IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Architecture Client-Serveur

Connaissances de base pour comprendre le Web

TAF DCL – DEVELOPPEMENT COLLABORATIF DE LOGICIELS

Issam REBAI – Équipe 3S – Département informatique – Campus de Brest
issam.rebai@imt-atlantique.fr

Version 0.2 [R.2020.04.02]

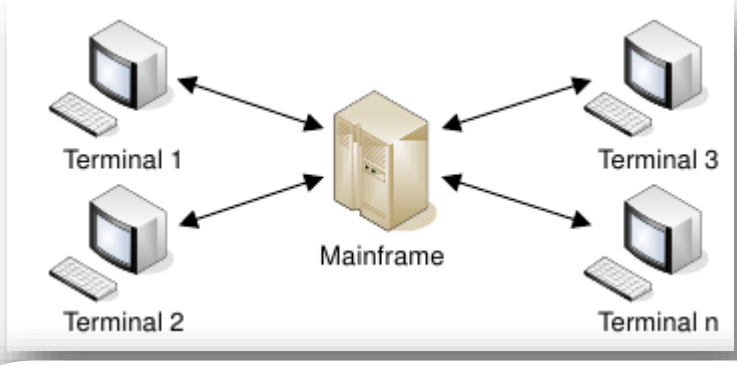
IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Définition
Termes
Typologie d'architectures
Dialogue Client – Serveur
Protocole HTTP

AU SOMMAIRE

Retour historique

- L'informatique centralisée
  - Modèle « *mainframe* »
    - Terminaux : clavier + écran + imprimante pour l'interaction
    - Super ordinateur : traitement + stockage des données
    - Communication : protocole RS-232, …
- Apparition des réseaux : Internet

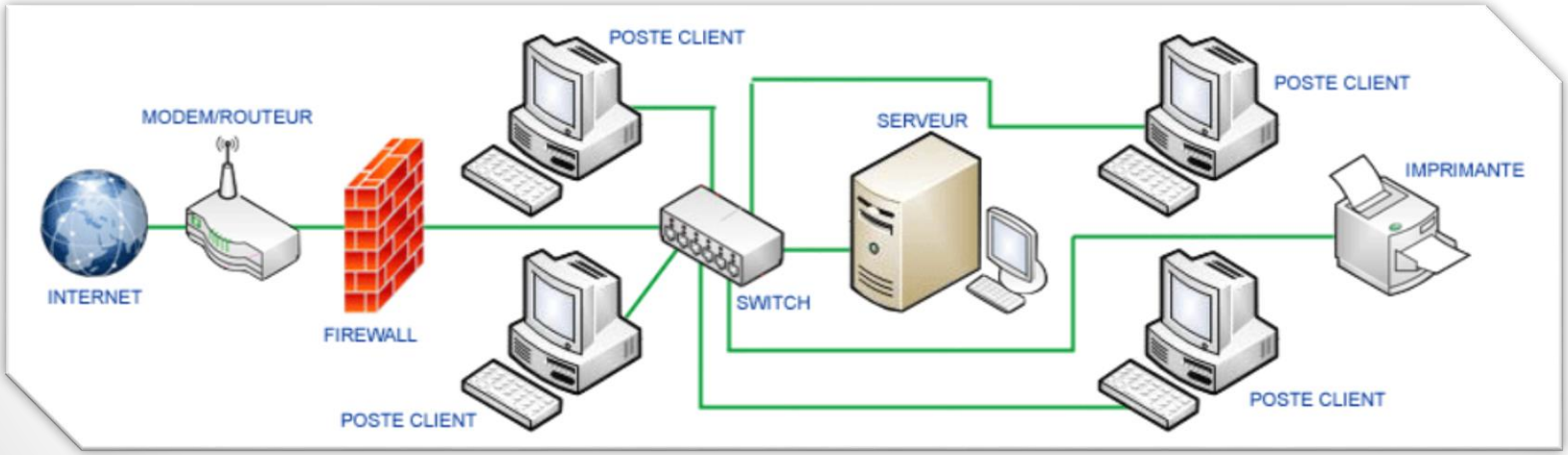



Mainframe :Whirlwind au MIT (1948)

