Web & DNS

UE LU3IN033 Réseaux
2020-2021

Prométhée Spathis
promethee.spathis@sorbonne-universite.fr

Plan du cours

- HTTP Hypertext Transfer Protocol
  - Les ingrédients du Web
    - les URL, le langage HTML et le protocole HTTP
  - Le protocole HTTP
    - le format des requêtes et des réponses HTTP
  - Les interactions de HTTP avec TCP et DNS
  - Les cookies
  - Le caching et les proxys Web
- DNS Dynamic Naming System
  - Nommage et indirection
  - Base de données répartie
  - Organisation hiérarchique des serveurs DNS
  - Le caching et les proxys DNS

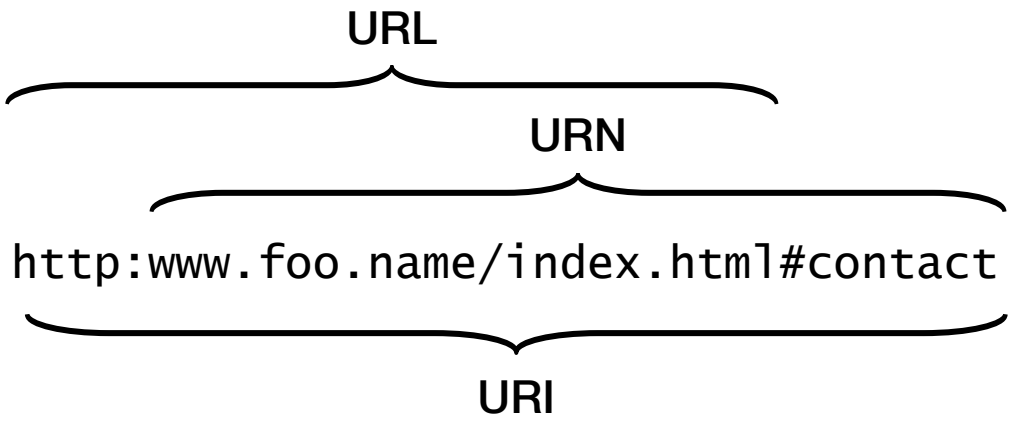
Les 3 composants du Web

- URL : Uniform Resource Locator
  - Localisation et identification d'une ressource web (un objet)

    http://www.foo.name/coolpic.gif
    - nom du serveur (www.foo.name) et de la ressource (coolpic.gif)
    - http : méthode pour obtenir la ressource
- HTML : HyperText Markup Language
  - Langage de balisage pour concevoir les pages web
  - Informations textuelles, inclusion d'images, ajout de hyperliens, …
  - Interprété par un navigateur pour afficher une page
- HTTP : HyperText Transfer Protocol
  - Protocole de couche 7 utilisé entre un navigateur et un serveur Web
  - Un client envoie une requête, le serveur renvoie un objet

HTML & URL

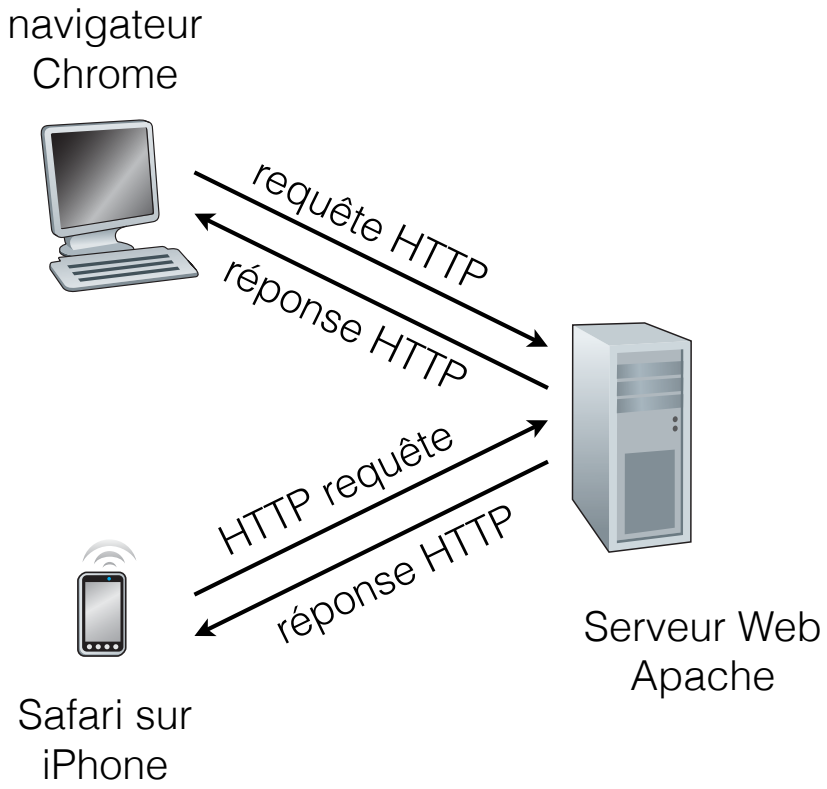
- Une page Web est composé de plusieurs objets :
  - le document de base (le fichier HTML) qui peur référencer …
  - … d'autres objets : des images jpeg, des fichiers audio, etc.
    - <img src="http://www.foo.name/coolpic.gif">
- Chaque objet est référencé par une URL :



Aperçu de HTTP

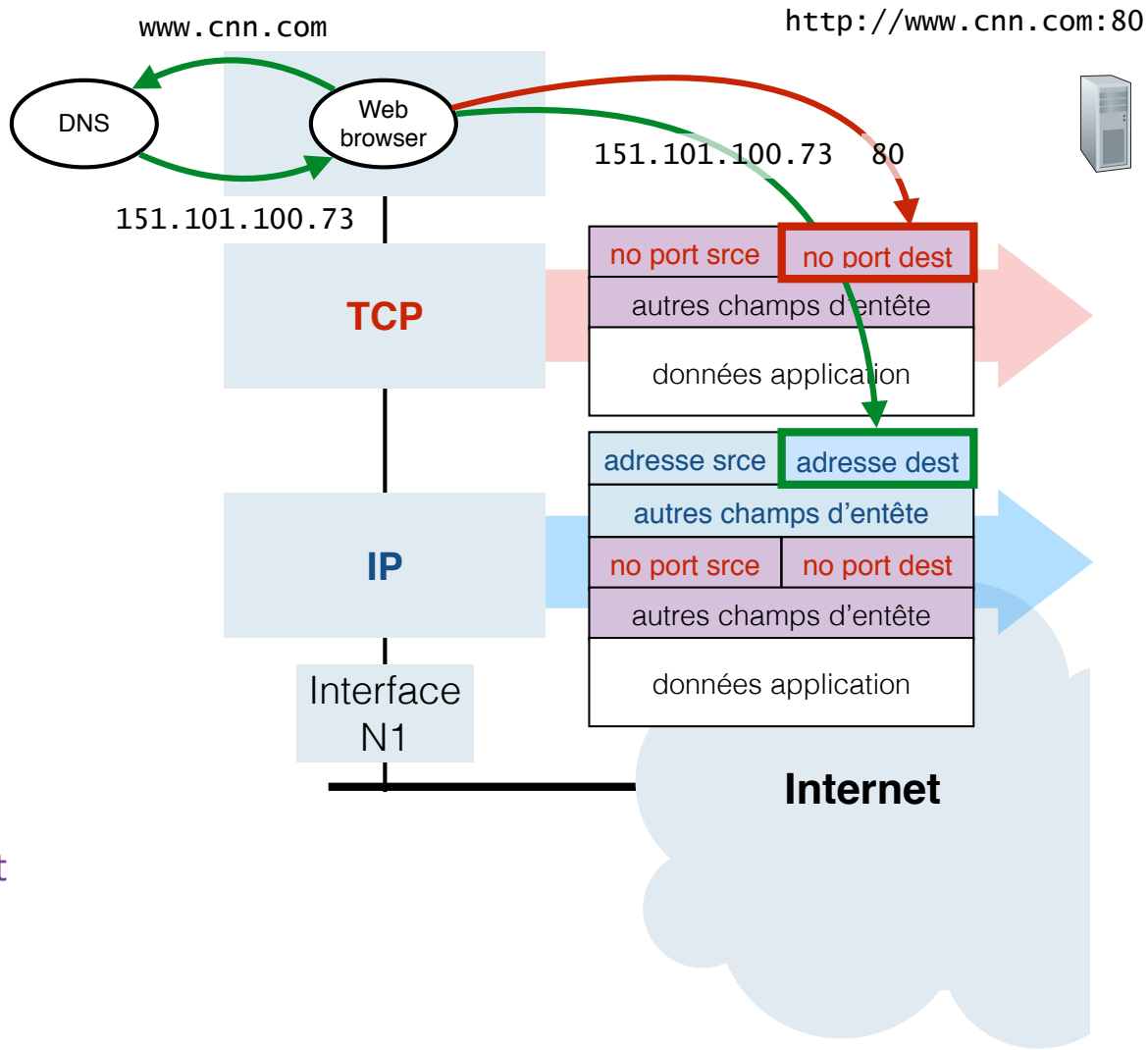
HTTP: HyperText Transfer Protocol

- Protocole de couche 7 utilisé par le Web
  - HTTP/1.0 : RFC 1945
  - HTTP/1.1 : RFC 2068
- Modèle Client/Serveur :
  - Client : le navigateur demande, reçoit et affiche des objets Web
  - Serveur: le serveur web envoie un objet en réponse à une requête de client
- En mode non connecté (pas d'état)
  - Utilisation de connexions TCP pour fiabiliser l'envoi des objets
- HTTP/2 (RFC 7540)
  - en mode connecté



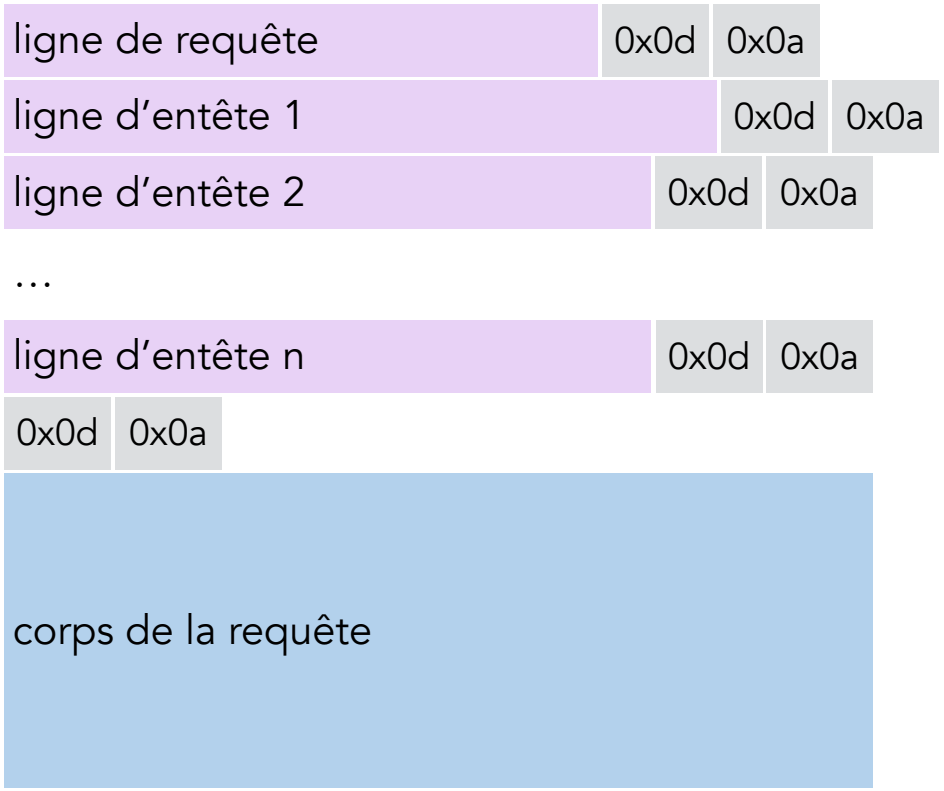
Exemple d'échange Web

- Un utilisateur saisit l'URL :
  `http://www.cnn.com/index.html`
- Son navigateur découvre l'adresse IP du serveur
  - appel de fonction
    `gethostbyname(www.cnn.com)`
  - …qui renvoie la valeur
    `151.101.100.73`
  - sélection d'un numéro de port aléatoire éphémère
  - ouverture d'une connexion TCP
- Le navigateur envoie une requête HTTP
  `"GET /index.html HTTP/1.1`
  `Host: www.cnn.com"`
- Le navigateur reçoit et affiche l'objet
  - l'objet peut être mis en cache
  - l'adresse IP du serveur aussi

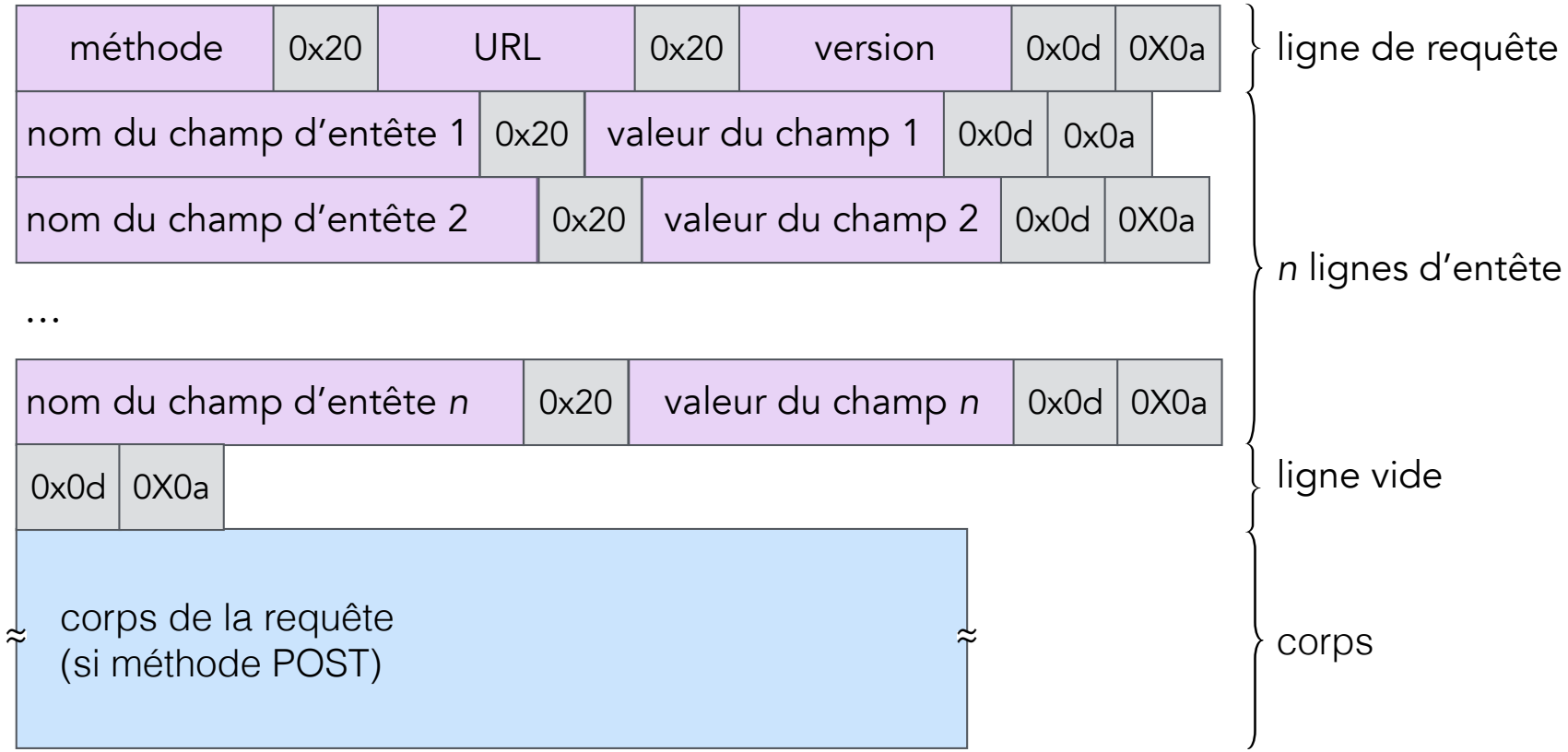


Format des requêtes HTTP

- Une requête HTTP est formée de :
  - d'une entête
    - une ligne de requête
    - des lignes d'entête
  - d'un corps de requête
- Les lignes dans l'entête sont séparées par un retour chariot et d'un saut de ligne
  - <CR><LF> (\r\n) soit en ascii 0x0d0a
- Les champs d'entête sont séparés par un espace
  - <SP> (\s) soit en ascii 0x20
- Un champ contient son nom suivi d'un espace et de sa valeur :
  - <SP> (\s) soit en ascii 0x20
- Le corps de la requête (même vide) et l'entête de la requête sont séparés par un retour chariot et d'un saut de ligne (en plus de ceux de la dernière ligne d'entête)
  - <CR><LF> (\r\n) soit en ascii 0x0d0a



