Synchronisation Distribuée
Introduction

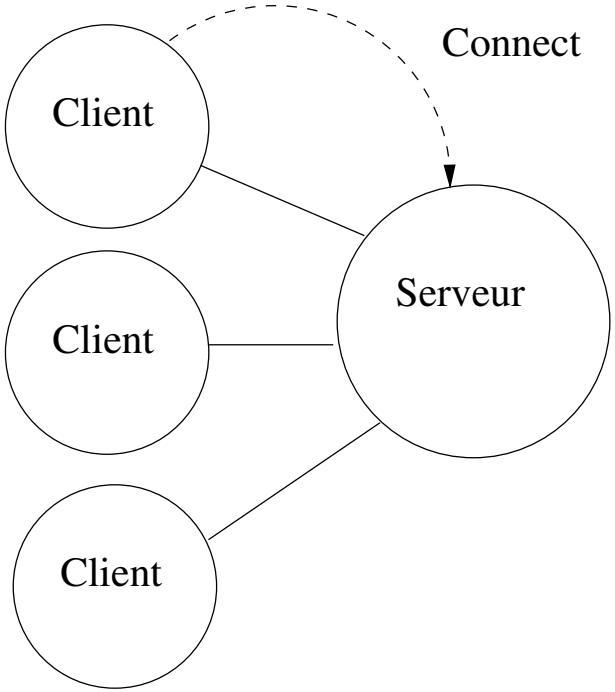
Laurent PHILIPPE

Master 2 Informatique
Ingénierie des Systèmes et Logiciels

2021/2022

Processus communicants

- API communication :
  - sockets : asynchrone, `send`, `recv`
  - RMI : synchrone, appel de fonction
- Problématiques :
  - Positionnement des données entre les processus
  - Quelles données sont échangées ?
  - Synchronisation : qui initie vs. attend la communication ?



Communication Client-Serveur

- Asymétrique
- Centralisée sur le serveur
- Pas de communication entre clients
- Synchronisation interne au serveur

Systèmes synchronisés (centralisés)

- Plusieurs processus/threads
- Problèmes : partage/accès à des ressource(s), réaliser une action en commun, ...
- Éventuellement différents modes d'accès
- Objectif est de garantir le bon déroulement
- Propriétés : famine, inter-blocage, équité, ...
- Solutions avec mémoire partagée : mutex, sémaphores, monitors, ...

Systèmes communicants **ET** synchronisés

- Plusieurs processus sur des ordinateurs différents
- Réaliser un objectif en commun...
- ... uniquement avec des communications

Qu'est-ce que la synchronisation distribuée ?

De l'algorithmique...

- Solution à un problème de traitement
- Tri
- Graphes
- Synchronisation

... Pour les systèmes distribués

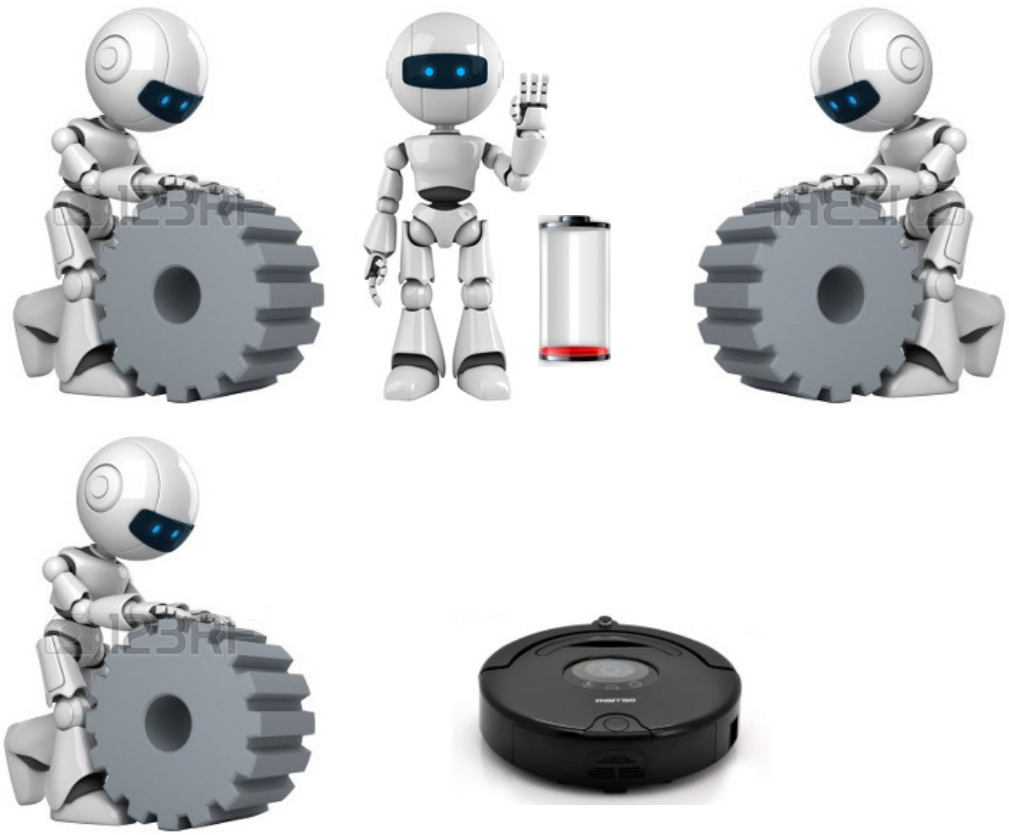
- Réalisation de l'algorithme à plusieurs
- Introduction de la communication

