Election de leader

Election de leader

Définition : Étant donné un ensemble de processus choisir un unique chef (sûreté) en un temps fini (vivacité)

Applications :

- réseaux radio et wifi
- réseaux filaires
  - Construction d'arbre couvrant : le chef devient la racine de l'arbre et peut initier cette construction par diffusion
- cloud ou systèmes « maître - esclave »
  - en cas de la défaillance du maître, élire un nouveau maître

Election de leader

Théorème Il n'existe pas d'algorithme déterministe d'élection de chef dans les réseaux anonymes et uniformes.

Idée de la preuve :

1. Le réseau est anonyme donc la configuration de départ peut être symétrique
2. La configuration objectif (celle où un leader est élu) est une configuration asymétrique
3. Il existe une exécution du système telle qu'à partir d'une configuration symétrique on passe toujours dans une configuration symétrique

Election de leader

- Contourner les résultats d'impossibilité :
  - utilisation des identifiants
    - Chang et Roberts
    - Hirshberg-Sinclair
    - Diffusion
  - utilisation des probabilistes
    - Itai et Rodeh

Algorithme Chang-Roberts

Algorithme Chang-Roberts

- **Topologie :** anneau unidirectionnel (chaque site «i» dispose d'un pointeur vers son successeur succ[i])

- **Idée :** chaque candidat diffuse autour de l'anneau sa candidature ; le processus ayant l'identifiant max gagne

Algorithme Chang-Roberts

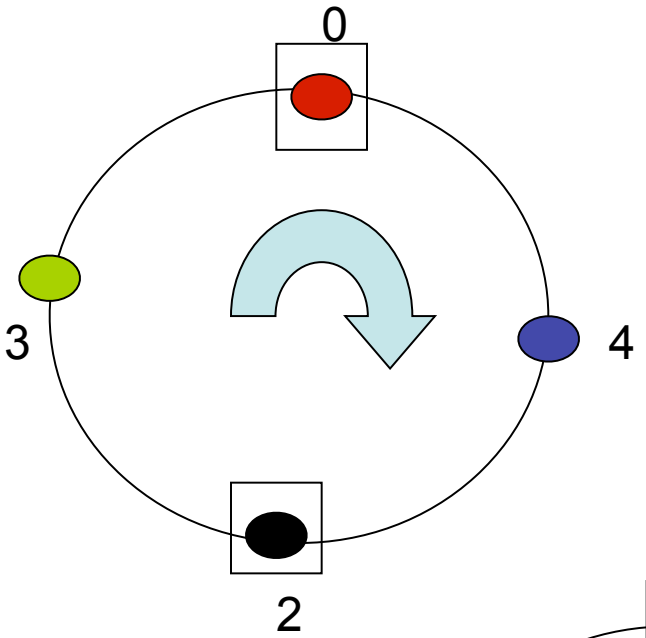
Candidature site «i»

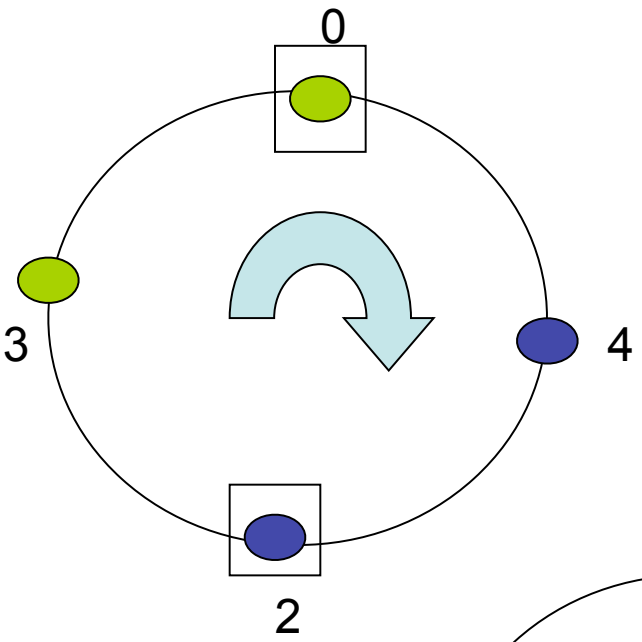
```
candidat_i = vrai
envoyer(CHEF,i) à succ[i] /* i diffuse sa candidature
```

