INF3271
Téléinformatique

Chargé de cours Ammar Hamad

UQAM

N.B. Ces diapositives sont dérivées du matériel pédagogique de J. Kurose et autres.

la « couche » application

Nos objectifs:

- Aspects conceptuels et d'implantation des protocoles d'application de réseautique, dont
  - Le modèle client-serveur
  - Les modèles de service
- Découvrir les protocoles en examinant des protocoles d'application communs

Autres

- Protocoles spécifiques :
  - http
  - ftp
  - smtp
  - pop
  - dns
- La programmation d'applications réseau
  - L'API des « prises »

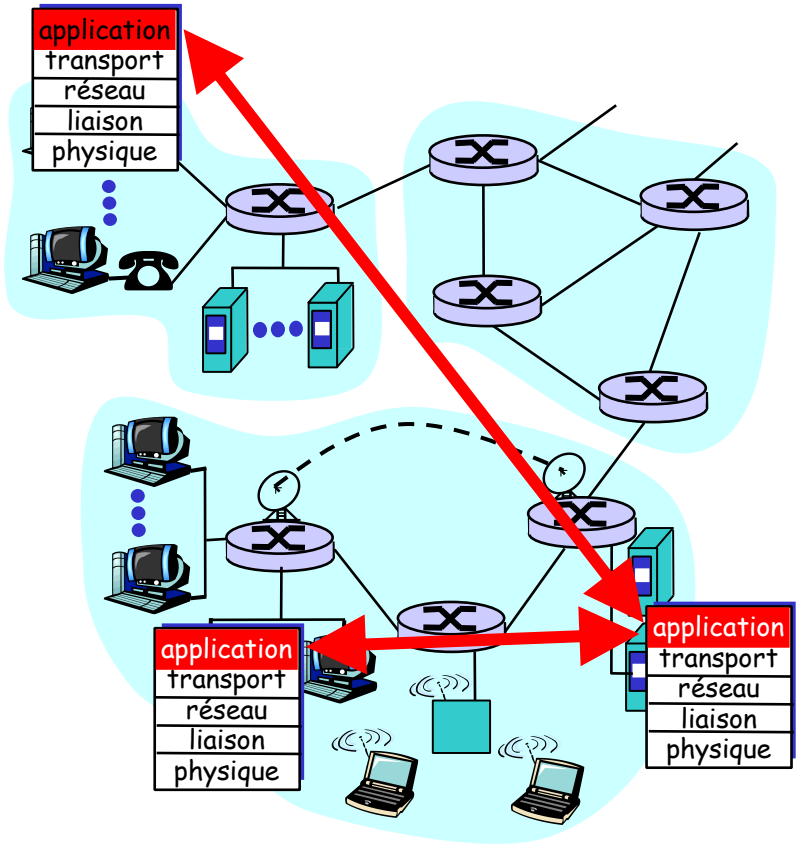
Applications et protocoles de la couche application

Application: des processus répartis communiquant

- Exécutés sur des hôtes en espace usager
- Échangent des messages pour exécuter l'application
- p.ex., courriel, ftp, Web

Protocole de la couche application

- Un morceau d'une application
- Définit les messages échangés par les applications et les actions résultantes
- Utilise les services de communication fournis par les couches de protocole inférieures (TCP, UDP)



Applications réseau : un peu de jargon

Processus: un programme exécuté sur un hôte.

- Sur un hôte, deux processus communiquent via des communications interprocessus (définies par l'OS).
- Les processus exécutés sur des hôtes différents communiquent grâce à un protocole de couche application

- Agent usager: un processus logiciel, interfacé avec un usager et le réseau.
  - implémente un protocole de niveau application
  - Web: fureteur
  - E-mail: lecteur
  - streaming audio/video: «media player»

Modèle client-serveur

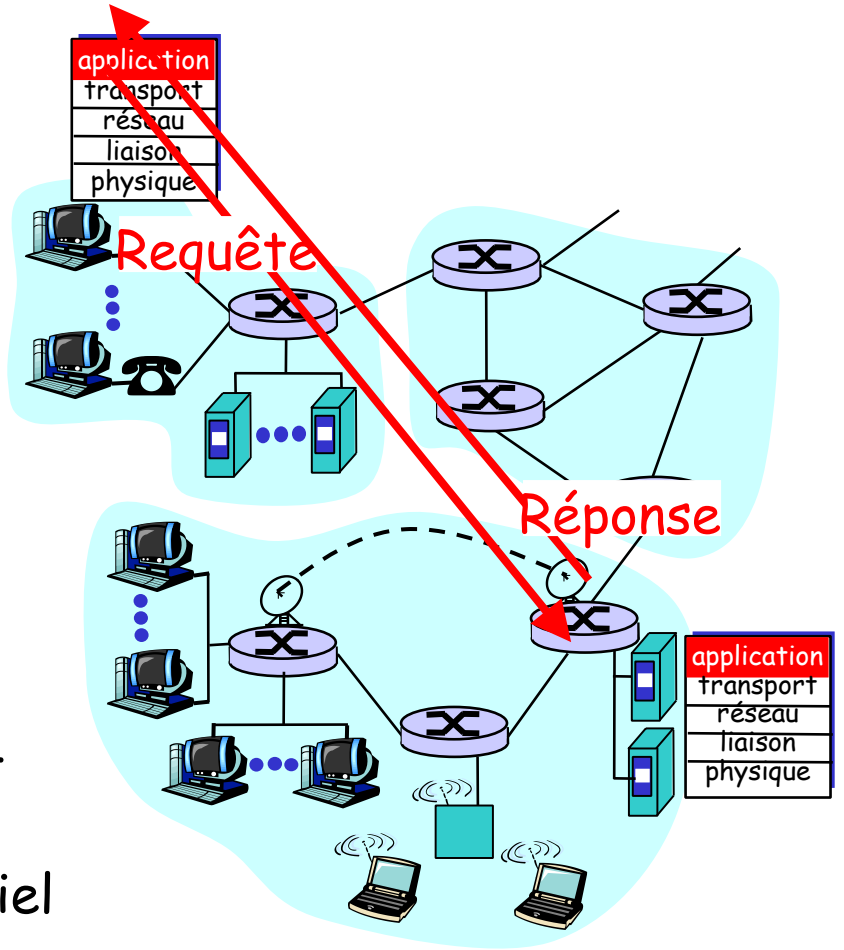
Une application réseau typique a deux composants : un *client* et un *serveur*

Client:

- Initie le contact avec le serveur («parle le premier»)
- Demande typiquement un service d'un serveur,
- Web: le client est implanté dans le fureteur ; e-mail: dans le lecteur de courriel

Serveur:

- Fournit le service demandé au client
- p.ex., le Web serveur envoie la page Web demandée, le serveur de courriel délivre le courrier.



Protocoles de la couche application (suite).

API: application programming interface

- Définit l'interface entre les couches application et de transport
- Prise (socket): API de l'Internet
  - Deux processus communiquent en envoyant, et en y lisant des données d'une prise

Q: comment un processus identifie-t-il le processus avec lequel il désire communiquer ?

- L'adresse IP de la machine exécutant l'autre processus
- Le «numéro de port» qui permet à l'hôte récepteur de déterminer à quel processus local le message doit être délivré.

... nous reviendrons sur ce point.

Quel service de transport est requis?

Pertes de données

- Certaines applications (p.ex., audio) peuvent tolérer certaines pertes
- D'autres (p.ex., transfert de fichiers, telnet) requièrent 100% de robustesse

Délais

- Certaines applications (p.ex., téléphonie Internet, jeux interactifs) requièrent de faibles délais pour être efficace

Bande passante

- Certaines applications (p.ex., multimédia) requièrent une bande passante minimale pour être efficaces
- D'autres (applications élastiques) utilisent la bande passante qu'elles peuvent obtenir

Les besoins en transport d'applications communes

Application	Perte de données	Bande passante	Sensitivité délai
transfert de fichiers	pas de perte	élastique	non
courriel	pas de perte	élastique	non
documents web	tolère les pertes	élastique	non
audio/vidéo T.R.	tolère les pertes	audio: 5Kb-1Mb vidéo:10Kb-5Mb	oui, 100's msec
audio/vidéo stockée	tolère les pertes	idem	oui, qques secs
jeux interactifs	tolère les pertes	quelques Kbps	oui, 100's msec
applis financières	pas de perte	élastique	oui et non

Les services des protocoles de transport de l'Internet

Service TCP:

- *Orienté connexion:* setup requis entre client, serveur
- *Transport fiable* entre processus expéditeur et récepteur
- *Contrôle de flot:* l'expéditeur ne surcharge pas le récepteur
- *Contrôle de congestion :* contreindre l'expéditeur quand le réseau est saturé
- *Ne donne pas:* garanties de délai et de bande passante minimale

Service UDP:

- Transfert de données non-fiable entre processus expéditeur et récepteur
- Ne fournit pas : établissement de connexion, fiabilité, contrôle de flot, de congestion, garanties de délai ou de débit.

Q: pourquoi l'utilise-t-on ? Pourquoi faut-il UDP?

