La disparition de l’ordinateur

Peter Van Roy
Département d’Ingénierie Informatique
Université catholique de Louvain
21 octobre, 1996

La crise du logiciel (1)

- Le réseau téléphonique de AT&T (1990)
  - Un réseau énorme qui couvre tous les Etats-Unis
  - Une panne d'un seul intérrupteur active l'intérrupteur de réserve, ensuite celui-ci tombe en panne aussi à cause d'une erreur de logiciel
  - Le réseau tombe en panne en cascade, causant une panne générale du service téléphonique pour 60000 utilisateurs pendant neuf heures
  - Problème: tolérance aux pannes du réseau

La crise du logiciel (2)

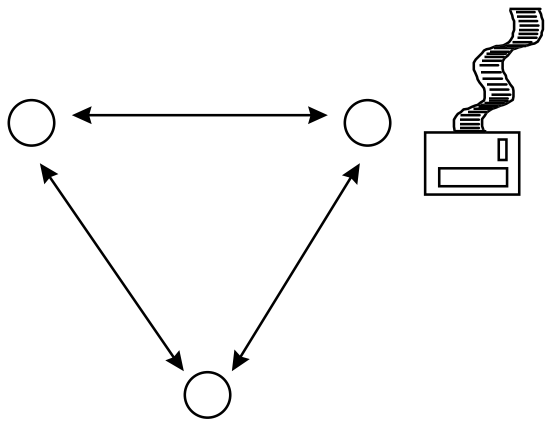
- L’aéroport international de Denver, Colorado (1993)
  - Gestion des bagages:
    - 100 ordinateurs en réseau, 5000 «yeux électriques», 400 récepteurs radio, 56 scanners bar-code
    - Coût: $193 million
  - L’ouverture de l’aéroport a été longtemps retardé à cause du logiciel responsable de la gestion des bagages
  - Problème: coordination de tâches réparties

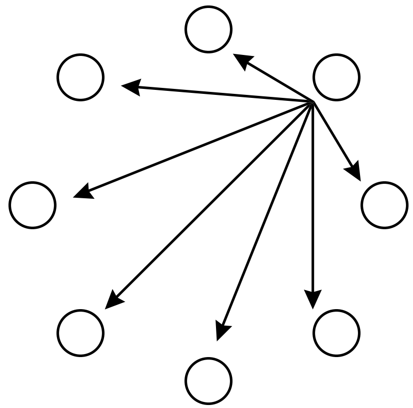
Le problème de la répartition

- La croissance exponentielle du nombre d'ordinateurs et de réseaux d'ordinateurs est un fait
- Dans l'avenir, l'ordinateur seul disparaîtra, pour être remplacé par un réseau de réseaux
- Un réseau d'ordinateurs est tout autre chose qu'un seul ordinateur!
- Il est important de comprendre ceci du point de vue théorique et pratique
- C'est le but du projet «Distributed Oz» à INGI

Pourquoi la répartition?

- Echange d'informations
- Partage de ressources
- Coordination de tâches
- Fiabilité
- Performance
- Modularité

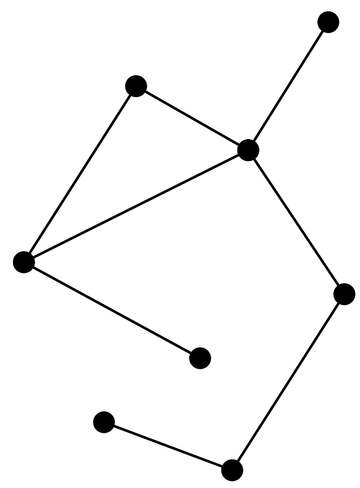




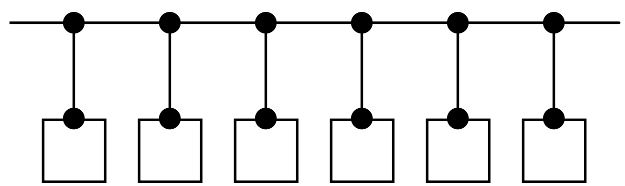
Qu'est-ce qu'un systéme réparti?