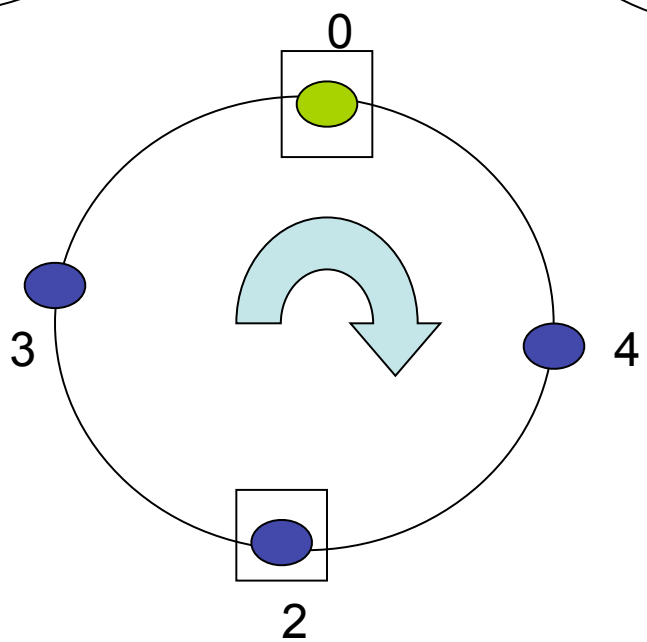
Réception sur site «i» du message (CHEF,j) depuis site «j»

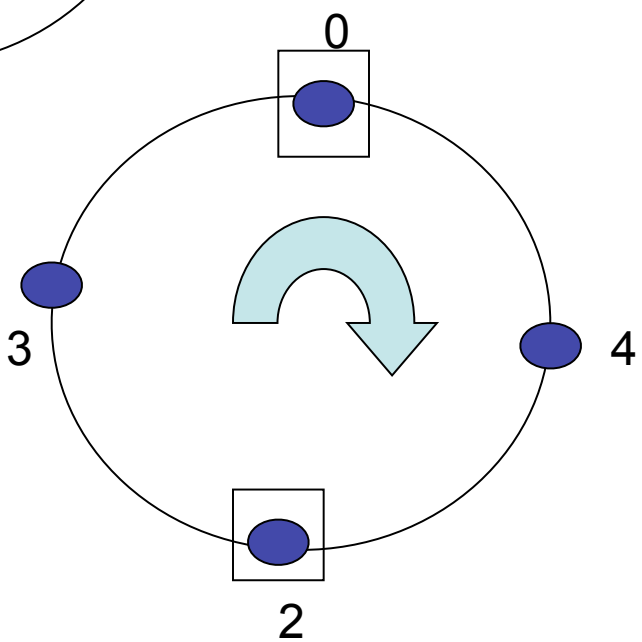
```
Case
   j>i: envoyer (CHEF,j) à succ[i]
   j<i: if not candidat_i
            candidat_i=vrai; envoyer(CHEF,i) à succ[i]
   j=i: diffusion(CHEF, i) /* envoyer à tous le résultat de
   l'élection
```

