SCIENCES SORBONNE UNIVERSITÉ

Détection Répartie de la Terminaison

Master informatique SU 2017-2018
UE AR (4I403)

Plan

- **Définition du Problème**
  - Exemple de mauvais algorithme
- **Exemple d'algorithmes**
  - Algorithme de Misra [1983]
  - Modèle à communication instantanée
    - Algorithme de Rana[1983]
    - Algorithme de Dijkstra [1983]
  - Modèle atomique :
    - Algorithme des quatre compteurs (Mattern [1987])

Détection Répartie de la Terminaison

- **Construction d'une couche de contrôle afin de détecter la terminaison d'une application répartie.**
  - Distinguer l'algorithme de détection de terminaison de l'algorithme de l'application.
    - Pas d'influence dans l'exécution de l'application
- **Configuration terminale**
  - aucune action supplémentaire de l'application ne peut être exécutée
  - Tous les canaux de communication sont vides

Détection Répartie de la Terminaison

- **État**
  - ***actif*** : si une action interne ou l'action *émettre()* est applicable
  - ***passif***
    - Dans le cas contraire
- **Message**
  - ***Applicatif ("basic message"):***
    - Message de l'application
  - ***Contrôle***
    - Message de l'algorithme de détection de la terminaison.

Détection Répartie de la Terminaison

- **Un modèle est défini pour une exécution répartie en définissant les actions des processus *actifs* et *passifs*.**

- **Les processus suivent les règles suivantes:**

  1. Initialement, chaque processus ***p*** peut être dans l'état ***actif*** ou ***passif***
  2. Un processus ***p*** peut passer spontanément de l'état ***actif*** à ***passif***.
  3. Seuls les processus ***actifs*** peuvent envoyer des massages applicatifs.
  4. Lors de la réception d'un message applicatif, un processus ***p passif*** passe à ***actif***.

     - Seule façon pour un processus ***passif*** de passer à ***actif***.

  - Observations :
    - Un message de contrôle ***émis*** lorsque le processus est *passif* ne le rend pas *actif*.
    - La ***réception*** d'un message de contrôle par un processus *passif* ne le rend pas *actif*.

Détection Répartie de la Terminaison

- **Terminaison**
  - Π : ensemble de processus
  - C : ensemble de canaux
  - Prédicat ***TERM*** :
    - **TERM <=> (∀ p ϵ Π : p passif) et (∀ c ϵ C: c vide)**
      - ***TERM*** est un prédicat stable :
        - TERM (t) = true => ∀ t' > t : TERM (t') = true

