**Université de Strasbourg**
**UFR Mathématique et Informatique**

**M1 Informatique**
**2021/2022**

# Algorithmes Distribués

## TD n° 4 : Terminaison

---

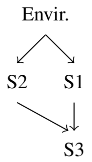
**CORRECTION**

---

**<u>Exercice 1</u> : Algorithme de détection de terminaison de Dijkstra-Scholten**

Le code de l'algorithme de calcul diffusant de Dijkstra et Scholten est rappelé en annexe 1.

**Q 1.** Complétez le tableau des valeurs des variables `père`, `defin`, `defout` des processus env, S1, S2 et S3 dans le scénario ci-dessous :




| | | env | S1 | S2 | S3 |
|---|---|---|---|---|---|
| | val. init. | - 0 0 | - 0 0 | - 0 0 | - 0 0 |
| | env. active s1 | ... | ... | ... | ... |
| | s1 active s3 | ... | ... | ... | ... |
| pas de changement, car $\text{defout}_1 \neq 0$ | calcul s1 terminé | ... | ... | ... | ... |
| | env. active s2 | ... | ... | ... | ... |
| | s2 active s3 | ... | ... | ... | ... |
| s3 envoie signal à s2 | calcul s3 terminé | ... | ... | ... | ... |
| s2 ne propage pas (non terminé) | | ... | ... | ... | ... |
| s3 envoie signal à s1 | calcul s3 terminé | ... | ... | ... | ... |
| s1 envoie signal à e (terminé) | | ... | ... | ... | ... |
| S2 envoie signal à e | calcul s2 terminé | ... | ... | ... | ... |

| | | env | S1 | S2 | S3 |
|---|---|---|---|---|---|
| | val. init. | - 0 0 | - 0 0 | - 0 0 | - 0 0 |
| | env. active s1 | .01 | e1. | ... | ... |
| | s1 active s3 | ... | .11 | ... | 11. |
| pas de changement, car $\text{defout}_1 \neq 0$ | calcul s1 terminé | ... | ... | ... | ... |
| | env. active s2 | .02 | ... | e1. | ... |
| | s2 active s3 | ... | ... | e11 | 12. |
| s3 envoie signal à s2 | calcul s3 terminé | ... | e11 | e10 | 11. |
| s2 ne propage pas (non terminé) | | .02 | e11 | e10 | 11. |
| s3 envoie signal à s1 | calcul s3 terminé | .02 | e10 | e10 | .0. |
| s1 envoie signal à e (terminé) | | .01 | .00 | .10 | ... |
| S2 envoie signal à e | calcul s2 terminé | .00 | .0. | .00 | ... |

**Q 2.** Montrez que lorsque le calcul diffusant se termine, alors le processus environnement revient à son état initial (defin = defout = 0).

Lorsque le calcul diffusant est terminé il n'y a plus d'activité et ni messages, ni signaux ne circulent. De plus les processus ne peuvent plus envoyer de signaux. On en conclut que pour tous les processus $P_i$ différents de l'environnement, on a :

$$defin \geq 0 \wedge defout \geq 0 \wedge \neg(defin > 1 \vee (defin = 1 \wedge defout = 0)$$

ce qui se simplifie en : $defin = 0 \vee (defin = 1 \wedge defout > 0)$
En effet, on a :

$$\neg(defin > 1) \Rightarrow defin = 1 \vee defin = 0 \; (1)$$

$$\neg(defin = 1 \wedge defout = 0) \Rightarrow defin \neq 1 \vee defout \neq 0 \ (2)$$

$$(1) \wedge (2) \Rightarrow defin = 0 \vee defin = 1 \wedge defout > 0$$

Car $defout \neq 0 \Rightarrow defout > 0$

De plus, pour le processus environnement (racine de l'arbre de recouvrement) qui n'a pas de prédécesseurs on a : $defin = 0$ et $defout \geq 0$.
On en conclut que pour tous les processus $P_i$ on a lorsque le calcul est terminé :

$$\forall P_i, defin_i \leq defout_i (i)$$

Or, comme il n'y a ni messages, ni signaux en transit on a :

$$\sum defin_i = \sum defout_i (ii)$$

Des relations $(i)$ et $(ii)$ on déduit que lorsque le calcul est terminé :

