

Propagation d'Information avec Retour (PIR) dans un réseau quelconque

Alain Cournier Stéphane Devismes

Université de Picardie Jules Verne

1^{er} octobre 2026



Plan

- 1 Modèle
- 2 Principe
- 3 Spécification
- 4 Algorithme
- 5 Correction et complexité de l'algorithme

Plan

- 1 **Modèle**
- 2 Principe
- 3 Spécification
- 4 Algorithme
- 5 Correction et complexité de l'algorithme

- Processus et canaux asynchrones.
- Canaux étiquetés de 1 à δ_p pour tout processus p .
- Pas de faute.
- Topologie connexe avec au moins deux nœuds.
- Mono-initiateur.

Plan

- 1 Modèle
- 2 Principe
- 3 Spécification
- 4 Algorithme
- 5 Correction et complexité de l'algorithme

- Un pointeur de canal *Père*, initialisé à $\perp$.

L'initiateur devra fixer son pointeur à $\top$.

Pour chaque suiveur, *Père* désignera l'arête incidente le reliant à son père dans l'arbre couvrant construit durant la première phase de l'algorithme (la phase de diffusion).

- Une variable *Cpt*, initialisée à 0.

Ce compteur contiendra le nombre de messages reçus par le processus.

Algorithme à vagues avec deux phases

Deux phases : la phase de diffusion et la phase de retour : deux type de messages *Brd* et *Ack*.

Algorithme à vagues avec deux phases

Deux phases : la phase de diffusion et la phase de retour : deux type de messages *Brd* et *Ack*.

- L'initiateur démarre la vague de PIR lorsqu'il a une donnée d à diffuser.

Dans ce cas, il initie la phase de diffusion en envoyant un message *Brd* contenant d à tous ses voisins.

Algorithme à vagues avec deux phases

Deux phases : la phase de diffusion et la phase de retour : deux type de messages *Brd* et *Ack*.

- L'initiateur démarre la vague de PIR lorsqu'il a une donnée d à diffuser.

Dans ce cas, il initie la phase de diffusion en envoyant un message *Brd* contenant d à tous ses voisins.

- Lorsqu'un processus p reçoit un message *Brd* de q :

Algorithme à vagues avec deux phases

Deux phases : la phase de diffusion et la phase de retour : deux type de messages *Brd* et *Ack*.

- L'initiateur démarre la vague de PIR lorsqu'il a une donnée d à diffuser.

Dans ce cas, il initie la phase de diffusion en envoyant un message *Brd* contenant d à tous ses voisins.

- Lorsqu'un processus p reçoit un message *Brd* de q :

- 1 Il incrémente son compteur.

Algorithme à vagues avec deux phases

Deux phases : la phase de diffusion et la phase de retour : deux type de messages *Brd* et *Ack*.

- L'initiateur démarre la vague de PIR lorsqu'il a une donnée d à diffuser.

Dans ce cas, il initie la phase de diffusion en envoyant un message *Brd* contenant d à tous ses voisins.

- Lorsqu'un processus p reçoit un message *Brd* de q :

- 1 Il incrémente son compteur.
- 2 Ensuite, si p est un suiveur et qu'il s'agit de sa toute première réception,
 - il affecte son pointeur $Père$ à q et
 - relaie le message à tous ses voisins sauf q .

Algorithme à vagues avec deux phases

Deux phases : la phase de diffusion et la phase de retour : deux type de messages *Brd* et *Ack*.

- L'initiateur démarre la vague de PIR lorsqu'il a une donnée d à diffuser.

Dans ce cas, il initie la phase de diffusion en envoyant un message *Brd* contenant d à tous ses voisins.

- Lorsqu'un processus p reçoit un message *Brd* de q :

- 1 Il incrémente son compteur.
- 2 Ensuite, si p est un suiveur et qu'il s'agit de sa toute première réception,
 - il affecte son pointeur $P_{\text{ère}}$ à q et
 - relaie le message à tous ses voisins sauf q .
- 3 Enfin dans tous les cas, p teste s'il a terminé sa participation au PIR : si Cpt est supérieur ou égal à son degré δ_p .

