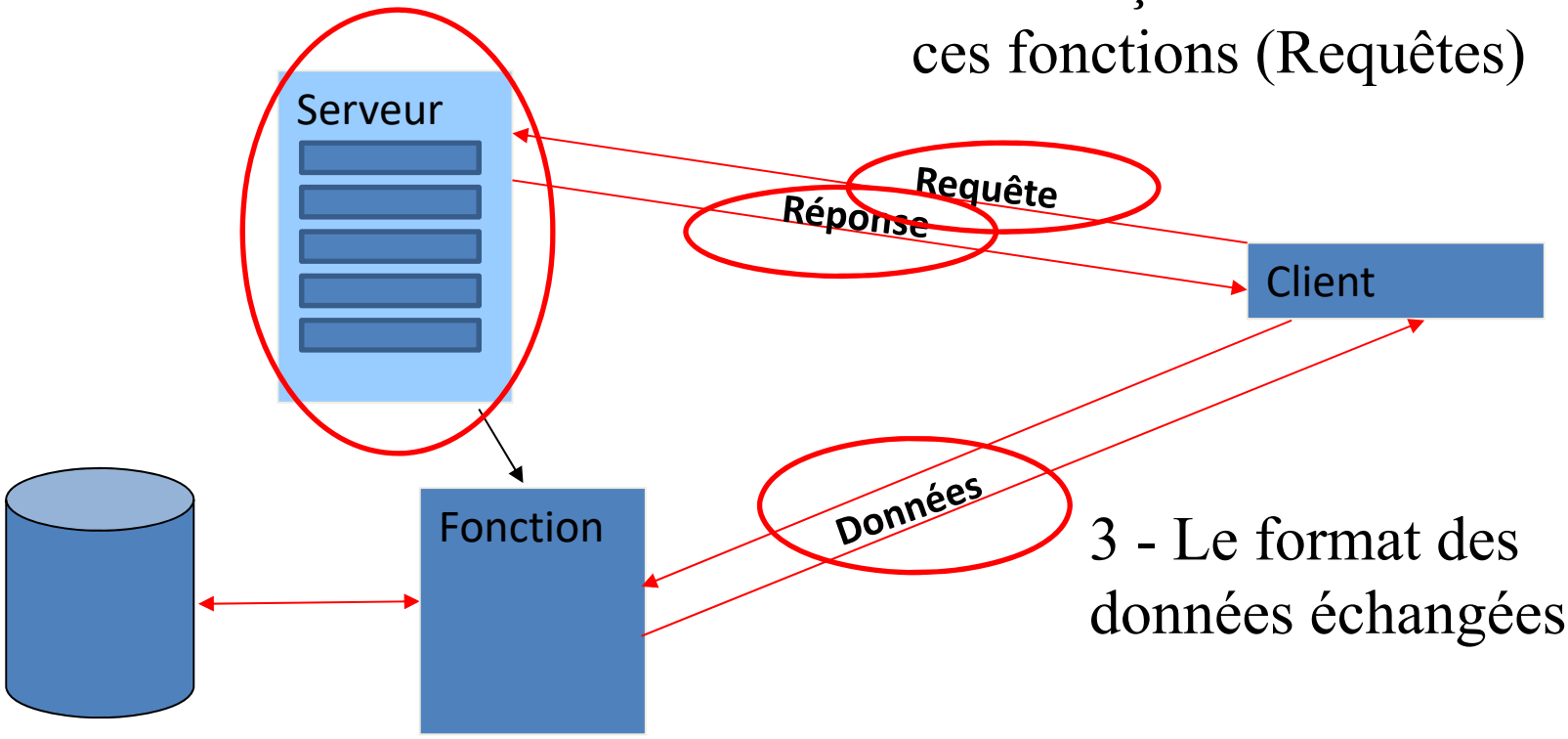
La normalisation des services

Normalisation des services

Normaliser un service, consiste à définir :

1 - Les fonctions du serveur, accessibles aux clients

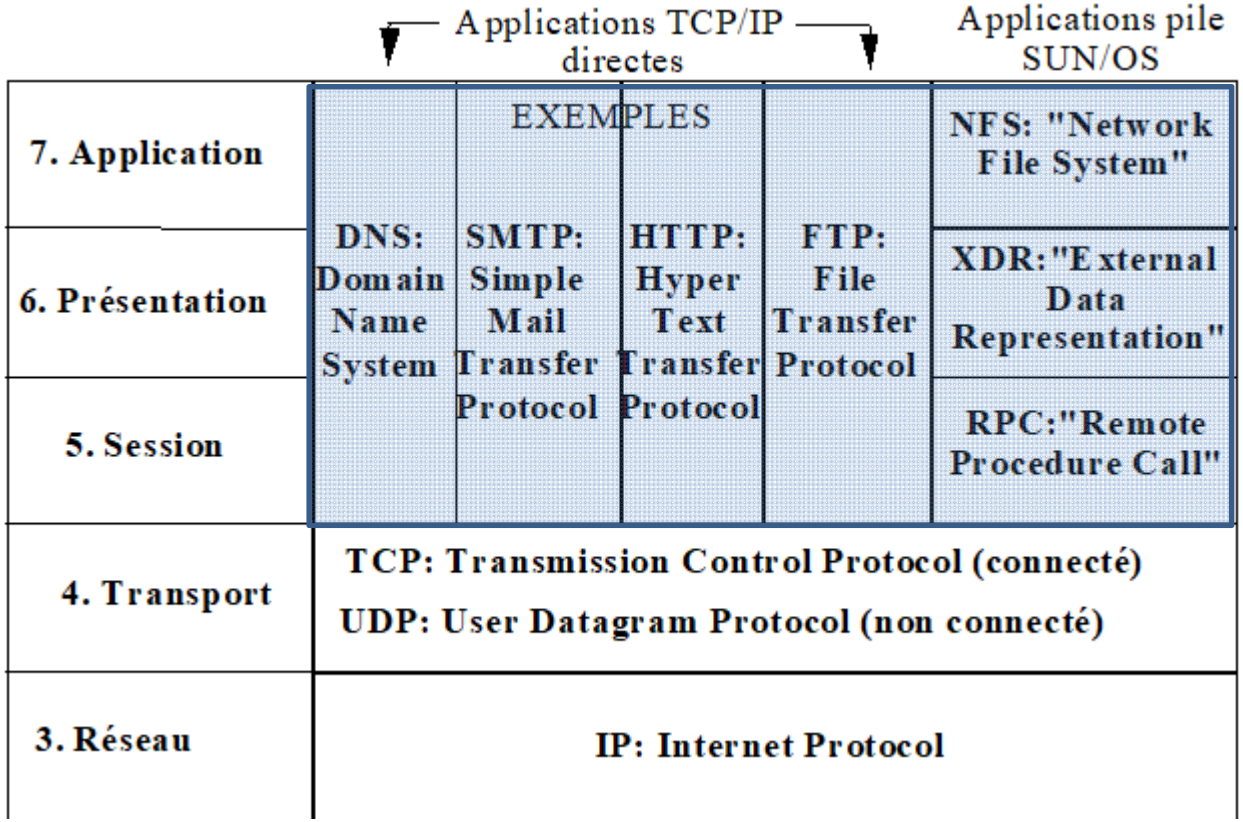
2 - La façon d'accéder à ces fonctions (Requêtes)



