Master Informatique M1

Réseaux et Applications Réparties
Introduction

Sacha Krakowiak
Université Joseph Fourier
Projet Sardes (INRIA et IMAG-LSR)

http://sardes.inrialpes.fr/people/krakowia

Site de ce cours :
http://sardes.inrialpes.fr/people/krakowia/Enseignement/M1/RAR-M1.html

Réseaux et Applications Réparties

La majorité des applications informatiques sont maintenant réparties, c'est-à-dire mettent en jeu un ensemble d'ordinateurs ou d'organes divers reliés par un réseau de communication

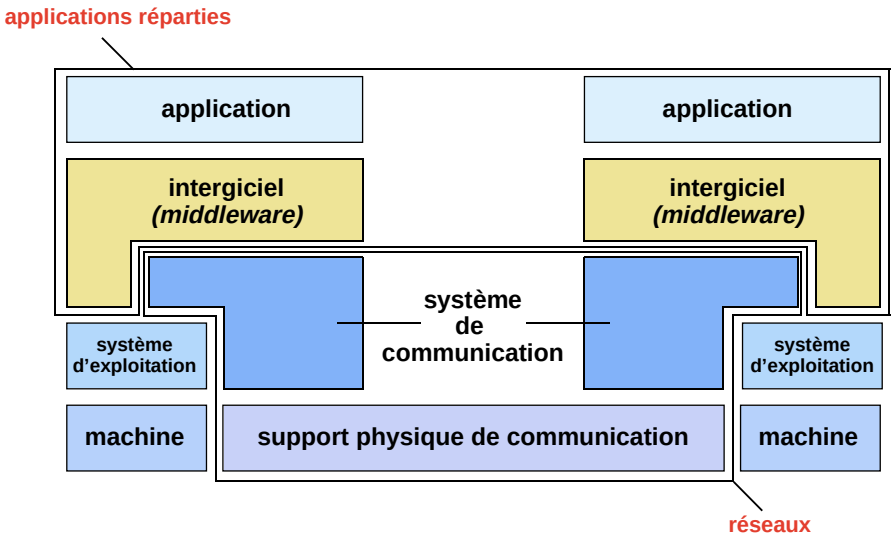
Objectif du cours

- Introduire les notions de base de l'architecture des réseaux et des applications réparties
- Présenter quelques exemples concrets de mise en œuvre à tous les niveaux

Esprit du cours

- Approche intégrée : les mêmes notions de base s'appliquent depuis le réseau physique jusqu'aux applications : service, interface, protocole, désignation, liaison
- Enseignement à dominante pratique : 12h de cours, 18h de TD sur machine)

Le domaine couvert



Services et interfaces

Définition

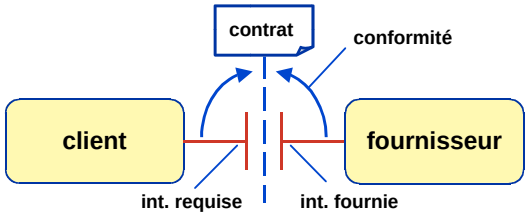
- Un système est un ensemble de composants (au sens non technique du terme) qui interagissent
- Un service est "un comportement défini par contrat, qui peut être implémenté et fourni par un composant pour être utilisé par un autre composant, sur la base exclusive du contrat" (*)

Mise en œuvre

- Un service est accessible via une ou plusieurs interfaces
- Une interface décrit l'interaction entre client et fournisseur du service
  - Point de vue opérationnel : définition des opérations et structures de données qui concourent à la réalisation du service
  - Point de vue contractuel : définition du contrat entre client et fournisseur

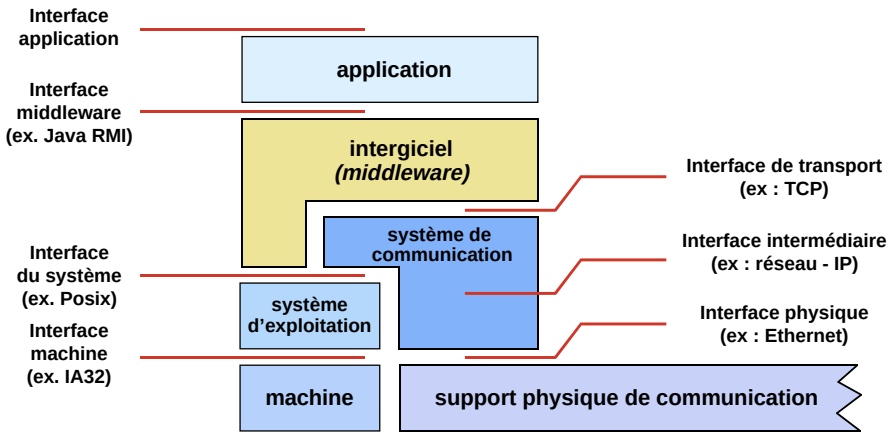
(*) Bieber and Carpenter, *Introduction to Service-Oriented Programming*, http://www.openwings.org

