Chapitre 2: la couche application

Objectifs:

- Comprendre le fonctionnement de quelques protocoles parmi les plus connus de la couche application
  - HTTP
  - FTP
  - DNS
- Conception et implémentation des protocoles des apps réseaux
  - Modèles de services de la couche transport
  - Le paradigme client-serveur

Chapitre 2: la couche application

1. Principes des apps réseaux
   1. Architectures des apps
   2. Interfaces de connexion (*Socket*)
2. Web et HTTP
3. Protocole de transfert de fichier (FTP)
4. Serveur de noms DNS

Quelques apps réseaux

- e-mail
- Accès à distance
- Web
- Commerce électronique
- messagerie instantanée
- Partage de fichiers P2P
- Voix sur IP (Skype)
- Jeux interactifs
- Diffusion de vidéos (YouTube, Netflix)
- Réseaux sociaux (Facebook, Instagram, WeChat…)
- Application basée sur la localisation (Waze,
- Informatique nuagique

(cloud computing: Amazon EC2, Google App engine, Azure…) …

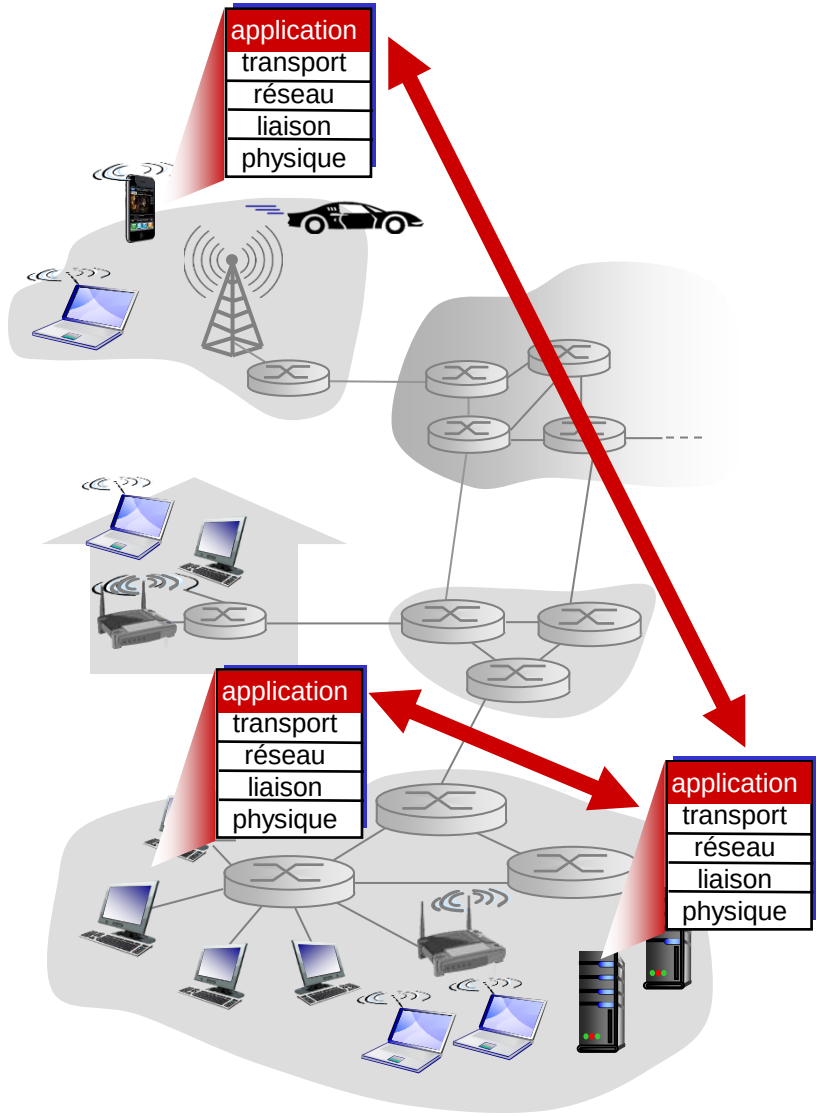
Créer une app réseau

Écrire des programmes qui

- Roulent sur des hôtes différents
- communiquent à travers le réseau
- Par exemple, un serveur web communique avec le navigateur web

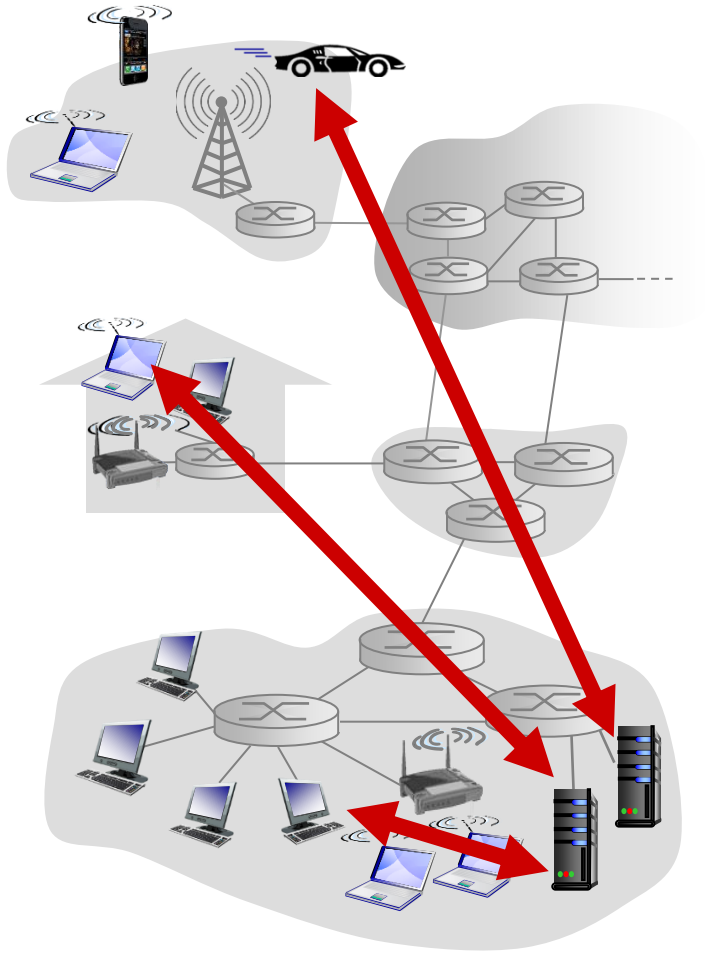
Pas besoin d'écrire du code pour les équipements du cœur du réseau