3 - Le format des données échangées

Normalisation des services

Tout comme le matériel, l'ISO recommande , dans couches hautes de l'OSI, des services normalisés.



Les descriptions de ces services, sont contenues dans des RFC (Requests For Comment)

Structure d’un service et RFC

Qu’est-ce qu’un RFC ? série numérotée de documents officiels décrivant les aspects techniques d'Internet (Wipipédia)

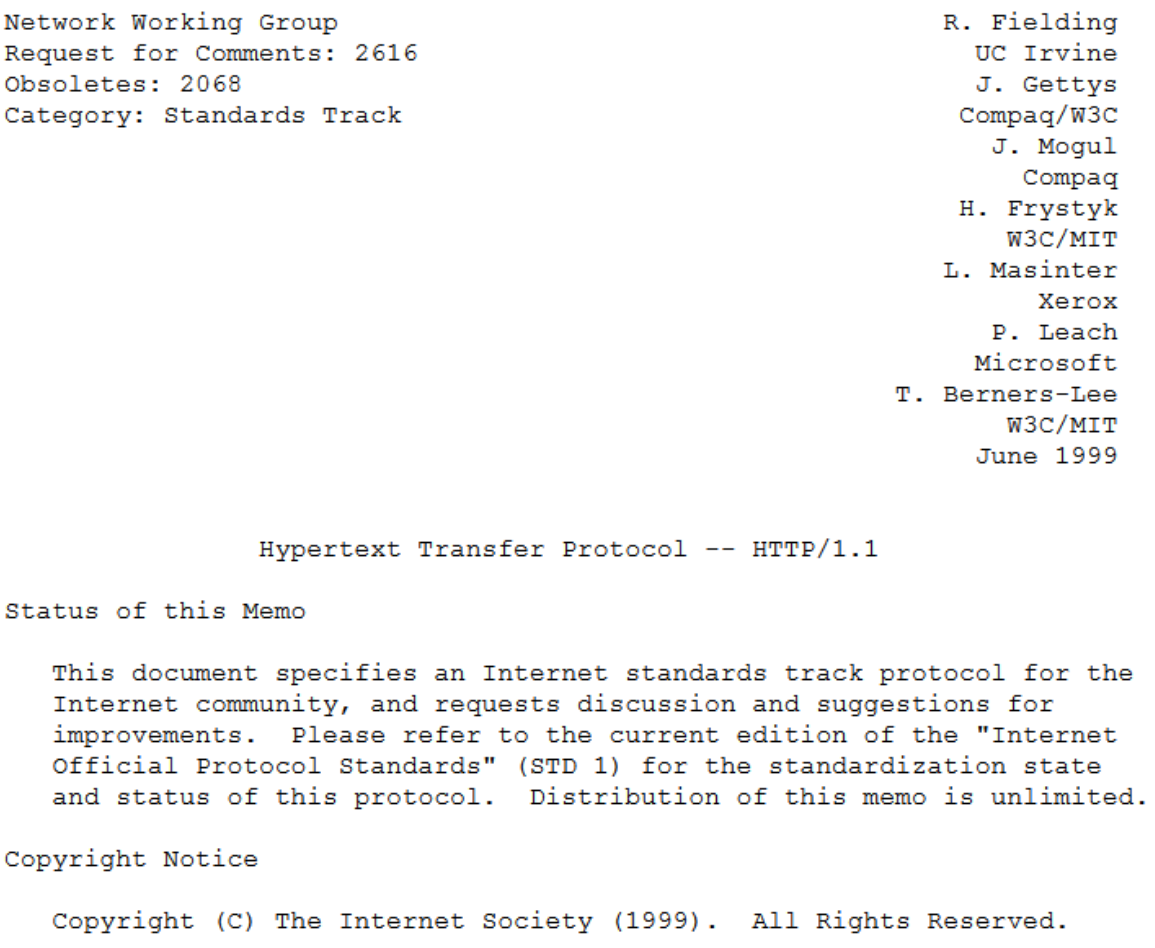
RFC 959 STD 9	FTP - File Transfer Protocol Protocole de la couche applicative. Utilisé pour le transfert fiable de fichiers sur Internet.
RFC 854 STD 8	TELNET - Protocole TELNET Protocole de la couche applicative. Utilisé pour se connecter à un serveur distant, Terminal Virtuel Internet.
RFC 830	DNS - Système réparti pour le service des noms Internet
RFC 827	EGP - Protocole de passerelle extérieure (EGP)
RFC 826 STD 37	ETHERNET - Protocole de résolution d’adresse Ethernet Conversion des adresses de protocole réseau en adresses Ethernet à 48 bits pour la transmission sur matériel Ethernet
RFC 821 STD 10	SMTP - Simple Mail Transfer Protocol Protocole de la couche applicative. Utilisé pour envoyer des e-mails par les logiciels de messagerie électronique (KMail, Messenger, etc.)
RFC 819	DNS - Convention de désignation de domaine pour les applications d’utilisateur de l’Internet

Structure d’un service et RFC

Les RFC sont rédigées sur l'initiative d'experts techniques, puis sont revues par la communauté Internet dans son ensemble.

