Algorithmique distribuée
INFO4

Karine Altisen

Résumé

Algorithmique distribuée = Algorithmique des réseaux

Différences principales avec l'algorithmique classique :

- ***Calcul réparti :*** Plusieurs entités de calcul autonome(processus)
- ***Données réparties :*** Chaque entité n'a qu'une vue partielle du système global
- ***Echange d'informations :*** Communication asynchrone par message
- ***Absence de temps global :*** Par exemple, pas d'horloge commune

INFO4 : Algorithmique distribuée **non-tolérante aux fautes**

~~**INFO5 :** Algorithmique distribuée **tolérante aux fautes**~~

➔ **une introduction en fin de module sur la tolerance aux fautes**

Objectifs

Comprendre les limitations fondamentales au développement de systèmes distribués

Connaître quelques problèmes et solutions typiques

Savoir écrire des algorithmes distribués

- Etre capable de spécifier formellement un problème
- A partir d'un ensemble d'hypothèses sur le système (topologie, pannes, …) être capable d'écrire un algorithme distribué réalisant la spécification.

Savoir prouver et analyser la complexité d'algorithmes distribués

Savoir déployer sur un simulateur les algorithmes vus en cours et TD

- Utilisation du simulateur événementiel SINALGO.

Plan

Chapitre 1 : Introduction

- Exclusion mutuelle / anneau

Chapitre 2 : Election / anneau

Chapitre 3 : Causalité et horloge de Lamport

- Exclusion mutuelle / graphe complet

Chapitre 4 : Algorithmes à vagues

- Circulation de jeton / arbre, graphe quelconque

Chapitre 5 : Etat global (snapshot) / introduction à la tolérance aux fautes

- Graphe quelconque

Modalités

10 cours magistraux – Karine.Altisen@univ-grenoble-alpes.fr

10 travaux dirigés – Vania.Marangozova@univ-grenoble-alpes.fr

Evaluation

- Devoir maison 15/2 → 14/3
- Devoir sur table 22/2 pendant le TD5
- Examen (2 heures, tous documents autorisés)
- Note = (3Exam + DS + DM)/5

Site web :

http://www-verimag.imag.fr/~altisen/APD/

Chapitre 1
Algorithmique distribuée
Un exemple introductif

Système distribué = ?

Algorithmes distribués = algorithmes pour les systèmes distribués

« Un système distribué (ou système réparti) est un ensemble d'unités de traitement autonomes et interconnectées entre elles »

Gerard Tel, *Introduction to Distributed Algorithms*

Système distribué =

« Un système distribué (ou système réparti) est un ensemble d'unités de traitement autonomes et interconnectées entre elles »

Ensemble d'unités → **Modélisation d'un RESEAU**

- Internet
- Réseau téléphonique filaire
- GPS
- Réseau de capteurs
- Threads sur une machine

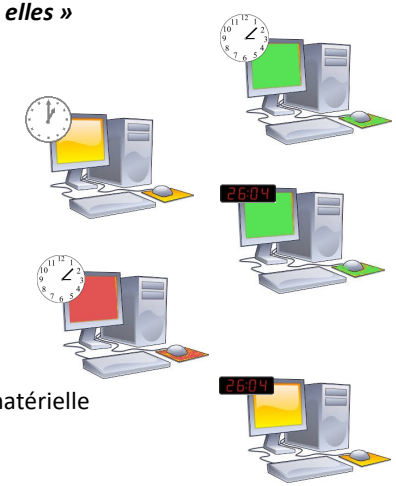
Unités de traitement = processus / processeurs / nœuds du réseau

Système distribué =

« Un système distribué (ou système réparti) est un ensemble d'unités de traitement autonomes et interconnectées entre elles »

