Élection

Chapitre 3

Observations

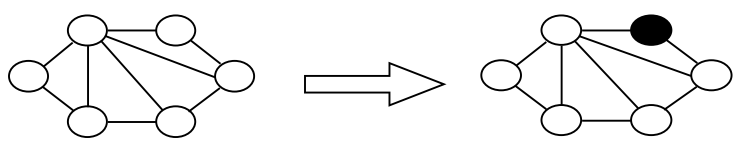
Élection dans l'anneau

Élection dans la grille

Élection dans l' Hypercube

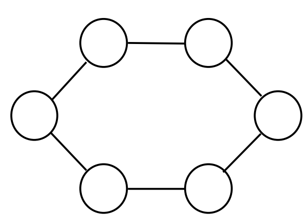
Élection dans les graphes arbitraires

Élection



Théorème *[Angluin 80]*

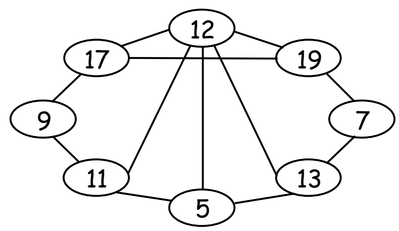
Le problème de l'élection ne peut pas être résolu si les entités n'ont pas des identificateurs uniques.

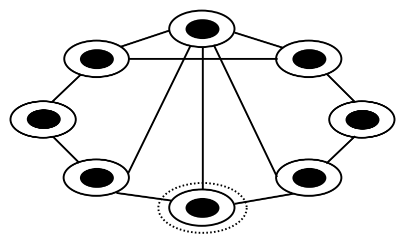


•identificateurs uniques
•Même état
•Anonyme
•Synchrone

À chaque instant, elles font la même chose.

Note: Avec identificateurs distincts, la **recherche du minimum** est une élection





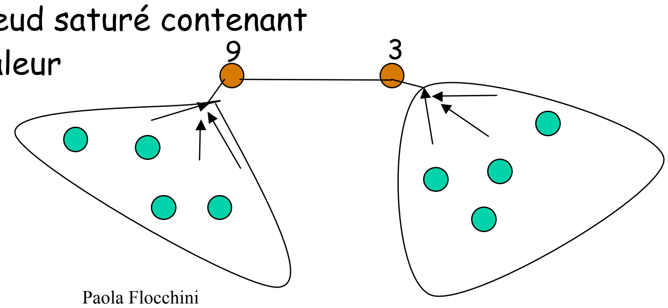
Paola Flocchini

Élection dans l'Arbre

Un identificateur distinct v(x) est associé à tout noeud x

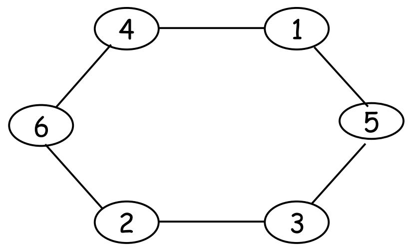
Un algorithme simple:

1) Exécuter l'agorithme de saturation,
2) Choisir le noeud saturé contenant la plus petite valeur



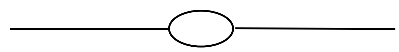
Paola Flocchini

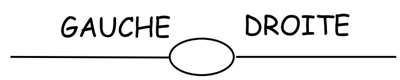
Anneau



- n entités
- m liens

- # entités = # liens
- Symétrique
- Chaque entité a deux voisins





Sens de l'orientation

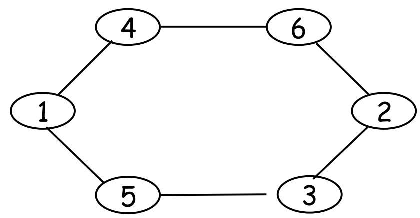
Algorithmes d'Élection dans l'Anneau

- All the way
- As Far
- Controlled distance
- Electoral stages
  --- bidirectional version
- Alternating steps

ÉLECTION DU MINIMUM

All the way

Idée générale: Chaque entité voit tous les identificateurs.



SUPPOSITIONS

- Anneau unidirectionnel ou bidirectionnel.
- Orientation locale.
- Identificateurs distincts.