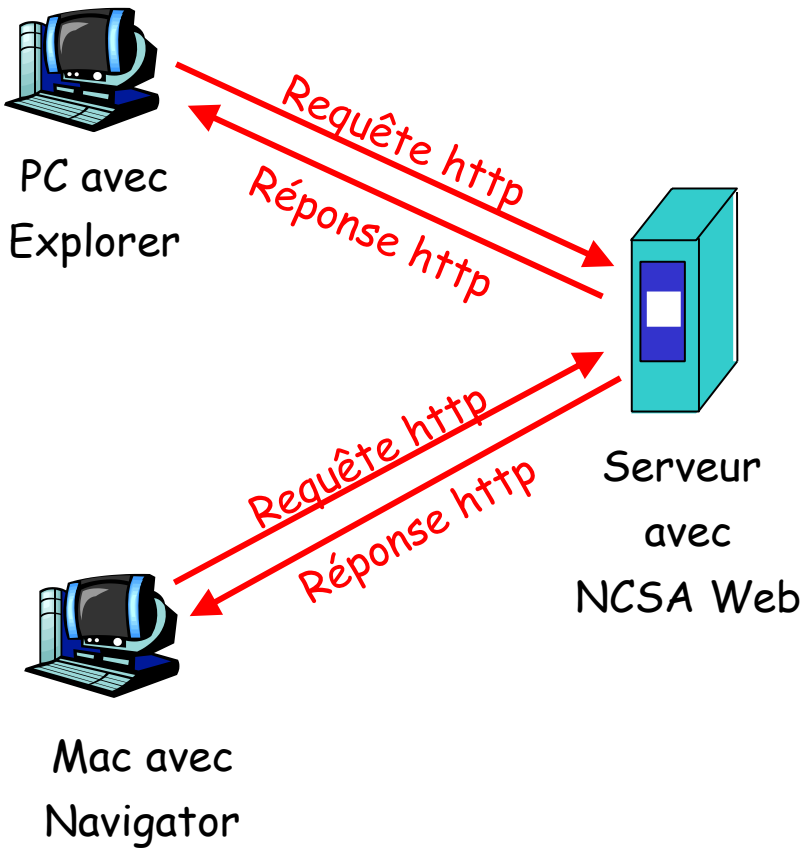
Applications de l'Internet: protocoles de transport utilisés

Application	Protocole de couche application	Protocole de transport utilisé
courriel	smtp [RFC 821]	TCP
émulation de terminal	telnet [RFC 854]	TCP
Web	http [RFC 2068]	TCP
transfert de fichiers	ftp [RFC 959]	TCP
streaming multimedia	propriétaire (e.g. RealNetworks)	TCP or UDP
serveur de fichiers	NFS	TCP or UDP
téléphonie Internet	propriétaire (p.ex., Vocaltec)	typiquement UDP

Le Web: le protocole http

http: hypertext transfer protocol

- Protocole de niveau application du Web
- Modèle client/serveur
  - *client:* fureteur qui requiert, reçoit et affiche des objets Web
  - *serveur:* le serveur Web envoie des objets en réponse aux requètes :
- http1.0: RFC 1945
- http1.1: RFC 2068



Le protocole http suite

http: service TCP :

- Le client initie une connexion TCP (crée la prise) au serveur, port 80
- Le serveur accepte la connexion TCP du client
- Les messages http sont changés entre le fureteur (client http) et le serveur Web (serveur http)
- Fermeture de la connexion TCP

Http est « sans état »

- Le serveur ne conserve aucune information sur les requêtes antérieures

Aparté

Les protocoles qui conservent un état sont complexes!

- L'historique (état) doit être préservé
- Si le serveur/client crashe, leur vision de l'état peut être incohérente, et doit être réconciliée

Exemple d'utilisation d'http

Supposons qu'un usager compose l' URL suivant :
www.someSchool.edu/someDepartment/home.index
(contient du texte, les références de 10 images jpeg)

1a. Le client http initie une connexion TCP au serveur http (processus) à www.someSchool.edu. Le port 80 est utilisé par défaut pour un serveur http.

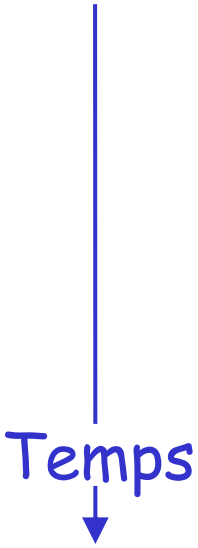
1b. Le serveur http sur l'hôte www.someSchool.edu attend une connexion TCP sur le port 80. Il accepte la connexion et notifie le client

2. Le client http envoie un *message de requête* http (contenant l'URL) dans une prise de connexion TCP

3. Le serveur http reçoit un message de requête, formule un *message de réponse* contenant l'objet requis (someDepartment/home.index), il envoie ce message dans la prise

Temps

Example d'utilisation d'http (suite)



4. Le serveur http ferme la connexion TCP.

5. Le client http reçoit un message de réponse contenant un fichier html, et l'affiche. En traitant le fichier html, il découvre 10 objets jpeg référencés

6. Les étapes de 1-5 sont répétées pour chaque objet jpeg

Connexions persistentes et non-persistentes

Non-persistente

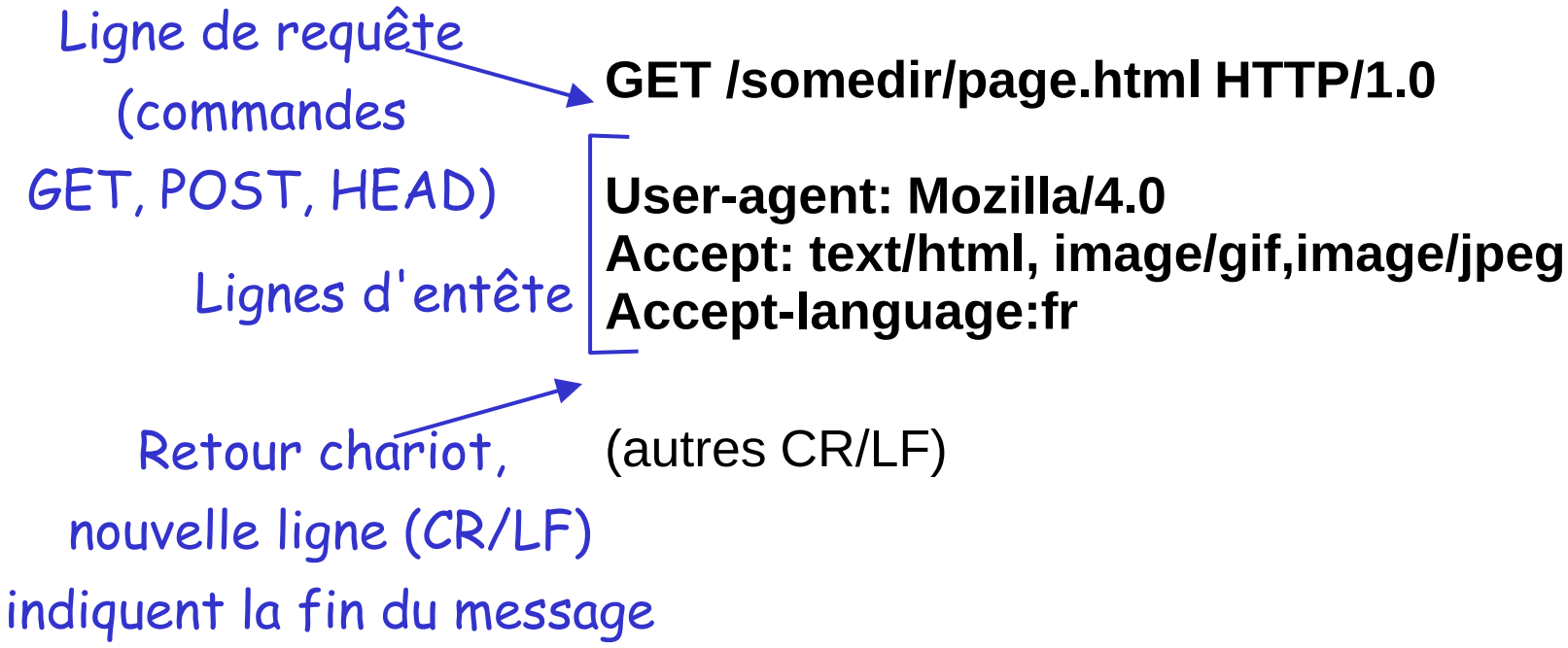
- http/1.0: le serveur traite la requête, répond, ferme la connexion TCP
- 2 RTTs pour aller chercher l'objet
  - La connexion TCP
  - Requête et transfert de l'object
- Chaque transfert souffre du «démarrage lent» de TCP
- Beaucoup de fureteurs ouvrent de multiples sessions en parallèle

