

ELECTION

4I403 – Algorithmique Répartie

Plan

2

I. Introduction

1. Principe
2. Applications
3. Impossibilité

II. Topologie en anneau

III. Topologie quelconque avec hypothèses

I.1 Principe

3

Objectif

Désigner un chef dans un ensemble de processus

Propriétés

1. Sûreté

Le chef doit être unique

2. Vivacité

La désignation doit se faire en un temps fini

I.2 Applications

4

Orchestration

- ▣ Mise en place d'un contrôle centralisé
- ▣ Allocation de ressources en exclusion mutuelle
- ▣ Accès à des données répliquées
- ▣ Consensus
- ▣ Résolution de situations de blocage

I.2 Applications

5

Recouvrement de défaillance

- ▣ Restauration d'un maître dans un système maître/esclaves
- ▣ Régénération de jeton perdu

Agrégation d'informations

- ▣ Calcul de la taille d'un réseau
- ▣ Détection de terminaison d'un algorithme

I.3 Impossibilité

6

[Angluin 1980]

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

Idée de la preuve

1. Réseau anonyme \Rightarrow config. de départ peut être symétrique
2. Config. objectif (leader unique élu) est asymétrique
3. Il existe une exécution E du système telle que
à partir d'une configuration symétrique
on passe **toujours** dans une configuration symétrique

I.3 Impossibilité

7

Contourner l'impossibilité

- ▣ via des identifiants

Relation d'ordre entre identifiants brise la symétrie

- ▣ via des algorithmes probabilistes

Tirages aléatoires sur chaque nœud

Tomber à l'infini sur des configs symétriques est quasi-impossible

Algorithmes Las Vegas

S'arrête toujours avec une réponse exacte

Toutes les configurations terminales sont correctes

Plan

8

I. Introduction

II. Topologie en anneau

1. Anneau unidirectionnel asynchrone (Chang-Roberts)
2. Anneau unidirectionnel synchrone anonyme (Itai-Rodeh)
3. Anneau bidirectionnel asynchrone (Hirschberg-Sinclair)

III. Topologie quelconque avec hypothèses

II.1 Unidirectionnel synchrone

9

[Chang & Roberts 1979] (adapté de LeLann77)

Hypothèses

- Les nœuds sont capables de s'organiser en anneau unidirectionnel
- Chaque nœud possède un identifiant unique dans l'anneau
- Chaque nœud i dispose d'une référence $\text{succ}[i]$ vers son successeur
- Aucun nœud ne connaît la taille de l'anneau
- Plusieurs candidats simultanés possibles
- Communications fiables asynchrones FIFO
- Exécution sans faute

Idée

- Chaque candidat propage sa candidature dans l'anneau
- Un candidat qui reçoit une candidature supérieure a perdu
- Le processus candidat d'identifiant maximal reçoit sa propre candidature

II.1 Unidirectionnel synchrone

10

[Chang & Roberts 1979] - Algorithme

Variables locales

Etat – non-candidat/candidat/élu/perdu, initialisé à non-candidat

leader – identité du leader, initialisé à NULL

succ[i] – successeur de i dans l'anneau

Site i se porte candidat

Etat = candidat

leader = i

envoyer(<ELEC, i>) à succ[i]

II.1 Unidirectionnel synchrone

11

[Chang & Roberts 1979] – Algorithme (suite)

Site i reçoit $\langle \text{ELEC}, j \rangle$

if $i > j$ then

if Etat = non-candidat then

Site i se porte candidat

else if $i < j$ then

Etat = perdu

leader = j

envoyer ($\langle \text{ELEC}, j \rangle$) à succ[i]

else if $i = j$ then

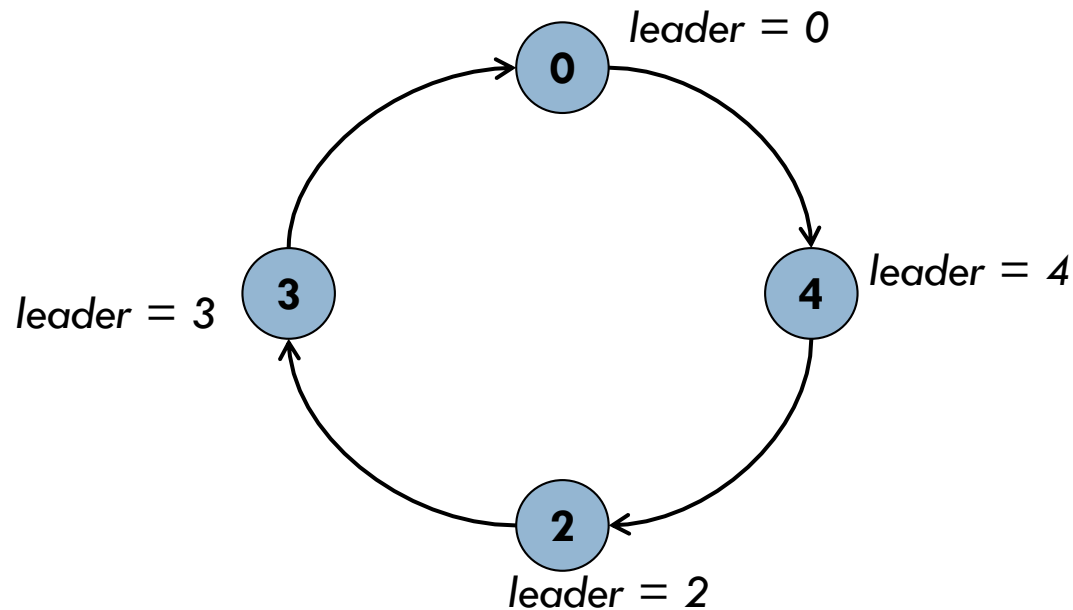
etat = élu

envoyer ($\langle \text{LEADER}, i \rangle$) à succ[i]

