
Algorithmes par vague

Master informatique UPMC 2016-2017
UE AR (4I403)

Plan

- **Définition Algorithme Total**
- **Exemple d'algorithme**
 - Algorithme de l'anneau
 - Algorithme de l'arbre
 - Algorithme de l'Echo
 - Algorithme de la Phase

Algorithme Total

■ Algorithme par Vague

- Les algorithmes par vague sont utilisés pour diffuser une information sur le réseau, découvrir la topologie du réseau, rassembler des informations du réseau.
- N nœuds
 - Un nœud ne se communique qu'avec ses voisins

• **Tous les nœuds du réseau participent avant qu'une décision soit prise**

Algorithme Total

- **Un algorithme à Vague doit satisfaire les trois propriétés:**
 - ***terminaison*** : toute exécution est finie
 - ***décision*** : une décision doit être prise à terme par au moins un processus.
 - ***dépendance*** : Une décision est précédée causalement par un événement de chaque processus.

Algorithme Total

- Types de nœuds :
 - **Initiateur** :
 - nœud qui spontanément décide de démarrer l'algorithme
 - **Non-initiateur** :
 - Ne commence à exécuter qu'après avoir reçu un message.
- e_p = premier événement qui a lieu en p lors d'une exécution de l'algorithme :
 - *Nœud initiateur* : e_p est un événement interne ou l'envoi de message
 - *Nœud non-initiateur* : e_p est l'événement réception de message.
- **Observation** :
 - Pour un même algorithme, pour des exécutions différentes, le nœud initiateur peut être différent.

Algorithme Total

➤ Caractéristiques :

■ Symétrie

□ *Algorithme symétrique* :

- L'algorithme est le même dans tous les nœuds

□ *Algorithme asymétrique*

- L'algorithme du nœud *initiateur* n'est pas le même que celui d'un nœud non *initiateur*.

■ Initialisation du calcul

□ *Algorithme centralisé*

- Il existe un unique processus qui est *initiateur* du calcul

□ *Algorithme décentralisé*

- Pour tout sous-ensemble $\Pi_0 \subseteq \Pi$, \exists un calcul de l'algorithme dont Π_0 est l'ensemble des initiateurs.

Algorithme Total

- Une décision est prise au plus une fois par un processus p
 d_p = événement correspondant à la décision prise par p

Définition (Tel) : une exécution d'un algorithme est totale si au moins un processus p décide et pour tout $q \in N$ et pour tout p qui prend une décision $e_q \rightarrow d_p$.

Un algorithme est **total** ssi toutes ses exécutions possibles sont totales

- Dans un algorithme total sur un réseau de N nœuds, il y a au moins $N-1$ message échangés.

Algorithmes par Vague

■ Notation (Tel) :

➤ Événements :

- S_p : envoie message
- R_p : réception message
- D_p : décision

➤ $E_p : \{condition\}$

- L'événement E_p est exécuté si la $\{condition\}$ est vraie.

1. L'algorithme de l'anneau

- Anneau unidirectionnel
- Algorithme asymétrique et centralisé
 - N noeuds
 - Les communications sont fiables, pas forcément FIFO, et tout message émis est reçu dans un temps fini mais arbitraire (modèle temporel asynchrone)
 - Un nœud ne connaît que l'identifiant de son successeur.
- Principe de l'algorithme :
 - Un seul initiateur à chaque exécution.
 - *Initiateur* envoie un jeton dans l'anneau.
 - Jeton doit être reçu par tous les nœuds.
 - L'*initiateur* décide lorsque le jeton lui est renvoyé.

1.L'algorithme de l'anneau

Variable :

booléen $Rec_p = \text{false};$
/*contrôle de la
réception du message*/

***p* initiateur :**

$S_p : \{ \text{Spontanément, une fois} \}$
envoie $\langle \rangle$ au successeur

$R_p : \{ \text{Un message } \langle \rangle \text{ arrive} \}$
réception de $\langle \rangle$;
 $Rec_p = \text{true};$

$D_p : \{ Rec_p \}$
Décision

***p* non_initiateur :**

$R_p : \{ \text{Un message } \langle \rangle \text{ arrive} \}$
réception de $\langle \rangle$;
 $Rec_p = \text{true};$

$S_p : \{ \mathbf{Recp} \}$
envoie $\langle \rangle$ au successeur;
 $Rec_p = \text{false};$

$\langle \rangle$: message vide.