Définitions d'interfaces (1)



- La fourniture d'un service met en jeu deux interfaces
  - Interface requise (côté client)
  - Interface fournie (côté fournisseur)
- Le contrat spécifie la compatibilité (conformité) entre ces interfaces
  - Au delà de l'interface, chaque partie est une "boîte noire" pour l'autre (principe d'encapsulation)
  - Conséquence : client ou fournisseur peuvent être remplacés du moment que le composant remplaçant respecte le contrat (est conforme)
- Le contrat peut en outre spécifier des aspects non contenus dans l'interface
  - Propriétés dites "non fonctionnelles", ou Qualité de Service

Quelques interfaces importantes



Chaque interface cache les interfaces de niveau inférieur

Introduction aux réseaux informatiques

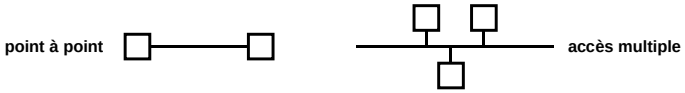
- Un réseau informatique (*computer network*) est un système de communication (ensemble matériel + logiciel) qui permet à un ensemble d'ordinateurs (au sens large) d'échanger de l'information
  - sens large : points d'accès, terminaux de paiement, téléphones, capteurs divers, etc.
- L'échange d'information n'est pas une fin en soi. Les réseaux servent avant tout à réaliser des services
  - accessibles à partir de tout organe connecté au réseau
  - mis en œuvre par un ensemble d'ordinateurs sur le réseau
  - exemples de services

    le courrier électronique (*mail*)
    le transfert de fichiers (*ftp*)
    l'accès à distance (*telnet*)

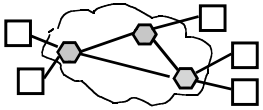
    l'accès au World Wide Web
    les services utilisant le Web :
    documentation, commerce électronique, etc.
- Lien entre réseaux et systèmes d'exploitation
  - Le réseau (support de communication) comme organe d'entrée-sortie
  - le réseau (ensemble de serveurs interconnectés) comme super-machine

Types de réseaux (1)

- Les réseaux peuvent être classés selon différents critères
  - Nature de la liaison entre les organes connectés (nœuds)
    - Liaison directe
      - il y a un lien direct entre deux nœuds du réseau



    - Liaison commutée
      - la liaison passe par des organes intermédiaires



(analogie avec le réseau téléphonique commuté)

    - Les supports physiques de la communication peuvent être divers
      - paires de fils, câble coaxial, fibre optique, radio, infra-rouge, etc.
  - Couverture géographique (réseau local, à grande distance, etc.)

Types de réseaux (2)

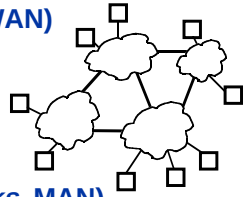
La classification par étendue de la couverture géographique est souvent utilisée, bien que non stricte

Réseaux locaux (*Local Area Networks*, LAN)

- Communication au sein d'une organisation (département d'entreprise, etc.)
- Administration unique
- Couverture géographique limitée (~1 km)
- Débit élevé, taux d'erreur faible
- Topologies diverses : bus, anneau

Réseaux à grande distance (*Wide Area Networks*, WAN)

- Communication entre des organisations diverses
- Administrations multiples
- Couverture géographique étendue : un pays, toute la planète
- Débit variable, taux d'erreur parfois non négligeable
- Topologie maillée ; interconnexion de réseaux (exemple : l'Internet)



Réseaux métropolitains (*Metropolitan Area Networks*, MAN)

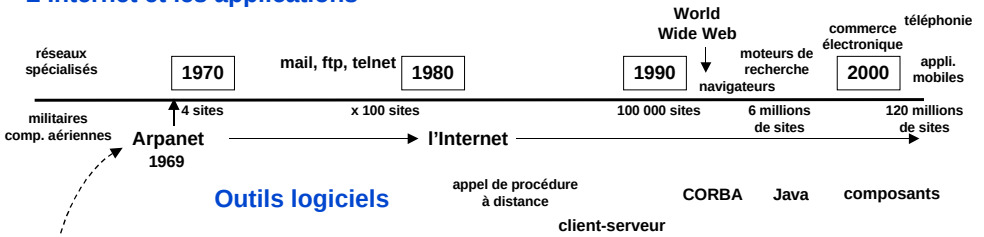
- Intermédiaires entre LAN et WAN - qq dizaines de km, ville ou région

Autres réseaux

- Interconnexion de grappes de machines (*clusters)* - *Desk Area Networks* (DAN) - *Storage Area Networks* (SAN) : réseaux pour le stockage de données
- Réseaux de mobiles