Dans ce cas,

- si p est l'initiateur, il décide,
- sinon il envoie un accusé réception *Ack* à son père (phase de retour).

Algorithme à vagues avec deux phases

Deux phases : la phase de diffusion et la phase de retour : deux type de messages *Brd* et *Ack*.

- L'initiateur démarre la vague de PIR lorsqu'il a une donnée d à diffuser.

Dans ce cas, il initie la phase de diffusion en envoyant un message *Brd* contenant d à tous ses voisins.

- Lorsqu'un processus p reçoit un message *Brd* de q :

- 1 Il incrémente son compteur.
- 2 Ensuite, si p est un suiveur et qu'il s'agit de sa toute première réception,
 - il affecte son pointeur $P_{\text{ère}}$ à q et
 - relaie le message à tous ses voisins sauf q .
- 3 Enfin dans tous les cas, p teste s'il a terminé sa participation au PIR :
si Cpt est supérieur ou égal à son degré δ_p .

Dans ce cas,

- si p est l'initiateur, il décide,
- sinon il envoie un accusé réception *Ack* à son père (phase de retour).

- Lorsqu'un processus p reçoit un message *Ack* de q :

Algorithme à vagues avec deux phases

Deux phases : la phase de diffusion et la phase de retour : deux type de messages *Brd* et *Ack*.

- L'initiateur démarre la vague de PIR lorsqu'il a une donnée d à diffuser.

Dans ce cas, il initie la phase de diffusion en envoyant un message *Brd* contenant d à tous ses voisins.

- Lorsqu'un processus p reçoit un message *Brd* de q :

- 1 Il incrémente son compteur.
- 2 Ensuite, si p est un suiveur et qu'il s'agit de sa toute première réception,
 - il affecte son pointeur $P_{\text{ère}}$ à q et
 - relaie le message à tous ses voisins sauf q .
- 3 Enfin dans tous les cas, p teste s'il a terminé sa participation au PIR :
si Cpt est supérieur ou égal à son degré δ_p .

Dans ce cas,

- si p est l'initiateur, il décide,
- sinon il envoie un accusé réception *Ack* à son père (phase de retour).

- Lorsqu'un processus p reçoit un message *Ack* de q :

- 1 Il incrémente son compteur.

Algorithme à vagues avec deux phases

Deux phases : la phase de diffusion et la phase de retour : deux type de messages *Brd* et *Ack*.

- L'initiateur démarre la vague de PIR lorsqu'il a une donnée d à diffuser.

Dans ce cas, il initie la phase de diffusion en envoyant un message *Brd* contenant d à tous ses voisins.

- Lorsqu'un processus p reçoit un message *Brd* de q :

- 1 Il incrémente son compteur.
- 2 Ensuite, si p est un suiveur et qu'il s'agit de sa toute première réception,
 - il affecte son pointeur $P_{\text{ère}}$ à q et
 - relaie le message à tous ses voisins sauf q .
- 3 Enfin dans tous les cas, p teste s'il a terminé sa participation au PIR :
si Cpt est supérieur ou égal à son degré δ_p .

Dans ce cas,

- si p est l'initiateur, il décide,
- sinon il envoie un accusé réception *Ack* à son père (phase de retour).

- Lorsqu'un processus p reçoit un message *Ack* de q :

- 1 Il incrémente son compteur.
- 2 Puis, il teste s'il a terminé sa participation au PIR.

Dans ce cas,

- si p est l'initiateur, il décide,
- sinon il envoie un accusé réception *Ack* à son père.

Plan

- 1 Modèle
- 2 Principe
- 3 Spécification**
- 4 Algorithme
- 5 Correction et complexité de l'algorithme

Spécification du PIR de la donnée d

Spécification du PIR de la donnée d

Sûreté

- 1 Au plus une décision est prise.
- 2 Si une décision est prise, alors elle dépend **causalement** d'un accusé de réception de d envoyé par chaque processus suiveur.