1.L'algorithme de l'anneau

- Supposons :
 - $(j+1)\%N$ est le successeur de j
 - s_j : événement envoi de message du site j
 - r_j : événement réception de message du site j
 - e_j : premier événement de j
 - i initiateur
 - d_i : événement décision.
 - $e_i = s_i$ et $e_j = r_j$ pour tout $j \neq i$.
 - $r_j \rightarrow s_j$ pour tout $j \neq i$
 - $s_j \rightarrow r_{j+1}$ pour tout j
 - $r_i \rightarrow d_i$. (seule initiateur décide)

$$e_i = s_i \rightarrow r_{i+1} = e_{i+1} \rightarrow s_{i+1} \rightarrow r_{j+2} \dots \rightarrow r_j = e_j \rightarrow s_j \dots \rightarrow r_i = d_i$$

- Complexité en nombre de messages et temps : \mathbf{N}

2. L'algorithme de l'arbre

- N nœuds
 - Un nœud ne connaît que l'identifiant de ses voisins
- Liens bidirectionnels
- Algorithme symétrique et (**pas centralisé ni décentralisé**)
- Principe de l'algorithme
 - Un nœud qui a reçu un message de tous ses voisins sauf un envoie un message à celui-ci.
 - En ne possédant qu'un voisin, les *feuilles* de l'arbre sont des nœuds *initiateurs*
 - Toutes les feuilles, (possibilité de sauf une) doivent être des initiateurs.
 - Un nœud qui a reçu un message de tous ses voisins décide.

2. L'algorithme de l'arbre

Variables :

set $Vois_p$; /* ensemble de voisins de p */

boo $Rec_p[q]$ = false; $\forall q \in Vois_p$ /*contrôle réception message*/

boo $Sent_p$ = false; /*contrôle envoi d'un message*/

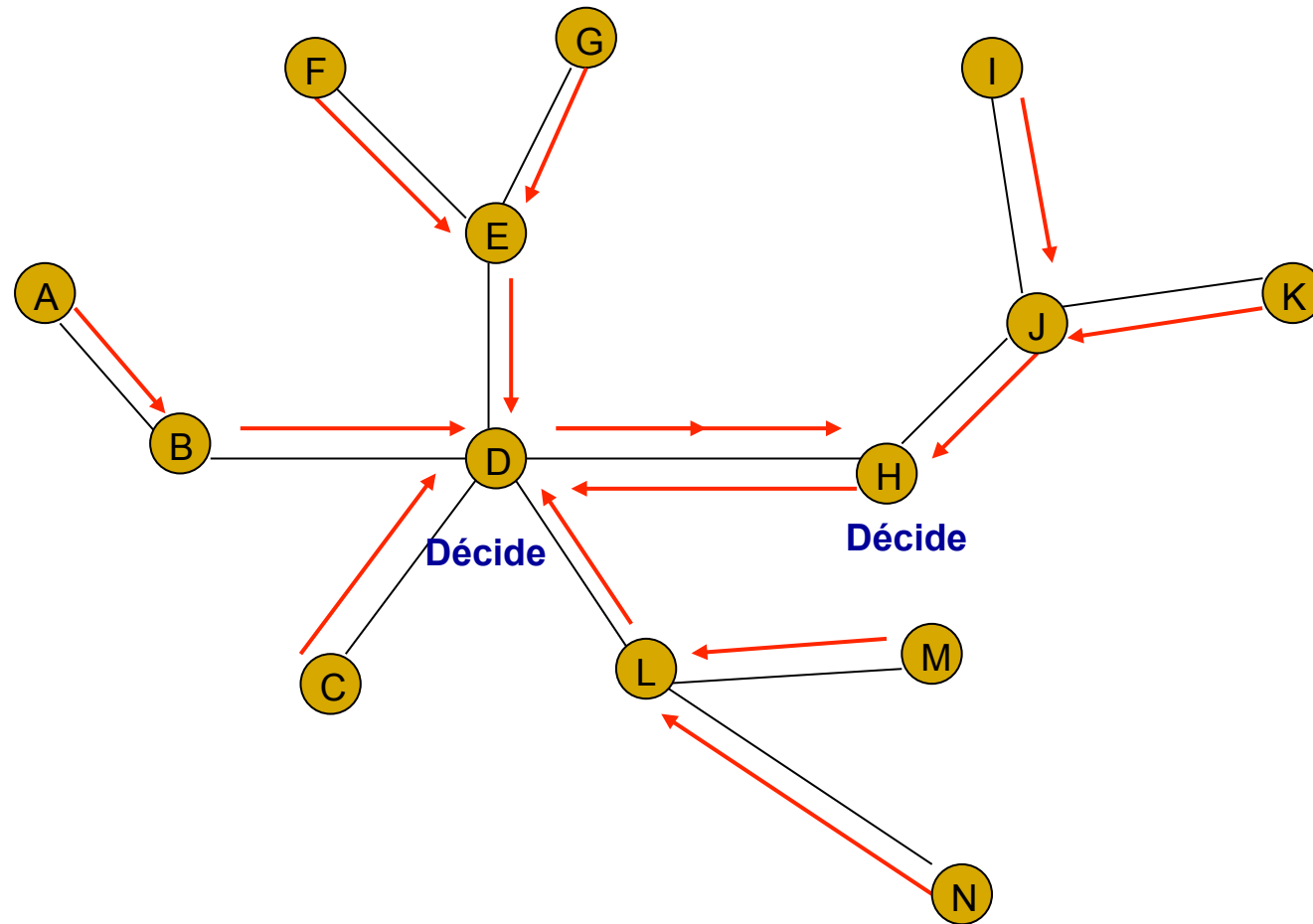
R_p : { Un message $\langle \rangle$ arrive de q }
réception de $\langle \rangle$;
 $Rec_p[q]$ = true;