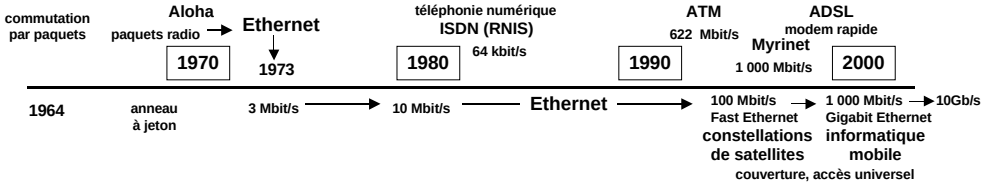
Bref historique des réseaux informatiques

L'Internet et les applications

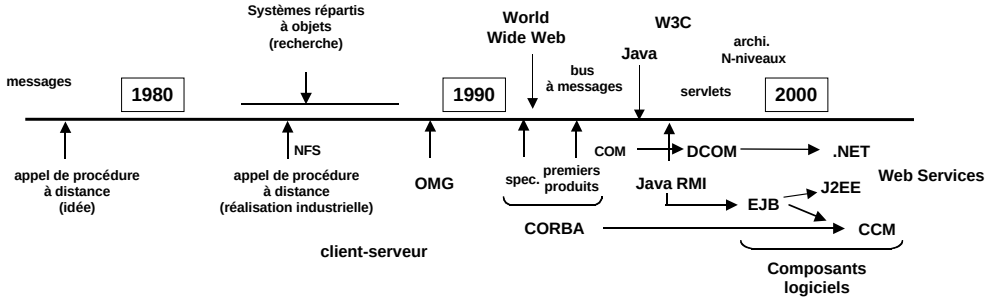


Outils logiciels

Quelques avancées technologiques



Bref historique du *middleware* (intergiciel)



Performances des communications (1)

Deux mesures principales de performance (à tous les niveaux)

- Débit (*throughput)* : quantité d'information par unité de temps. Unité : bit/s (Kbit/s, Mbit/s, …), ou paquets/s, etc.
  - une mesure corrélée au débit est la bande passante (Hz, KHz, ...), cf plus loin
- Latence *(latency)* : temps écoulé entre l'émission d'une unité de transmission et sa réception. Unité : s (ms, µs, …)
  - on s'intéresse aussi parfois au temps d'aller-retour (*round trip time* ou RTT)

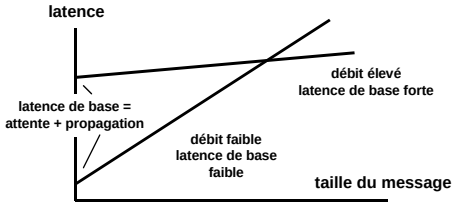
Facteurs de performance

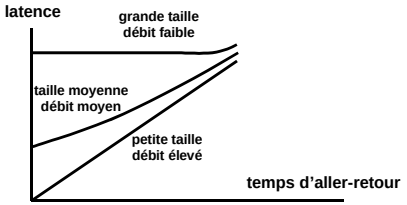
Latence = durée de transmission + temps de propagation + temps d'attente

- Durée de transmission : taille du message / débit
  - c'est le temps nécessaire pour transmettre les données (les envoyer sur le réseau)
- Temps de propagation : distance / vitesse de propagation
  - c'est le temps nécessaire pour que les données aillent de l'émetteur au récepteur
- Temps d'attente
  - c'est le temps "perdu" par le système de communication (notamment à cause de l'occupation des ressources)
- La somme (temps de propagation + temps d'attente), ou latence de base, est un délai incompressible (temps écoulé avant de recevoir le 1-er bit d'un message)

Performances des communications (2)

- **Débit et latence ont un impact variable selon les applications**
  - Exemple 1 : message bref (interaction question-réponse)
    - Envoi 1 Kbit, retour 1 Kbit, distance 10 000 km. Temps incompressible d'aller-retour = 100 ms (si vitesse = 2c/3). Temps de transmission = 1 ms à 1 Mbit/s, 0,01 ms à 100 Mbit/s.
    - Conclusion : la latence est dominante, le débit a peu d'influence
  - Exemple 2 : message très gros (fichier multimédia)
    - Envoi 100 Mbit, distance 10 000 km. Temps incompressible de propagation = 50 ms. Temps de transmission = 100 s à 1 Mbit/s, 1 s à 100 Mbit/s.
    - Conclusion : le débit est dominant, la latence a peu d'influence



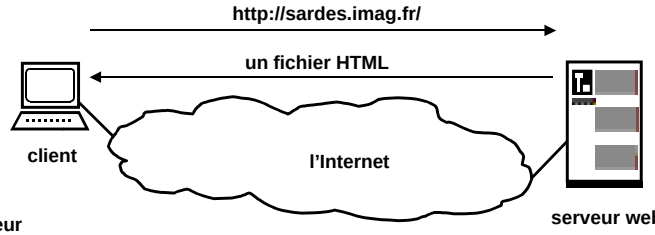


Qualité de service

- **La qualité de service (*Quality of Service*, QoS) désigne un ensemble de facteurs de qualité nécessaires aux besoins d'une application particulière**
  - Cette définition est générique et doit être précisée dans chaque cas
  - Les besoins en QoS dépendent de la nature des applications
- **Exemples**
  - Faible taux d'erreur (probabilité pour qu'un bit ou un message soit perdu ou modifié)
    - nécessaire si les données sont peu redondantes
  - Stabilité de la latence (absence de gigue)
    - nécessaire pour les applications multimédia (son et vidéo)
  - Garantie d'une limite supérieure sur la latence
    - nécessaire pour les applications critiques liées au temps réel
- **Garanties de qualité de service**
  - Problème difficile !
  - Voies d'approche : réservation de ressources, régulation de charge