Vivacité

- 1 L'exécution termine.
- 2 Tout processus suiveur reçoit d .
- 3 Tout processus suiveur accuse réception de d .
- 4 L'initiateur finit par décider.

Spécification du PIR de la donnée d

Sûreté

- 1 Au plus une décision est prise.
- 2 Si une décision est prise, alors elle dépend **causalement** d'un accusé de réception de d envoyé par chaque processus suiveur.

Vivacité

- 1 L'exécution termine.
- 2 Tout processus suiveur reçoit d .
- 3 Tout processus suiveur accuse réception de d .
- 4 L'initiateur finit par décider.

Un algorithme de PIR est un algorithme à vagues

Une exécution se découpe en trois types d'évènements :

❶ Evènement interne

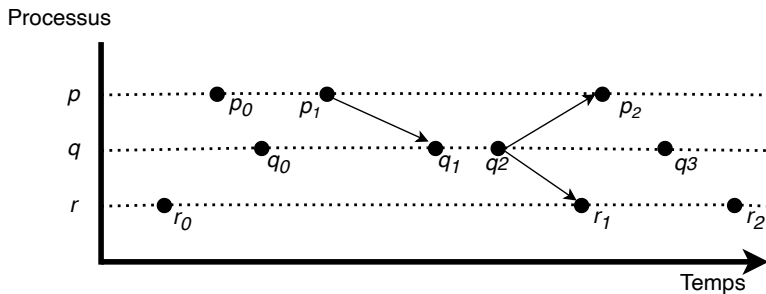
Une ou plusieurs instructions de calcul consécutives

❷ L'envoi d'un ou plusieurs messages

❸ La réception d'un message

La relation de causalité $\prec$ sur l'ensemble des événements est la relation la plus petite vérifiant les trois conditions suivantes :

- 1 Si a et b sont deux événements d'un même processus et que a arrive avant b , alors $a \prec b$.
- 2 Si a est l'envoi d'un message par un processus et que b est la réception du même message par un autre processus, alors $a \prec b$.
- 3 Si $a \prec b$ et $b \prec c$, alors $a \prec c$ (transitivité).



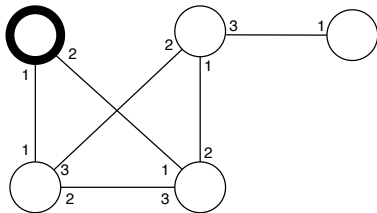
Exemple :

- $p_0 \prec p_1 \prec q_1 \prec q_2 \prec r_1 \prec r_2$ et donc $p_0 \prec r_2$ par transitivité
- $r_0 \not\prec q_1$ et $p_2 \not\prec r_1$

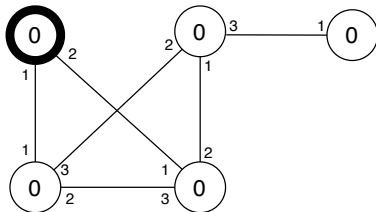
Plan

- 1 Modèle
- 2 Principe
- 3 Spécification
- 4 Algorithme**
- 5 Correction et complexité de l'algorithme

Trace exemple



Trace exemple



- Un pointeur de canal *Père*, initialisé à $\perp$.

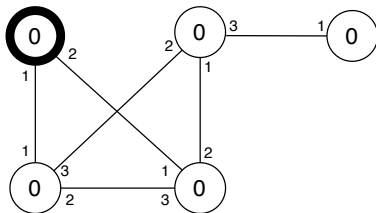
L'initiateur devra fixer son pointeur à T.

Pour chaque suiveur, *Père* désignera l'arête incidente le reliant à son père dans l'arbre couvrant construit durant la première phase de l'algorithme (la phase de diffusion).

- Une variable *Cpt*, initialisée à 0.

Ce compteur contiendra le nombre de messages reçus par le processus.

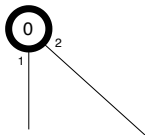
Trace exemple



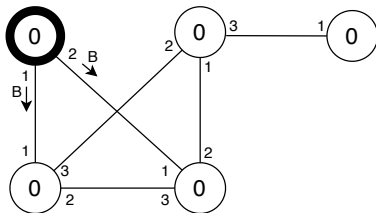
- L'initiateur démarre la vague de PIR lorsqu'il a une donnée d à diffuser.