S_p : { $\exists q \in Vois_p$: $\forall r \in Vois_p, r \neq q$: $Rec_p[r]$ **et** **!Sent_p** }
envoie $\langle \rangle$ à q
Sent_p = true;

D_p : { $\forall q \in Vois_p$: $Rec_p[q]$ }
Décision

$\langle \rangle$: message vide.

2. L'algorithme de l'arbre



2. L'algorithme de l'arbre

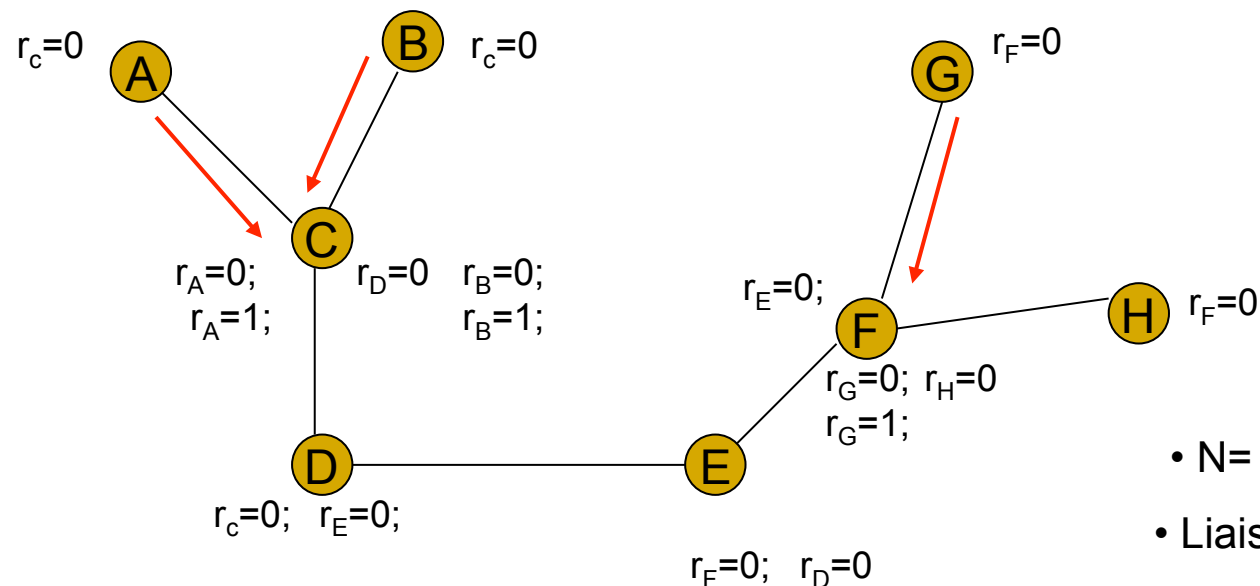
Théorème 1: Tant qu'un état permettant la décision n'est pas atteint, il y a toujours une émission ou une réception possible.