Exactitude et Terminaison

Pour pouvoir terminer:

Deux options: *n est connu* ou les messages sont ordonnés selon le principe du *premier arrivé, premier servi* (FIFO).

Note: la taille *n* peut etre calculée

States: S={ASLEEP, AWAKE, FOLLOWER, LEADER}
S INIT={ASLEEP};
S_TERM={FOLLOWER, LEADER}.

ASLEEP

Spontaneously
INITIALIZE
become AWAKE

Receiving(``Election'', value, counter)
INITIALIZE;
send(``Election'', value, counter+1)
to other
min:= Min{min, value}
count:= count+1
become AWAKE

<u>INITIALIZE</u>

count:= 0
size:= 1
known:= false
send("Election",id(x),size) **to right**;
min:= id(x)

AWAKE
Receiving ("Election", value, counter)
```
If value ≠ id(x) then
    send ("Election",value,counter+1) to other
    min:= MIN{min,value}
    count:= count+1
    if known = true then
        CHECK
    endif
else
    ringsize:= counter
    known:= true
    CHECK
endif
```

<u>CHECK</u>
```
if count = ringsize then
    if min = id(x) then
        become LEADER
    else
        become FOLLOWER
    endif
endif
```

Complexité

Chaque identificateur traverse tous les liens --> n^2

La taille de chaque message est log(*id*)

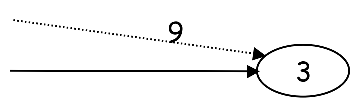
$O(n^2)$ messages
$O(n^2 \log(id_M))$ bits

Observations:

1. L'algorithme résout aussi le problème de la collecte de données.
2. Il fonctionne aussi quand l'anneau est bidirectionnel.

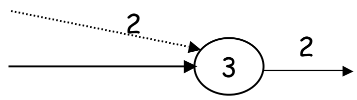
AsFar (as it can)

Idée générale: Il n'est pas nécessaire d'envoyer et de recevoir des messages contenant des identificateurs plus grands que ceux que l'on a déjà vu.

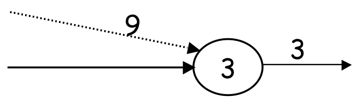


SUPPOSITIONS

- Anneau unidirectionnel ou bidirectionnel
- Identificateurs différents
- Orientation locale

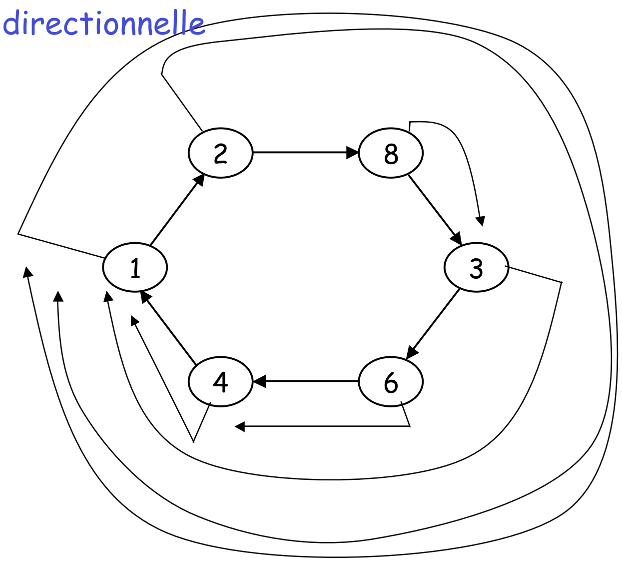


Receiving y plus petit que moi
send(y) aux voisins

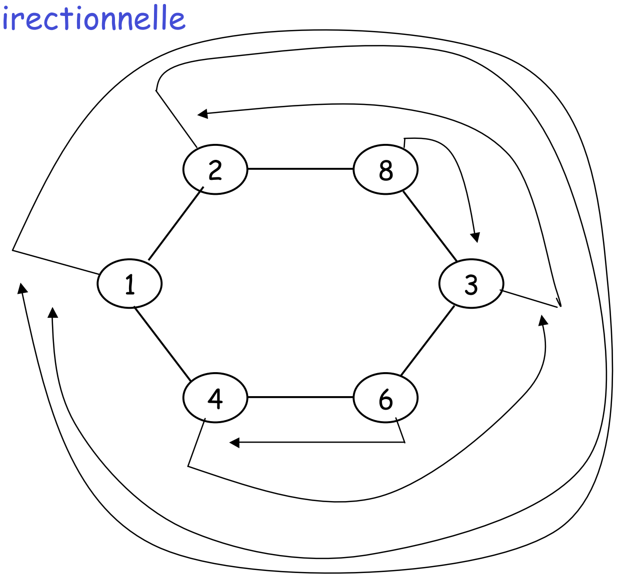


Receiving y plus grand que moi
send(x) aux voisins
(si il n'a pas encore été envoyé)