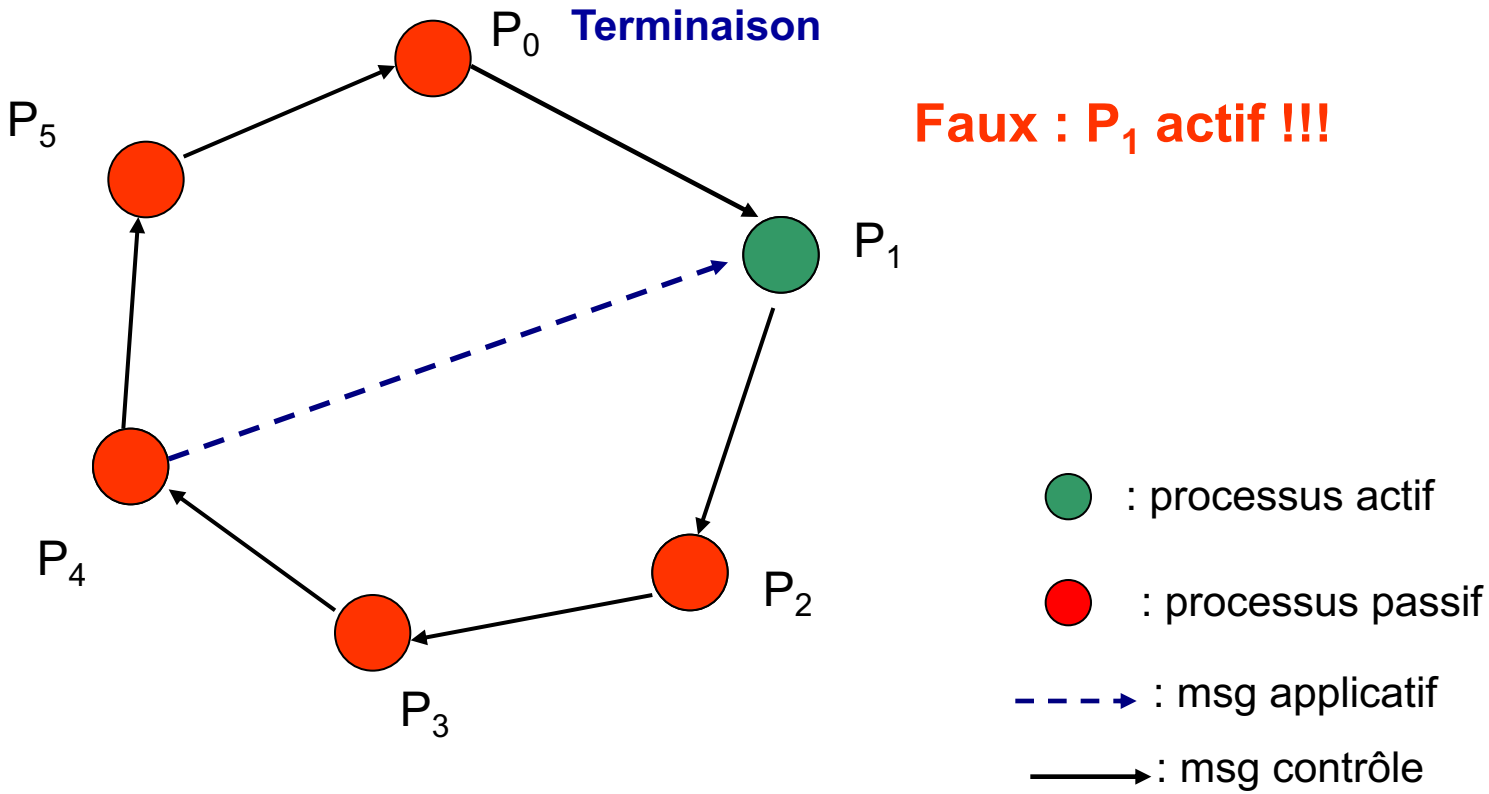
Détection Répartie de la Terminaison

- **Propriétés :**
  - ***Sûreté*** *:*
    - Si un processus détecte la terminaison à l'instant *t*, alors *TERM(t) = true*
      - Pas de fausse détection
  - ***Vivacité*** *:*
    - Si à un instant *t*, *TERM(t) = true*, alors l'algorithme de détection finira par détecter cette terminaison.

Détection Répartie de la Terminaison

- **Exemple d'un mauvais algorithme de détection répartie de la terminaison**
  - Les sites se trouvent soit dans l'état ***passif*** soit dans l'état ***actif***
  - Algorithme :
    - Faire circuler un jeton (message de contrôle) selon une structure d'anneau, envoyé initialement par P_0.
    - Lorsqu'un site est *passif* et possède le jeton, il l'envoie au site suivant.
    - Lorsque le jeton revient à P_0, la terminaison est détectée.

Mauvais algorithme de détection de la terminaison



Terminaison sur un anneau

- **Algorithme de Misra**
  - Anneau logique
    - Canaux FIFO unidirectionnels.
  - Chaque site une couleur *noir* ou *blanc*.
    - *noir =actif*
    - *blanc = passif*
  - Jeton porte un compteur
    - Nombre de sites trouvés *passif* par le jeton.
  - Terminaison détectée : tous les sites sont blancs après un tour.

Algorithme de Misra

N sites

init:
state =actif
color = black
if (i==0)
token = true
else
token = false;

Upon fin:
state =passif

Upon reception application msg:
etat =actif
color = black

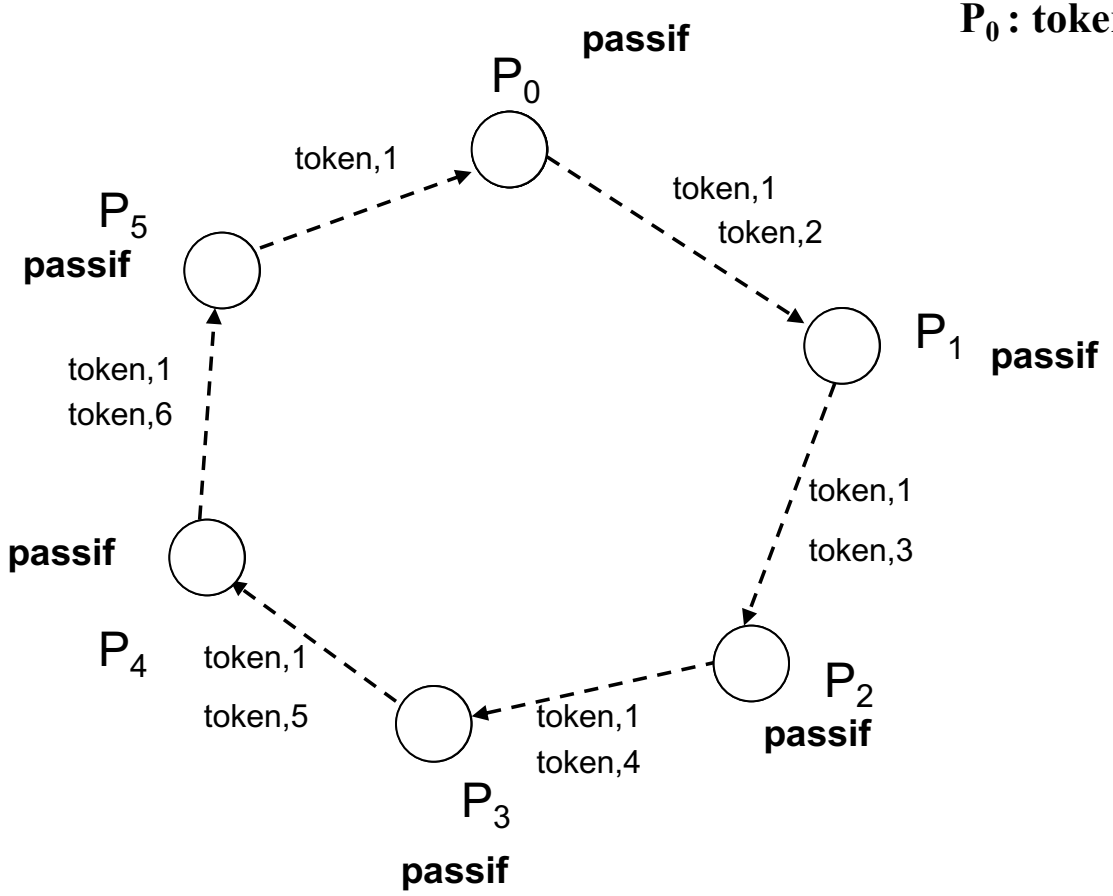
Upon reception TOKEN (count)
token = true;
Nb=count;
if ((Nb== N) and (color== white))
termination detection;

