



# Terminaison

## Master 2 Informatique - UFR S.A.T

Pr. Ousmane THIARE

ousmane.thiare@ugb.edu.sn  
[[www.ousmanethiare.com](http://www.ousmanethiare.com)]

16 avril 2020

# Chapitre 6 : Terminaison

## Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

- 1 Introduction
- 2 Cas d'un graphe en anneau uni-directionnel
- 3 Cas d'un arbre couvrant
- 4 Cas général



## Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

# Chapitre 6 : Terminaison

# Introduction

## Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

Soit un ensemble de  $n$  tâches interdépendantes au sein d'une application. Ces  $n$  tâches ou processus sont répartis sur un graphe de communication. Chaque processus en cours d'exécution, exécute un algorithme séquentiel et échange des messages avec les autres processus via le graphe de communication.

Le problème est le suivant : l'arrêt de tous les processus correspond-il à l'arrêt définitif de l'application ou n'est-ce qu'un état transitoire ? (c'est-à-dire un message peut être en transit et provoquer le redémarrage d'un, puis de tous les processus) ?



# Introduction

## Définitions

### Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

- Un processus est soit actif soit inactif : il est actif s'il exécute du code, il est inactif s'il n'a rien à faire ;



# Introduction

## Définitions

### Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

- Un processus est soit actif soit inactif : il est actif s'il exécute du code, il est inactif s'il n'a rien à faire ;
- Seuls les processus actifs peuvent envoyer des messages ;



# Introduction

## Définitions

### Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

- Un processus est soit actif soit inactif : il est actif s'il exécute du code, il est inactif s'il n'a rien à faire ;
- Seuls les processus actifs peuvent envoyer des messages ;
- Un processus ne peut passer d'un état inactif à un état actif que sur réception d'un message : à tout message correspond du code à exécuter. Par contre, il peut passer de l'état actif à inactif à tout moment (il a fini son code) ;



# Introduction

## Définitions

### Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

- Un processus est soit actif soit inactif : il est actif s'il exécute du code, il est inactif s'il n'a rien à faire ;
- Seuls les processus actifs peuvent envoyer des messages ;
- Un processus ne peut passer d'un état inactif à un état actif que sur réception d'un message : à tout message correspond du code à exécuter. Par contre, il peut passer de l'état actif à inactif à tout moment (il a fini son code) ;
- Tous les processus sont actifs au lancement de l'application.





# Introduction

## Définitions

### Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

Un deuxième problème auquel nous ne nous intéresserons pas, c'est de savoir si les processus ont "bien terminés" c'est-à-dire de savoir si le résultat est satisfaisant localement (chaque processus a bien fait le calcul prévu : cette estimation est en général faite par le processus lui-même) et globalement (le résultat global est bien le résultat recherché).



# Introduction

## Hypothèses

### Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

- Tout processus qui ne reçoit plus de message se termine dans un temps fini.



# Introduction

## Hypothèses

### Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

- Tout processus qui ne reçoit plus de message se termine dans un temps fini.
- Le réseau est FIFO



# Cas d'un graphe en anneau uni-directionnel

Introduction

**Cas d'un graphe  
en anneau  
uni-directionnel**

Cas d'un arbre  
couvrant

Cas général

Les processus sont “ordonnés” sur le réseau par ordre croissant de leur numéro.



# Cas d'un graphe en anneau uni-directionnel

Principe de l'algorithme (Dijkstra et Van Gasten - 1983)

Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

On utilise un jeton à deux états Termine ou État\_Transitoire. Le processus  $P_0$  qui veut déterminer si l'application est terminée, émet le jeton dans l'état Terminé. Si celui-ci lui revient dans le même état, l'application est terminée. Sinon (donc le jeton revient dans l'état État\_Transitoire), il faut relancer la détection de la terminaison.



# Cas d'un graphe en anneau uni-directionnel

## L'algorithme

Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

Chaque processus  $P_i$  maintient les deux variables locales suivantes :

- $couleur_i = \text{Blanc ou Noir initialisé à Blanc}$

Initialement, le jeton est à l'état Terminé en  $P_0$  qui lance la détection dès que son état passe à Inactif en transmettant le jeton à  $P_1$ .