Version unidirectionnelle



Version bidirectionnelle



States: S={ASLEEP, AWAKE, FOLLOWER, LEADER}
S_INIT={ASLEEP}
S_TERM={FOLLOWER, LEADER}

--- Version unidirectionnelle

ASLEEP

Spontaneously

```
    send("Election",id(x)) to right
    min:= id(x)
    become AWAKE

Receiving("Election", value)
    send("Election",id(x)) to right
    min:= id(x)
    If value < min then
        send("Election", value) to other
        min:= value
    endif
    become AWAKE
```

/* on pourrait eviter ça si id(x)>value

AWAKE

```
Receiving("Election'", value)
    if  value < min then
        send("Election", value)  to other
        min:= value
    else
        If value= min then NOTIFY endif
    endif

Receiving(Notify)
    send(Notify)  to other
    become  FOLLOWER
```

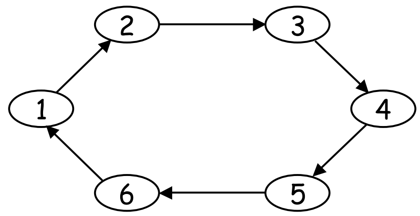
```
NOTIFY
    send(Notify) to  right
    become LEADER
```


Exactitude et Terminaison

Un noeud sait qu'il est le leader lorsqu'il reçoit son propre message

Une notification est necessaire !

Complexité: pire cas (version unidirectionnelle)

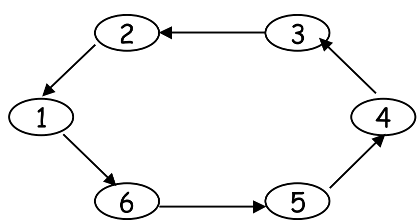


1 ---> n liens
2 ---> n - 1 liens
3 ---> n - 2 liens
… … …
n ---> 1 lien

$n + (n - 1) + \ldots + 1 = \sum_{i=1}^{n} i = (n+1)(n)/2$

Total: $(n+1)(n)/2 + n = O(n^2)$
Dernier terme n : notification

Complexité: meilleur cas (version unidirectionnelle)



1 ---> n liens
$\forall i \neq 1$ ---> 1 lien (--> total = n - 1)

Total: $n + (n - 1) + n = O(n)$
Dernier terme n: notification

Complexité: Moyenne

Les entités sont ordonnées de façon à ce que les probabilités soient égales.

J-ème plus petite id - traverse (n / J) liens

$$\sum_{J=1}^{n} (n / J) = n * Hn$$

Série harmonique de n nombres
(approx. 0.69 log n)

Total: n * Hn + n = 0.69 n log n + O(n) = O(n log n)

Controlled Distance

Idée générale: déroulement en phases. Une entité garde le contrôle de son message.

SUPPOSITIONS

- Anneau bidirectionnel
- identificateurs distinct
- Orientation locale

avec sens de la direction pour simplicité (pas nécessaire)

Les Idées

1) Distance limitée (pour éviter que les "grands" identificateurs voyagent trop)
Ex: phase i: distance 2^{i-1}

2) Messages de retour (si le msg rencontre un identificateur plus petit, il ne continue pas)

3) Vérification des deux côtés

4) Le plus petit gagne toujours (peu importe le numéro de phase)

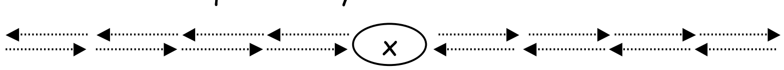
Les *candidats* commencent l'algorithme

Phase i:

- Chaque *candidat* envoie un message avec son propre identificateur dans les deux directions

-Le message voyage jusqu'à ce qu'il rencontre un noeud ayant un identificateur plus petit ou jusqu'à une distance prédéterminé

- S'il ne rencontre pas un identificateur plus petit, le message revient au noeud qui l'a envoyé.

