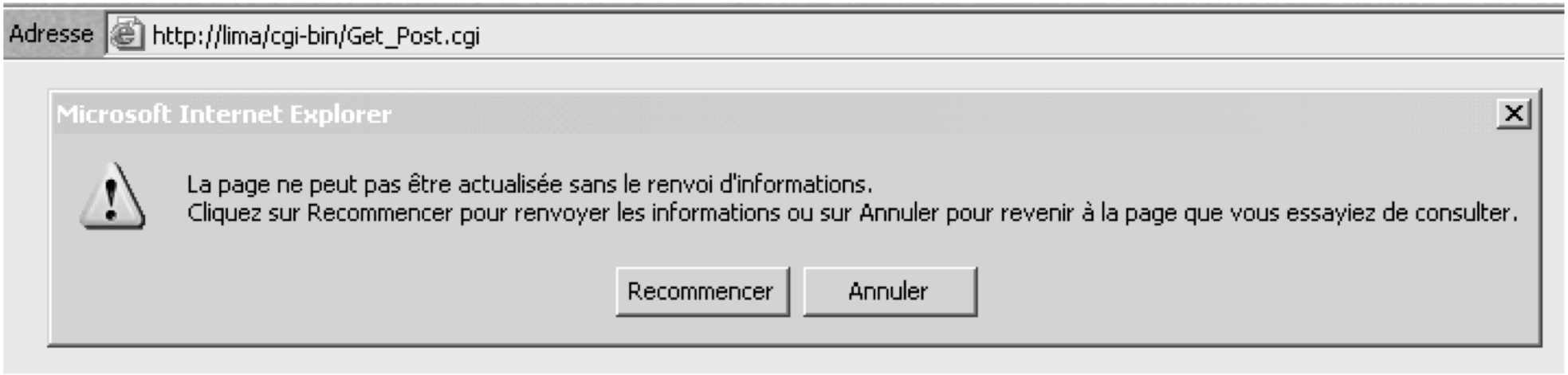
Formulaire



Méthode GET

Méthode POST

Méthode POST et "Actualiser"



Les variables d'environnement (1)

- Elles sont positionnées par le serveur HTTP pour fournir au CGI des infos sur le serveur, le client, …

SERVER_SOFTWARE : nom/version
nom et version du démon HTTP

SERVER_NAME : nom
nom ou adresse IP de la machine serveur HTTP

GATEWAY_INTERFACE : CGI/version (CGI/1.1)
version des spécifications CGI utilisées par le serveur

SERVER_PROTOCOL : protocole/version (HTTP/1.1)
protocole et version de la requête en cours de traitement

SERVER_PORT : port
numéro du port (TCP) vers lequel la requête a été envoyée

Les variables d'environnement (2)

REQUEST_METHOD : method (GET/POST/…)
méthode associée à la requête en cours de traitement

SCRIPT_NAME : nom (/cgi-bin/mon_cgi.cgi)
chemin du CGI à partir de la racine du serveur HTTP

REMOTE_HOST : nom
nom de la machine d'où vient la requête

REMOTE_ADDR : adresse_IP
adresse IP de la machine d'où vient la requête

AUTH_TYPE : authentification
méthode d'authentification de l'utilisateur s'il y a lieu

REMOTE_USER : login
si authentification, nom de l'utilisateur associé à la requête

REMOTE_IDENT : login_os
login de connexion de l'utilisateur (pas souvent supporté)

Les variables d'environnement (3)

CONTENT_TYPE : type/subtype (application/x-www-form-urlencoded)

type MIME des données véhiculées dans la requête

CONTENT_LENGTH : lg (en octets)

longueur des données véhiculées dans la requête (POST)

PATH_INFO : path

chaîne entre SCRIPT_PATH et QUERY_STRING dans l'URL

QUERY_STRING : nom1=val1&nom2=val2…

données transmises au CGI via l'URL (GET)

pour afficher la chaîne : echo $_SERVER['QUERY_STRING'];

HTTP_XXX (une variable pour chaque champ contenu dans l'en-tête HTTP de la requête)

HTTP_ACCEPT, HTTP_USER_AGENT, ...

