Modèle Client / Serveur

Introduction à l'architecture client serveur

Mise en œuvre dans l'environnement TCP/IP

Exemples de services Internet basés sur ce modèle

- Service de gestion de noms (Domain Name Service)
- Le Web

Limites du modèle client serveur pour la mise en œuvre des traitements répartis

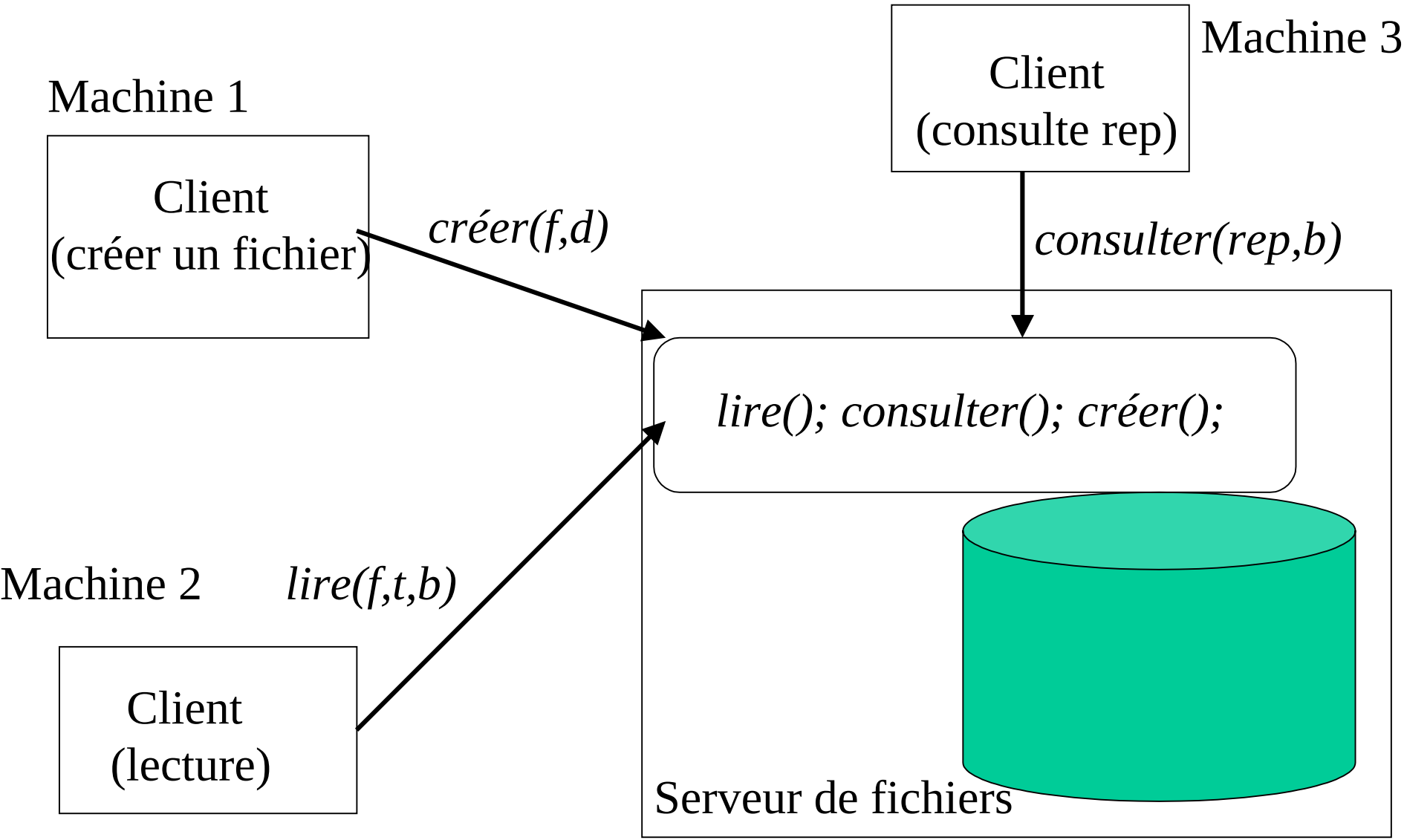
Architecture client serveur

Application = un ensemble de processus coopérants

- Deux rôles pour les processus de l'application :
- ***Serveur***
  - Réalise les services offerts aux utilisateurs et gère l'état de l'application
  - Offre une interface permettant aux processus clients de requérir des services pour leurs utilisateurs. Une interface définit l'ensemble des opérations acceptables. Une opération est caractérisée par son type et ses arguments
- ***Client***
  - Fait appel aux services réalisés par des serveurs
  - Peut être un processus autonome ou un logiciel servant point d'accès au service pour les utilisateurs finaux (Browser Web)

- Exemples de services se conformant à cette architecture
  - Network File System, Word Wide Web, Domain Name Service

Exemple d’architecture client-serveur