Algorithme Chang-Roberts



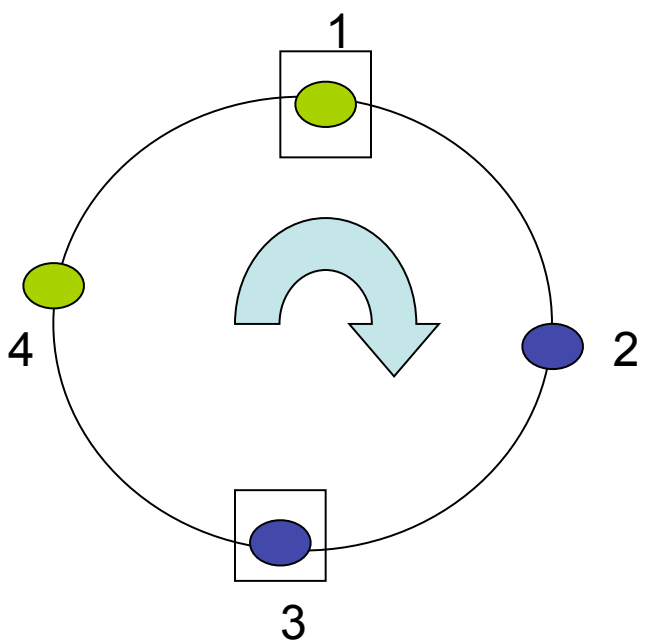






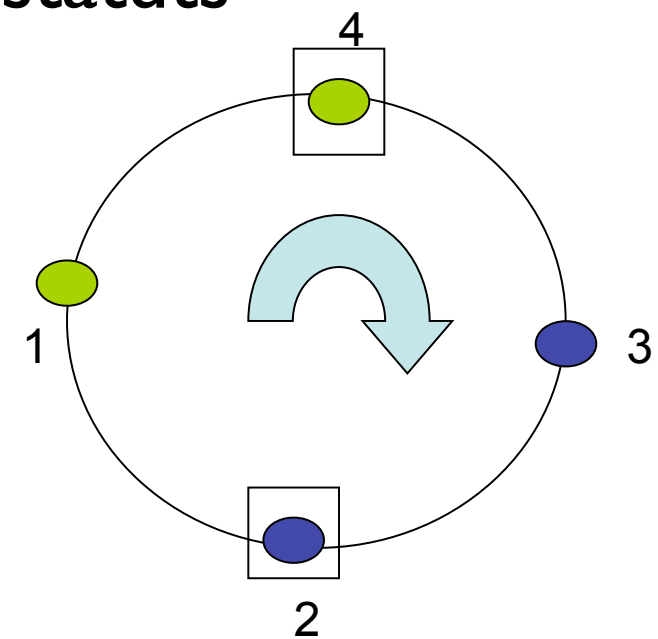
Complexité
Meilleur cas : O(n)

- Les identifiants sont ordonnés dans l'ordre croissant autour de l'anneau



Complexité
Pire cas : $O(n^2)$

- Les identifiants sont ordonnés dans l'ordre décroissant autour de l'anneau
- L'identifiant « i » visite i noeuds avant de décider son statuts



Complexité
Moyenne : O(nlog(n))

- Répertorier toutes les possibilités d'arranger les identifiants autour de l'anneau : (n-1)!
- Variable aléatoire X_k: Nombre de messages si l'élection était partie du noeud k
  - E[∑X_k] = ∑ E[X_k] pour k de 1 à n

Algorithme «Diffusion»

Algorithme «Diffusion I»

- **Idée :**
  - chaque candidat envoie son identité aux autres nœuds du réseau
  - un site répond à ceux de numéro inférieur au sien
  - un processus qui ne reçoit pas de réponse est le chef

- **Hypothèse :** communication fiable et synchrone (borne connue sur le temps de communication)

Algorithme «Diffusion 2»

- **Idée :**
  - chaque candidat envoie son identité aux autres nœuds du réseau et attend les identités des autres sites
  - Calcul du max/min sur l'ensemble d'identités reçues

- **Hypothèse :** communication fiable et connaissance du nombre de processus dans le réseau