- Les équipements du cœur de réseau n'exécutent pas les applications des utilisateurs
- L'exécution des apps sur les hôtes seulement facilite leur développement et accélère leur adoption



Architecture client-serveur

client/serveur

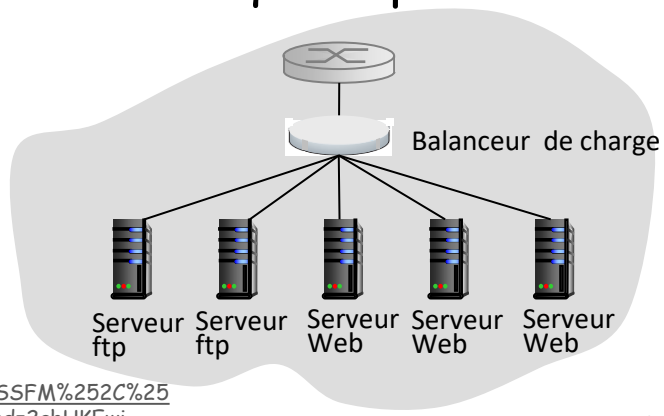


Serveur:

- Hôte toujours actif
- Adresse IP permanente
- Centre de données

Client:

- Communique avec le serveur
- Connections intermittentes
- Adresses IP peuvent être dynamiques
- Ne communiquent pas directement entre eux



https://www.google.com/search?q=server+farm&client=firefox-b-d&tbm=isch&source=iu&ictx=1&fir=SnrmCKtsMxI1DM%253A%252CgX3WNGsor3SSFM%252C%252Fm%252F01n806&vet=1&usg=AI4_-kTTqisqp9E-yuEjJrHCkEdA1mQRiw&sa=X&ved=2ahUKEwi-kc3PhsTkAhXOVN8KHeLEDE4Q_B0wE3oECAUQAw#imgrc=SnrmCKtsMxI1DM:

Chapitre 2: la couche application

1. Principes des apps réseaux
   1. Architectures des apps
   2. Interfaces de connexion (*Socket*)
   3. Exigences des apps en termes de performance et de fiabilité
2. Web et HTTP
3. Protocole de transfert de fichier (FTP)
4. Serveur de noms DNS

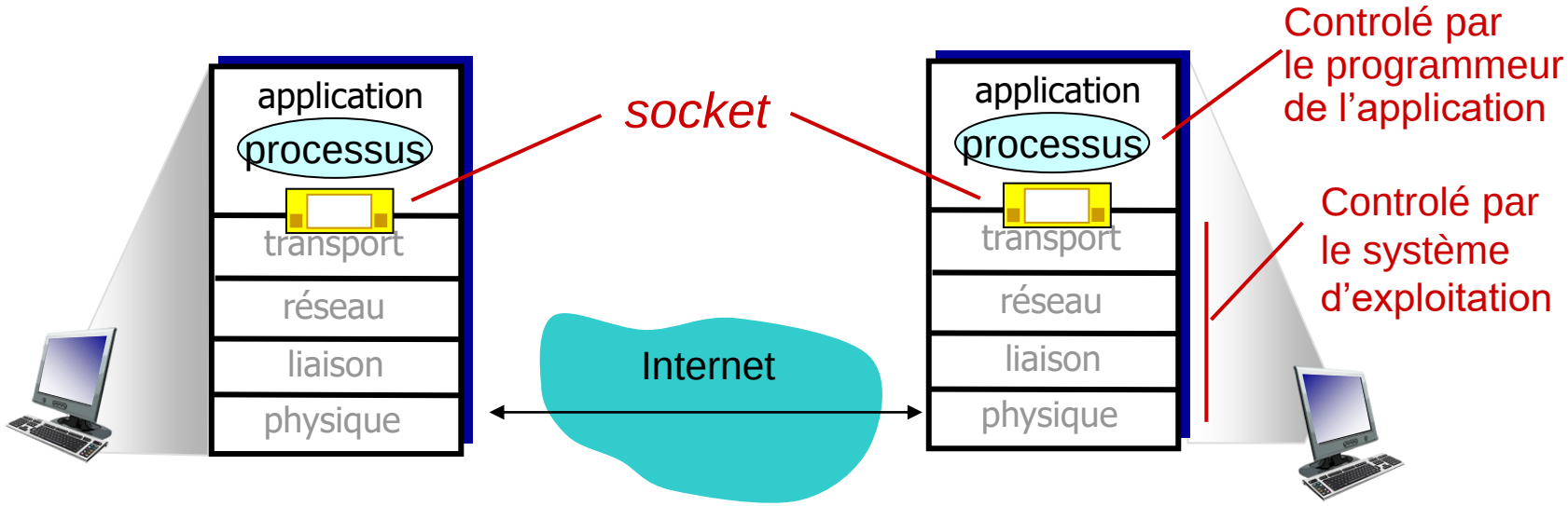
Communication entre processus

Processus: un programme exécuté sur un hôte.

