Université de Gabés
Faculté des Sciences de Gabés

Licence en génie logiciel et systèmes d'information
Niveau: 3eme année
Semestre 1

Chapitre 2 : Modèles de communication dans un système réparti

Dhikra KCHAOU
dhikrafsegs@gmail.com

2023-2024

Plan du chapitre 2

1. Introduction
2. Rappel sur les réseaux
3. Protocoles de communication
4. Modèles d'interactions
   - Modèles client/serveur
   - Diffusion de messages
   - Pair à pair
5. Synthèse

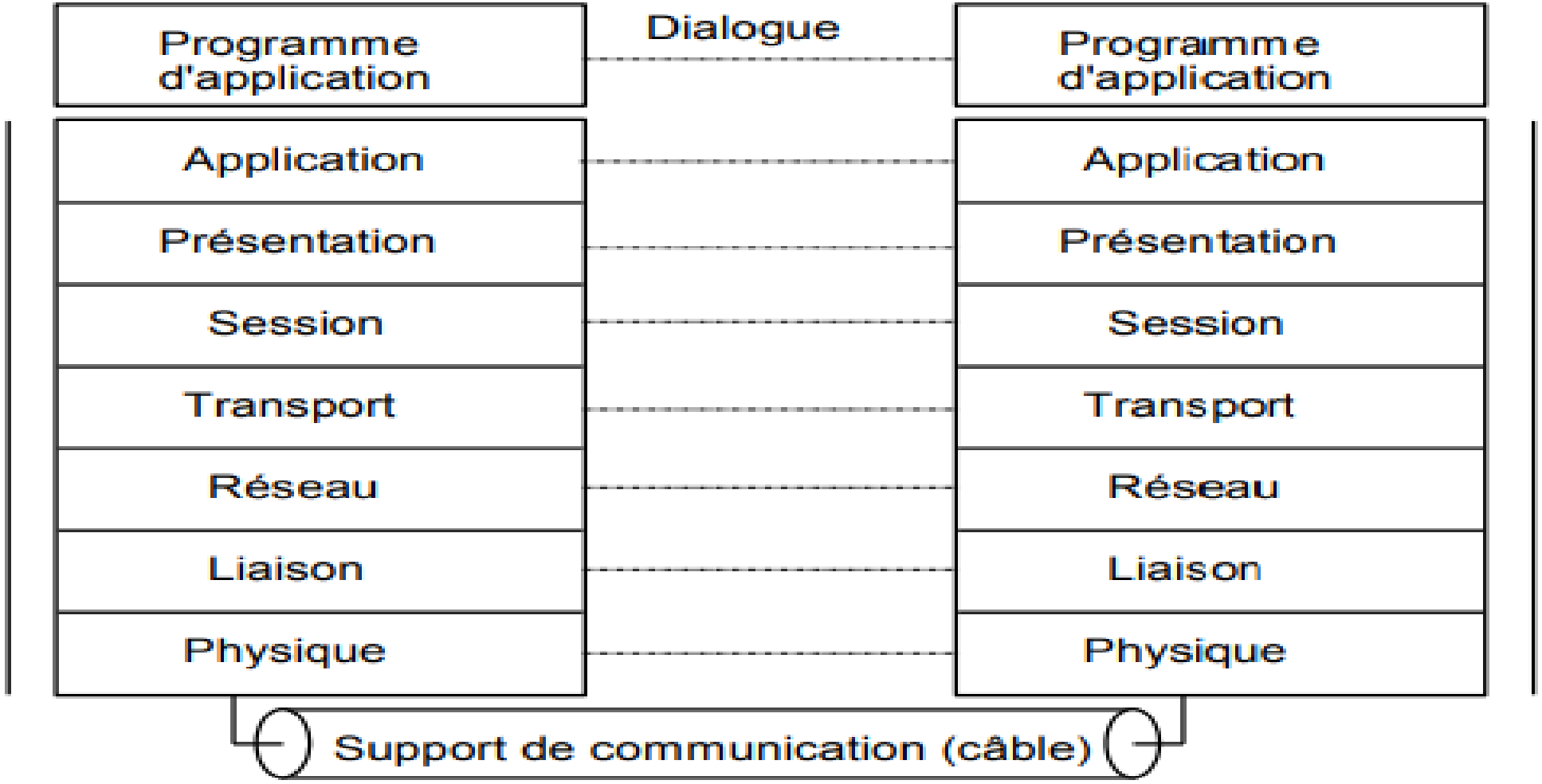
Introduction

- Système distribué : Ensemble de machines indépendantes qui communiquent via un réseau d'une manière transparente.
- Application répartie : met en jeu des parties qui s'exécutent sur plusieurs machines reliées par un système de communication.
- Exemples :
  - L'application **Web** est constituée de deux logiciels communiquants : le navigateur client qui effectue une requête pour disposer d'un document présent sur le serveur Web
  - L'application **telnet** : un terminal virtuel sur le client, un serveur telnet distant qui exécute les commandes

Nécessité de disposer d'un protocole de communication applicatif

Rappel sur les réseaux

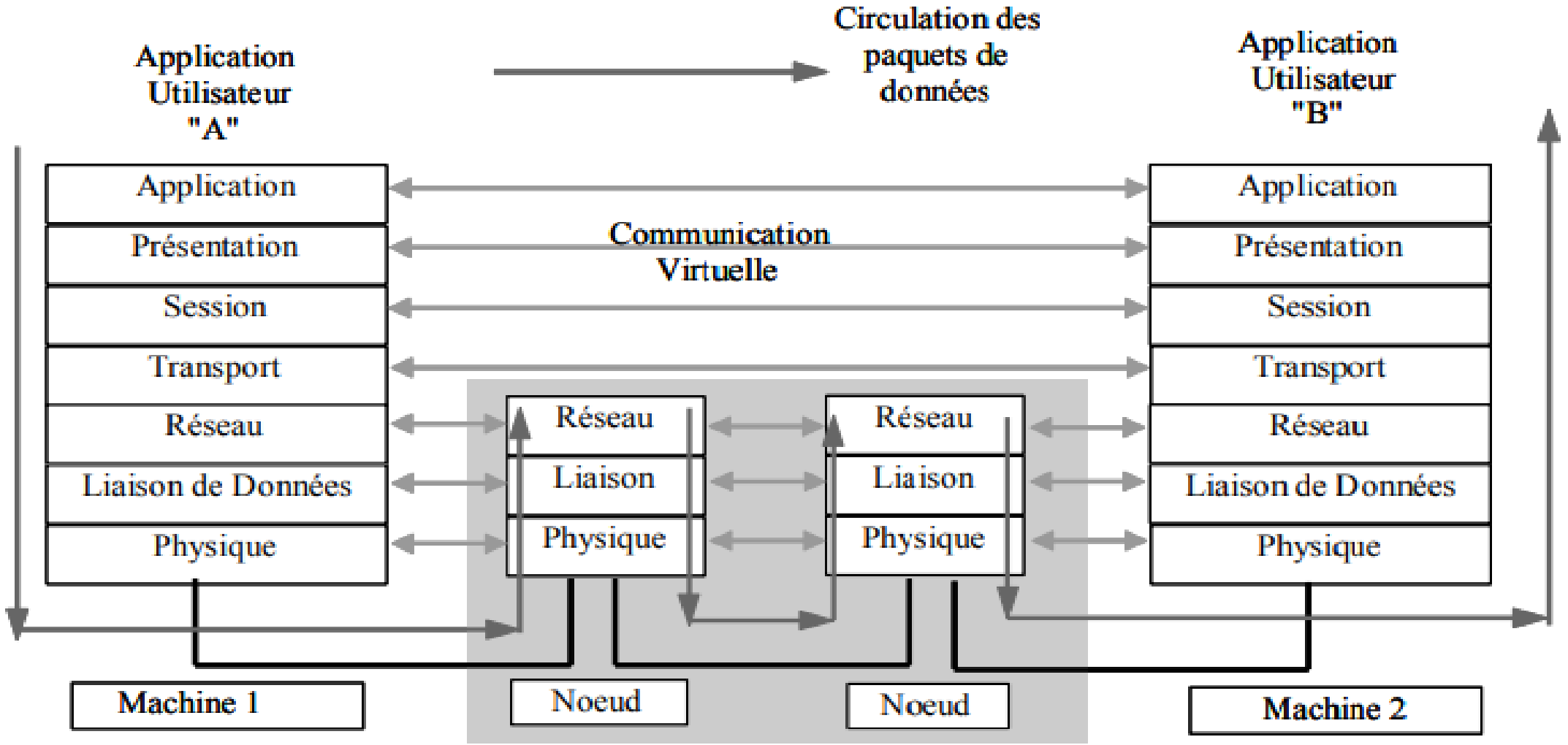
- Norme OSI de l'ISO : architecture en **7 couches**