Autonomes =

- Programmes locaux
- Mémoires locales
- Asynchrones – Pas de temps global



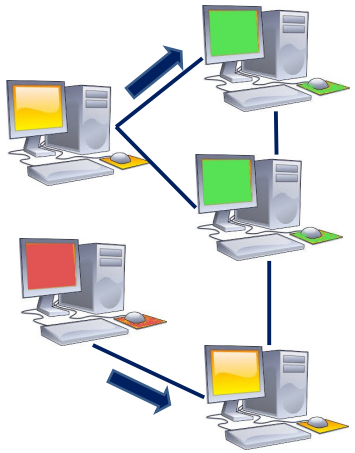
algorithmique parallèle :
→ pas de synchronisation logicielle ou matérielle

Système distribué =

« Un système distribué (ou système réparti) est un ensemble d'unités de traitement autonomes et interconnectées entre elles »

Interconnexion =

Echange d'informations
Par envoi de messages



Exemple :

connexion filaire (câble RJ45, fibre…)
connexion par onde (Wifi...)

Modèle OSI

Algorithmique Distribuée (protocoles réseau)

Application	*données*
Présentation	*données*
Session	*données*
Transport	*segments*
Réseau	*paquets*
Liaison des données	*trames*
Physique	*bits*

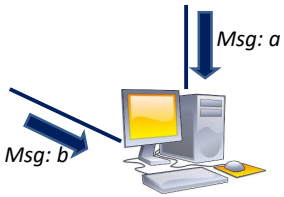
Centralisé *vs* Distribué

Absence de connaissance globale

Un processus calcule en fonction de sa mémoire locale

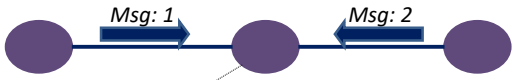
Absence de temps global

temps d'exécution propre
communications asynchrones
dérive éventuelle des horloges locales



Non déterminisme

Toute entrée étant fixée, une exécution du système peut aboutir à des états différents



Code : prendre la couleur rouge
si la première valeur reçue vaut 1, verte sinon

Caractéristiques d'un système distribué : Topologie

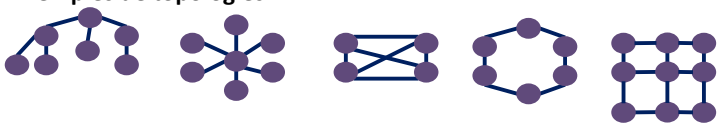
Topologie = le réseau d'interconnexion

Graphe orienté – Voisinage d'un nœud

Bidirectionnel ?

Connexe ? (il existe un chemin entre toute paire de nœuds)

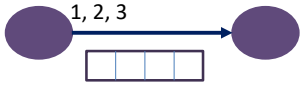
Exemples de topologies



Dans ce cours

Topologie connexe
Unidirectionnelle ou bidirectionnelle
Anneau, arbre, graphe complet, graphe quelconque

Caractéristiques d'un système distribué : Liens de communication



Abstraction du canal de communication

Communication par message

- **Fiabilité ?** (fiable = pas de perte, duplication ou création)
- **Ordre d'arrivée ? – FIFO ?**
- **Capacité ?** (bornée ou non, de taille connue ou non)
- **Temps d'acheminement**
  - Synchrone ou asynchrone,
  - Temps maximal d'acheminement connu ou non

Dans ce cours

- **Canaux de communication fiables**
- **Asynchrones**
- **Capacités finies mais non bornées**
- **Ordre d'arrivée FIFO ou non**

Caractéristiques d'un système distribué : Processus

Sont ils sujets aux pannes ?

Sont ils tous les mêmes ?

→ Ont-ils tous le même algorithme ?

→ Des algorithmes différents ?

Processus **initiateurs** : démarrage spontané

Processus non-initiateurs : démarrage suite à une communication

Taille de la mémoire locale : bornée ou non

Dans ce cours

- **Pas de panne**
- **Taille de mémoire bornée (à vérifier pour chaque algorithme)**
- **Avec initiateur ou non, selon les cas**

Caractéristiques d'un système distribué : Connaissance des processus

Hypothèse supplémentaire :

Les processus ont-ils connaissance d'informations globales ?