II.1 Unidirectionnel synchrone

12

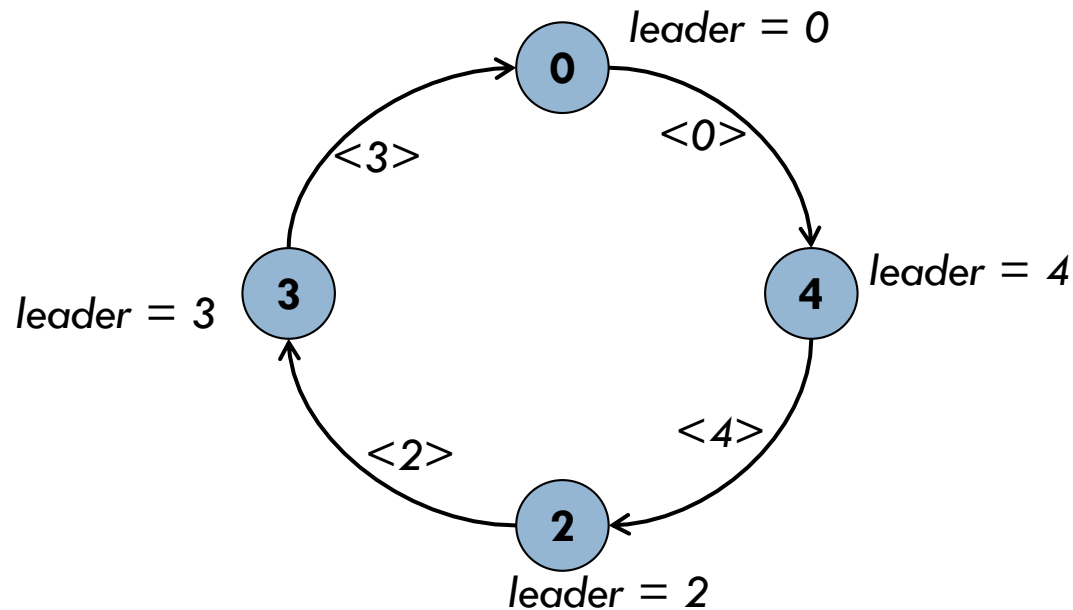
[Chang & Roberts 1979] - Exemple



II.1 Unidirectionnel synchrone

13

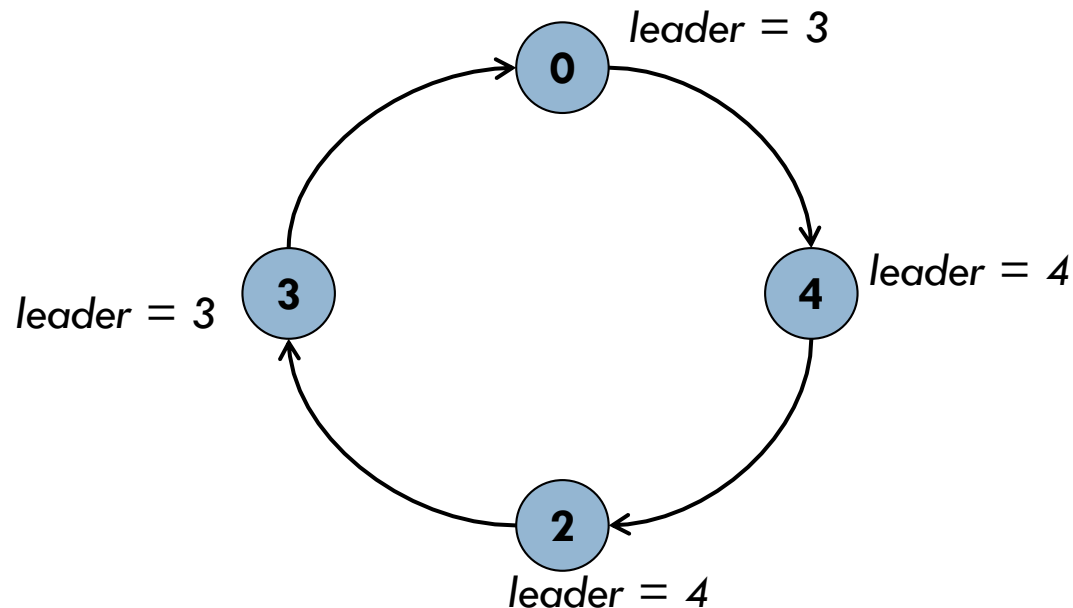
[Chang & Roberts 1979] - Exemple



II.1 Unidirectionnel synchrone

14

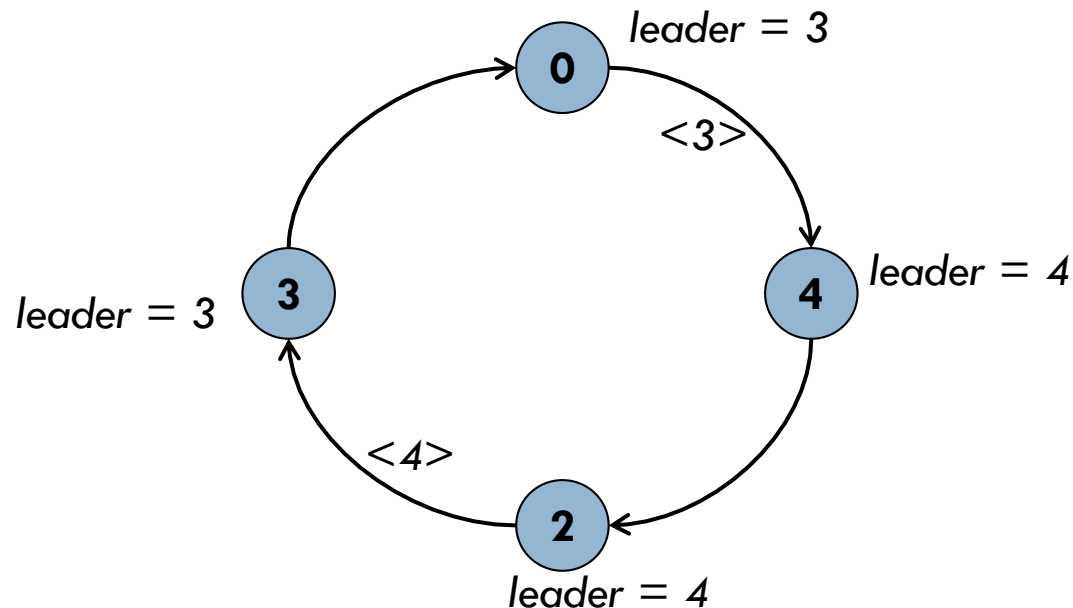
[Chang & Roberts 1979] - Exemple



II.1 Unidirectionnel synchrone

15

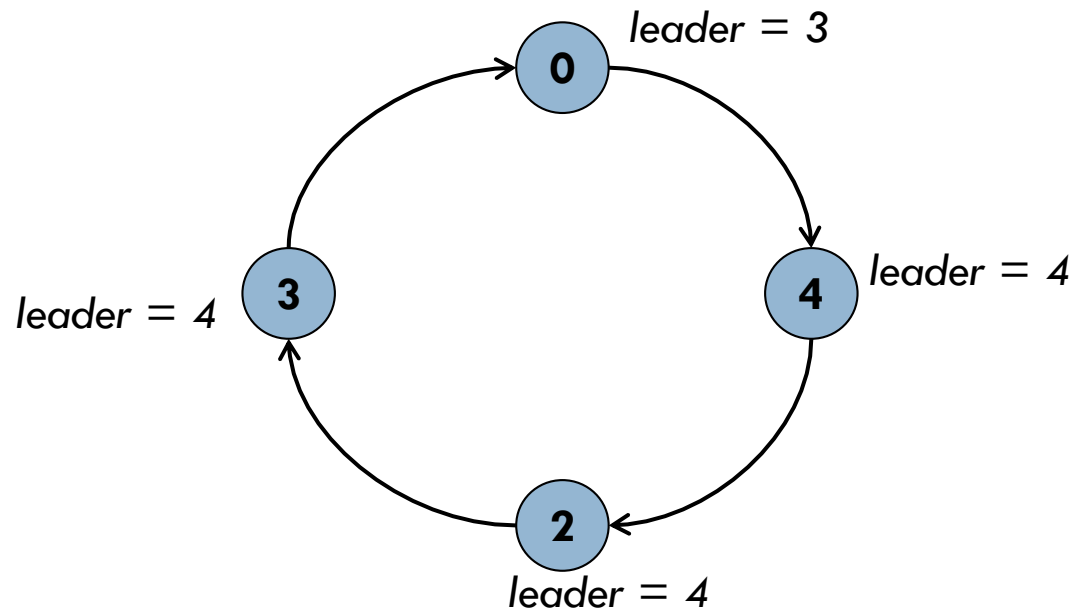
[Chang & Roberts 1979] - Exemple



II.1 Unidirectionnel synchrone

16

