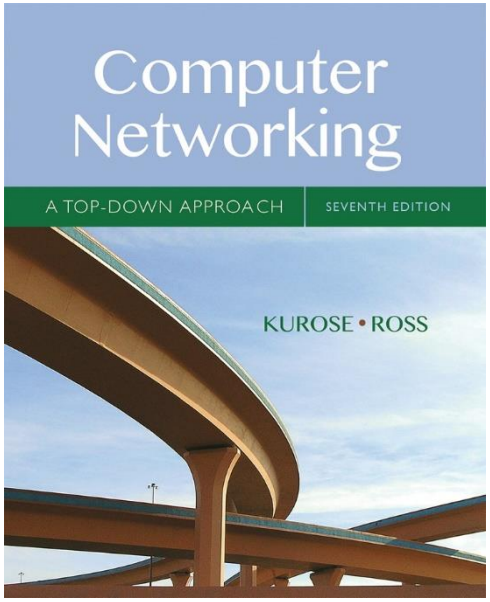
Chapitre 11
La couche application



Computer Networking: A Top Down Approach
7ème édition
Jim Kurose, Keith Ross
Addison-Wesley
2017

La couche application: plan

2.1 principes des applications réseaux

2.2 Web and HTTP

2.3 FTP

2.4 courriel

- SMTP, POP3, IMAP

2.5 DNS

2.6 applications P2P

La couche application

Objectifs:

- Aspects des protocoles de la couche application:
  - paradigme client-serveur
  - paradigme pair-à-pair
  - services fournis par la couche transport
- protocoles populaires
  - HTTP
  - FTP
  - SMTP / POP3 / IMAP
  - DNS

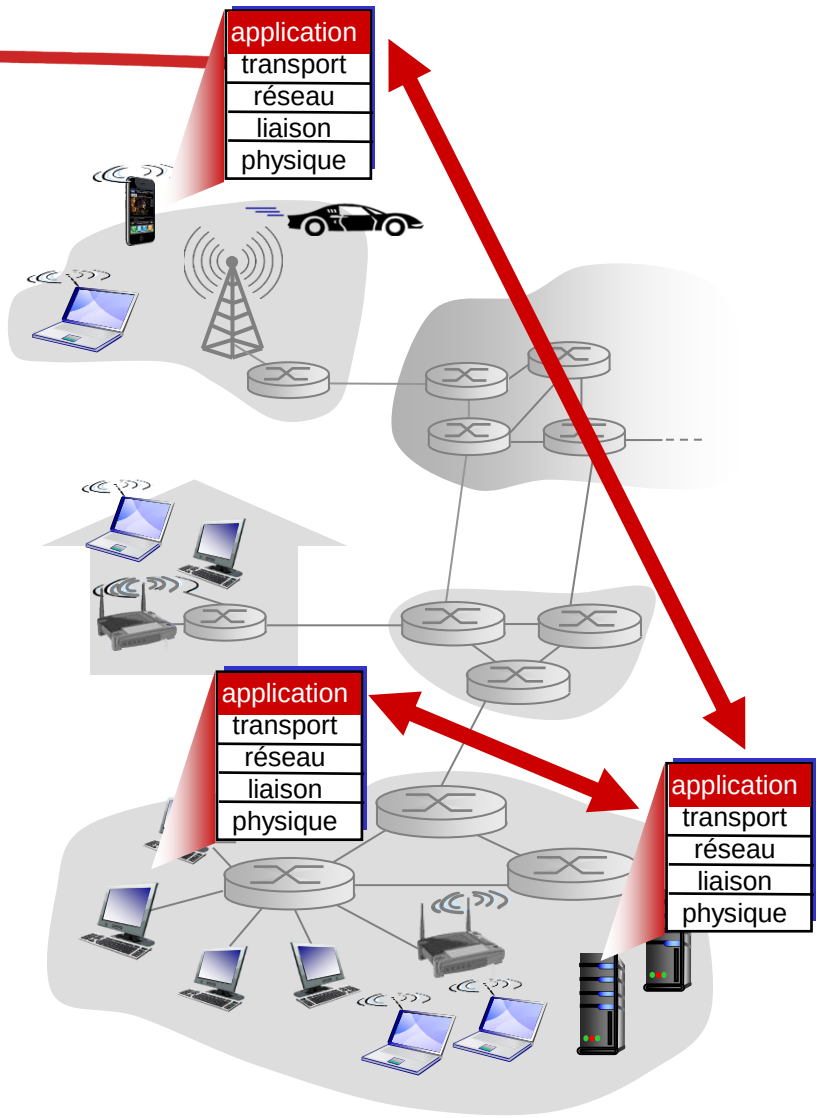
Quelques applications réseaux

- courriel
- web
- messagerie
- connexion à distance
- partage P2P
- jeux multijoueur
- diffusion en flux (streaming)
- voix sur IP (ex., Skype)
- visioconférence
- réseaux sociaux
- recherche
- …
- …

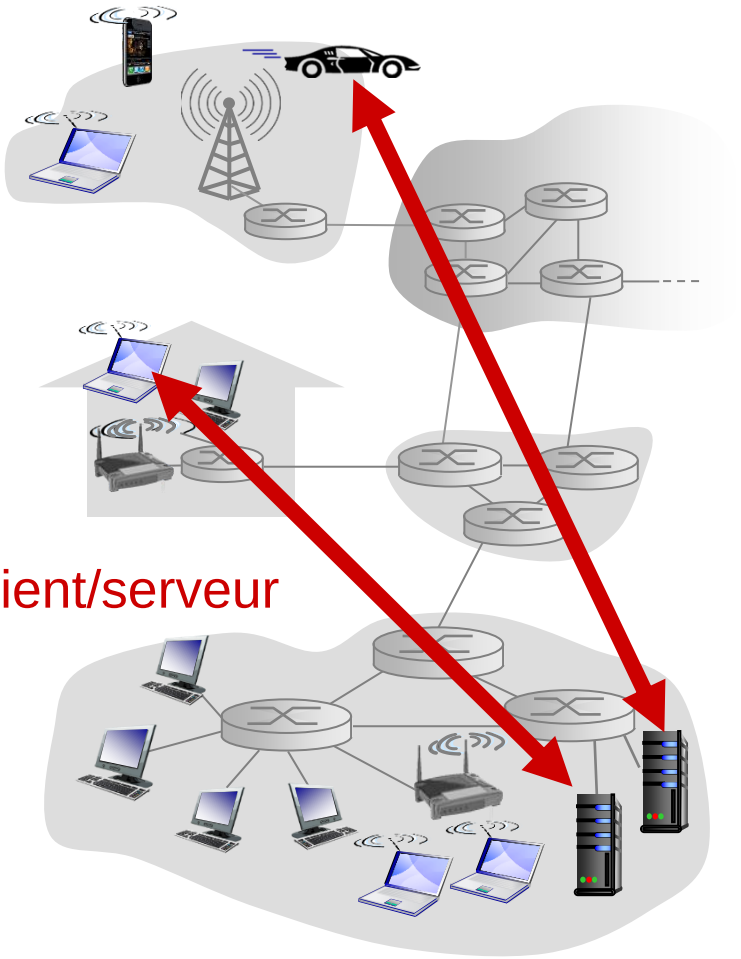
Architectures

architectures des applications:

- client-serveur
- pair à pair (P2P)



Architecture client-serveur



serveur:

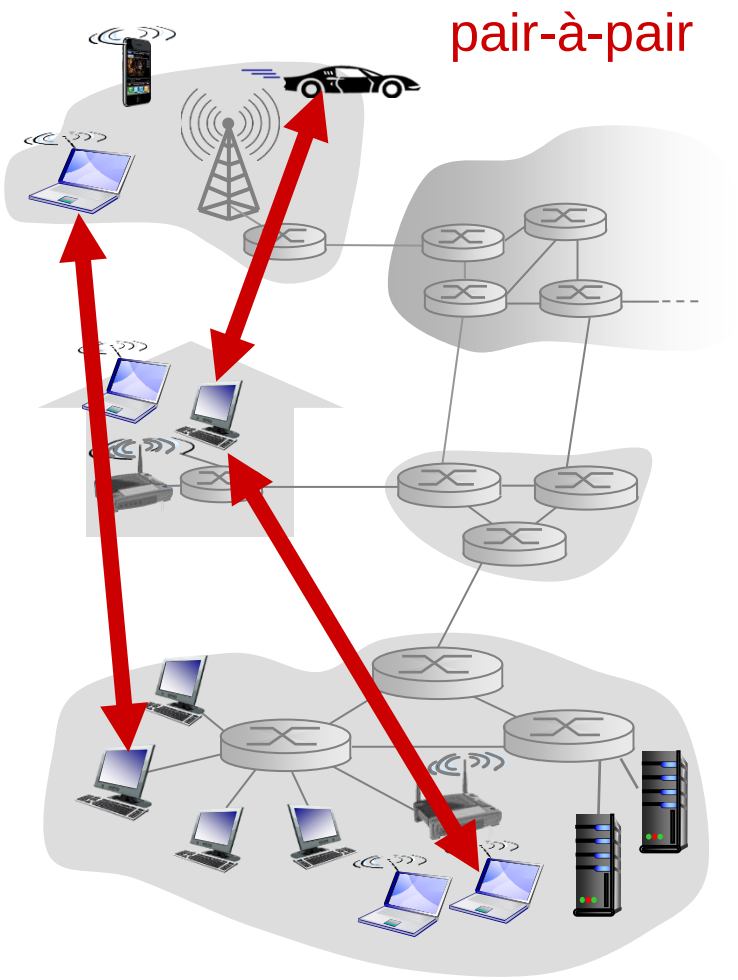
- toujours en marche
- adresse IP permanente

clients:

- communiquent avec un serveur
- peuvent se déconnecter
- adresse IP dynamique
- ne peuvent communiquer directement

Architecture P2P

- *pas besoin d'un serveur*
- les terminaux communiquent directement
- chaque pair demande un service (requête) et offre un service (réponse)
  - *auto-scalabilité (mise en échelle)*
- les pairs apparaissent et disparaissent en changeant leurs adresses IP
  - gestion complexe



Communication entre processus

processus: programme qui tourne dans un hôte

- dans le même hôte, 2 processus communiquent par une communication inter-processus (définie OS)
- processus sur des hôtes différents communiquent par échange de messages

clients, serveurs

processus client : processus qui initie la communication

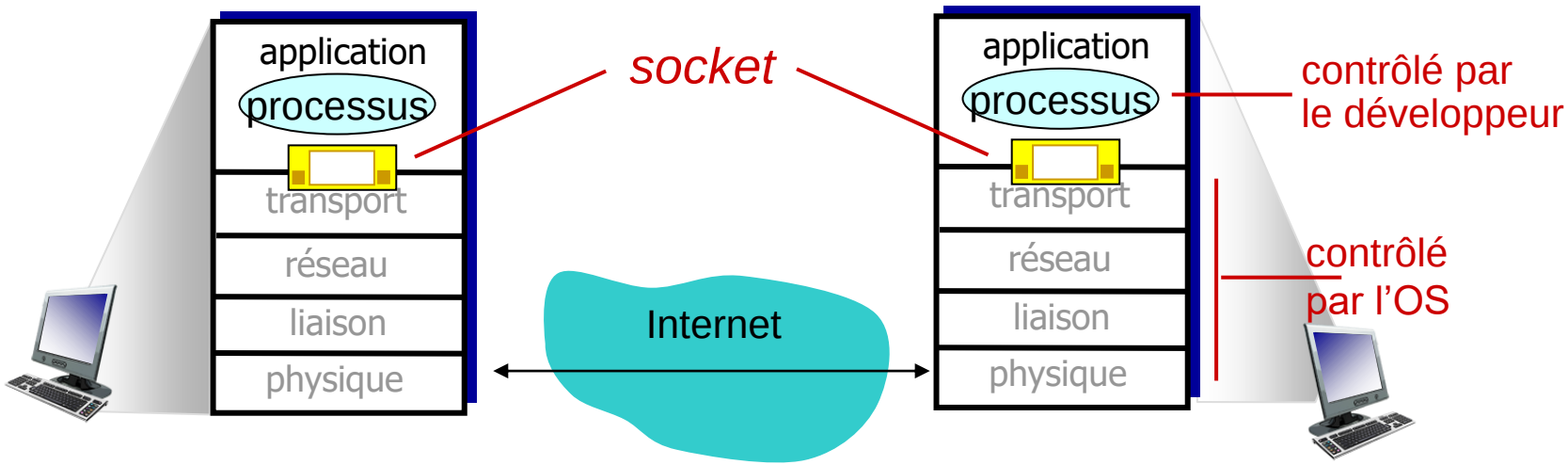
processus serveur : processus qui attends d'être contacté

- les applications dans P2P architectures ont des processus clients et serveurs

Sockets

- un processus envoie/reçoit des messages par sa socket

- analogie: socket = porte
  - la couche de transport : livraison des messages de porte en porte



Adressage des processus

- pour recevoir des messages, un processus a besoin d'un *identificateur*
- chaque hôte possède une adresse IP (sur 32 bits)
- *Q:* Est-ce que l'adresse IP est suffisante comme identificateur?
  - *A:* non, *plusieurs* processus sur le même hôte

- *identificateur* = IP adresse + numéros de port associés au processus.
- exemples de num. de port:
  - HTTP (serveur): 80
  - courriel (serveur): 25

protocole de couche application

- types des messages échangés,
  - ex., requête, réponse
- syntaxe des messages:
  - les champs et leurs délimitations
- sémantique des messages
  - signification des champs
- règles
  - quand et comment envoyer/recevoir des messages

protocoles libres:

- définis dans RFCs
- interopérabilité
- ex., HTTP, SMTP

protocoles propriétaires:

- ex., Skype

Quel service de transport?

fiabilité du transfert

- des apps (ex., transfert de fichiers, transactions) requièrent un transfert100% fiable
- d'autres apps (ex. audio) peuvent tolérer de la perte

délai

- des apps (ex., téléphonie IP, jeux interactifs) requièrent un délai limité

débit

- des apps (ex., multimédia) requiert un débit minimal
- d'autres apps ("élastiques") utilisent le débit disponible

sécurité

- chiffrement, disponibilité, …

protocoles de transport: deux services

service TCP:

- *transport fiable* entre les processus
- *contrôle de flux:* l'émetteur ne sature pas le récepteur
- *contrôle de congestion :* prévenir l'émetteur quand le réseau est surchargé
- *n'assure pas:* délai, débit, sécurité
- *orienté connexion*

service UDP:

- *n'assure pas:* fiabilité, contrôle de flux, contrôle de congestion, délai, débit, sécurité

Q: alors pourquoi UDP existe?