Terminal
IBM 3270

Définition de l'architecture client-serveur

- C'est un mode de communication entre plusieurs entités d'un réseau
- Des machines clientes contactent un serveur (machine généralement puissante), qui leur fournit des services
- Le client pose une question (ou donne un ordre)… et le serveur répond à la question (ou obéit)



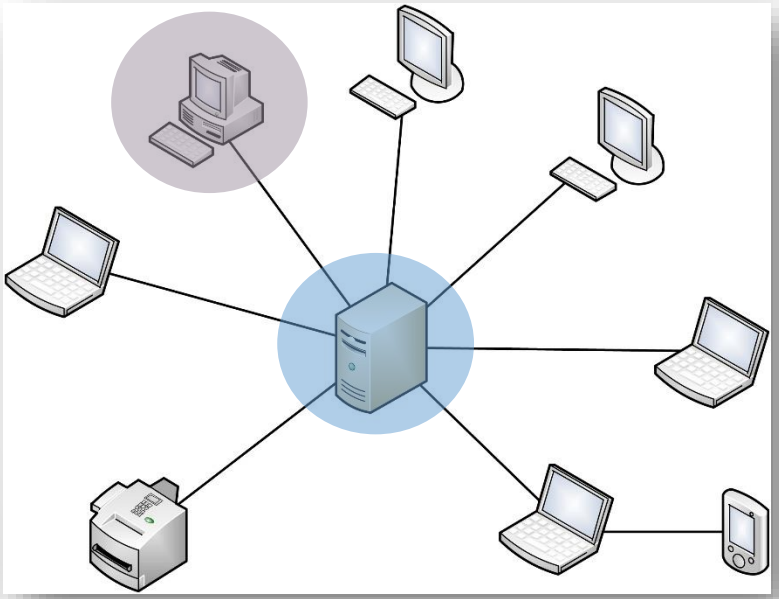
Source: https://www.supinfo.com/articles/single/2519-architecture-client-serveur

Termes

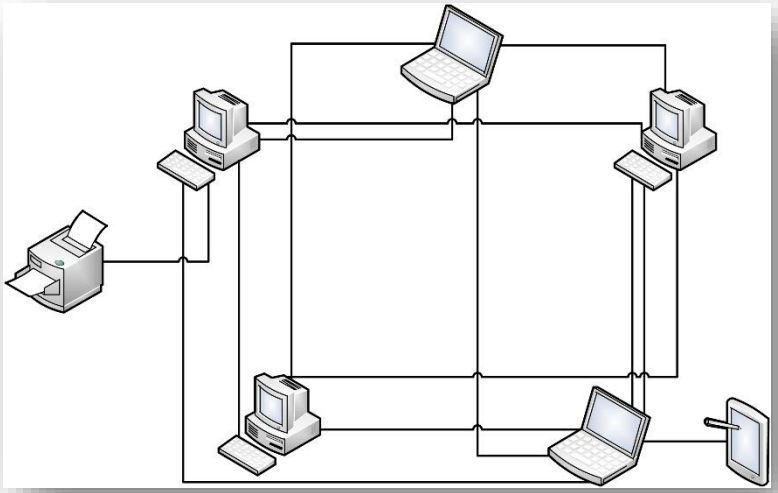
- *Client* : processus actif demandant l'exécution d'un service à un autre processus par envoi d'une requête contenant le descriptif de l'opération à exécuter.
  Le client reste en attente de la réponse à la requête (un message en retour).
- *Serveur* : processus passif en attente des requêtes des clients. Il exécute une opération sur demande d'un client, et lui transmet le résultat.
- *Requête* : message transmis par un client à un serveur décrivant l'opération à exécuter.
- *Réponse* : message transmis par un serveur à un client suite à l'exécution d'une opération, contenant le résultat de l'opération.