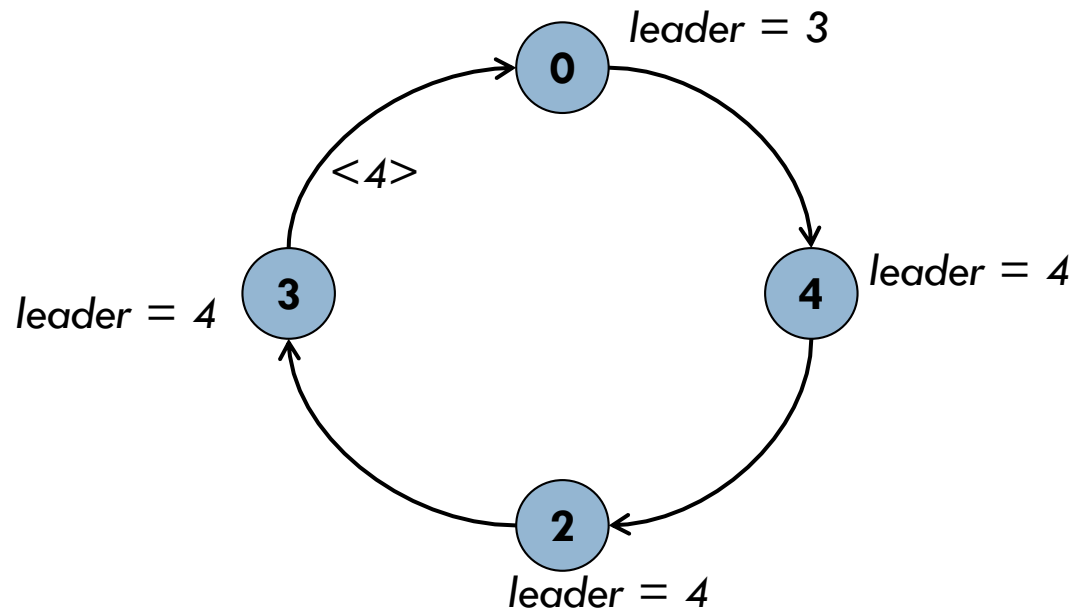
[Chang & Roberts 1979] - Exemple



II.1 Unidirectionnel synchrone

17

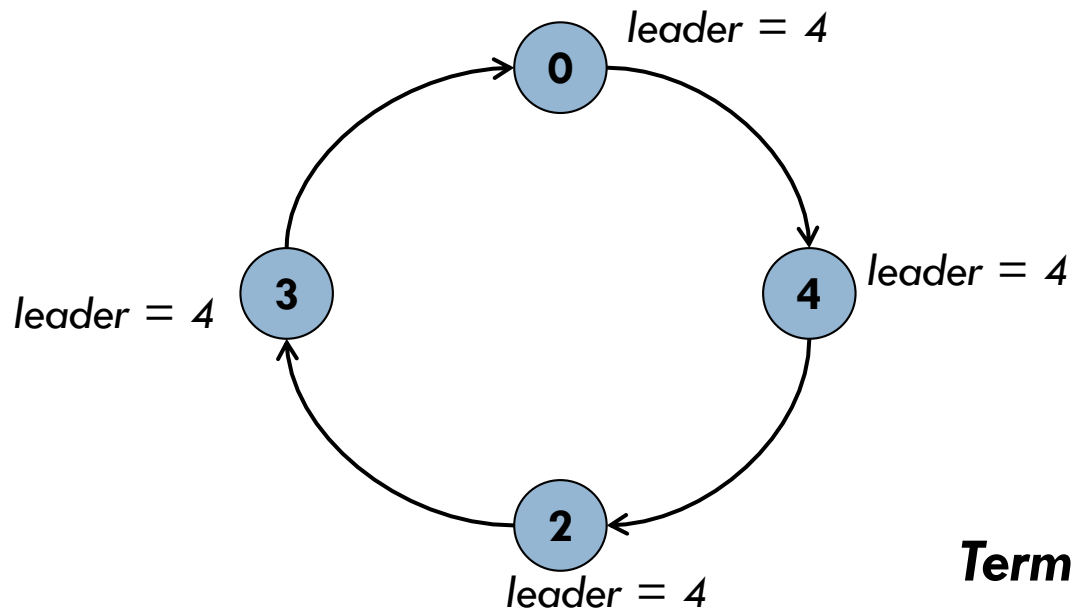
[Chang & Roberts 1979] - Exemple



II.1 Unidirectionnel synchrone

18

[Chang & Roberts 1979] - Exemple



Terminaison ?

II.1 Unidirectionnel synchrone

19

[Chang & Roberts 1979] – Complexité

Complexité en temps : $O(N)$

Complexité en mémoire : $O(\log N)$

II.1 Unidirectionnel synchrone

20

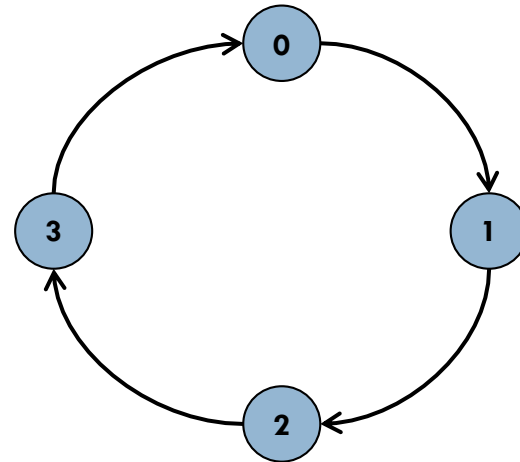