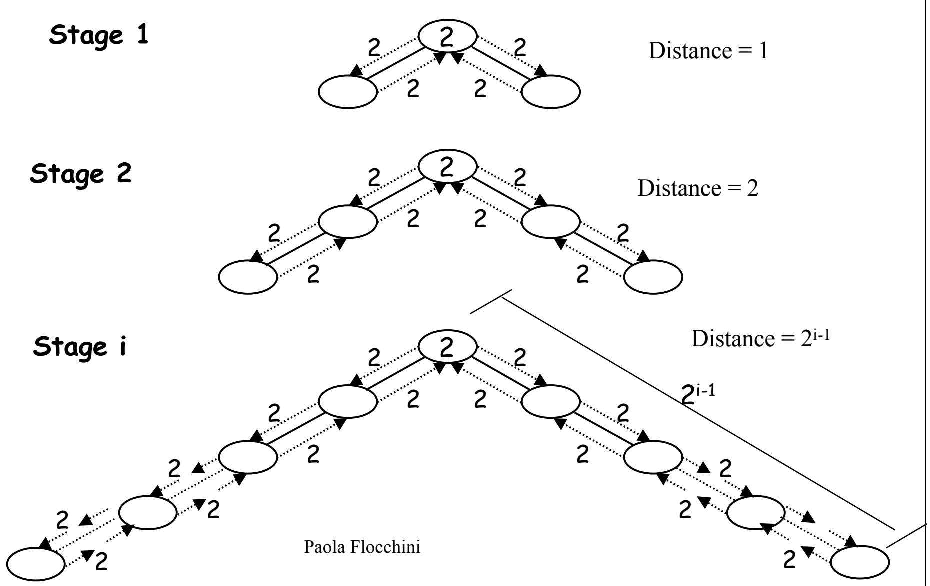


-Le candidat qui reçoit son propre identificateur dans le deux directions, passe à la prochaine phase

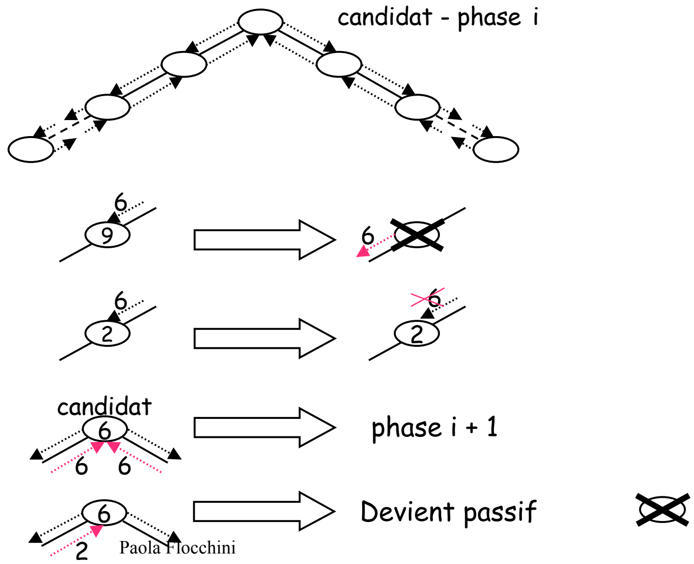
Considérons les entités sur le chemin d'un message Id:

- Chaque entité *i* avec identificateur Id_i plus grande que Id devient passive
- Une entité passive achemine les messages qu'elle reçoit. Si le message reçu est un message de terminaison, elle termine.

Exemple



Exemple



Exactitude et Terminaison

Si un candidat reçoit son message dans la direction opposée à la direction d'envoi. Il devient leader et il averti les autres noeuds de l'élection.

- L'identificateur le plus petit traverse toujours la distance maximale, rendant passive toutes les entités qu'il rencontre

-La distance croit de façon monotone et elle devient forcément plus grande que *n* à une certaine phase.