Mise en oeuvre basée sur le protocole d'appel / réponse

API de base offre deux méthodes

- send(message, taille_message, destinataire)
- receive(buffer, taille_buffer, adresse_emetteur, taille_adresse)

Quand un client veut un service

- Le client génère un message contenant sa requête et l'envoie au serveur en utilisant la méthode *send()*

À la réception du message par le serveur grâce à receive(), il

- Extrait la requête du message d'appel
- Identifie ensuite le service demandé et l'exécute.
- À la fin de l'exécution, le serveur génère un message contenant la réponse et l'envoie au client en utilisant la méthode *send()*

Le client reçoit le message de réponse en utilisant la méthode *receive()* et extrait la réponse à sa requête.

Synchronisation des appels à send et receive

- Schéma asynchrone avec send non bloquant

  L'émetteur reprend la main dès que la demande est traitée localement.

  Mécanismes nécessaires pour que celà fonctionne

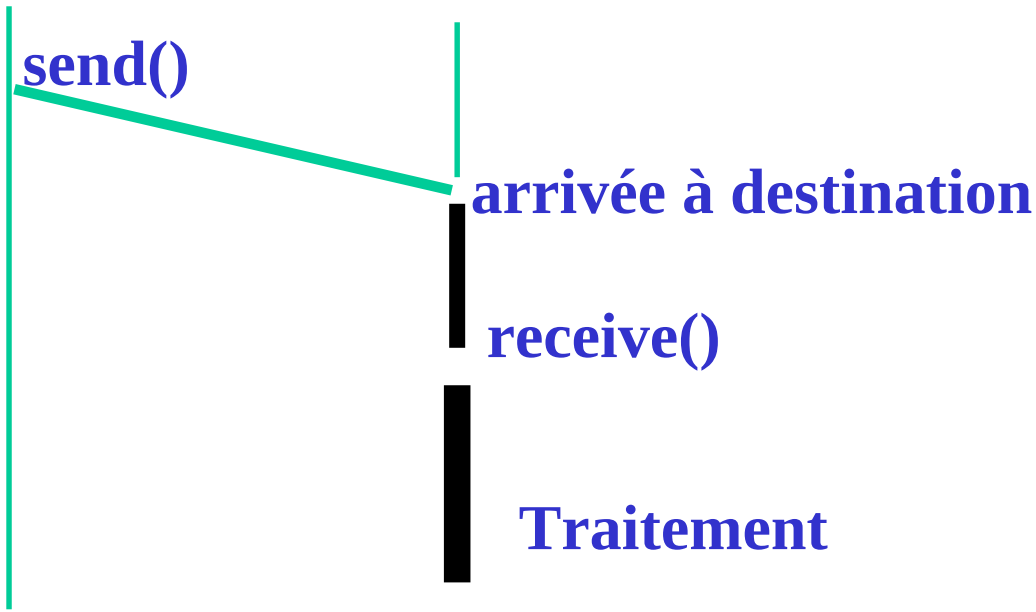
  - Gestion des buffers pour stocker les messages non attendus ?
  - Copie des buffers du côté émetteur ou mécanisme de notification de fin de transmission

