Algorithmique répartie & Simulation

Olivier FLAUZAC
olivier.flauzac@univ-reims.fr
http://www.flauzac.eu

Les systèmes répartis

Définition

- Opposition avec
  - les systèmes centralisés
  - les systèmes à flux d'exécution unique
- Système distribué :
  - ensemble de flux d'exécutions connectés entre eux par un réseau
  - vu par l'utilisateur comme une entité unique

Exemple de système répartis

- Réseau informatique
  - réseau filaire
  - réseau sans fil (Wifi, ad hoc)
- Cluster
- Machine parallèle
- Machine multi processeur / multi coeur
  - serveurs / ordinateurs de bureaux
  - smartphones / tablettes
  - les systèmes répartis nous entourent !!!

Précision de la définition

- Système distribué :
  - ensemble de flux d'exécutions connectés entre eux par un réseau
  - vu par l'utilisateur comme une entité unique
  - communication par échange de messages

Problématiques

- Structurer les système
  - organiser les éléments du système
- Acheminer l'information
  - connaître les chemins qui relient les éléments
  - faire transiter l'information
    - assurer de la garantie de service

Problématiques (fin)

- Synchroniser les exécutions
  - plus de scheduler global
  - gestion / attente des échanges
- Gestion des erreurs
  - environnements d'exécution différents

Algorithmique répartie

- Ensemble de règles qui spécifient le comportement d'un système
- Exécutions simultanées de code dans des noeuds
- Pas de partage de données entre les noeuds
- Juste un échange d'informations
- Pas d'horloge centralisée

Première approche de modélisation

- Modélisation du système
  - noeuds
  - liens de communications
- Modélisation des algorithmes
  - données en chaque site, contenu des messages, exécutions
  - gestion et impact des pannes
- Modélisation des exécutions

Topologie

Notion de topologie

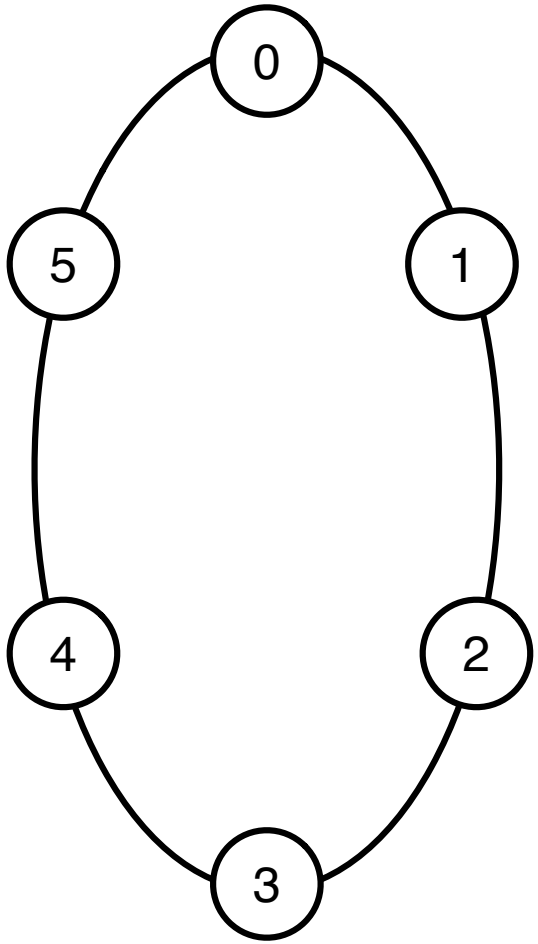
- Représentation du système sous la forme d'un graphe
- G=(V,E)
  - V : ensemble de noeuds
  - E : ensemble de liens
- Topologie : architecture d'implantation et de communication des noeuds
- Impact sur les méthodes et les performances des algorithmes
  - gain de performances
  - simplification des algorithmes

Chaîne

- Suite de noeuds connectés
- Deux types de noeuds :
  - extrémités (1 seul voisin)
  - interne (2 voisins)
- Communication simple
- n noeuds et (n - 1) liens de communication