■ **Preuve:**

- A chaque liaison bidirectionnelle, on associe 2 bits r correspondant aux 2 sens d'émission, initialisés à 0. Le bit est mis à 1 quand le site à l'extrémité a reçu un message.
- Soit :
 - K - le nombre de sites ayant émis
 - M - le nombre de messages en transit
 - F - le nombre total de bits r à 0.

2. L'algorithme de l'arbre

- La topologie est un arbre, il y a donc $2(N - 1)$ liaisons (orientées).
- Le nombre de messages reçus est $(K - M)$.
- On a donc à tout instant $F = 2(N - 1) - (K - M)$



$M=0$; $K=0$

$M=3$; $K=3$

$M=0$; $K=3$

• $F=14$

• $F=11$

• $N=8$

• Liaisons orientées : $2*(N-1) = 14$

Note : Un site i peut émettre lorsqu'il n'a qu'un r_{ij} à 0

2. L'algorithme de l'arbre

- A tout instant : $F = 2(N - 1) - (K - M)$:
 - $M > 0$: il y a une réception possible.
 - $M = 0$: (*preuve par contradiction*)
 - Puisqu'on suppose qu'on n'est pas dans un état où la décision n'a pas eu lieu, tous les sites ont (au moins) un bit r à 0. Donc $F \geq N$.
 - Supposons qu'il n'y a pas d'émission possible, les $(N - K)$ sites n'ayant pas émis ont (au moins) un deuxième bit r à 0. Donc, $F \geq N + (N - K) = 2N - K$. Mais comme $M = 0$, la formule générale de F donne $F = 2N - K - 2$. On arrive donc à une *contradiction*. Donc, il y a des émissions possibles.

Note : Un site i peut émettre lorsqu'il n'a qu'un r_{ij} à 0

2. L'algorithme de l'arbre

- **En prenant la contraposée du *Théorème 1*: s'il n'y a ni émission ni réception possible, alors on est dans un état permettant la décision.**
 - Conséquence : comme le nombre d'émissions (donc de réceptions) est borné par le nombre de sites N (chaque processus n'envoie au plus qu'un message), on parvient toujours à une décision dans un temps fini.