- Dans le même hôte, deux processus communiquent avec communication interprocessus (définie par le système d'exploitation).
- Processus dans différents hôtes communiquent en échangeant des messages

Interface de connexion (*Sockets*)

- Processus émet/reçoit des messages à travers son socket
- Socket analogue à une porte
  - Processus d'émission met les messages à l'extérieur
  - Processus d'émission compte sur l'infrastructure transport de l'autre côté de la porte pour transférer les messages à la porte du côté réception



Socket: Interface entre un processus de la couche application et un protocole de la couche transport. Permet de différencier les données sortant d'un processus ou allant vers un processus.

Donner une adresse à un processus

- Pour recevoir des messages, le processus doit avoir un *identifiant*
- L'équipement hôte a une adresse IP unique 32-bit (IPv4)
- L'adresse IP seule ne suffit pas pour identifier le processus car *plusieurs processus peuvent rouler sur le même hôte*
- *L'identifiant* comprend l'adresse IP et le numéro de port associé au processus.
- Exemples de numéros de port:
  - serveur HTTP : 80
  - serveur courriel SMTP: 25
- Pour émettre des messages HTTP au serveur www.etsmtl.ca:
  - Adresse IP: 142.137.250.114
  - Numéro de port: 80

- Une connexion est définie par : adresses IP source et destination + numéros de port source et destination + protocole de transport (TCP ou UDP)

Le protocole d'application définit

- Le type des messages échangés,
  - Par ex., requête, réponse
- La syntaxe du message :
  - Quels sont les champs contenus dans le message & comment ils sont délimités
- La sémantique du message
  - Sens de l'information contenu dans les champs
- Règles pour déterminer quand et comment les processus échangent des messages

Protocoles ouverts:

- définis dans des RFCs
- Permettent l'interopérabilité
- Par ex., HTTP, SMTP

Protocoles Propriétaires:

- Par ex., Skype

Les services des protocoles de transport de l'Internet

service TCP :

- *transport fiable :* entre les processus client et serveur
- *Contrôle du flux :* l'émetteur ne submergera pas le récepteur
- *Contrôle de congestion :* réduire le débit de transmission quand le réseau est surchargé
- *Ne fournit pas de :* timing, garantie de débit minimal, sécurité
- *Orienté connexion :* création d'une connexion entre les processus client et serveur

service UDP :

- Un transfert de données non fiable entre les processus client et serveur
- Pas de: établissement de connexion, fiabilité, contrôle de flux, contrôle de congestion, synchronisation, débit minimal, ou sécurité

Chapitre 2: la couche application

1. Principes des apps réseaux
   1. Architectures des apps
   2. Interfaces de connexion (*Socket*)
2. Web et HTTP
3. Protocole de transfert de fichier (FTP)
4. Courrier électronique (SMTP, POP3, IMAP)
5. Serveur de noms DNS

Web et HTTP

Un peu de jargon

- Une page Web contient des objets
- Un objet peut être un fichier HTML, une image JPEG, une applet Java, un fichier audio,…
- Une page web contient un fichier de base sous format HTML qui contient des références à des objets
- Chaque objet a une adresse URL (Unique Resource Locator)
- Exemple URL:

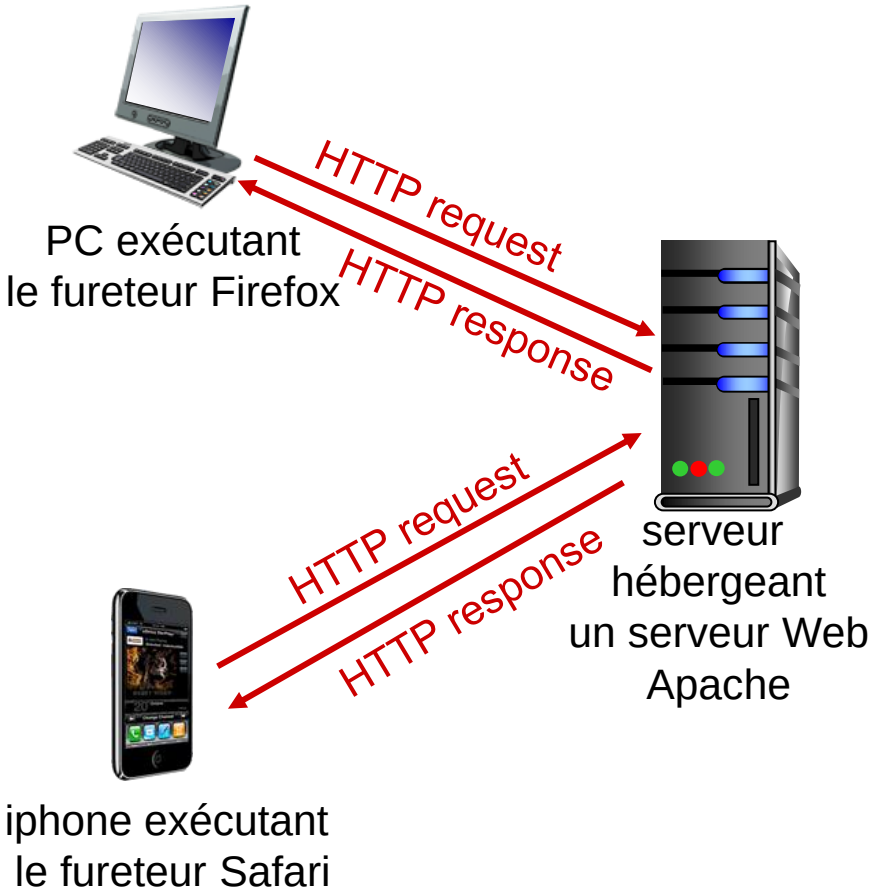
`www.someschool.edu` / `someDept/pic.gif`

Nome de l'hôte — Nom du chemin

HTTP en bref

HTTP : *hypertext transfer protocol*

- Protocole de la couche application pour le Web
- client/serveur utilisant HTTP :
  - *Le client:* fureteur qui envoie des requêtes, reçoit les objets Web et les affiche
  - *Le serveur:* le serveur Web envoie les objets Web comme une réponse à une requête



HTTP en bref

utilise TCP:

- Le client initie la connexion TCP (crée un socket) au serveur, port 80
- Le serveur accepte la connexion TCP du client
- Les messages HTTP échangés entre le fureteur (client HTTP) et le serveur web (serveur HTTP)
- Fermeture de la connexion TCP

HTTP est "sans état"

- Le serveur ne garde aucune information sur les requêtes précédentes

Les protocoles qui conservent "l'état" sont complexes!