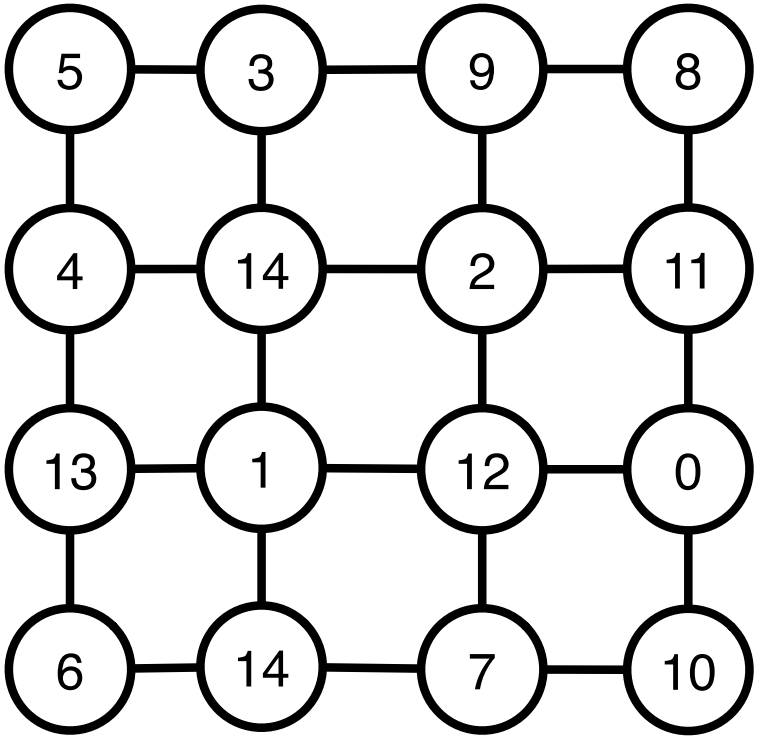
Anneau

- Chaîne dont les extrémités sont reliées
- Pas de noeud distingué
- Chaque noeud a 2 voisins
- Orientation possible
- n noeuds , n liens de communication



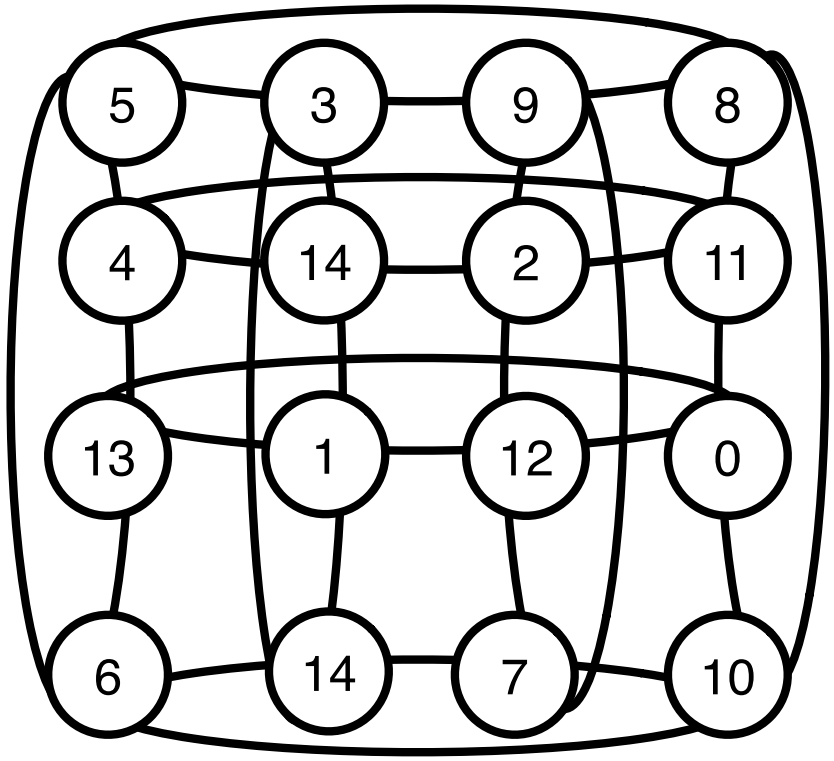
Grille

- Topologie régulières
- 3 types de noeuds
  - coins (2 voisins)
  - bords (3 voisins)
  - interne (4 voisins)
- Repérage des coordonnées