=> Contraintes spécifiques

Exemple

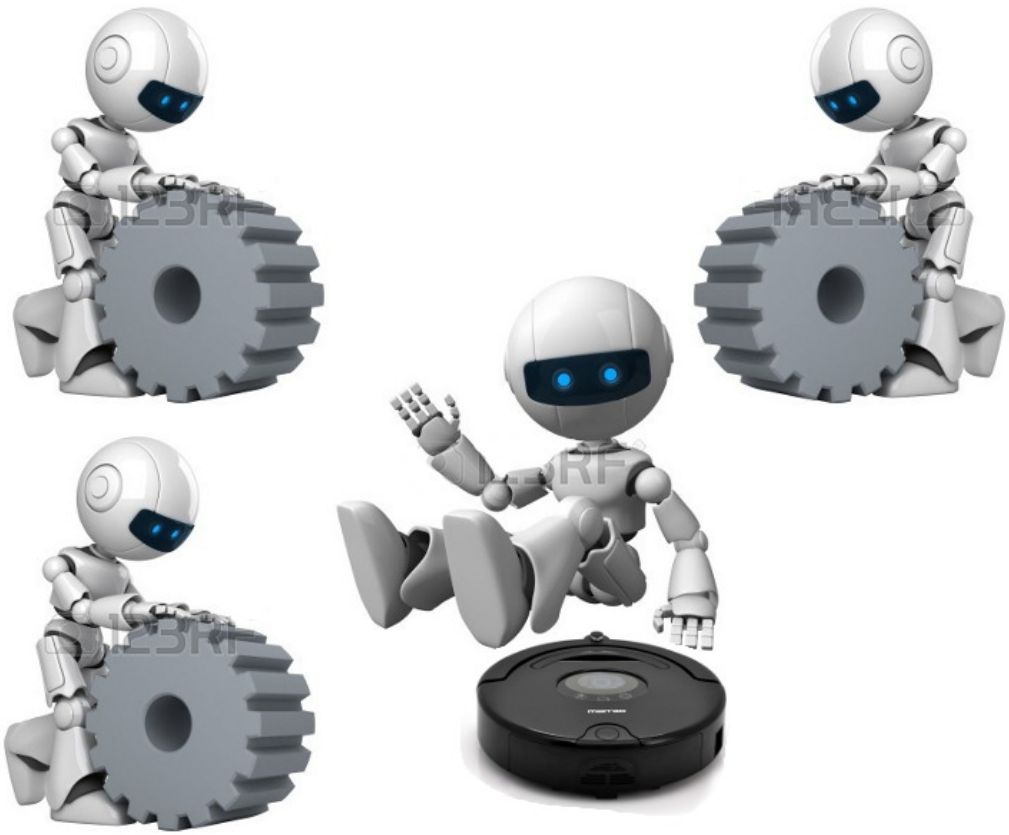
Un ensemble de robots autonomes et leur base de rechargement