[Chang & Roberts 1979] – Complexité

Complexité en temps : $O(N)$

Complexité en mémoire : $O(\log N)$

Complexité en messages :

Meilleur des cas : $O(N)$



II.1 Unidirectionnel synchrone

21

[Chang & Roberts 1979] – Complexité

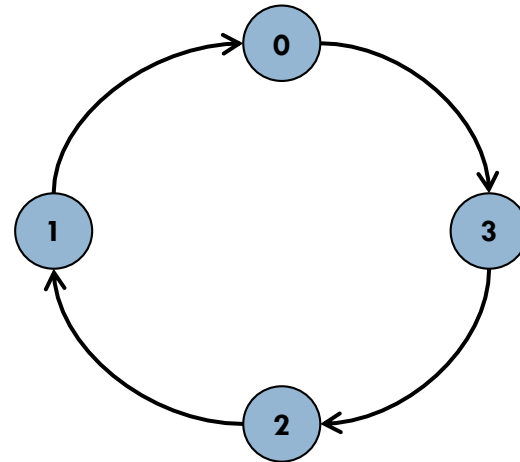
Complexité en temps : $O(N)$

Complexité en mémoire : $O(\log N)$

Complexité en messages :

Meilleur des cas : $O(N)$

Pire des cas : $O(N^2)$



II.2 Unidirectionnel synchrone anonyme

22

[Itai & Rodeh 81]