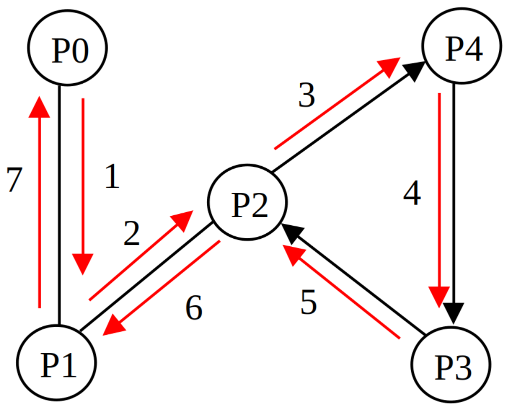
$$\forall P_i, defin_i = defout_i$$

et comme pour le processus environnement defin = 0 on a aLors defout = 0 On en conclut donc que si le calcul diffusant se termine, le processus environnement reviendra à l'état initial (defin = defout = 0).

Remarque : la réciproque prouve qu'il n'y a pas de fausses terminaisons, i.e que lorsque l'environnement retourne à son état initial, le calcul diffusant s'est terminé. En conclusion, le principe du calcul diffusant et son contrôle par une structure arborescente ont été appliqués à d'autres problèmes tels que la détection d'interblocage (algo de Chandy, Misra et Haas vu au chapitre précédent).

**Exercice 2 : Algorithme de détection de terminaison de Misra**

L'algorithme de détection de terminaison de Misra (sur un réseau quelconque) est fourni en Annexe 2.
**Q 1.** Appliquez l'algorithme de Misra au scénario suivant :




Initialement P0, P1, P2 sont passifs et P3, P4 sont actifs.

1. P0 lance une détection de terminaison
   couleur ← ............... ; nb ← ....... ; jeton_présent ← .......... ;
   envoi (jeton, 0) à .......
2. P3 envoie un message à P2
3. ....... reçoit (jeton, 0) de P0
   jeton_présent ← vrai
   Comme état = passif on exécute l'émission du jeton
   nb ← ....... ; (car couleur = noir)
   envoi (jeton, .......) à ..........
   couleur ← ................ ; jeton_présent ← ................ ;

4. ......... reçoit (jeton, .......)
   jeton_présent ← vrai
   Comme état = passif on exécute l'émission du jeton
   nb ← ....... ;
   envoi (jeton, .......) à .......
   couleur ← ............... ; jeton_présent ← ............... ;
5. P2 reçoit le message envoyé par P3
   etat ← ............... ; couleur ← ............. ;
6. ....... reçoit (jeton, .......)
   jeton_présent ← .......
   ....... attends la condition (état = passif)
   Au bout d'un temps fini état ← passif
   Comme état = passif on exécute l'émission du jeton
   nb ← ....... ;
   envoi (jeton, .......) à .......
   couleur ← ............... ; jeton_présent ← ............. ;
7. ....... reçoit (jeton, .......)
   jeton_présent ←
   Le processus attends la condition (état = passif)
   Au bout d'un temps fini état ← passif
   Comme état = passif on exécute l'émission du jeton
   nb ← ....... ;
   envoi (jeton, .......) à .......
   couleur ← ............... ; jeton_présent ← ............... ;
8. ....... reçoit (jeton, .......)
   jeton_présent ← vrai
   Le processus attends la condition (état = passif)
   Au bout d'un temps fini état ← passif
   Comme état = passif on exécute l'émission du jeton
   nb ← ....... ;
   envoi (jeton, .......) à .......
   couleur ← ............... ; jeton_présent ← ............... ;
9. ....... reçoit (jeton, .......)
   jeton_présent ← vrai
   Comme état = passif on exécute l'émission du jeton
   nb ← ....... ;
   envoi (jeton, .......) à ....... couleur ← ............... ; jeton_présent ← ............... ;
10. ....... reçoit (jeton, .......)
    jeton_présent ← vrai
    Comme état = passif on exécute l'émission du jeton
    nb ← ....... ;
    envoi (jeton, .......) à .......
    couleur ← ............... ; jeton_présent ← ............... ;
11. ....... reçoit (jeton, .......)
    jeton_présent ← vrai
    Comme état = passif on exécute l'émission du jeton
    nb ← ....... ;
    envoi (jeton, .......) à ............
    couleur ← ............... ; jeton_présent ← ............... ;
12. l'algorithme continue de la même manière (tous les processus sont passifs)
    - ....... reçoit (jeton, .......) et envoie (jeton, .......) à .......
    - ....... reçoit (jeton, .......) et envoie (jeton, .......) à .......
    - ....... reçoit (jeton, .......) et envoie (jeton, .......) à .......
    - ....... reçoit (jeton, .......) et envoie (jeton, .......) à .......
    - ....... reçoit (jeton, .......) = taille (C) et couleur = blanc => terminaison détectée