- L'historique (état) doit être conservé
- si un serveur/client tombe en panne, leurs perceptions de "l'état" peuvent être inconsistantes et doivent être récupérées

Connexions HTTP

HTTP Non persistant

- Un seul objet est envoyé sur une connexion TCP.
- Une connexion TCP pour chaque objet

HTTP Persistant

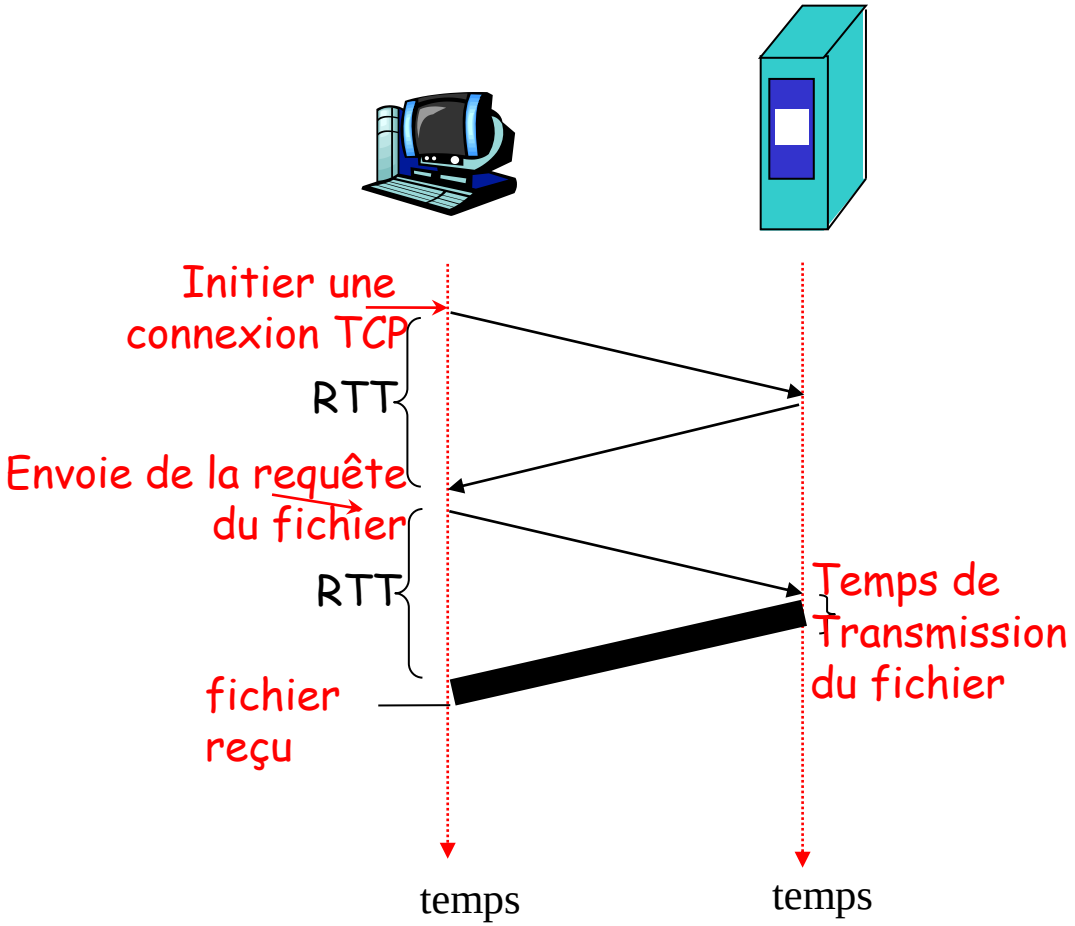
- Plusieurs objets peuvent être envoyés sur une seule connexion TCP entre le client et le serveur.

HTTP Non-Persistant
temps de réponse

Définition du RTT (Round Trip Time): le temps nécessaire à un petit paquet pour faire un aller-retour entre le client et le serveur.

temps de réponse :

- un RTT pour initier la connexion TCP
- un RTT pour la requête HTTP et les premiers octets de la réponse HTTP
- Temps de transmission du fichier



total = 2RTT + temps de transmission de l'objet demandé

HTTP Persistant

Problèmes de HTTP Non-persistant :