- Le leader recevra son message dans les directions opposées

Note: Il n'est pas nécessaire que l'ordre des messages soit conservé. Qu est-ce qu'il arrive si une entité reçoit un message d'une phase superieure à la sienne ?

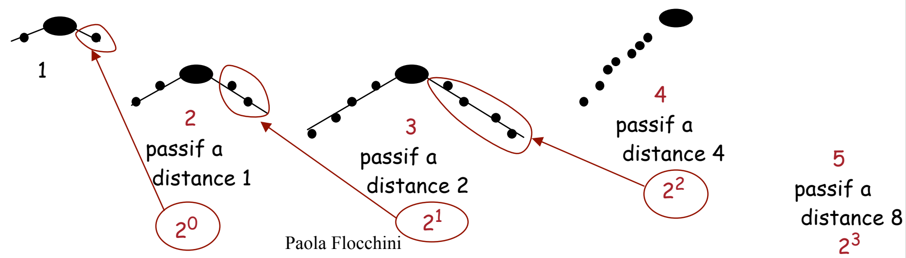
Complexité en Message

Quand la distance double à chaque phase dis(i) = 2^{i-1}:

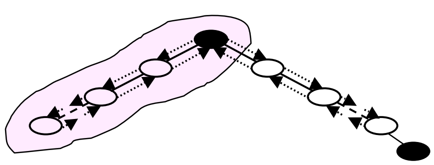
Notion de phase logique

n_i entités commencent la phase i

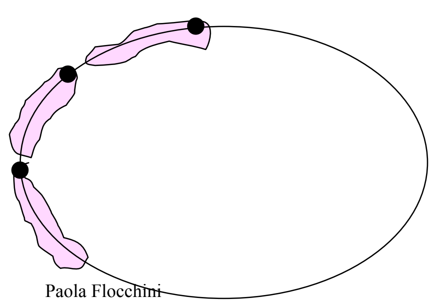
Si x commence la phase i (il a donc survécu la phase i-1) l' identificateur de x doit être plus petit que les identificateurs des ses voisins jusqu'à une distancede 2^{i-2} dans les deux directions



Dans chaque groupe de $2^{i-2}+1$ entités consécutives, une au plus peut survivre la phase i-1.



$$n_i \leq n/(2^{i-2}+1)$$



En commencant la phase i:

Messages "Forth" :

Chaque message traverse au plus 2^{i-1} noeuds dans les deux directions

Tot: $2\, n_i\ \ 2^{i-1}$

Messages "Back" :

Chaque entité qui survit reçoit un message de chaque côte

$$2\, n_{i+1}\ \ 2^{i-1}$$

Chaque entité qui a commencé la phase mais n'a pas survécu reçoit un seul ou aucun message $\leq (n_{i+1} - n_i)\ 2^{i-1}$

Tot: $2\, n_{i+1}\ \ 2^{i-1} + (n_{i+1} - n_i)\ 2^{i-1}$

Phase i>1

Tot: $2\, n_i\ \ 2^{i-1} + 2\, n_{i+1}\ \ 2^{i-1} + (n_{i+1} - n_i)\ 2^{i-1}$

$= (3\, n_i + n_{i+1})\ 2^{i-1}$

$\boxed{n_i \leq n/\ (2^{i-2} + 1)}$

$\leq\ (3\ \lfloor n/\ (2^{i-2} + 1)\rfloor + \lfloor\ n/\ (2^{i-1} + 1)\ \rfloor\ )\ 2^{i-1}$

$< \dfrac{3\, n\, 2^{i-1}}{(2^{i-2} + 1)} + \dfrac{n\, 2^{i-1}}{(2^{i-1} + 1)}$

$< \dfrac{3\, n\, 2^{i-1}}{2^{i-2}} + \dfrac{n\, 2^{i-1}}{2^{i-1}}$

$= 6\, n + n = 7\, n$

La première phase est un peu différente:

Si toutes les entités commencent:

Les survivants: $4\ n_2\ 2^0$ 2 "forth", 2 "back"

Les autres: $3\ (n- n_2)\ 2^0$ 2 "forth", 1 "back"

$$\boxed{n_2 \leq n/\ (2^0+1)}$$

$$\begin{aligned} 4\ n_2 + 3\ n - 3n_2 &= n_2 + 3\ n \\ &= n/2 + 3\ n \\ &< 4n \end{aligned}$$