Plusieurs liens existent :

http://www.rfc.fr/
http://abcdrfc.free.fr/
http://www.ietf.org/rfc.html



Network Working Group
Request for Comments: 2616
Obsoletes: 2068
Category: Standards Track

R. Fielding
UC Irvine
J. Gettys
Compaq/W3C
J. Mogul
Compaq
H. Frystyk
W3C/MIT
L. Masinter
Xerox
P. Leach
Microsoft
T. Berners-Lee
W3C/MIT
June 1999

Hypertext Transfer Protocol -- HTTP/1.1

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Quelques RFC incontournables

Il existe plus de 8500 RFCs

RFC 791 → IP V4
RFC 2460 → IP V6
RFC 2616 → HTTP 1.1
RFC 1866 → HTML 2.0
RFC 1034 /1035 → DNS
RFC 2131 → DHCP
RFC 5321 → SMTP (messagerie)
RFC 1736 → URL

Le premier document RFC date du 7 avril 1969 et est dédié à Arpanet, l'ancêtre du réseau Internet tel que nous le connaissons.

Exemple de service : HTTP

Le protocole HTTP

Protocole de communication client-serveur défini à partir de 1989 pour supporter les échanges de données du 'World Wide Web‘ :
conçu à l’origine pour transporter des textes HTML.

Il est définit par les RFC 1945 (1.0), 2616 (1.1)

Le protocole HTTP

Caractéristiques de base :

- HTTP : un protocole client-serveur basé sur le mode message

→ Tous les échanges sont basés sur un couple de **messages requête réponse.**

→A chaque couple requête réponse est associé une '**méthode'.**

- **Approche asynchrone :** un client n'est pas bloqué par une requête (il peut émettre plusieurs requêtes en parallèle).

Le protocole HTTP

Caractéristiques de base :

- HTTP utilise le **transport fiable TCP** , mais d'autres réseaux sont possibles).

-une ressource est un **ensemble d'informations** identifié par une URL (Le plus souvent un **fichier)**

http	:	//	www.iut.fr	:	80	/	index.html

Protocole HTTP - RFC

Dans les RFC on trouve :

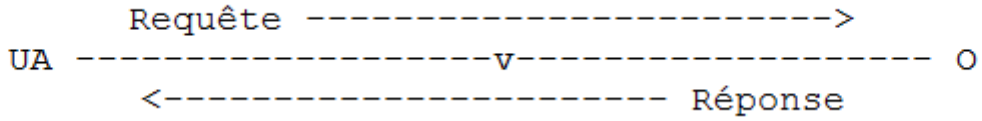
1 – La façon d'accéder au service (URL)

```
http_URL              = "http:" "//" host [ ":" port ] [ chem_abs ]

host                  = <un nom Internet d'ordinateur valide ou une adresse IP
                      (sous forme numérique), comme définie en Section 2.1 de la
                      RFC 1123>

port                  = *DIGIT
```

2 – La structure générale des échanges



Le protocole HTTP

Logique des échanges HTTP

1 - Le client HTTP/1.0 ouvre une **connexion TCP (le numéro de port par défaut de** http est 80, un autre numéro est possible).

2 - Le client envoie **un message de requête**(exemple type : obtenir une ressource en fonction de différents paramètres).

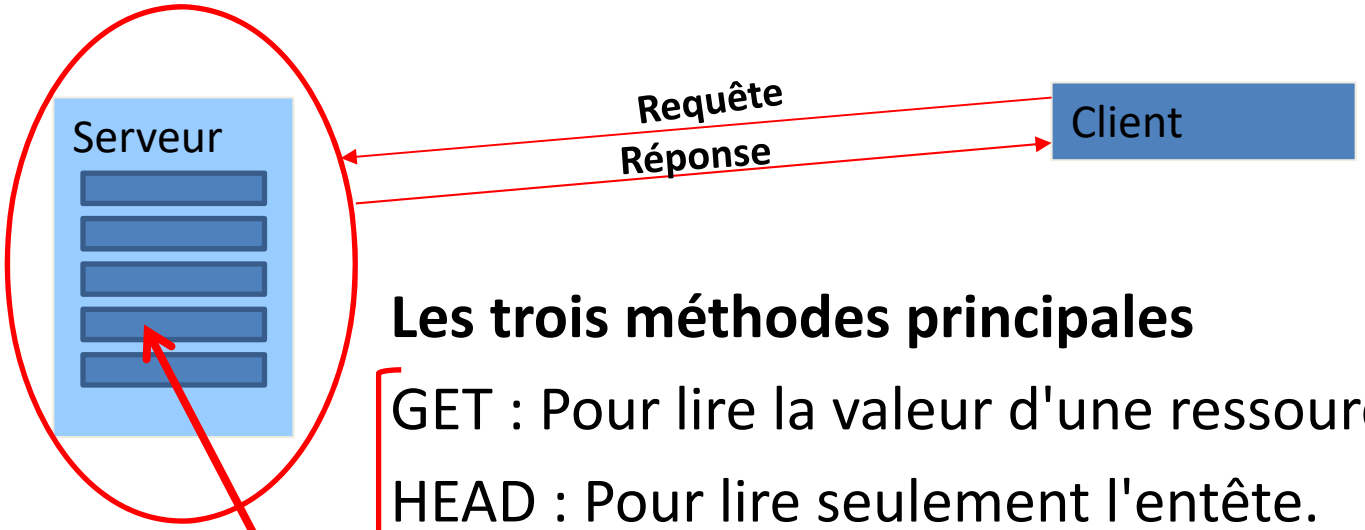
3 - Le serveur répond par **un message de réponse (la ressource demandée).**

4 - Le client **ferme la connexion TCP**

Remarque : En HTTP chaque ressource nécessite sa propre connexion TCP.