Typologie d'architectures C/S

Réseau Client – Serveur
(centralisé)
Le serveur centralise toutes les ressources du système d'information



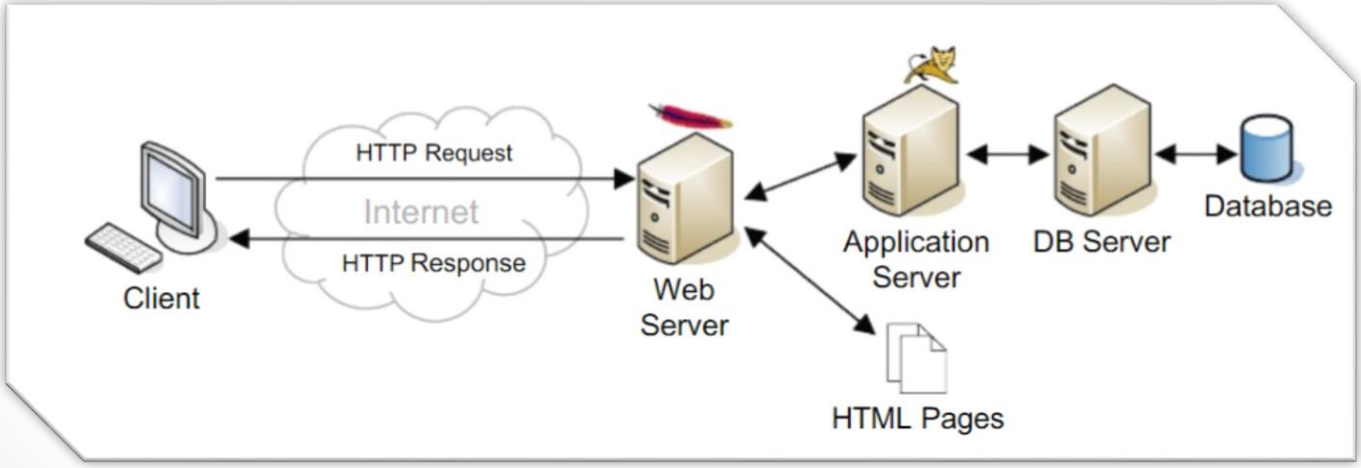
Réseau pair à pair (P2P : « *peer to peer* »)
(décentralisé)
L'application installée sur chaque machine est client et serveur à la fois (« *Servent* »)



Typologie d'architecture C/S

- Architecture à 2 niveaux (2-tiers)
  - Client demande une ressource (fichier)
  - Serveur fournit la ressource qu'il héberge
- Architecture à 3 niveaux (3-tiers)
  - Client (navigateur client, programme)
  - Serveur d'application (*middleware*) fournit la ressource en sollicitant un autre serveur
  - Serveur de données qui fournit au middleware les données nécessaires pour répondre au client
- Architecture à n niveaux (multi-tiers)
  - Multiplier les couches pour spécialiser les serveurs par tâche ➔ plus de flexibilité, sécurité, performance, complexité

Exemple :



Les parties d'une application Client – Serveur

Présentation

- Interfaces textuelles ou graphiques
- Interactions, validation, etc.