Upon (token== true) **and** (state==passif)
if (color == white)
send (TOKEN, Nb+1)
else
send (TOKEN,1):
color = white;
token = false;

Algorithme de Misra

N: 6
P_0 : token

Terminaison detectée



Détection Répartie de la Terminaison

- **Modèles afin de simplifier le problème :**
  - ***A communication instantanée :***
    - Communication synchrone : exemple CSP
    - **TERM <=> (∀ p ϵ Π: p passif)**
  - ***Atomique*** :
    - Le moment d'activité des processus est négligeable.
      - **TERM <=> (∀ c ϵ C: c vide)**

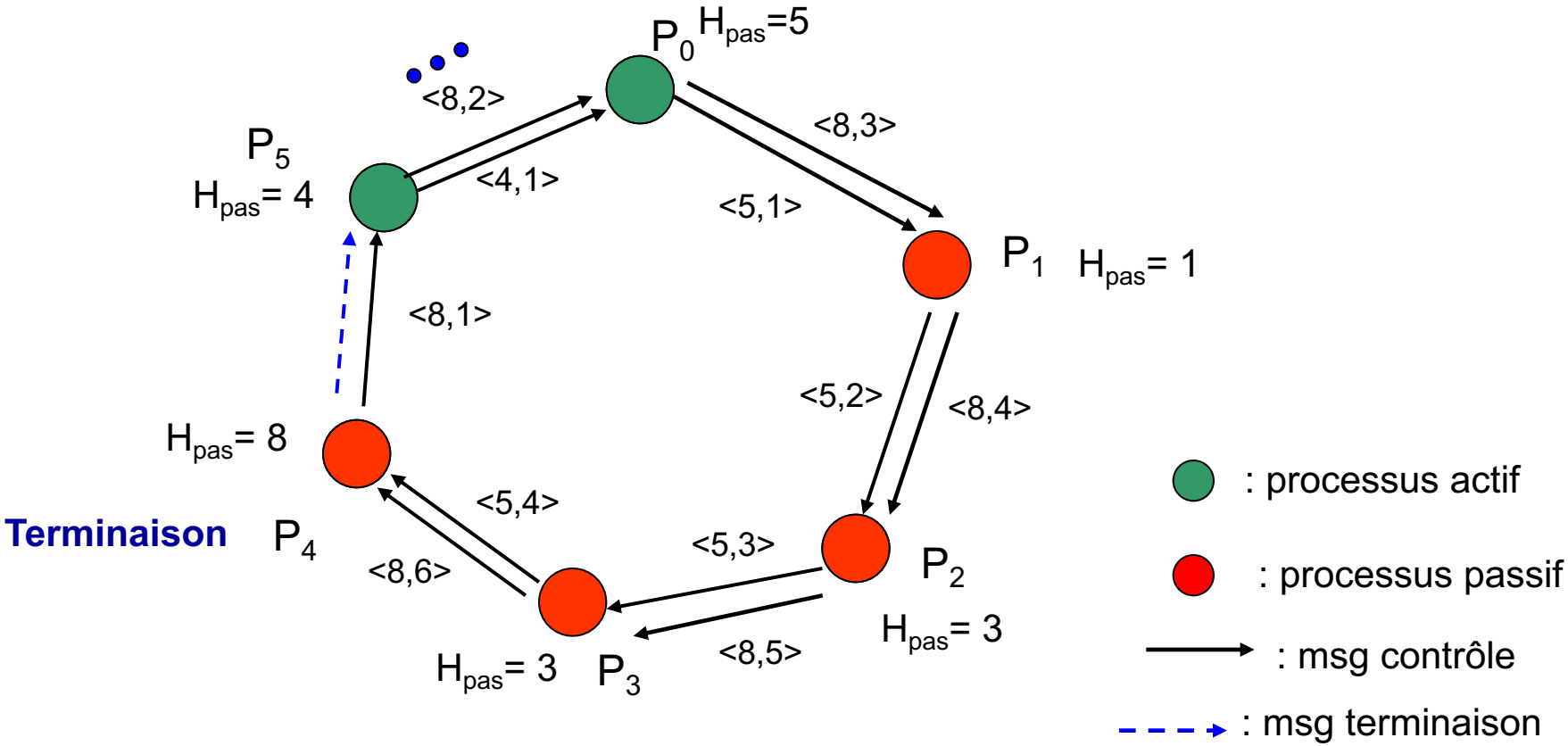
Modèle à communication instantanée

- **Algorithme de Rana [1983]**
  - Communication instantanée (e.g. CSP)
  - *N* sites organisés dans un anneau logique unidirectionnel.
    - Messages transmis sur l'anneau.
  - A chaque fois qu'un processus reçoit soit un message *applicatif* soit un message de *contrôle*, il met son ***horloge logique locale*** à jour.
  - Les messages de contrôles circulent sur l'anneau.
    - Message de contrôle: <*H, compteur*>
    - Chaque site envoie le message de contrôle à son successeur et le reçoit de son prédécesseur;
  - ***Observation*** : Huang [1988] a étendu l'algorithme de Rana
    - TD terminaison

Algorithme de Rana

- **Lorsqu'un processus devient passif, il enregistre la valeur de son horloge locale(H_{pas}) et envoie le message de contrôle $< H_{pas}, 1 >$ à son successeur;**
- **Lors de la réception d'un message de contrôle :**
  - Si le site est actif, il ignore le message;
  - Sinon
    - Si (compteur !=N)
      - Si la valeur de son passage à passif H_{pas} $> H_{msg}$ du message de contrôle reçu, le message est ignoré;
      - Sinon, le message est envoyé à son successeur avec le compteur incrémenté $< H_{pas}, compteur+1 >$;
    - Sinon
      - Terminaison détectée.
      - Le site envoie à son successeur un message de terminaison; Le message fera le tour de l'anneau.