Comment fonctionne un réseau ? (1)

une requête sur le Web

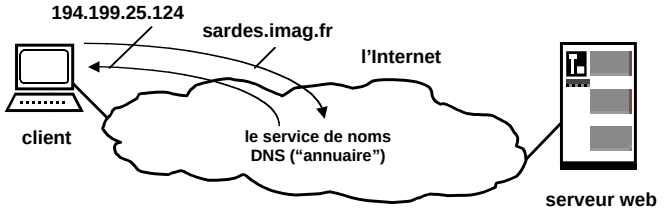


Vu de l'utilisateur
- on clique sur un lien
- une "page web" s'affiche sur l'écran

Sur la station client
- le programme navigateur envoie une requête au serveur correspondant à l'URI (*Uniform Resource Identifier*) associé au lien
- (si tout se passe bien) le programme navigateur reçoit un fichier HTML qu'il sait afficher sur l'écran

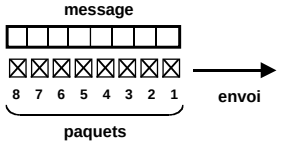
Sur le réseau, il faut :
- trouver le bon serveur (celui qui correspond à l'URI)
- transporter la requête depuis la station client vers le serveur
- transporter le fichier depuis le serveur à la station client

Comment fonctionne un réseau ? (2)

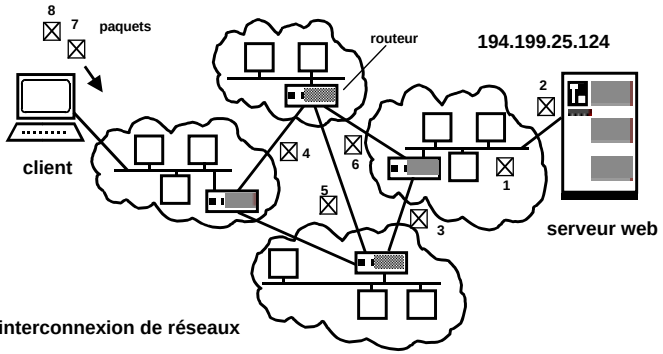


Première étape : localiser le serveur
- Via le service de noms de l'Internet (DNS) qui associe un nom à une "adresse IP" (fonctionne comme un annuaire)
- Toute machine connectée à l'Internet a une adresse IP
  - Question : comment trouver l'annuaire ?
  - Réponse : on connaît l'adresse IP d'un point d'entrée

Deuxième étape : envoyer la requête au serveur
- Comment est transmise la requête ?
- La requête est un message (une suite de bits).
- Elle est découpée en "paquets" de taille fixe
- Chaque paquet est envoyé sur le réseau



Comment fonctionne un réseau ? (3)



L'Internet est une interconnexion de réseaux (*internetwork)*
Les réseaux sont reliés entre eux par des routeurs

Chaque paquet contient son numéro et son adresse de destination. Quand il arrive sur un réseau, et que le site destinataire n'en fait pas partie, le paquet est transmis à un routeur.
Les routeurs contiennent les informations qui permettent d'acheminer le paquet vers son site destinataire

Comment fonctionne un réseau ? (4)

Fonctionnement du serveur

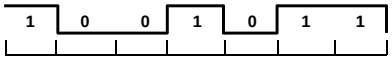
Le serveur reconstruit le message initial en mettant les paquets dans l'ordre de leurs numéros
Que fait le serveur à la réception de la requête ?
Le site serveur interprète la requête comme une demande de fourniture de fichier (HTML)
Il envoie le fichier au client (sous forme d'une suite de paquets, comme précédemment)

Que se passe-t-il si un paquet s'est perdu, ou a mal été transmis ?
Le destinataire est capable de le détecter et demande qu'on lui renvoie le paquet manquant ou erroné.