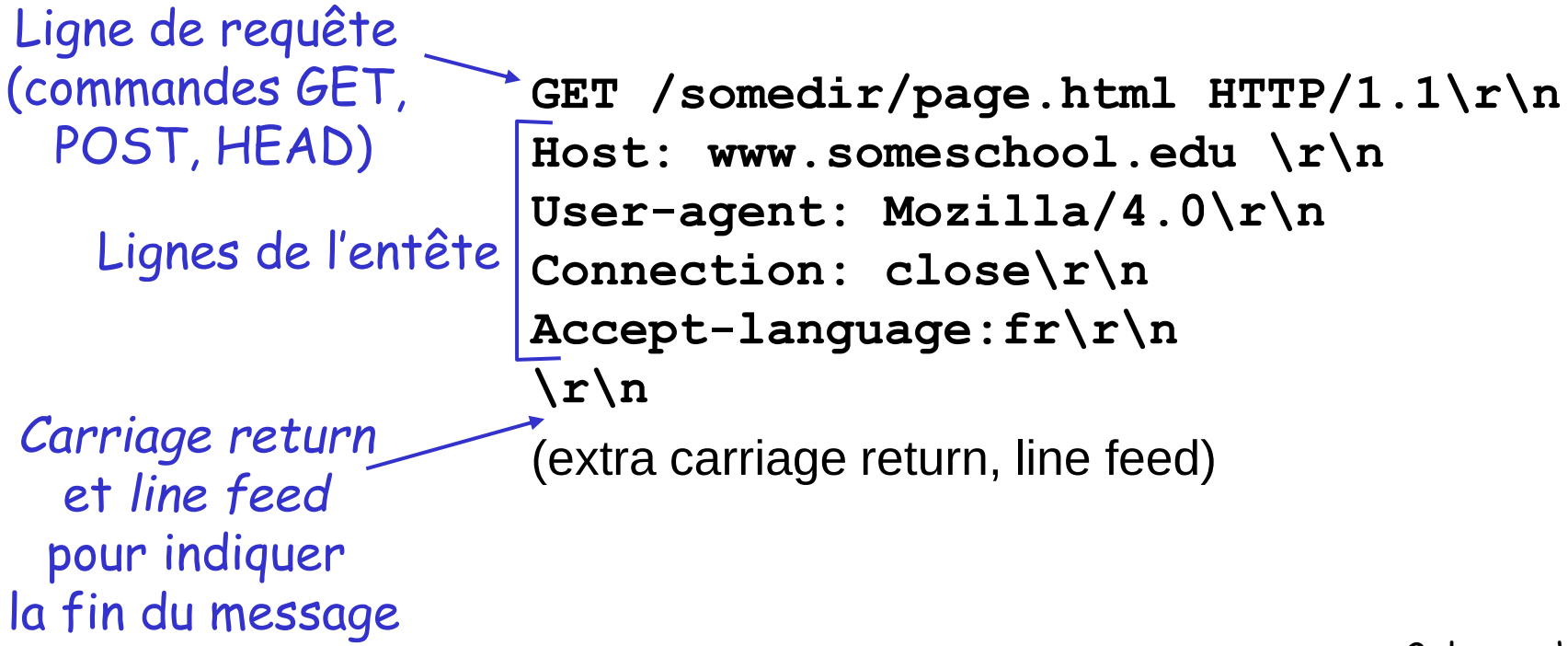
- demande 2 RTTs par objet
- Surchage le système d'exploitation pour chaque connexion TCP
- Les fureteurs souvent ouvrent plusieurs connexions TCP en parallèle pour télécharger les fichiers référencés

HTTP persistant

- Le serveur laisse la connexion ouverte après l'envoie d'une réponse
- Les messages HTTP subséquents entre les mêmes client/serveur sont envoyés sur la même connexion
- Le client envoie des requêtes dès qu'il trouve un objet référencé
- Un RTT pour tous les objets référencés

Message de requête HTTP

- Deux types de messages HTTP: *requête, réponse*
- message de requête HTTP :
  - ASCII (format lisible par les humains)



Envoie d'un formulaire d'entrée

Méthode Post :

- La page Web comprend souvent un formulaire
- Les données saisies dans le formulaire sont envoyées au serveur dans le corps de l'entité

Méthode GET :

- Utilise la méthode GET
- Les données du formulaire sont envoyées dans le champ URL de la requête :

```
GET /somedir/page.html?animal=monkey HTTP/1.1\r\n
```

Types de méthodes

HTTP/1.0

- GET
- POST
- HEAD
  - Demande au serveur de laisser les objets requis à l'extérieure de la réponse

HTTP/1.1

- GET, POST, HEAD
- PUT
  - Téléverser (Uploader) le fichier dans le corps à un chemin spécifié dans l'URL
- DELETE
  - Effacer un fichier spécifié dans l'URL

Codes d'état de réponse HTTP

- Dans la première ligne de la réponse
- Quelques exemples de codes:

200 OK

- Requête réussie, objet requis viendra plus tard dans le msg

301 Moved Permanently

- L'objet requis est déplacé, la nouvelle localisation est spécifiée plus tard dans le message (Location:)

400 Bad Request

- La requête n'est pas comprise par le serveur

404 Not Found

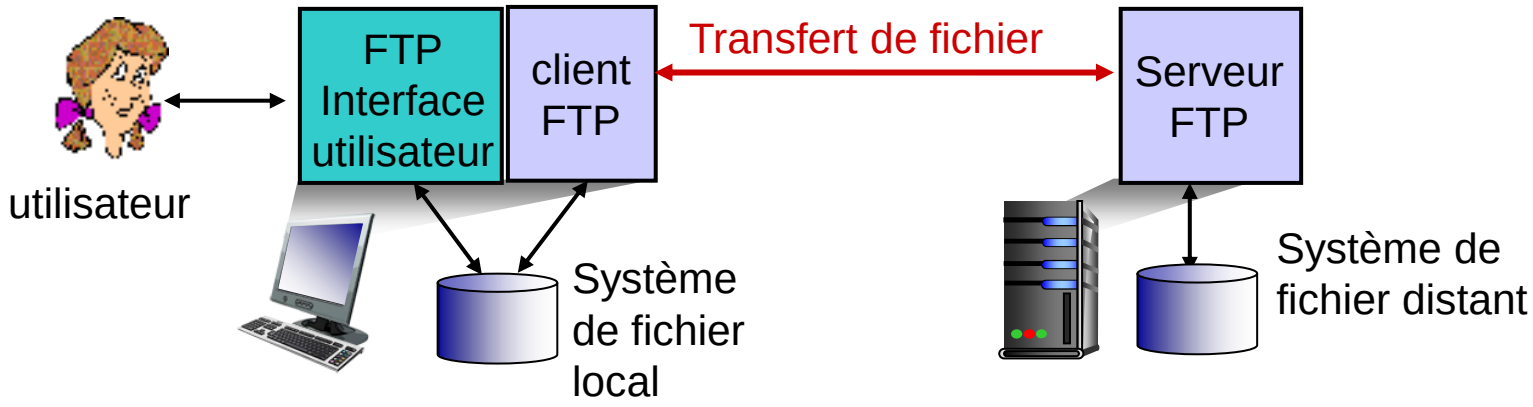
- Le document requis n'est pas sur ce serveur