Nombre total de phases

L'anneau est traversé complètement quand
2^{i-1} est plus grand ou égale à n

$$2^{i-1} \geq n$$

C'est à dire quand:

$$i \geq \log n + 1$$

----> $\log n + 1$ phases

Première phase

$$TOT \leq \sum_{i=1}^{\log n} 7n \quad + O(n)$$

$$= n \sum_{i=2}^{\log n} 7 = 7\, n \log n + O(n)$$

$$\boxed{O(n \log n)}$$

Conjecture:

Dans les anneaux unidirectionnels, la complexité dans le pire cas est $\Omega(n^2)$; pour obtenir une complexité de $O(n \log n)$ messages, la bidirectionnalité est nécessaire.

FAUX !

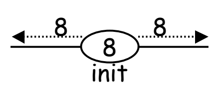
Stages

Idée générale: par phases.

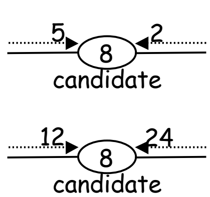
SUPPOSITIONS

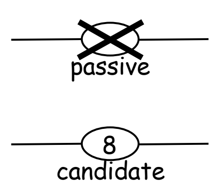
- Identificateurs différents
- Anneau bidirectionnel (+ version unidirectionnelle)
- Orientation locale
- Messages ordonnées

Chaque entité *candidate* envoie son propre *identificateur* dans les deux directions.

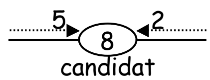


Quand une entité candidate *i* reçoit deux messages Id_j (de la droite) et Id_k (de la gauche), elle détermine si elle devient *passive* (= ce n'est pas la plus petite), ou si elle demeure *candidate* (= c'est la plus petite).





Quand un *candidat i* recoit deux messages
Id_j (de droite) and Id_k (de gauche),

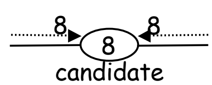


Apres avoir reçu le premier message: **`close-port`**
(stoque les messages qui pourraient arriver plus tard)

Apres avoir reçu le deuxième message: exécute l' action
et re-**`open-port`**

Exactitude et Terminaison

L'entité minimale n'arrêtera jamais d'envoyer
des messages.

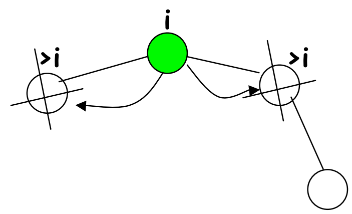


Quand une entité sait qu'elle est le *leader*, elle envoie
un message de *notification* qui voyage autour de l'anneau.

Complexité - Pire Cas

À chaque phase: Au moins la moitié des entités deviennent passives.

$$n_{i+1} \leq \frac{n_i}{2}$$



$n_0 = n$
$n_1 \leq n/2$

$n_i \leq n/2^i$

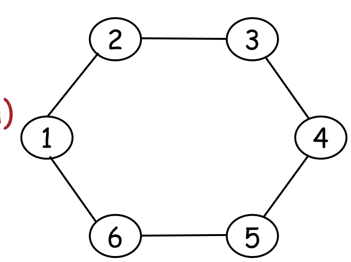
$\Rightarrow$ $n/2^k \leq 1$ quand $k \geq \log n$

 # phases: Au plus $\lfloor (\log n) \rfloor$

Chaque entité envoie ou laisse passer 2 messages.
 # messages: 2n
 # bits: $2n * \lfloor (\log n) \rfloor$

Dernière entité: 2n messages pour comprendre qu'elle est la dernière entité active, et ensuite, n messages de notification.