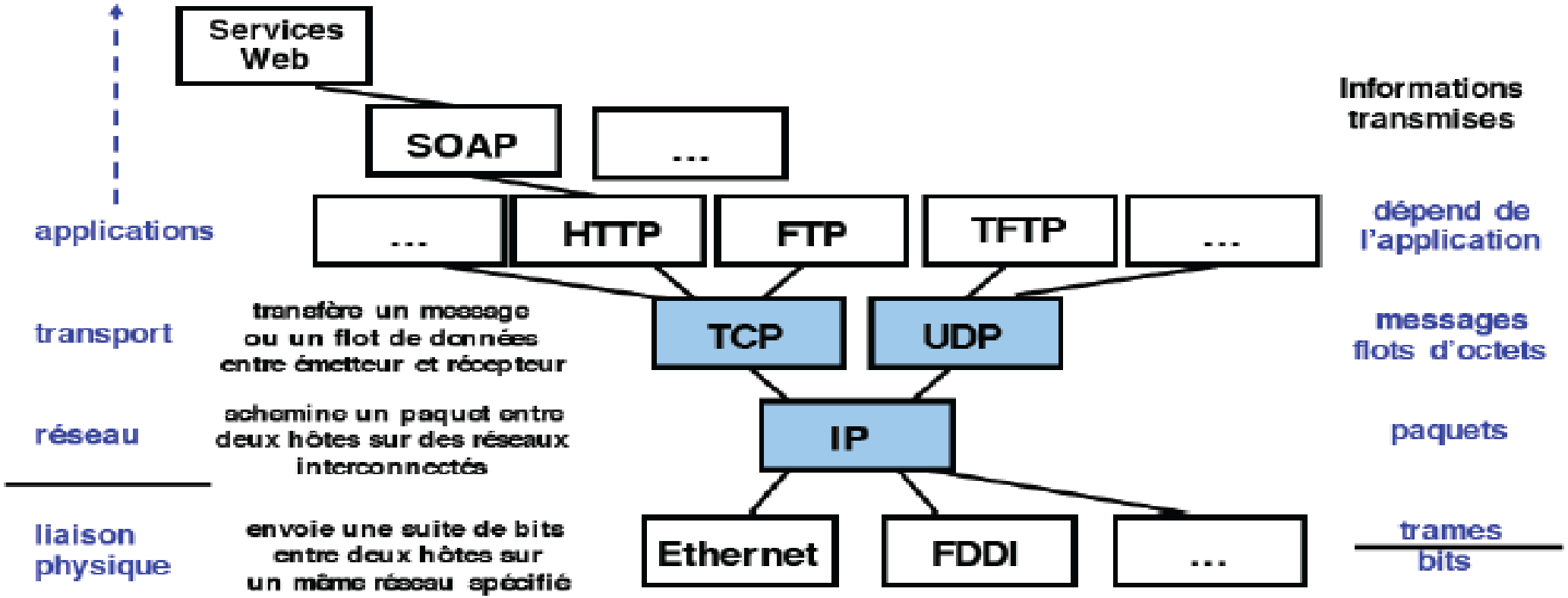
Rappel sur les réseaux

- Norme OSI de l'ISO : architecture en 7 couches
  - **Physique** : transmission des données binaires sur un support physique
  - **Liaison** : gestion d'accès au support physique, assure que les données envoyées sur le support physique sont bien reçues par le destinataire
  - **Réseau** : transmission de données sur le réseau**, trouve les routes** à travers un réseau pour accéder à une machine distante
  - **Transport** : transmission (fiable) entre 2 applications
  - **Session** : synchronisation entre applications, reprises sur pannes
  - **Présentation** : indépendance des formats de représentation des données (entiers, chaînes de caractères...)
  - **Application** : protocoles applicatifs (HTTP, FTP, SMTP ...)

Communication en réseaux



Protocoles de communication



HTTP : *HyperText Transfer Protocol* : protocole du Web
TFTP, FTP : (*Trivial*) *File Transfer Protocol*) : transfert de fichiers
TCP : *Transmission Control Protocol* : transport en mode connecté
UDP : *User Datagram Protocol* : transport en mode non connecté
IP : *Internet Protocol* : Interconnexion de réseaux, routage

Adressage pour communication entre applications

Adresse « réseau » d'**une application** = couple de 2 informations :

Adresse IP et **numéro de port**

- Couche réseau : adresse IP, Ex : 192.129.12.34
- Couche transport : numéro de port TCP ou UDP
  - Ce numéro est en entier d'une valeur quelconque
    - Ports < 1024 : réservés pour les applications ou protocoles systèmes
    - Exemple : 80 = HTTP, 21 = FTP, ...
  - Sur un port : réception ou envoi de données
    - Adresse notée : **@IP:port** ou **nomMachine:port**
    - 192.129.12.34:80 : accès au serveur Web tournant sur la machine d'adresse IP 192.129.12.34

Protocoles de communication

- **Protocole de communication** : Ensemble de règles et de contraintes gérant une communication entre plusieurs entités
- **But du protocole** :
  - Se mettre d'accord sur la façon de communiquer pour bien se comprendre
  - S'assurer que les données envoyées sont bien reçues
- Plusieurs types d'informations circulent entre les entités
  - Les données à échanger
  - Les données de contrôle et de gestion du protocole

Exemple basique de protocole

- Une entité envoie des données à une deuxième entité
- La deuxième entité envoie un acquittement pour prévenir qu'elle a bien reçue les données
- Mais si le réseau est non fiable ou aux temps de transmission non bornés
  - Comment gérer la perte d'un paquet de données ?
  - Comment gérer la perte d'un acquittement ?
  - Comment gérer qu'un message peut arriver avant un autre alors qu'il a été émis après ?

Protocoles de communication

- **Protocole TCP/IP**
  - Couche réseau : IP (Internet Protocol)
    - Gestion des communications et connexions entre les machines à travers le réseau
    - Recherche des routes à travers le réseau pour accéder à une machine
  - Couche transport : TCP/UDP
    - **TCP** (Transmission Control Protocol)
    - **UDP** (User Datagram Protocol)

Protocoles de communication

- Deux modes de communication réseaux :
  - **Mode déconnecté**: par paquet (**DataGram)** → **Protocole UDP**
    - Pas de garantie que le paquet arrive à destination
    - Les paquets n'arrivent pas dans le même ordre d'envoi
    - Vitesse de transmission plus rapide que TCP
  - **Mode connecté**: par flux (streams) → Protocole **TCP**
    - Garantie que les paquets arrivent à destination et dans l'ordre d'envoi
    - Communication temps réel
    - Vitesse de transmission de données plus lente par rapport à UDP

Modèles de communication

- Les éléments distribués interagissent, communiquent entre eux selon plusieurs modèles possibles :
  - Client/serveur
  - Pair à pair
  - Diffusion de messages
  - ...