Le protocole HTTP

La liste des méthodes HTTP



Les trois méthodes principales

GET : Pour lire la valeur d'une ressource.

HEAD : Pour lire seulement l'entête.

POST: Pour faire exécuter un programme distant.

Les méthodes annexes

PUT : Pour envoyer une ressource sur un site distant.

DELETE : Pour détruire une ressource distante.

LINK : Pour établir un lien entre deux ressources

Protocole HTTP - RFC

3 – La liste des requêtes

```
Méthode                 = "GET"                 ; Section 8.1
                        | "HEAD"                ; Section 8.2
                        | "POST"                ; Section 8.3
                        | nom_de_méthode
```

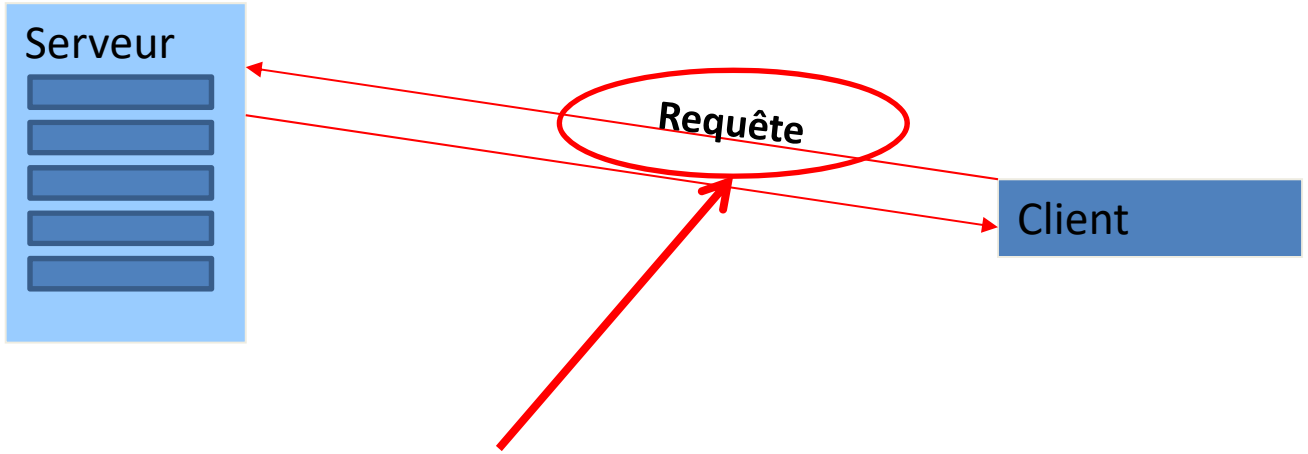
4 – La description des requêtes

8.1 GET

La méthode GET signifie "récupérer" le contenu quel qu'il soit de la ressource (sous forme d'une entité) identifiée par l'URI-visée. Si l'URI-visée identifie un processus générant dynamiquement des données, ce sont les données produites qui sont renvoyées dans l'entité au lieu du source de l'exécutable appelé, sauf si ce texte lui-même est la sortie du processus.

Le protocole HTTP

Le format de la ligne des requêtes



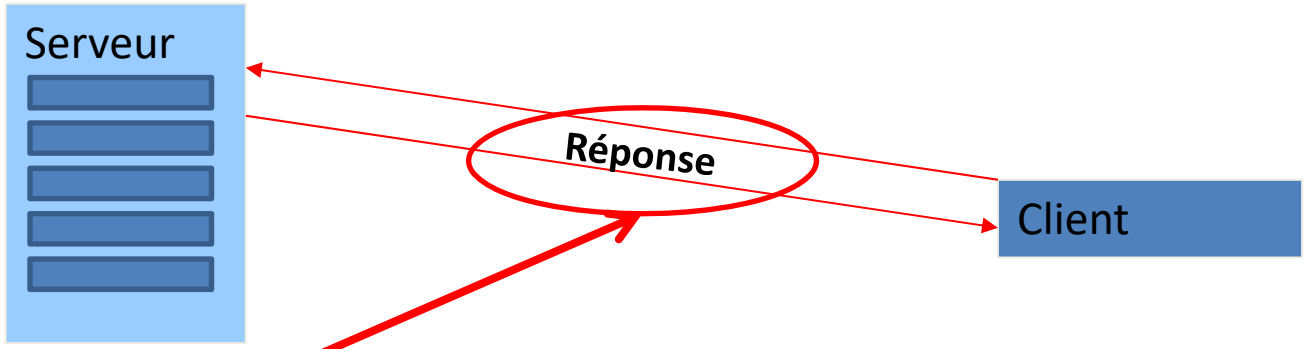
Trois parties séparées par des espaces

- Un **nom de méthode,**
- Le **chemin d'accès à la ressource,**
- La **version du protocole HTTP.**

Exemple : GET /chemin HTTP/1.0

Le protocole HTTP

Le format des réponses



La réponse se compose de trois parties :