→ ***Connaissance sur la topologie***

Forme, nombre de nœuds, degré maximal …

→ ***Distinction entre les processus***

Processus munis d'un identifiant unique qui peut être utilisé par l'algorithme

Processus anonymes

Distinction partielle des nœuds : réseau enraciné

→ ***Propriété locale sur le voisinage***

Degré, identité des voisins,

étiquetage local des canaux entrant et sortant…

Exemples au tableau

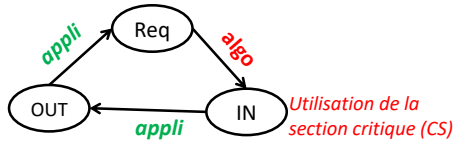
Un exemple introductif :

Exclusion mutuelle sur anneau orienté

Algorithme de Le Lann

Problème d'exclusion mutuelle - Spécification

Schéma de principe
Pour UN processus



Spécification

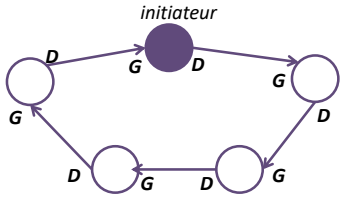
Sûreté : ***« Rien de mal ne peut jamais arriver »***

→ *Il y a toujours au plus un processus en CS*

Vivacité : ***« Quelque chose de bien finit par arriver »***

→ Tout processus demandeur (Req) finit par entrer en CS
(= en temps fini)

Hypothèses pour la solution



Topologie :

Anneau unidirectionnel, avec orientation consistante

Au moins 2 processus

Un initiateur unique

Processus :

Pas de faute

Processus et canaux asynchrones

L'utilisation de la CS dure un temps fini

Algorithme de Le Lann

→ *Voir algorithme*

→ *Simulation dans SINALGO*

Sémantique du modèle d'exécution

p exécute : **Envoyer <*M*> à *S***

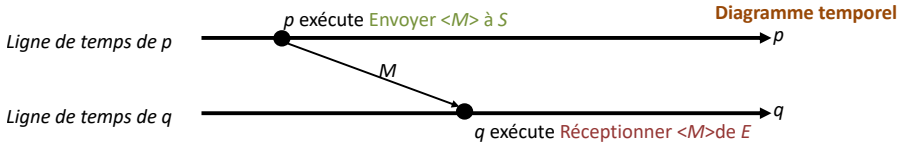
après exécution de cette instruction, le canal *E* de *q* contient *M*

q exécute : **Réceptionner <*M*> de E**

instruction bloquante jusqu'à exécution

pour éviter le blocage : **Si Réceptionner <*M*> de E alors …**

filtre possible sur le contenu, par exemple ***<Type, Tag, M>***



Algorithme de Le Lann : preuve de correction

Preuve de sûreté

Lemme (Sûreté) : jamais plus d'un processus n'est en CS.

Preuve au tableau

Preuve de vivacité

Lemme (Vivacité) : tout demandeur acquiert la ressource en temps fini.

Preuve au tableau

Correction vis-à-vis de la spécification

Théorème : l'algorithme résout le problème d'exclusion mutuelle dans un anneau unidirectionnel.

Toutes les hypothèses sont utilisées dans la preuve !!

Algorithme de Le Lann : complexité

On compte : le nombre de messages envoyés

On note : *n le nombre de nœuds dans l'anneau*

***Nb** : le jeton fait un tour de l'anneau à l'aide de ? n messages*

***Temps de service** = nombre de processus pouvant exécuter la CS avant qu'un processus ne l'obtienne*

→ ***meilleur cas** = ? 0*

→ ***pire cas** = ? n - 1*

Ratio nombre de messages / nombre de demandes

→ ***meilleur cas** = ? 1*

→ ***pire cas** = ? ∞*

Solution « proactive » ≠ Solution « réactive », algorithmes à permission