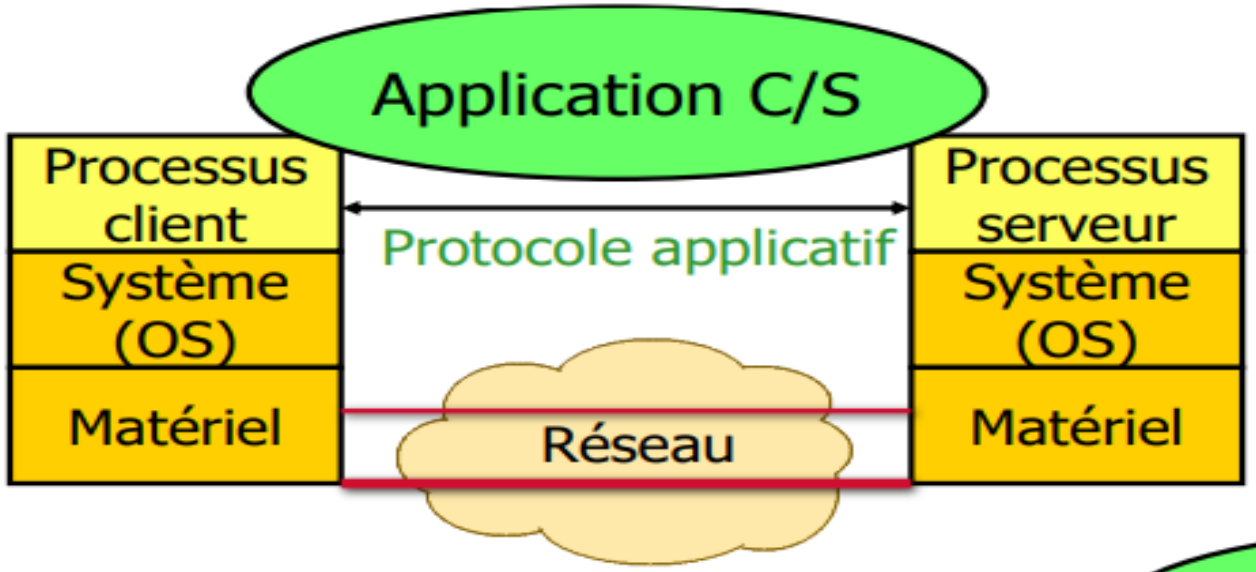
Modèle client/serveur

- Modèle apparu dans les années 1970 (Xerox PARC)
- Deux rôles distincts:
  - **Un serveur** est un ordinateur (et/ou un programme informatique) offrant un service ou une ressource sur un réseau.
  - **Un client** est un programme informatique contactant un serveur via un réseau, afin de bénéficier d'un service ou d'une ressource.
- Le client :
  - effectue une demande de service auprès du serveur (requête)
  - initie le contact (parle en premier), ouvre la session
- Le serveur
  - est la partie de l'application qui offre un service
  - est à l'écoute des requêtes clientes
  - répond au service demandé par le client (réponse)

Modèle client/serveur

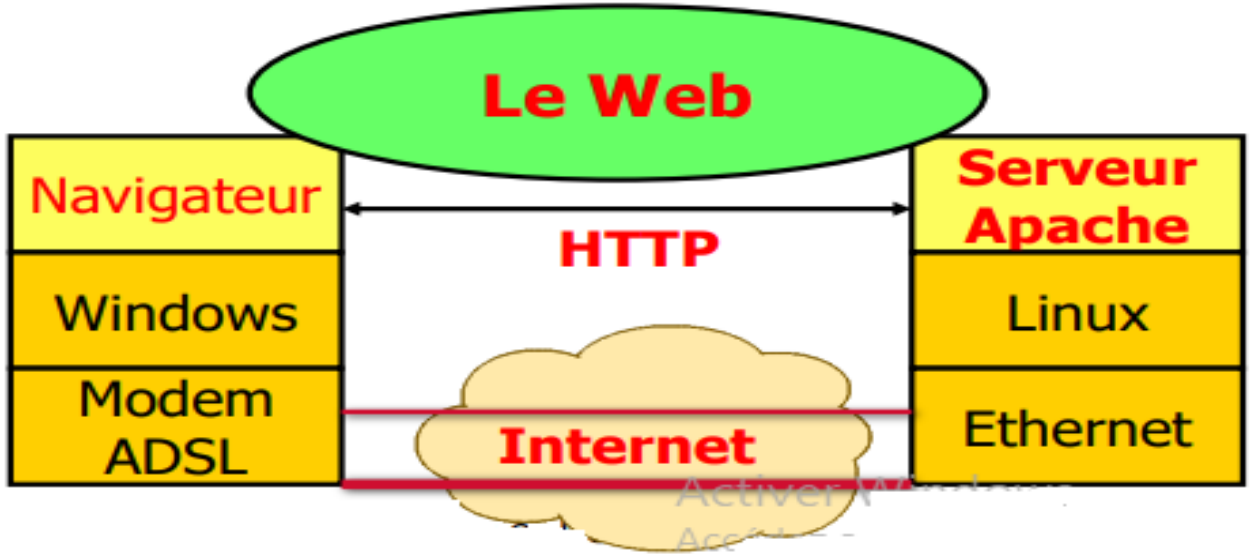
- Le client et le serveur ne sont pas identiques, ils forment un système coopératif
  - les parties client et serveur de l'application peuvent s'exécuter sur des systèmes différents
  - une même machine peut implanter les côtés client et serveur de l'application
  - un serveur peut répondre à plusieurs clients simultanément

Modèle client/serveur



L'application est répartie sur le client et le serveur qui dialoguent selon un protocole applicatif spécifique

L'exemple du Web

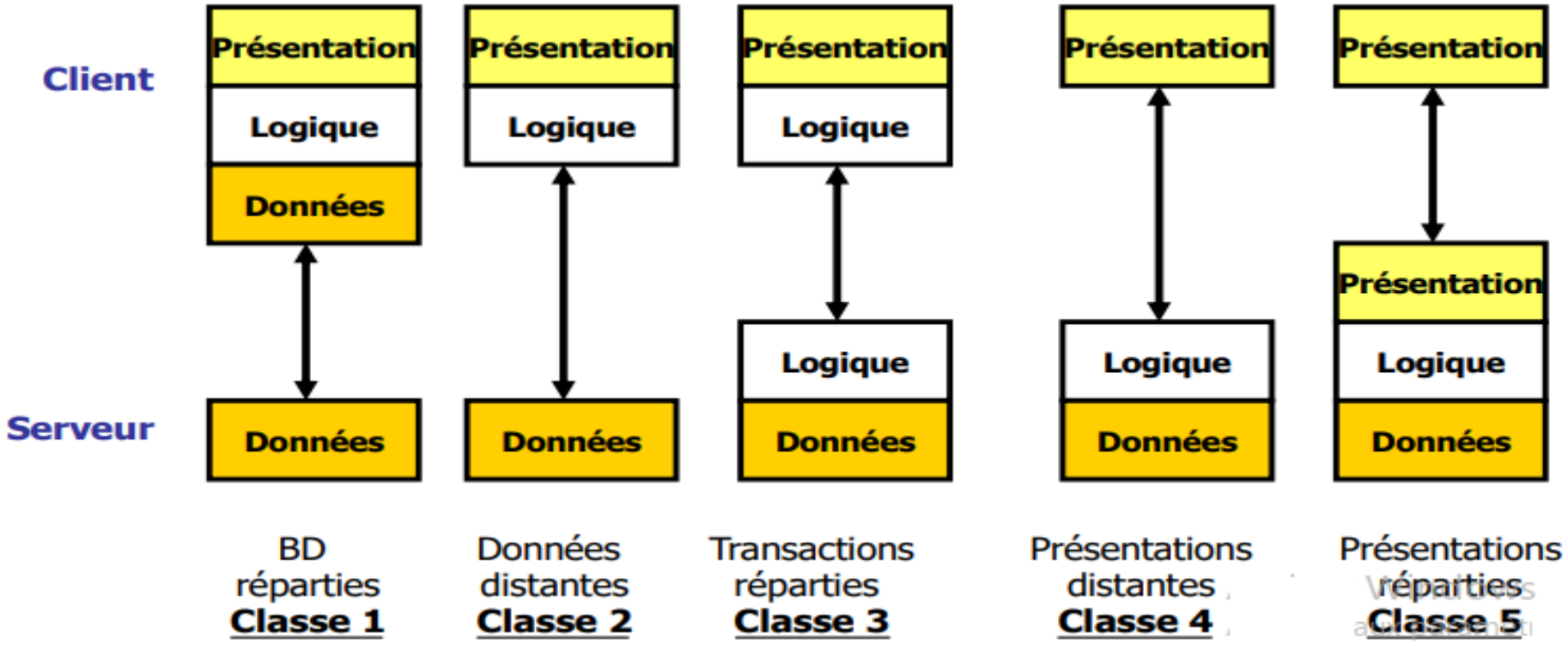


Conception d'une application Client/serveur

- Une application informatique est représentée selon un modèle en trois couches :
  - la couche présentation (interface Homme/Machine) : gestion de l'affichage…
  - la couche traitements (ou logique) qui assure la fonctionnalité intrinsèque de l'application (algorithme)
  - la couche données qui assure la gestion des données de l'application (stockage et accès)
- Ces 3 niveaux peuvent être regroupés ou répartis: Architecture 1-tiers, Architecture 2-tiers, Architecture 3-tiers, Architecture N-tiers.