Exemple



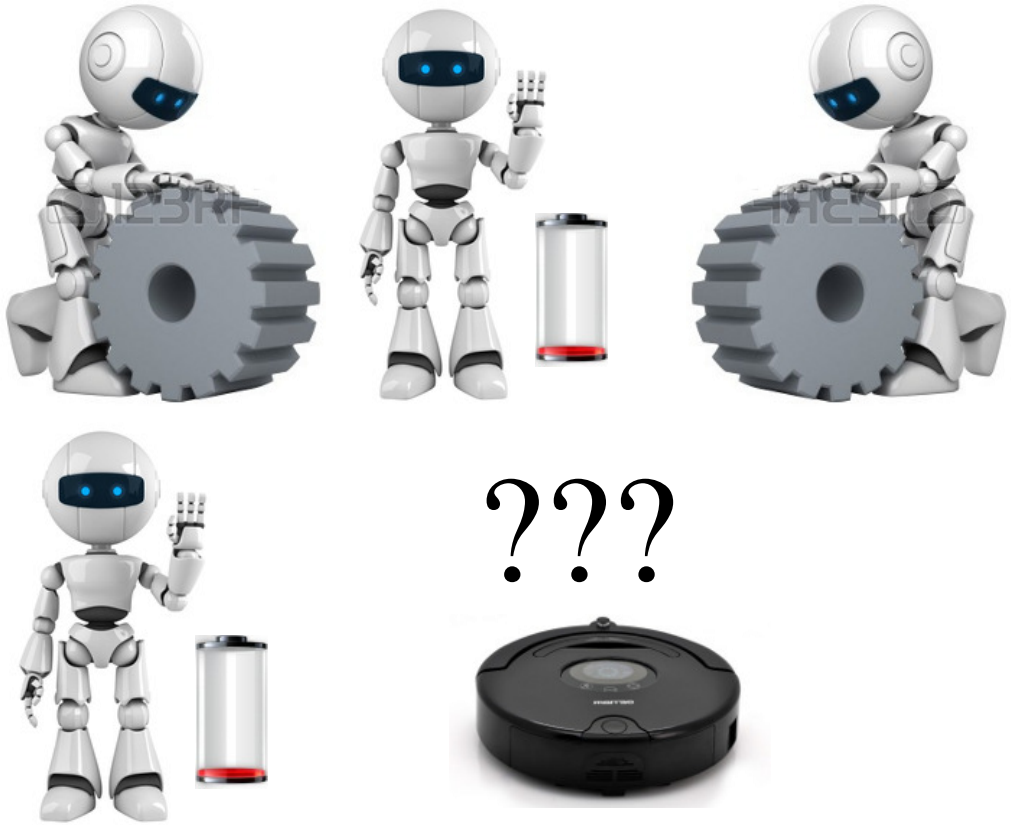
Évènement : la batterie d'un robot passe sous le seuil d'alerte

Exemple



Le robot va se recharger

Exemple



Il y a conflit si deux robots ont besoin de recharger au même moment : comment résoudre le problème ?

Exemple

Les robots

- Autonomes : processeur, mémoire
- Peuvent communiquer par échange de messages
- Autonomie en batterie dépend de la tâche -> variable