Apps Internet : protocoles transport/application

application	protocole de la couche application	protocole de la couche transport
courriel	SMTP [RFC 2821]	TCP
accès distant	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
transfert de fichiers	FTP [RFC 959]	TCP
streaming multimédia	HTTP (ex., YouTube), RTP [RFC 1889]	TCP ou UDP
téléphonie IP	SIP, RTP, propriétaire (ex., Skype)	TCP ou UDP

Chapitre 2: plan

2.1 principes des applications réseaux

2.2 Web and HTTP

2.3 FTP

2.4 courriel

- SMTP, POP3, IMAP

2.5 DNS

2.6 applications P2P

Web et HTTP

- *une page web* consiste en des *objets*
- objet = fichier HTML, image JPEG, applet Java, fichier audio,…
- une page web contient *un fichier HTML de base* qui inclue *plusieurs objets*
- chaque objet est adressable par un *URL,* ex.

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

nom de l'hôte | chemin

HTTP: vue d'ensemble

HTTP: hypertext transfer protocol

- modèle client/serveur
  - *client:* navigateur
  - *serveur:* serveur Web

HTTP: vue d'ensemble

utilise TCP:

- le client initie une connexion TCP (crée socket) au serveur, port 80
- serveur accepte la connexion TCP
- échange de messages HTTP
- fermeture de la connexion TCP

HTTP est "sans état"

- serveur ne garde aucune information sur les anciennes requêtes

note

Les protocoles "avec état" sont plus complexes!

connexions HTTP

HTTP non-persistant

- au plus un objet par connexion TCP
  - connexion se ferme après le transfert

HTTP persistant

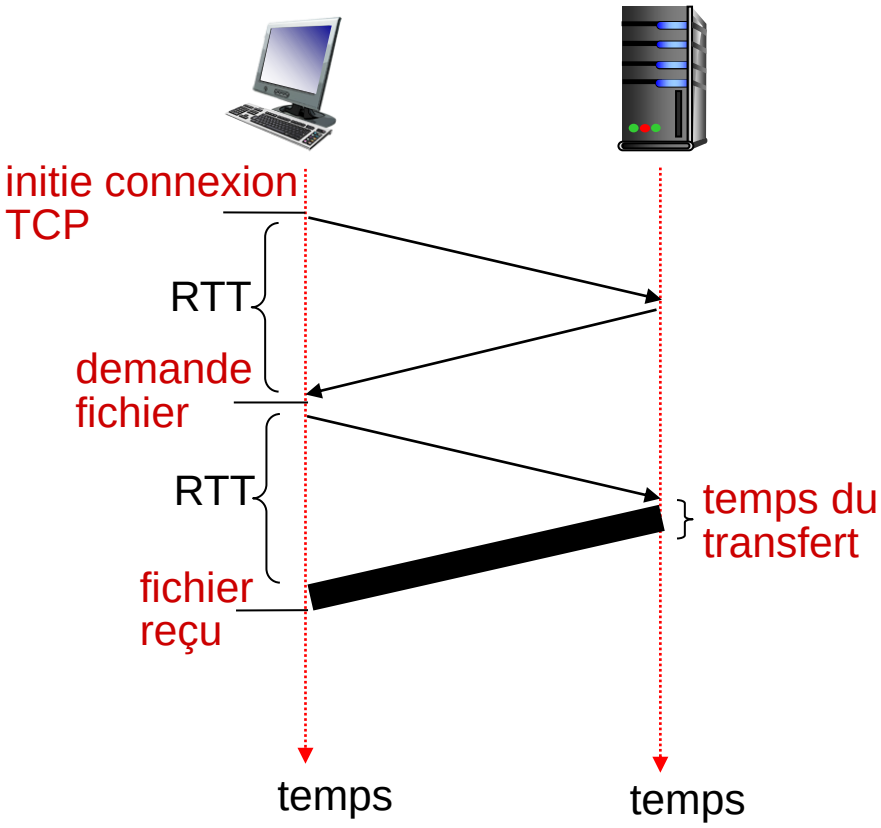
- une seule connexion TCP peut permettre le transfert de plusieurs objets

HTTP non-persistant: temps de réponse

RTT (définition): temps nécessaire pour un petit paquet pour effectuer un aller-retour

temps de réponse HTTP :

= 2RTT+ temps du transfert



HTTP persistant

inconvénients non-persistant

- 2 RTTs par objet
- surcharge causée par le nombre de connexions TCP

HTTP persistant:

- Presque une seule RTT pour tous les objets

requête HTTP

- deux types de messages HTTP: *requête, réponse*
- *requête* HTTP :
  - en ASCII

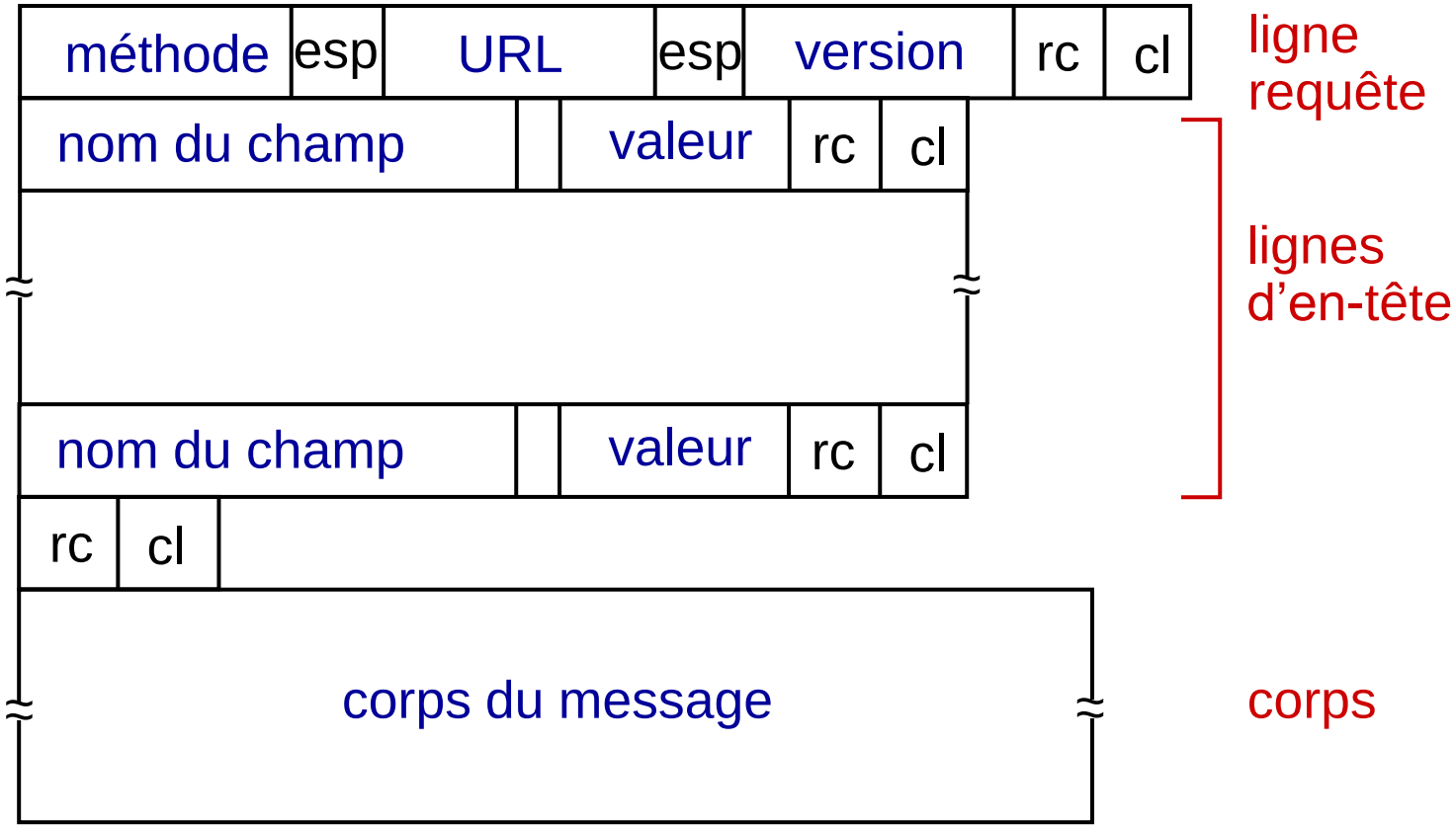
ligne de demande (commandes GET, POST, HEAD)