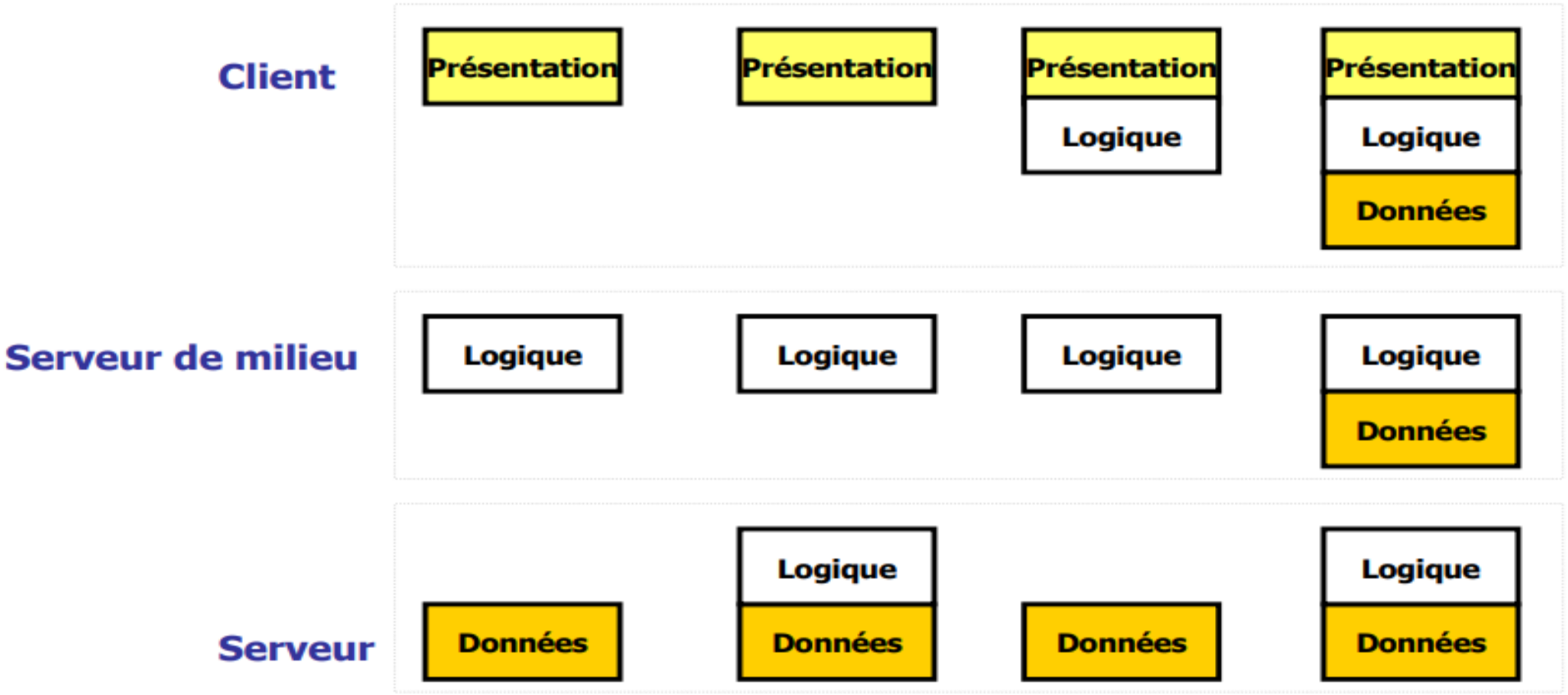
Client-serveur 2 niveaux (2-tiers)

- Les trois couches de l'application sont déployées sur deux machines différentes.
- Classification de Gartner Group:



Client-serveur 3 niveaux (3-tiers)

- Modèle de Gartner pour les systèmes à 3 niveaux (3-tiers) :



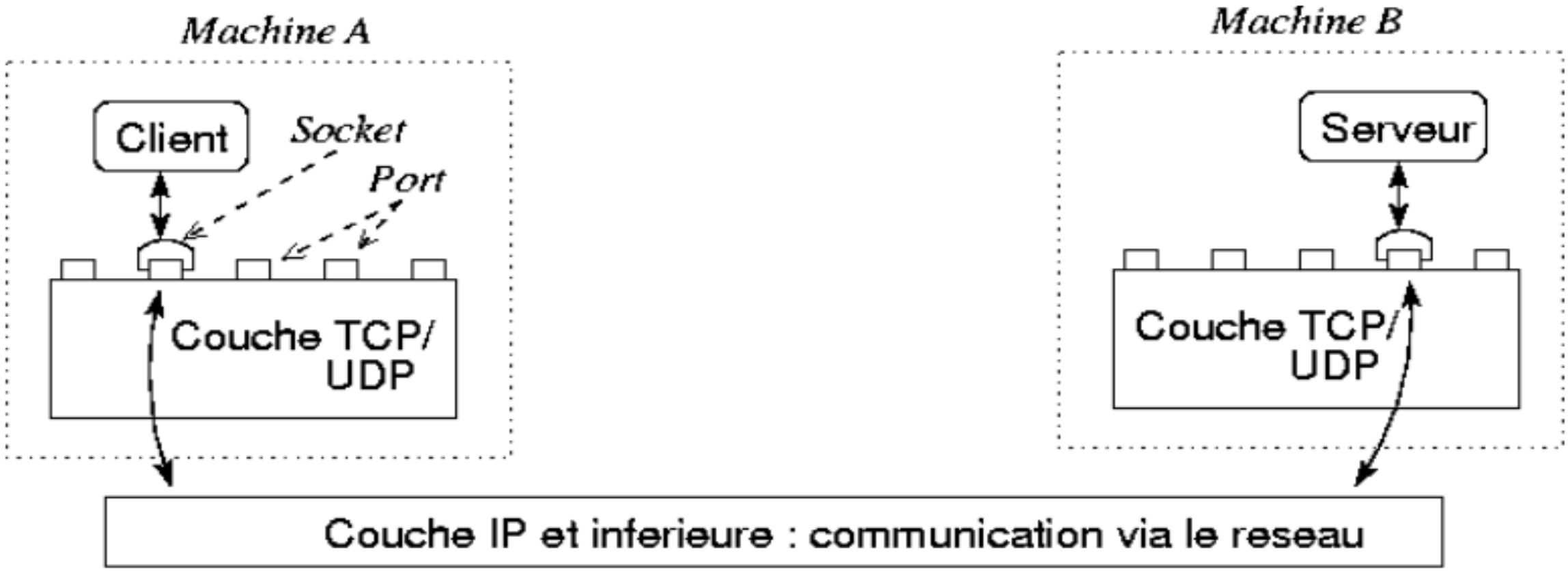
Application Client/Serveur - récapitulatif

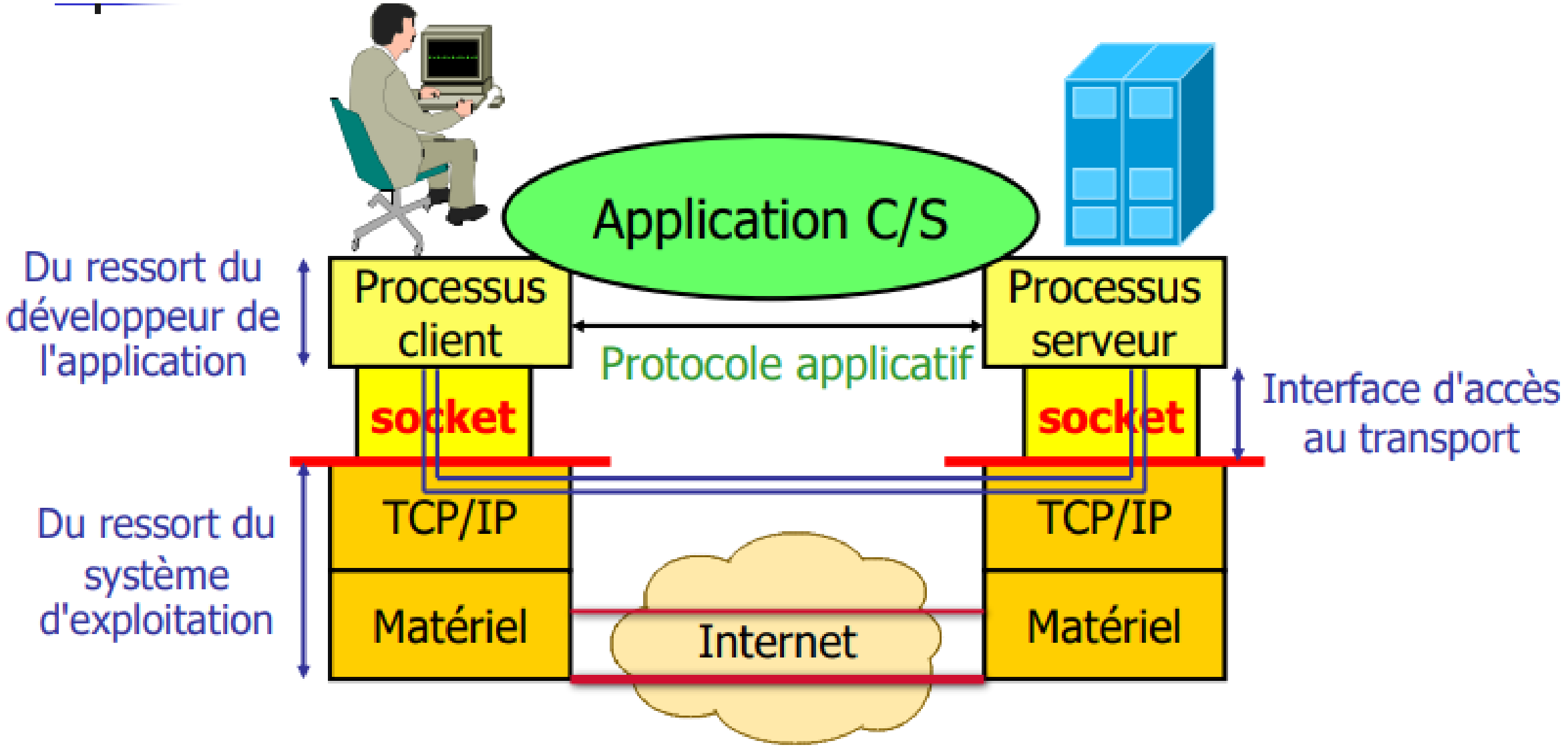
- Une application Client/Serveur, c'est
  - **une partie cliente** qui demande des requêtes vers un serveur
  - **une partie serveur** qui traite les requêtes clientes et y répond
  - **un protocole applicatif** qui définit les échanges entre un client et un serveur
  - **un accès via une API** (interface de programmation) à la couche de transport des messages
- Bien souvent les parties cliente et serveur ne sont pas écrites par les mêmes programmeurs (Navigateur Netscape/Serveur apache)