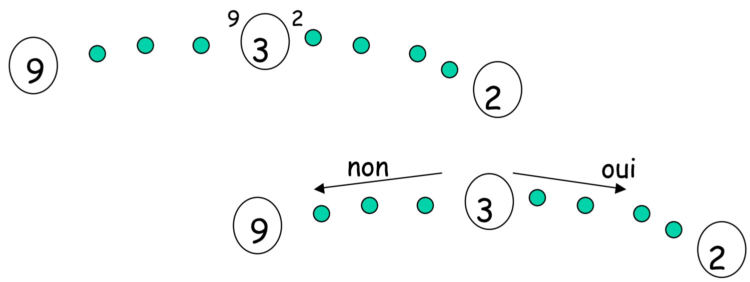
Total: $2n * \lfloor (\log n) \rfloor + 3n = O(n \log n)$



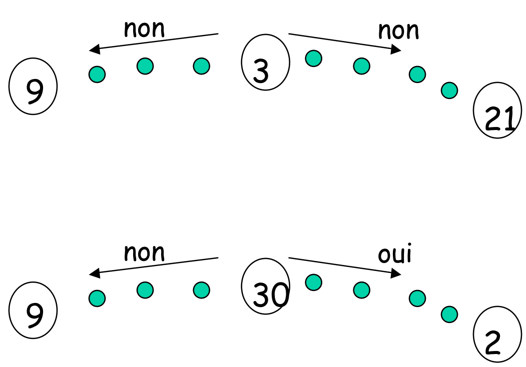
Meilleur Cas?

Stages avec Feedback

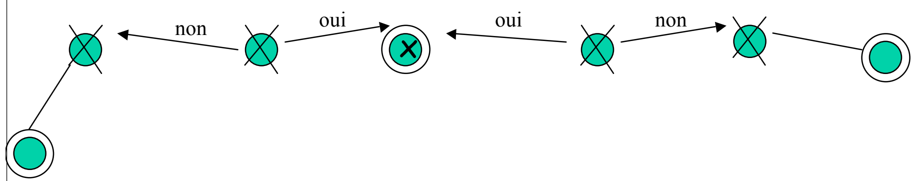
Un “feedback” est envoyé au candidat voisin



send oui au plus petit de deux s'il est plus petit que moi
(sinon send no)
send no aux autres



Si *x* survit, il doit avoir reçu un feedback de la part des deux candidats voisins…



$$n_{i+1} \leq \frac{n_i}{3}$$

Version unidirectionnelle

Simulation de l'algorithme bidirectionnel avec la même complexité.

La conjecture d'Hirschberg and Sinclair est fausse.

Exemple

Alternating Steps

Idée générale: Les directions alternent.

- Identificateurs uniques.
- Anneau bidirectionnel et sens de l'orientation.
- Orientation locale.
- Messages ordonnés.

Envoi à gauche
commence l'élimination (si possible)
envoi à droite

Algorithme:

1. Chaque entité envoie un message à sa droite. Ce message contient l'*identificateur* de l'entité.

2. Chaque entité compare l'*identificateur* qu'elle a reçue de sa gauche avec le sien.

3. Si son propre identificateur est plus grand que celui reçu, l'entité devient passive.

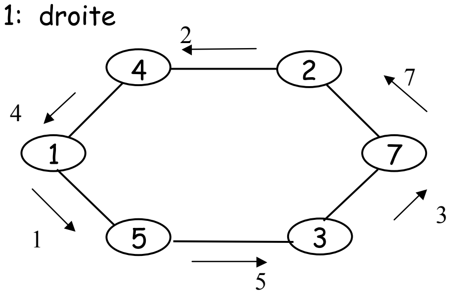
4. Toutes les entités qui sont demeurées actives (survivantes) envoient leurs *idetificateurs* à leur gauche.

5. Une entité survivante compare l'*identificateur* qu'elle a reçu de sa droite avec le sien.

6. Si son propre *identificateur* est plus grande que celui reçu, l'entité devient passive.

7. Retourner à l'étape 1 et répéter jusqu'à ce qu'une entité reçoive son propre identificateur et devienne *leader*.

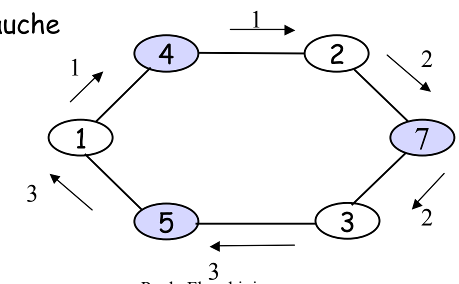
Phase 1: droite



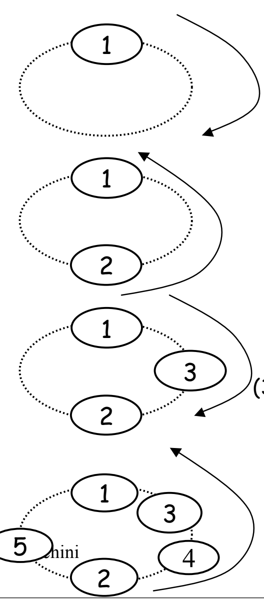
candidat

vaincu

Phase 2: gauche



Complexité



Dernière phase k

1 entité active

Phase k - 1

au moins 2 entités actives

(2) deviendra passive à la prochaine étape.