Algorithme de Rana



Modèle à communication instantanée

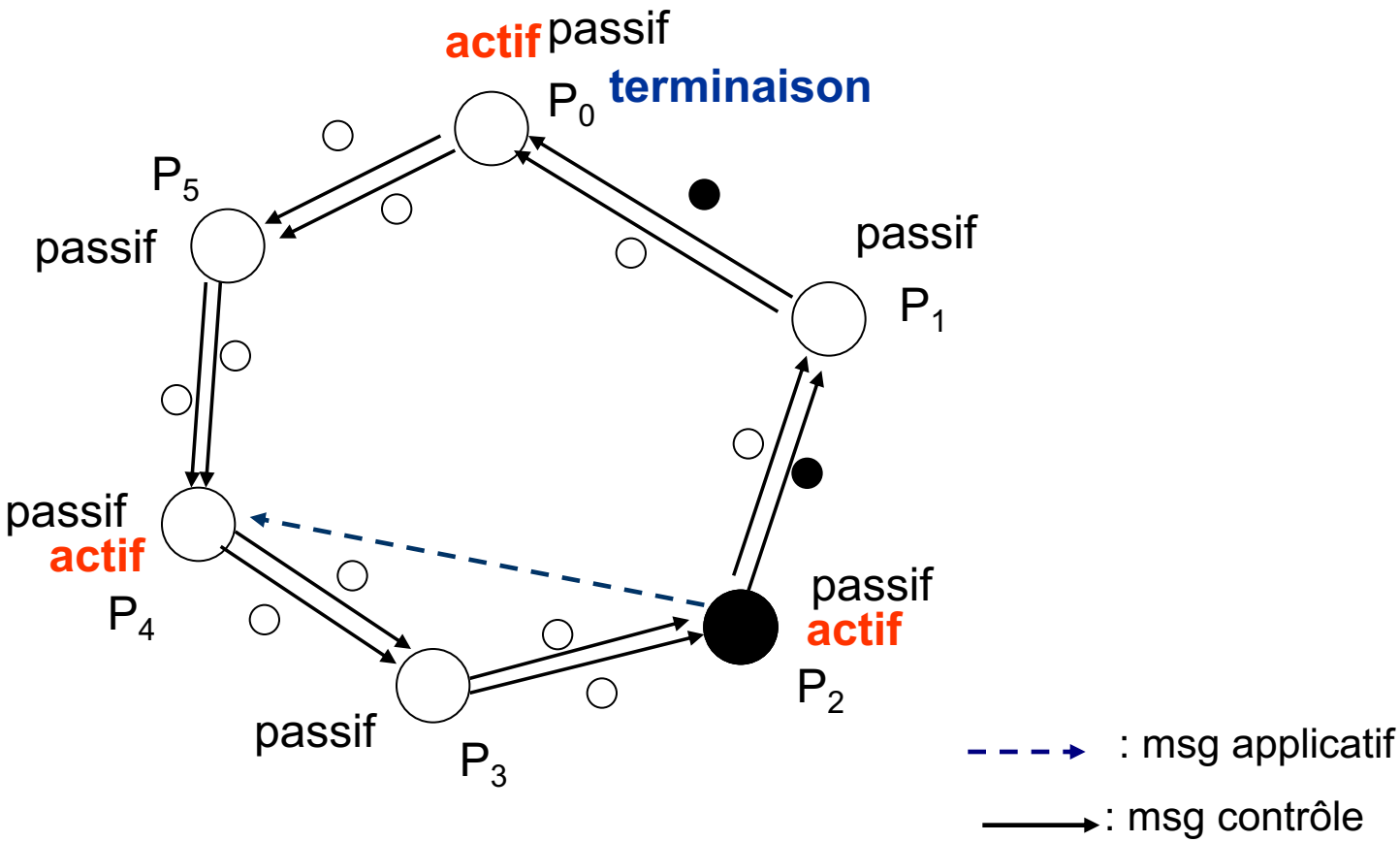
- Algorithme de **Dijkstra** [1983]
  - Modèle à communication instantanée
  - *N* sites organisés dans un anneau logique.
  - Existence d'un jeton
  - Les sites peuvent être de couleur blanche ou noire ainsi que le jeton.
    - Initialement tous les sites et le jeton sont blancs.

Algorithme de Dijkstra

- **Il y a un site initiateur P_0.**
  - Quand P_0 devient passif, il envoie le jeton couleur blanche à P_{N-1}.
- **Lorsque le site P_i, qui détient le jeton, devient *passif*, P_i envoie le jeton au site P_{i-1} :**
  - Si P_i est blanc :
    - P_i envoie à P_{i-1} le jeton sans changer la couleur du jeton ;
  - Sinon,
    - P_i change la couleur du jeton à noire avant de l'envoyer à P_{i-1}.
    - P_i devient blanc ;
- **Un site P_i devient noire en envoyant un message *applicatif* au site P_j.**
- **Lorsque P_0 reçoit le jeton :**
  - Si le ***jeton*** est ***blanc*** et P_0 est ***blanc*** et dans l'état ***passif***
    - **terminaison détectée**
  - Sinon
    - lorsque P_0 devient ***passif***, il renvoie le jeton couleur blanche à P_{N-1}.