Correction:

1. P0 lance une détection de terminaison
   couleur ← blanc ; nb ← 0 ; jeton_présent ← faux ;
   envoi (jeton, 0) à P1
2. P3 envoie un message à P2
3. P1 reçoit (jeton, 0) de P0
   jeton_présent ← vrai
   Comme état = passif on exécute l'émission du jeton
   nb ← 0 ; (car couleur = noir)
   envoi (jeton, 0) à P2
   couleur ← blanc ; jeton_présent ← faux ;
4. P2 reçoit (jeton, 0)
   jeton_présent ← vrai
   Comme état = passif on exécute l'émission du jeton
   nb ← 0 ;
   envoi (jeton, 0) à P4
   couleur ← blanc ; jeton_présent ← faux ;
5. P2 reçoit le message envoyé par P3
   etat ← actif ; couleur ← noir ;
6. P4 reçoit (jeton, 0)
   jeton_présent ← vrai
   P4 attends la condition (état = passif)
   Au bout d'un temps fini état ← passif
   Comme état = passif on exécute l'émission du jeton
   nb ← 0 ;
   envoi (jeton, 0) à P3
   couleur ← blanc ; jeton_présent ← faux ;
7. P3 reçoit (jeton, 0)
   jeton_présent ← vrai Le processus attends la condition (état = passif)
   Au bout d'un temps fini état ← passif
   Comme état = passif on exécute l'émission du jeton
   nb ← 0 ;
   envoi (jeton, 0) à P2
   couleur ← blanc ; jeton_présent ← faux ;
8. P2 reçoit (jeton, 0)
   jeton_présent ← vrai
   Le processus attends la condition (état = passif)
   Au bout d'un temps fini état ← passif
   Comme état = passif on exécute l'émission du jeton
   nb ← 0 ;
   envoi (jeton, 0) à P1
   couleur ← blanc ; jeton_présent ← faux ;
9. P1 reçoit (jeton, 0)
   jeton_présent ← vrai
   Comme état = passif on exécute l'émission du jeton
   nb ← 1 ;
   envoi (jeton, 1) à P0 couleur ← blanc ; jeton_présent ← faux ;
10. P0 reçoit (jeton, 1)
    jeton_présent ← vrai
    Comme état = passif on exécute l'émission du jeton
    nb ← 2 ;
    envoi (jeton, 2) à P1
    couleur ← blanc ; jeton_présent ← faux ;

11. P1 reçoit (jeton, 2)
    jeton_présent ← vrai
    Comme état = passif on exécute l'émission du jeton
    nb ← 3 ;
    envoi (jeton, 3) à P2
    couleur ← blanc ; jeton_présent ← faux ;
12. l'algorithme continue de la même manière (tous les processus sont passifs)
    - P2 reçoit (jeton, 3) et envoie (jeton, 4) à P4
    - P4 reçoit (jeton, 4) et envoie (jeton, 5) à P3
    - P3 reçoit (jeton, 5) et envoie (jeton, 6) à P2
    - P2 reçoit (jeton, 6) et envoie (jeton, 7) à P1
    - P1 reçoit (jeton, 7) = taille (C) et couleur = blanc => terminaison détectée

## Annexe 1 : algorithme de détection de terminaison de Dikkstra et Scholten

Pour chaque processus $P_i$ deux variables entières sont définies, defin et defout. Elles comptent respectivement la somme des déficits des arcs entrant vers le processus et la somme des déficits des arcs sortant. Initialement ces deux compteurs sont nuls.