Mise en œuvre d'un Client Serveur

- Communication entre entités logicielles via l'**API:**
  - **Le plus basique (bas niveau)**: directement en appelant les services des couches TCP ou UDP → **les sockets**
  - **Plus haut niveau** :
    - Appel de procédure à distance (**RPC**), dans un langage particulier, exemple : C
    - Appel de méthode à distance, dans les langages à objets (**Java RMI**)
    - Middleware intégré : **CORBA, EJB, .Net, Web Services**

Les sockets


Machine A
Client
Socket
Port
Couche TCP/
UDP
Machine B
Serveur
Couche TCP/
UDP
Couche IP et inferieure : communication via le reseau



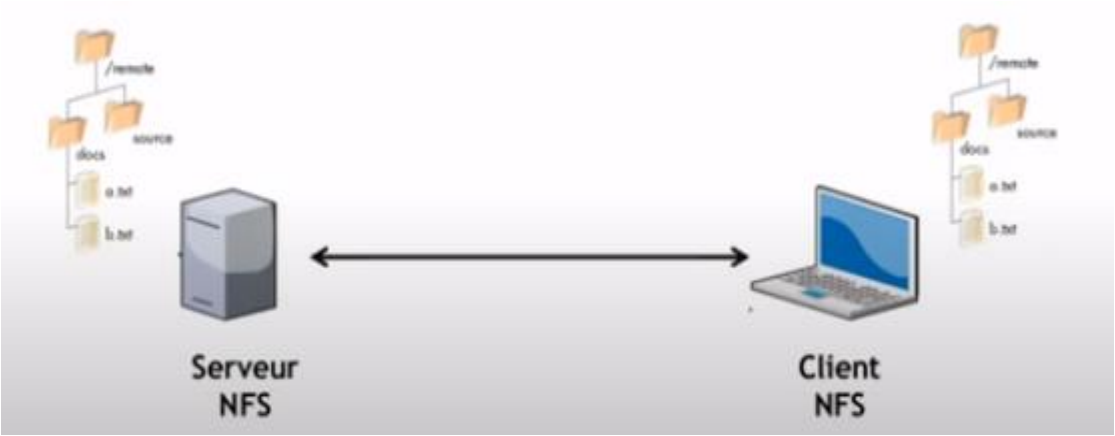
Une socket : interface locale à l'hôte, créée par l'application, contrôlée par l'OS

Porte de communication entre le processus client et le processus serveur

RPC

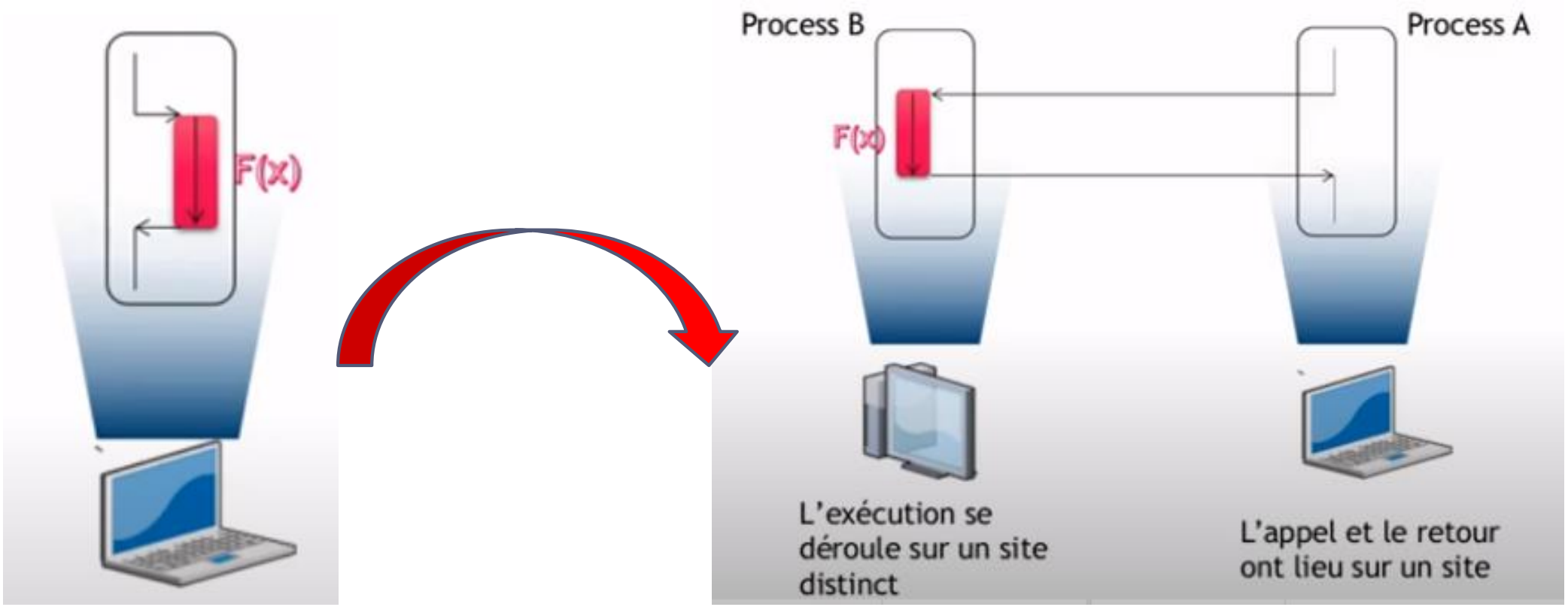
RPC: Remote Procedure Call, Appel de procédure à distance

- RPC a été développé par Sun Microsystems pour **NFS** (Network File System), au début des années 1980.



- But : effet identique à ceux d'un appel local sans se préoccuper ni de la localisation de la procédure ni du traitement des pannes.