Détection Répartie de la Terminaison

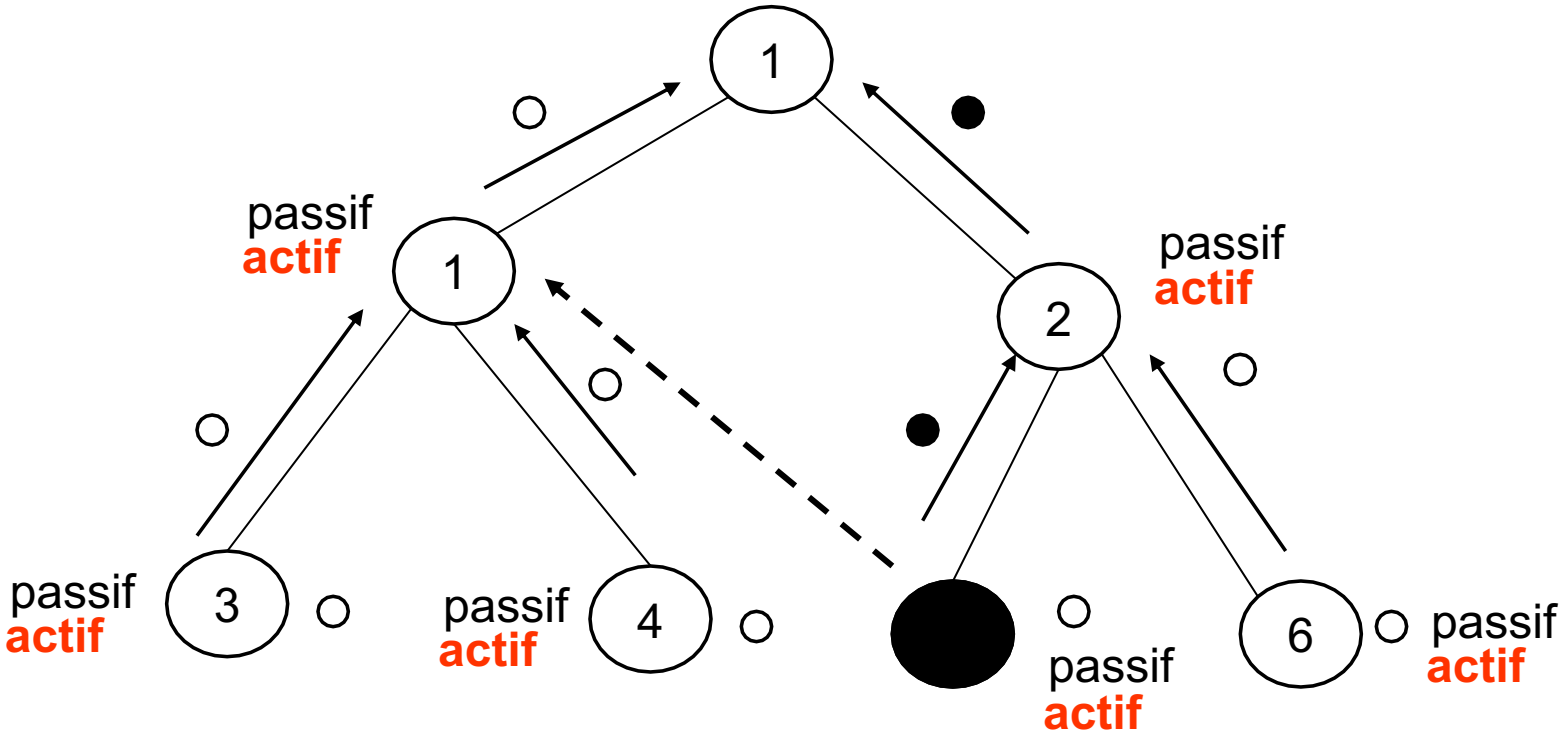
Algorithme de Dijkstra



Algorithme de Djkstra en utilisant un arbre couvrant

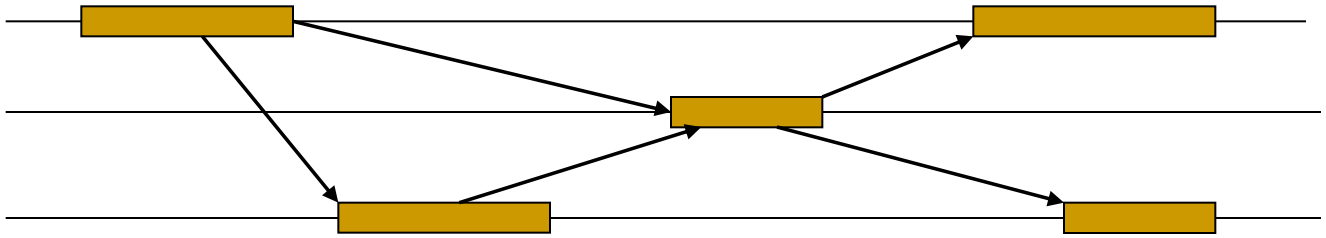
- **Racine informe aux feuilles de commencer la détection;**
  - Chaque feuille a un jeton blanc;
  - Un site P_i devient noire en envoyant un message *applicatif* au site P_j.
  - Si P_i est noir:
    - P_i change la couleur du jeton à noire avant de l'envoyer à *son père*
    - P_i devient blanc ;
  - Si P_i a reçu un jeton noir d'un de ses enfants, il envoie un jeton noir à son père;
  - La racine conclut que l'application a terminé si:
    - Sa couleur est blanche;
    - Elle est dans l'état passif
    - Elle a reçu un jeton blanc de tous ses enfant
  - Sinon
    - Informe aux feuilles de recommencer la détection