Logique d'application

- Traitements
- Classes et fonctions

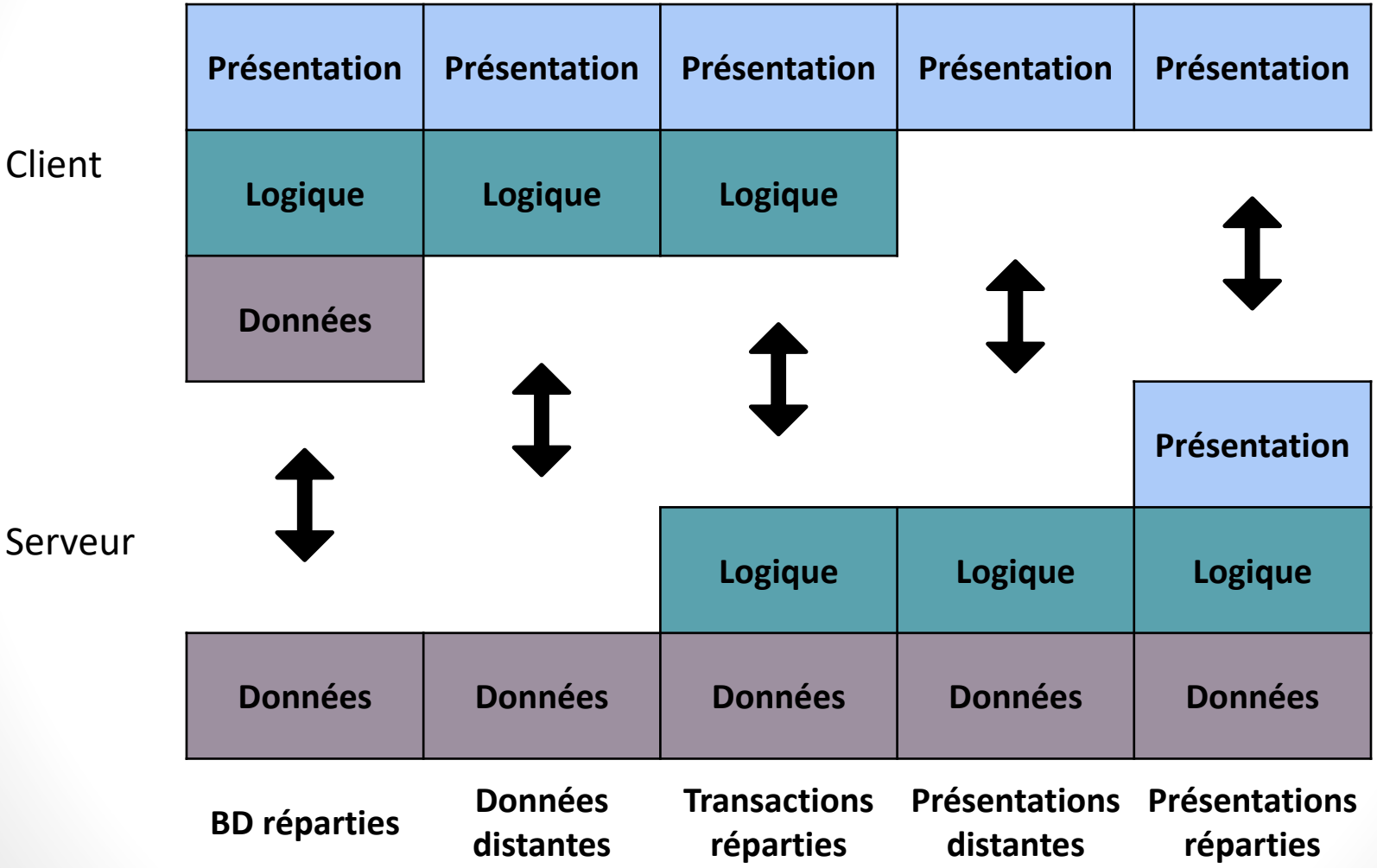
Accès aux données

- Stockage fichiers
- Base de données
- Etc.

Types de clients

- Client « léger »
  - Client : navigateur Web
    - Fonctionnalité minimale
  - Serveur : application entièrement sur le serveur
    - Beaucoup de charge sur le serveur (et le réseau)
- Client « lourd »
  - Client : OS + application « *standalone* »
    - Le client effectue une bonne partie du traitement : stocke les données et les applications localement
  - Serveur : traitement
    - Le serveur a une charge plus allégée: stocke les données mises à jour
- Client « riche »
  - Client : interface graphique évoluée (comme client lourd) + moins de traitement
  - Serveur : majorité du traitement, réponse semi-finie envoyée au client