Persistente

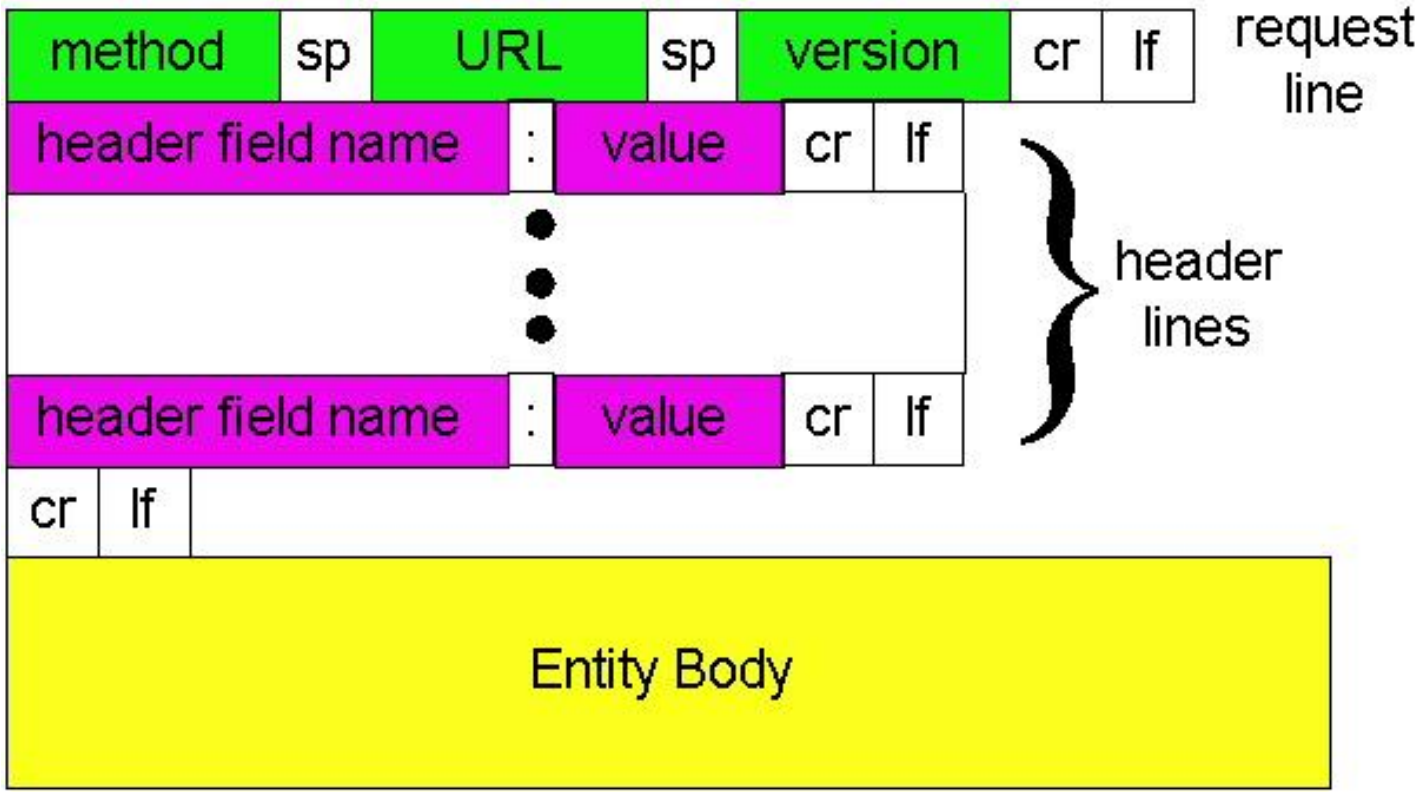
- Par défaut pour http/1.1
- Sur la même connexion TCP: serveur, traitement de la requête, répond, traite une nouvelle requête
- Le client envoie une requête pour tous les objets référencés dès qu'il reçoit le HTML de base
- Moins de RTT et de démarrages lents.

Format de messages http : la requête

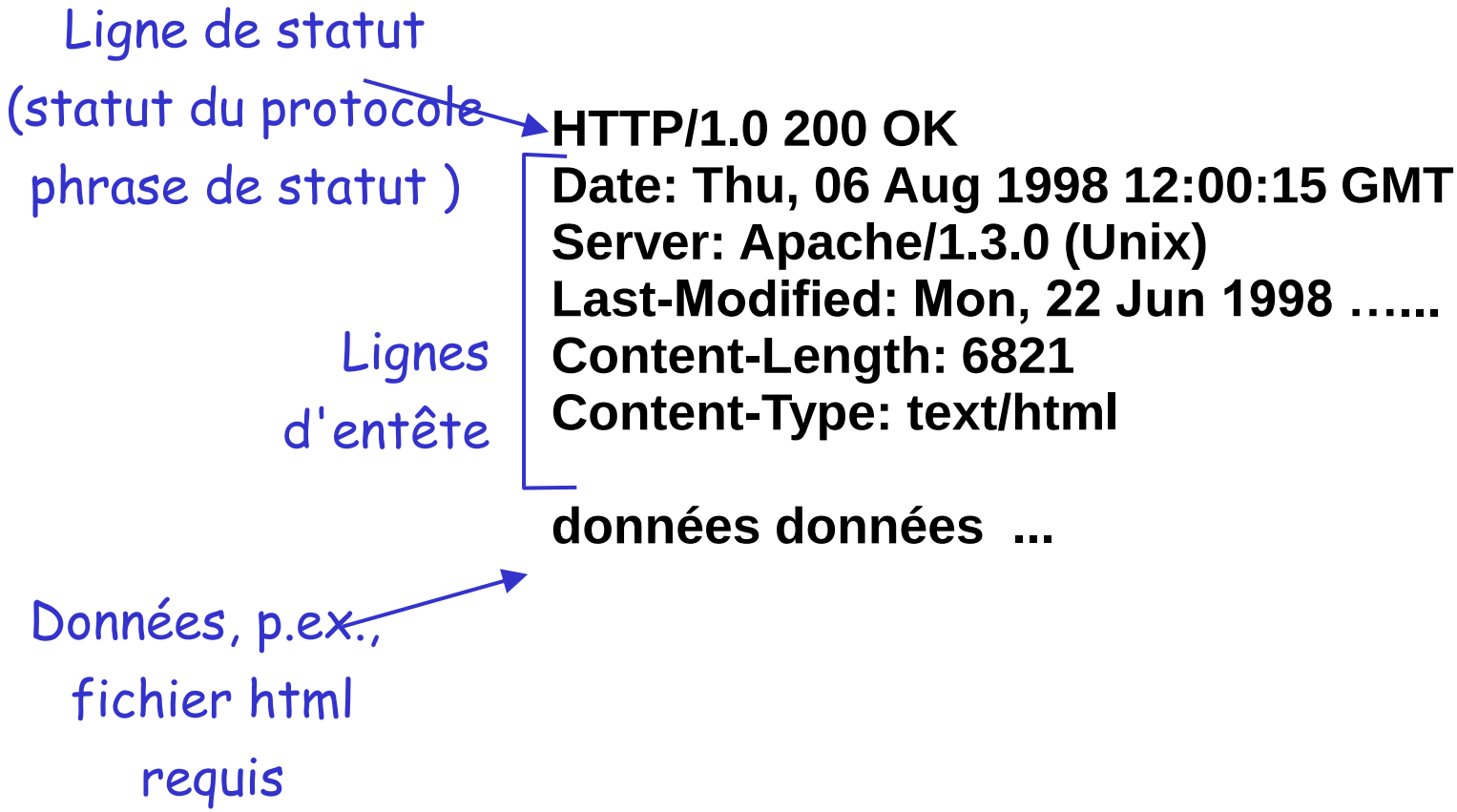
- deux types de messages http : *la requête, la réponse*
- Message de requête http:
  - ASCII (format lisible)



Message de requête http : format général



Format de message http : réponse



Codes de statut de la réponse http

Dans la première ligne du message de réponse du serveur au client

Quelques exemples:

200 OK

- La requête a réussi, l'objet demandé suit dans le message

301 Moved Permanently

- L'objet demandé a été déplacé, une nouvelle localisation est spécifiée plus loin dans le message (Location:)

400 Bad Request

- Le serveur n'a pas compris la requête

404 Not Found

- Le serveur n'a pas trouvé le document demandé

505 HTTP Version Not Supported

Essayez d'imiter un client http

1. Telnet à votre serveur Web préféré:

Telnet www.eurecom.fr 80

Ouvre une connexion TCP au port 80 (port par défaut) sur www.eurecom.fr. Tout ce qui est tapé est envoyé au port 80 sur www.eurecom.fr