Transmission de l'information

Comment sont transmis concrètement les paquets ?
Chaque paquet est une suite de bits. La transmission utilise un support physique : fils, fibre optique, ondes radio, etc., sur lequel sont envoyés des signaux (électriques, lumineux, etc.). Chaque bit (0 ou 1) est représenté par une configuration particulière du signal

Exemple (non réaliste)

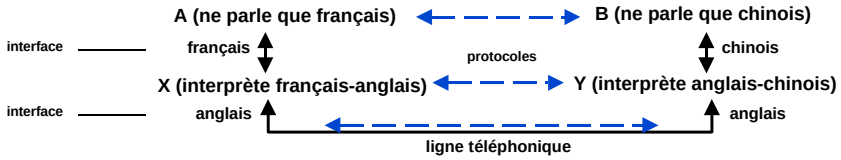


ceci explique le lien entre débit et bande passante

Notions de protocole et d'interface (1)

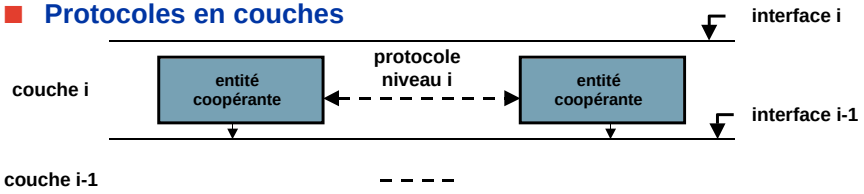
- L'exemple de la requête sur le web permet d'identifier divers "niveaux" d'échange entre le client et le serveur
  - le niveau de l'application : le client clique sur un lien, le serveur renvoie une page web
  - le niveau des messages : le client envoie un message contenant une URI, le serveur renvoie un message contenant un fichier HTML
  - le niveau des paquets : le message du client est découpé en paquets, les routeurs du réseau les acheminent vers le serveur (idem pour le retour)
  - le niveau de la transmission des bits : pour envoyer les paquets, chaque bit (0 ou 1) est transmis comme un signal électrique sur une ligne.
  - chaque niveau utilise les fonctions du niveau inférieur
- Les notions de protocole et d'interface visent à représenter ce mode de fonctionnement

Analogie : deux personnes peuvent dialoguer même si elles ne parlent pas la même langue

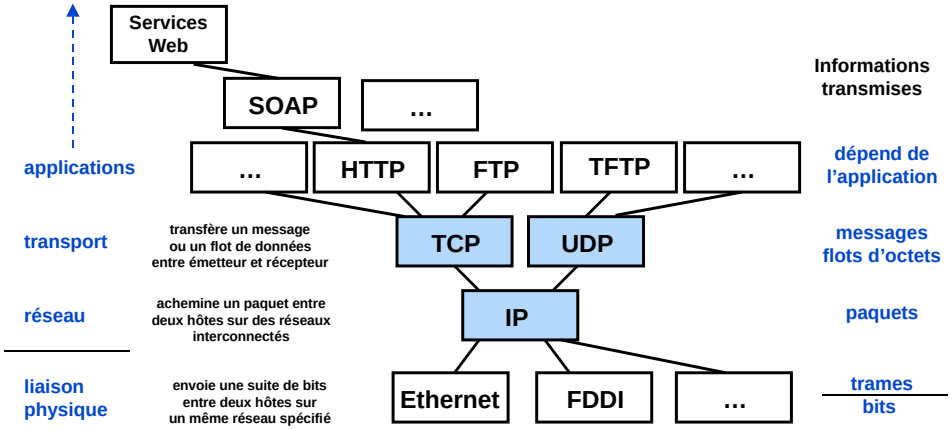


Notions de protocole et d'interface (2)

- Interface (d'un service) : ensemble de fonctions (logicielles ou matérielles) et de règles d'accès utilisables pour accéder au service
- Protocole : ensemble de conventions définissant les échanges entre des entités qui coopèrent pour réaliser un service
- Relations entre protocoles et interfaces
  - une interface définit l'accès à un service, un protocole définit la réalisation d'un service
  - la construction d'un protocole utilise souvent des protocoles de niveau inférieur (plus élémentaires), en accédant à leurs interfaces
- Protocoles en couches



Les protocoles de l'Internet



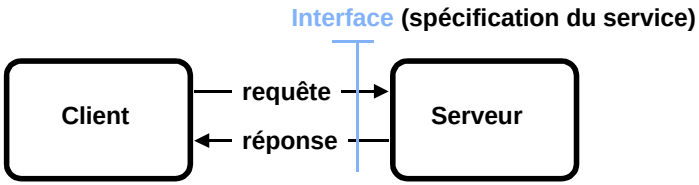
HTTP : HyperText Transfer Protocol : protocole du Web
TFTP, FTP : (Trivial) File Transfer Protocol : transfert de fichiers
TCP :Transmission Control Protocol : transport en mode connecté
UDP : User Datagram Protocol : transport en mode non connecté
IP : Internet Protocol : Interconnexion de réseaux,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

Quelques protocoles au niveau *middleware*

- **Objectif**
  - Permettre la construction, la composition et l'exécution d'applications réparties
  - Fournir des services communs aux applications
- **Position**
  - Utilisent les protocoles de transport des réseaux (souvent via l'interface *socket*)
  - Sont utilisés par les applications finales
- **Exemples**
  - RPC (*Remote Procedure Call*)
  - Java RMI *(Remote Method Invocation*)
  - CORBA : objets répartis hétérogènes
  - HTTP : protocole du World Wide Web
  - SOAP (au-dessus de HTTP) : services Web
  - …