Adresse http://lima/cgi-bin/Env.cgi?email=entrez+votre+email+ici&pass=toto&s=login

DOCUMENT_ROOT --> /var/www
GATEWAY_INTERFACE --> CGI/1.1
HTTP_ACCEPT --> */*
HTTP_ACCEPT_ENCODING --> gzip, deflate
HTTP_ACCEPT_LANGUAGE --> fr
HTTP_CONNECTION --> Keep-Alive
HTTP_HOST --> lima
HTTP_USER_AGENT --> Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
PATH --> /bin:/usr/bin:/sbin:/usr/sbin
QUERY_STRING --> email=entrez+votre+email+ici&pass=toto&s=login
REMOTE_ADDR --> 140.77.13.102
REMOTE_PORT --> 3304
REQUEST_METHOD --> GET
REQUEST_URI --> /cgi-bin/Env.cgi?email=entrez+votre+email+ici&pass=toto&s=login
SCRIPT_FILENAME --> /home/ogluck/public_html/cgi-bin/Env.cgi
SCRIPT_NAME --> /cgi-bin/Env.cgi
SERVER_ADDR --> 140.77.13.131
SERVER_ADMIN --> olivier.gluck@ens-lyon.fr
SERVER_NAME --> lima
SERVER_PORT --> 80
SERVER_PROTOCOL --> HTTP/1.1
SERVER_SIGNATURE -->
Apache/1.3.28 Server at lima Port 80

SERVER_SOFTWARE --> Apache/1.3.28 (Debian GNU/Linux) PHP/3.0.18
UNIQUE_ID --> P7J-K4xNDYMAAAMwBLI

Pour aller plus loin sur le chiffrement...

SSH : chiffrement

- Deux types d'algorithmes
  - symétrique : même clé privée secrète partagée utilisée pour le chiffrement et le déchiffrement
    - l'émetteur et le récepteur doivent se mettre d'accord sur **la** clé à utiliser
  - asymétrique : utilisation d'une clé publique pour le chiffrement et d'une clé privée pour le déchiffrement

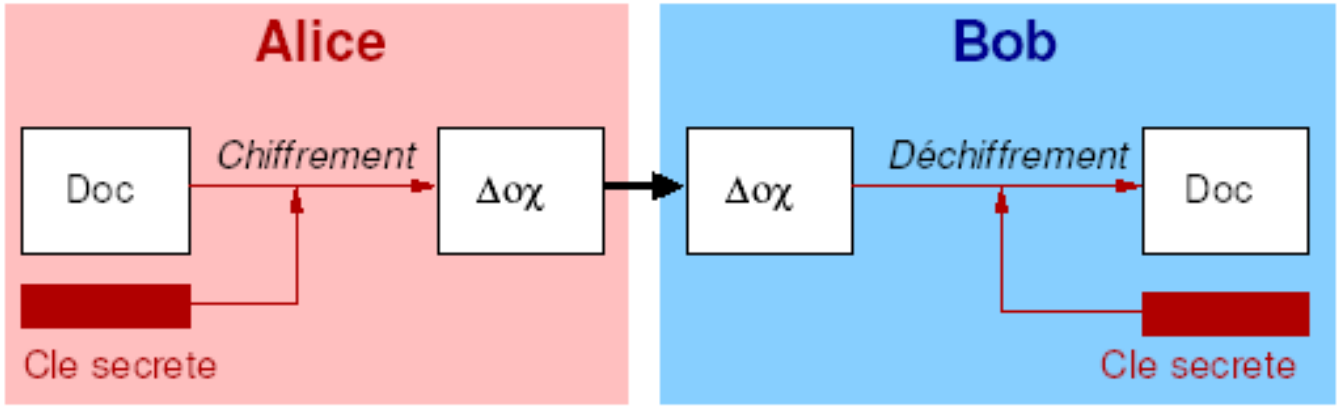
$m_{chiffré} = f(m_{clair}, c_{publique})$ et $m_{clair} = g(m_{chiffré}, c_{privée})$

  - pour qu'un émetteur envoie un message chiffré, il suffit qu'il connaisse **la** clé publique du destinataire
  - pb : comment être sûr que la clé publique est bien celle du destinataire escompté ?
  - certificat : association d'une clé publique et d'un nom de destinataire signée par un tiers de confiance