# Cas d'un graphe en anneau uni-directionnel

## L'algorithme

Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

Chaque processus  $P_i$  maintient les deux variables locales suivantes :

- $couleur_i$  = Blanc ou Noir initialisé à Blanc
- $etat_i$  = Actif ou Inactif initialisé à Actif

Initialement, le jeton est à l'état Terminé en  $P_0$  qui lance la détection dès que son état passe à Inactif en transmettant le jeton à  $P_1$ .



# Cas d'un graphe en anneau uni-directionnel

## L'algorithme

Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

## État du processus et transmission du jeton

- Lorsque  $P_i$  reçoit un message, il passe à l'état Actif





# Cas d'un graphe en anneau uni-directionnel

## L'algorithme

Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

### État du processus et transmission du jeton

- Lorsque  $P_i$  reçoit un message, il passe à l'état Actif
- Un processus Actif ne transmet pas le jeton, il le transmettra dès qu'il deviendra inactif.



# Cas d'un graphe en anneau uni-directionnel

## L'algorithme

Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

### État du processus et transmission du jeton

- Lorsque  $P_i$  reçoit un message, il passe à l'état Actif
- Un processus Actif ne transmet pas le jeton, il le transmettra dès qu'il deviendra inactif.
- Lorsqu'il a fini de calculer, il passe à l'état Inactif.



# Cas d'un graphe en anneau uni-directionnel

## L'algorithme

Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

### État du processus et transmission du jeton

- Lorsque  $P_i$  reçoit un message, il passe à l'état Actif
- Un processus Actif ne transmet pas le jeton, il le transmettra dès qu'il deviendra inactif.
- Lorsqu'il a fini de calculer, il passe à l'état Inactif.
- Lorsqu'il envoie un message vers un processus  $P_j$  tel que  $j < i$ , il passe à l'état Noir.



# Cas d'un graphe en anneau uni-directionnel

## L'algorithme

Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

**Pourquoi ?** Parce qu'il se peut que le processus  $P_j$  soit ré-activé par ce message APRÈS que  $P_j$  ait transmis le jeton à l'état État\_Transitoire. Donc, si  $P_i$  transmet un jeton Termine et que tous les processus  $P_{k>i}$  sont inactifs, alors le jeton arrivera en  $P_0$  dans l'état Termine. Donc si  $P_i$  se contentait de transmettre le jeton en signalant qu'il est inactif, rien ne préviendrait  $P_0$  du réveil de  $P_j$ . D'où lorsque  $P_i$  recevra le jeton, dès qu'il sera inactif, il transmettra le jeton dans l'état Etat\_Transitoire (en fait, cela revient à ce que  $P_i$  demande un nouveau tour). Puis il pourra repasser à Blanc.

D'où, un processus Inactif transmet le jeton d'état :

- identique à l'état du jeton reçu si il est dans l'état Blanc



# Cas d'un graphe en anneau uni-directionnel

## L'algorithme

Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

**Pourquoi ?** Parce qu'il se peut que le processus  $P_j$  soit ré-activé par ce message APRÈS que  $P_j$  ait transmis le jeton à l'état État\_Transitoire. Donc, si  $P_i$  transmet un jeton Termine et que tous les processus  $P_{k>i}$  sont inactifs, alors le jeton arrivera en  $P_0$  dans l'état Termine. Donc si  $P_i$  se contentait de transmettre le jeton en signalant qu'il est inactif, rien ne préviendrait  $P_0$  du réveil de  $P_j$ . D'où lorsque  $P_i$  recevra le jeton, dès qu'il sera inactif, il transmettra le jeton dans l'état Etat\_Transitoire (en fait, cela revient à ce que  $P_i$  demande un nouveau tour). Puis il pourra repasser à Blanc.

D'où, un processus Inactif transmet le jeton d'état :

- identique à l'état du jeton reçu si il est dans l'état Blanc
- Etat\_Transitoire si il est dans l'état Noir, puis passe à l'état Blanc.