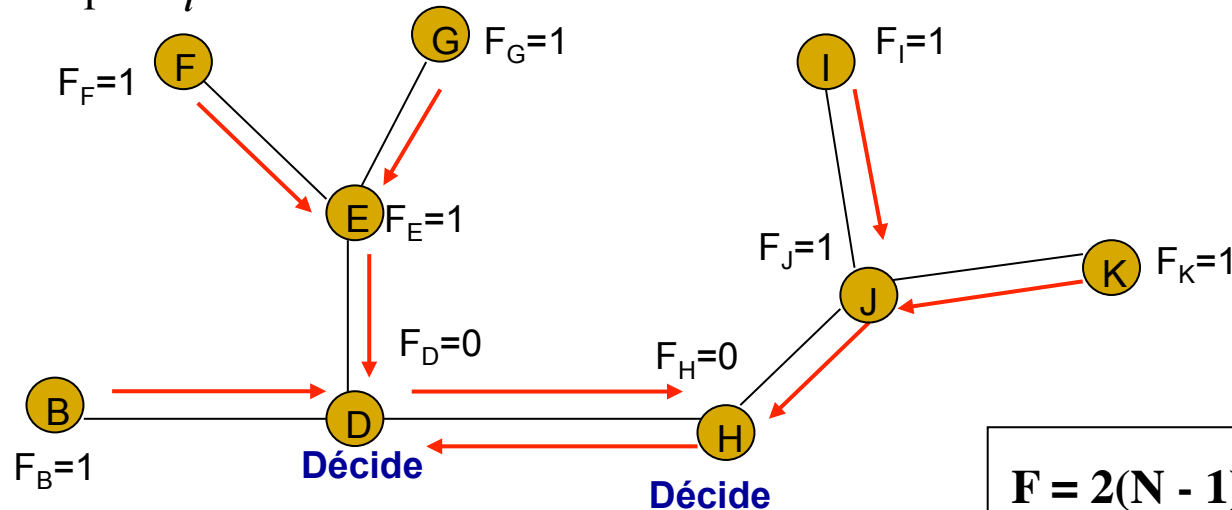
2. L'algorithme de l'arbre

- **Montrer que dans un état terminal (plus d'émission, de réception ou de décision possible), tous les sites ont émis une fois.**
 - **Preuve par contradiction**
 - Supposons qu'un site p_0 n'a pas encore émis.
 - Soit tous les voisins de p_0 ont émis. Comme ils n'ont pas reçu de message de p_0 , c'est donc à lui qu'ils ont envoyé. Donc, p_0 a reçu un message de tous ses voisins et il peut décider (il peut aussi émettre). Le fait qu'il peut émettre *contredit* le fait que l'état est terminal.
 - Soit au moins un de ses voisins n'a pas émis : soit p_1 ce voisin. Si p_1 ne peut pas émettre, c'est qu'au moins un autre de ses voisins (p_2) ne lui a pas envoyé de message, donc n'a pas émis. Le site p_2 doit lui aussi avoir un deuxième voisin qui n'a rien envoyé. Comme le graphe est acyclique, il ne peut pas s'agir de p_0 mais forcément d'un autre site p_3 . En itérant le raisonnement, on conclut qu'aucun site n'a émis. Or comme les feuilles peuvent émettre initialement, on aboutit à une *contradiction*.

2.L'algorithme de l'arbre

■ Montrer qu'il y a exactement deux décideurs.

- Lorsque tous les sites ont émis, et que leurs messages sont arrivés, on a ($K = N, M = 0$) : $F = N - 2$.
- Soit F_i le nombre de bits à 0 pour le site i . Comme i a émis, $F_i \leq 1$. F est la somme des F_i . On a donc $(N - 2)$ sites tels que $F_i = 1$. Donc il n'y a que 2 sites tels que $F_i = 0$. Ces deux sites sont les décideurs.



$$F = 2(N - 1) - (K - M)$$

Note : Un site i peut émettre lorsqu'il n'a qu'un r_{ij} à 0

2. L'algorithme de l'arbre

■ Conclusions :

- Il y a exactement *deux décideurs*
 - Les deux décideurs sont voisins
 - Le dernier nœud dont un décideur a reçu un message est aussi un décideur.
- Dans un état terminal, tous les nœuds ont émis une fois
 - Il n'y a qu'un message qui circule dans un lien sauf celui entre les 2 décideurs où circule deux messages
 - **Complexité en terme de messages :**
 - $\text{Nb}_{\text{lien}} \text{ d'un arbre} = N-1 \Rightarrow \text{Nb message} = (N-1)+1 = N$
 - **Complexité en temps :**
 - $O(D)$ où D = diamètre de l'arbre.
- Algorithme *n'est pas décentralisé* parce que si Π_0 contient d'autres nœuds que les feuilles alors Π_0 n'est pas un ensemble d'initiateurs.

3. Algorithme de l'Echo

- Proposé par Chang [1982]
- Topologie arbitraire
 - Graphe connexe bidirectionnel
- Algorithme centralisé – un seul initiateur
- Principe de l'algorithme
 - *initiateur* :
 - envoie un message à tous ses voisins
 - lorsqu'il a reçu un message de tous ses voisins, il décide
 - Un nœuds *non-initiateur* :
 - sauvegarde le lien par où le premier message a été reçu ("père")
 - émet à tous les voisins sauf le père
 - lorsqu'il a reçu un message de tous ses voisins, il envoie un message à son "père"

3. Algorithme de l'Echo

<> : message vide.

Variables :

set $Vois_p$; /* ensemble de voisins de p */

boo $Rec_p[q]$ = false; $\forall q \in Vois_p$ /*contrôle réception message*/

int $père_p$ = nil;

p initiateur :

S_p : { spontanément une fois }
 $\forall q \in Vois_p$, envoie <> à q

R_p : { Un message <> arrive de q }
réception de <>;
 $Rec_p[q]$ = true;