Phase k - 2

au moins 3 entités actives

(3) doit être là; autrement, (2) deviendrait vaincu.

Phase k - 2

au moins 5 entités actives

1 2 3 5 8 13 21 ....

 # phases =
index du plus petit nombre de Fibonacci >= n

$F_1 = 1$
$F_2 = 2$
$F_3 = 3$
$F_4 = 5$
$F_5 = 8$
...

$F_k = i = ?$
$= \text{approx. } 1.45 \log_2 n$

Total $= \text{approx. } 1.45\, n \log_2 n$

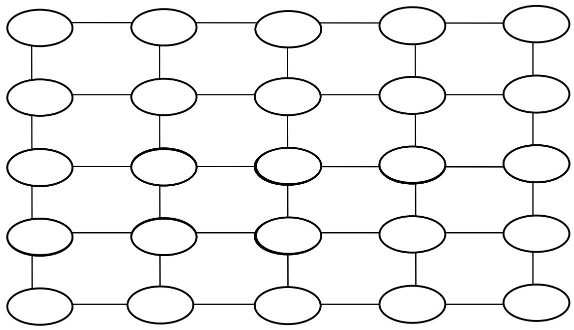
Limites superieures

Bidirectional		Unidirectional	
LeLann (1977) "All the way"	n^2	LeLann (1977) Unidirectional simulation	n^2
Chang & Roberts (1979) "As far as you can" average case n log n	n^2	Chang & Roberts	n^2
Hirshberg & Sinclair (1980) stages message control	$7n \log n$		
Franklin (1982) stages	$2n \log n$	Dolev, Klawe & Rodeh Unidirectional simulation	$2n \log n$
Peterson (1982) Alternate	$1.44n \log n$	Peterson 1982 Unidirectional simulation	$1.44n \log n$
		Dolev, Klawe & Rodeh (1982)	$1.36n \log n$
		Higham, Przytycka (1984)	$1.22n \log n$

Limites inferieures

Burns	$0.5n \log n$
Pachl, Korach Rotem (1984)	$0.69n \log$

Grille



Si c'est une grille carré: n noeuds = $n^{\frac{1}{2}} \times n^{\frac{1}{2}}$

$m = O(n)$

Topologie asymétrique
- coins
- bordure
- noeuds internes

Idée: Élire un des coins

Trois phases:

1) Réveil (Wake up)

2) Élection (sur le bord, parmi les coins)

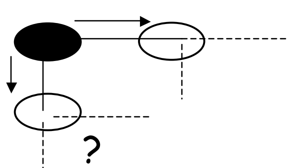
3) Notification

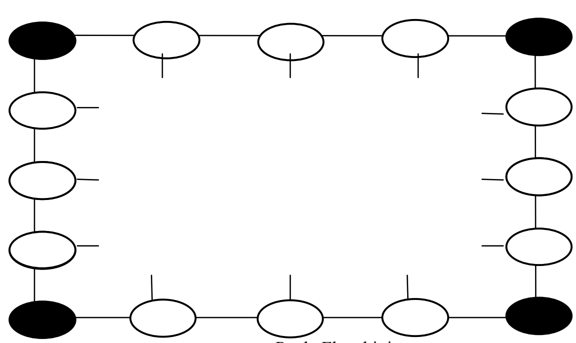
1) Wake up

- Chaque initiateur envoie un message de réveil à ses voisins
- A non-initiateur qui reçoit un message de réveil l'envoie à ses autres voisins

$O(m) = O(n)$

2) Election sur le bord, commencé par les coins





$O(\sqrt{n})$

3) Notification

Flooding

$O(m) = O(n)$

TOT: $O(n)$

Torus