Format des requêtes HTTP



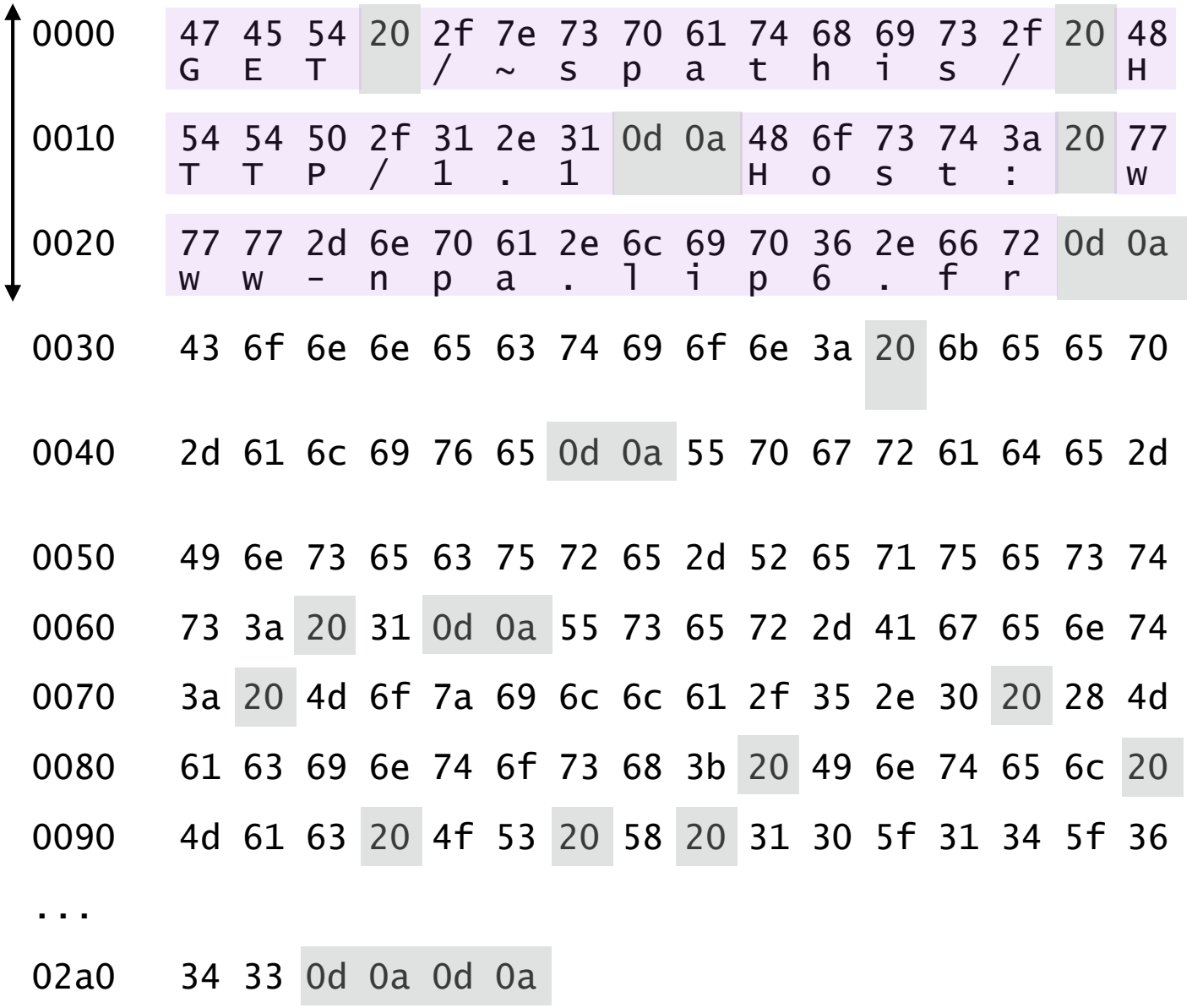
- La longueur des champs dépend des valeurs véhiculées
- Les valeurs des champs sont codées en ASCII
  - Exemple : Champ 'méthode' : 0x474554 (valeur codée GET, longueur 3 octets)

requête http://www-npa.lip6.fr/~spathis/

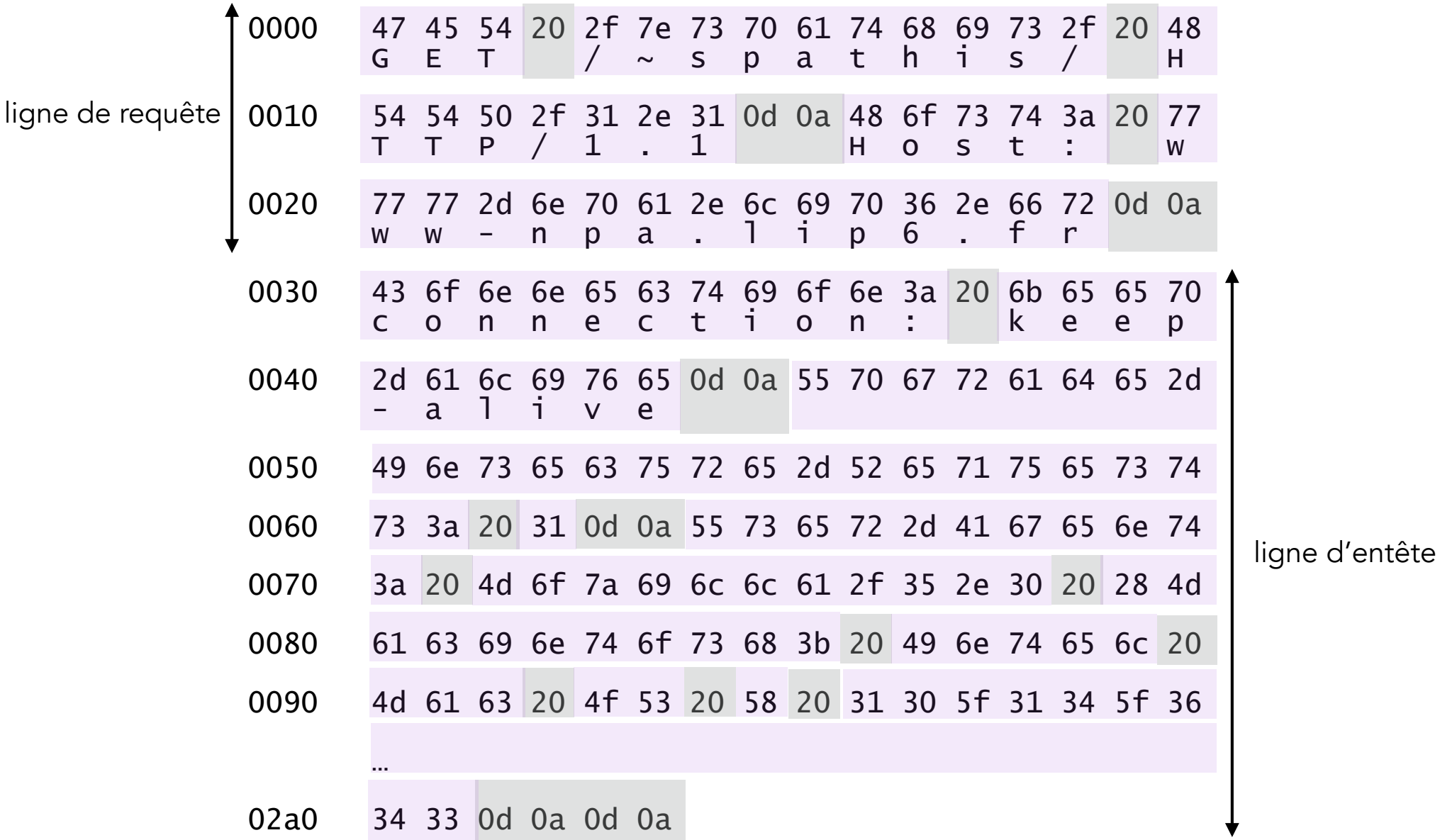
```
0000   47 45 54 20 2f 7e 73 70 61 74 68 69 73 2f 20 48
0010   54 54 50 2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 77
0020   77 77 2d 6e 70 61 2e 6c 69 70 36 2e 66 72 0d 0a
0030   43 6f 6e 6e 65 63 74 69 6f 6e 3a 20 6b 65 65 70
0040   2d 61 6c 69 76 65 0d 0a 55 70 67 72 61 64 65 2d
0050   49 6e 73 65 63 75 72 65 2d 52 65 71 75 65 73 74
0060   73 3a 20 31 0d 0a 55 73 65 72 2d 41 67 65 6e 74
0070   3a 20 4d 6f 7a 69 6c 6c 61 2f 35 2e 30 20 28 4d
0080   61 63 69 6e 74 6f 73 68 3b 20 49 6e 74 65 6c 20
0090   4d 61 63 20 4f 53 20 58 20 31 30 5f 31 34 5f 36
...
02a0   34 33 0d 0a 0d 0a
```

requête http://www-npa.lip6.fr/~spathis/

ligne de requête



requête http://www-npa.lip6.fr/~spathis/



Types de requête : les méthodes

- GET
  - méthode qui permet d'obtenir un objet identifié par la valeur du champ URL
  - les valeurs saisies dans un formulaire sont passées dans l'URL
- HEAD
  - retourne une réponse sans corps (seulement l'entête)
- POST
  - permet de transmettre des données au serveur dans le corps de la requête
  - les valeurs saisies dans un formulaire par exemple
- PUT
  - permet d'envoyer un objet que le serveur stockera à l'emplacement identifié par le champ URL
- DELETE
  - permet de supprimer un objet situé à l'emplacement identifié par le champ URL
- …

Ligne de requête
(commandes HEAD, POST, GET)

```
GET /~spathis/index.html HTTP/1.1
Host: www-npa.lip6.fr
User-agent: Mozilla/5.0
Connection: close
Accept: text/html, image/jpeg
Accept-language:fr
<CRLF>
```

Entête de la requête

Retour chariot résultant en une ligne vide, indique la fin de l'entête

Demande de fermeture de la connexion TCP après envoi de l'objet demandé

Ligne de réponse
(Version + code statut HTTP de la réponse)

date de dernière modification de l'objet

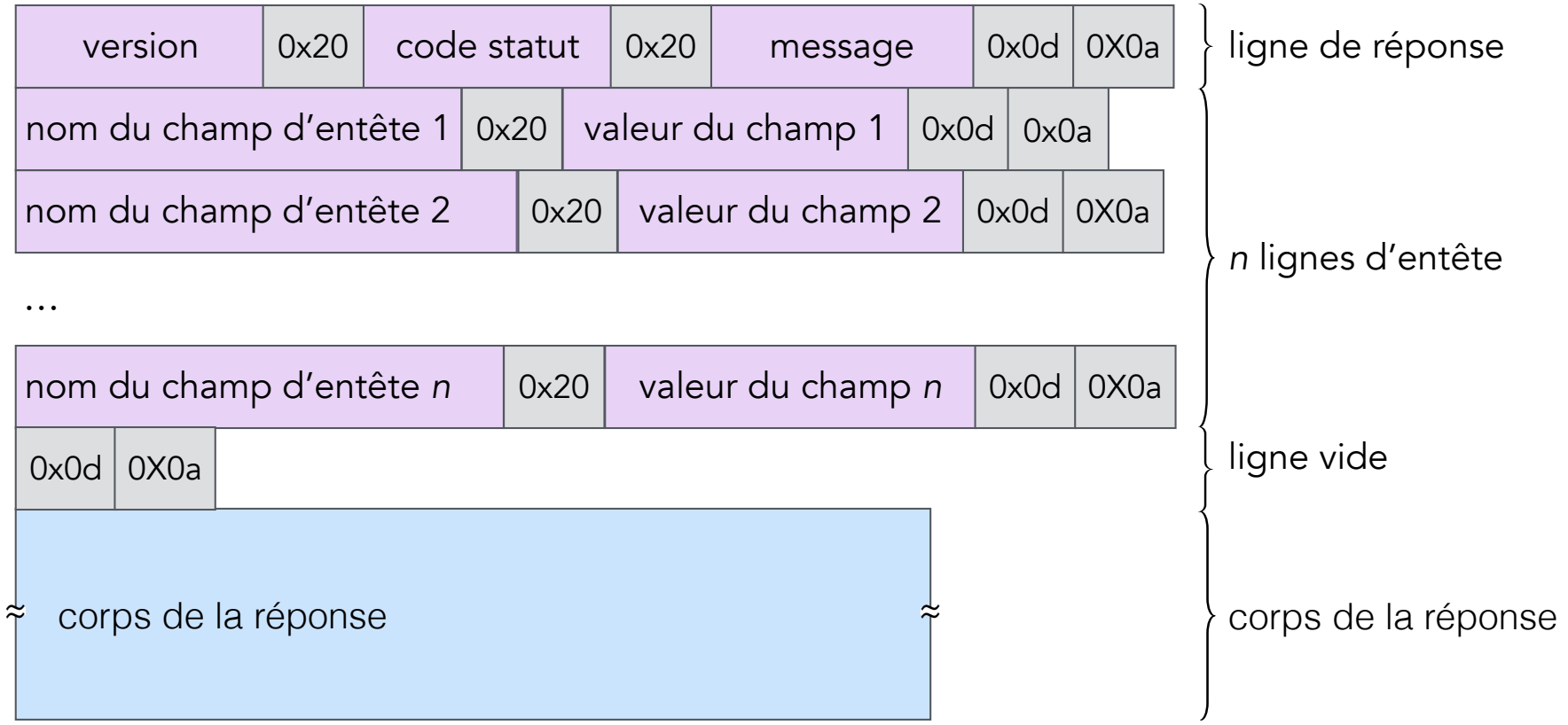
```
HTTP/1.1 200 OK
Connection close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 6 Feb 2017 11:12:23 GMT
Content-Length: 6821
Content-Type: text/html
<CRLF>
data data data data data ...
```