. Un **code retour,**

. Un entête,

. Un **texte (à balises) représentant la page demandée.**

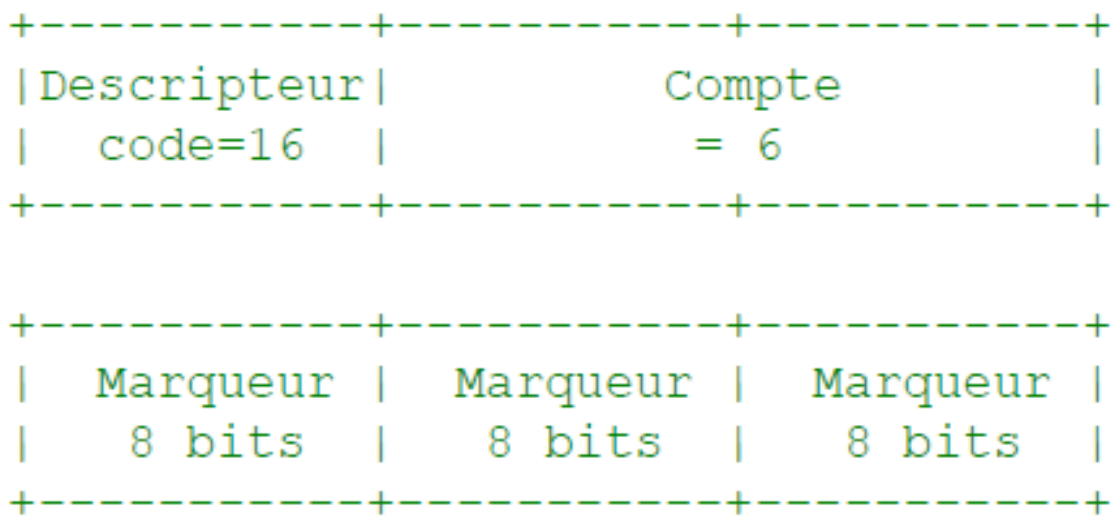
HTTP/1.1 200 OK

Date: Wed, 25 Oct 2000 10:02:12 GMT
Server: Apache/1.3.6 (Unix) PHP/3.0.7
Last-Modified: Wed, 18 Oct 2000 11:07:00 GMT
ETag: "ac-221e-39ed8454"
Accept-Ranges: bytes
Content-Length: 8734
Connection: close
Content-Type: text/html

<html>
<head>
<link rel="stylesheet" …….
……

Structure d’un service et RFC

7 – Le format des données à échanger



Le protocole HTTP

Les codes réponses les plus courants

"200" ; OK
"201" ; Created
"202" ; Accepted
"204" ; No Content
"301" ; Moved Permanently
"302" ; Moved Temporarily
"304" ; Not Modified
"400" ; Bad Request
"401" ; Unauthorized
"403" ; Forbidden
"404" ; Not Found
"500" ; Internal Server Error

Protocole HTTP - RFC

5 – La liste des réponses

```
Code_état   = "200" ; OK                        OK
            | "201" ; Created                   Créé
            | "202" ; Accepted                  Accepté
            | "204" ; No Content                Pas de contenu
            | "301" ; Moved Permanently        Changement définitif
            | "302" ; Moved Temporarily        Changement temporaire
            | "304" ; Not Modified             Non modifié
            | "400" ; Bad Request              Requête incorrecte
            | "401" ; Unauthorized             Non autorisé
            | "403" ; Forbidden                Interdit
            | "404" ; Not Found                Non trouvé
            | "500" ; Internal Server Error    Erreur interne serveur
            | "501" ; Not Implemented          Non implémenté
            | "502" ; Bad Gateway              Erreur de routeur
            | "503" ; Service Unavailable      Indisponible
```

6 – La description des réponses

9.2 Succès 2xx

Cette classe précise que la requête du client a été correctement transmise, interprétée, et exécutée.

200 OK

La requête a abouti. L'information retournée en réponse dépend de la requête émise, comme suit:

Le protocole HTTP

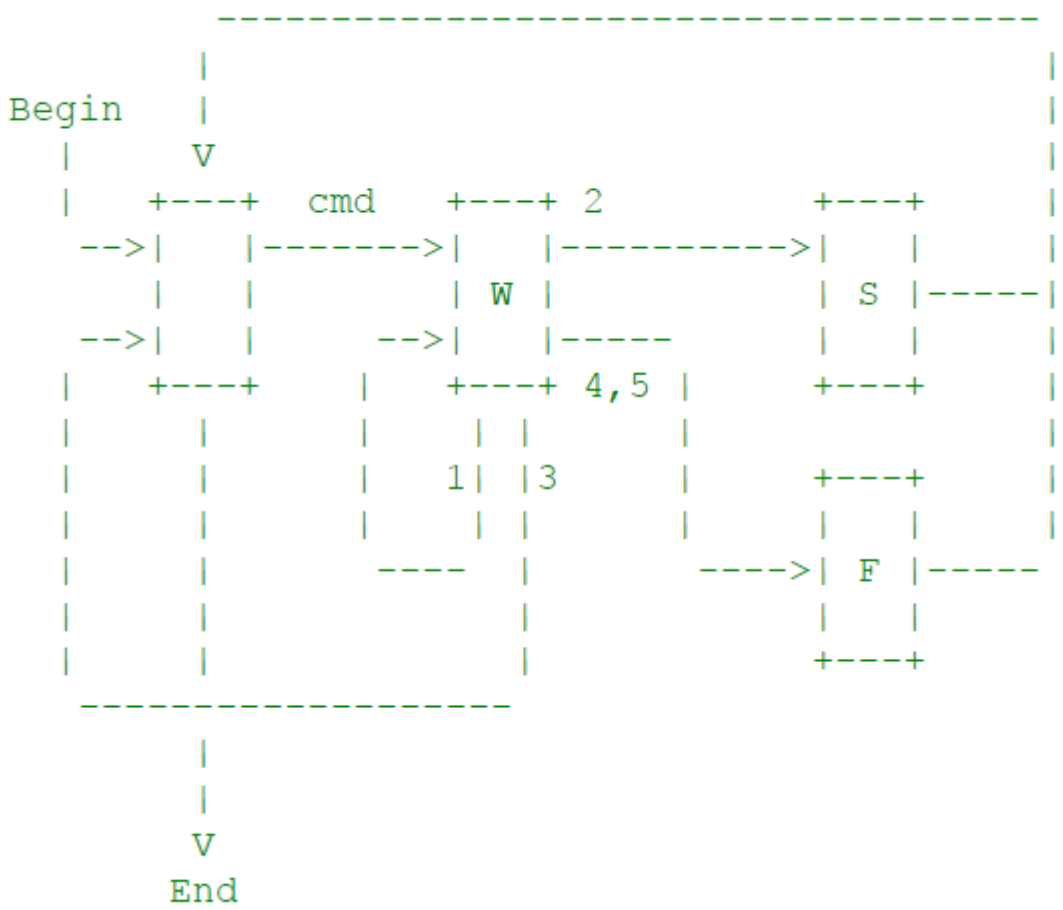
Exemple complet d'échange

Poste A ***Poste B***
10.20.1.1 ***66.249.85.99***

TCP 33667 > http [SYN] Seq=0 Ack=0 Win=5840 Len=0 →
← *TCP http > 33667 [SYN, ACK] Seq=0 Ack=1 Win=8190 Len=0*
TCP 33667 > http [ACK] Seq=1 Ack=1 Win=5840 Len=0 →
HTTP GET / HTTP/1.0 →
← *TCP http > 33667 [ACK] Seq=1 Ack=426 Win=7765 Len=0*
← ***HTTP HTTP/1.0 200 OK (text/html)***
TCP 33667 > http [ACK] Seq=426 Ack=1431 Win=8580 Len=0 →
← ***HTTP Continuation***
TCP 33668 > http [ACK] Seq=520 Ack=2062 Win=11440 Len=0 →
← ***HTTP Continuation***
← ***HTTP Continuation***

Protocole HTTP - RFC

8 – Les échanges types – Diagrammes d'états



Le protocole HTTP

Il existe de nombreuses implémentations du protocole HTTP

- Apache (Apache Groupe)
- IIS (Microsoft)
- Nginx (Logiciel libre)
- LiteSpeed (LiteSpeed Technologies)

Remarques :

- En 2021 on référence plus de 1,88 milliards de sites webs
  (1 milliard en 2017 → 88% progression).
- Parts de marché 2022 : Apache 40,17 % , Nginx 27,72%, IIS 11,03%

(source hostadvice 2022)

Conclusion

Le succès de l'internet est essentiellement du :

- Au grand nombre de services proposés,
- A l'amélioration des interfaces, facilitant l'accès aux services.

Tous ces services, permettent à plus de 60% de la population mondiale (soit 5 milliards de personnes – chiffres 2022) de profiter de l'internet.

MAIS …

L'accès aux services, en bas niveau, reste toujours possible.
Avec l'augmentation du nombre d'internautes et la multiplication des formations réseaux , il y a de plus en plus de personnes qui exploitent des failles connues dans les services.

Ceci est de nature à mettre en danger un grand nombre de machines :
on parle de sécurité des systèmes

Le protocole DNS

Solution avec annuaires distribués: le DNS / RFC 1034, 1035

Création d'un service **d'annuaire distribué** (base de données distribuée d'informations) accessible au moyen **d'un espace de nommage hiérarchique unifié**.

Fonction principale= correspondance:

nom logique de site <-> adresse IP

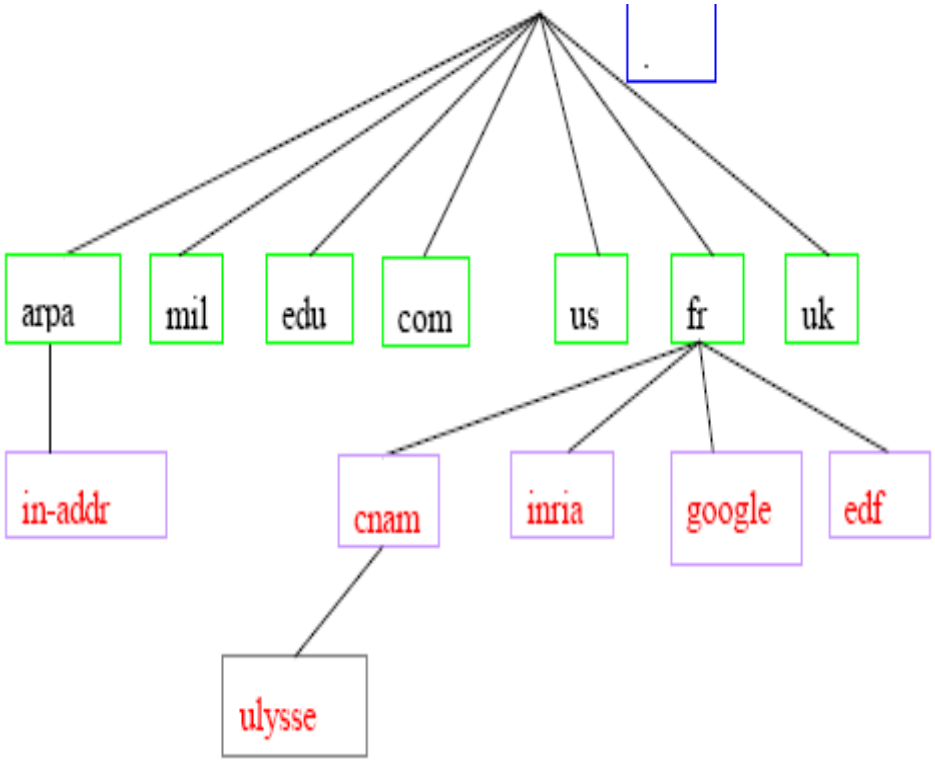
+ Différentes autres fonctions d'accès.

Utilisable par tout système (petit ou grand)

. une procédure client (le **résolveur**)

. et un **serveur** DNS qui répond.

« Domain Name Space »
Une hiérarchie de noms logiques



Niveau le plus haut: Plusieurs centaines de noms de domaines.

Niveaux intermédiaires: Des noms de domaines composés **d'ensembles de ressources**

Niveau des feuilles: des noms **d'hôtes** ('ulysse', 'savitri') ou **de services** ('www', 'ftp').