lignes d'en-tête

indique la fin de l'en-tête

```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

requête HTTP: format général



Uploader des données de formulaire

méthode POST :

- les pages web incluent souvent des formulaires
- les données entrées sont uploadées au serveur en utilisant le corps du message HTTP

méthode URL:

- utilise la méthode GET
- les données entrées sont uploadées au serveur en utilisant l'URL:

```
www.somesite.com/animalsearch?monkeys&banana
```

Types de méthodes

HTTP/1.0:

- GET
- POST
- HEAD
  - demande au serveur de ne pas envoyer l'objet demandé dans la réponse

HTTP/1.1:

- GET, POST, HEAD
- PUT
  - uploader un fichier au serveur
- DELETE
  - supprimer un fichier du serveur

Réponse HTTP

ligne d'état

lignes d'en-tête

données, ex., fichier HTML demandé

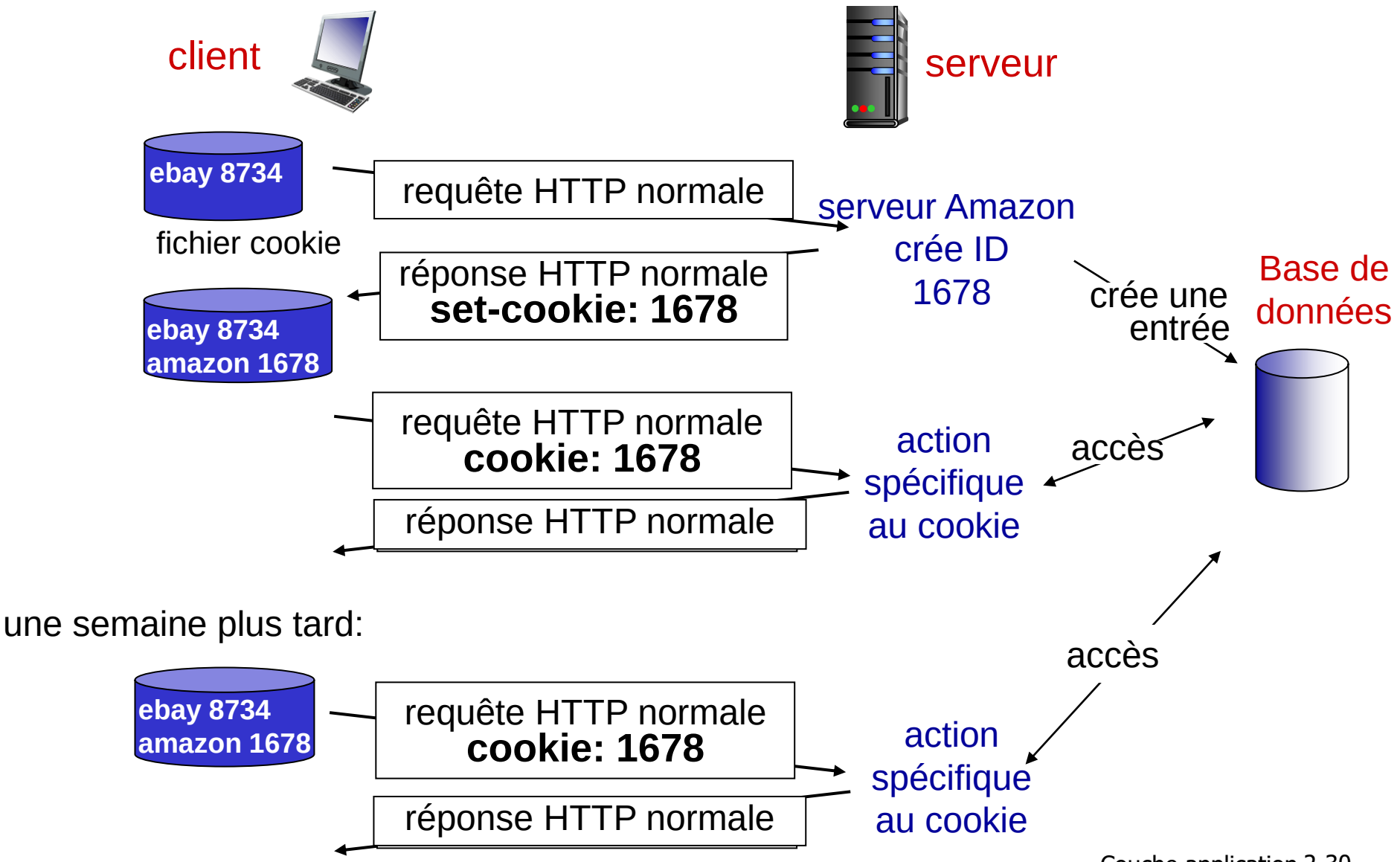
```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02
   GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-
   1\r\n
\r\n
data data data data data ...
```

Réponse HTTP: codes d'état

- le code d'état apparait à la première ligne de la réponse
- quelques codes:

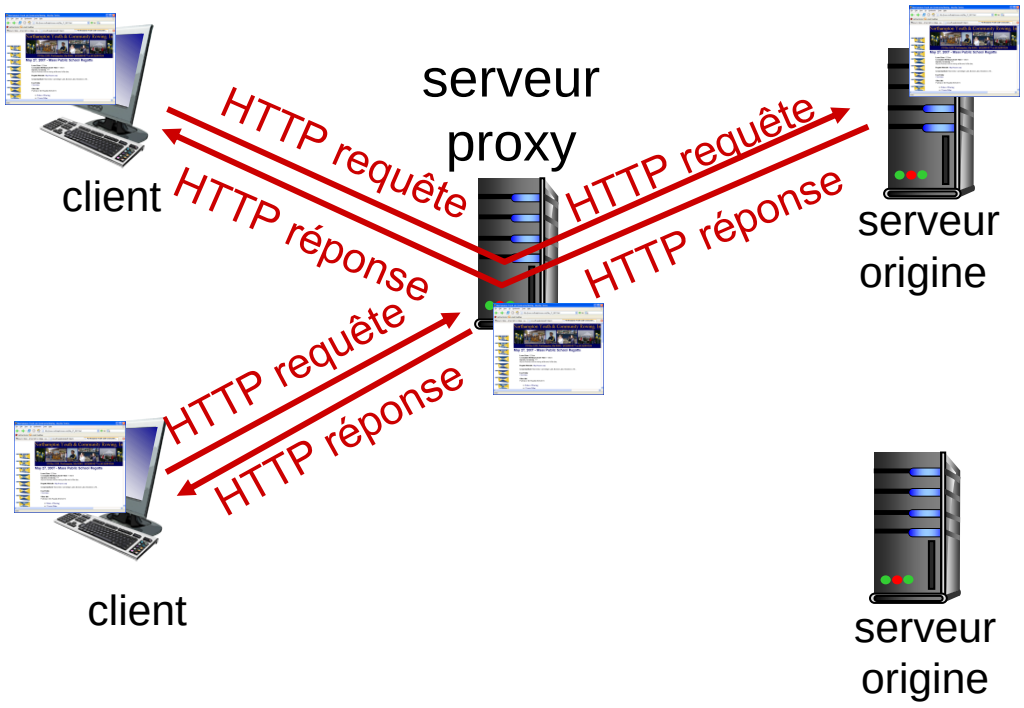
```
200 OK
301 Moved Permanently
400 Bad Request
404 Not Found
505 HTTP Version Not Supported
...
```

Cookies: garder l'"état" (ex.)



Serveur web cache (serveur proxy)