Basé sur l'algorithme de Chang et Roberts

Hypothèses

Les nœuds sont capables de s'organiser en anneau unidirectionnel

~~Chaque nœud possède un identifiant unique dans l'anneau~~

Chaque nœud i dispose d'une référence $\text{succ}[i]$ vers son successeur

Chaque nœud connaît la taille N de l'anneau

Plusieurs candidats simultanés possibles

Communications fiables et synchrones

Exécution sans faute

II.2 Unidirectionnel synchrone anonyme

23

[Itai & Rodeh 81]

Initialement, tous les processus sont actifs

Chaque processus maintient etat (init. actif) et tournoi (init. 1)

Fonctionnement en tournois entre nœuds actifs

Chaque nœud actif tire un identifiant id aléatoirement dans $[1, k]$ ($k \geq N$) puis envoie (id, 1, 1, vrai) au successeur

Lorsqu'un processus p reçoit (#id, #tournoi, #saut, unique)

Si p est inactive, alors il envoie (#id, #tournoi, #saut+1, unique) au successeur

sinon // p est active alors

si #saut = N alors // je suis l'initiateur du tournoi

alors si unique alors etat = élu

sinon tournoi = tournoi + 1

p tire un nouvel id aléatoirement dans $[1, k]$ puis envoie (id, tournoi, 1, vrai) au successeur

sinon // #saut \neq N

si (id, tournoi) =_{lex} (#id, #tournoi) alors p envoie (#id, tournoi, #saut+1, faux) au successeur

si (id, tournoi) >_{lex} (#id, #tournoi) alors

etat = inactive;

p envoie (#id, tournoi, #saut+1, unique) au successeur

II.2 Unidirectionnel synchrone anonyme

29

[Itai & Rodeh 81] - Analyse

C'est un algorithme Las Vegas

Termine toujours

Cas moyen

Forte probabilité d'un seul candidat à chaque phase

Complexité en nombre de messages est $O(N^2)$

II.3 Bidirectionnel asynchrone

30

[Hirschberg & Sinclair 1980]

Hypothèses

Les nœuds sont capables de s'organiser en anneau bidirectionnel

Chaque nœud possède

- un identifiant unique dans l'anneau

- une référence succ[i] vers son successeur

- une référence pred[i] vers son prédécesseur

Aucun nœud ne connaît la taille de l'anneau

Plusieurs candidats simultanés possibles

Communications fiables et asynchrones

Exécution sans faute

II.3 Bidirectionnel asynchrone

31

[Hirschberg & Sinclair 1980] - Idée

Fonctionnement en rondes asynchrones

- Un nœud actif candidate

- Il émet son identifiant dans les 2 directions sur des distances croissantes

- Le vainqueur de chaque ronde est celui qui a transmis sur toute la distance

- Un identifiant qui fait le tour de l'anneau désigne le leader

Complexité en messages : $O(N \log N)$

II.3 Bidirectionnel asynchrone

32

[Hirschberg & Sinclair 1980] - Algorithme

Valeurs locales

state = active

leader = local_id

round = 0

Début de ronde k (init. à 0)

if state = active

envoyer $\langle \text{leader}, 2^k \rangle$ à voisins de gauche et droite

attendre retour des deux messages

k++ ; cpt = 0

II.3 Bidirectionnel asynchrone

33

[Hirschberg & Sinclair 1980] – Idée de l'algorithme

Sur réception de $\langle j, TTL \rangle$ venant de voisin gauche (droite)

if $j = i$ then

if $TTL > 0$ then **se déclarer vainqueur**

if $j > i$ and $TTL \geq 1$ then

state = inactive

leader = j

if $TTL > 1$ then

envoyer $\langle j, TTL - 1 \rangle$ a voisin droite (gauche)

else

envoyer $\langle j, 0 \rangle$ a voisin gauche (droit)

if $TTL = 0$ then

if $(j \neq i)$

envoyer $\langle j, 0 \rangle$ a voisin droite (gauche)