# Cas d'un graphe en anneau uni-directionnel

## L'algorithme

Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

**Détection de la terminaison** Si le jeton revient en  $P_0$   
avec l'état

- Terminé alors FIN : Terminaison Détectée.



# Cas d'un graphe en anneau uni-directionnel

## L'algorithme

Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

**Détection de la terminaison** Si le jeton revient en  $P_0$  avec l'état

- Terminé alors FIN : Terminaison Détectée.
- Etat\_Transitoire ou si  $P_0$  est actif, alors dès que  $P_0$  redevient inactif, il remet le jeton à l'état Terminé et devient Blanc. Puis il relance la détection en transmettant le jeton à  $P_1$ .



# Cas d'un arbre couvrant

## Rappel

Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

Lorsque l'on est en phase d'initialisation des calculs, seuls les pères peuvent transmettre des demandes vers les fils directs.

Pour un sommet  $P_i$  de l'arborescence, posons

$$etat_{i,local}(P_i) = \begin{cases} 1 & \text{si } P_i \text{ a fini son code} \\ 0 & \text{sinon} \end{cases} \quad (1)$$

et  $D_{P_i}$  ensemble des descendants de  $P_i$  à qui  $P_i$  a transmis une demande.

Soit  $etat_i(P_j)$  la vision qu'a le processus  $P_i$  de l'état du processus  $P_j$  alors

$$etat_i(P_i) = \begin{cases} etat_{i,local}(P_i) & \text{si } D_i = \emptyset \\ etat_{i,local}(P_i) \times \prod_{P_j \in D_i} etat_i(P_j) & \text{sinon} \end{cases} \quad (2)$$





# Cas d'un arbre couvrant

## Rappel

### Introduction

### Cas d'un graphe en anneau uni-directionnel

### Cas d'un arbre couvrant

### Cas général

Lorsque l'on est en phase de calcul : tous les calculs ont été initiés, il suffit alors que chaque processus transmette la vision de son état dès que celui-ci est égal à 1, pour qu'un noeud puisse calculer son état. D'où :

### **En phase d'initialisation**

- $etat_0(P_0) = 0$

- 
1.  $etat_i(P_i)$  a été remis à 0 à la réception  $P_i$  du message demandant un calcul



# Cas d'un arbre couvrant

## Rappel

### Introduction

### Cas d'un graphe en anneau uni-directionnel

### Cas d'un arbre couvrant

### Cas général

Lorsque l'on est en phase de calcul : tous les calculs ont été initiés, il suffit alors que chaque processus transmette la vision de son état dès que celui-ci est égal à 1, pour qu'un noeud puisse calculer son état. D'où :

### En phase d'initialisation

- $etat_0(P_0) = 0$
- chaque fois qu'un message part d'un processus  $P_i$  vers un fils  $P_j$  :

---

1.  $etat_i(P_i)$  a été remis à 0 à la réception  $P_i$  du message demandant un calcul



# Cas d'un arbre couvrant

## Rappel

### Introduction

### Cas d'un graphe en anneau uni-directionnel

### Cas d'un arbre couvrant

### Cas général

Lorsque l'on est en phase de calcul : tous les calculs ont été initiés, il suffit alors que chaque processus transmette la vision de son état dès que celui-ci est égal à 1, pour qu'un noeud puisse calculer son état. D'où :

### En phase d'initialisation

- $etat_0(P_0) = 0$
- chaque fois qu'un message part d'un processus  $P_i$  vers un fils  $P_j$  :
  - $P$  remet  $etat_i(P_i)$  à  $0^1$ ,  $D_i = D_i \cap \{j\}$  puis  $P_i$  "attend" l'état de  $P_j$  pour recalculer son propre état ;

---

1.  $etat_i(P_i)$  a été remis à 0 à la réception  $P_i$  du message demandant un calcul