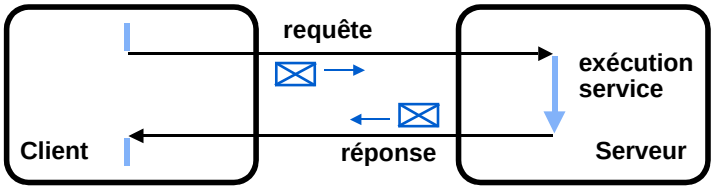
Le modèle client-serveur (1)

- **Définition**
  - Un schéma général d'interaction au niveau application
  - Mis en œuvre sous des formes diverses (différents protocoles)
    - RPC, Java RMI, Services Web, etc.
- **Propriétés**
  - Le *client* demande l'exécution d'un service ; le *serveur* réalise le service
  - Client et serveur sont (en général, pas nécessairement) localisés sur deux machines distinctes
  - Indépendance interface-réalisation



Le modèle client-serveur (2)

- **Communication par messages** (plutôt que par partage de données, mémoire ou fichiers)
  - Requête : paramètres d'appel, spécification du service requis
  - Réponse : résultats, indicateur éventuel d'exécution ou d'erreur
  - Communication *synchrone* (dans le modèle de base) : le client est bloqué en attente de la réponse



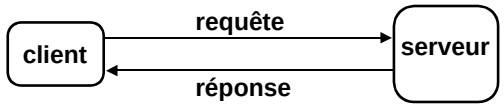
Intérêt du modèle Client-Serveur

- **Structuration**
  - fonctions bien identifiées
  - séparation interface du service - réalisation
  - client et serveur peuvent être modifiés (remplacés) indépendamment
- **Protection**
  - client et serveur s'exécutent dans des domaines de protection différents
- **Gestion des ressources**
  - le serveur peut être partagé entre plusieurs clients

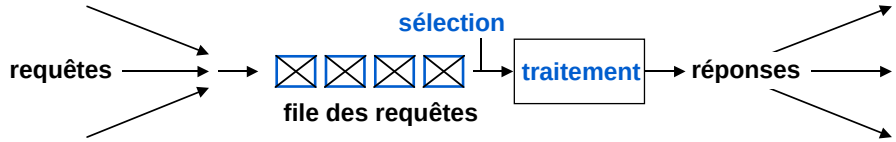
ces considérations sont indépendantes de la répartition

Modèle Client-Serveur : partage du serveur

- **Vu du client**



- **Vu du serveur**
  - Gestion des requêtes (priorité)
  - Exécution du service (séquentiel, concurrent)
  - Mémorisation ou non de l'état du client



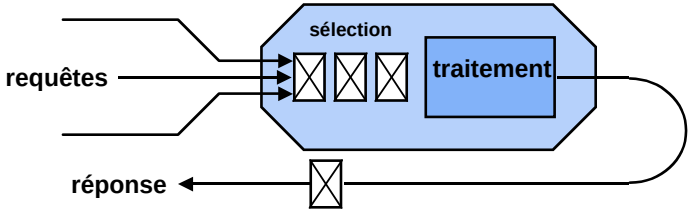
Modèle Client-Serveur : gestion des processus

- **Client et serveur exécutent des processus distincts**
  - le client est suspendu lors de l'exécution de la requête (appel synchrone)
  - éventuellement, plusieurs requêtes peuvent être traitées concurremment par le serveur
    - parallélisme réél (ex : multiprocesseur, entrées -sorties)
    - pseudo-parallélisme
  - la concurrence peut prendre plusieurs formes
    - plusieurs processus (une mémoire virtuelle par processus)
    - plusieurs processus légers (*threads*) dans le même espace virtuel (contexte restreint : pile, mot d'état, registres)

Gestion des processus dans le serveur (1)

- **Processus serveur unique**

```
while (true) {
    receive (client_id, message) ;
    extract (message, service_id, params) ;
    do_service[service_id] (params, results) ;
    send (client_id, results) ;
} ;
```



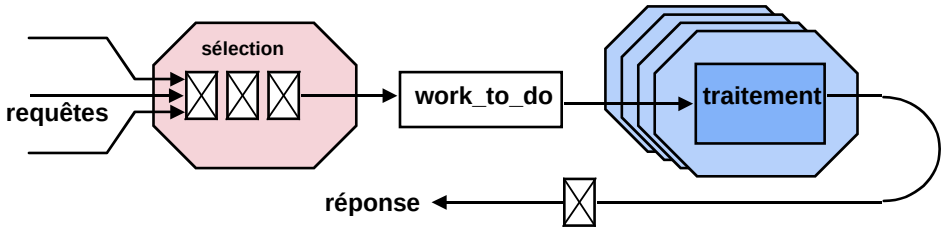
Gestion des processus dans le serveur (2)