505 HTTP Version Not Supported

Chapitre 2: la couche application

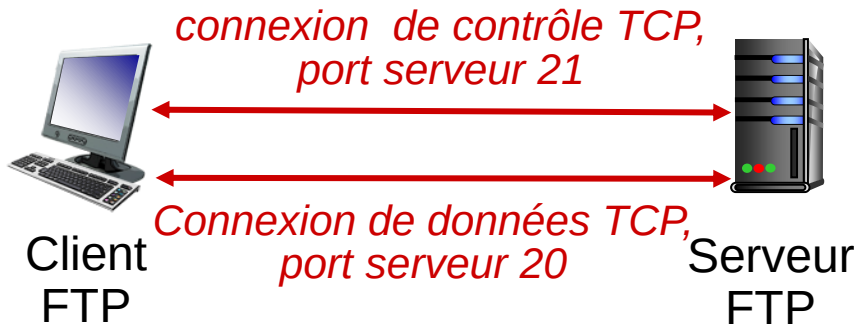
1. Principes des apps réseaux
   1. Architectures des apps
   2. Interfaces de connexion (*Socket*)
   3. Exigences des apps en termes de performance et de fiabilité
2. Web et HTTP
3. Protocole de transfert de fichier (FTP)
4. Courrier électronique (SMTP, POP3, IMAP)
5. Serveur de noms DNS

FTP: protocole de transfert de fichiers



- Transfert de fichier vers/de une machine distante
- Modèle client/serveur
  - *Le client :* initie le transfert (de ou vers une machine distante)
  - *Le serveur :* machine distante
- ftp: RFC 959
- Le client et le serveur ftp utilisent les **ports 21 et 20**

FTP: sépare les connexions: contrôle, données



- Connexion de contrôle: est utilisée pour envoyer les commandes
  - « Hors-bande » (*out of band*)
- Chaque fois que le serveur a un fichier à transmettre, il ouvre une nouvelle connexion TCP appelée connexion de données
- Le serveur FTP maintient "l'état": répertoire actuel, authentification précédente

1. Le client FTP contacte le serveur FTP au port 21 à travers TCP (connexion de contrôle)
2. Le client est authentifié sur la connexion de contrôle
3. Le client parcours le répertoire de fichier distant en envoyant des commandes sur la connexion de contrôle.
4. Quand le serveur reçoit une commande de transfert d'un fichier,
   - il ouvre une *2ème connexion* TCP (appelée connexion de données) avec le client pour transmettre le fichier
   - Après le transfert du fichier, le serveur ferme la connexion de données.

Commandes et réponses FTP

Ex. commandes:

- Envoyées en texte ASCII sur la connexion de contrôle
  - **`USER` *`username`***
  - **`PASS` *`password`***
  - **`LIST`** donne la liste des fichiers dans le répertoire actuel
  - **`RETR filename`** demande un fichier (gets)
  - **`STOR filename`** stocke (puts) le fichier sur la machine distante

Ex. codes de retour

- Code d'état et message d'état (comme en HTTP)
  - **`331 Username OK, password required`**
  - **`125 data connection already open; transfer starting`**
  - **`425 Can't open data connection`**
  - **`452 Error writing file`**

Chapitre 2: la couche application

1. Principes des apps réseaux
   1. Architectures des apps
   2. Interfaces de connexion (*Socket*)
   3. Exigences des apps en termes de performance et de fiabilité
2. Web et HTTP
3. Protocole de transfert de fichier (FTP)
4. Courrier électronique (SMTP, POP3, IMAP)
5. Serveur de noms DNS

DNS: Domain Name System

Humains : plusieurs identifiants :

- NAS, nom, passeport

hôtes Internet et routeurs :

- Adresse IP (32 bit) – utilisée pour l'adressage des datagrammes
- "nom", par ex., www.yahoo.com – utilisé par les humains

Q: comment faire la correspondance entre un nom et une adresse IP?

- Système de noms de domaines (Domain Name System) :
  - Base de données distribuée implémentée en hiérarchie de plusieurs serveurs de noms et qui maintiennent les correspondances : (nom de domaine -> adresse IP)
- Protocole Domain Name Service (DNS) :
  - Les hôtes et routeurs doivent communiquer avec les serveurs de noms à travers le protocole DNS pour trouver la correspondance (nom, adresse IP)
  - Protocole de niveau application qui utilise soit TCP soit UDP au niveau transport

Note : la résolution des noms de domaines est une fonction de base dans l'Internet, mais implémentée comme un protocole de niveau application

DNS

Les services DNS

- traduction du nom d'une machine en une adresse IP
- dénomination d'hôte
  - Nom canonique
  - Alias
- attribution d'un alias à un serveur de messagerie
- distribution de charge
  - Plusieurs instances du même serveur Web : plusieurs adresses IP pour le même nom

Pourquoi ne pas centraliser le DNS?

