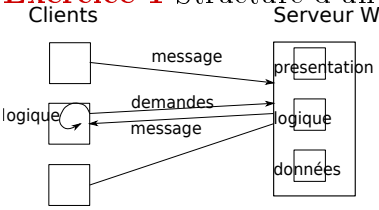
Pourquoi des applications réparties ?

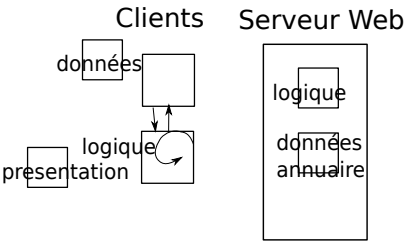
- Partage et mise en commun des ressources
- Domaines nouveaux d'applications
- Intégration d'objets du monde réel : informatique pervasive
- Convergence forte entre informatique et télécommunications
- Parallélisme pour la puissance de traitements (grid computing)

Définition : Qu'est-ce qui caractérise une application répartie ?

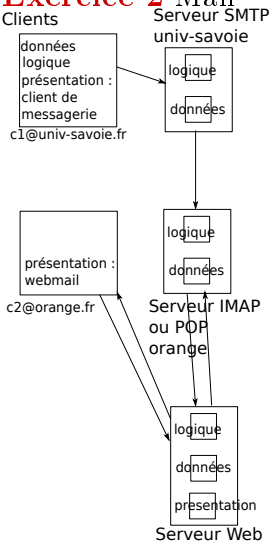
- Le temps
- La distance
- Multimachine
- Persistance des données

Exercice 1 Structure d'un chat





Exercice 2 Mail



Exercice 3 Service d'échanges de fichiers

Ce que doit offrir un système réparti

- Transparence à la localisation
- Transparence d'accès
- Transparence à l'hétérogénéité
- Transparence aux pannes

- Transparence à l'extension des ressources

Tendance générale : Intégration de haut niveau

Le client Effectue une demande de service (une requête). C'est lui qui initie le contact.

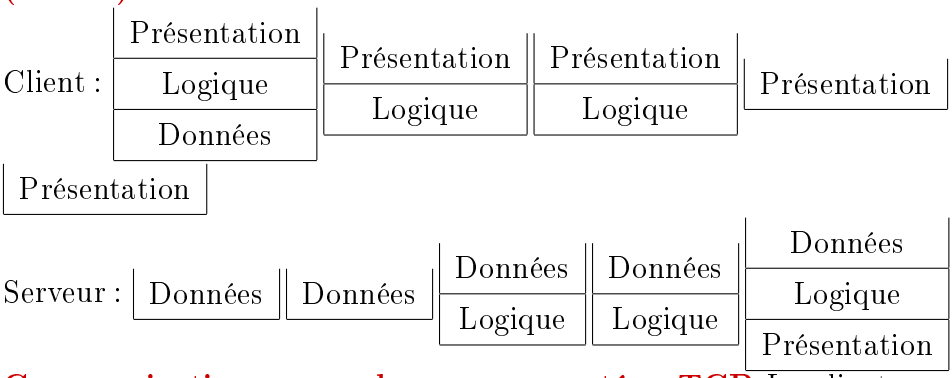
Le serveur Offre le service, répond aux requêtes.

Système coopératif Les parties clients et serveurs peuvent s'exécuter sur des systèmes différents. Un serveur peut répondre à plusieurs clients.

Problème de temps principalement pour la voix et la visioconférence

Couches concernées applicative, traitement(= logique), données.

Questions : Comment répartir le découpage d'une application informatique de type clients / serveurs ? **Modèle de Gartner (2 tiers)**



Communication en mode non connecté : TCP Le client émet une requête, le serveur se réveille et répond.

Communication en mode connecté : UDP Le client fait une demande de connexion, le serveur crée un contexte, le client en-

voie des requêtes et le serveur répond.

Serveurs itératifs ou concurrents

- Itératif : traite séquentiellement les requêtes (souvent en mode non connecté)
- Concurrent : les serveur accepte les requêtes puis les délègue à un processus fils (souvent en mode connecté)

Notion d'état

- Service avec états (mode connecté). Le serveur conserve localement un état pour chacun des clients connectés
- Service sans état : ne conserve aucune info sur l'enchaînement des requêtes
- Sans état est plus performant, Avec état est plus résistant aux pannes

Mémoire partagée

- Communications locales : processus sur une même machine
- Communication distante : la mémoire partagée est physiquement répartie.

Communication par passage de messages

- Les processus n'ont pas accès à des variables communes
- Ils communiquent en s'échangeant des messages : 2 primitives : send et rcv

- ...
- Eviter les écritures concurrentes
- Opérations bloquantes / non bloquantes :
  - rcv peut rendre la main quand les données ont été reçues et recopiées depuis le tampon de réception local.
  - send peut rendre la main : aussitôt (ou plus tard par sécurité : attente de la réception ou de l'enregistrement ou de la consommation des données par le receveur).

Interface de programmation

Sockets : principes généraux canal de com entre deux programmes ou processus. Echange peut se faire dans les deux sens une fois le canal de com en place. Au dessus de la couche transport.

4 types

- Stream
- Datagram
- Raw
- Sequenced packet

Exercice : à quoi correspondent ces primitives ?

- socket() : création

- close() : fermeture
- accept() : accepter la connection (en mode connecté)
- send() : envoie (en mode connecté)
- bind() : relier le socket à un port
- recvfrom() : recevoir (en mode non connecté)
- connect() : (en mode connecté : TCP)
- receive() : recevoir (en mode connecté)
- listen() : écouter sur un port (serveur)
- sendto() : envoie (en mode non connecté)