Processus veilleur

```
while (true) {
      receive (client_id, message)
      extract (message, service_id,
                params) ;
      work_to_do.put (client_id,
          service_id, params) ;
};
```

Pool d'exécutants

```
while (true) {
      work_to_do.get (client_id,
          service_id, params) ;
      do_service{service_id]
          (params, results)
      send (client_id, results)
      };
```



Gestion des processus dans le serveur (3)

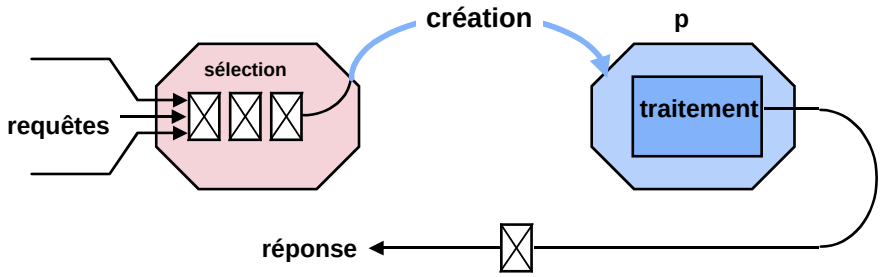
Processus veilleur

```
while (true) {
      receive (client_id, message)
      extract (message, service_id,
                params) ;
      p=create_thread (client_id,
          service_id, params) ;
};
```

Création dynamique des exécutants

programme de p

```
do_service{service_id]
    (params, results)
send (client_id, results)
exit ;
```



Mise en œuvre du schéma client-serveur

- **Par des opérations de "bas niveau"**
  - Utilisation de primitives du système de communication
  - Exemple : *sockets*
    - Mode non connecté (UDP)
    - Mode connecté (TCP)
- **Par des opérations de "haut niveau"**
  - Utilisation d'un *middleware* spécialisé
  - Contexte : langage de programmation
    - Appel de procédure à distance
  - Contexte : objets répartis
    - Appel de méthodes, création d'objets à distance

Plan de la suite

- **Introduction aux réseaux informatiques** (P. Sicard)
  - Séance 2 : Couches et protocoles : TCP UDP IP ARP
  - Séance 3 : Couche de liaison, détect/correct. Erreurs. Exemple : Ethernet
  - Séance 4 : Contrôle de flux. Exemple : TCP
  - Séance 5 : Routage. Mise en œuvre dans IP ; Exemple : RIP
- **Introduction à l'intergiciel et à la construction d'applications réparties** (S. Krakowiak)
  - Séance 6 :, Mise en œuvre du modèle client-serveur (*sockets*, RPC)
  - Séance 7 : Objets répartis : principes, exemple de Java RMI
  - Séance 8 : Services répartis : exemples (DNS, Web)

TP1 : Réseau Ethernet : mise en place, observation	TP4 : Appli. répartie avec *sockets*
TP2 : Observation et analyse de protocoles : ARP, TCP, UDP	TP5 : Appli. répartie avec Java RMI (1)
TP3 : Observation et analyse de protocoles : IP, routage	TP6 : Appli. répartie avec Java RMI (2)

Références

- **Bibliographie sommaire sur les réseaux**

•L. Toutain, *Réseaux locaux et Internet*, 2ème édition, Hermès, 1999

•J. F. Kurose, K. W. Ross, *Computer Networking, a top-down approach featuring the Internet*, Addison-Wesley, 2001

•L. Peterson, B. S. Davie, *Computer Networks, a systems approach*, 2nd edition, Morgan Kaufmann, 1999 (il existe une traduction française, mais seulement pour la 1-ère édition actuellement)

•A. Tanenbaum, *Les réseaux*, 3-ème édition, Dunod, 1999

- **Bibliographie sommaire sur le *middleware***

•W. Emmerich, *Engineering Distributed Objects*, John Wiley, 2000

•W. Grosso, *Java RMI*, O'Reilly, 2001

•J.-M.Geib, Ch.Gransart, Ph.Merle, *Corba : des concepts à la pratique*, 2ème éd., InterEditions, 2001

Compléments

Matériau complémentaire, non traité en cours

Problèmes communs

(à tous les niveaux)

- **Désignation (*naming*)**
  - Fonction : désigner (identifier) une entité physique ou logique : machine, réseau, service, fichier, etc.
  - Principe : "espace de noms", règles de composition des noms
    - Exemples : adresses physiques en mémoire ; adresses de machines sur un réseau local (48 bits) ; numéros de porte ; nom de *sockets ;* adresses IP sur l'Internet (32 bits, IPv4, 128 bits, IPv6) ; noms de domaines sur l'Internet ; noms de services
  - Outils : service de noms
    - Fonction : resolution des noms (passer d'un nom à la localisation de l'entité qu'il désigne)
    - Problème : comment commencer (trouver le service de noms)
- **Liaison (*binding*)**
  - Fonction : pouvoir accéder à l'entité désignée par un nom
    - Liaison ≠ résolution : connaître la localisation d'un objet ne suffit pas pour y accéder : il faut un protocole pour réaliser l'accès effectif
  - Nature et moment de la liaison
    - Statique : une fois pour toutes (ex : adresses sur un réseau local)
    - Dynamique : au moment de l'accès (ex : adresses IP sur l'Internet, via routage ; noms de domaines sur l'Internet, via DNS ; services d'applications via service de noms ou de courtage)