Répartition des charges sur une architecture C/S à 2 niveaux

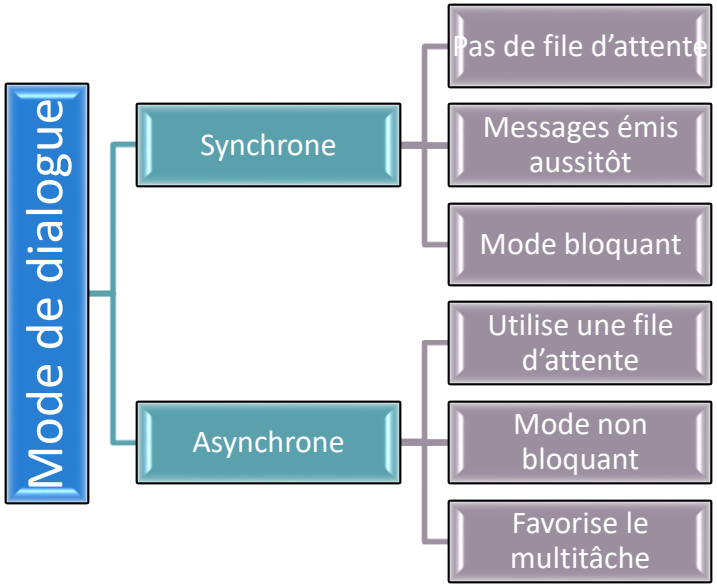


Le dialogue Client ⇔ Serveur

- Le dialogue Client – Serveur ne peut s'établir sans la définition d'un protocole de communication commun
- Protocole de communication ?

« Dans un réseau, ensemble de règles assurant la transmission de données entre deux ordinateurs, deux composants d'un ordinateur ou deux programmes.»

Office québécoise de la langue française, 1998



Le dialogue Client ⇔ Serveur

- Un serveur peut servir simultanément plusieurs clients dans la limite
  - De ses capacités physiques : bande passante, mémoire, puissance processeur, etc.
  - Des restrictions définies lors de la configuration du serveur
- Le type de service fourni par le serveur, définit le protocole de communication à appliquer
- Pour le Web, il existe, au niveau de
  - La couche de transport, deux protocoles
    - TCP (**T**ransmission **C**ontrol **P**rotocol)
      - Protocole stateful (connexion, donc avec état)
      - Mode connecté (duplex)
      - ☺ : acquittement de réception, intégrité assurée
      - ☹ : permet uniquement l'unicast, lent, moins d'espace pour les données utiles
      - Exemples : HTTP, FTP, POP3, IMAP, SMTP, …
    - UDP (**U**ser **D**atagram **P**rotocol)
      - Protocole stateless (sans état)
      - Mode non-connecté (simplex)
      - ☺ : permet le multicast et le broadcast, rapide, plus d'espace pour les données utiles
      - ☹ : aucun acquittement, intégrité non garantie, perte d'information, moins fiable
      - Exemples : DNS, Streaming, VoIP, WOL, …

Le dialogue Client ⇔ Serveur

- La couche application, plusieurs protocoles :
  - HTTP (**H**yper**T**ext **T**ransfer **P**rotocol) : échange de pages WEB
  - FTP (**F**ile **T**ransfer **P**rotocol) : échange de fichiers
  - TELNET (**TEL**etypewriter **N**etwork **P**rotocol) : terminal distant
  - SMTP (**S**imple **M**ail **T**ransfer **P**rotocol) : envoi de courrier
  - POP, IMAP, IMAPS : réception de courrier
  - DNS (**D**omain **N**ame **S**ystem) : « *mapping* » nom domaine ⇔ @IP
  - SNMP (**S**imple **N**etwork **M**anagement **P**rotocol) : administration de réseaux (interrogation, configuration des équipements, …)
  - Etc.