but: répondre à la requête du client à la place du serveur d'origine



- navigateur doit être configuré

Serveur cache (exemple)

hypothèses:

- taille moyenne par requête: 1 Mb
- taux moyen des requêtes :15/sec
- RTT entre le routeur institutionnel et un serveur: 2 sec

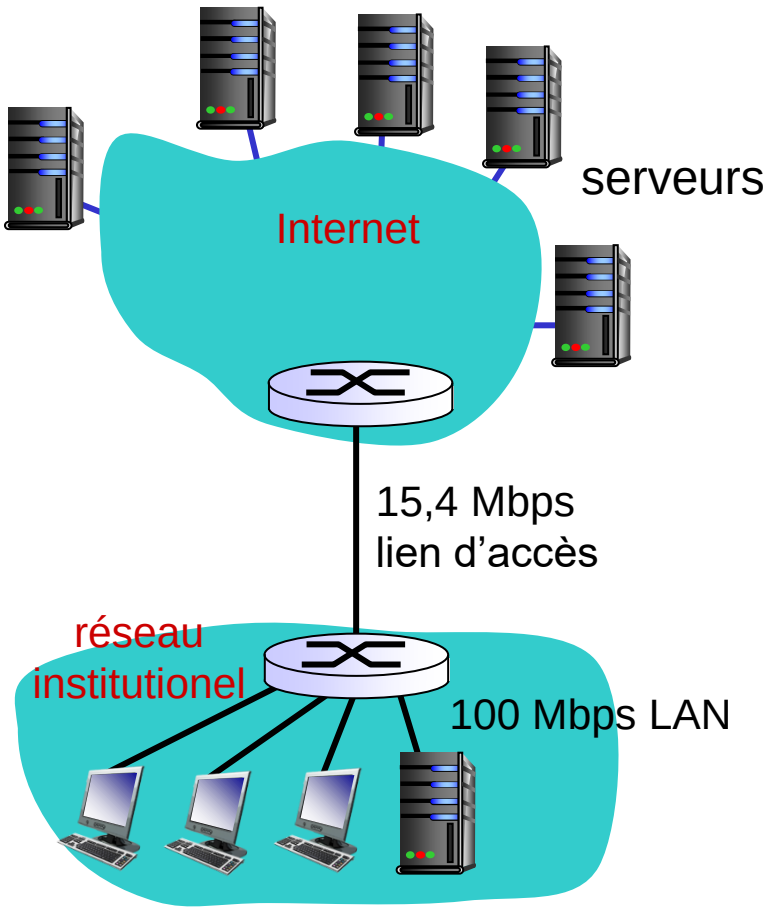
conséquences:

- utilisation du LAN : 15%
- utilisation du lien d'accès = 99%

problème!

- délai total = délai Internet + délai d'accès + délai LAN

  = 2 sec + minutes + msecs



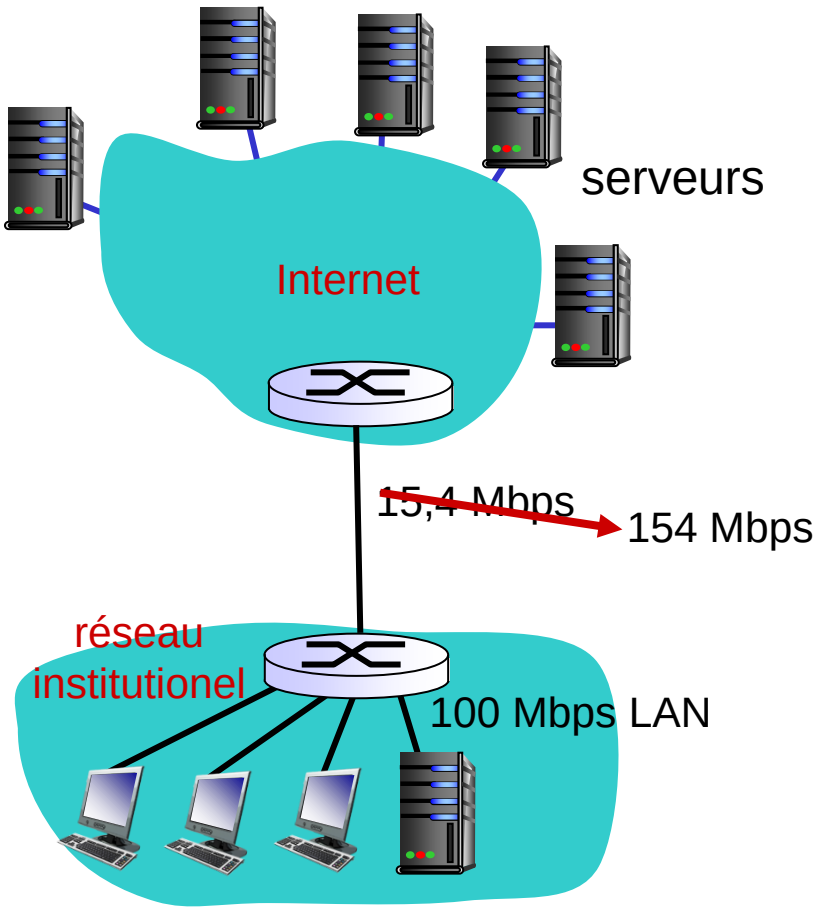
Serveur cache (exemple)

hypothèses:

- taille moyenne de l'objet: 1 Mbits
- taux moyen des requêtes :15/sec
- RTT entre le routeur institutionnel et un serveur: 2 sec
- débit d'accès: 15,4 Mbps → 154 Mbps

conséquences:

- utilisation LAN : 15%
- utilisation lien d'accès = 99% → 9.9%
- délai total = délai Internet + délai d'accès + délai LAN
  = 2 sec + minutes + msecs → msecs

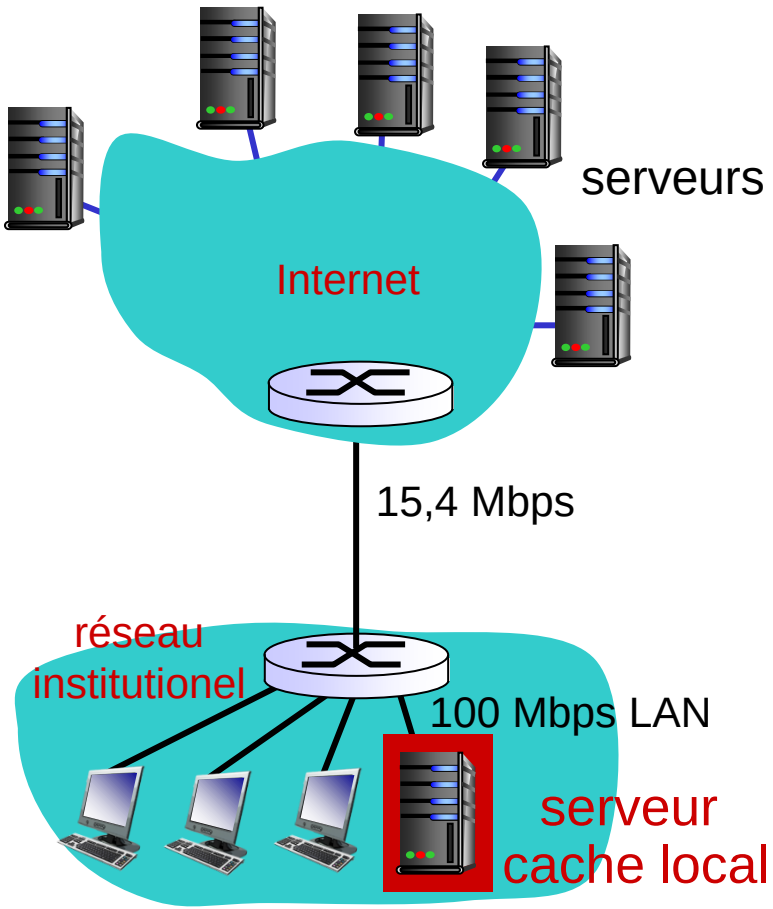


Coût: améliorer le débit du lien accès est coûteux

Serveur cache (solution)

hypothèses:

- taille moyenne de l'objet: 1 Mbits
- taux moyen des requêtes :15/sec
- RTT entre le routeur institutionnel et un serveur: 2 sec
- débit du lien d'accès: 15,4 Mbps



conséquences:

- utilisation LAN : 15%
- utilisation lien d'accès = ?
- délai total = ?

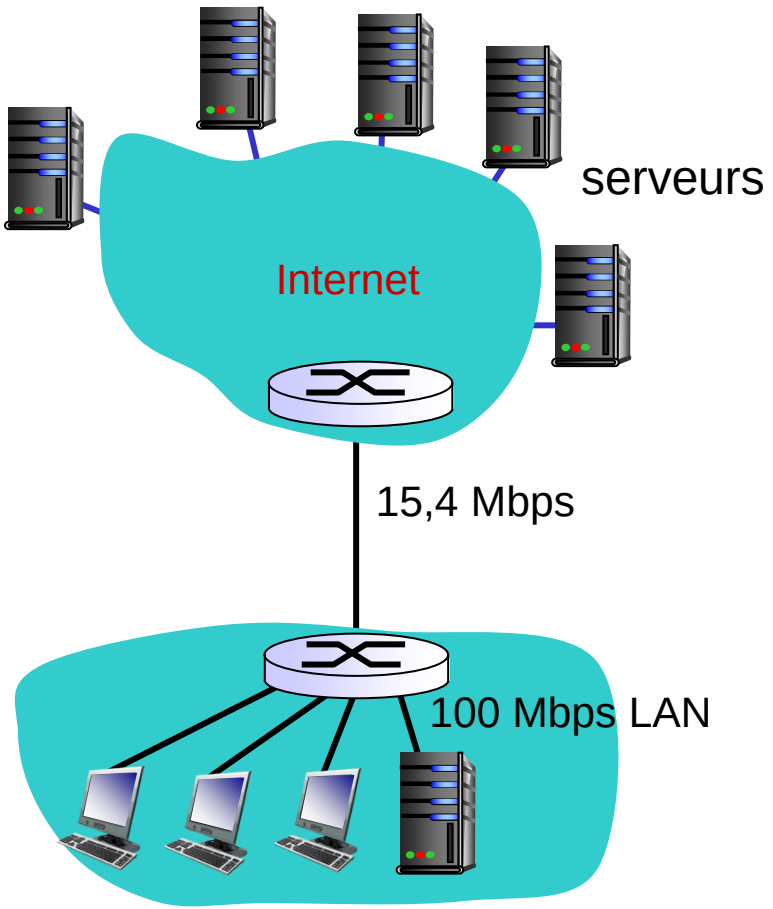