$P_i$ gère ses compteurs defin et defout de la façon suivante :

— envoi d'un message : defout ← defout + 1 ;
— envoi d'un message : defout ← defout + 1 ;
— envoi d'un signal : defin ← defin – 1 ;
— réception d'un signal : defout ← defout – 1.

Outre les compteurs defin et defout (entiers entre 0 et nbmax, initialisés à 0), le processus $P_i$ est doté de deux variables :

— père (entier de 1 à n) : qui représente son père dans l'arbre de recouvrement du graphe des processus (celui-ci est déterminé par la première communication qu'a le processus) ;
— appelants (sac de 1 à n) : ensemble dans lequel le processus mémorise les noms des processus autres que le père qui lui ont envoyé un message.

Algorithme :

```
Lors de réception de (message, expd) de expd ;
début
  si defin = 0 alors
    père ← expd ;
  sinon
    appelants ← appelants ⊕ expd ;
  fin
  defin ← defin + 1 ;
fin

Lors de réception de (signal, expd) de expd ;
début
  defout ← defout – 1 ;
fin

Lors de désir d'émettre (message, i) vers j ;
début
  si defin ≠ 0 alors
    defout ← defout + 1 ;
    envoyer (message, i) à j ;
  fin
fin
```

Lors de désir d'émettre (signal) ;
**début**
  **si** (defin > 1) ou (defin = 1 et defout = 0) **alors**
    **si** defin = 1 **alors**
      /* envoi du dernier signal : defout = 0 */ envoyer (signal, i) à père ;
    **sinon**
      aux ← un élément de appelants ;
      appelants ← appelants \ {aux} ;
      envoyer (signal, i) à aux ;
    **fin**
    defin ← defin – 1 ;
  **fin**
**fin**

## Annexe 2 : algorithme de détection de terminaison de Misra sur un réseau de topologie quelconque

Hypothèses :

- pas de perte des messages;
- pas de déséquencement (canaux FIFO).

On suppose que sur le circuit C (pré-calculé) les deux fonctions suivantes sont définies :

- fonction taille (C : circuit) résultat entier : donne la taille du circuit;
- fonction successeur (C : circuit, i : 1 .. n) résultat 1 .. n : donne pour le processus $P_i$ qui l'exécute, le numéro du successeur de $P_i$ sur le circuit.

Chaque processus $P_i$ est doté des variables suivantes :

- couleur : (blanc, noir) initialisé à noir;
- état : (actif, passif) initialisé à actif;
- $jeton_{present}$ : booléen initialisé à faux;
- nb : entier initialisé à 0.

La variable couleur est associée au processus et nb sert à mémoriser la valeur associée au jeton entre sa réception et sa réémission. Les messages sont de deux types : messages, jeton.

Algorithme :

Lors de réception de (message, m) ;
**début**
  état ← actif ;
  couleur ← noir ;
**fin**

Lors de attente (message, m) ;
**début**
  état ←passif;
**fin**

Lors de réception de (jeton, j) ;
**début**
  nb ←j ;
  $jeton_{present}$ ←vrai;
  **si** nb = taille (C) et couleur = blanc **alors**
    terminaison détectée ;
  **fin**
**fin**

Lors de émission de (jeton, j) ;
**début**
    **si** $jeton_{present}$ et état = passif **alors**
        **si** couleur = noir **alors**
            nb ← 0;
        **fin**
        **sinon**
            nb ← nb + 1;
        **fin**
        envoyer (jeton, nb) à successeur (C, i) ;
        couleur ← blanc ;
        $jeton_{present}$ ← faux ;
    **fin**
**fin**