- fragilité d'un site central unique
- volume de trafic
- Une base de données centralisée peut être située trop loin par rapport à quelques utilisateurs
- maintenance

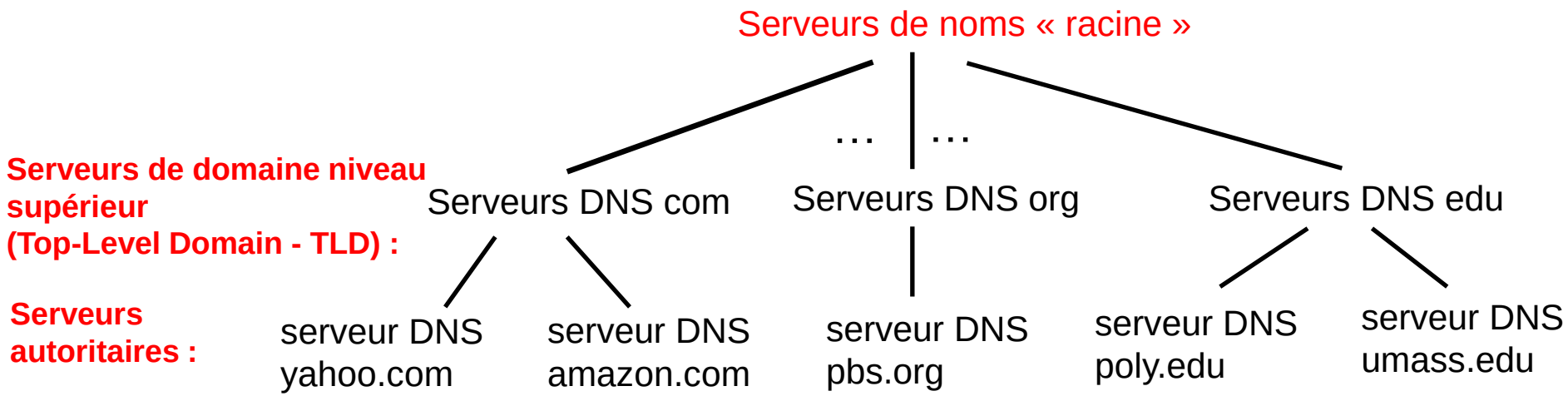
Pas de mise à l'échelle!

Les serveurs de nom de domaine écoutent sur le **port 53**

Les noms de domaines

- Noms des hôtes :
  - www.yahoo.com, mail.yahoo.com, help.yahoo.com
  - signets.etsmtl.ca, ens.etsmtl.ca, profs.etsmtl.ca, safirh.etsmtl.ca
- Un domaine est un ensemble d'ordinateurs :
  - yahoo.com, etsmtl.ca, google.ca
- Il est possible de créer des sous-domaines :
  - *logti.etsmtl.ca, departements.etsmtl.ca*
- Le domaine de niveau supérieur (Top-Level Domain – TLD)
  - .ca, .fr. .com, .org

Base de données répartie, hiérarchique



Un client veut l'adresse IP de www.amazon.com :

1. Le client demande au serveur DNS racine de trouver le serveur DNS .com
2. Le client demande au serveur DNS .com de trouver le serveur DNS de amazon.com
3. Le client demande au serveur DNS de amazon.com de trouver l'adresse IP de www.amazon.com

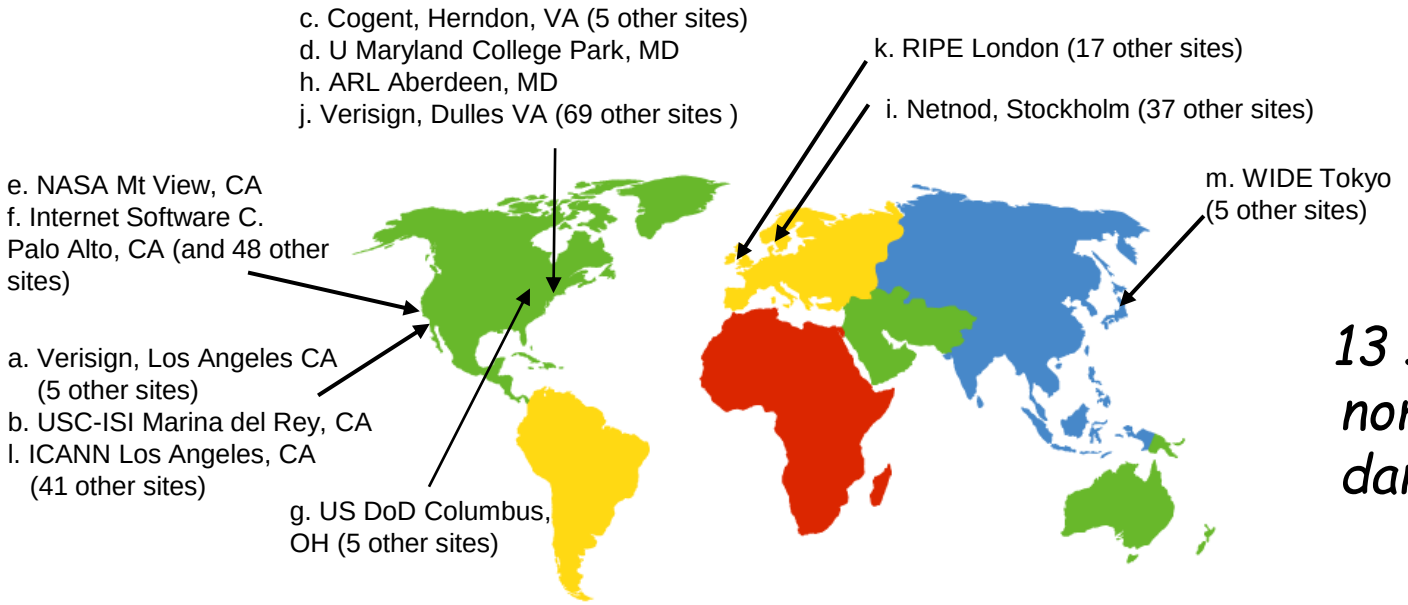
TLD et serveurs DNS autoritaires

- Serveurs de noms autoritaires:
  - Serveur DNS appartenant à l'organisation, donne la correspondance nom-adresse IP pour les serveurs de l'organisation (par ex., Web, mail).
  - Peut être maintenu par une organisation ou par un fournisseur d'accès