Entête de la réponse

Données (ex : le fichier HTML demandé)

Fermeture de la connexion TCP après envoi de cette réponse

Format des réponses HTTP



- La longueur des champs dépend des valeurs véhiculées
- Les valeurs des champs sont codées en ASCII
  - Exemple : Champs 'code statut' et 'message' :
    - 0x323030204f4b : 200 OK)

Code de statut HTTP

- 200 OK
  - requête en succès : l'objet demandé se trouve dans le corps de la réponse
- 301 Moved Permanently
  - l'objet demandé a changé d'emplacement
  - la nouvelle URL de l'objet se trouve dans le corps de la réponse
- 400 Bad Request
  - la requête n'a pas été comprise (erreur de syntaxe par exemple)
- 404 Not Found
  - l'objet demandé n'a pas été trouvé à l'URL demandé
- 505 HTTP Version Not Supported
  - la version du protocole HTTP n'est pas supportée par le serveur
- …

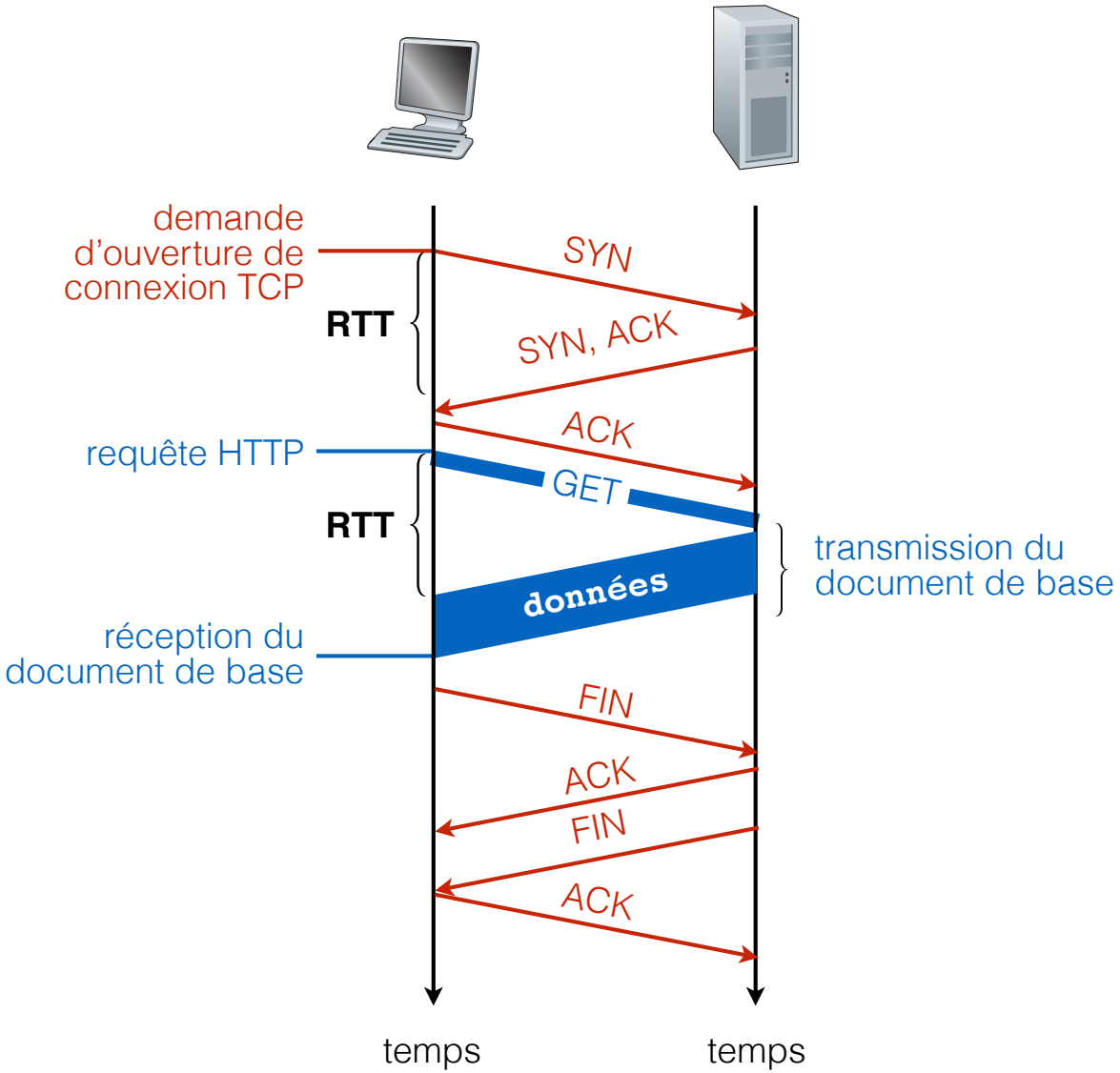
Caching et requête conditionnelle

- Un objet est retourné uniquement si l'objet a changé sur le serveur

```
GET /~spathis/index.html HTTP/1.1
Host: www-npa.lip6.fr
User-Agent: Mozilla/4.03
If-Modified-Since: Mon, 6 Feb 2017 11:12:23 GMT
<CRLF>
```

- Le serveur économise les ressources nécessaires à l'envoi d'un objet
  - en inspectant la valeur "last modified" de l'objet
  - en comparant cette valeur à celle du champ "if-modified-since"
  - en retournant "304 Not Modified" si l'objet n'a pas changé
  - … ou "200 OK" avec la version la plus récente de l'objet
- Utilisation du champ ETag qui contient une empreinte numérique de l'objet

Interaction HTTP/TCP



Interaction HTTP/TCP

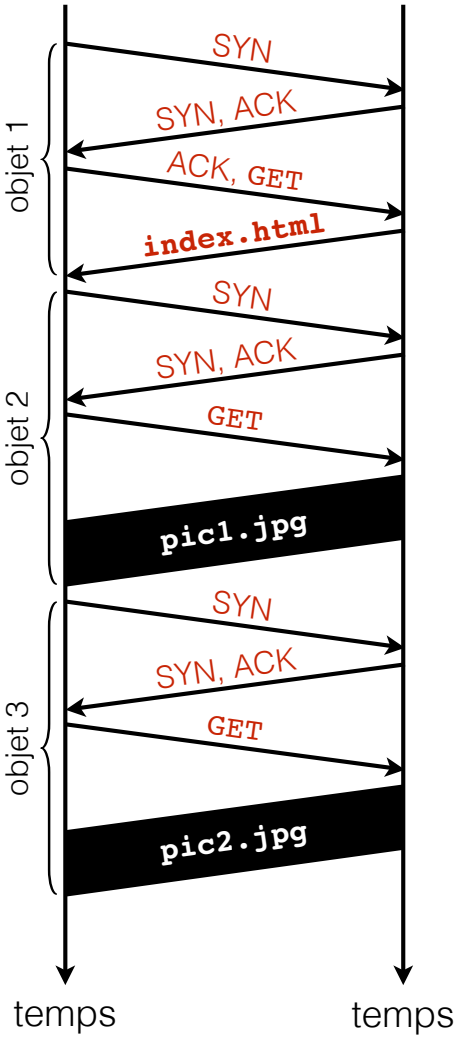
- Une page web est composée de plusieurs objets
  - le document de base qui inclut le texte
  - des images, sa feuille de style (CSS), …
- HTTP envoie une requête par objet
  - le première requête concerne le document de base
  - le navigateur envoie alors autant de requêtes que d'objets référencés
- HTTP fonctionne en mode non connecté
  - HTTP s'en remet à TCP pour fiabiliser la réception des objets

Interaction HTTP/TCP

- HTTP fonctionne en mode non connecté
  - HTTP s'en remet à TCP pour fiabiliser l'envoi des objets
- Une page contient plusieurs objets :
  1. Combien faut-il de connexions TCP ?
     - une par objet : mode non persistant
     - une pour l'ensemble des objets : mode persistant
  2. En mode persistant, doit-on attendre la réception d'un objet avant de réclamer le suivant ?
     - non : mode non persistent sans pipelining
     - oui : mode persistent avec pipelining

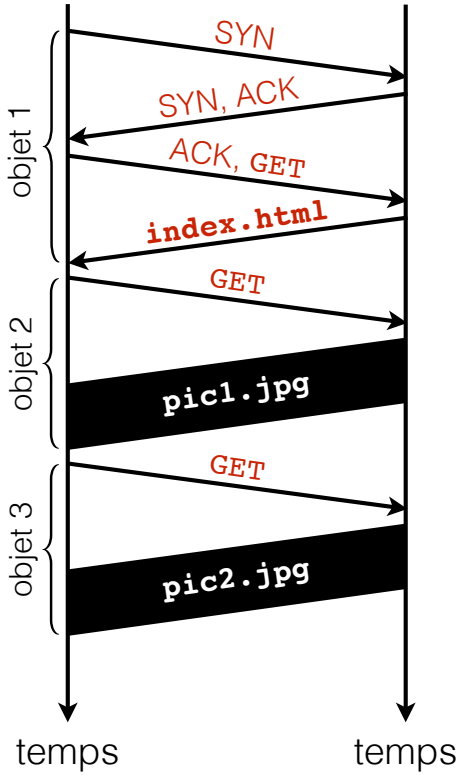
Les modes de fonctionnement de HTTP

HTTP/1.0 (mode non persistant)

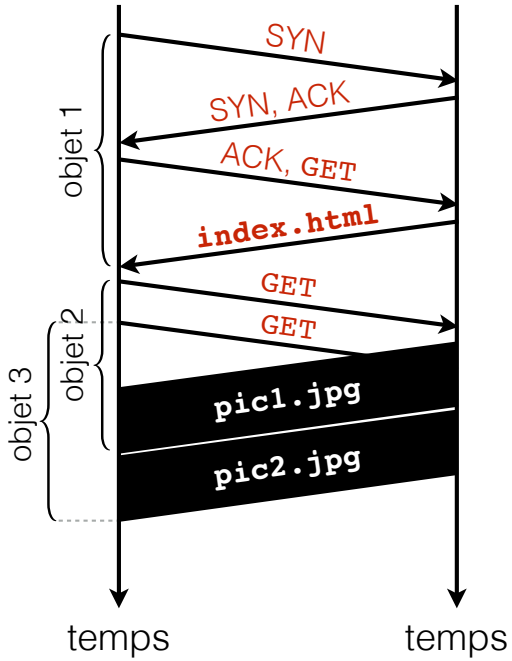


HTTP/1.1 (mode persistant)

sans pipelining

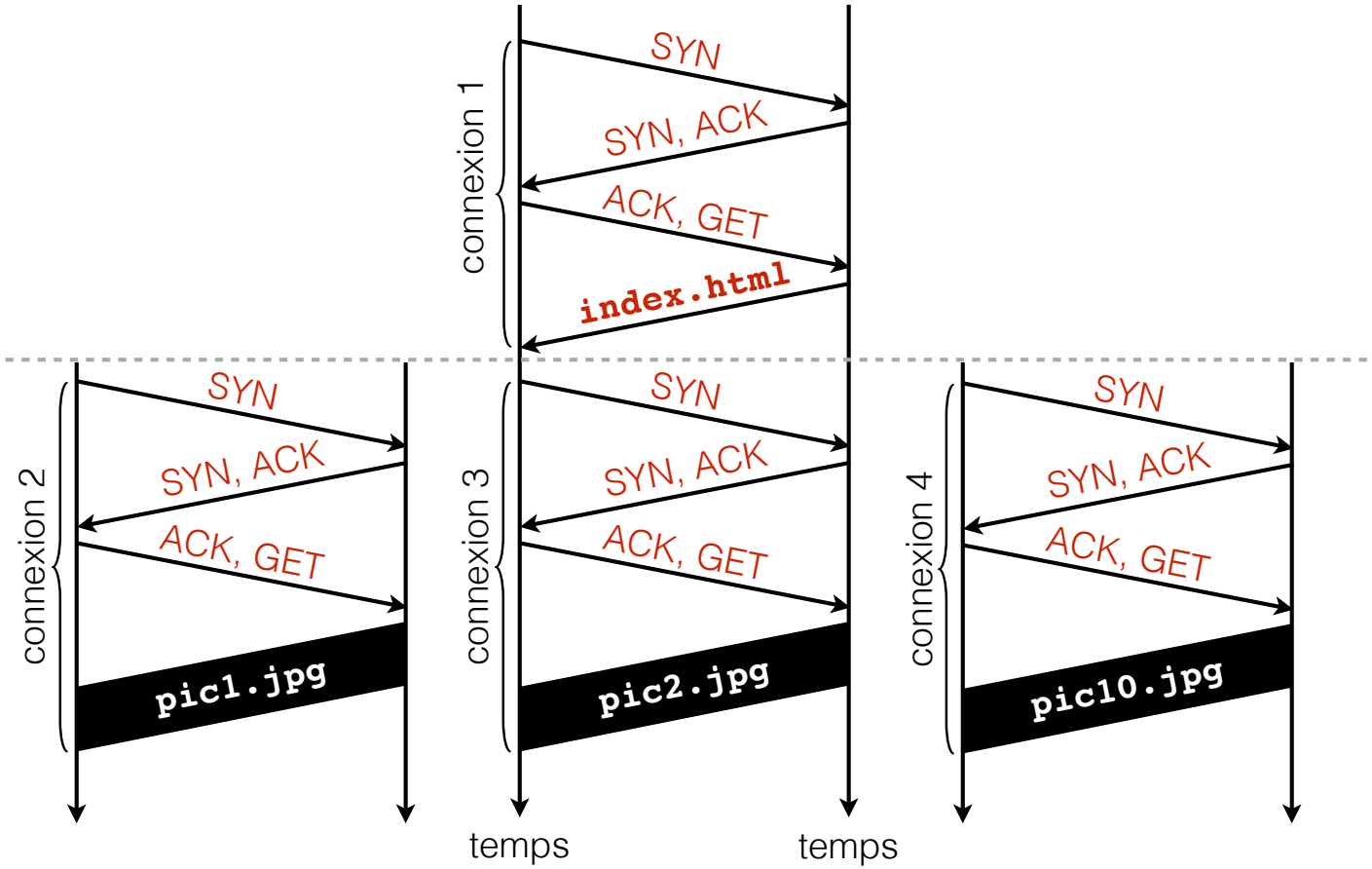


avec pipelining



Les modes de fonctionnement de HTTP

HTTP/1.0 (mode non persistant)