Désignation

- **Définition d'un nom**
  - Information attachée à une entité et remplissant 2 fonctions
    - Désigner cette entité (la distinguer des autres) => identificateur
    - Permettre d'accéder à cette entité (la localiser) => référence
- **Exemples de noms**
  - Dans un programme
    - Identificateur = nom de variable, de méthode, etc.
    - Référence = adresse de la représentation correspondante en mémoire
  - Dans un SGF
    - Identificateur = nom symbolique de fichier (/usr/bin/prog)
    - Référence = i-node correspondant (permet de localiser le contenu du fichier, par les adresses sur disque des blocs qui le composent)
  - Sur un réseau
    - Identificateur = nom symbolique d'une machine (kernighan.imag.fr)
    - Référence = adresse IP (195.221.225.6) : réseau (+ sous-réseau) + n° hôte

Organisation des noms

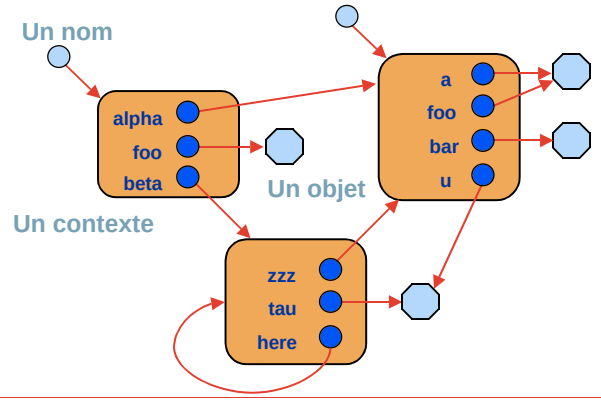
- **Système de désignation**
  - Règles permettant de désigner une catégorie d'entités (usagers, fichiers, machines, ec.)
- **Espace de noms**
  - Ensemble des noms utilisables dans un système de désignation

Un espace unifome ("plat") est peu commode :

- recherche peu efficace
- regroupement impraticable

D'où une organisation par contextes (locaux)

Nombreux exemples dans les réseaux et applications réparties



Désignation et liaison

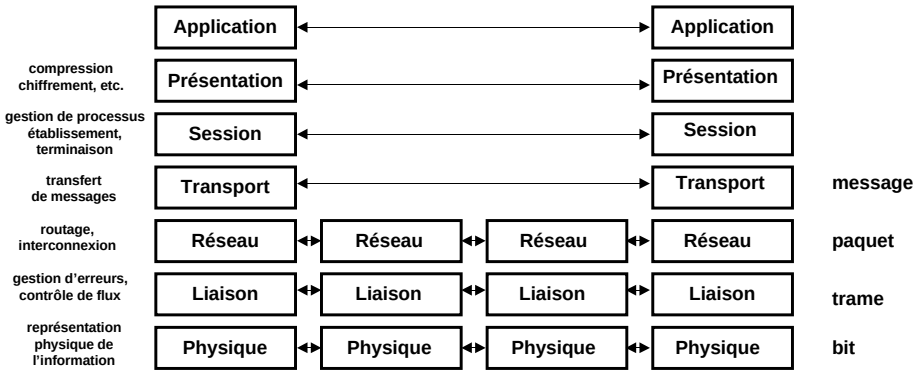
- **Résoudre un nom** (dans un contexte)
  - À partir du nom, trouver l'objet
  - Processus récursif

    target = *context.resolve*(*name*) [ou name.resolve()]

    3 issues possibles pour *target*
    - Une valeur typée : c'est l'objet
    - Une référence (ex : adresse) => l'objet est localisé
    - Un autre nom (dans un autre contexte) => on rappelle *resolve*
- **Lier un nom**
  - À partir du nom, construire une chaîne d'accès à l'objet
  - Différence majeure entre centralisé et réparti :
    - En centralisé, la référence donne accèsà l'objet désigné (exemple : adresses, i-nodes) ; donc résolution = liaison
    - En réparti, une référence (adresse IP, n° de porte) ne suffit pas pour accéder à l'objet ; donc résolution ≠ liaison !
    - La liasion nécessite de construire explicitement une chaîne d'accès sous forme de protocoles de communication et de points d'entrée

Les protocoles normalisés de l'ISO (*International Standards Organisation*)

Open Systems Interconnection (OSI)



Les protocoles OSI servent plutôt de cadre de référence pour la définition des fonctions que de normes de réalisation. La normalisation de fait est autour de TCP/IP