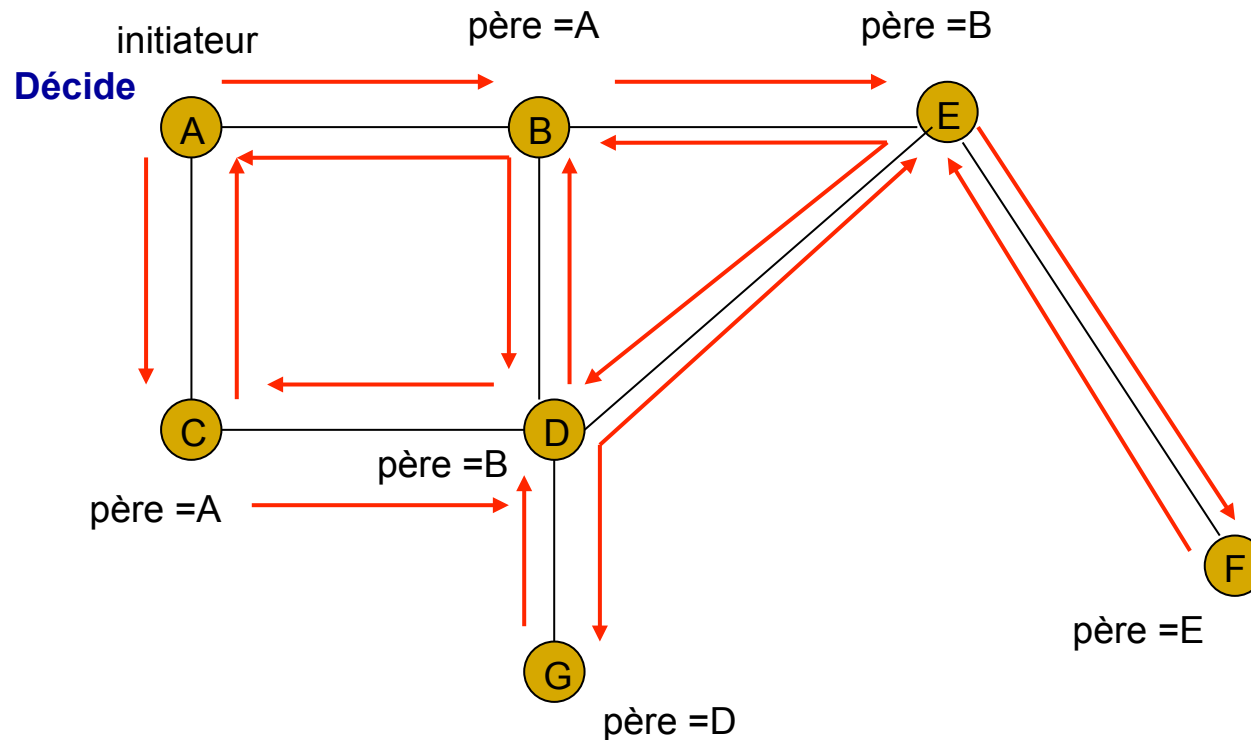
D_p : { $\forall q \in Vois_p$: $Rec_p[q]$ }
Décision

p non_initiateur :

R_p : { Un message <> arrive de q }
réception de <>;
 $Rec_p[q]$ = true;
if ($père_p$ = nil)
 $père_p$ = q ;
 $\forall r \in Vois_p - \{q\}$, envoie <> à r

S_p : { $\forall q \in Vois_p$, : $Rec_p[q]$ }
 envoie <> à $père_p$

3. Algorithme de l'Echo



3. Algorithme de l'Echo

■ Conclusions :

- L'initiateur est le décideur
- Complexité en terme de messages
 - $2 * N_{\text{lien}}$ (N_{lien} = nombre de liens)
- Complexité en terme de temps
 - $O(D)$ où D =diamètre du réseau.
- Possibilité de construire un arbre de recouvrement (voir TD)

4. Algorithme de la Phase

- Topologie arbitraire
 - Graphe orienté fortement connexe
- Algorithme décentralisé et symétrique
- Diamètre D du réseau est connu de tous les sites
- Principe de l'algorithme :
 - Chaque processus envoie D fois un message à tous ses voisins sortants.
 - Un processus n'a le droit d'émettre un message à tous ses voisins sortants pour la $(i+1)^{\text{ème}}$ fois qu'après avoir reçu le $i^{\text{ème}}$ message de tous ses voisins entrants.
 - Un processus décide lorsqu'il a reçu D messages de tous ses voisins entrants.