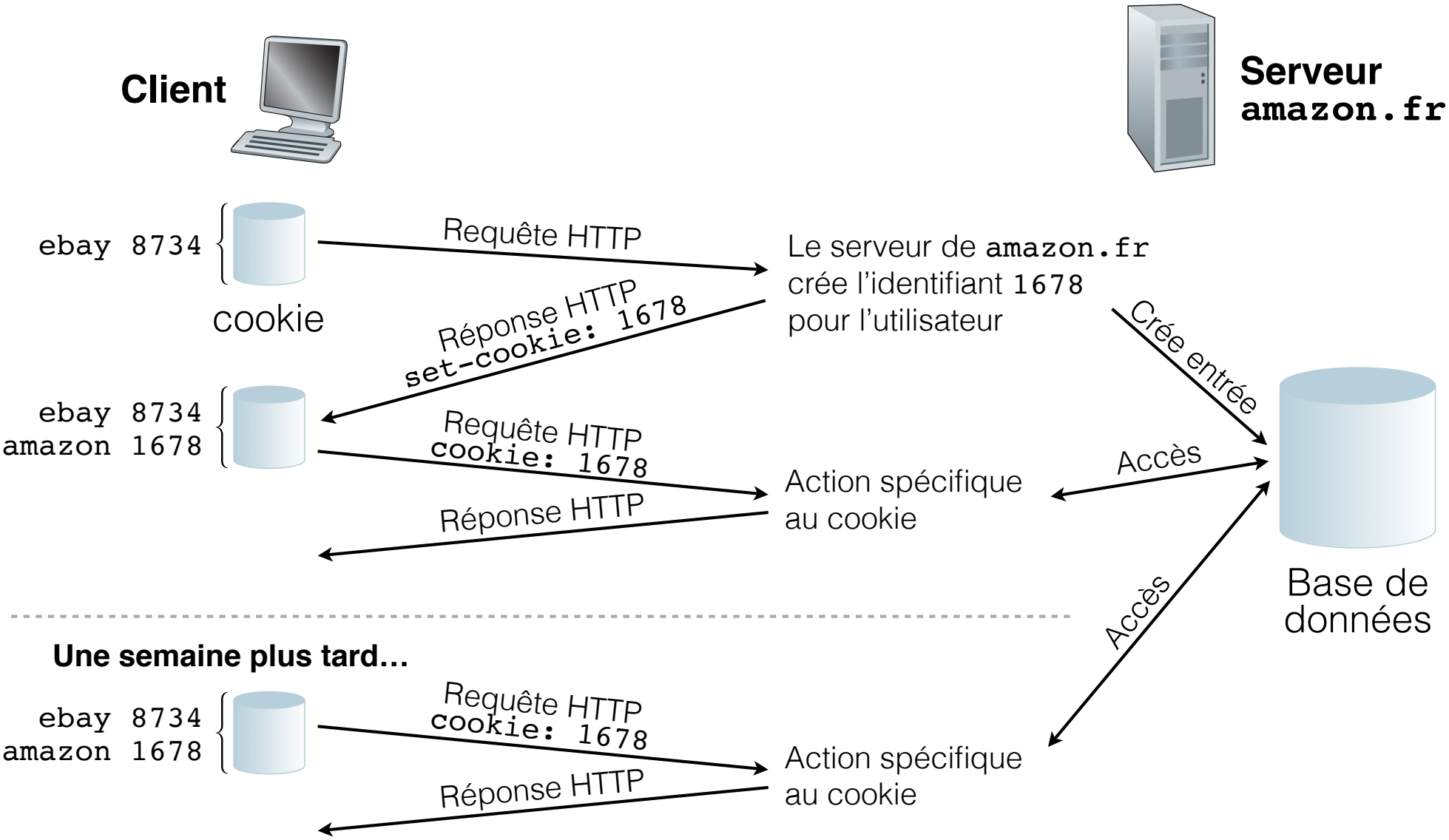
Mode persistant vs non persistant

- Mode non persistant : HTTP/1.0
  - Les navigateurs ouvrent plusieurs connexions TCP concurrentes
  - Surcharge liée à la gestion des connexions TCP
  - Equité vis à vis des trafics utilisant une seule connexion TCP empruntant les mêmes chemins
- Mode persistant : HTTP/1.1
  - Evite de gérer plusieurs connections TCP
    - Google Chrome : 6, MS Internet Explorer : 8
  - Une connexion TCP longévive :
    - permet une meilleure estimation du RTT
    - atteint un débit d'émission acceptable
  - En pratique, le pipelining n'est pas utilisé !
    - des serveurs Web ne répondent pas toujours selon l'ordre de réception des requêtes
    - problème du HOL (Head of the line) : le premier objet demandé, si volumineux, bloque l'affichage du reste de la page

Cookies

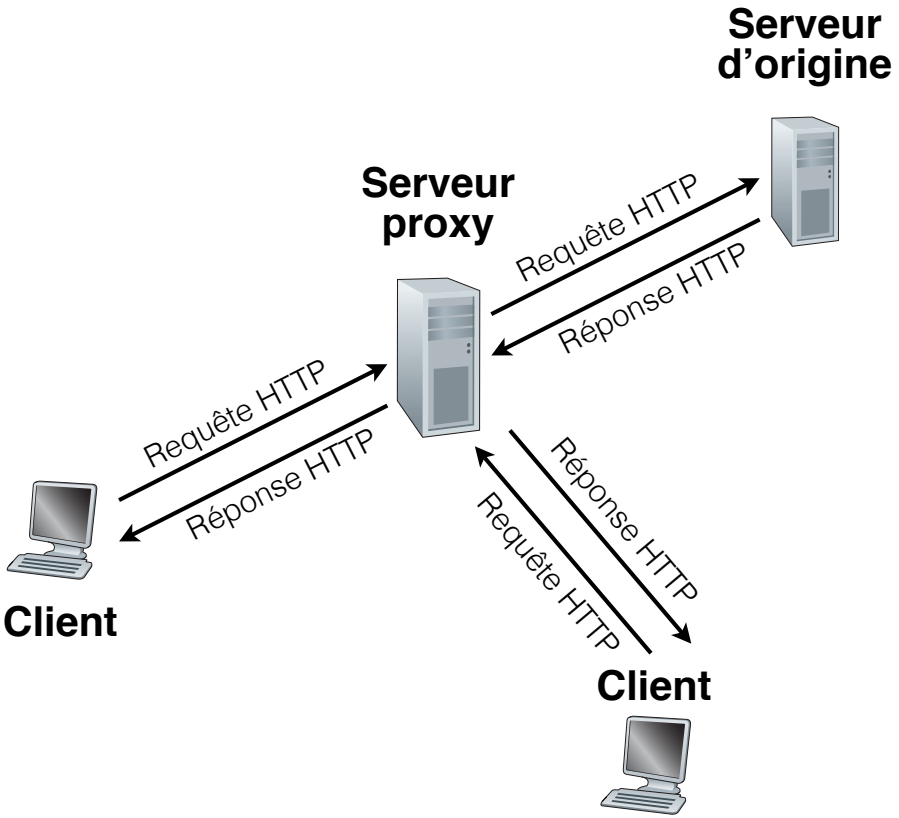
- Un cookie permet aux serveurs Web d'identifier leurs clients récurrents
- Les serveurs Web maintiennent des états persistants décrivant les activités de ces clients lors de leurs visites
- Composé de 4 éléments :
  - champ cookie dans l'entête des réponses HTTP
  - champ cookie dans l'entête des requête HTTP
  - un fichier cookie installé par le navigateur
  - une base de données côté serveur qui contient une entrée par cookie
- Exemple :
  - Alice utilise toujours le même ordinateur (hôte) pour accéder à Internet
  - Elle visite un site de e-commerce (ex : eBay) pour la première fois
  - Avec la requête HTTP initiale, le site crée : un identifiant unique et une entrée dans la base de données
- Apport des cookies :
  - Authentification, panier d'achat, recommandation (publicités), informations sur la session utilisateur (Web, email…)

Cookies


Client
Serveur
amazon.fr
ebay 8734
cookie
Requête HTTP
Le serveur de amazon.fr
crée l'identifiant 1678
pour l'utilisateur
Réponse HTTP
set-cookie: 1678
Crée entrée
ebay 8734
amazon 1678
Requête HTTP
cookie: 1678
Accès
Action spécifique
au cookie
Réponse HTTP
Base de
données
Accès
Une semaine plus tard…
ebay 8734
amazon 1678
Requête HTTP
cookie: 1678
Action spécifique
au cookie
Réponse HTTP

Caches Web
(serveur *proxy*)

- **But** Satisfaire une requête en minimisant les ressources nécessaires à l'envoi d'un objet
- Un navigateur envoie ses requêtes au proxy
  - le proxy Web est connu des navigateurs
    - sans résoudre le nom du serveur d'origine
  - le proxy Web est transparent
    - il intercepte les requêtes des clients
- Si l'objet est présent, le cache vérifie auprès du serveur si l'objet est valide (à jour)
  - si l'objet est confirmé comme valide :
    - le cache renvoie l'objet au client
  - sinon le serveur renvoie la version valide du l'objet au cache
    - l'objet est mis en cache
    - le cache renvoie l'objet au client



Caches Web
(serveur *proxy*)

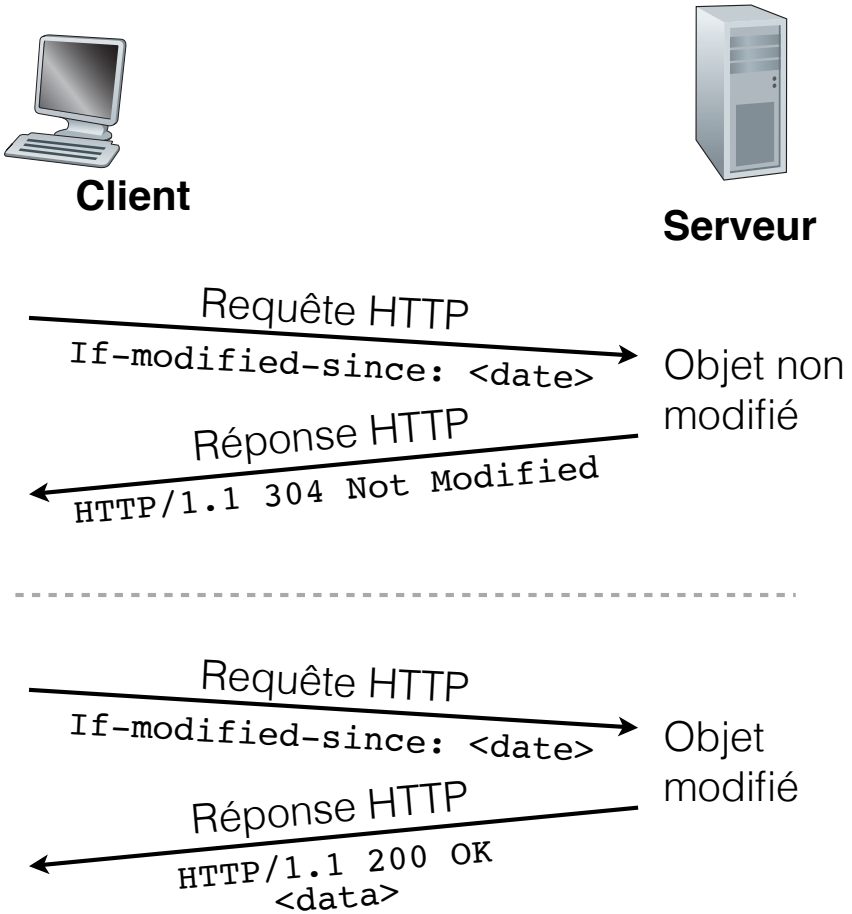
- Un cache agit à la fois comme client **et** serveur
  - Comme serveur pour la requête initiale du client
  - Comme client vis-à-vis du serveur d'origine
- Côté client, les caches sont installés par les ISP (université, entreprise, FAI)
- Côté serveur, les caches sont gérer par des sociétés privées appelées CDN
  - Akamai, Limelight, Cloudflare, …

- Pourquoi faire du *caching* ?
  - Réduction du temps de réponse aux requêtes de client
  - Réduction du trafic sur le lien d'accès d'un site
  - La bande passante est inversement proportionnelle au RTT du chemin

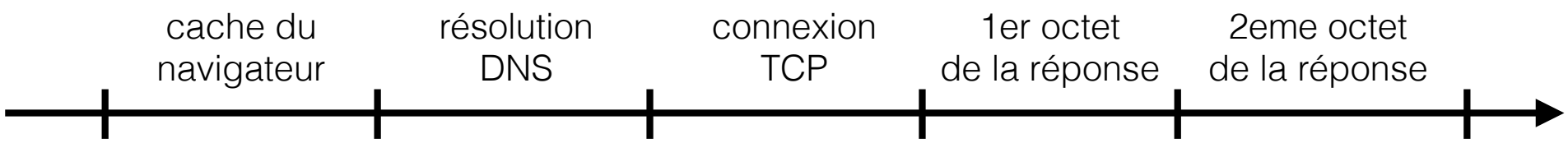
`Get` conditionnel

- **But** Envoyer un objet uniquement si le cache a une version à jour de l'objet

- Le cache spécifie la date de dernière modification de l'objet dans sa requête HTTP
  - `If-modified-since: <date>`
- La réponse du serveur ne contient pas l'objet si la copie cachée est à jour :
  - `HTTP/1.1 304 Not Modified`



Téléchargement Web



Domain Name System
DNS

Noms vs Adresses IP

- Dans Internet, les machines sont identifiées par :
  - **Adresse IP** utilisée pour adresser les paquets
    - Adresses numériques appréciées des routeurs
    - Longueur fixe, nombres binaires (32 bits pour IPv4 ou 128 bits pour IPv6)
    - Hiérarchiques en rapport avec la localisation topologique des hôtes
  - **“Nom”** (*ex* : www.google.fr) utilisé par les humains
    - Noms ménomiques appréciés des utilisateurs
    - Longueur variable, caractères alpha-numériques
    - Donnent peu (voir aucune) d’information sur la location

Nommage et adressage

- Les noms sont plus simples à retenir

  `www.google.fr` vs. `74.125.133.94`

- Les adresses peuvent changer
  - Déplacer le serveur `www.google.fr` à l'adresse `74.125.133.100`
  - *Exemple* : renumérotation en cas de changement de fournisseur (*provider*)
- Un nom peut cacher plusieurs adresses IP
  - Le serveur Web `www.google.fr` est répliqué à plusieurs endroits
- Faire correspondre une adresse différente selon notre position
  - L'adresse IP d'une copie du serveur Web proche
  - *Ex* : pour réduire la latence, ou retourner des contenus différents
- Plusieurs noms pour une même adresse IP
  - *Ex* : alias tels que `ee.mit.edu` et `cs.mit.edu`

DNS, qu'est ce que c'est ?