- Un ensemble de processus autonomes, lié par un réseau
- Caractéristiques:
  - Pas d'informations globales (delais imprévisibles)
  - Pas de temps global (pas d'ordre total entre les événements)
  - Concurrence et nondéterminisme
  - Grande probabilité de pannes localisées

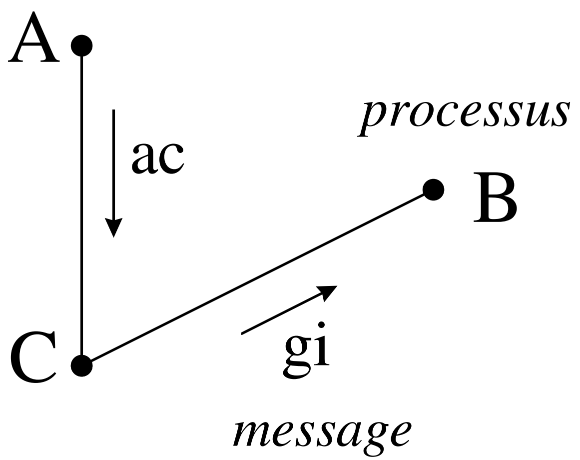
Exemples de systèmes répartis



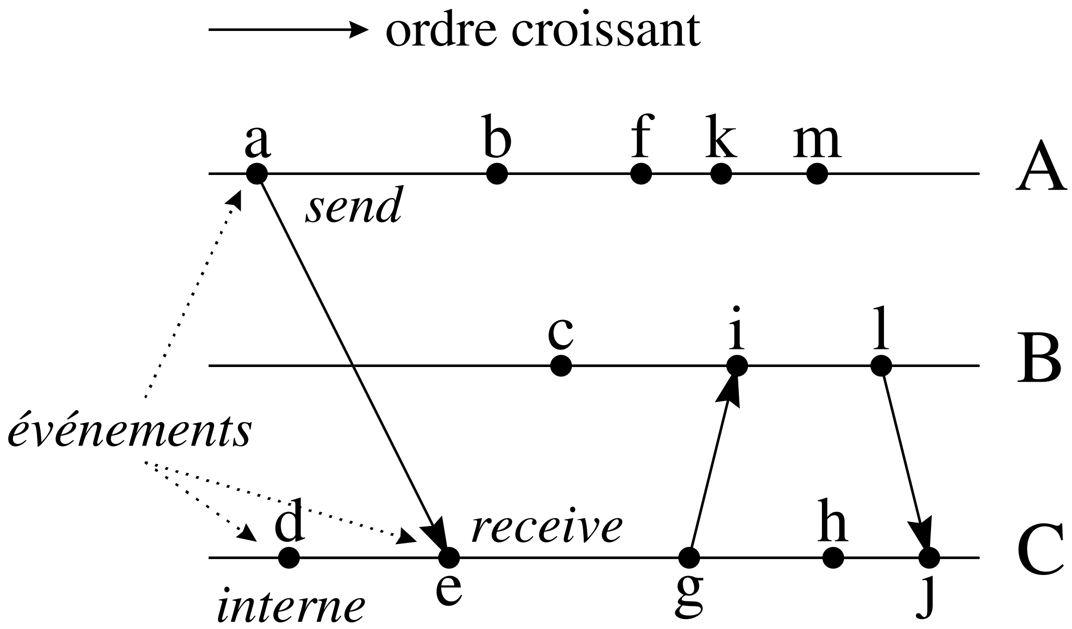
- WAN (wide-area network)
  - Exemple: Internet = réseau de réseaux, chaque utilisateur a son adresse personnelle
  - Couches réparties sur Internet: Web, messagerie, newsgroups, ftp, telnet, ...
- LAN (local-area network)
  - Typiquement pour échange d'informations et partage de ressources