  Rend le système qui l'utilise difficile à débugger
- Schéma asynchrone avec send bloquant

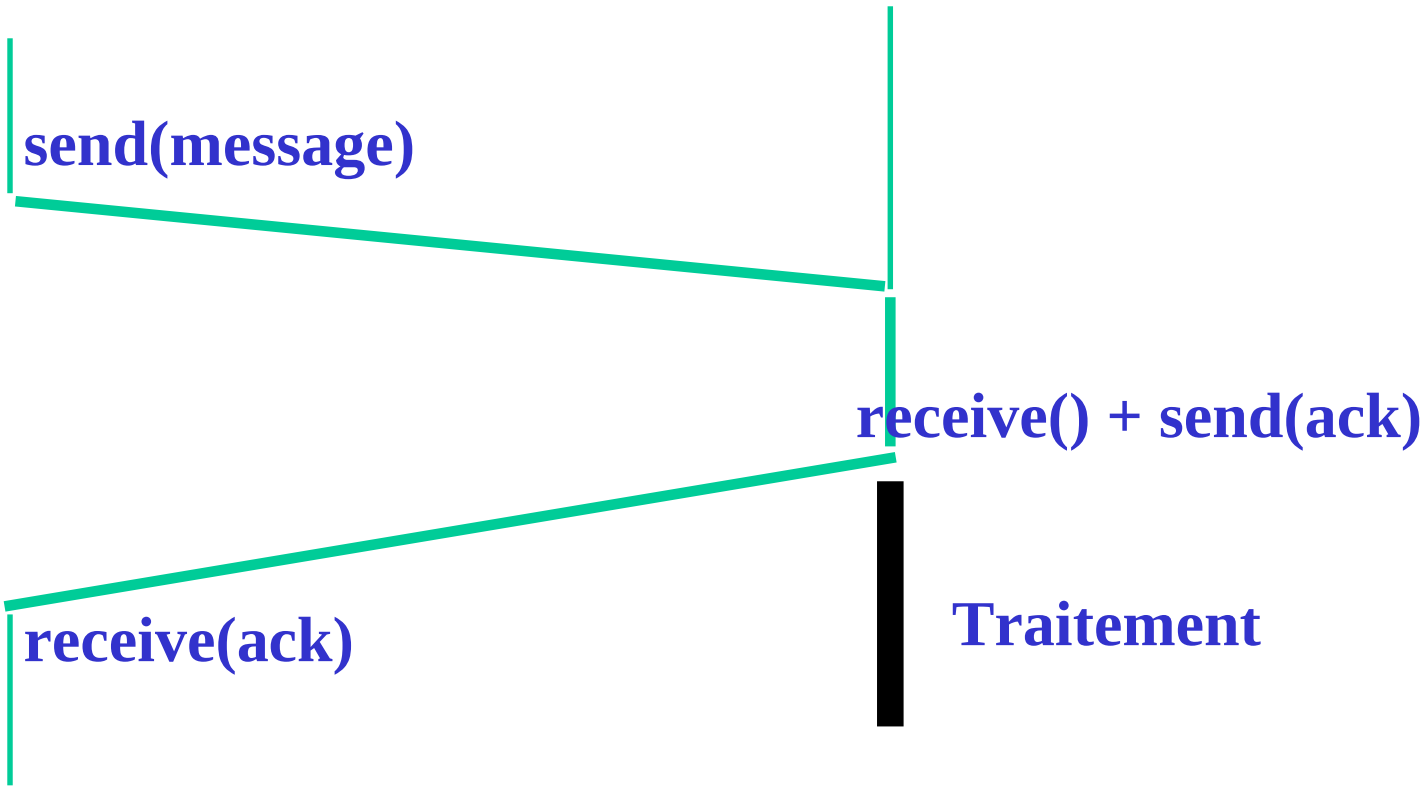
  L'émetteur est bloqué jusqu'à ce que le messsage soit transmis
- Le rendez-vous entre send et receive

  Send (ou receive) est bloqué jusqu'à ce que son homologue soit exécuté.