- **Le DNS (*Domain Name System*) définit :**
  - le **format** des noms utilisés pour identifier les machines dans Internet
  - la **méthode d'attribution** des noms garantissant leur unicité
  - une **base de données répartie** contenant les correspondes entre noms et adresses IP
  - une **collection de serveurs** qui gère chacun, une partie des correspondances entre noms et adresses IP
  - un **protocole de résolution des noms** (*ex* : `www.google.com`) en adresses IP (*ex* : `74.125.133.94`)

DNS
Domain Name System

Base de données distribuée mondiale

Propriétés de DNS

Espace de nommage hiérarchique divisé en zones

Les noms d'une même zone sont gérés par un serveur DNS d'autorité (redondé)

Hiérarchie de serveurs DNS

Serveurs DNS racine (*root servers*)

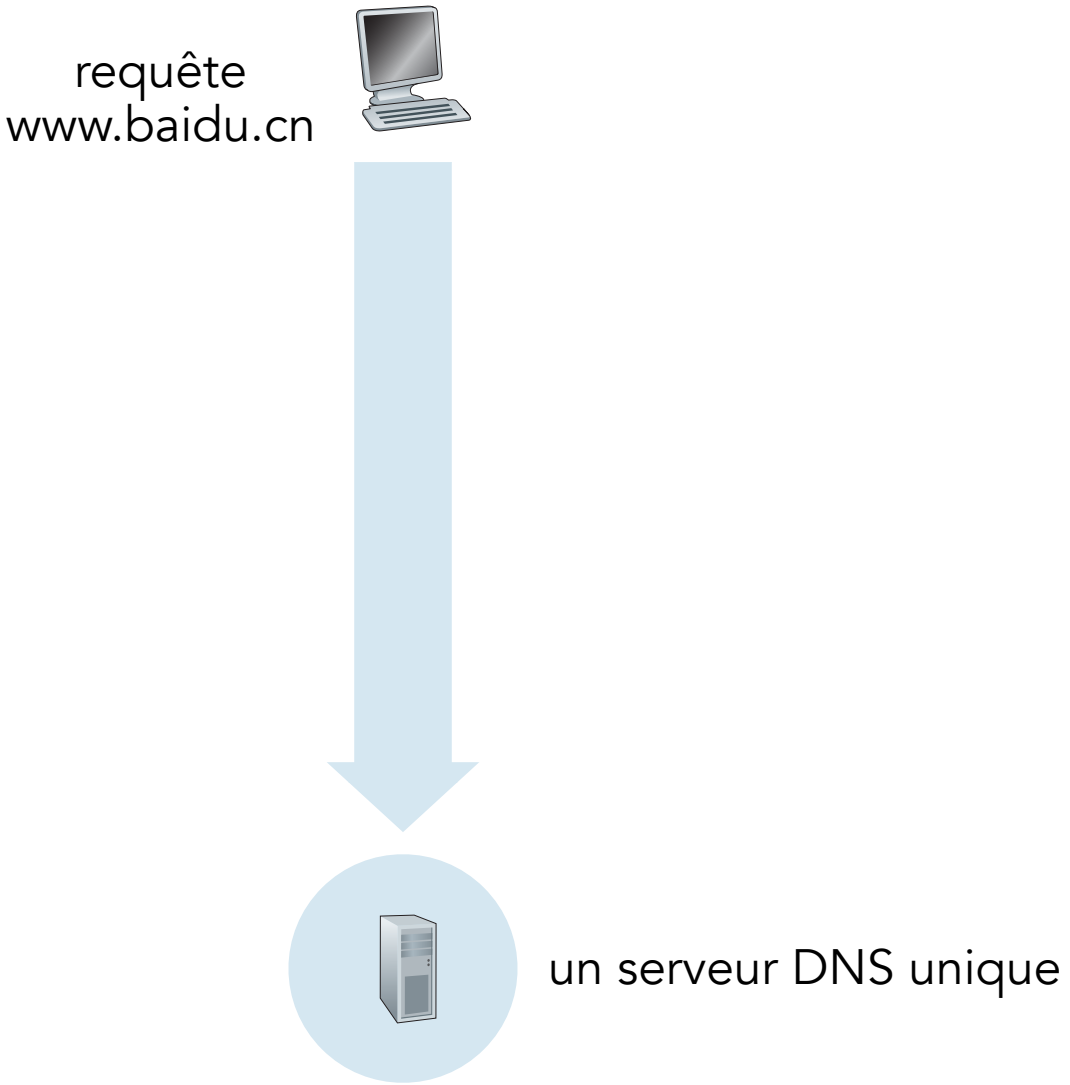
Serveurs TLD (*Top-Level Domain servers*)

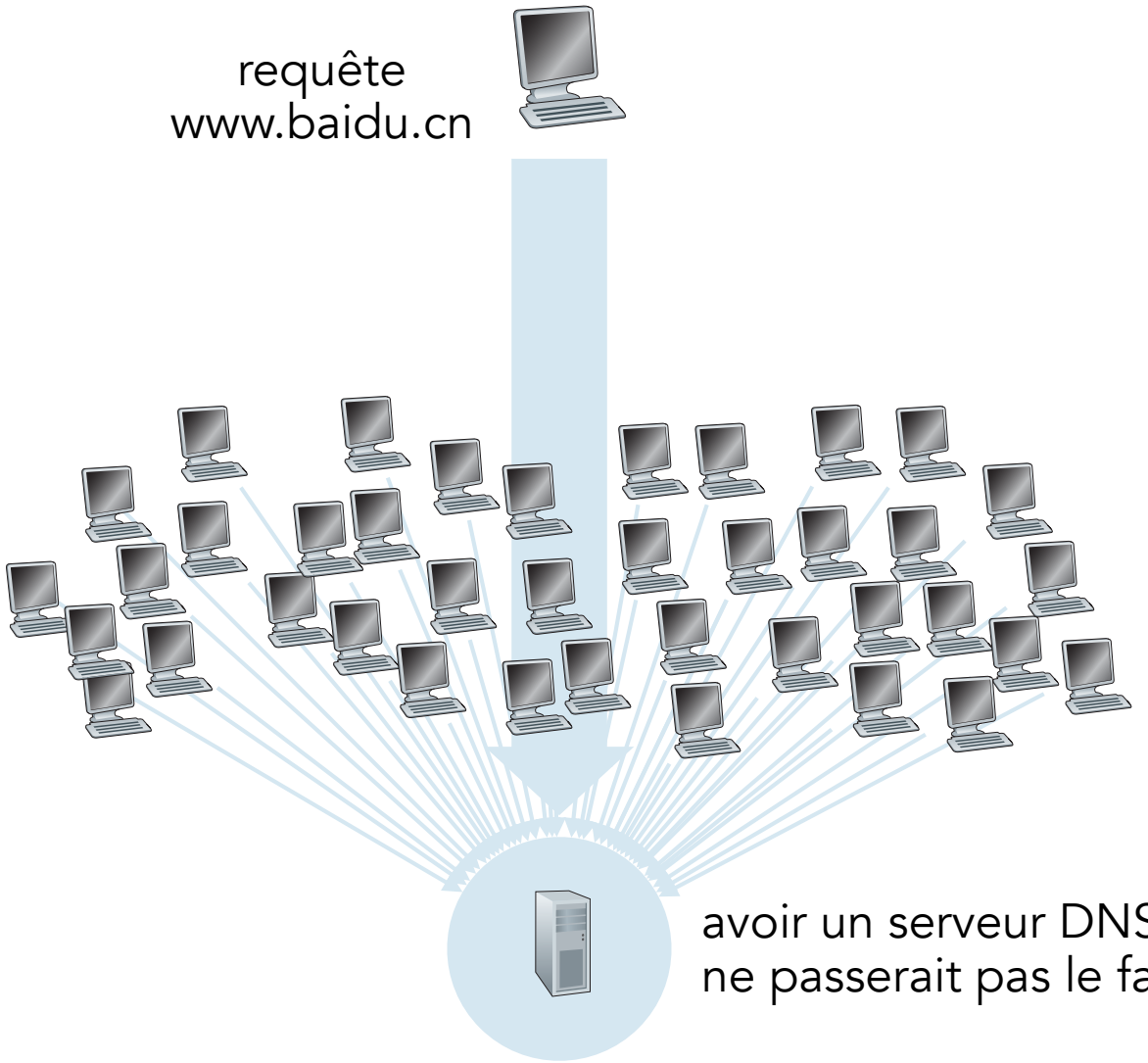
Serveurs DNS d'autorité (*Authoritative DNS servers*)

Execution des requêtes

Serveurs DNS locaux (*local DNS servers*)

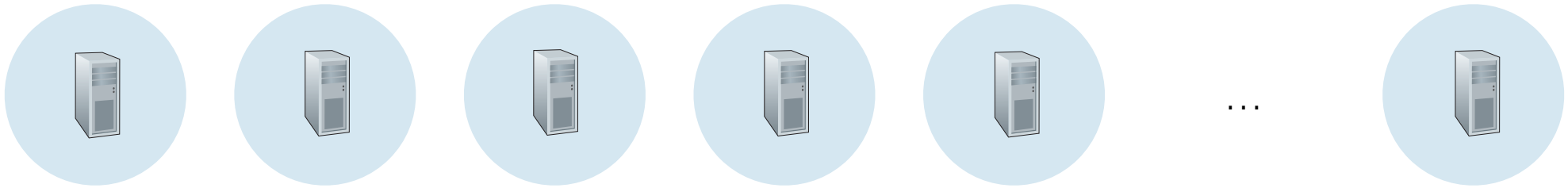
Logiciel de résolution (*resolver software*), *ex* : navigateur Web


requête
www.baidu.cn
un serveur DNS unique



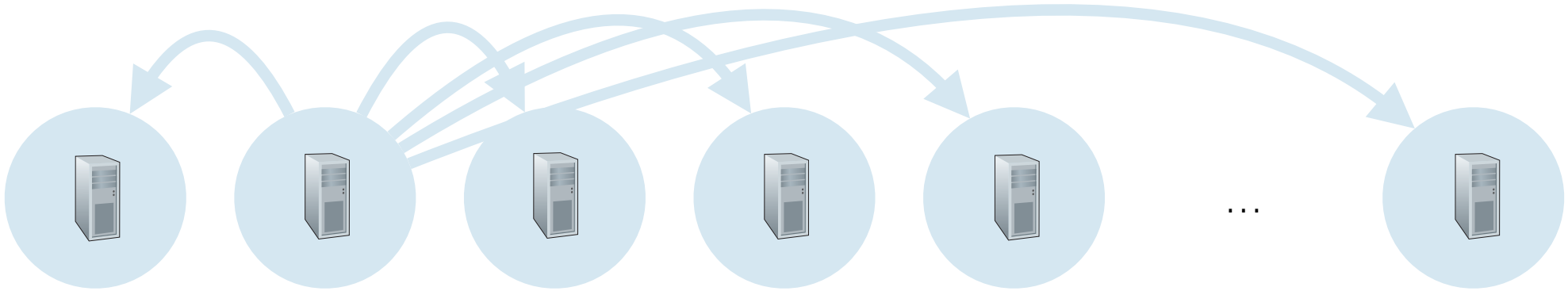
avoir un serveur DNS unique
ne passerait pas le facteur d'échelle