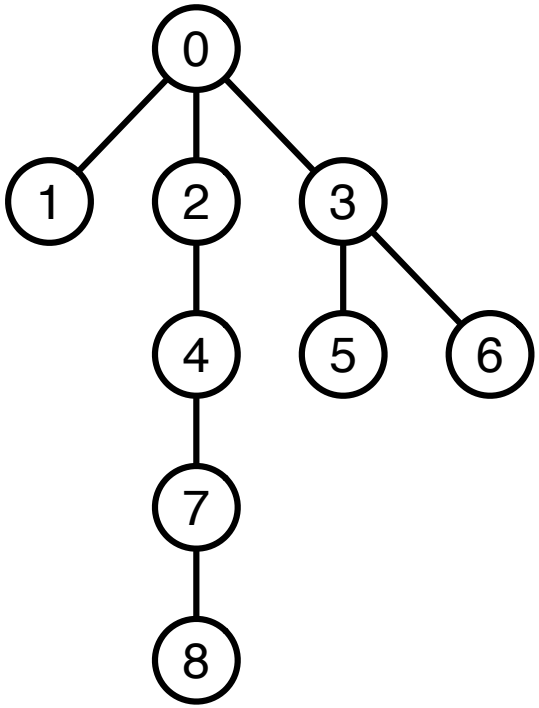
Tore (grille torique)

- Topologie régulière
- Extension de la grille
- Chaque noeud a 4 voisins
- Repérage des coordonnées



Arbre

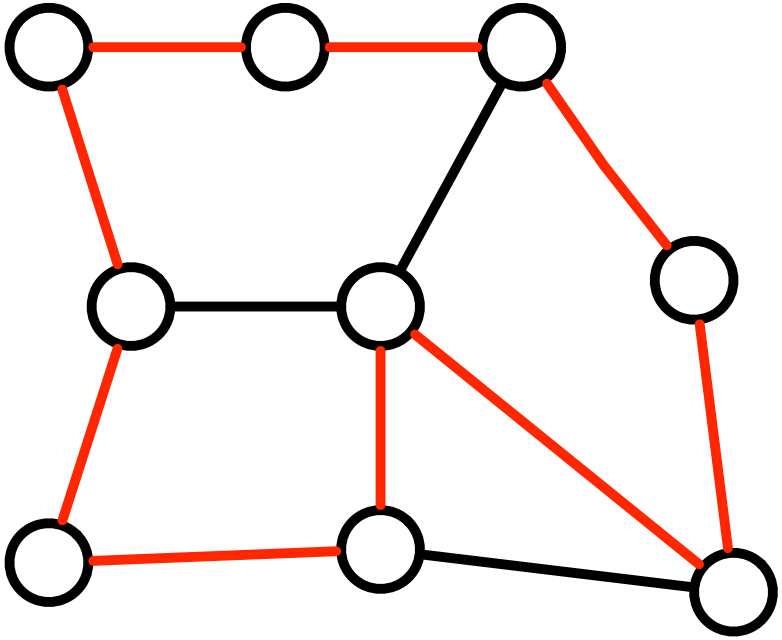
- Graphe de n noeuds
  - connexe
  - n-1 arrêtes
- Hiérarchisation implicite

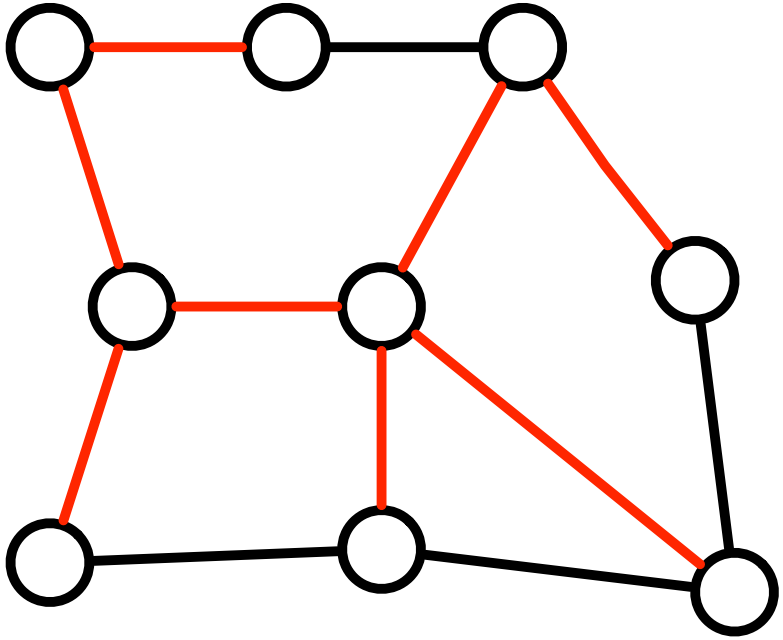


Topologie physique / topologie logique

- Topologie physique
  - implantation réelle des noeuds
- Topologie logique
  - organisation spécifique logique des noeuds
- Placement dans un cadre particulier
- Bénéficier des «facilités» d'une topologie