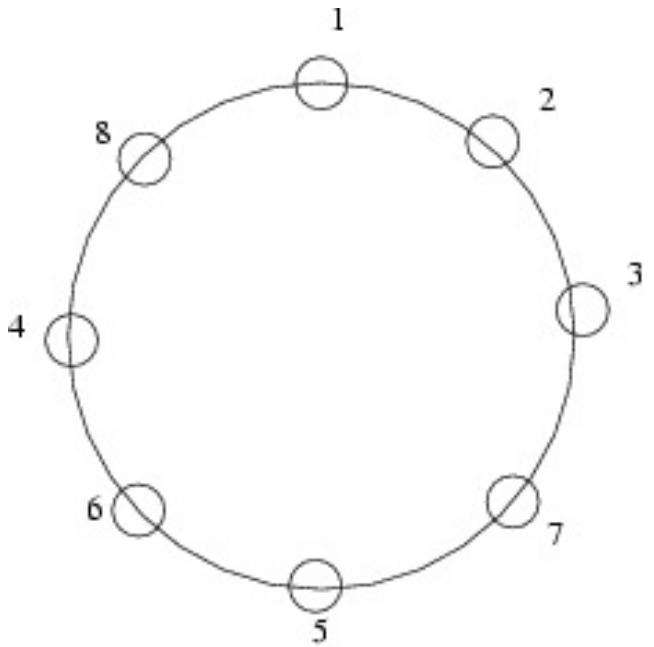
Algorithme Hirschberg-Sinclair

Hirschberg-Sinclair

- Topologie : anneau bidirectionnel
- L'algorithme travaille en rounds
- Uniquement les processus qui gagnent l'élection du round *r participent au round r + 1*
- Algorithme: P_i est le leader dans le round *r* ssi P_i est l'identifiant maximal dans l'ensemble de noeuds à distance au plus 2^r de P_i

Hirschberg-Sinclair

- Initialement:
  - Tous les processus sont chefs
- Round 0:
  - 6 , 7 et 8 sont chefs
- Round 1:
  - 7, 8 sont chefs
- Round 2:
  - 8 est le seul chef
  - au plus log(N) rounds



(b)

Algorithme Itai-Rodeh

Algorithme Itai-Rodeh

- Basé sur l'algorithme de Chang et Roberts
- Chaque processus choisit aléatoirement un identifiant dans l'ensemble 1..n (deux processus peuvent choisir le même identifiant)
- Chaque processus candidat envoie un jeton avec deux champs :
  - "counter" initialisé à 1
  - "another" initialisé à faux (dès que le jeton rencontre un candidat avec le même identifiant, "another" passe à vrai)
- Les leaders de la round "i" recommencent l'algorithme
- L'algorithme se termine avec probabilité 1

Election de leader Application

Construction d'un arbre couvrant (diffusion)

- Un unique arbre couvrant est construit (sûreté) en un temps fini (vivacité)
- Utilisation d'un chef
  - Le chef commence la construction de l'arbre couvrant en envoyant un message spécifique M à ses voisins; un processus qui reçoit le message M prend comme père l'expéditeur du message et le diffuse à son tour

Construction d'un arbre couvrant (diffusion)

- Structures :
  - Parent : pointer vers le père du nœud dans l'arbre couvrant (initialement NULL)
  - Children : ensemble des fils (initialement vide)
  - Others : les voisins qui ne sont pas de fils (initialement vide)

Construction d'un arbre couvrant (diffusion)

Site i ne reçoit pas de message

```
if chef and parent=NULL then
    envoyer M aux voisins; parent=i
```