Algorithme de Djkstra en utilisant un arbre couvrant

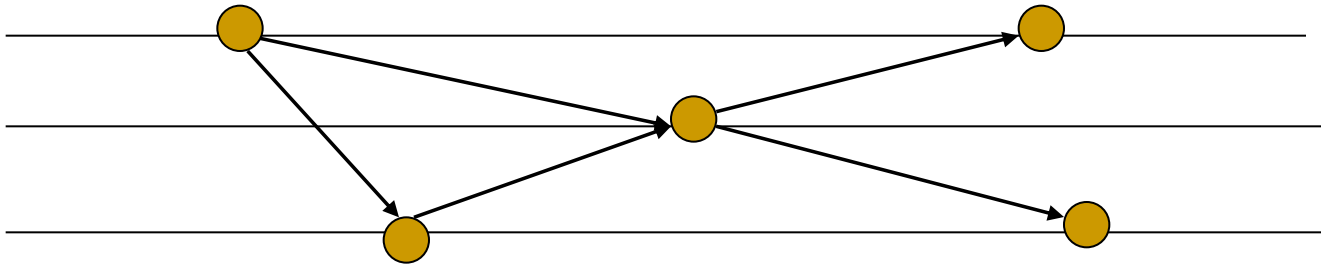


Modèle atomique

- **L'algorithme de détection ne "voit" jamais un processus local dans l'état actif : l'algorithme n'est activé que lorsque le processus est passif**

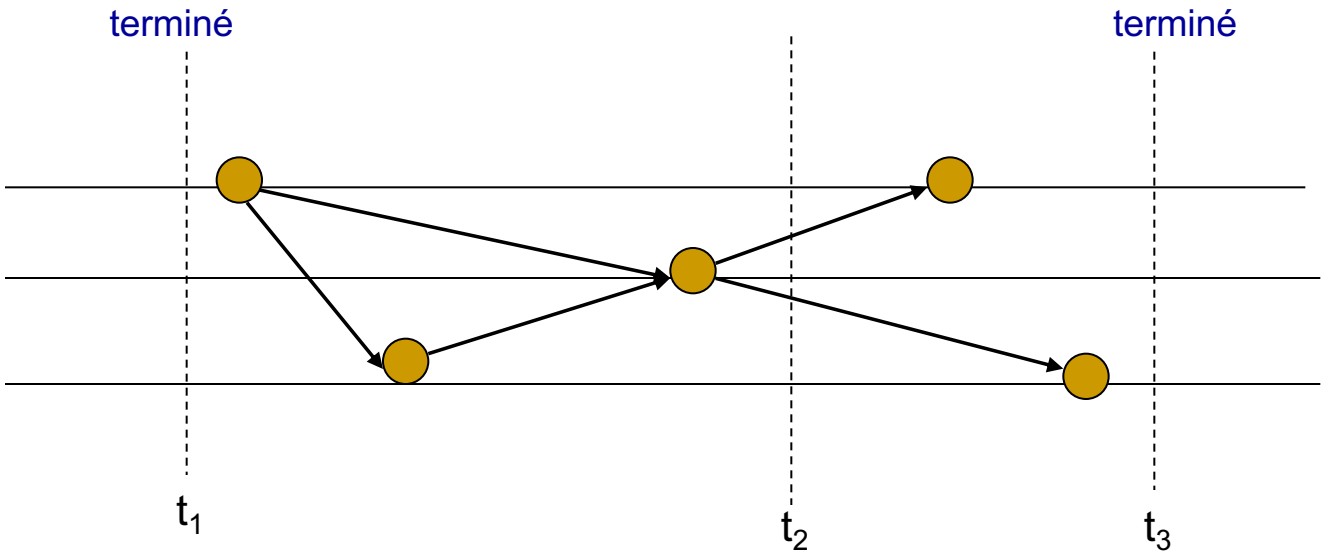


Devient:



Modèle atomique

- **Terminaison détectée lorsque tous les canaux son vides.**



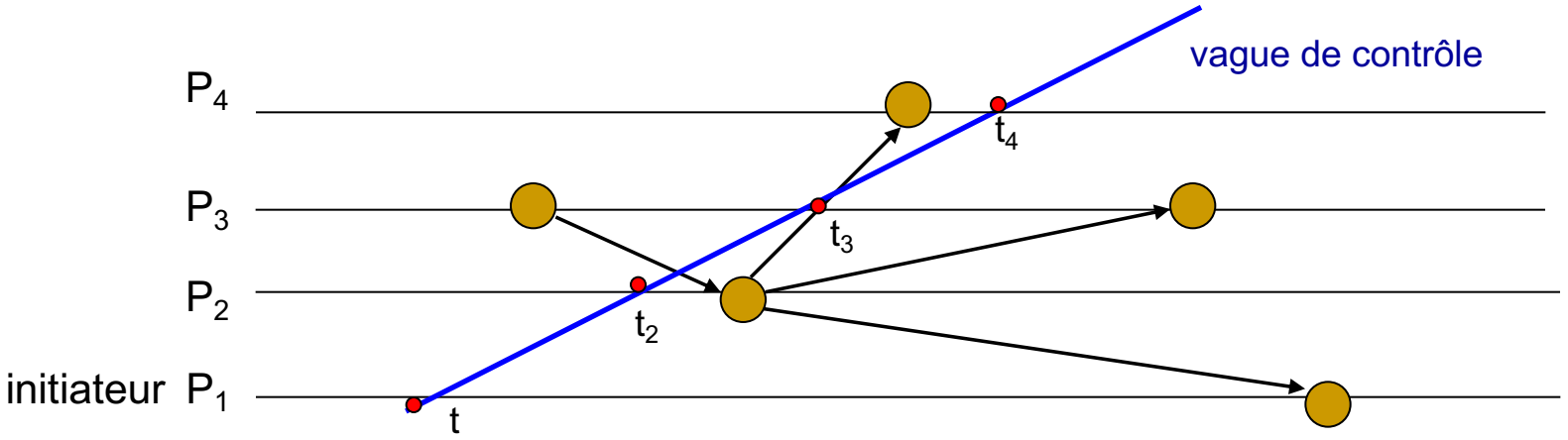
Détection Répartie de la Terminaison

- Modèle atomique :
  - Une **mauvaise solution** avec **deux** compteurs
    - N processus
    - Supposons qu'un processus ***i*** (initiateur) veut savoir si le système se trouve dans un état terminal : tous les canaux vides
      - *i* envoie un message de contrôle à tous les N-1 autres processus à un instant *t*.
    - Chaque processus *j* répond à *i* avec le nombre de messages reçus $r_j(t)$ et nombre de messages envoyés $s_j(t)$;
    - En recevant tous les messages, le site i calcule :
      - $S(t) = \Sigma\, s_j(t_j)$ et $R(t) = \Sigma\, r_j(t_j)$
      - Si S(t) = R(t), le nombre de messages envoyés = nombre de messages reçus alors

        - les canaux sont vides => **détection de la terminaison** **FAUX !!!**
    - **Pourquoi?**

Détection Répartie de la Terminaison

- Inexistence d'un temps global absolu: le moment où les processus *j* ont reçu les messages de contrôle est t_j et non pas *t,* le moment de l'envoi du message de contrôle par *i*.
  - La ligne qui connecte tous les t_j forme une vague de contrôle ("a time cut") .



$s_1(t)=0$; $s_2(t_2)=0$; $s_3(t_3)=1$; $s_4(t_4)=0$;
$r_1(t)=0$; $r_2(t_2)=0$; $r_3(t_3)=0$; $r_4(t_4)=1$;

$S(t) = \Sigma\, s_i(t_i) = 1 = R(t) = \Sigma\, r_i(t_i) = 1$

S(t) = R(t) : canaux vides : Détection de la terminaison => FAUX !!!

Détection Répartie de la Terminaison

- **Solution : L'algorithme des quatre compteurs**
  - Mattern [1987].
  - Compter deux fois :
    - Fin de la première vague de contrôle: l'initiateur accumule les valeurs de $s_i(t_i)$ et $r_i(t_i)$ $\forall$ i : $1 \leq i \leq N$ dans S* et R*.
    - Fin de la deuxième vague de contrôle: l'initiateur accumule les valeurs de $s_i(t_i)$ et $r_i(t_i)$ $\forall$ i : $1 \leq i \leq N$ dans S'* et R'* (depuis le début de la première vague).
  - L' exécution est terminé si :

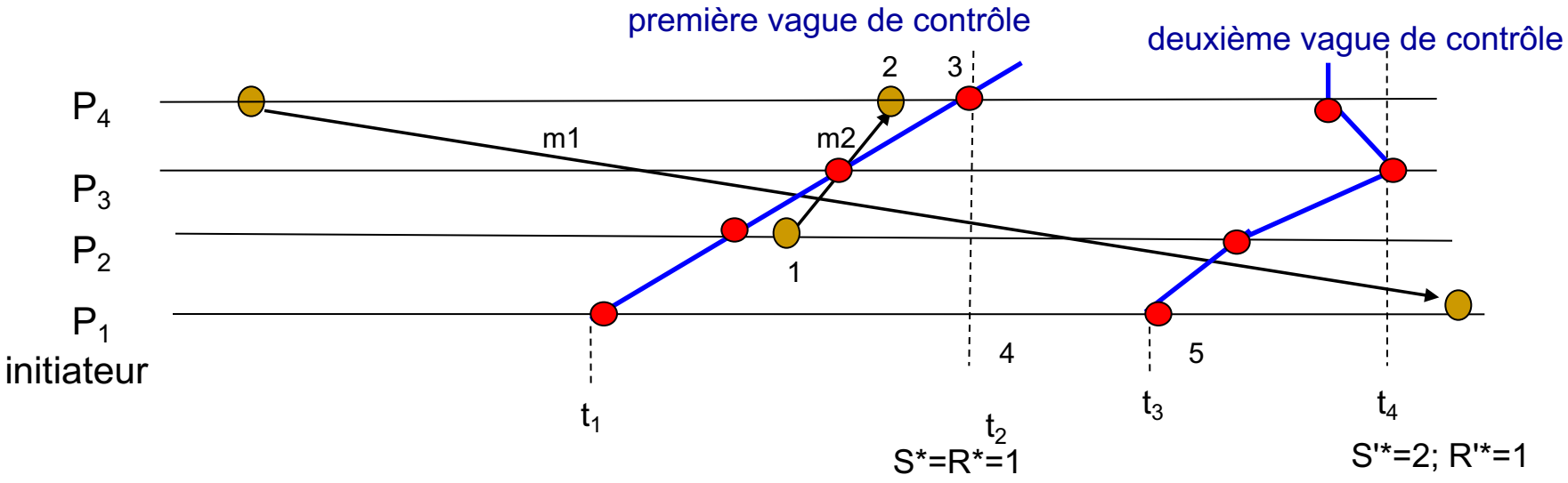
$$\mathbf{S^* = R^* = S'^* = R'^*}$$

    - L'exécution est terminée à la fin de la première vague.