Plongement de topologie





III. Composants des systèmes répartis

Eléments d'un système

- Modélisation sous la forme d'un graphe
- Composants principaux
  - noeuds (sites, processeurs ...)
  - canaux de communication

Les noeuds

- Unité communicantes de calcul
- Gestion de l'émission et de la réception des messages
  - interfaces et files de communication
- Gestion des calculs
- Stockage en interne
- Identification possible
- Gestion et connaissance du voisinage

Les liens de communication

- Moyens d'échange
- Assure la liaison entre les noeuds
- Toutes propriétés définissables
  - ordre des échanges : FIFO, non FIFO
  - taille des canaux
  - propriété de transfert : latence, vitesse
  - types de communication : point à point, diffusion

IV. Exécution

Système de transition

- Modélisation de l'exécution
- Permet l'analyse et la preuve des algorithmes
- Etude des impacts des opérations réalisées
- Définitions des étapes successives d'une exécution

Système de transition

- Ensemble S de
  - tous les états possibles du système (C)
  - les transitions possibles ($\rightarrow$)
  - sous-ensemble des états de départ (I)
- S=(C,$\rightarrow$,I)

Etat / configuration

- Etat :
  - chaque site repéré par son état
  - union de l'ensemble des états des éléments du site
- Configuration
  - ensemble des états des sites à un instant donné
  - système de transition : suite des configurations

Evénement

- Nécessité de provoquer les transitions
- A chaque transition est associé un événement
- Les événements dirigent le système
- Une exécution définit une séquence d'événement
- Quelle règle pour l'enchaînement des événements ?

VI. Des algorithmes répartis

Ecriture des algorithmes

- Algorithmes écrits sous la forme de gardes
- Garde : événement + condition
    - *A la réception d'un message de k*
- Algorithme dirigé par les événements
- Contenu des algorithmes
    - opérations internes
    - gestion des communications (send / receive)
    - horloge de garde (timeout)

Emission d'un message

- Primitive d'émission : send
- Permet d'émettre selon le modèle
- Emission simple : un seul voisin
  - `send(id,msg)`
- Diffusion : tous les voisins
  - `send(Neigh,msg)`

Réception d'un message

- Primitive de réception
- Associé à un événement
  - évolution du système de transitions
  - déclenche les exécutions
- Type de réception
  - aveugle
  - depuis une interface / un voisin précis
  - réception typée

Timeout

- Evénement interne
- Comparable à un «auto-message»
- Permet l'évolution d'un algorithme si aucun message
  - évite le communication deadlock

Initialisation

- Première phase des algorithmes
- Permet de définir l'état de chaque noeud à l'origine
- Permet d'initier les premier envois
- Garde de démarrage
- Pas toujours obligatoire !

Circulation de jeton sur un anneau

- Modèle :
  - canaux de communication FIFO
  - organisation en anneau orienté
  - chaque noeud détient 2 voisins (next,prev)
  - communication point à point
  - noeuds identifiés de manière unique