# Cas d'un arbre couvrant

## Rappel

### Introduction

### Cas d'un graphe en anneau uni-directionnel

### Cas d'un arbre couvrant

### Cas général

Lorsque l'on est en phase de calcul : tous les calculs ont été initiés, il suffit alors que chaque processus transmette la vision de son état dès que celui-ci est égal à 1, pour qu'un noeud puisse calculer son état. D'où :

### En phase d'initialisation

- $etat_0(P_0) = 0$
- chaque fois qu'un message part d'un processus  $P_i$  vers un fils  $P_j$  :
  - $P$  remet  $etat_i(P_i)$  à  $0^1$ ,  $D_i = D_i \cap \{j\}$  puis  $P_i$  "attend" l'état de  $P_j$  pour recalculer son propre état ;
  - $P_j$  remet  $etat_j$ ,  $local(P_j)$  et  $etat_j(P_j)$  à 0, passe à l'état Actif et exécute son code.

---

1.  $etat_i(P_i)$  a été remis à 0 à la réception  $P_i$  du message demandant un calcul



# Cas d'un arbre couvrant

## Rappel

### Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

## En phase de détection

- dès que  $etat_{i \neq 0}(P_{i \neq 0}) = 1$ ,  $P_i$  transmet 1 à son père



# Cas d'un arbre couvrant

## Rappel

### Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

## En phase de détection

- dès que  $etat_{i \neq 0}(P_{i \neq 0}) = 1$ ,  $P_i$  transmet 1 à son père
- dès que  $etat_0(P_0) = 1$ , l'application est terminée.



# Cas général

## Introduction

### Cas d'un graphe en anneau uni-directionnel

### Cas d'un arbre couvrant

### Cas général

**Algorithme de Misra (1983)** On suppose que le le graphe de communication est fortement connexe (c'est-à-dire que tout processus  $P_i$  peut envoyer un message à tout  $P_{0 < j \neq i \leq n}$  : pas nécessairement directement mais en passant éventuellement en passant d'autres processus). Dans ce cas, la théorie des graphes nous assure qu'il existe un circuit<sup>2</sup>  $C$  qui comprend chaque arc (au moins une fois) du graphe du réseau.

---

2. ce circuit n'est pas toujours évident à trouver avec un algorithme réparti



# Cas général

## Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

**Algorithme de Misra (1983)** On va utiliser un jeton pour trouver le nombre  $nb$  de processus visités suivant le circuit  $C$  qui soient restés inactifs entre deux passages de ce jeton.

Il est donc évident que l'application sera terminée lorsque  $nb = \text{taille}(C)$ , où  $\text{taille}(C)$  est la taille du circuit en nombre de visites de processus (peut être supérieur à  $n$  car on peut passer plusieurs fois par un même processus).





# Cas général

Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

## Algorithme de Misra (1983)

Posons :

- $\text{succ}(P_i)$  successeur de  $P_i$  donné par le circuit<sup>3</sup> C

---

3. que l'on suppose construit



## Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

## Algorithme de Misra (1983)

Posons :

- $\text{succ}(P_i)$  successeur de  $P_i$  donné par le circuit<sup>3</sup> C
- Pour chaque processus  $P_i$  les quatre variables locales suivantes :

---

3. que l'on suppose construit



## Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

## Algorithme de Misra (1983)

Posons :

- $\text{succ}(P_i)$  successeur de  $P_i$  donné par le circuit<sup>3</sup>  $C$
- Pour chaque processus  $P_i$  les quatre variables locales suivantes :
  - $\text{couleur}_i = \text{Blanc ou Noir initialisé à Blanc}$

---

3. que l'on suppose construit



# Cas général

## Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

## Algorithme de Misra (1983)

Posons :

- $\text{succ}(P_i)$  successeur de  $P_i$  donné par le circuit<sup>3</sup>  $C$
- Pour chaque processus  $P_i$  les quatre variables locales suivantes :
  - $\text{couleur}_i = \text{Blanc ou Noir}$  initialisé à Blanc
  - $\text{etat}_i = \text{Actif ou Inactif}$  initialisé à Actif