2. Composez une requête http GET:

GET /~ross/index.html HTTP/1.0

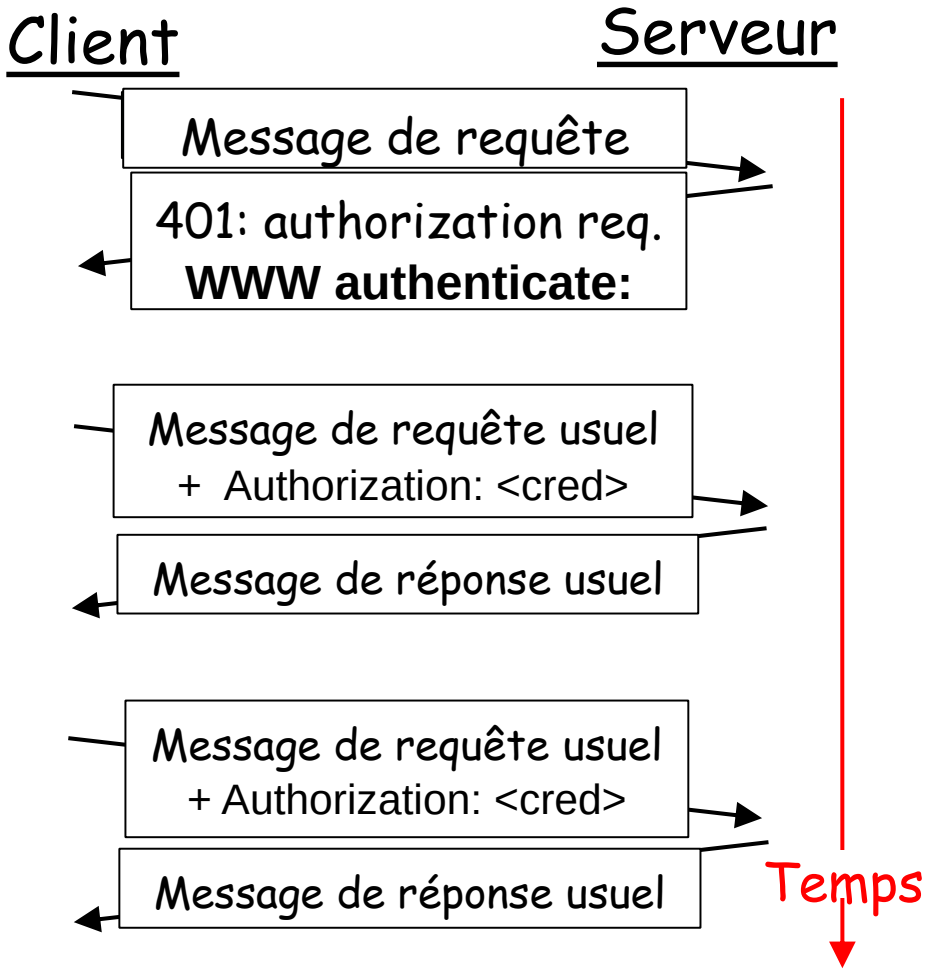
En tapant ceci (et deux «enter»), vous transmettez cette requête GET minimale (mais complète) au serveur http

3. Regardez la réponse transmise par le serveur.

Interaction usager-serveur: authentification

Authentification : contrôle d'accès au serveur de contenu

- Vérification des pièces d'identité: typiquement: nom, mot de passe
- Sans état: le client doit présenter une authorisation dans chaque requête
  - autorisation: ligne d'entête dans chaque requête
  - S'il n'y a pas d'entête d'autorisation, le serveur refuse l'accès, envoie l'entête **WWW authenticate:** dans la réponse

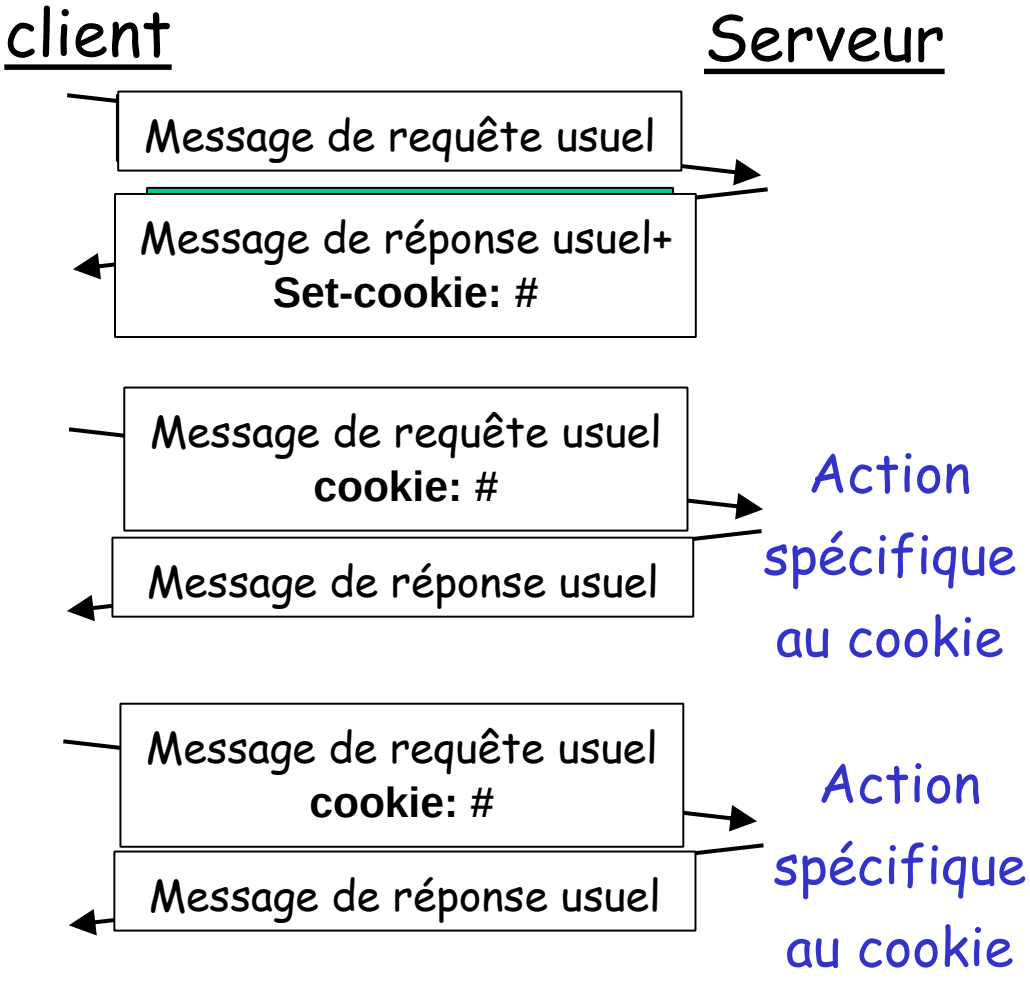


Cookies: conserver un « état »

- Un nombre généré et mémorisé par le serveur, utilisé par la suite pour:
  - Authentification
  - Mémoriser des préférences de l'usager et des choix antérieurs
- Le serveur transmet un «cookie» au client dans sa réponse

  Set-cookie: 1678453
- Le client inclut le « cookie » dans ses requêtes ultérieures

  cookie: 1678453

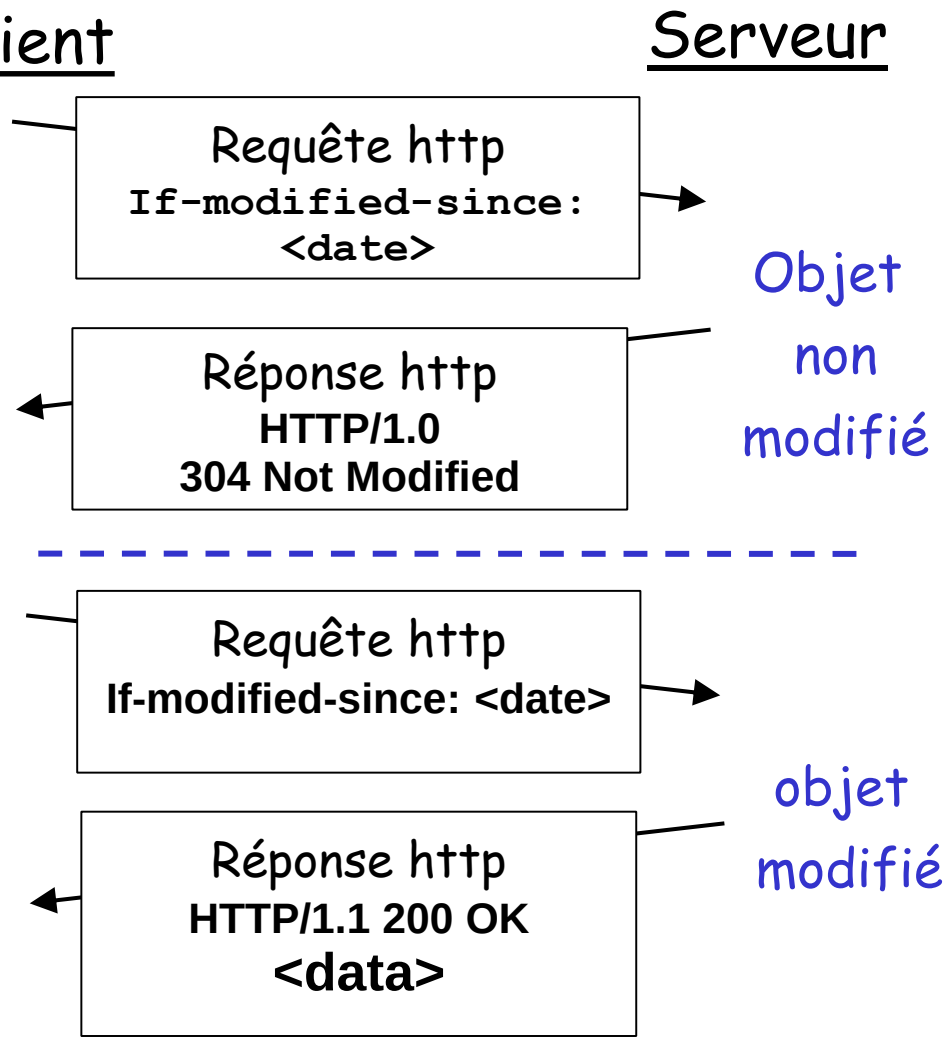


GET conditionnel: cache du coté du client

- Objectif: ne pas transmettre l'objet si le client en a une copie à jour dans sa cache
- client: spécifier la date de l'objet caché dans la requête

  If-modified-since: <date>
- serveur: la réponse ne contient pas d'object si la copie cachée est à jour:

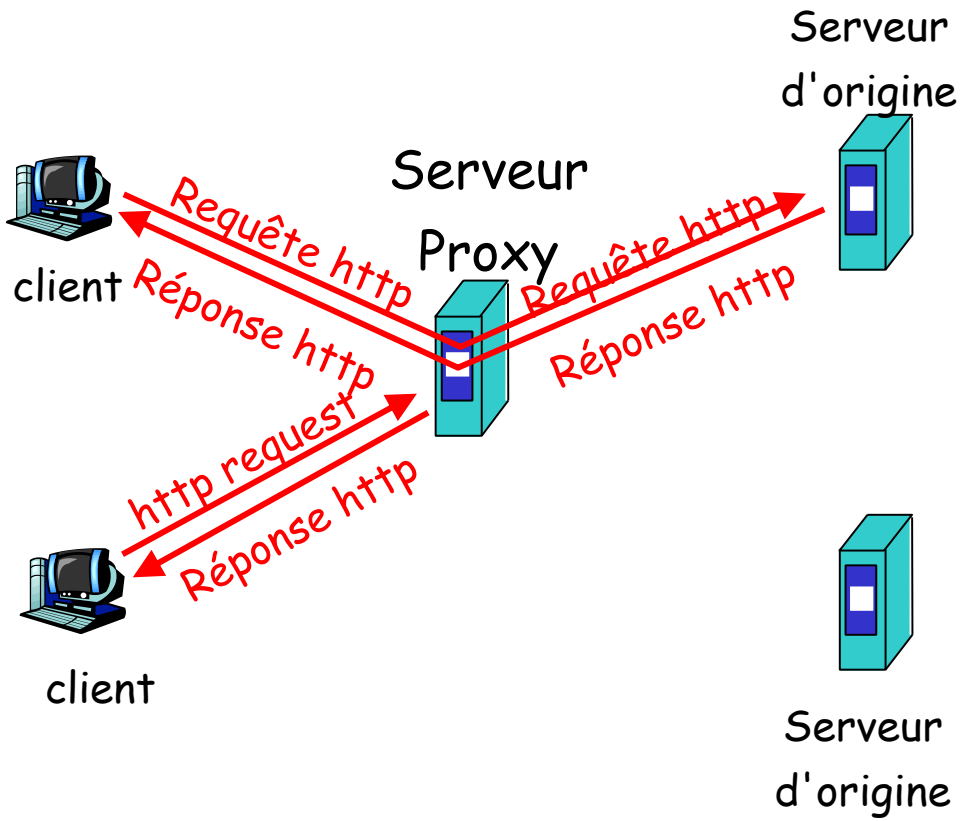
  HTTP/1.0 304 Not Modified



Caches Web (serveur « de substitution» dit proxy)

Objectif: répondre aux requêtes des clients sans faire intervenir le serveur d'origine.

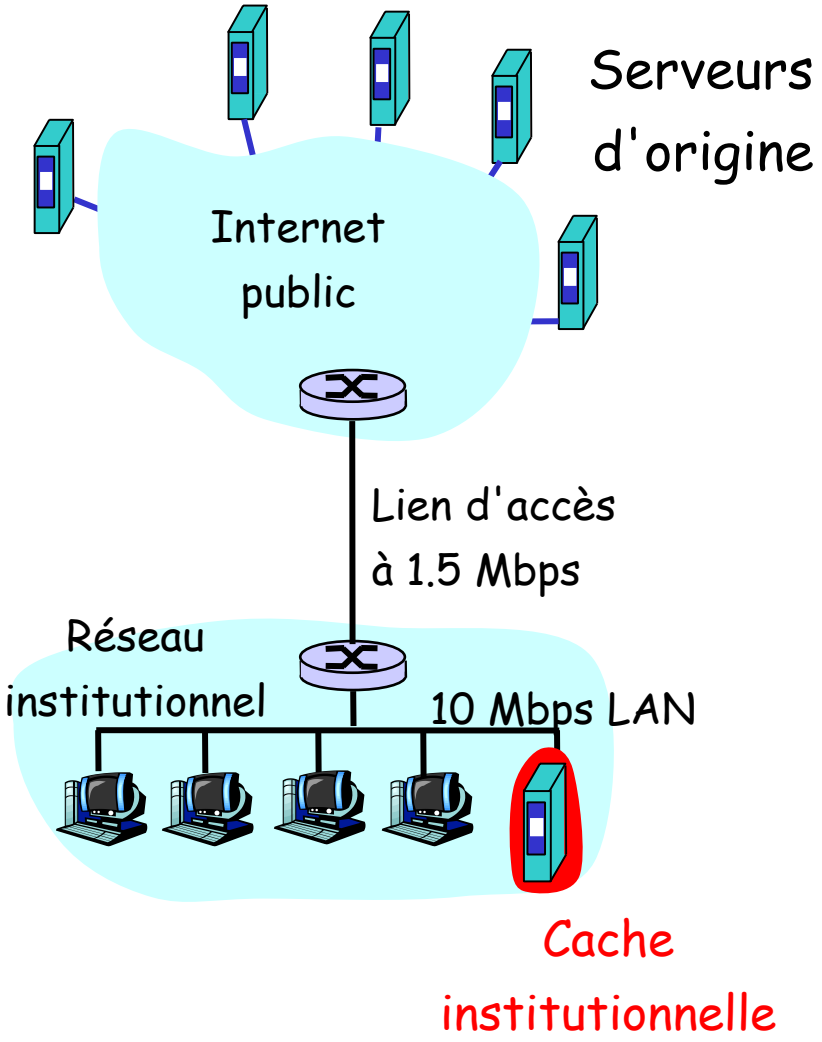
- L'usager configure le fureteur: accès Web via une cache web
- Le client transmet toutes les requêtes http à la cache web
  - objet dans la cache web : la cache web retourne l'object
  - Sinon la cache web demande l'objet au serveur d'origine, puis le transmet au client



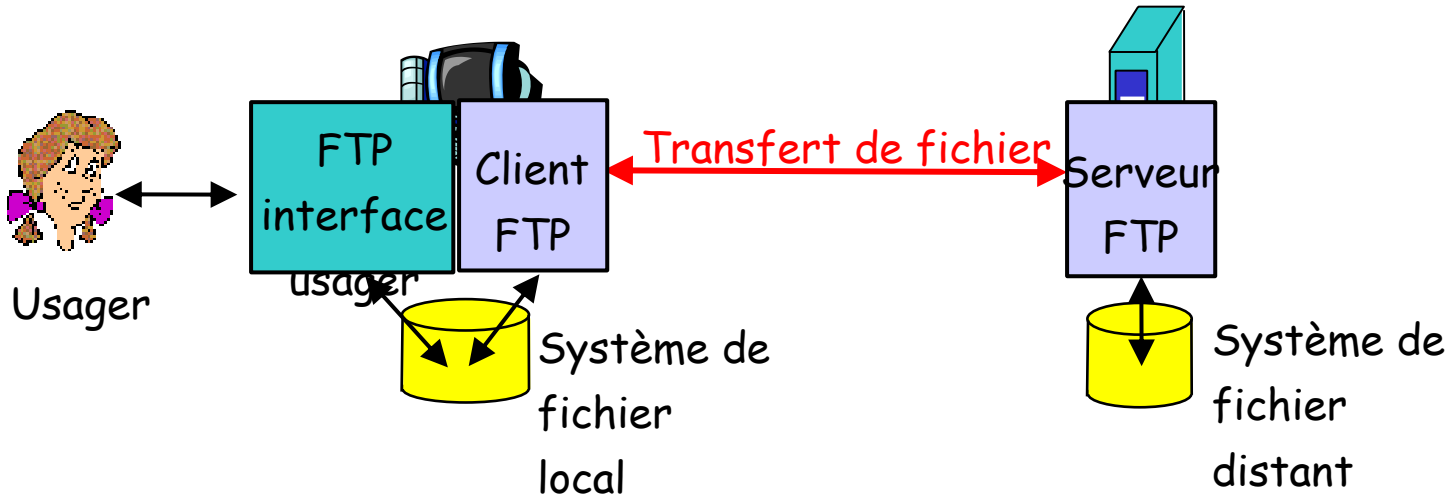
Pourquoi avoir des caches web ?

Supposons: la cache est «proche» du client (p.ex., dans le même réseau)

- Temps de réponse plus courts
- Réduction du trafic aux serveurs distants.
  - Le lien de l'institution au réseau local de l'ISP est souvent le goulot d'étranglement



Ftp: le protocole de transfert de fichiers

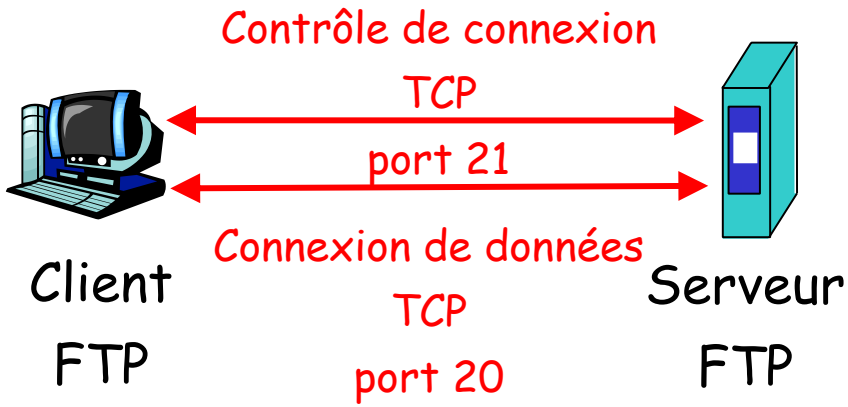


- Transfert de fichier de/à un hôte distant
- Modèle client/serveur
  - *client:* côté qui initie le transfert (soit du/au noeud distant)
  - *serveur:* hôte distant
- ftp: RFC 959
- Serveur ftp: port 21

ftp: séparation des connexions de contrôle et de données

- Le client ftp contacte le serveur ftp au port 21, en spécifiant TCP comme protocole de transport
- Deux connexions TCP sont ouvertes en parallèle:
  - Contrôle : échange de commandes, réponses entre client et serveur.