Dans ce cas, il **initie la phase de diffusion** en envoyant un message Brd contenant d à tous ses voisins.

Vue locale :



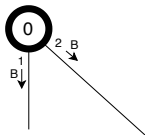
Trace exemple



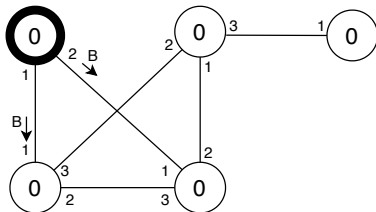
- L'initiateur démarre la vague de PIR lorsqu'il a une donnée d à diffuser.

Dans ce cas, il **initie la phase de diffusion** en envoyant un message Brd contenant d à tous ses voisins.

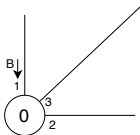
Vue locale :



Trace exemple



Vue locale :

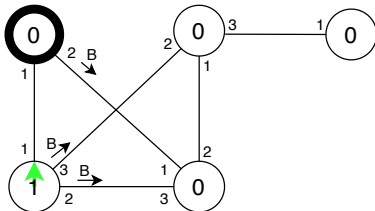


- Lorsqu'un processus p reçoit un message Brd de q :
 1. Il incrémente son compteur.
 2. Ensuite, si p est un suiveur et qu'il s'agit de sa toute première réception,
 - il affecte son pointeur $Père$ à q et
 - relaie le message à tous ses voisins sauf q .
 3. Enfin dans tous les cas, p teste s'il a terminé sa participation au PIR :
si Cpt est supérieur ou égal à son degré δ_p .

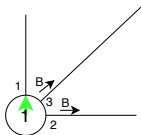
Dans ce cas,

 - si p est l'initiateur, il décide,
 - sinon il envoie un accusé réception Ack à son père (phase de retour).

Trace exemple



Vue locale :



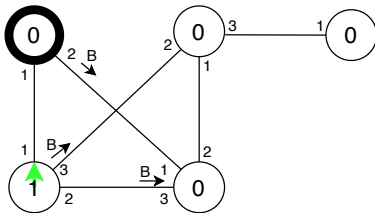
• Lorsqu'un processus p reçoit un message Brd de q :

1. Il incrémente son compteur.
2. Ensuite, si p est un suiveur et qu'il s'agit de sa toute première réception,
 - il affecte son pointeur $Père$ à q et
 - relaie le message à tous ses voisins sauf q .
3. Enfin dans tous les cas, p teste s'il a terminé sa participation au PIR :
 si Cpt est supérieur ou égal à son degré δ_p .

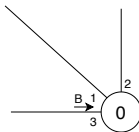
Dans ce cas,

- si p est l'initiateur, il décide,
- sinon il envoie un accusé réception Ack à son père (phase de retour).

Trace exemple

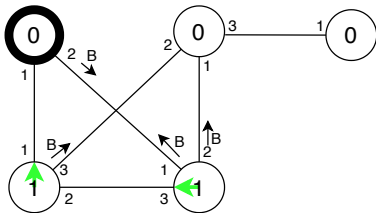


Vue locale :

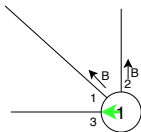


- Lorsqu'un processus p reçoit un message Brd de q :
 1. Il incrémente son compteur.
 2. Ensuite, si p est un suiveur et qu'il s'agit de sa toute première réception,
 - il affecte son pointeur $Père$ à q et
 - relaie le message à tous ses voisins sauf q .
 3. Enfin dans tous les cas, p teste s'il a terminé sa participation au PIR :
 si Cpt est supérieur ou égal à son degré δ_p .
 Dans ce cas,
 - si p est l'initiateur, il décide,
 - sinon il envoie un accusé réception Ack à son père (phase de retour).