else

cpt++

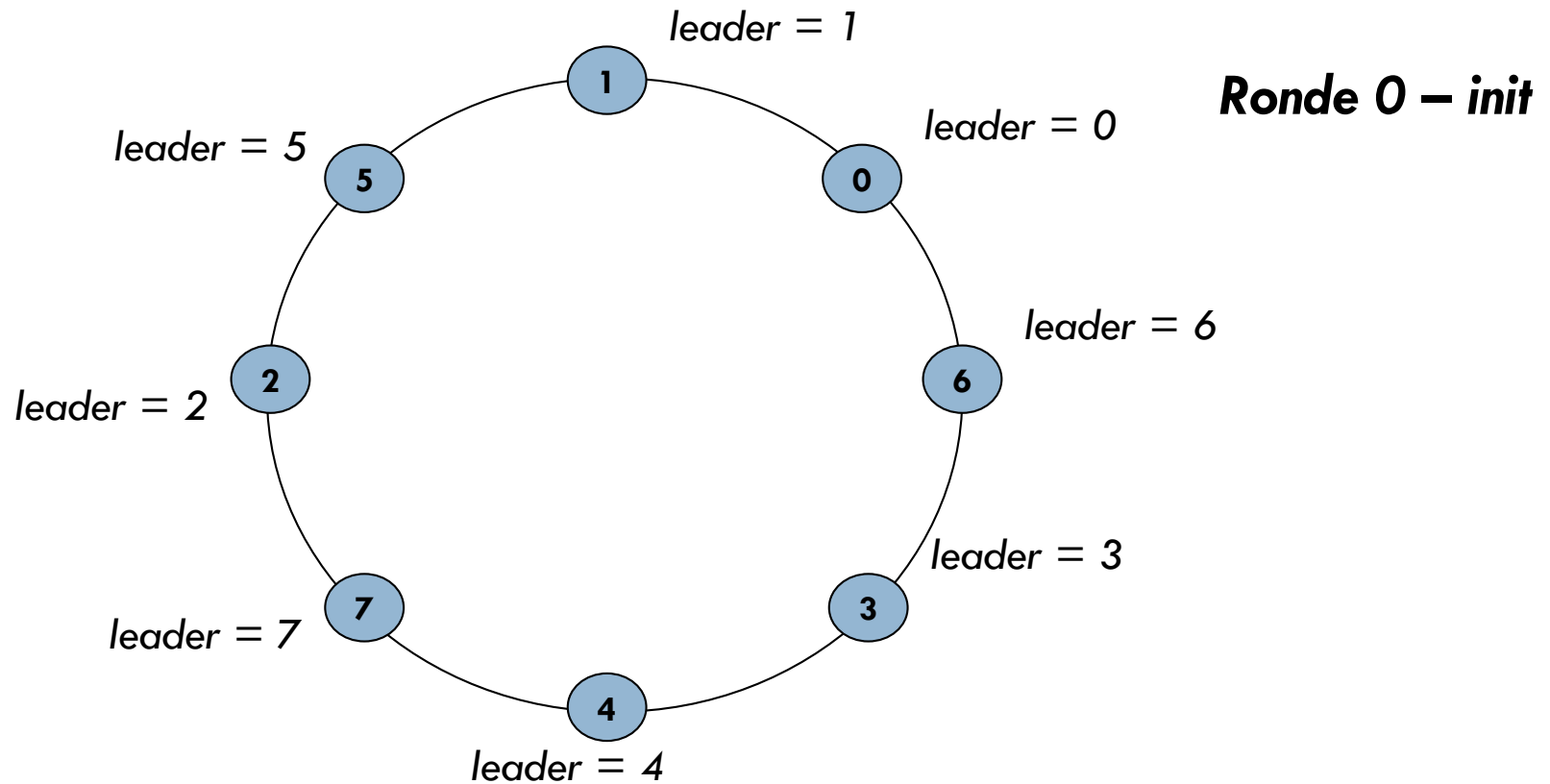
if (cpt = 2)