Exemple de mise en œuvre d'un Modèle client/serveur : RPC

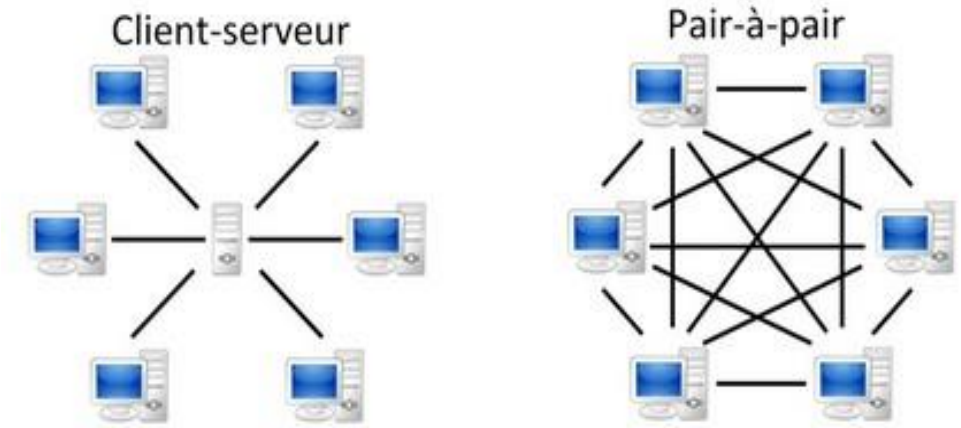


Extension des RPC

- RPC a été mis en œuvre en utilisant différents noms:
  - Sun RPC : langage C, (pour NFS, début 1980)
  - CORBA : approche essentiellement orientée objet, langage c++ et JAVA (1991)
  - Microsoft DCOM: pour Windows 95 (1995)
  - Java RMI: orienté objet mais uniquement en Java (1997)
    - ➢Invocation de méthodes distantes
  - Soap/XML-RPC: RPC basé sur XML et RPC (1998)
  - AJAX (Asynchronous Javascript and XML) dans le web browser (1999)
  - Rest avec JSON (2000)
  - ……

Modèle pair à pair P2P

- Motivations
  - Besoin de partage de ressources
  - Besoin de calcul
- Définitions



« *Le Pair à Pair est une classe de systèmes auto organisants qui permettent de réaliser une fonction globale de manière décentralisée, en tirant parti du partage des ressources et de la puissance de calcul disponibles sur un grand nombre d'extrémités de l'Internet* »

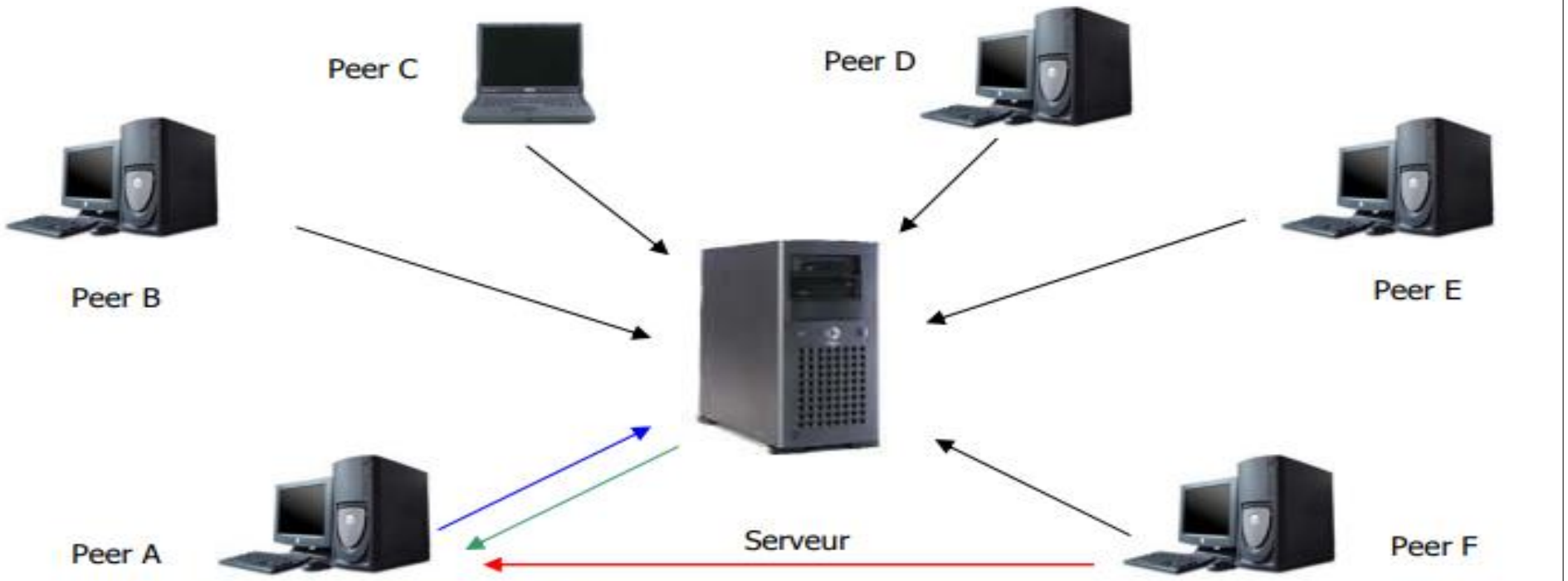
Bruno Richard - HP Laboratories Grenoble

Principes généraux de P2P

- Chaque participant est à la fois client et serveur («servent»)
- Complètement décentralisé :
  - aucun contrôle/coordination centrale
  - aucune base de données centrale
  - le comportement global émerge des interactions locales
  - Tous les services et données sont accessibles par tous les «peers»
  - Les «peers» sont autonomes.
  - Les «peers» et leurs interconnexions ne sont pas fiables
- «Modèle d'affaire»: chaque nœud contribue par la mise à disposition de (quelques unes de) ses ressources

Types d'architectures P2P

- **Architecture centralisée : le peer-to-peer assisté**

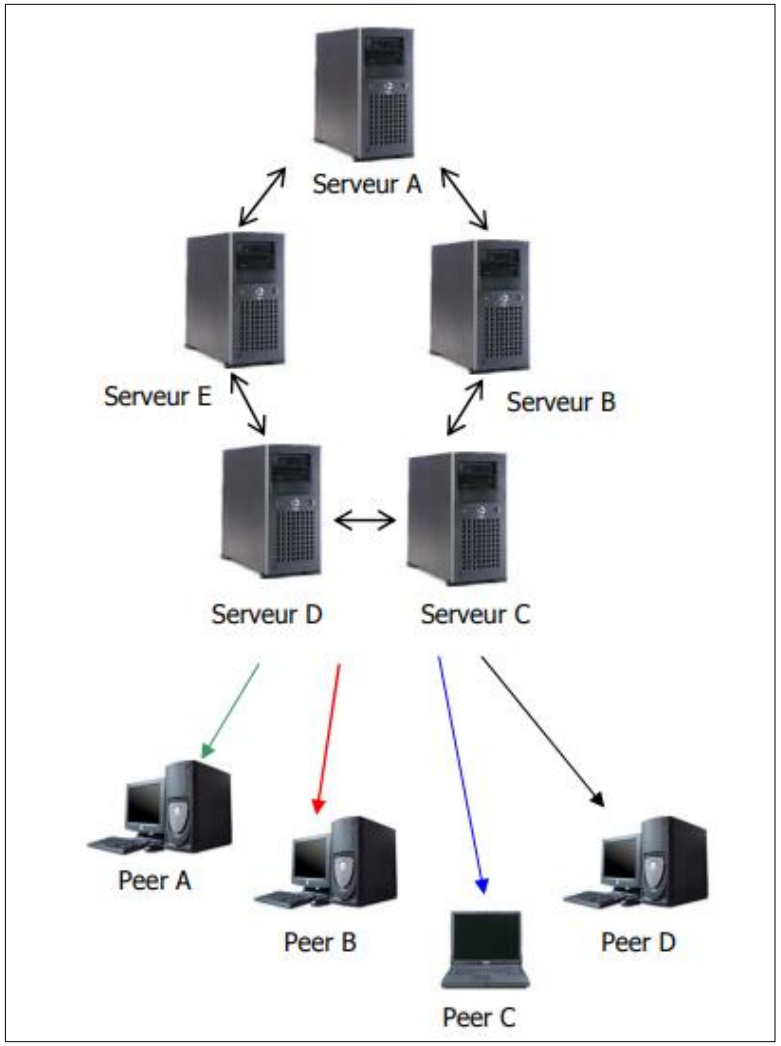


- Un utilisateur recherche un fichier ressource en envoyant une requête au serveur central
- Le serveur central répond et transmet la liste des ordinateurs utilisateurs proposant le fichier demandé.
- L'utilisateur télécharge le fichier directement à partir d'un des ordinateurs renseignés par le serveur.