Modèle théorique



Systéme réparti



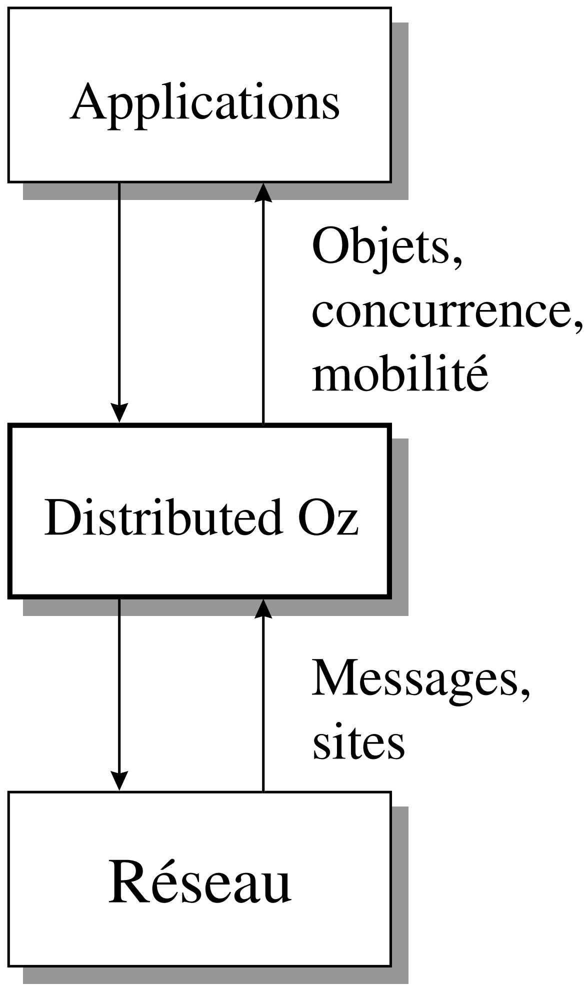
Modèle: système de transitions

Faciliter la programmation d’un système réparti

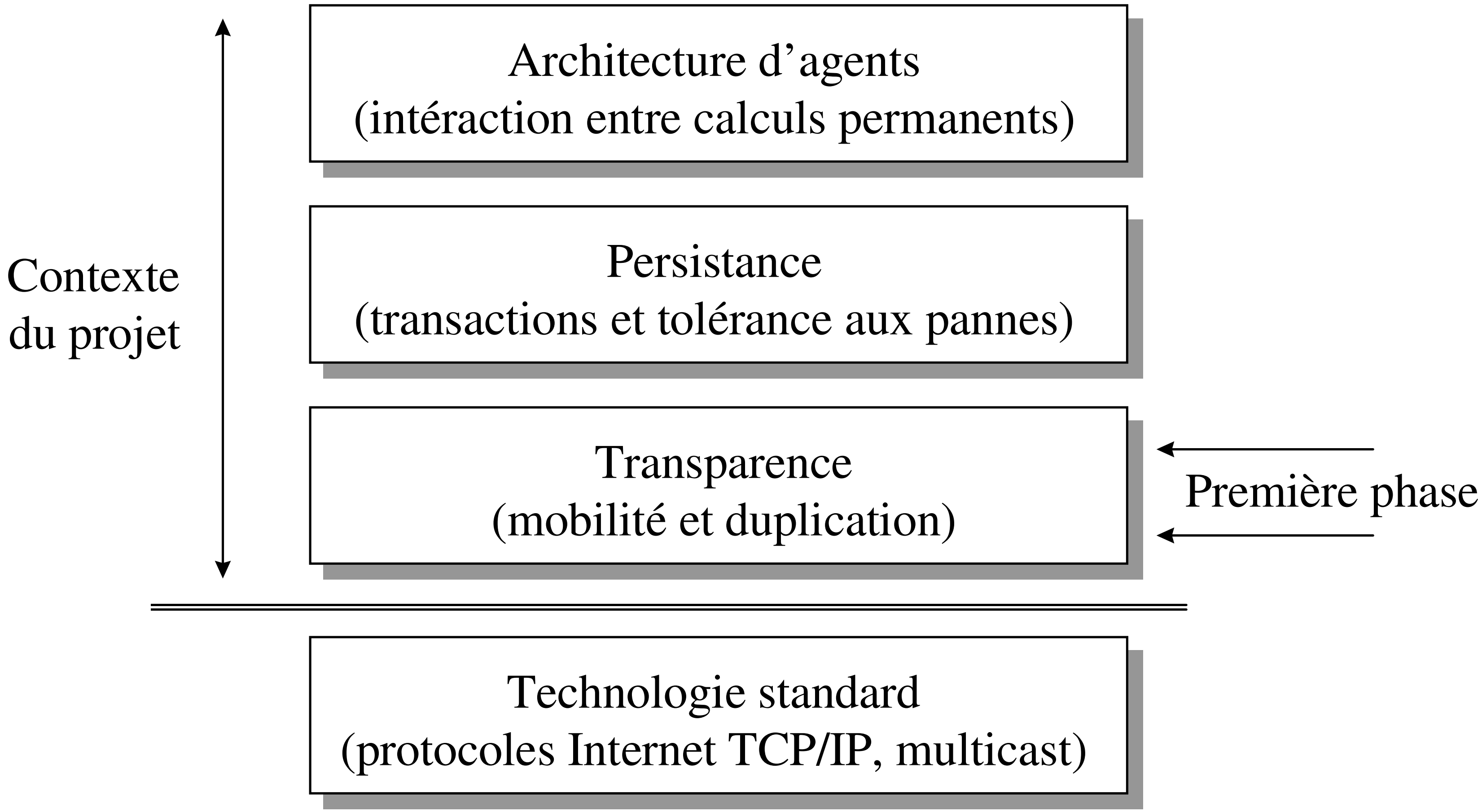
- Abstraction
  - Principe général d’organisation de systèmes complexes
  - Abstraction du réseau: répartition *transparente*
- Raisonnement formel
  - Logique temporelle
  - Propriétés de «safety» et «liveness»
- Construction de système
  - Petit nombre d’utilisateurs exigeants
  - Feed-back entre développeurs et utilisateurs