Les domaines de plus haut niveau (TLD 'Top Level Domains')

Domaines génériques (organisationnels)

.com : Organismes commerciaux (Verisign)
.net : Prestataires réseaux (Verisign)
.aero : Industries aéronautiques (SITA)
.biz : Affaires (NeuLevel, Inc).
.coop : Associations cooperative (Dot Cooperation LLC).
.info : Orgs d'information (Afilias Limited)
.museum : Musées (Museum Domain Management Association).
.name : Individus (Global Name Registry).
.org : Autres organisations (Verisign)

Domaines non ouverts

.edu : Institutions d'éducation US .fr : France
.gov : Organisations gouvernement US .uk : Grande-Bretagne
.mil : Armée US
.int : Organisations internationales.

Codes de pays ISO 3166

La structure des noms

Ils sont construits en suivant l'arbre **des feuilles vers la racine** qui est notée '.'

- On sépare les noms intermédiaires par des .
Exemple : cnam.net..

- Un nom de domaine a **moins de 63 cars**.

- Un nom complet a **moins de 255 cars**.

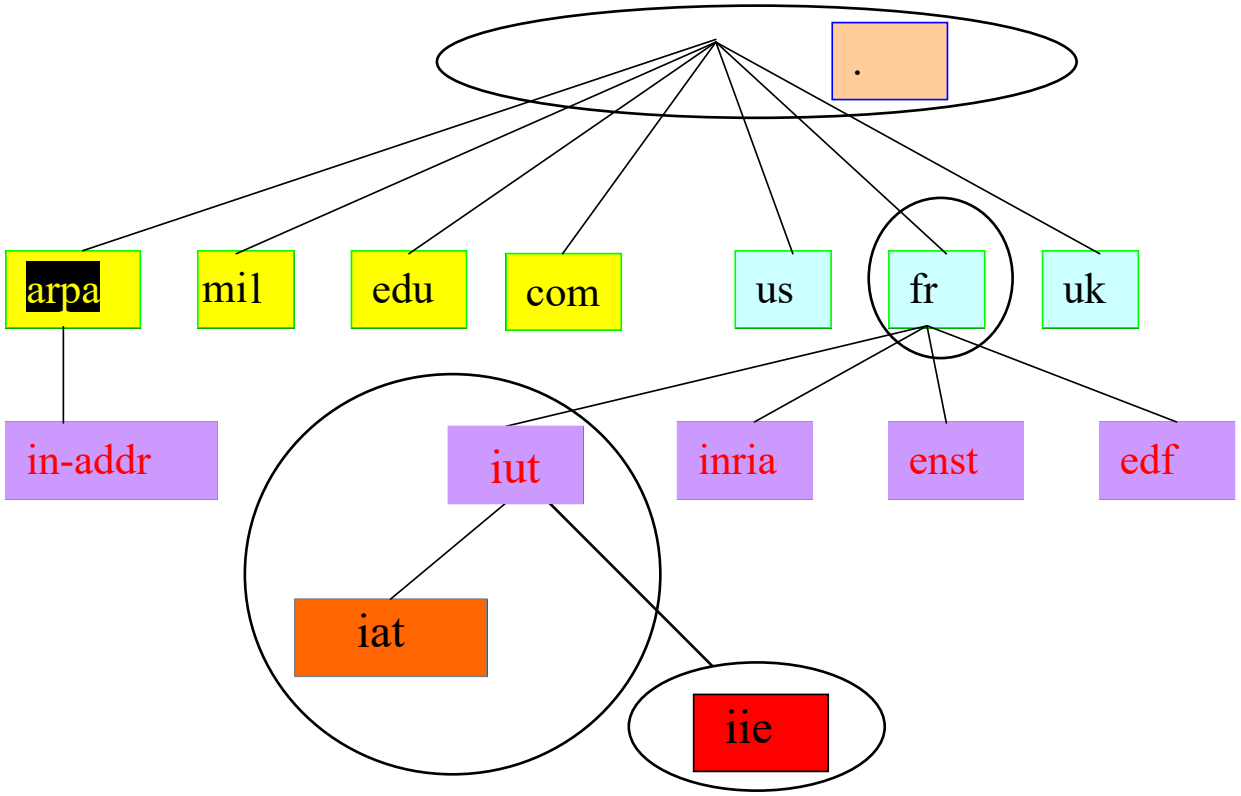
- La casse est **non significative**: iut = Iut = IUT.

Les serveurs DNS et la résolution des requêtes

Notion de zone

Une **zone** = **unité d'administration** (tous les membres d'une zone sont servies par un même serveur).

Une zone regroupe **un ensemble de domaines voisins** qui ne se recouvrent pas.



Les serveurs DNS et la résolution des requêtes

Notion de serveur primaire

Une zone est servie par un (ou plusieurs) serveurs **primaires** alimentés directement en informations par l'administrateur système.

Les fichiers d'informations des serveurs **primaires 'font autorité'** ('Authoritative Answer').

Notion de serveur secondaire

Pour des raisons de **performances** (partage de charge) et de **fiabilité** (tolérance aux pannes de serveurs) on créé des serveurs secondaires.

Les serveurs secondaires ne sont pas alimentés directement mais **recopient périodiquement** en utilisant TCP la base d'informations DNS d'un serveur primaire.