Trace exemple



Vue locale :



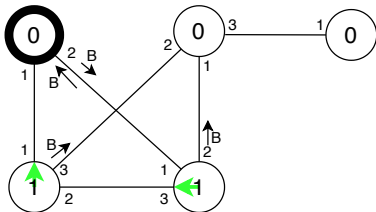
• Lorsqu'un processus p reçoit un message Brd de q :

1. Il incrémente son compteur.
2. Ensuite, si p est un suiveur et qu'il s'agit de sa toute première réception,
 - il affecte son pointeur $Père$ à q et
 - relaie le message à tous ses voisins sauf q .
3. Enfin dans tous les cas, p teste s'il a terminé sa participation au PIR :
 si Cpt est supérieur ou égal à son degré δ_p .

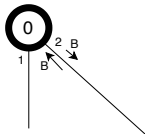
Dans ce cas,

- si p est l'initiateur, il décide,
- sinon il envoie un accusé réception Ack à son père (phase de retour).

Trace exemple

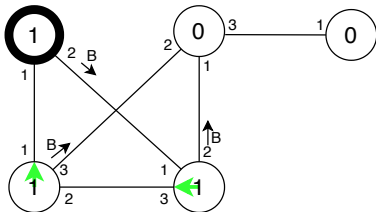


Vue locale :

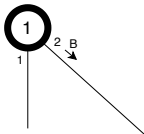


- Lorsqu'un processus p reçoit un message Brd de q :
 1. Il incrémente son compteur.
 2. Ensuite, si p est un suiveur et qu'il s'agit de sa toute première réception,
 - il affecte son pointeur $Père$ à q et
 - relaie le message à tous ses voisins sauf q .
 3. Enfin dans tous les cas, p teste s'il a terminé sa participation au PIR :
si Cpt est supérieur ou égal à son degré δ_p .
- Dans ce cas,
- si p est l'initiateur, il décide,
 - sinon il envoie un accusé réception Ack à son père (phase de retour).

Trace exemple



Vue locale :

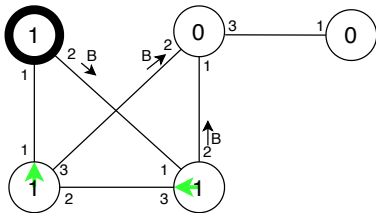


- Lorsqu'un processus p reçoit un message Brd de q :
 1. Il incrémente son compteur.
 2. Ensuite, si p est un suiveur et qu'il s'agit de sa toute première réception,
 - il affecte son pointeur $Père$ à q et
 - relaie le message à tous ses voisins sauf q .
 3. Enfin dans tous les cas, p teste s'il a terminé sa participation au PIR :
si Cpt est supérieur ou égal à son degré δ_p .

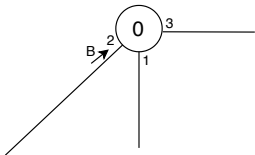
Dans ce cas,

 - si p est l'initiateur, il décide,
 - sinon il envoie un accusé réception Ack à son père (phase de retour).

Trace exemple



Vue locale :

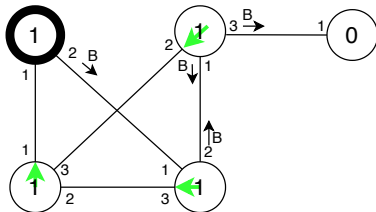


- Lorsqu'un processus p reçoit un message Brd de q :
 1. Il incrémente son compteur.
 2. Ensuite, si p est un suiveur et qu'il s'agit de sa toute première réception,
 - il affecte son pointeur $Père$ à q et
 - relaie le message à tous ses voisins sauf q .
 3. Enfin dans tous les cas, p teste s'il a terminé sa participation au PIR :
si Cpt est supérieur ou égal à son degré δ_p .

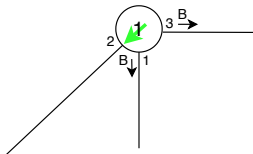
Dans ce cas,

 - si p est l'initiateur, il décide,
 - sinon il envoie un accusé réception Ack à son père (phase de retour).