Détection Répartie de la Terminaison

L'algorithme des Quatre Compteurs

Application n'a pas terminé : S*=R*=R'*=1 mais S'*=2



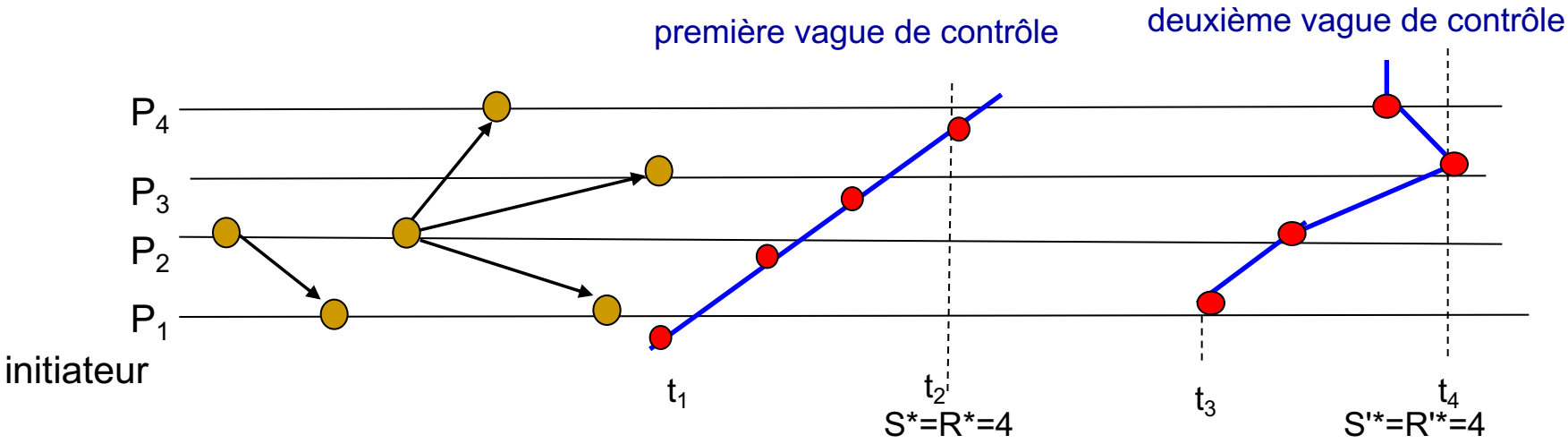
● : Site P_i reçoit le msg de contrôle de P_1 et renvoi les information sur $s(t_2)_i$ et $r(t_2)_i$

Deuxième vague commence après la reception de tous les messages de contrôle: après t_2

Détection Répartie de la Terminaison

L'algorithme des Quatre Compteurs

Application a terminé: S*=R*=S'*=R'*=4



R* = S'* => l'exécution s'est terminée à la fin de la première vague: t_2

Terminaison détectée à la fin de la deuxième vague : t4

Détection Répartie de la Terminaison

- **L'algorithme des quatre Compteurs (cont.)**
  - R*= S'*, alors l'exécution répartie s'est terminée à la fin de la première vague.
    - Soient t_2 la date où la première vague s'est terminée et $t_3 \geq t_2$ la date du début de la deuxième vague.

$$R^* = S'^* \Rightarrow R(t_2) = S(t_2)$$

Détection Répartie de la Terminaison

- **L'algorithme des quatre Compteurs**

(1) Les compteurs locaux sont monotones, $t \leq t'$ implique $s_i(t) \leq s_i(t')$ et $r_i(t) \leq r_i(t')$.
- *Preuve* : suit de la définition.

(2) Le nombre de messages envoyés et reçus est monotones, $t \leq t'$ implique $S(t) \leq S(t')$ et $R(t) \leq R(t')$.
- *Preuve* : suit de la définition et (1).

(3) $R^* \leq R(t_2)$.
- *Preuve* : suit de (1) et le fait que toutes les valeurs de r_i sont collectées avant ($\leq$) t_2.

(4) $S'^* \geq S(t_3)$.
- *Preuve* : suit de (1) et le fait que toutes les valeurs de s_i sont collectées après ($\geq$) t_3.

(5) $\forall$ t, $R(t) \leq S(t)$.
- *Preuve*: la différence non négative $D(t) = S(t) - R(t)$ correspond au nombre de messages en transit. $D(t) \geq 0$.

Détection Répartie de la Terminaison

- **L'algorithme des Quatre Compteurs**

$$R^* = S'^* \Rightarrow R(t_2) \geq S(t_3) \quad (3,4)$$
$$\Rightarrow R(t_2) \geq S(t_2) \quad (2)$$
$$\Rightarrow R(t_2) = S(t_2) \quad (5)$$

- Cela dit, l'exécution s'est terminée à l'instant t_2

Détection Répartie de la Terminaison

- **Bibliographie**
  - J. Misra, Detecting termination of distributed computations using markers. PODC, pages 290-294.
  - E.W.Dijkstra, Derivation of a termination detection algorithm for distributed computations. *Information Processing Letters* 16, pages 217-219, 1983
  - F. Mattern, Algorithms for distributed termination detection. *Distributed Computing,* Vol 2, pages 161-175, Springer–Verlag, 1987.
  - S. P. Rana, A distributed solution of distributed termination problem. *Information Processing Letters* 17, pages 43-46, 1983.
  - J. Matocha and T. Camp, A taxonomy of distributed termination detection algorithms. *The Journal of Systems and Softwares 43*, pages 207-221, 1998.