Schéma asynchrone



Rendez-vous



Que faire des messages lorsque le récepteur n'est pas prêt ?

Les supprimer

- Simple mais peut provoquer des rétransmissions inutiles
- Détection des fausses pannes et limitation du parallélisme

Bufférisation pour une durée limitée

- Taille du buffer ? Allocation dynamique ou pool pré-alloué ?

Stocker des messages en attente dans des mail boxes?

- Association d'une boîte à lettres à chaque adresse de réception.
- Tout message envoyé à cette adresse est déposé dans la boîte.
- Les appels à receive() prélèvent les messages de la boite.

Fiabilité

Problème : pertes de messages d'appel ou de reponse

- Fautes de communication
- Crashes de l'appelé avant d'avoir retourné la réponse
- Crashes de l'appelant laissant l'appel orphelin

Solutions

- Laisser la charge à l'utilisateur : un message de réponse acquitte la réception du message d'appel
- Le protocole de coopération des garanties concernant la sémantique des accès à distance

Garanties possibles pour des accès à distance

- Le service sera exécuté *peut-être une fois* (*maybe*)
- Le service sera exécuté *au moins une fois* (*at least once*)
- Le service sera exécuté *au plus une fois* (*at most once*)
- Le service sera exécuté *exactement une fois* (*exactly once*)

Localisation de serveurs

- Un client a besoin de connaître la localisation du serveur
  - L'adresse IP de la machine sur laquelle se trouve le serveur
  - Le protocole (TCP ou UDP) et le numéro du port pour communiquer avec le serveur
- Deux solutions de base
  - Disposer des informations statiques
    - Nom de la machine serveur
    - Protocole et port fixés statiquement
    - ➔ Besoin d'un service pour traduire le nom du serveur en adresse IP (DNS)
  - Utiliser un service de liaison

Serveur de liaison

- Assimilable à un serveur de noms
  - Gestion des associations (interface, localisations)
  - Interface de base offerte par un serveur de liaison:
    - Enregistrer(nom_interface, localisation_serveur)
    - Retirer(nom_interface, localisation_serveur)
    - Rechercher(nom_interface)
- Permet
  - la transparence de la localisation
  - La transparence de la migration de serveur grâce à
    - Une coopération entre le serveur qui migre et le serveur de liaison
    - Un mécanisme de reliaison au niveau du client.

Localisation du serveur de liaison

- Serveur de liaison sur une machine dédiée
  - Identificateur connu de tous
  - Re-compilation des clients et serveurs lorsque la machine dédiée au serveur de liaison change
- Fourniture des informations par les clients et serveurs
  - Variables d'environnement
  - Implique mises à jour des variables d'environnement en cas de changement.
- Serveur banalisé
  - Peut être lancé sur une machine quelconque
  - Localisation grâce à un protocole de découverte de ressource

Appel / Réponse : limitations

- Pas de transparence d’accès
  - Accès aux serveurs distants différents des accès aux serveurs locaux
- Pas de transparence de localisation
  - Le client manipule les identificateurs de communication
  - Migration du serveur invalide son identificateur
- Communications non typées
  - Pas de garantie du type de données envoyées au serveur
  - Pas de moyen de vérifier le type de données reçues

Autres paradigmes de mise en œuvre

Appel de procédure à distance

- Transparence d'accès

Invocation d'objet distant

- Permet de bénéficier à la fois de l'approche objet et du modèle client – serveur
- Exemples: Java RMI et CORBA

Partage de mémoire répartie