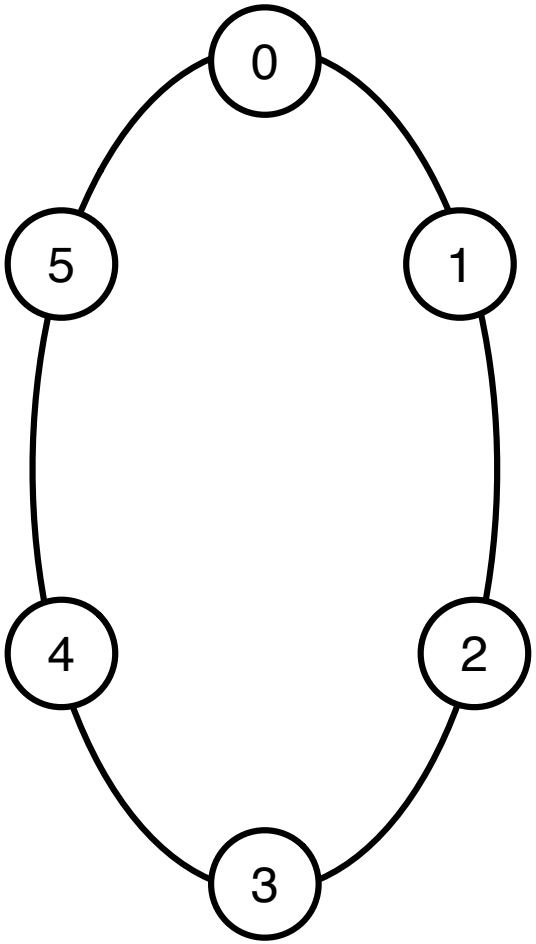
Algorithme 1

- Init

```
Si (id == 0){
  send(next,Message)
}
```

- A la réception d'un message

```
send(next,Message)
```



Algorithme 2

- Init

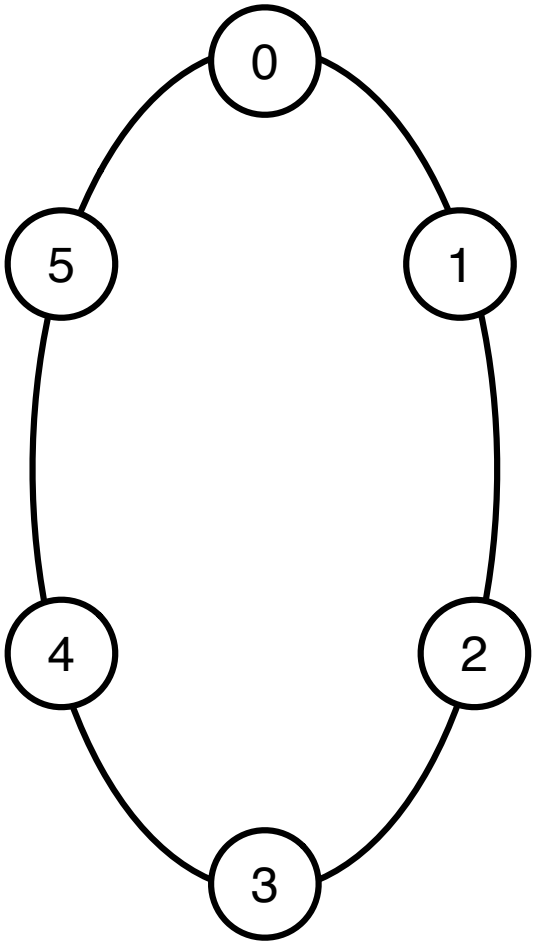
```
Si (id == 0){
  send(next, Message)
  send(prev, Message)
}
```

- A la réception d'un message de prev

```
send(next,Message)
```

- A la réception d'un message de next

```
send(prev,Message)
```



Vers la simulation

Expérimentation

- In Vivo
  - exécution en environnement réel
- In Vitro
  - exécution réelle en environnement confiné
- In Silico
  - exécution simulé

Simulation

- Reproduction par des modèles numériques
- Exécution reproductible
- Types de simulation
  - simulation en temps continu
    - systèmes d'équations, équations différentielle
  - simulation à événements discrets

Simulation à événement discrets

- Mise en place d'un système de transition
- Evolution du système selon les événements
  - émission / réception de messages
  - timeouts
- Basé sur la règle de causalité
  - la réception d'un message est postérieure à son émission

Evénement

- Signature
  - Type
  - Position d'occurence
  - Date d'occurence
- Traitement
  - génération d'un nouvel événement
    - émission génère une réception
  - règles de traitement

Simulation dirigé par le temps

- Mise en place d'une horloge supérieure
- Enregistrement des événements en fonction de leur date
- Principe
  - le simulateur déroule le temps et exécute les événements enregistrés
  - pour chaque événement
    - calcul des opérations associées
    - création des événements déduits

Simulation dirigé par les événements

- Gestion de liste d'événements
- Insertion des événements selon leur date
- Exécution des codes associés à chaque événement

Structure d'un simulateur

- Gestionnaire d'événements
- Hiérarchie de *callbacks* qui permettent l'association événement / code à exécuter
- Modèles spécifiques pour
  - les communications
  - les noeuds
- API étendue
  - nombre aléatoires, topologie …

Quelques simulateurs

- NS2, NS3
- Omnet++
- Simgrid