Trace exemple

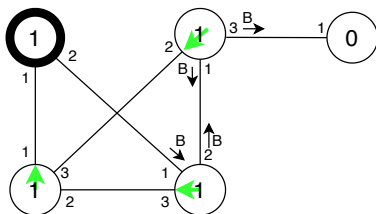


Vue locale :

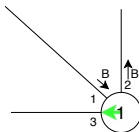


- Lorsqu'un processus p reçoit un message Brd de q :
 1. Il incrémente son compteur.
 2. Ensuite, si p est un suiveur et qu'il s'agit de sa toute première réception,
 - il affecte son pointeur $Père$ à q et
 - relaie le message à tous ses voisins sauf q .
 3. Enfin dans tous les cas, p teste s'il a terminé sa participation au PIR :
 si Cpt est supérieur ou égal à son degré δ_p .
 Dans ce cas,
 - si p est l'initiateur, il décide,
 - sinon il envoie un accusé réception Ack à son père (phase de retour).

Trace exemple



Vue locale :

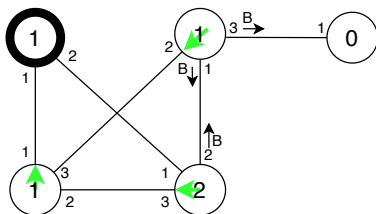


- Lorsqu'un processus p reçoit un message Brd de q :
 1. Il incrémente son compteur.
 2. Ensuite, si p est un suiveur et qu'il s'agit de sa toute première réception,
 - il affecte son pointeur $Père$ à q et
 - relaie le message à tous ses voisins sauf q .
 3. Enfin dans tous les cas, p teste s'il a terminé sa participation au PIR :
si Cpt est supérieur ou égal à son degré δ_p .

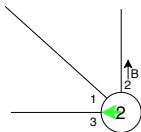
Dans ce cas,

 - si p est l'initiateur, il décide,
 - sinon il envoie un accusé réception Ack à son père (phase de retour).

Trace exemple



Vue locale :

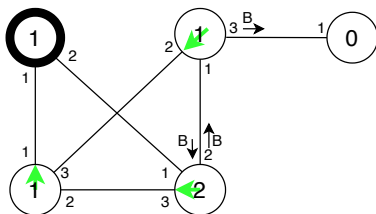


- Lorsqu'un processus p reçoit un message Brd de q :
 1. Il incrémente son compteur.
 2. Ensuite, si p est un suiveur et qu'il s'agit de sa toute première réception,
 - il affecte son pointeur $Père$ à q et
 - relaie le message à tous ses voisins sauf q .
 3. Enfin dans tous les cas, p teste s'il a terminé sa participation au PIR :
si Cpt est supérieur ou égal à son degré δ_p .

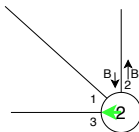
Dans ce cas,

 - si p est l'initiateur, il décide,
 - sinon il envoie un accusé réception Ack à son père (phase de retour).

Trace exemple



Vue locale :



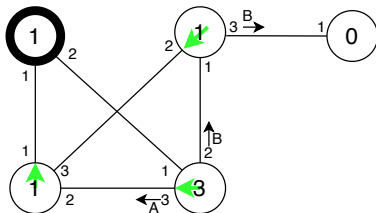
● Lorsqu'un processus p reçoit un message Brd de q :

1. Il incrémente son compteur.
2. Ensuite, si p est un suiveur et qu'il s'agit de sa toute première réception,
 - il affecte son pointeur $Père$ à q et
 - relaie le message à tous ses voisins sauf q .
3. Enfin dans tous les cas, p teste s'il a terminé sa participation au PIR :
 si Cpt est supérieur ou égal à son degré δ_p .

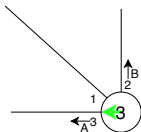
Dans ce cas,

- si p est l'initiateur, il décide,
- sinon il envoie un accusé réception Ack à son père (phase de retour).