début de nouvelle étape

II.3 Bidirectionnel asynchrone

34

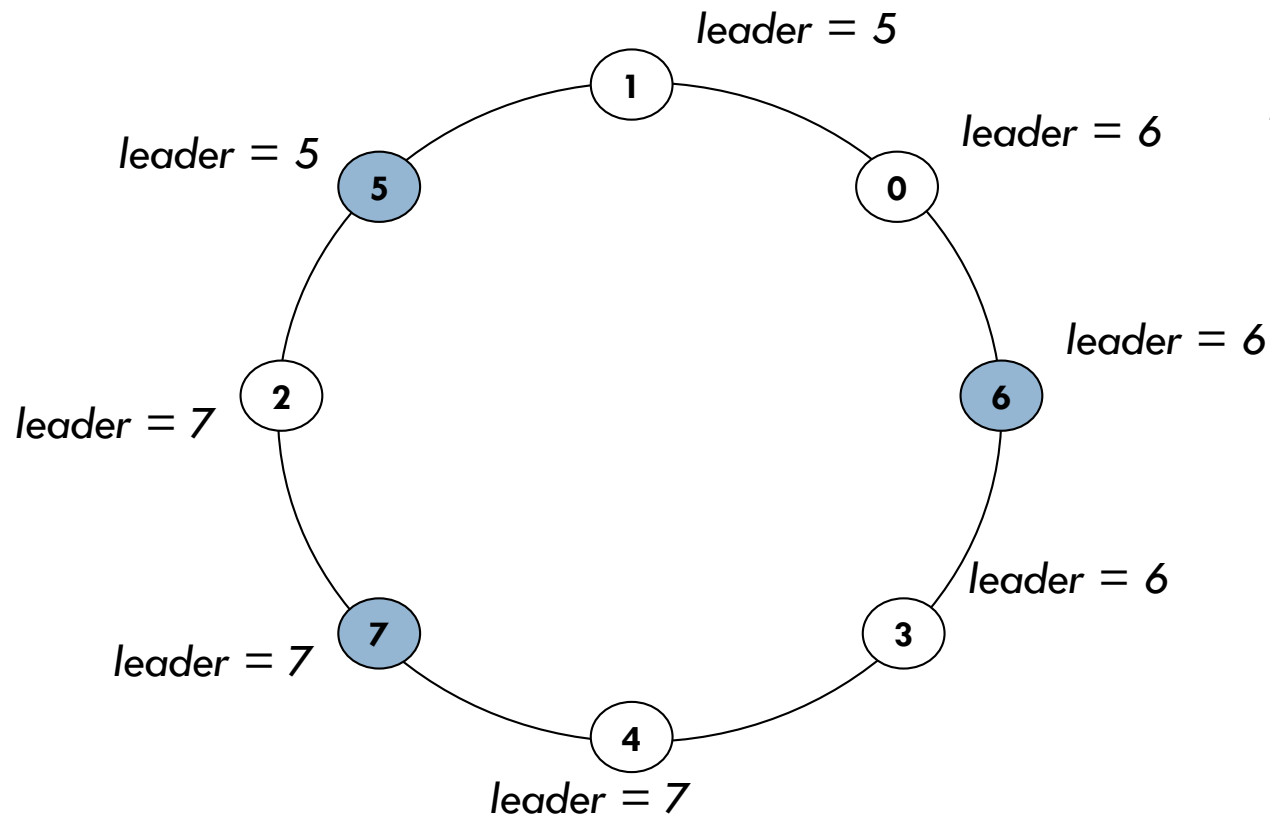
[Hirschberg & Sinclair 1980] – Exemple



II.3 Bidirectionnel asynchrone

35

[Hirschberg & Sinclair 1980] – Exemple

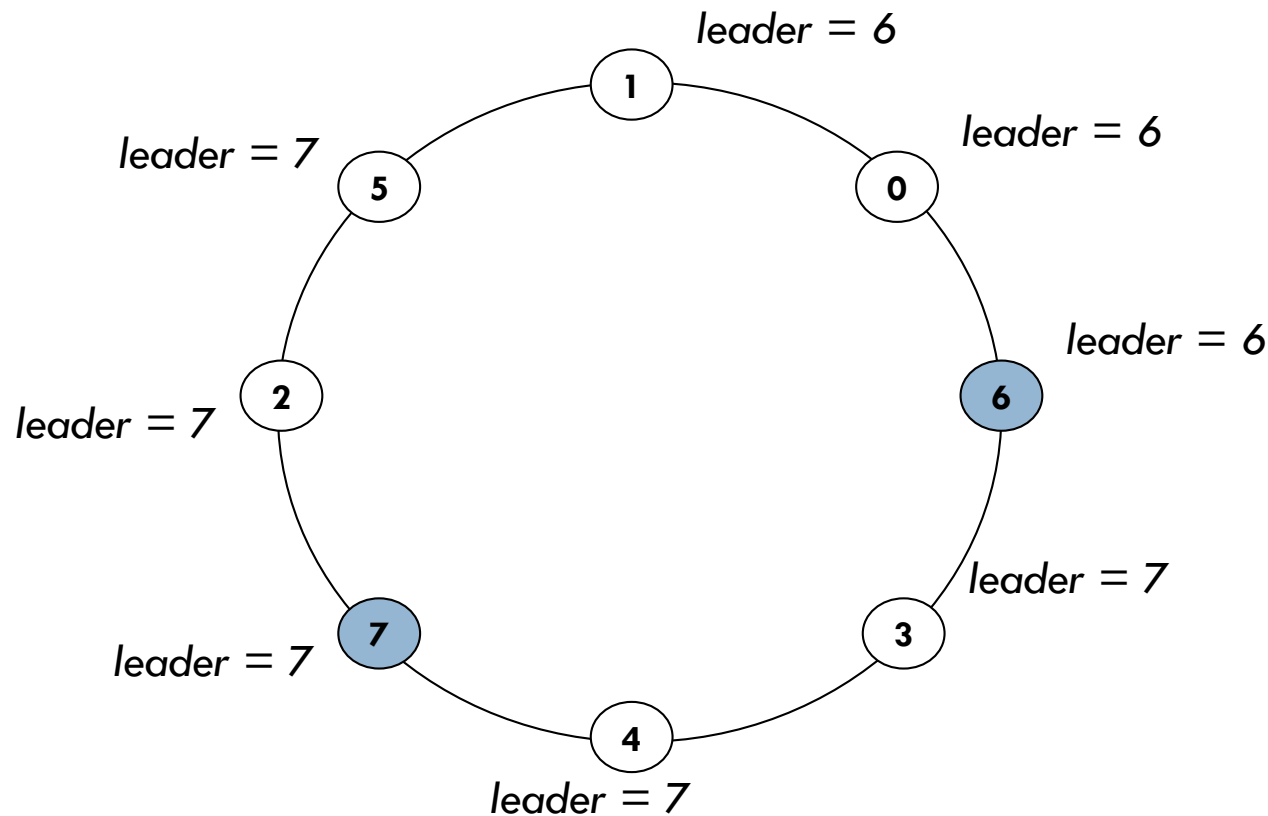


Ronde 1 – init

II.3 Bidirectionnel asynchrone

36

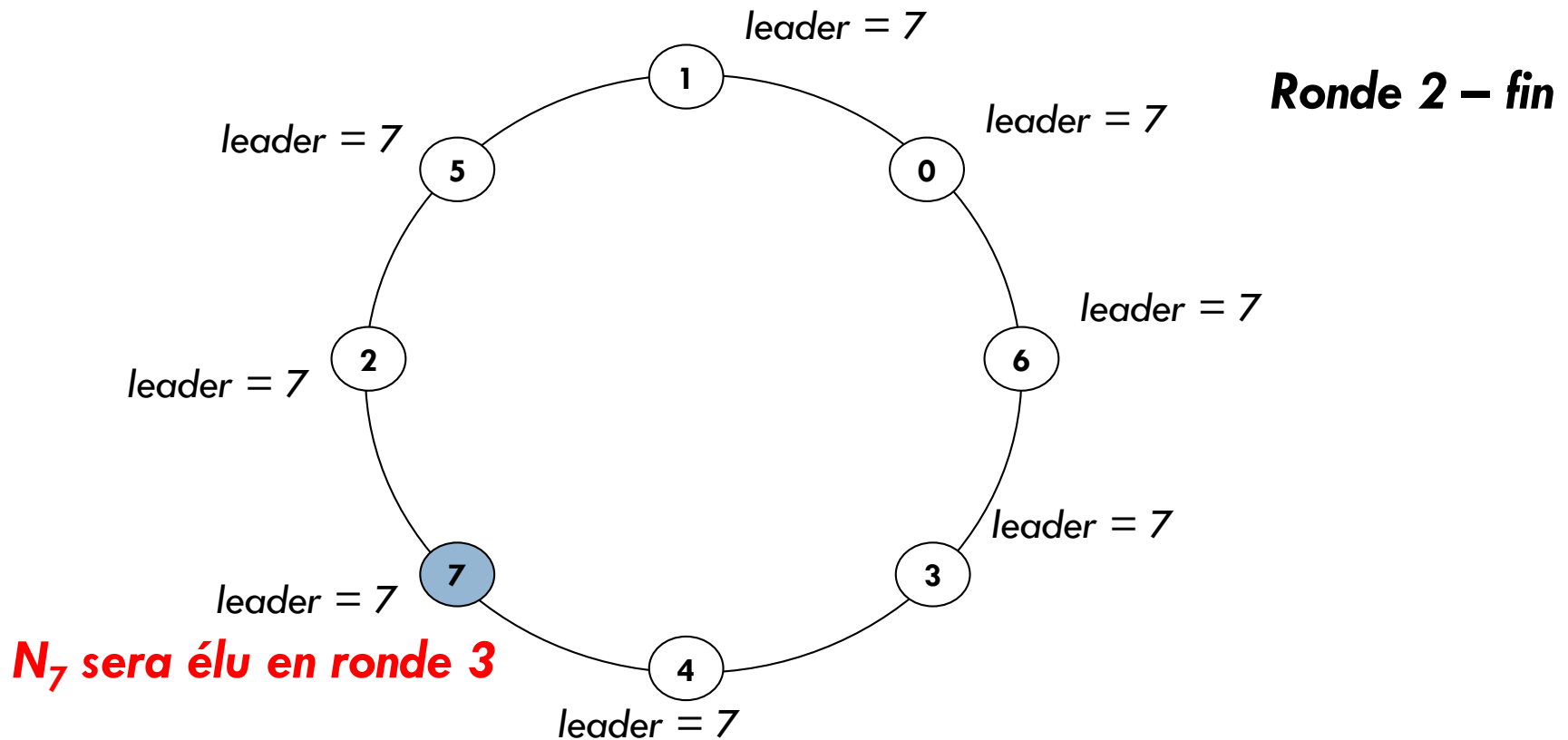
[Hirschberg & Sinclair 1980] – Exemple



II.3 Bidirectionnel asynchrone

37

[Hirschberg & Sinclair 1980] – Exemple



Plan

38

- I. Introduction
- II. Topologie en anneau
- III. Topologie quelconque avec hypothèses
 - 1. Diffusion avec identifiants
 - 2. Diffusion anonyme

III.2 Diffusion avec identifiants

39

Calcul de maximum

Hypothèses

- ▣ Processus avec identifiants distincts
- ▣ Taille du système connue de tous
- ▣ Accès à un broadcast
- ▣ Communications fiables et synchrones

Idée

Diffusion de son identité et attente des identités de tous les autres

Le processus d'identifiant max devient le chef

Complexité en nombre de messages est $O(M)$ dans tous les cas
[$O(NM)$ si initiateurs multiples $\rightarrow O(N^3)$]

III.2 Diffusion avec identifiants

40

FloodMax - Calcul de maximum par propagation

Hypothèses

- Processus avec identifiants distincts
- Diamètre du réseau D connu de tous
- Communications fiables et synchrones

Idée

Valeur *leader* initialisée avec l'identifiant local du processus

D phases

Chaque processus diffuse sa valeur *leader* à ses voisins

Sur réception de *leader'*, $leader = \max(leader, leader')$

Après D phases, tout le monde connaît la valeur maximale pour *leader*