Site i reçoit message M depuis site j

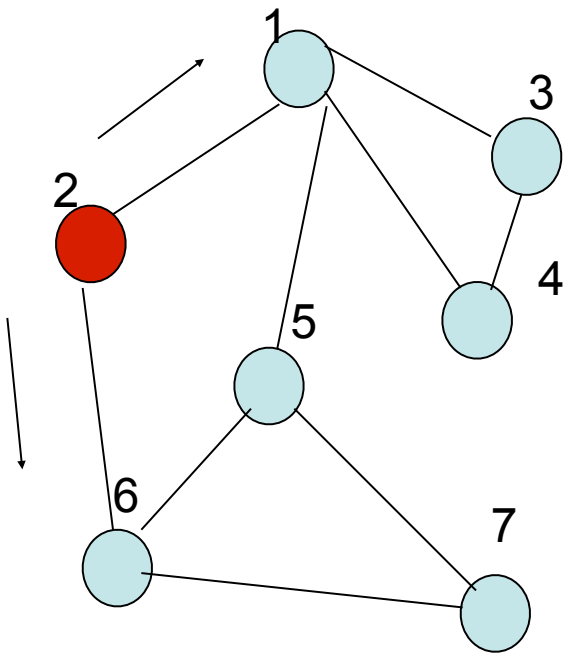
```
    if parent=NULL then
      parent=j
      envoyer <parent> à j
      envoyer M aux voisins k, tels que k≠j
    else envoyer <rejet> to j
```

Site i reçoit <parent> depuis site j

```
    children=children U {j}
    if children U others=voisins \ {parent} then
      fin
```

Site i reçoit <rejet> depuis site j

```
    others=others U {j}
     if children U others=voisins \ {parent} then
      fin
```



Construction d'un arbre couvrant (diffusion)

Site i ne reçoit pas de message

```
if chef and parent=NULL then
    envoyer M aux voisins; parent=i
```

Site i reçoit message M depuis site j

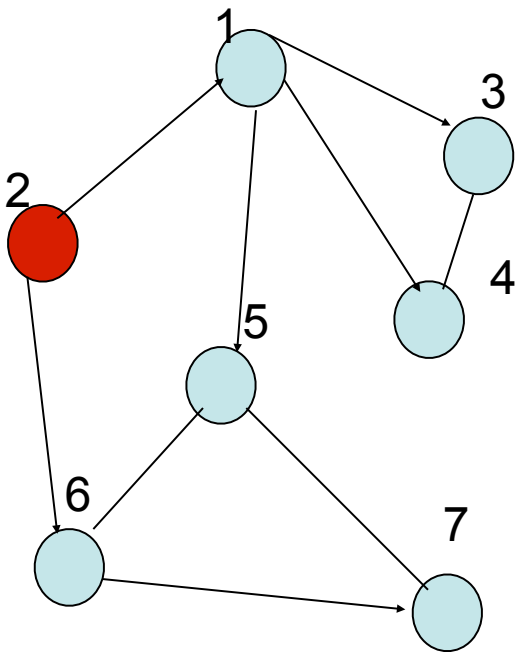
```
if parent=NULL then
  parent=j
  envoyer <parent> à j
  envoyer M aux voisins k, tels que k≠j
else envoyer <rejet> to j
```

Site i reçoit <parent> depuis site j

```
children=children U {j}
if children U others=voisins \ {parent} then
  fin
```

Site i reçoit <rejet> depuis site j

```
others=others U {j}
 if children U others=voisins \ {parent} then
  fin
```



Others(6)={5}
Others(5)={7,6}
Others(3)={4}
Others(4)={3}
Others(7)={5}

Applications

- Calcul de la taille d'un réseau
- Mettre en place d'un système de type publish/subscribe
- Implémenter l'allocation de ressources en exclusion mutuelle
- Accès aux données répliquées
- Implémenter le consensus
- Détecter la terminaison d'un algorithme
- Sortir des situations de blocage

Conclusion

- Réseaux de robots
  - Solutions probabilistes
- Réseaux de capteurs, wifi, radio
  - Mise en place des algorithmes locaux probabilistes
- Réseaux P2P
  - Difficile de choisir un leader car il peut à tout moment quitter le système (ici des solutions alternatives s'imposent)