Comment effectuer ces calculs?

Coût: un serveur cache est beaucoup moins cher

Serveur cache (solution)

Calcul de l'utilisation et du délai au niveau d'accès:

- supposons que
  - 40% requêtes traitées au cache, 60% requêtes traitées au serveur
- utilisation du lien d'accès:
  - 60% des requêtes utilisent le lien
- utilisation passe de 99% à 59%



- délai total
  - = 0.6 * (délai des serveurs d'origine) +0.4 * (délai des serveurs cache)
  - = 0.6 (2.01) + 0.4 (~msecs)
  - = ~ 1.2 secs
  - plus intéressant que la première solution

Chapitre 2: plan

2.1 principes des applications réseaux

2.2 Web and HTTP

2.3 FTP

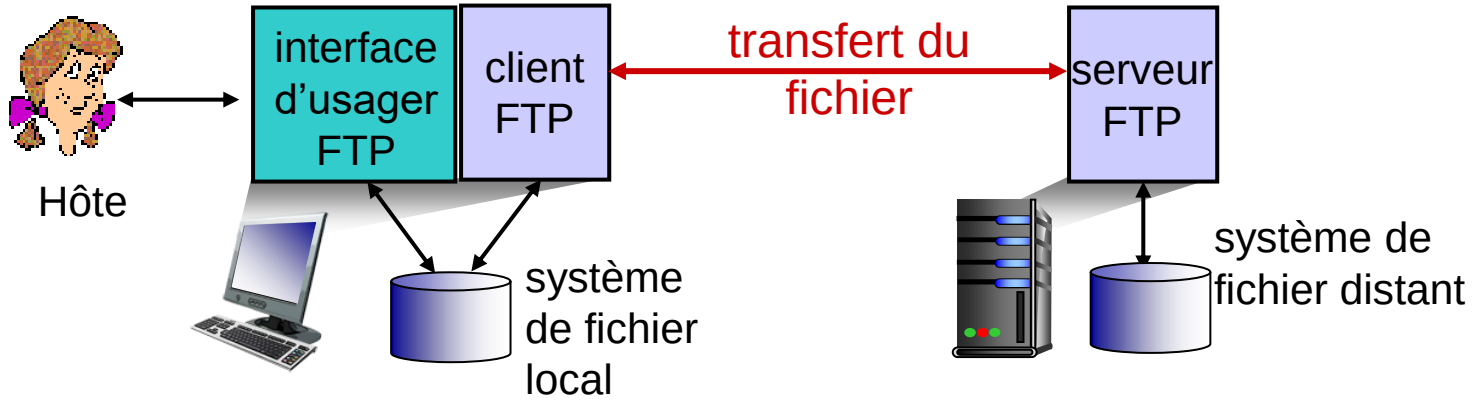
2.4 courriel

- SMTP, POP3, IMAP

2.5 DNS

2.6 applications P2P

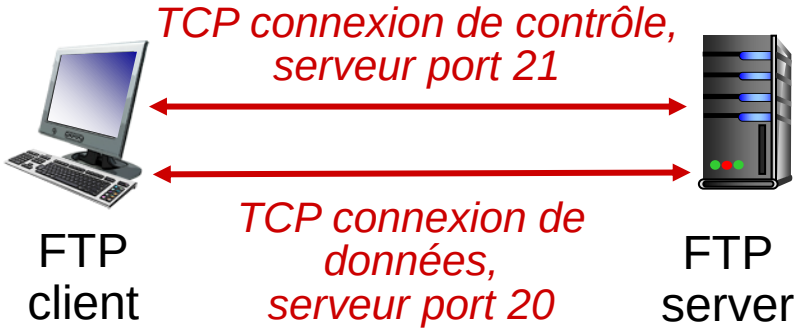
FTP: file transfer protocol



- transférer un fichier vers/depuis un hôte distant
- modèle client/serveur
- ftp: RFC 959

FTP: connexions données/contrôle

- Client contacte le serveur au port 21, par TCP
- Client doit être autorisé
- Client envoie des commandes



- Serveur ouvre une 2*ième* connexion TCP pour les données
- après le transfert ferme la connexion

- connexion de contrôle: *« hors bande »*
- Protocole *« avec état »*

FTP commandes, réponses

commandes:

- envoyées en ASCII sur la connexion de contrôle
- ex. **`USER, PASS, LIST, RETR, STOR`**

codes de réponse