Le protocole HTTP

- HTTP : protocole de communication client-serveur de la couche application pour le transfert de documents hypertexte (fichiers texte encodés en ASCII ou en UTF)
  - *Fondateur : Tim Berners-Lee (1989-1992)*
  - Nécessite une connexion fiable → TCP
  - Clients : navigateurs Web, robots d'indexation, aspirateurs Web
  - Serveurs (port 80) : plusieurs logiciels
- HTTPS : variante sécurisée (port 443) par l'usage des protocoles SSL ou TLS
- Évolution
  - HTTP 0.9 (1990) → 1.0 (1996)→ 1.1 (1997)→ 2.0 (2012) démarrage à l'IETF→ 1.1 bis (2014) republication en plusieurs RFC et correction
- Spécification
  - Voir le site W3C : https://www.w3.org/Protocols/
  - Voir le site de l'IETF : http://www.ietf.org/rfc/rfc2616.txt

Le protocole HTTP ▸ Serveurs

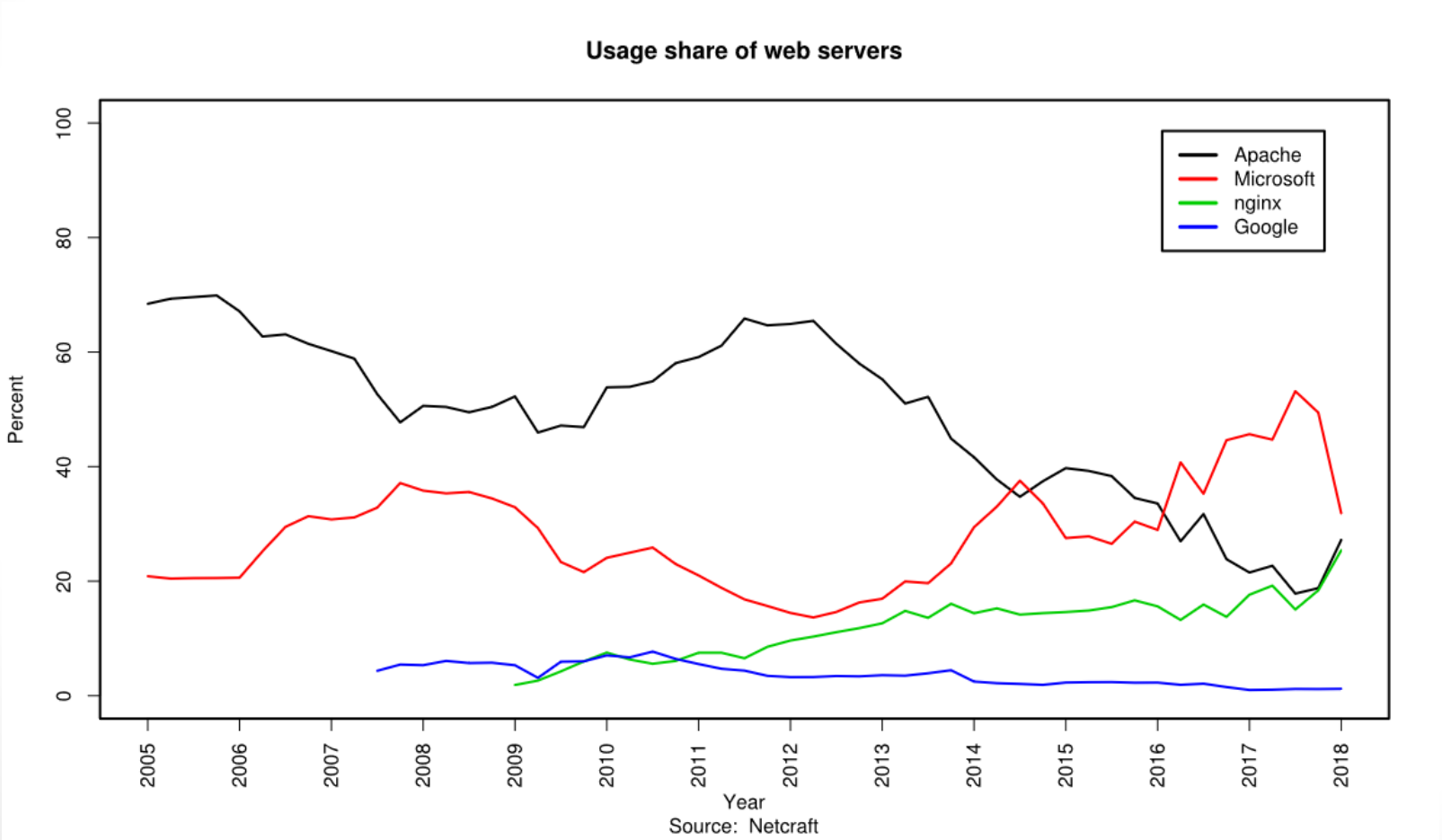
(source https://fr.wikipedia.org/wiki/Serveur_HTTP)