requête
www.baidu.cn



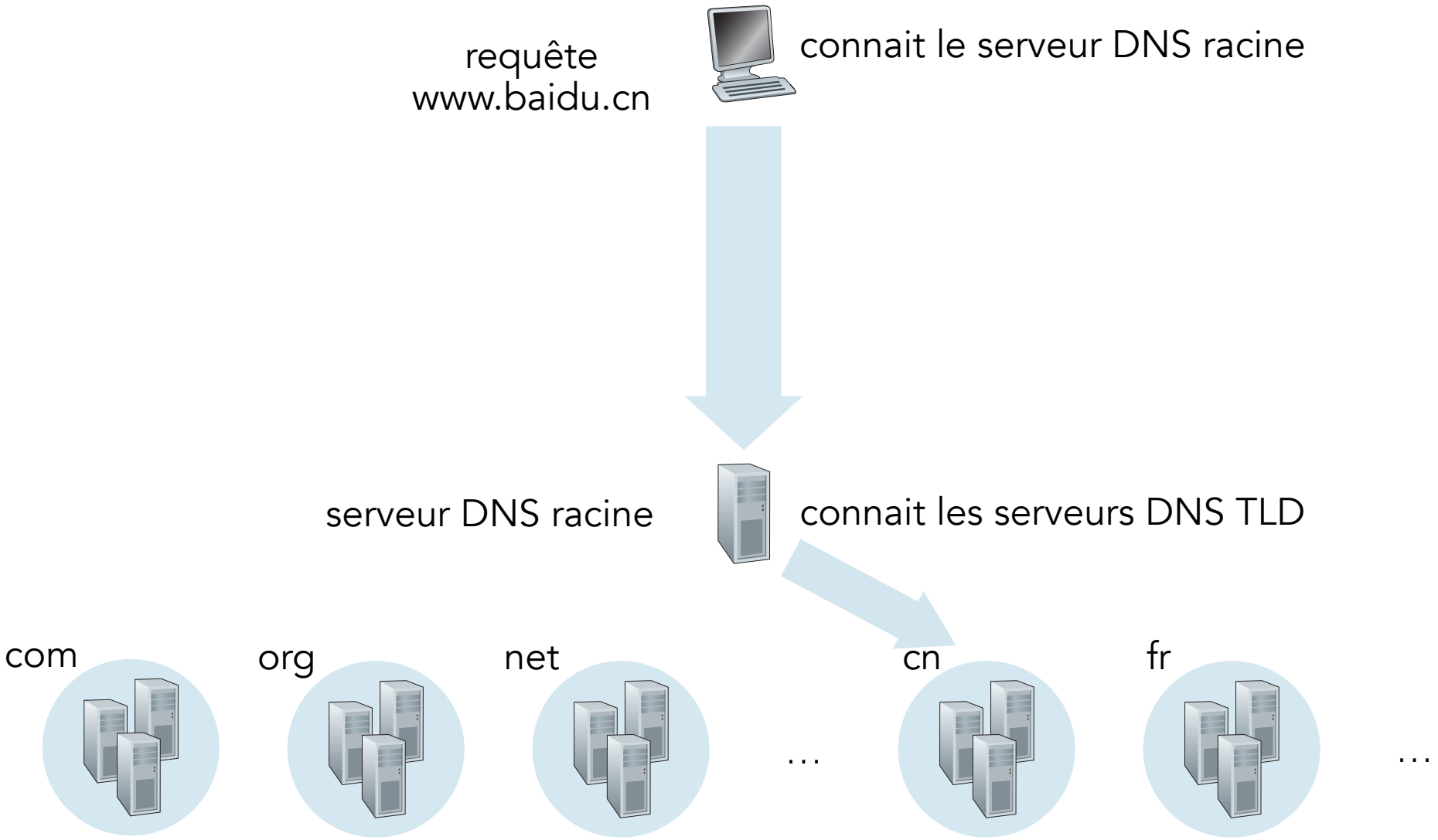
répliquer le même serveur DNS

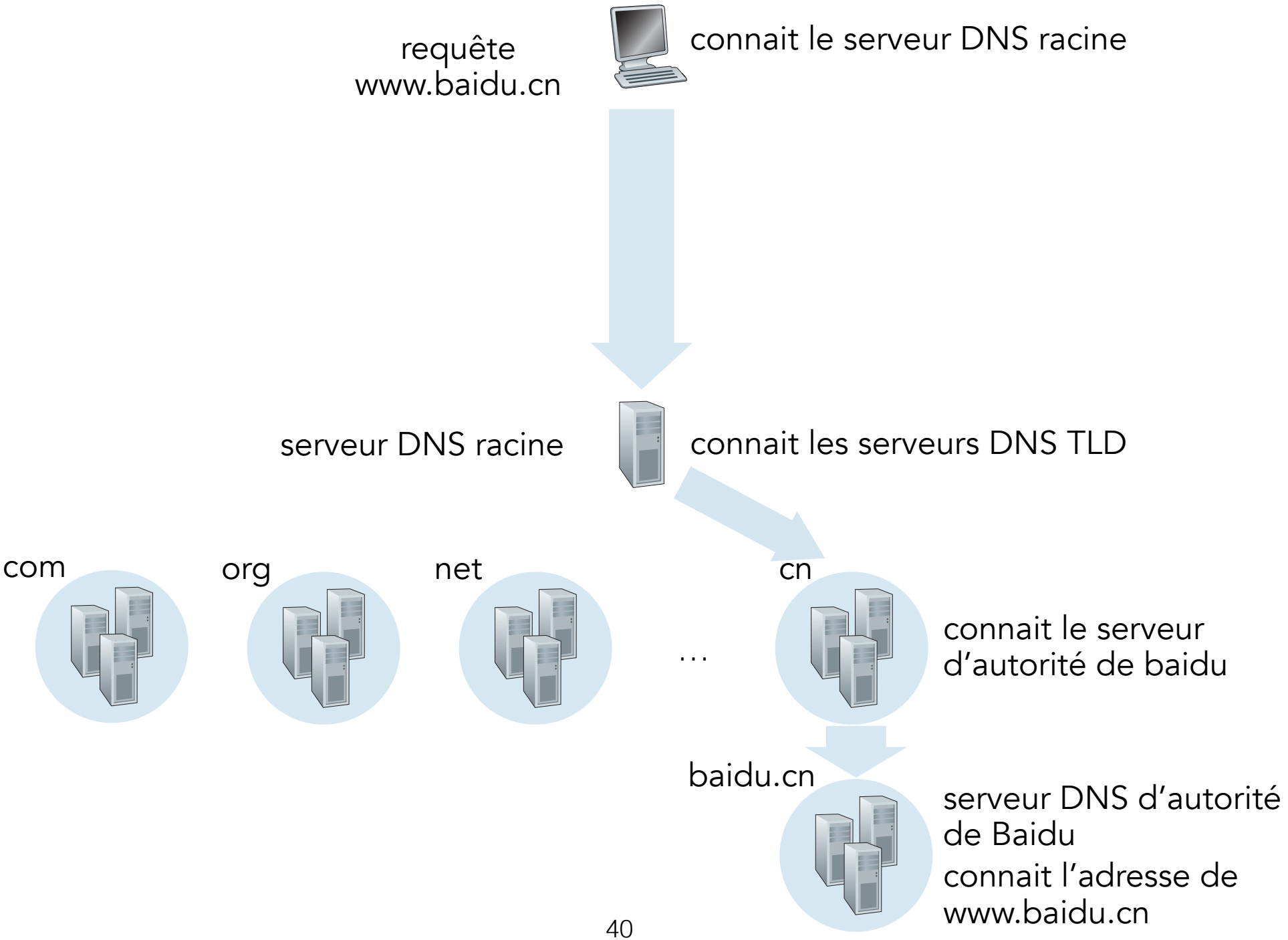
requête
www.baidu.cn

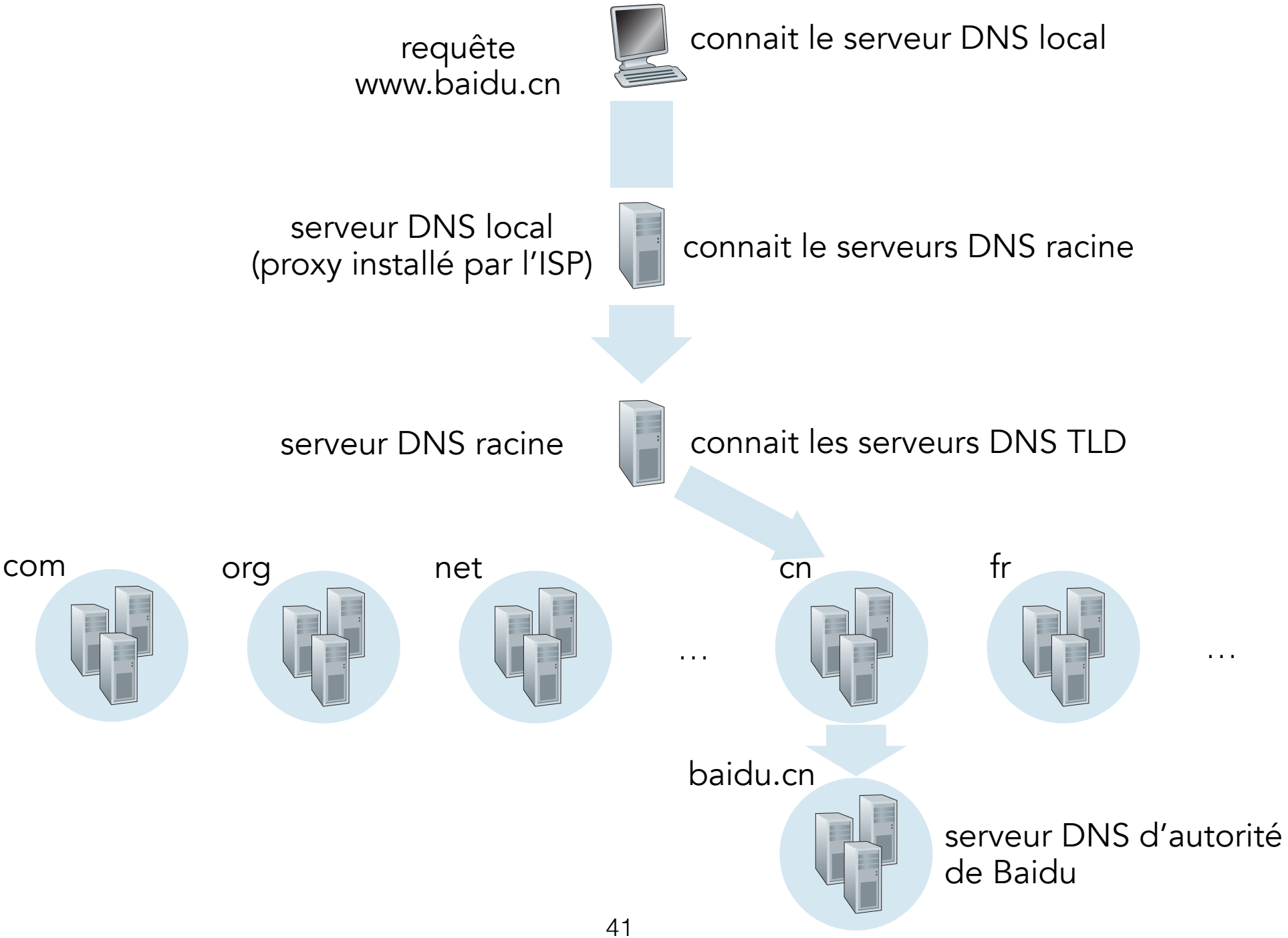


répliquer le même serveur DNS
nécessite de synchroniser les replicas

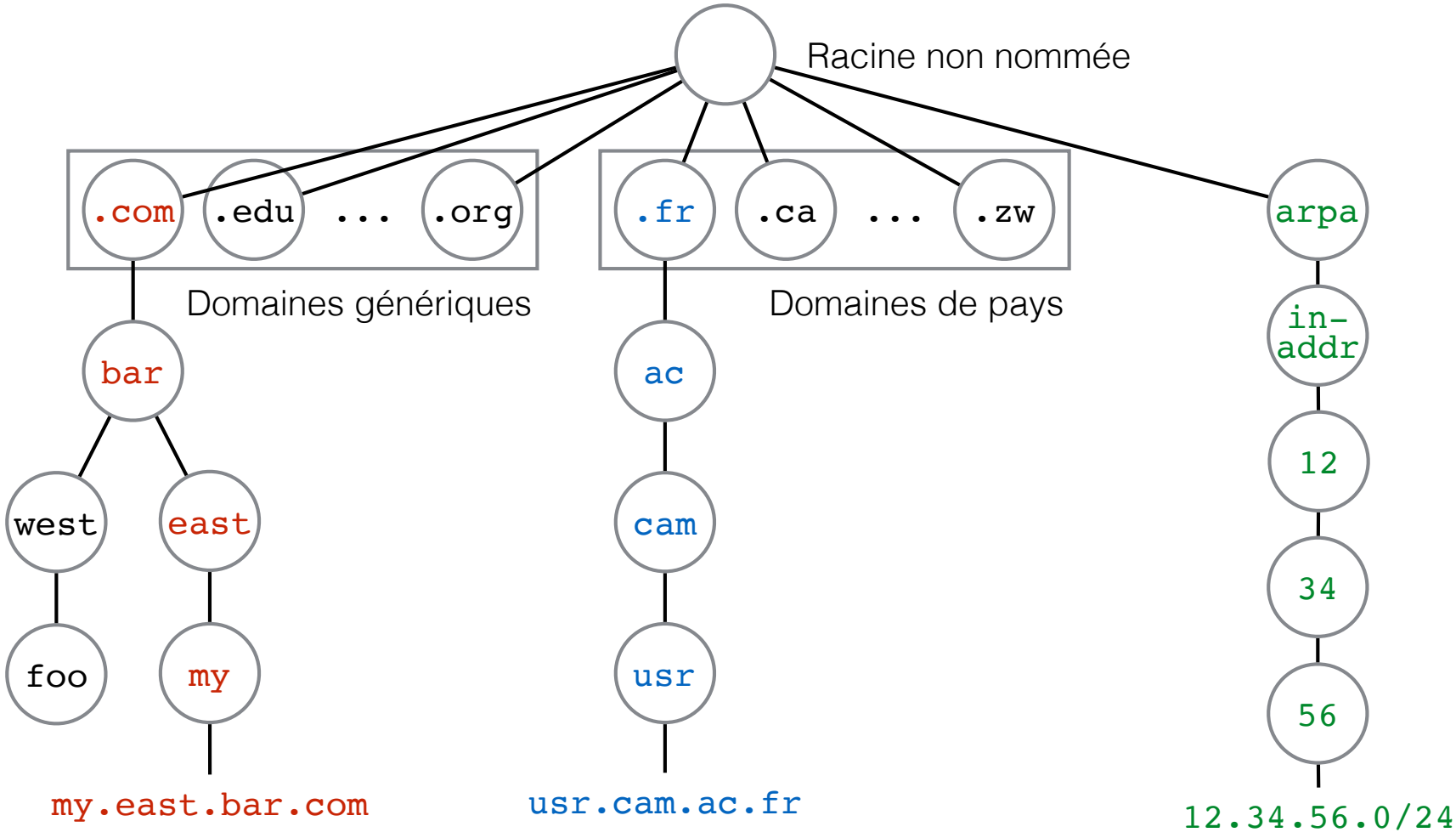
- la base de données des correspondances (nom d'hôte, adresse IP) est découpée et répartie sur plusieurs serveurs DNS
- chaque serveur DNS ne connait les adresses IP des machines dont le nom se termine par un certain suffixe


requête
www.baidu.cn
connait le serveur DNS racine
serveur DNS racine
connait les serveurs DNS TLD
com
org
net
…
cn
fr
…

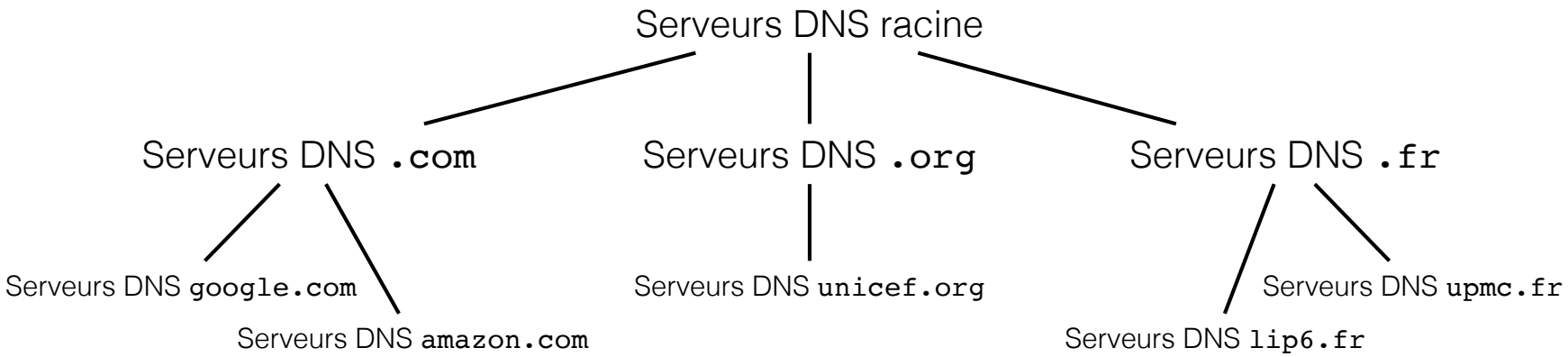

requête
www.baidu.cn
connait le serveur DNS racine
serveur DNS racine
connait les serveurs DNS TLD
com
org
net
cn
…
connait le serveur
d'autorité de baidu
baidu.cn
serveur DNS d'autorité
de Baidu
connait l'adresse de
www.baidu.cn


requête
www.baidu.cn
connait le serveur DNS local
serveur DNS local
(proxy installé par l'ISP)
connait le serveurs DNS racine
serveur DNS racine
connait les serveurs DNS TLD
com
org
net
…
cn
fr
…
baidu.cn
serveur DNS d'autorité
de Baidu