Complexité en nombre de messages est $O(D.M)$ avec M le nombre de liens

III.2 Diffusion avec identifiants

41

Bully [Garcia-Molina 1982]

Hypothèses

- ▣ Processus avec identifiants distincts
- ▣ Accès à un broadcast
- ▣ Communications fiables et synchrones
- ▣ **Pannes franches**

Idée

Chaque candidat envoie son identité aux autres nœuds du réseau

Un site répond à ceux de numéro inférieur au sien. Devient candidat

Un processus qui ne reçoit pas de réponse se déclare le chef

Complexité en nombre de messages est $O(N^3)$ dans le pire des cas

III.3 Diffusion anonyme

42

Candidature aléatoire

Hypothèses

- ▣ Nombre total de nœuds connu de tous
- ▣ Accès à un broadcast
- ▣ Communications fiables et **synchrones**
- ▣ Pannes franches

Idée

Algorithme par phases, avec tous les nœuds initialement actifs

1. Chaque nœud actif décide aléatoirement s'il candidate
2. Un nœud actif diffuse sa (non-)candidature et attend des réponses
3. Si un seul processus candidate il devient le chef
4. Sinon phase suivante

Complexité en nombre de messages est $M(N \log N)$ dans tous les cas

III.3 Diffusion anonyme

43

Candidature aléatoire

Variables locales

State - Le site est à l'état *active*, *inactive* ou *leader*

Values [N] - Tableau contenant les valeurs émises par tous les sites lors d'une phase

Algorithme

State = actif

repeat

$b = \text{random}\{0,1\}$

 broadcast(b)

 while (timer \neq 0) do

 receive($\langle b' \rangle$)

 Values $+= b'$

 if ($b = 1 \wedge \text{Values} = 0$) then State = leader

 if ($b = 0 \wedge \text{Values} > 0$) then State = inactive

until (State \neq active)