Abstraction du réseau

- Les langages centralisés (genre C++) ne conviennent pas à la programmation répartie
- Idée fondamentale: abstraction du réseau
  - *Transparence*: Les calculs obtiennent le même résultat sur un site que sur plusieurs sites.
  - *Conscience du réseau*: Le contrôle des communications est basé sur la *mobilité*
- La transparence facilite la répartition et la conscience du réseau la rend efficace

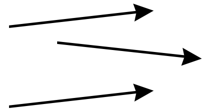


Couches d'abstraction

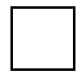


Pour permettre une répartition transparente, un langage doit:

- Etre concurrent

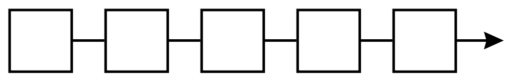


  - L'expression d'activités qui exécutent en même temps
  - C'est tout autre chose que le parallélisme!
- Etre conscient de l'état (valeurs et objets)
  - Distinction fondamentale entre valeurs qui ne changent pas (elles peuvent être copiées) et objets qui sont évolutifs (ils doivent rester localisés)



valeur

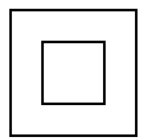
  - Les objets sont *mobiles* ou *stationnaires*



t

objet

- Cloisonner les références
  - Permet une sécurité totale au niveau du langage



La mobilité

- Notre but est l’abstraction du réseau, avec le maintien d’un contrôle à la fois *prévisible* et *complet* des communications
- Pour atteindre ce but, il est important que la *mobilité* soit impliqué dès le début
- Actuellement, tous les langages et systèmes répartis sont basés sur des objets qui ne changent pas de site
- Mobilité = possibilité d’un objet de rester sur un site ou de changer de site

Le projet:
Distributed Oz

- Un système pour faciliter la programmation répartie
  - Solides bases théoriques
  - Utilisation pratique dans les réseaux haute-performance
- Collaboration avec:
  - Swedish Institute of Computer Science, Stockholm
  - Université de Sarrebruck, Allemagne
- Possibilités de collaboration avec le CERTI

Conclusions

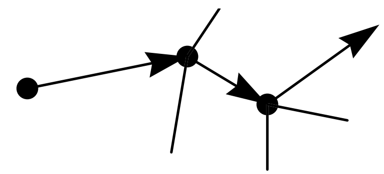
- Un système réparti est un ensemble d'ordinateurs lié par un réseau. Dans l'avenir, l'ordinateur seul disparaîtra, pour être remplacé par un réseau de réseaux.
- Un réseau est actuellement *très difficile* à programmer. Maintes débâcles industrielles en sont une bonne illustration.
- Le but du projet Distributed Oz à INGI est de résoudre ce problème du point de vue théorique et pratique

Raisonner avec le modèle

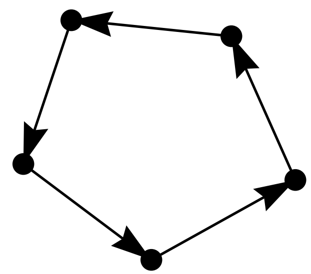
- Définir un algorithme comme un système de transitions
- Prouver des propriétés avec une logique temporelle d’événements
- Possibilités:
  - Systèmes synchrones / asynchrones
  - Messages en ordre / sans ordre
  - Temps de transmission borné / non-borné
- Propriétés importantes:
  - Safety: «Les mauvaises choses n’arrivent jamais»
  - Liveness: «Les bonnes choses arriveront»

Quelques algorithmes importants

- Protocoles (en rapport avec la transmission)
  - Fiabilité point-point (corriger les erreurs de transmission)
  - Routage (trouver le chemin)
  - Eviter deadlock (blocages mi-chemin)



- Algorithmes de base
  - Ondes (événement qui dépend de tous les processus)
  - Election (choisir un processus)
  - Détection de terminaison
  - Photo (consistante avec vue globale)



- Tolérance aux pannes
  - Un processus qui tombe en panne
  - Un processus avec un comportement «Byzantin»