DNS
Base de données distribuée, hiérarchique



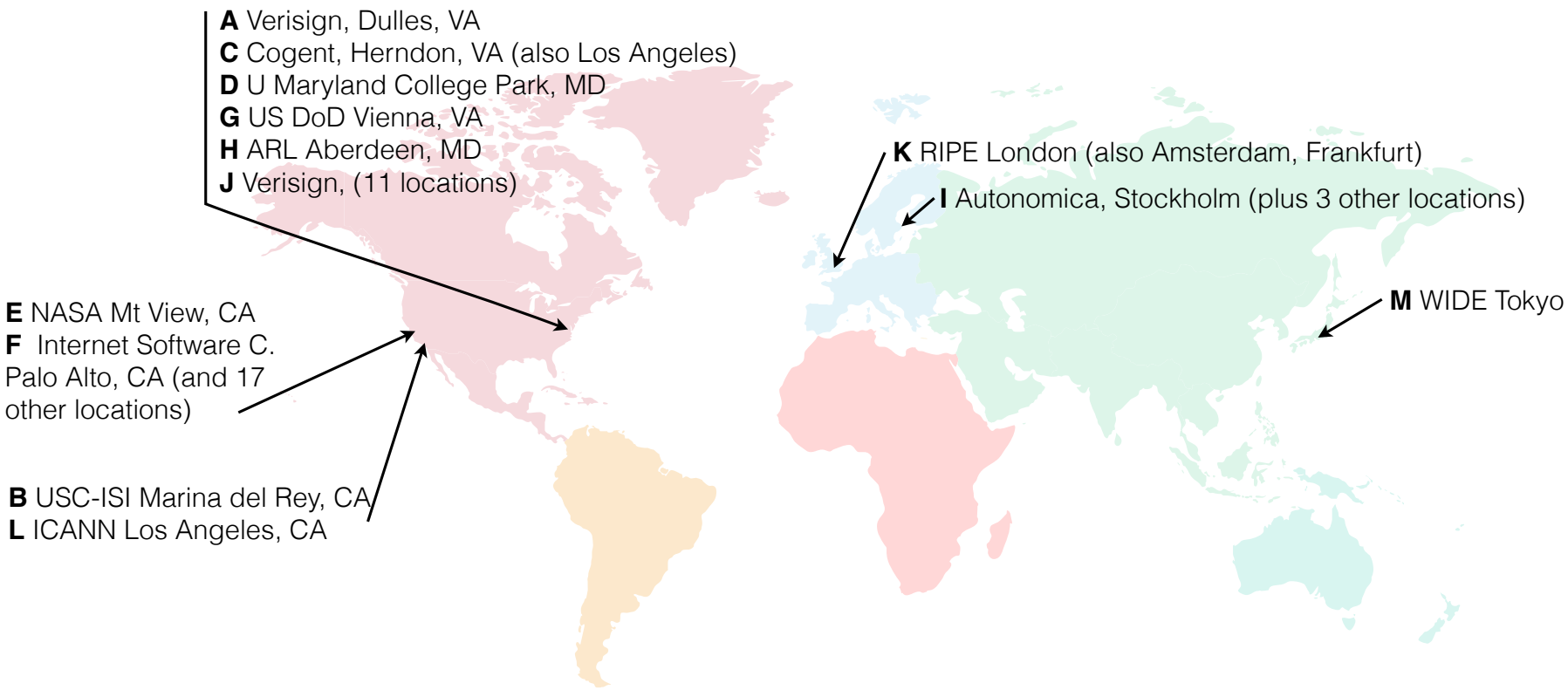
DNS
Base de données distribuée, hiérarchique



Un client veut l'@ IP de www.amazon.com

- Le client envoie une requête au serveur DNS local de son ISP (qu'il découvre par DHCP)
- Le serveur DNS local envoie une requête au serveur racine pour trouver l'adresse IP du serveur DNS TLD .com
- Le serveur DNS local envoie une requête au serveur DNS TLD .com pour trouver l'adresse IP du serveur DNS d'autorité de amazon.com
- Le serveur DNS local envoie une requête au serveur DNS d'autorité amazon.com pour obtenir l'adresse IP de www.amazon.com
- Le serveur DNS local retourne l'adresse IP de www.amazon.com au client

Serveurs DNS racine



13 serveurs DNS racine (voir http://www.root-servers.org/)

connus des serveur DNS locaux qui ne savent résoudre aucun nom (à part ceux en cache)

Serveurs TLD et d'autorité

Serveurs TLD (*Top-Level Domain*)

- Responsables des noms de domaine génériques (*ex* : .com, .org, .edu)
- Responsables des noms de domaine de pays (*ex* : .fr, .ca, .jp)
- Noms de domaine gérés par des sociétés privées
  - *Network Solutions* est en charge du nom de domaine .com
  - *Educause* est en charge du nom de domaine .edu
- Connus des serveurs DNS racine

Serveurs DNS d'autorité (*authoritative*)

- Gérés par une institution ou pour le compte d'une institution par un fournisseur de service
- Connaissent les correspondances pour les machines appartenant à l'institution qu'ils gèrent (*ex* : serveurs mail, Web, …)
- Les seuls serveurs DNS à connaitre les adresses IP des machines

Serveurs DNS locaux

N'appartiennent pas vraiment à la hiérarchie DNS

Configuration : Chaque institution (FAI, entreprise, université) en met à disposition des machines hôte de son réseau

- Aussi appelé "serveur de nom par défaut", *default name server*
- Connu des machines hôte par configuration manuelle ou DHCP

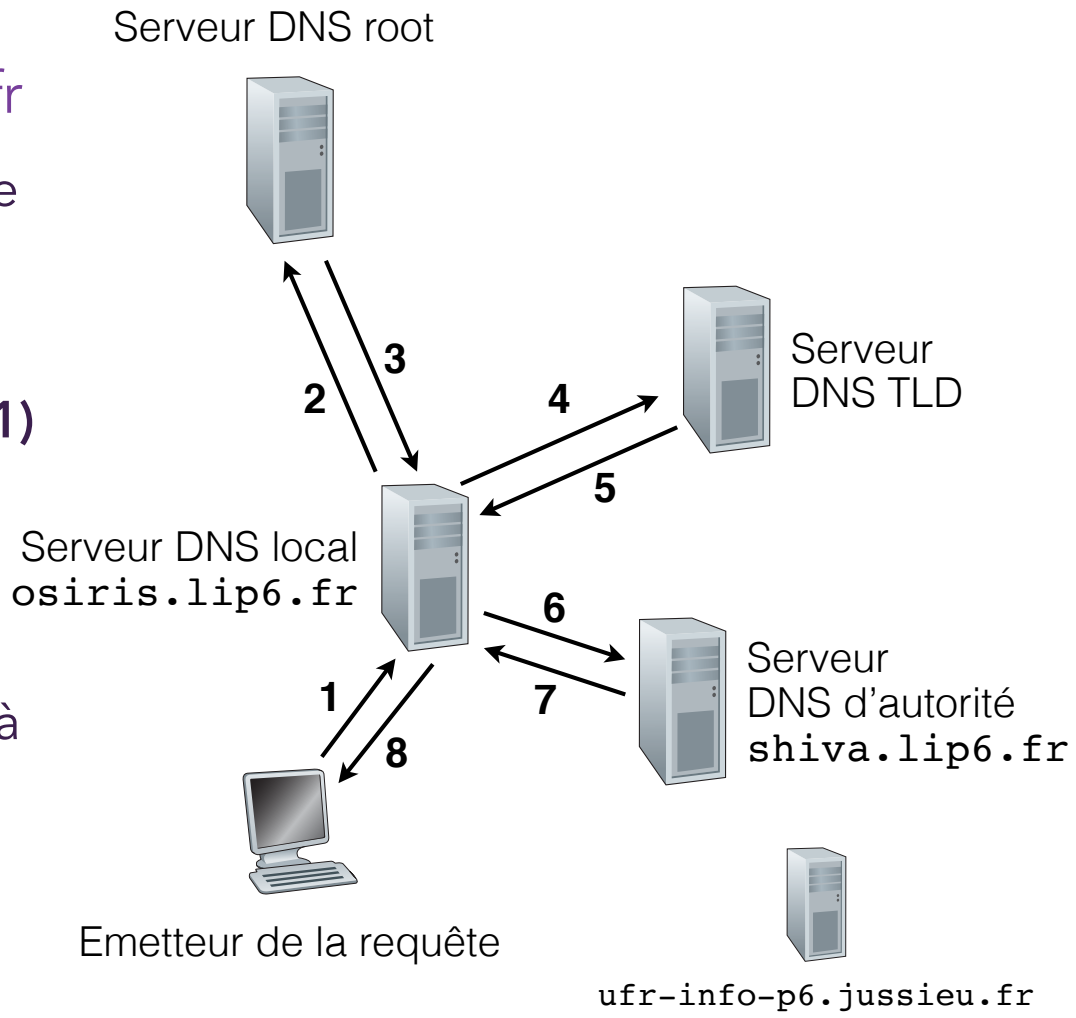
Procédé : Lorsqu'un hôte émet une requête DNS, celle-ci est envoyée au serveur DNS local du réseau

- Gère un cache local contenant les correspondances récemment obtenues
- Joue le role de proxy en acheminant les requêtes pour le compte des clients du réseau
- Les seuls à connaitre les 13 serveurs DNS racine

Résolution d'un nom DNS Requête **hybride**

Un hôte veut l'adresse IP du serveur Web ufr-info-p6.jussieu.fr

- L'hôte soumet une requête de type *recursion-requested* à son serveur DNS local osiris.lip6.fr
- L'hôte fait une requête **récursive (1)**
- Le serveur DNS local fait une recherche **itérative (2-4-6)**. Il contacte d'autres serveurs DNS avant de donner la réponse finale à l'hôte



DNS et caches

La résolution de toutes ces requêtes prend du temps

- Nécessaire pour que la communication ait lieu
- *Ex* : 1 seconde de latence avant de télécharger un fichier Web

Mettre en cache les données permet de réduire cette latence

- Les adresses IP des serveurs DNS TLD ne changent presque pas
  (de ce fait les serveurs racine ne sont presque jamais visités)
- Les sites populaires (*ex* : www.google.fr) qui sont souvent visités

Mise en place d'un cache DNS

- Les serveurs DNS mettent en cache les correspondances récemment obtenues
- Les réponses contiennent un champ TTL (*Time To Live*)
- Les correspondances sont retirées des caches à l'expiration de leur TTL

Fiabilité de DNS

UDP est utilisé pour les requêtes

- La fiabilité est prise en charge par DNS

Les serveurs DNS sont **répliqués**

- Le service de résolution de nom est disponible si au moins un replica est disponible
- La charge des requêtes est répartie parmi les replicas (*load balancing*)

Essayer des serveurs DNS **alternatifs** en cas d'expiration de temporisateur (*timeout*)

- *Exponential backoff* si l'on essaye avec le meme serveur

Même identifiant pour toutes les requêtes

- Le serveur qui répond n'a pas d'importance

Cache négatif

Principe Se rappeler des noms qui n'ont pu être résolus

- Erreurs de frappe comme `wwww.google.fr` et `www.goooogle.fr`
- La requête pour un nom mal saisi reste sans réponse
- Une requête sans réponse est interprétée comme perdue
- Le resolver renvoie une requête 16 fois avant d'abonner !
- Se rappeler des noms non résolus protège les serveurs de DNS (racine et TLD) des requêtes inutiles

DNS Resource Records

DNS est une base de données qui contient des
Resource Records (**RR**)
`(name, value, type, ttl)`

`type = A`

`name` est le nom d'hôte

`value` est l'adresse IP

`type=NS`

`name` est le domaine (ex : `foo.com`)

`value` est le nom du serveur DNS d'autorité pour ce domaine

`type=CNAME`

`name` est un alias pour un nom canonique (réel)
Ex : `www.ibm.com` est en fait `servereast.backup2.ibm.com`

`value` est le nom canonique

`type=MX`

`value` dest le nom du serveur de mail associé à `name`

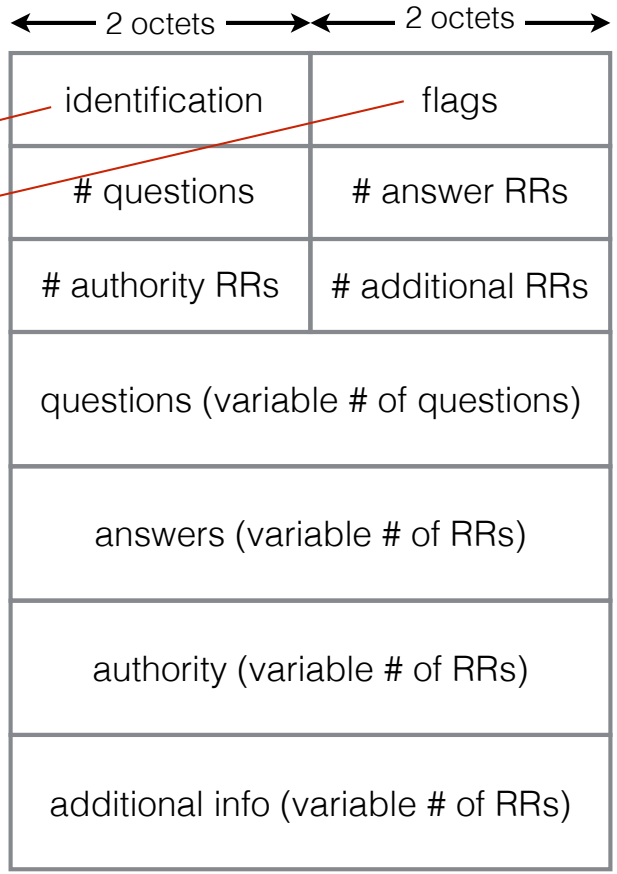
Protocole DNS

Les messages de **requête** et de **réponse** sont ont le **même format**

identification nombre codé sur 16 bits pour identifier la requête
La réponse utilise le même identifiant