    Contrôle « hors bande »
  - Données : fichier du/au serveur
- Le serveur ftp gère un « état »: répertoire courant, authentification



Commandes et réponses ftp

Exemples de commandes:

- Transmis comme texte ASCII sur le canal de contrôle
- **USER *username***
- **PASS *password***
- **LIST** liste des fichiers dans le répertoire courant
- **RETR filename** récupère (gets) le fichier
- **STOR filename** stocke (puts) un fichier sur le système distant

Exemples de réponses

- Code de statut et phrase (comme pour http)
- **331 Username OK, password required**
- **125 data connection already open; transfer starting**
- **425 Can't open data connection**
- **452 Error writing file**

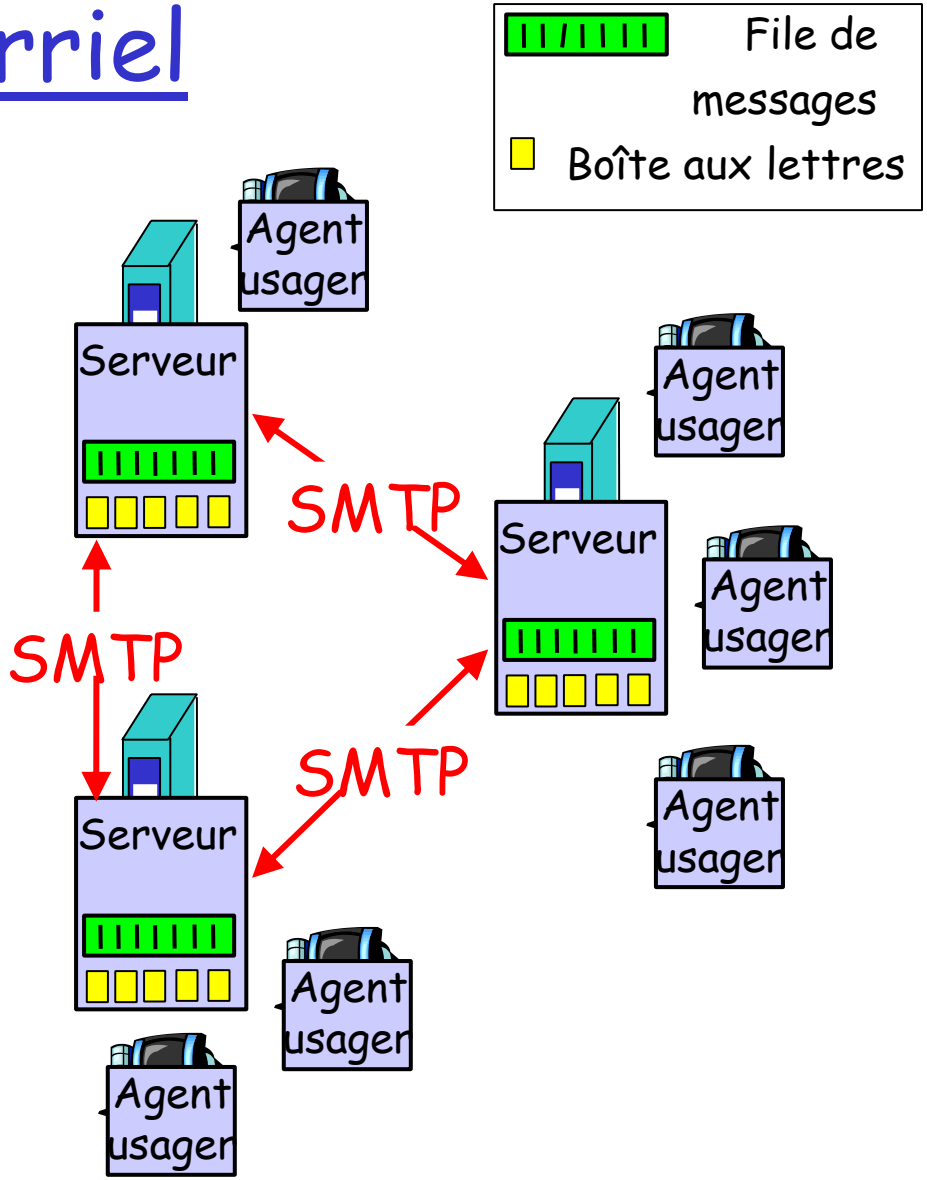
Courriel

Trois composants majeurs :

- Agents usager
- Serveurs de courriel
- simple mail transfer protocol: smtp

Agent Usager

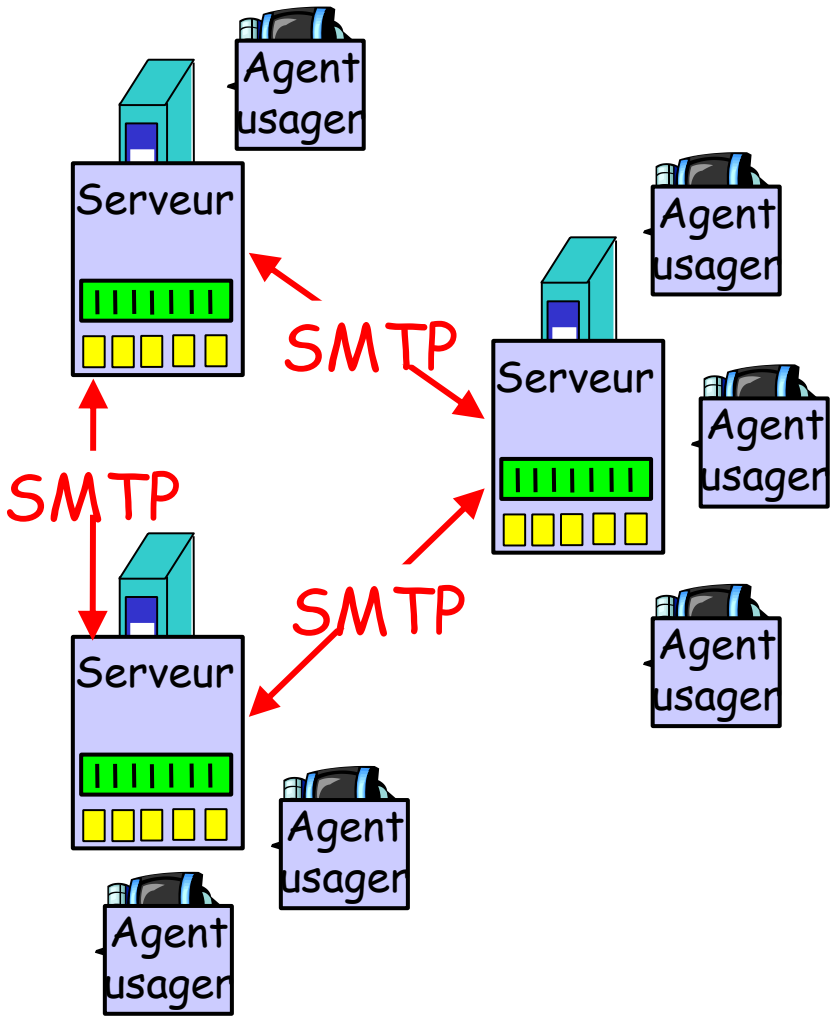
- Composer, outlook, eudora, mutt, elm
- composition, edition, lecture de messages
- Les messages en arrivage ou en partance sont stockés sur le serveur



Courriel : les serveurs

Serveurs

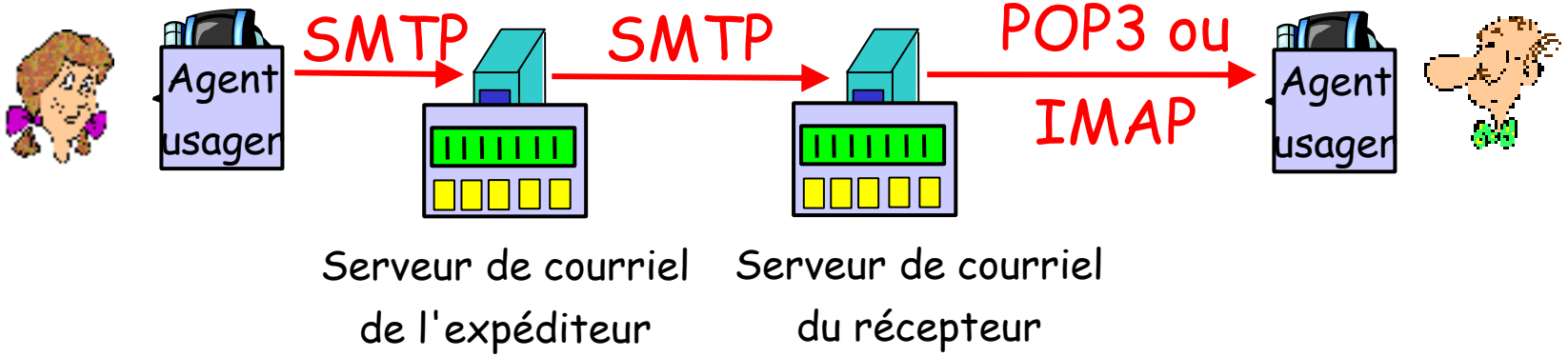
- Une boîte aux lettres contient les messages reçus pour l'usager
- Une file de messages contient les courriels en partance
- Le protocol smtp est utilisé entre les serveurs de courriel pour transmettre les messages
  - client: le serveur expéditeur
  - «serveur» : le serveur récepteur