La base

- Ne communique pas
- Un seul robot se recharge à la fois

Problème

- Une seule base
- Les robots doivent se mettre d'accord

=> Concevoir un algorithme pour éviter les conflits

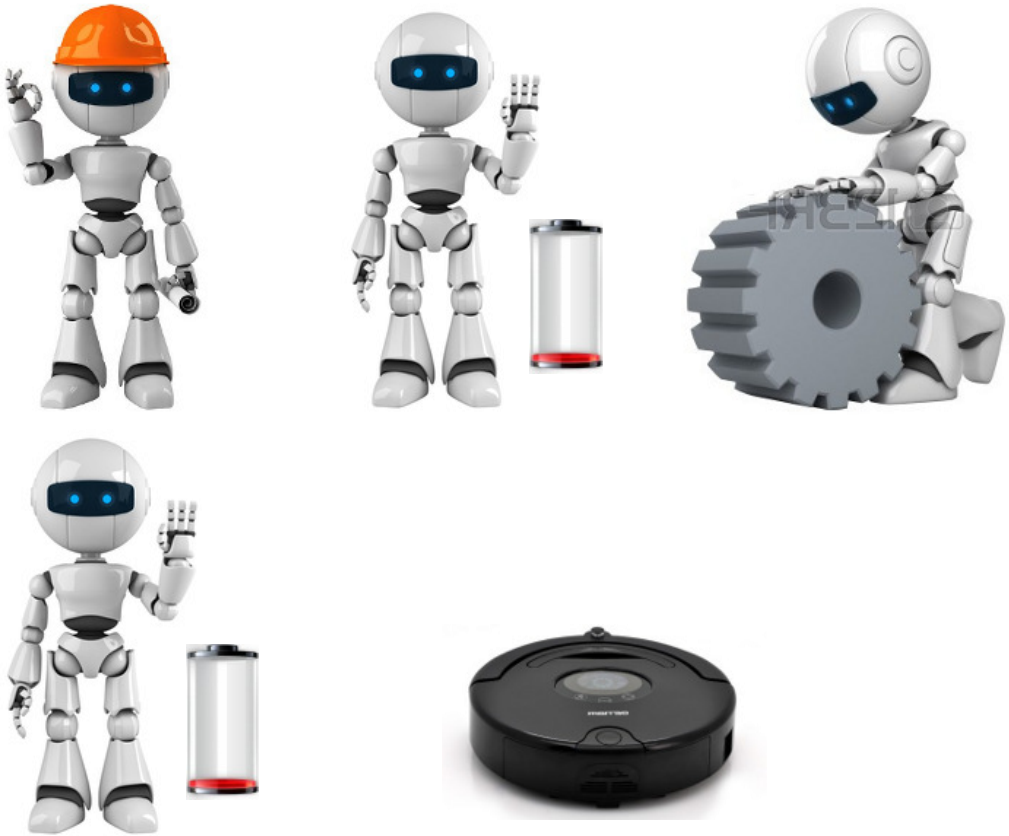
Exemple

Outils pour écrire l'algorithme

Pour communiquer : modèle de communication simple

- Chaque robot dispose d'un identificateur : numéro `num`
- Fonctions de communication :
  - Envoyer un message : P_i **envoie** `TYPE(données)` à P_j
  - Recevoir un message : P_i **recoit** `TYPE(données)` de P_j
- Fonctions internes :
  - Attente : `attendre( condition )`
  - Aller à la base et se recharger : `charge( tps )`

Exemple



Solution simple : nommer un robot maître, décide qui accède à la base

Exemple

Variable pour chaque robot R_i :
$acces_i$: booléen, droit d'accès à la base, initialisé à FAUX
Variable pour le robot maître R_0 :
util : booléen, base utilisée, initialisé à FAUX
file : file de demandes, initialisée à vide

Messages utilisés :
$REQ()$: Message de demande d'accès
$ACK()$: Message d'autorisation d'accès
$REL()$: Le robot a libéré la base

Exemple

Règle 1 : *Alerte batterie de* R_i
début
    R_i **envoie** $REQ()$ à R_0
    attendre($acces_i$ = VRAI)
    charge()
    R_i **envoie** $REL()$ à R_0

Règle 2 : R_0 **reçoit** $REQ()$ *de* R_i
début
    si *util* = *FAUX* **alors**
        R_0 **envoie** $ACK()$ à R_i
        util ← VRAI
    sinon
        queue(file) ← R_i

Règle 3 : R_i **reçoit** $ACK()$ *de* R_0
début
    $acces_i$ ←VRAI

Règle 4 : R_0 **reçoit** $REL()$ *de* R_i
début
    R_j ← tête(file)
    si $R_j \neq NULL$ **alors**
        R_0 **envoie** $ACK()$ à R_j
    sinon
        util ← FAUX

Exercices : synchronisation des robots

Exercices

Plan

Plan du cours

- Contexte et modèles
- Synchronisation distribuée
- Datation dans les systèmes distribués
- Synchronisation distribuée baséé sur la date
- Diffusion fiable