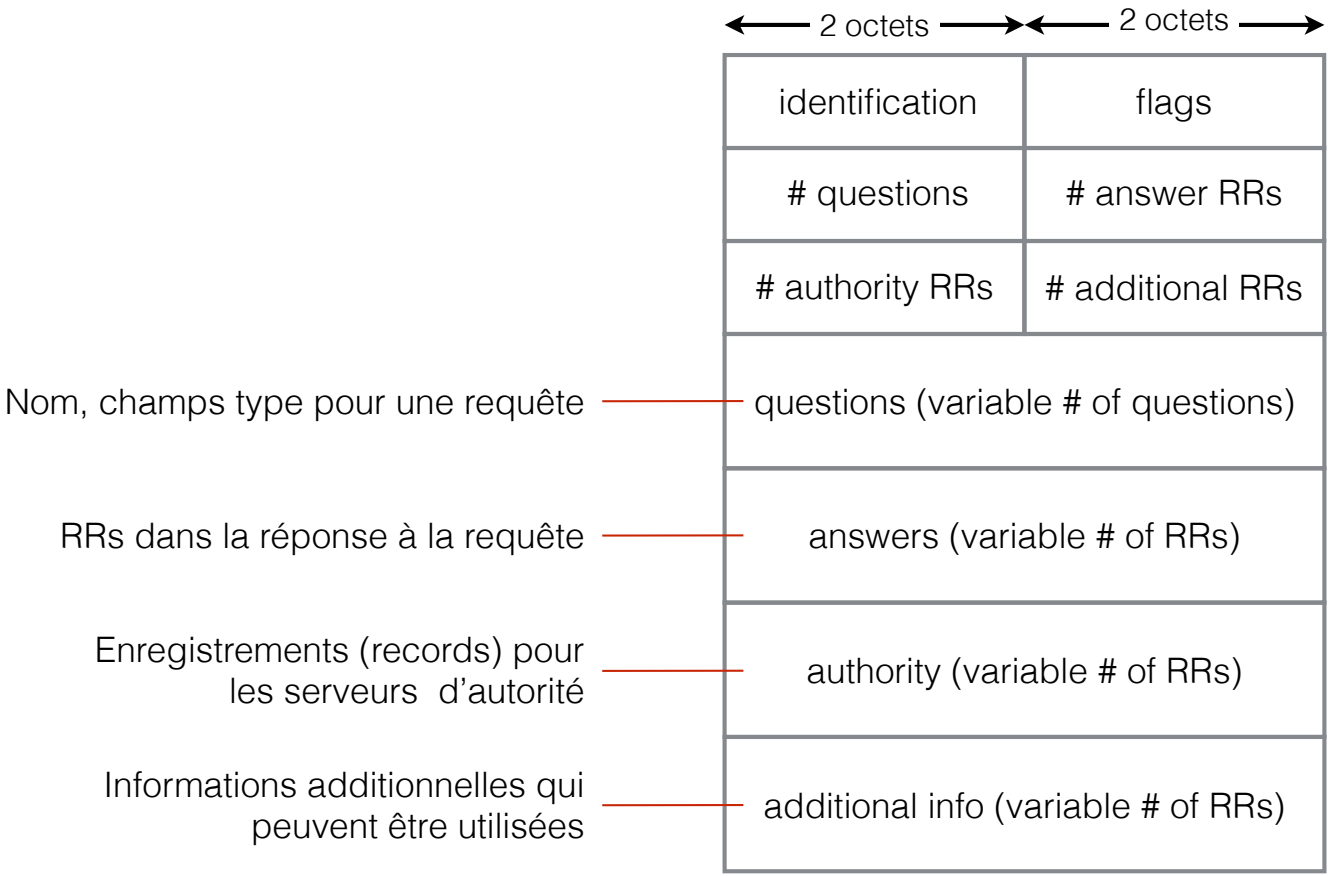
flags

- Requête ou réponse
- Récursion désirée ou non
- Récursion disponible
- Réponse provient dun serveur DNS



Protocole DNS

Les messages de **requête** et de **réponse** sont ont le **même format**



Création et enregistrement d'un nom de domaine

Par exemple Une nouvelle startup "Network Utopia, Inc" souhaite enregistrer un domaine networkutopia.com

Enregistrer le nom networkutopia.com auprès d'un **registrar DNS** (*ex* : Network Solutions)

- Fournir les noms et adresses IP des serveurs de nom d'autorité (primaire et secondaire)
- Le registrar DNS insère deux RRs dans le serveur DNS TLD .com :
  (networkutopia.com, dns1.networkutopia.com, NS)
  (dns1.networkutopia.com, 212.212.212.1, A)

Création d'un enregistrements auprès des serveurs d'autorité de la startup :

- Type A pour www.networkutopia.com
- Type MX (mail) pour networkutopia.com

DNS et requêtes HTTP (1)

- Un utilisateur saisit ou clique sur une URL
  - http://www.google.fr/maps
- Le navigateur extrait le nom du site web (ex : www.google.fr)
  - Le navigateur appelle la primitive gethostbyname() pour découvrir l'adresse IP correspondante
- Le navigateur exécute le code du resolver
  - Une requête est envoyée au serveur DNS local (dont l'adresse IP est connue par configuration manuelle ou par DHCP)
- Le resolver finit par avoir une réponse
  - Si la requête aboutit, l'adresse IP du serveur Web www.google.fr est retournée au navigateur
  - Le navigateur établit une connexion TCP et envoie sa requête HTTP Si le navigateur est configuré avec un proxy Web

DNS et requêtes HTTP (2)

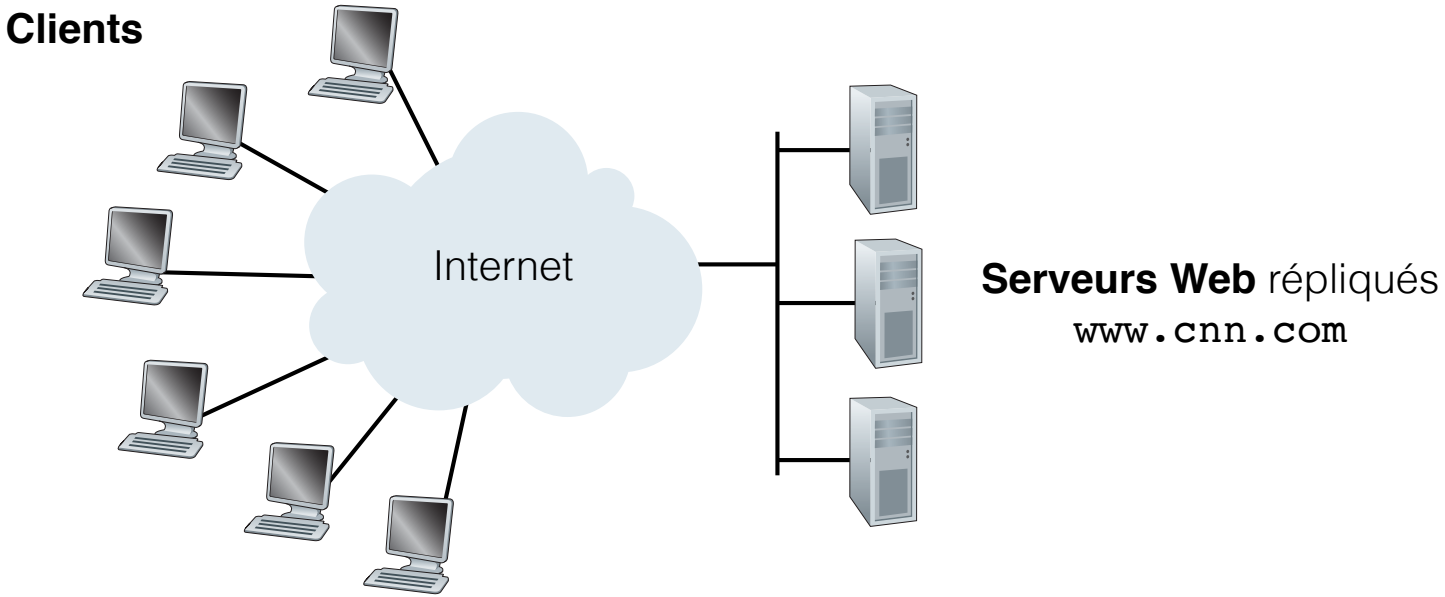
- Souvent, une page Web contient plusieurs objets
  - Par exemple : un fichier HTML avec des images JPG
- Chaque objet a sa propre URL
  - éventuellement situé sur différents serveurs
  - Par exemple :
    - http://www1.myimages.com/image1.jpg
    - http://www2.myimages.com/image2.jpg
- Le navigateur résout l'URL des objets
  - Nécessite plusieurs requêtes DNS …
  - une pour chaque objet si les objets ont des URL différentes
    - www1.myimages.com
    - www2.myimages.com
- Le navigateur établit plusieurs connexions TCP
  - une par objet

Quand est-ce que les requêtes DNS ne sont pas nécessaires ?

- Si le navigateur est configuré avec un proxy Web
  - Le navigateur envoie toutes ses requêtes HTTP au proxy
  - Le proxy prend en charge l'envoi des requêtes DNS
- Les ressources Web demandées sont mises en cache localement
  - Par exemple : Le cache du navigateur contient la page http://www.cnn.com/2015/leadstory.html
  - Si la page est à jour au moment de la requête, inutile de demander à nouveau la ressource, donc inutile d'émettre une requête DNS
- Le navigateur a récemment envoyé une requête DNS pour le nom d'hôte
  - Par exemple un utilisateur a récemment visité www.cnn.com
  - Le navigateur a déjà exécuté gethostbyname()
    - l'adresse IP résultante est peut-être déjà dans le cache

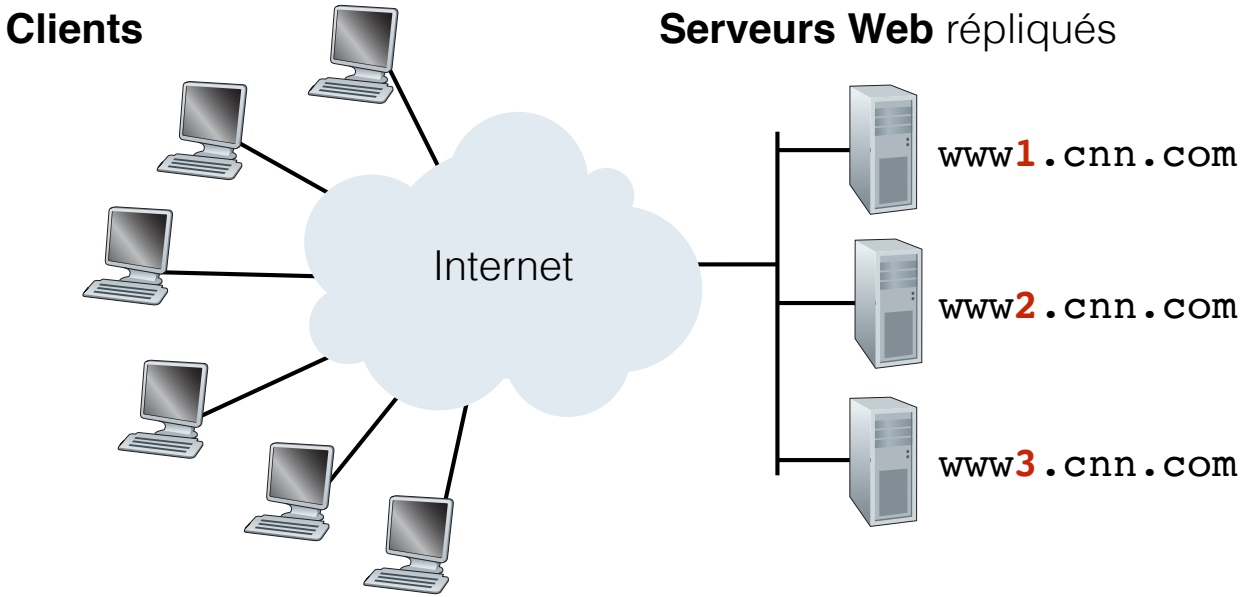
Replicas de serveurs Web

Les sites web populaires peuvent être rapidement surchargés par les requêtes, d'où l'utilisation de plusieurs serveurs



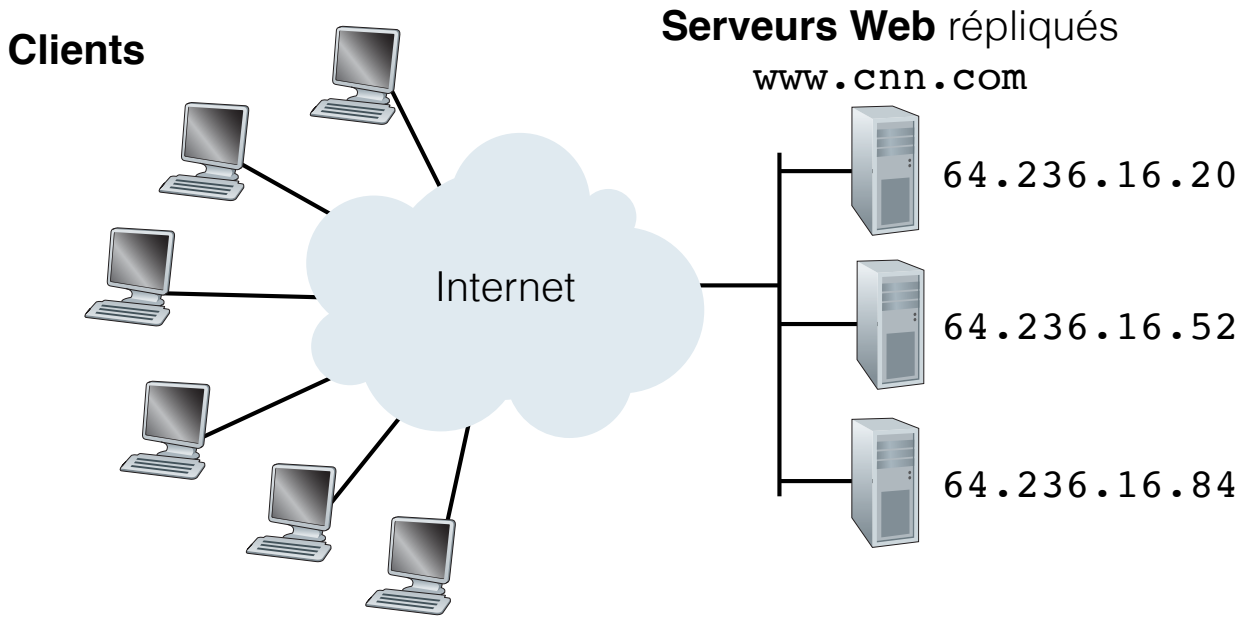
Diriger les clients vers les replicas

Utiliser différents noms pour chaque replica
Demander aux clients de choisir un replica



Diriger les clients vers les replicas ‘adéquates’

Utiliser un seul nom pour tous les replicas
Le choix du replica dépend de l’adresse IP que renvoie le DNS



Le serveur DNS d’autorité retourne :

- l’adresse IP d’un replica ou
- les adresses IP de tous les replicas, le client choisit le premier replica de la liste

Conclusions

- **HTTP** *Hypertext Transfer Protocol*
  - Protocole en mode non connecté
  - Les pages contiennent plusieurs objets
  - Les performances de HTTP dépendent de TCP et du DNS
  - Utilisation de proxys cache pour améliorer les performances
- **DNS** *Domain Name System*
  - Mécanisme d'allocation et de résolution des noms dans Internet
  - Base de données répartie sur plusieurs serveurs
  - Organisation en arborescence des serveurs DNS
  - Utilisation de proxys et de caches pour améliorer les performances