Courriel: smtp [RFC 821]

- Utilise TCP pour transférer les messages du client au server, port 25, de manière robuste
- Transfert direct: du serveur d'expédition au serveur de réception
- Trois phases de transfert
  - Poignée de main (salutation)
  - Transfert de messages
  - Fermeture
- Interaction commande/réponse
  - commandes: texte ASCII
  - réponse: code de statut et phrase
- Les messages doivent être codés en ASCII 7-bit

Protocoles d'accès au courriel



- SMTP: livraison et entreposage sur le serveur du récepteur
- Protocole d'accès au courriel: récupération du serveur
  - POP: Post Office Protocol [RFC 1939]
    - Autorisation (agent <-->serveur) et chargement
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - Plus de features, donc de complexité
    - Manipulation de données sauvegardées sur le serveur
  - HTTP: Hotmail , Yahoo! Mail, etc.

DNS: Domain Name System

Personnes: plusieurs identificateurs:

- SSN, name, passport #

Hôtes et routeurs Internet:

- addresse IP (32 bit) utilisée pour les datagrammes
- un « nom », p.ex., gaia.cs.umass.edu utilisé en général

Q: comment faire la correspondance entre addresse IP et nom ?

Domain Name System:

- *Une base de donnée répartie,* implantée dans une hiérarchie de *serveurs de dénomination*
- *Un protocole de niveau application* entre hôtes, routeurs, serveurs de dénomination qui leur permet de *résoudre* des noms (traduction addresse/nom)
  - Note : fonction essentielle de l'Internet, implantée comme protocole de niveau application
  - Une complexité à la périphérie du réseau.

DNS serveurs de dénomination

Pourquoi ne pas centraliser DNS?

- Défaillances « single point of failure »
- Volume de trafic
- Base de donnée centralisée à distance
- Mises à jour

Ne peut pas grandir.

- Aucun serveur ne possède toutes les résolutions nom-adresse

Serveurs locaux :

- chaque ISP, compagnie possède un *serveur de nom local (par défaut)*
- Une requête DNS d'un hôte va d'abord au serveur local.

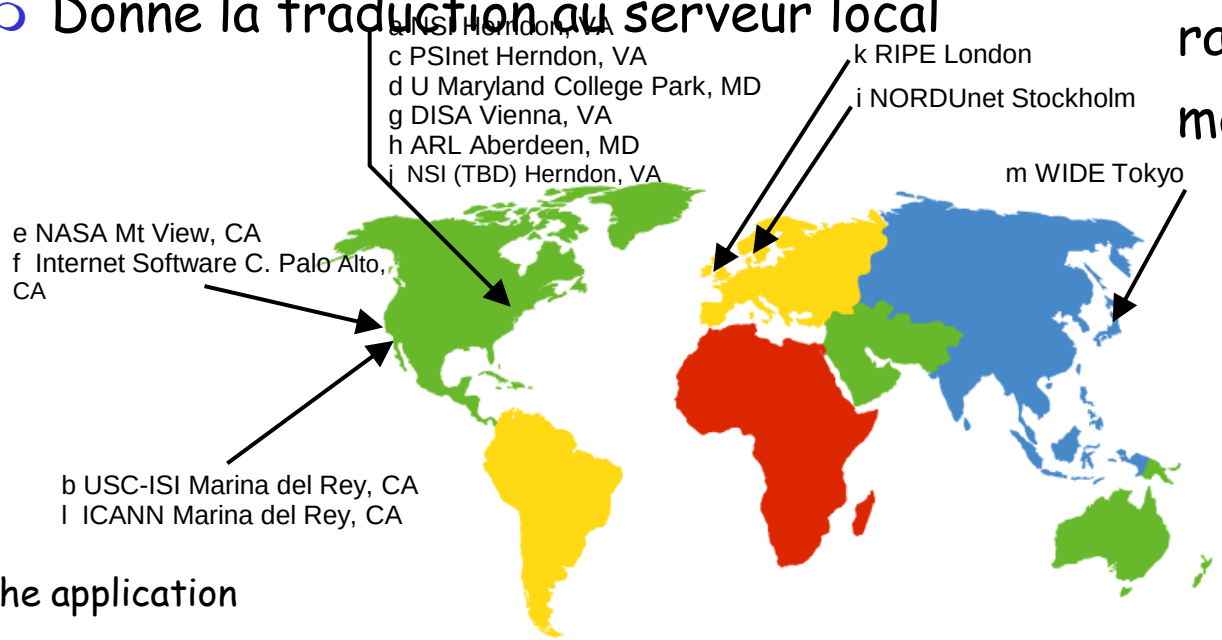
Serveur de nom faisant autorité :

- Pour un hôte: mémorise noms et adresse IP
- Peut effectuer des traductions nom-adresse pour le(s) nom(s) de cet hôte.

DNS: serveurs de nom racines

- Contactés par un serveur de local qui n'arrive pas à résoudre un nom
- Le serveur de nom racine:
  - Contacte le serveur faisant autorité si la traduction est inconnue
  - Acquiert la traduction
  - Donne la traduction au serveur local

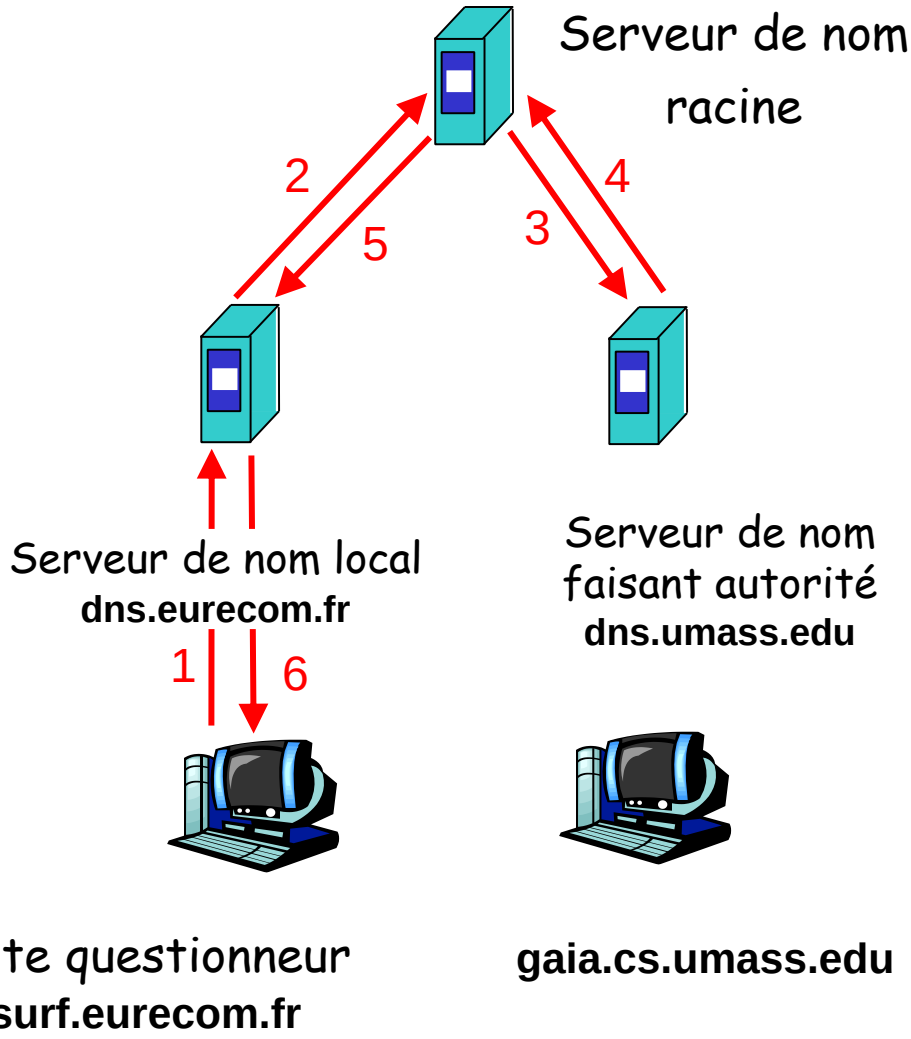
13 serveurs de nom racines à travers le monde