- Serveurs de domaines de niveau supérieur (*Top-Level Domain servers – TLD servers*):
  - responsables pour les domaines génériques tels que: com, org, net, edu, … et tous les domaines de niveau supérieur des pays tels que: ca, uk, fr.
  - Environ 560 domaines TLD, 260 nationaux et 300 génériques

DNS: serveurs de nom « racine »

- Contacté par le serveur de nom local qui ne peut pas trouver l'adresse IP qui correspond à un nom
- Serveur de nom « racine » (*root name server*) :
  - Retourne l'adresse(s) de serveur(s) TLD responsable d'un domaine (edu, ca, fr…)



13 serveurs de nom « racine » dans le monde

Serveur DNS local

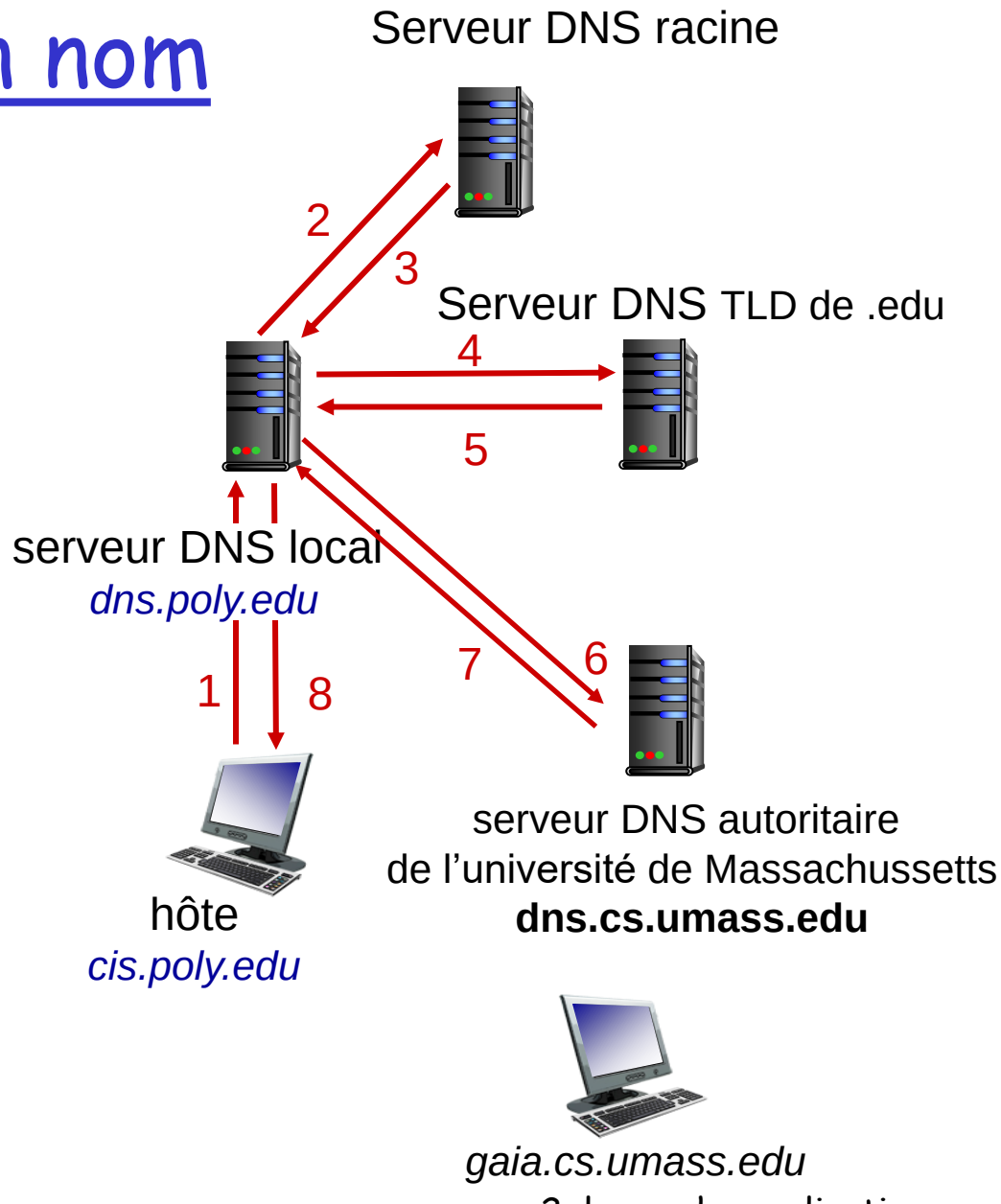
- N'appartient pas à la hiérarchie
- Chaque FAI (FAI résidentiel, compagnie, université) en a un.
  - appelé aussi "serveur de noms par défaut"
- Quand un hôte envoie une requête DNS, la demande est envoyée à son serveur DNS local
  - Il possède un cache local contenant les couples (nom, adresse IP) récemment demandés.
  - Il agit comme proxy, transfère les demandes dans la hiérarchie

Exemple de résolution d'un nom

- Un hôte cis.poly.edu souhaite l'adresse IP de gaia.cs.umass.edu (umass: university of Massachussetts)

recherche itérative:

- Le serveur DNS contacté répond avec le nom du serveur DNS à contacter : "je ne connais pas ce nom mais demande à ce serveur"



Exemple de résolution d'un nom

recherche récursive:

- Mettre le fardeau sur le serveur DNS contacté
- grande charge sur les serveurs DNS racine?