- similaire à HTTP
- ex. **`331 Username OK, password required`**

Chapitre 2: plan

2.1 principes des applications réseaux

2.2 Web and HTTP

2.3 FTP

2.4 courriel

- SMTP, POP3, IMAP

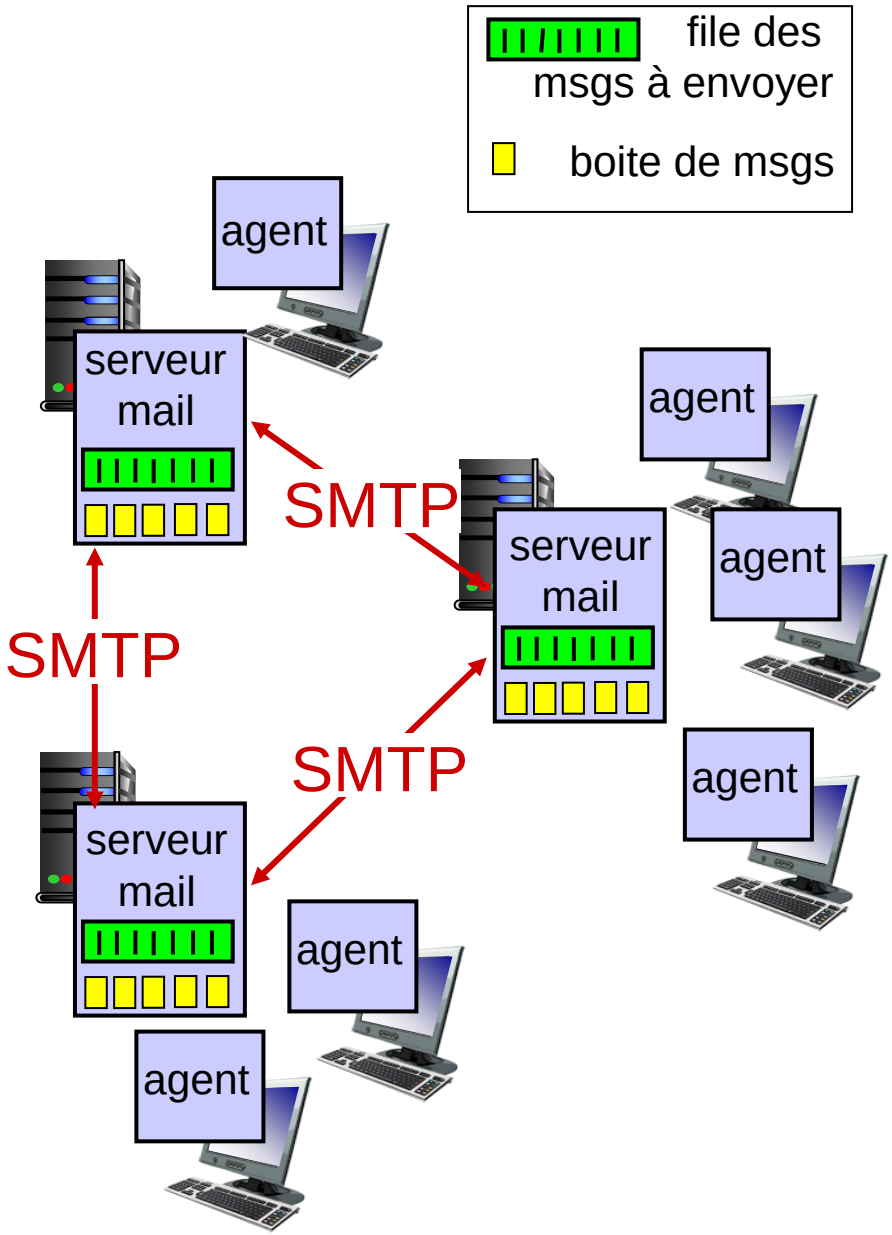
2.5 DNS

2.6 applications P2P

Courriel

Trois composantes:

- agents utilisateurs
- serveurs de messagerie
- simple mail transfer protocol: SMTP

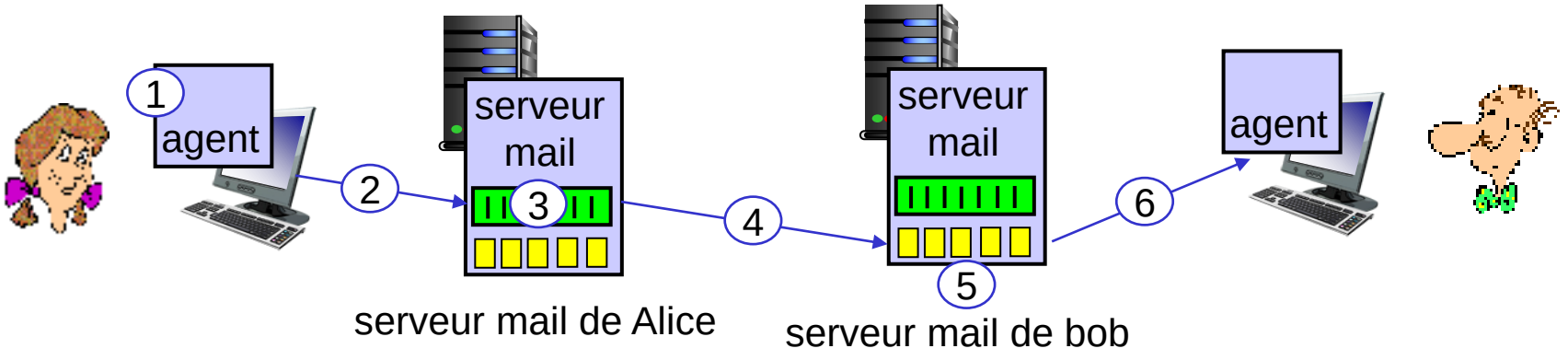


Courriel : SMTP [RFC 2821]

- utilise TCP : port 25
- transfert direct : serveur émetteur au serveur récepteur
- trois phases
  - poignée de mains (salutations)
  - transfert des messages
  - fermeture
- interaction par commande/réponse (HTTP, FTP)
  - commandes: ASCII
  - réponse: codes d'état

Scénario: Alice envoie message à Bob

1) Alice utilise AU pour composer le message “to” `bob@universite.ca`
2) le AU d’Alice envoie le message à son serveur mail; qui place le message dans la file
3) côté client de SMTP ouvre une connexion TCP avec le serveur mail de Bob
4) le client SMTP envoie le message d’Alice sur la connexion TCP
5) Le serveur mail de Bob met le message dans la boite de Bob
6) Bob utilise son AU pour lire le msg



SMTP: exemple

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

SMTP

- SMTP utilise une connexion persistante
- message SMTP doit être en ASCII 7-bit

comparaison avec HTTP:

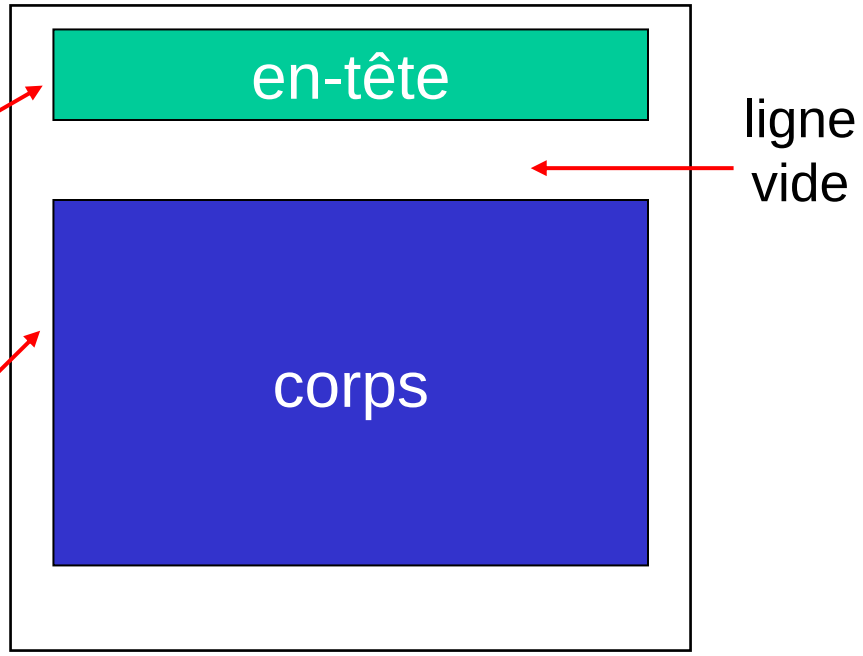
- HTTP: pull (retirer)
- SMTP: push (transmettre)
- les deux utilisent des commande/réponse ASCII, codes d'état
- HTTP: chaque message encapsule un objet
- SMTP: plusieurs objets dans un seul msg

Courriel: format du message

RFC 822:

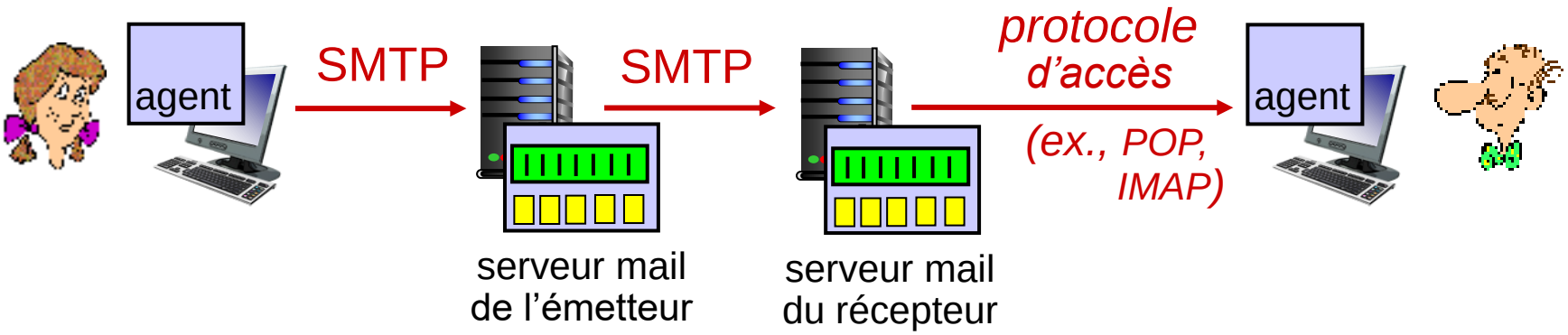
- lignes d'en-tête, ex.,
  - To:
  - From:
  - Subject:

  différent des commandes!



- Corps: le "message"
  - ASCII seulement

Protocoles d'accès au courriel



- SMTP: délivrer le message au serveur mail du récepteur
- protocole d'accès: récupérer le message depuis le serveur
  - POP: Post Office Protocol [RFC 1939]
  - IMAP: Internet Mail Access Protocol [RFC 1730]
  - HTTP

Protocole POP3

étape d'authentification

- commandes client :
  - **`user`**: nom d'utilisateur
  - **`pass`**: mot de passe
- réponses server
  - **`+OK`**
  - **`-ERR`**

étape de transaction, client:

- **`list`**: lister les numéros des messages
- **`retr`**: récupérer le message par son numéro
- **`dele`**: supprimer
- **`quit`**

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

POP3 et IMAP

POP3 (suite)

- l'exemple utilise le mode "télécharger et supprimer"
- POP3 "télécharger et conserver"
- POP3 est sans état (entre les sessions)

IMAP

- garde les messages au serveur
- organisation dans des dossiers
- IMAP est avec état