Exemple simple de DNS

L'hôte **surf.eurecom.fr** désire l'adresse IP de **gaia.cs.umass.edu**

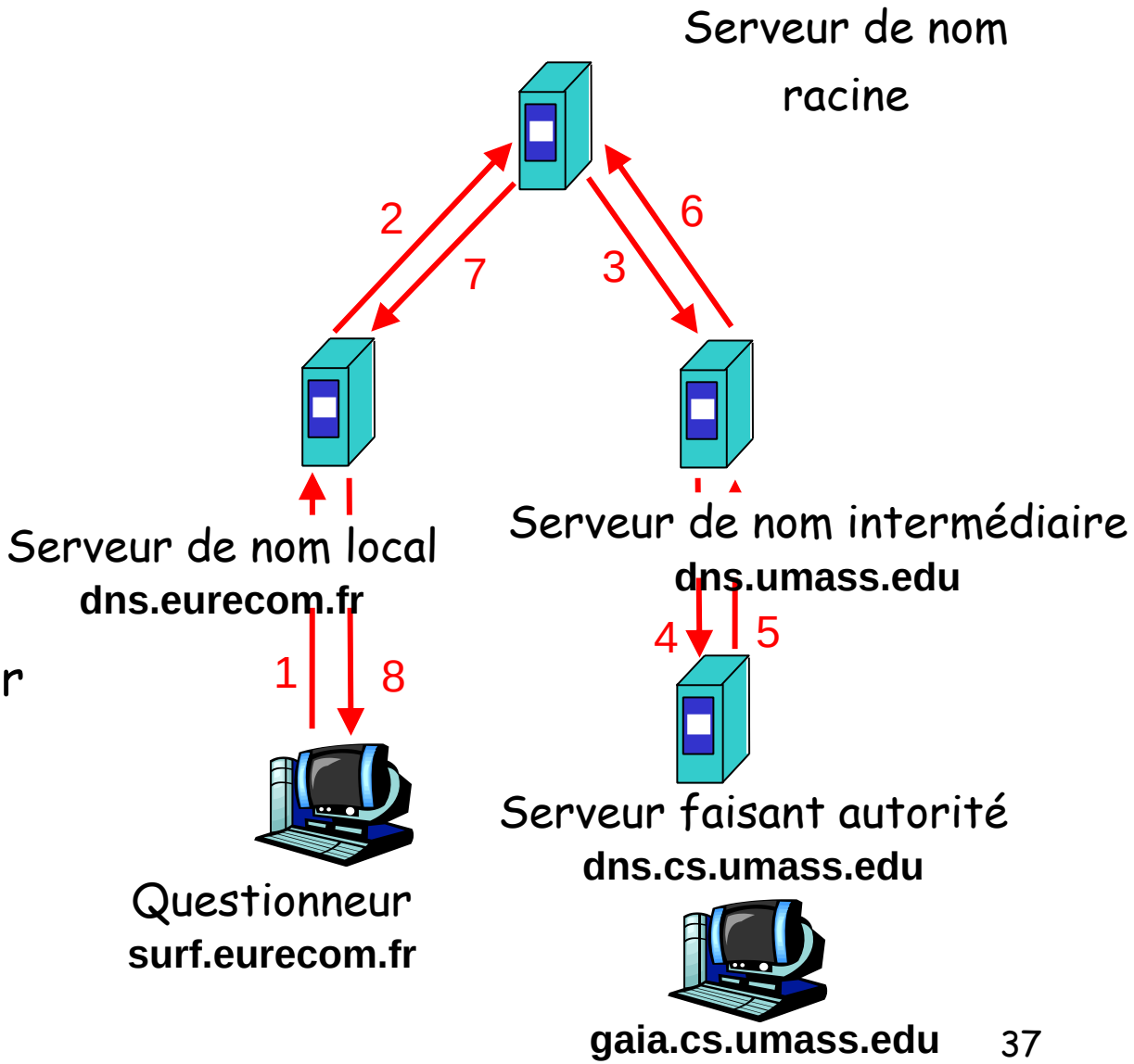
1. Il contacte son serveur DNS local, **dns.eurecom.fr**
2. **dns.eurecom.fr** contacte un serveur de nom racine, si nécessaire
3. Le serveur de nom racine contacte le serveur de nom faisant autorité, **dns.umass.edu**, si nécessaire



Exemple DNS

Serveur de nom racine:

- Peut ne pas connaître un serveur faisant autorité
- Peut connaître un *serveur de nom intermédiaire:* qui contacter pour trouver le serveur faisant autorité



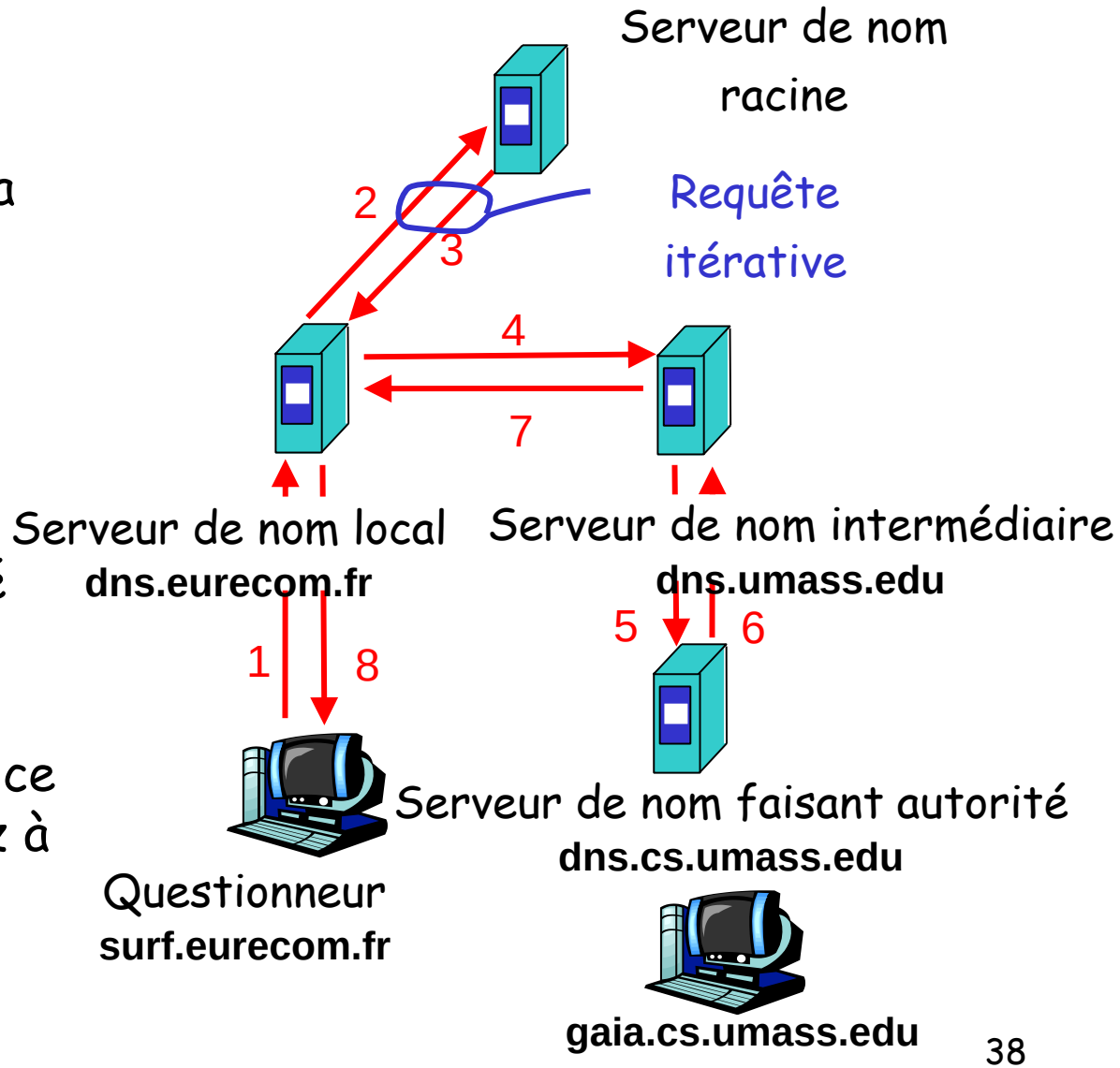
DNS: requêtes itératives

Requête récursive:

- Met le fardeau de la résolution sur le serveur contacté
- Charge excessive?

Requête itérative:

- Le serveur contacté répond avec le nom d'autres contacts
- « Je ne connais pas ce nom, mais demandez à ce serveur »



DNS: cache et mise à jour

- Tout serveur mémorise dans une cache une traduction nouvellement apprise
  - Ces traductions disparaissent après un certain temps.
- D'autres mécanismes sont à l'étude à l'IETF
  - RFC 2136
  - http://www.ietf.org/html.charters/dnsind-charter.html

Enregistrements DNS

DNS: une base de données répartie de « resource records (RR) »

Format RR : (**nom, valeur, type,ttl**)

- Type=A
  - **nom** est le nom de l'hôte
  - **valeur** est l'addresse IP
- Type=NS
  - **nom** est le domaine (p.ex. foo.com)
  - **valeur** est l'addresse IP du serveur faisant autorité dans ce domaine
- Type=CNAME
  - **name** est un alias pour un nom de référence

    www.ibm.com est servereast.backup2.ibm.com
  - **valeur** est le nom de référence
- Type=MX
  - **valeur** est le nom du serveur de courriel associé à **nom**