Types d'architectures P2P

Une amélioration du Peer-to-Peer centralisé :

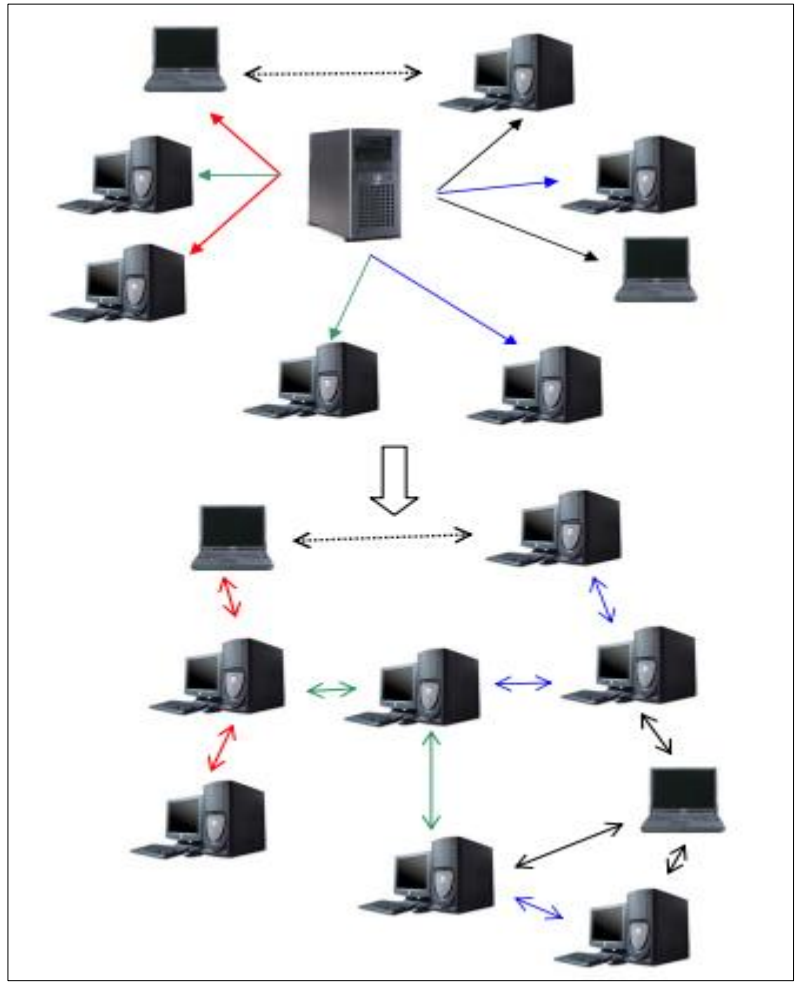
- le serveur central de l'architecture centralisée est remplacé par un anneau de serveur.
- Ceci permet d'éviter la chute du réseau si une panne se produit sur un serveur, car il y a toujours un point de connexion valide aux serveurs.
- l'utilisation de plusieurs serveurs permet de mieux répartir les demandes de connexions et donc de limiter la chute de bande passante.



Types d'architectures P2P

Architecture décentralisée

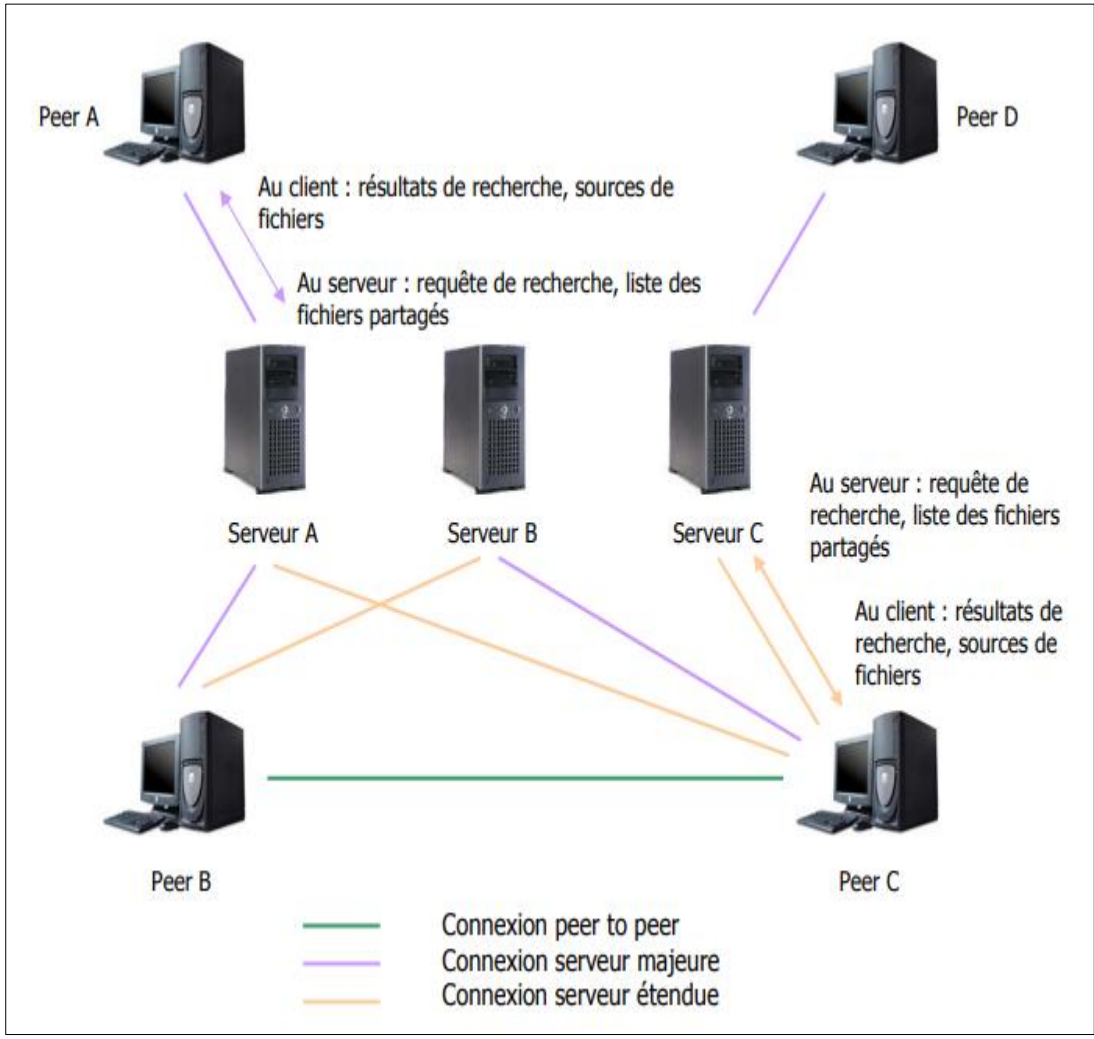
- L'objectif est de ne pas utiliser les serveurs centralisés → trouver le moyen de constituer un annuaire sur chaque client, puis de les faire communiquer.
- Tous les éléments du réseau vont jouer ce rôle du serveur.
- Chaque machine dans ses rôles est identique à une autre, c'est pour cela que l'on appelle ces types de réseaux pur peer to peer.