Quelques serveurs

- **Apache HTTP Server** de la **Apache Software Foundation**
- **Apache Tomcat** de la **Apache Software Foundation**, évolution de Apache pour J2EE
- BusyBox HTTPD, utilisé dans le domaine de l'embarqué, et notamment avec OpenWRT2
- **Google Web Server** de **Google**
- **Internet Information Services (ISS)** de **Microsoft**
- lighttpd de Jan Kneschke
- Monkey Web Server de Eduardo Silva Pereira, dédié au noyau Linux
- NGINX de Igor Sysoev
- **NodeJS** sous licence MIT conçu par Ryan Lienhart Dahl en lignes de **programmation** en **JavaScript**
- Sun Java System Web Server de Sun Microsystems
- Tengine, fork de NGINX, de Taobao (9e rang mondial Alexa en juillet 2014)
- Zeus Web Server de Zeus Technology
- Gunicorn est un serveur Web HTTP WSGI écrit en Python pour Unix

Le protocole HTTP ▸ Serveurs

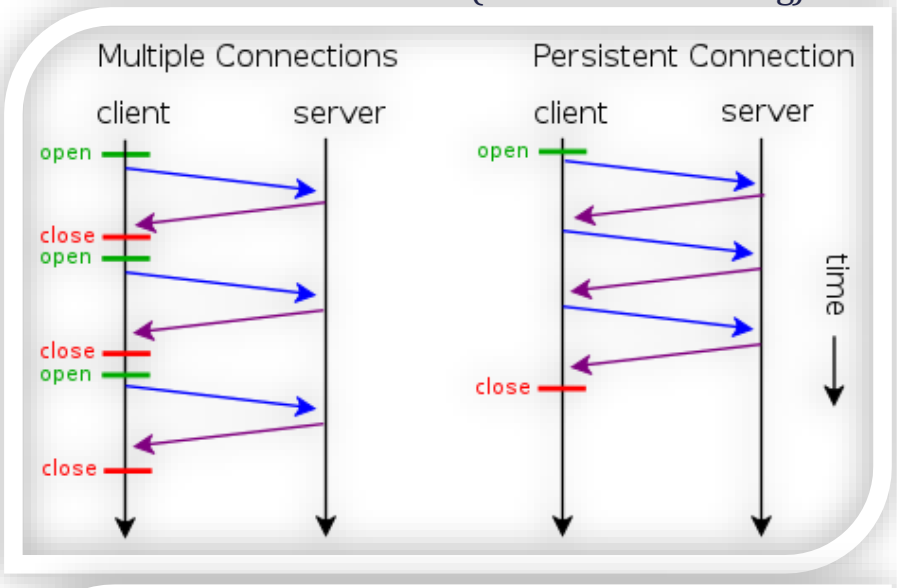
(source https://fr.wikipedia.org/wiki/Serveur_HTTP)