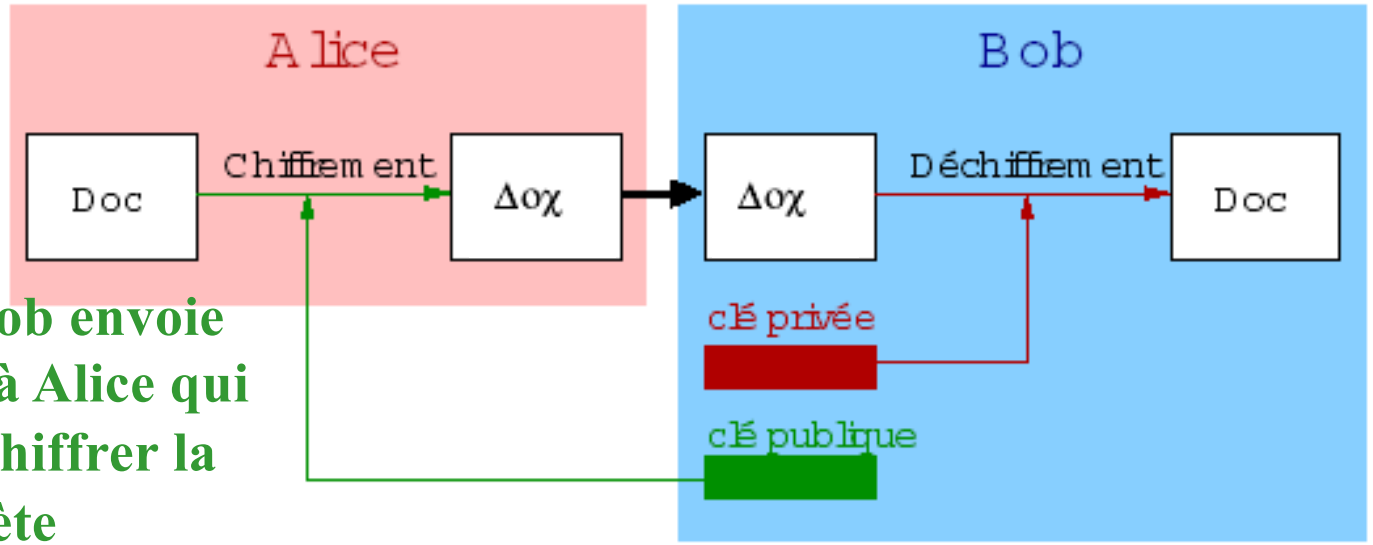
Chiffrement symétrique et asymétrique

source : M. Herrb

Symétrique : DES / AES



Assymétrique : RSA (DSA - El-Gamal)



Le client ssh Bob envoie sa clé publique à Alice qui l'utilise pour chiffrer la clé secrète

SSH : chiffrement

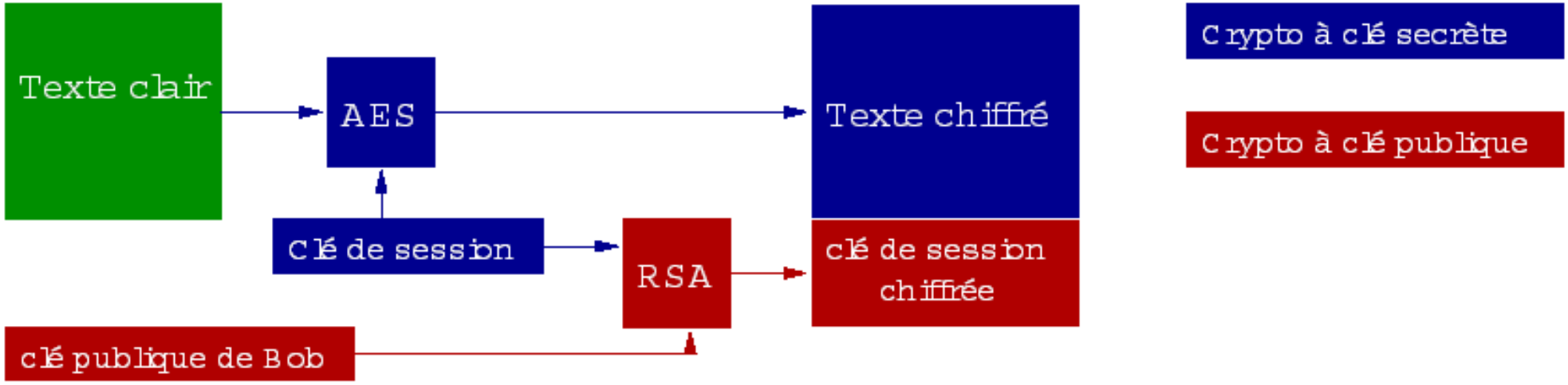
- Dans SSH :
  - algorithme asymétrique pour l'authentification (généralement RSA : basé sur l'arithmétique modulo)
  - algorithme symétrique pour les communications
    - utilisation de RSA pour échanger la clé de l'algorithme symétrique
    - chiffrement et déchiffrement moins coûteux

Chiffrement pratique

source : M. Herrb

Fonctions à clé publique très coûteuses → utilisation d'une **clé de session**

Chiffrement par Alice :



Longueur des clés sûres (2003) :

clé secrète : 128 bits
clé publique/privée : 1024 bits

ssh de Bob vers Alice, Bob envoie sa clé publique à Alice, Alice fabrique une clé secrète (la clé de session) pour l'envoyer chiffrée à Bob. C' est cette clé secrète qui sera utilisée ensuite pour chiffrer/déchiffrer de manière symétrique avec AES.

Déchiffrement pratique

source : M. Herrb

Déchiffrement par Bob :