---

3. que l'on suppose construit



## Algorithme de Misra (1983)

Posons :

- $\text{succ}(P_i)$  successeur de  $P_i$  donné par le circuit<sup>3</sup> C
- Pour chaque processus  $P_i$  les quatre variables locales suivantes :
  - $\text{couleur}_i$  = Blanc ou Noir initialisé à Blanc
  - $\text{etat}_i$  = Actif ou Inactif initialisé à Actif
  - $\text{jeton\_present}_i$  = Vrai ou Faux initialisé à Faux sauf pour  $P_k$ ,  $k$  choisi aléatoirement

---

3. que l'on suppose construit



## Algorithme de Misra (1983)

Posons :

- $\text{succ}(P_i)$  successeur de  $P_i$  donné par le circuit<sup>3</sup> C
- Pour chaque processus  $P_i$  les quatre variables locales suivantes :
  - $\text{couleur}_i$  = Blanc ou Noir initialisé à Blanc
  - $\text{etat}_i$  = Actif ou Inactif initialisé à Actif
  - $\text{jeton\_present}_i$  = Vrai ou Faux initialisé à Faux sauf pour  $P_k$ , k choisi aléatoirement
  - $\text{nb}_i$  = entier initialisé à 0

---

3. que l'on suppose construit



# Cas général

Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

## Etat d'un processus

Lorsque  $P_i$  reçoit un message, il passe à l'état actif par :

$etat_i = \text{Actif}$

Chaque fois qu'il envoie un message, il mémorise ce fait par :  $couleur_i = \text{Noir}$ . Ainsi, indirectement, il mémorise qu'il est possible qu'il ait réveillé un processus.

Lorsqu'il a fini de calculer, il passe à l'état inactif par :  $etat_i = \text{Inactif}$

Lorsqu'il reçoit le jeton, il effectue :  $nb_i = nb$  et  $jeton\_present_i = \text{Vrai}$  puis

s'il est inactif :

- soit il détecte la terminaison : FIN

s'il est Actif : il ne le transmettra que lorsqu'il redeviendra inactif.



# Cas général

Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

## Etat d'un processus

Lorsque  $P_i$  reçoit un message, il passe à l'état actif par :

$etat_i = \text{Actif}$

Chaque fois qu'il envoie un message, il mémorise ce fait par :  $couleur_i = \text{Noir}$ . Ainsi, indirectement, il mémorise qu'il est possible qu'il ait réveillé un processus.

Lorsqu'il a fini de calculer, il passe à l'état inactif par :  $etat_i = \text{Inactif}$

Lorsqu'il reçoit le jeton, il effectue :  $nb_i = nb$  et  $jeton\_present_i = \text{Vrai}$  puis

s'il est inactif :

- soit il détecte la terminaison : FIN
- soit il doit retransmettre le jeton

s'il est Actif : il ne le transmettra que lorsqu'il redeviendra inactif.





# Cas général

Introduction

Cas d'un graphe  
en anneau  
uni-directionnel

Cas d'un arbre  
couvrant

Cas général

## Transmission du jeton

Si  $P_i$  est Inactif (ou si  $P_i$  passe à Inactif) avec  
 $\text{jeton\_present}_i = \text{Vrai}$  alors /\*  $P_i$  transmet le jeton \*/ : si  
 $\text{couleur}_i = \text{Noir}$ <sup>4</sup>

alors /\* on recommence la détection \*/

$\text{nb}_i = 1$

sinon /\* on incrémente le nombre de processus inactifs  
déjà visité \*/

$\text{nb}_i = \text{nb}_i + 1$

$\text{couleur}_i = \text{Blanc}$   $\text{jeton\_present}_i = \text{Faux}$

$\text{envoyer}(\text{jeton}, \text{nb} = \text{nb}_i)$  à  $\text{succ}(P_i)$

## Détection de la terminaison

si  $\text{nb} = \text{taille}(C)$  et  $\text{couleur}_i = \text{Blanc}$  alors Terminaison  
Détectée

---

4. il a donc envoyé un message pendant sa phase d'activité