La base de données distribuée du DNS

Le DNS gère une **base de données répartie**

. D'unités d'informations baptisées **RR ‘Resources Records’**

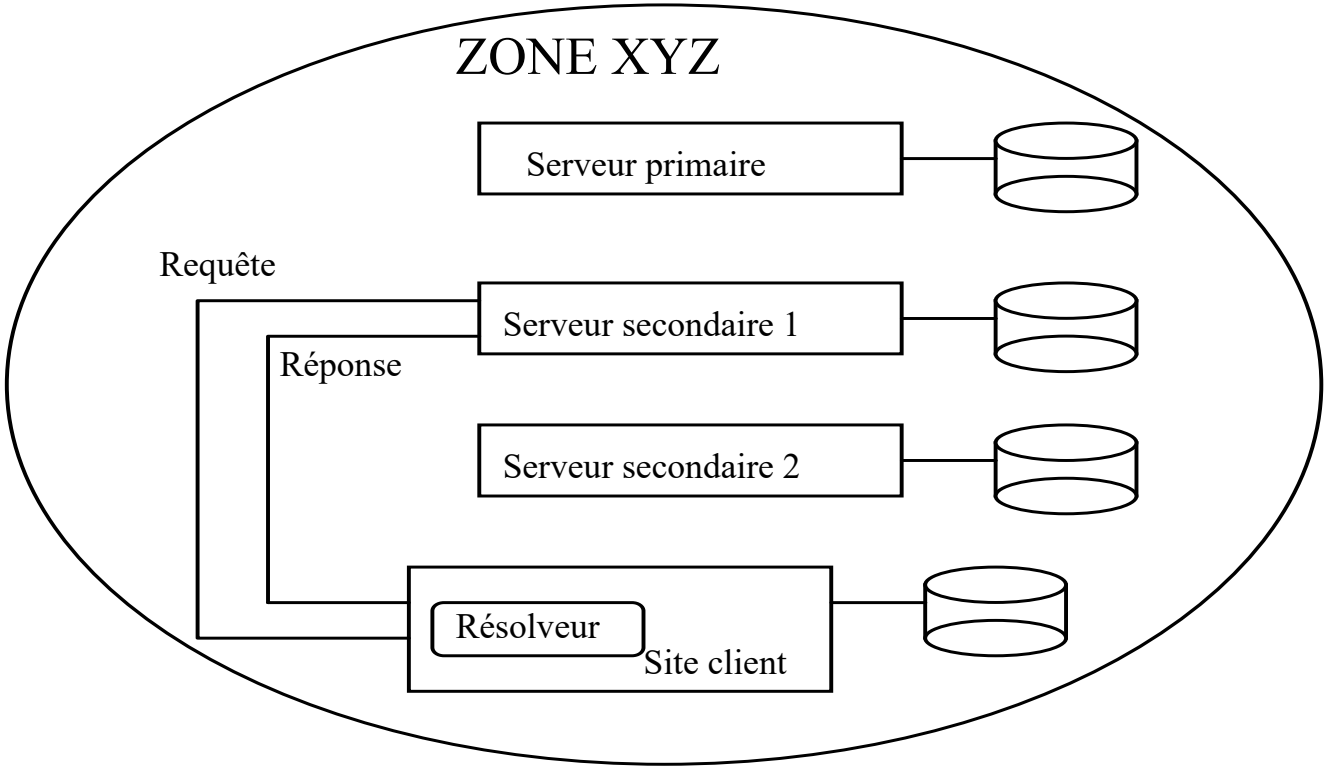
. Pour satisfaire des besoins **variés** de stockage et des protocoles différents.

RR = {nom_de_domaine, durée_de_vie, classe, type, valeur}

- **nom de domaine** (‘**name**’) = un nœud de l'arborescence.
- **durée de vie** (**TTL ‘Time To Live’**) = durée de validité de l'information dans un cache (nombre entier de secondes).
- **classe** (‘**class**’) = protocole utilisateur (essentiellement **In** pour Internet).
- **Type** = **type de donnée** stockée : type adresse A, type serveur de noms d'un domaine NS, type relais de courrier MX.
- **valeur** (‘**rdata’**) = données significatives associées au type : une adresse, un nom de domaine, une chaîne de caractères.

Notion de client DNS

On appelle résolveur une procédure (exemple ***gethostbyname ou gethostbyaddr)*** invoquée sur un client pour un accès DNS.



Notion de cache

Les serveurs primaires ou secondaires conservent dans un cache les réponses à des requêtes récentes :

. pour **gagner du temps**.

. pour éviter de **charger le réseau**.

Problème de la cohérence des caches: que se passe t'il quand l'administrateur modifie le serveur primaire.

Solution DNS : **cohérence très faible** → Simple limitation de **la durée de vie** dans un cache (notion de TTL 'Time To Live').

.Les données stockées **changent lentement** (cohérence faible suffisante).

. On donne la **priorité à la rapidité d'**accès sur la garantie de cohérence.

La résolution des requêtes

La résolution d'une requête est fondée sur une recherche qui peut **concerner plusieurs serveurs avant de trouver la réponse**.

- Le résolveur s'adresse tout d'abord à un **serveur DNS local** (un serveur primaire ou secondaire du domaine courant) **en présentant une question** concernant un nom de domaine.

. Si le serveur **local connaît la réponse** (dans sa base de données ou dans son cache) retour au client de la réponse.

. Si la requête porte sur un sous domaine : **Recherche dans les sous domaines.**

. Sinon **recherche sur d'autres serveurs DNS** qui peuvent être parcourus selon deux méthodes :

- Recherche **récursive**.
- Recherche **itérative**.

DNS Avantages

Un système **d'annuaire** distribué au niveau mondial qui remplit très bien son rôle initial

Une organisation en **arbre** simple et efficace.

Très utilisé (standard de facto), **très rodé**.

Les problèmes de **performances** et de **sûreté** sont résolus par les serveurs **primaires** et **secondaires** mais aussi par l'utilisation de **caches**

DNS Limites

- Faiblesse des mécanismes **de sécurité** dans la version de base: correction en cours de déploiement, nombreuses **RFC DNS sécurisé.**

- **D'autres besoins** comme la distribution de nouveaux types d'informations nécessaires aux utilisateurs ne sont pas couverts

Exemple : identificateurs d'usagers 'login', dictionnaires de données, …

=> Le DNS n'est pas très facilement adaptable.

=> Emergence de plusieurs autres systèmes **de serveurs d'annuaires** (LDAP, NIS+, UDDI, …).