Les réseaux peer-to-peer actuels

e-Donkey et e-Mule

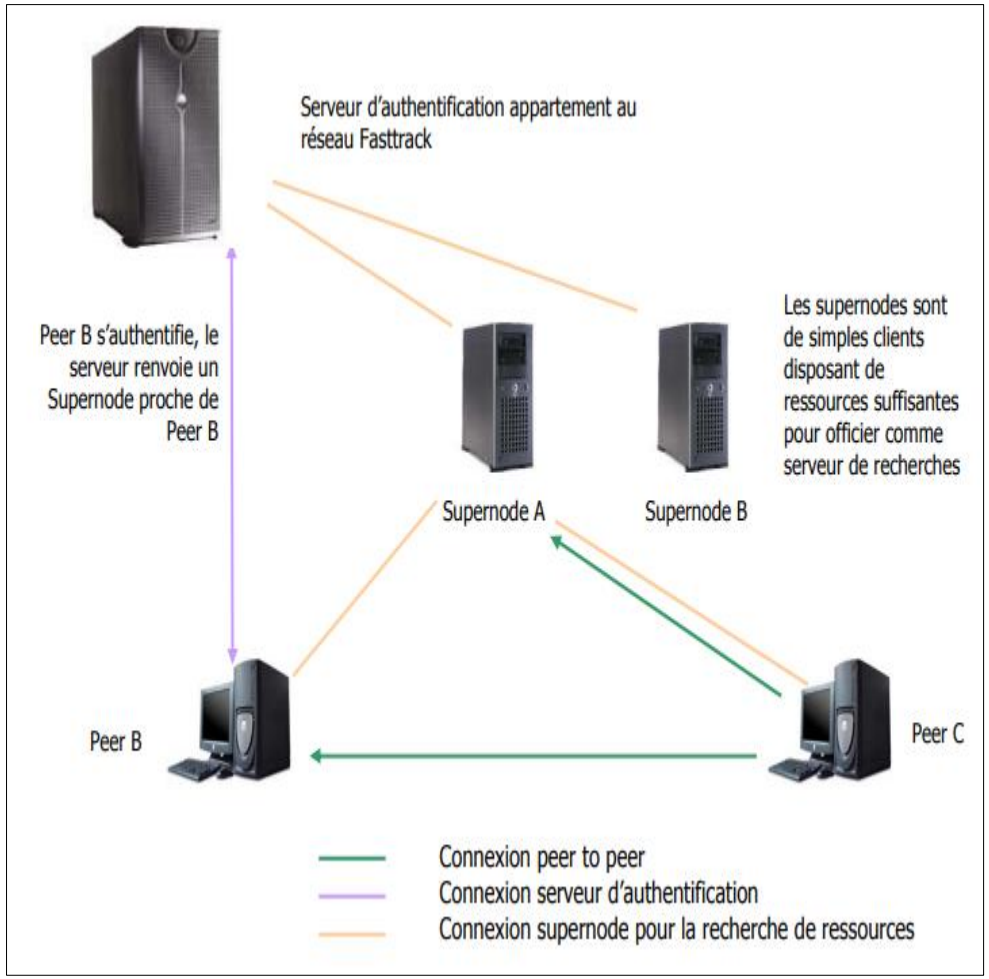
- e-Donkey fonctionne dans un mode décentralisé avec une multitude de serveurs.
- Chaque utilisateur peut ouvrir son propre serveur et tous les serveurs peuvent être reliés entre eux.
- Lorsqu'un client se connecte sur un serveur il lui fournit la liste des fichiers partagés. Une fois connecté, il peut rechercher un fichier. Soit il le recherche sur le serveur sur lequel il est connecté, soit sur tous les serveurs connus par l'intermédiaire des autres peer.



Les réseaux peer-to-peer actuels

FastTrack: Kazaa, Grokster, Morpheus, Imesh

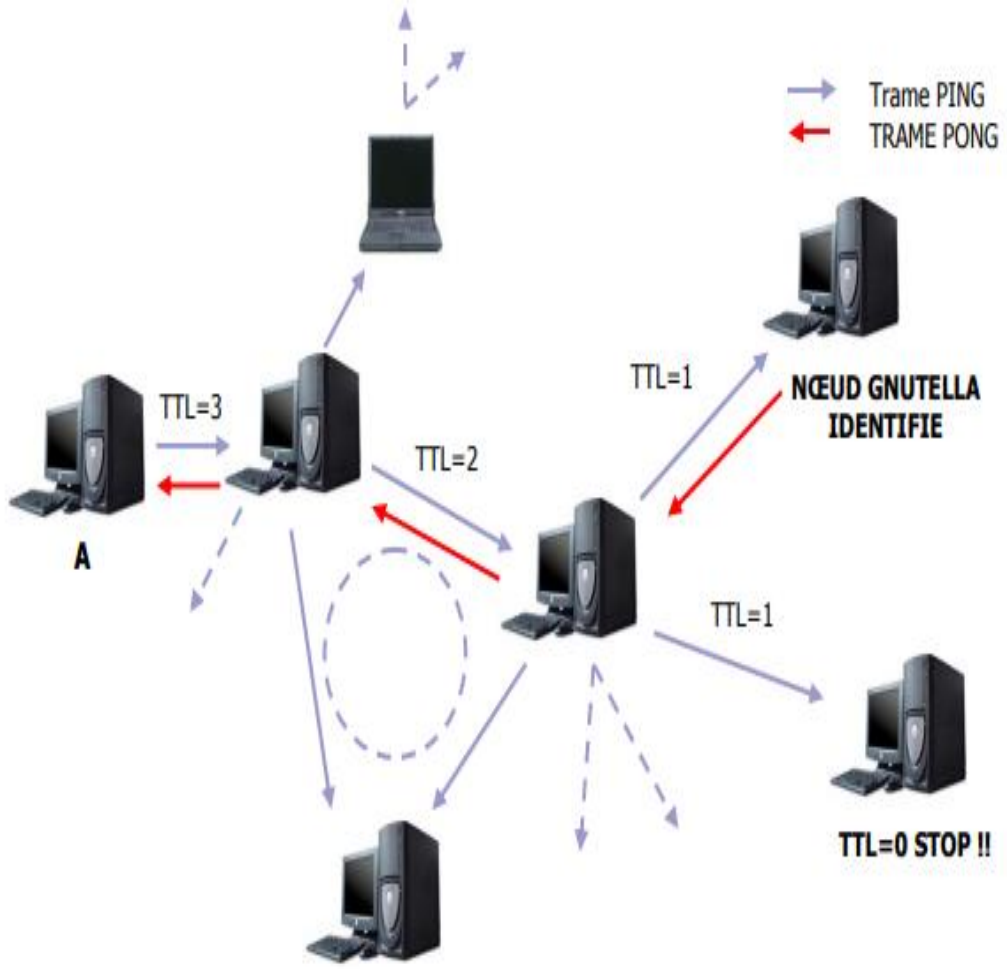
- Le réseau FastTrack, dont le client le plus connu du grand public est Kazaa peut être comparé à eDonkey sur plusieurs points. Tout comme lui, il permet le partage de fichiers de tous types et emploie le protocole MFTP.



Les réseaux peer-to-peer actuels

Gnutella:

- Il s'agit d'un peer-to-peer décentralisé qui a évolué au fil du temps.
- Son architecture était comparable à e-Donkey avec des fonctionnalités en moins (pas de recherches sur les serveurs voisins etc.).



Les inconvénients du peer-to-peer

- **Manque d'un minimum de contrôle** → la diffusion de virus ou le freeloading (Les freeloaders prennent sans donner).
- **Une atteinte à la vie privée**:
  - les adresses IP des utilisateurs peuvent être récupérées lorsque le logiciel est centralisé.
  - les utilisateurs sont identifiables par leur fournisseur d'accès dans le cas d'un logiciel distrubué.
- **La pollution des réseaux** → en incorporant des fichiers de moindre qualité ou incorrects.

Modèles de communication par messages

- Une application produit des messages (producteur) et une application les consomme (consommateur).
- Le producteur (l'émetteur) et le consommateur (récepteur) ne communiquent pas directement entre eux mais utilisent un objet de communication intermédiaire (boite aux lettres ou file d'attente)
- Communication asynchrone:
  - Les deux composants n'ont pas besoin d'être connecté en même temps grâce au système de file d'attente.
  - Émission non bloquante : l'entité émettrice émet son message, et continue son traitement sans attendre que le récepteur confirme l'arrivée du message
  - Le récepteur récupère les messages **quand il le souhaite.**

Modèles de communication par messages

- Adapté à un système réparti dont les éléments en interaction sont faiblement couplés:
  - Éloignement géographique des entités communicantes
  - Possibilité de déconnexion temporaire d'un élément
- Deux modèles de communication:
  - **Point à point**: le message n'est lu que par un seul consommateur. Une fois lu, il est retiré de la file d'attente.
  - **Multipoints**: le message est diffusé à tous les éléments d'une liste de destinataires.

Mise en œuvre de la communication par message

- **MOM (Message Oriented Middleware)** :
  - Modèle de communication entre logiciels
    - Intégration de modules hétérogènes distribués
    - Indépendance (asynchronisme)
    - Fiabilité
  - Plusieurs implantation
    - IBM (MOM websphere)
    - Application Servers (Sun, Oracle, etc.)
    - MOM open source : Joram, etc.
  - Une interface normalisée (en Java)
    - JMS (Java Message Services) de SUN

Synthèse : Solutions d'interaction dans un système réparti

- Deux grande familles:
  - **Bas niveau** : Utilisation directe de la couche transport → **Sockets**
  - **Haut niveau** :
    - **Les technologies d'appel à procédure à distance** (RPC, remote procedure call), regroupant des standards tels que CORBA, RMI, SOAP … Le principe réside sur l'invocation d'un service (une procédure ou une méthode d'un objet) situé sur une machine distante indépendamment de sa localisation et de son implémentation.
    - **Les technologies d'échange de messages** : échange véhiculé par l'infrastructure MOM (message oriented middleware) Exemple, JMS