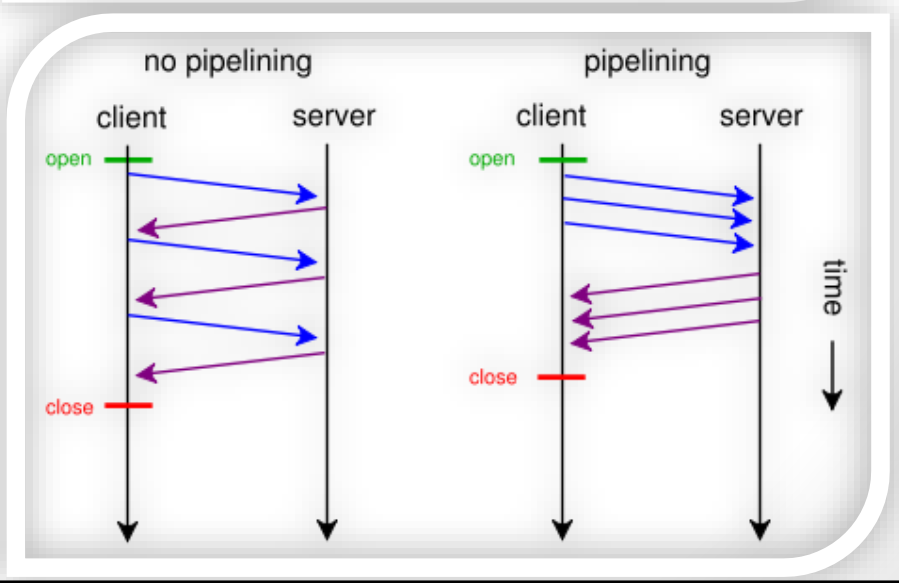


Le protocole HTTP ▸ Processus

- Le client se connecte au serveur
- Le client envoie une commande au serveur
- Le serveur répond à la commande
- Le client ferme la connexion (HTTP 0.9 & HTTP 1.0)
- Optimisations
  - Connexion persistante : une seule connexion permet d'échanger plusieurs commandes (versions récentes de HTTP)
  - *« Pipelining »* : le client envoie plusieurs commandes avant de recevoir les réponses du serveur (HTTP 1.1)

(source fr.wikibooks.org)





Le protocole HTTP ▸ Méthodes

- HTTP défini plusieurs méthodes pour requêter sur un serveur Web : GET, HEAD, POST, PUT, DELETE, OPTIONS, CONNECT, TRACE
- Une méthode est une commande envoyée au serveur pour réaliser une action
- Une action cible une ressource identifiée par une URL
- Structure d'une méthode
  - <Nom de la méthode> <URL de ressource>
  - Exemple : `GET 192.168.1.25/index.html`
- Liste des méthodes
  - GET : obtenir la page web demandée
  - HEAD : obtenir des informations sur la page, sans la consulter
  - POST : envoyer une ressource sur le serveur (un message sur un forum, par exemple)
  - PUT : remplace ou ajoute une ressource sur le serveur
  - DELETE : supprime une ressource sur le serveur
  - OPTIONS : obtenir les options des communications utilisées par le serveur
  - CONNECT : commande spécialisée pour les proxys
  - TRACE : permet de tester la liaison entre serveur et client

Le protocole HTTP ▸ Réponses

- La réponse du serveur respecte un contenu standardisé
  - Entête : un code statut
  - Un corps : exemple pour GET le corps contient la ressource demandée
- Codes statut les plus fréquents
  - 200 : tout s'est bien passé
  - 301 et 302 : redirection vers une autre page
  - 403 : accès refusé
  - 404 : page non trouvée
  - 500 : erreur interne au serveur
  - Pour plus de détails : https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

Le protocole HTTP ▸ Exemples

Requêtes

- **GET /dir/page.html HTTP/1.1**
  Host: www.server.org
  Connection: close
  User-agent: Mozilla/5.0
  Accept-language: fr
- **HEAD http://3s-cms.enstb.org/ HTTP/1.1**
  Accept-Encoding: gzip,deflate
  Host: 3s-cms.enstb.org
  Connection: Keep-Alive
  User-Agent: Apache-HttpClient/4.1.1 (java 1.5)

Réponses

- **HTTP/1.1 200 OK**
  Connection: close
  Date: Tue, 09 Aug 2011 15:44:04 GMT
  Server: Apache/2.2.3 (CentOS)
  Last-Modified: Tue, 09 Aug 2011 15:11:03 GMT
  Content-Length: 6821
  Content-Type: text/html
  Page content
- **HTTP/1.1 400 Bad Request**
  Server: nginx/1.6.2
  Date: Tue, 09 Oct 2018 13:58:59 GMT
  Content-Type: text/html
  Content-Length: 172
  Connection: close