4. Algorithme de la Phase

Variables :

set In_p ; /* ensemble de voisins de p entrant */

set Out_p ; /* ensemble de voisins de p sortant */

int $RCount_p[q] = 0$; $\forall q \in In_p$ /*contrôle réception message*/

int $SCount_p = 0$ /*contrôle envoi message*/

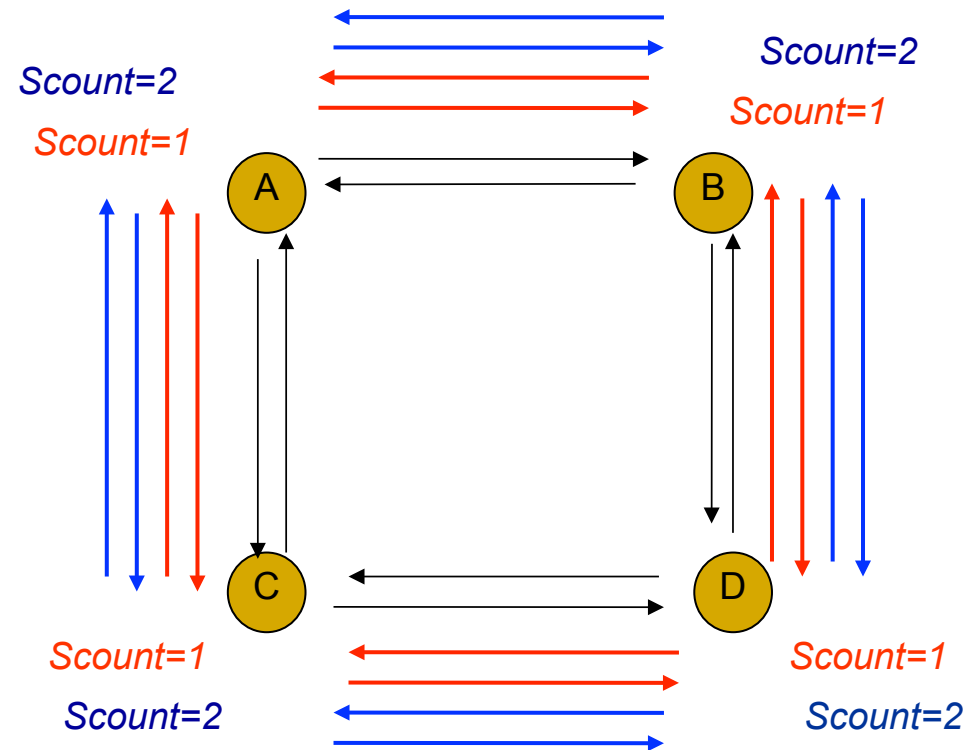
$R_p : \{ \text{Un message } \langle \rangle \text{ arrive de } q \}$
réception de $\langle \rangle$;
 $RCount_p[q]++$;

$S_p : \{ \forall q \in In_p : RCount_p[q] \geq SCount_p \text{ et } SCount_p < D \}$
for all $r \in Out_p$ envoi $\langle \rangle$ à r ;
 $SCount_p++$;

$D_p : \{ \forall q \in In_p : RCount_p[q] \geq D \}$
Décision

Observation: La primitive S_p doit être exécutée initialement par (au moins) un processus, l'initiateur.

4. Algorithme de la Phase



Diamètre =2

4. Algorithme de la Phase

- **Plusieurs messages peuvent être envoyés sur un lien**
- **Tous les processus peuvent décider**
 - Plus d'émission ni de réception possible
- **Complexité :**
 - Messages = $Nb_{lien} * D$ Nb_{lien} = nombre de lien
 - Temps = $O(D)$ D = diamètre

Résumé Algorithmes par Vague

Algorithme	Topologie	Centr./ Décen.	Déci- deur	Symé- trie	Nb. Mess.	Temps
<i>Anneau</i>	Anneau unidirec.	Centralisé	1 initiateur	Non	N	N
<i>Arbre</i>	Arbre bidirect.	Pas centr. Pas decen.	2	Oui	N	O(D)
<i>Echo</i>	arbitraire bidirect.	centralisé	1 initiateur	Non	$2 * Nb_{lien}$	O(D)
<i>Phase</i>	arbitraire	décentralisé	Tous	Oui	$Nb_{lien} * D$	O(D)

Bibliography

- Gerard Tel, *Introduction to Distributed Algorithms*, Cambridge University Press, 1994, 2000 (2ème edition).
- Gerard Tel, **Total Algorithms**, *ALCOM: Algorithms Review, Newsletter of the ESPRIT II Basic Research Actions Program Project no. 3075*
- Ernest J.H. Chang, *Echo Algorithms: Depth Parallel Operations on General Graphs*, IEEE Transactions on Software Engineering, Vol. 8, No. 4, July 1982