Trace exemple

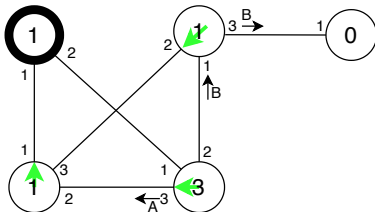


Vue locale :

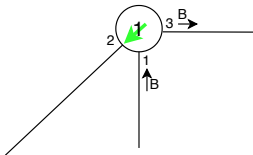


- Lorsqu'un processus p reçoit un message Brd de q :
 1. Il incrémente son compteur.
 2. Ensuite, si p est un suiveur et qu'il s'agit de sa toute première réception,
 - il affecte son pointeur $Père$ à q et
 - relaie le message à tous ses voisins sauf q .
 3. Enfin dans tous les cas, p teste s'il a terminé sa participation au PIR :
si Cpt est supérieur ou égal à son degré δ_p .
- Dans ce cas,
- si p est l'initiateur, il décide,
 - sinon il envoie un accusé réception Ack à son père (phase de retour).

Trace exemple



Vue locale :

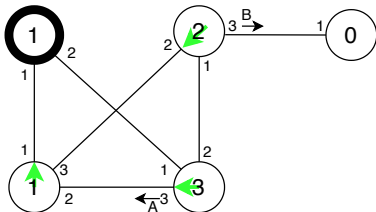


- Lorsqu'un processus p reçoit un message Brd de q :
 1. Il incrémente son compteur.
 2. Ensuite, si p est un suiveur et qu'il s'agit de sa toute première réception,
 - il affecte son pointeur $Père$ à q et
 - relaie le message à tous ses voisins sauf q .
 3. Enfin dans tous les cas, p teste s'il a terminé sa participation au PIR :
si Cpt est supérieur ou égal à son degré δ_p .

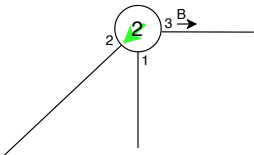
Dans ce cas,

 - si p est l'initiateur, il décide,
 - sinon il envoie un accusé réception Ack à son père (phase de retour).

Trace exemple

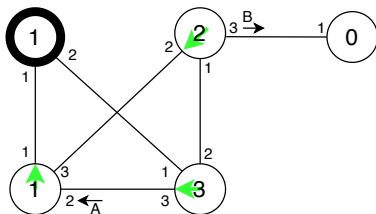


Vue locale :

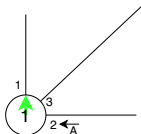


- Lorsqu'un processus p reçoit un message Brd de q :
 1. Il incrémente son compteur.
 2. Ensuite, si p est un suiveur et qu'il s'agit de sa toute première réception,
 - il affecte son pointeur $Père$ à q et
 - relaie le message à tous ses voisins sauf q .
 3. Enfin dans tous les cas, p teste s'il a terminé sa participation au PIR :
si Cpt est supérieur ou égal à son degré δ_p .
- Dans ce cas,
- si p est l'initiateur, il décide,
 - sinon il envoie un accusé réception Ack à son père (phase de retour).

Trace exemple



Vue locale :



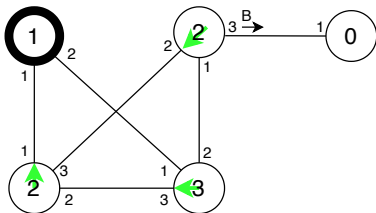
• Lorsqu'un processus p reçoit un message Ack de q :

1. Il incrémente son compteur.
2. Puis, il teste s'il a terminé sa participation au PIR.

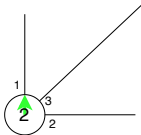
Dans ce cas,

- si p est l'initiateur, il décide,
- sinon il envoie un accusé réception Ack à son père.

Trace exemple

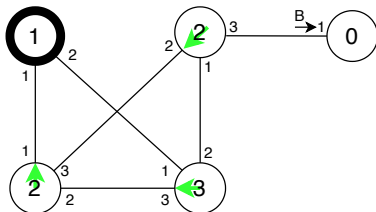


Vue locale :

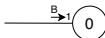


- Lorsqu'un processus p reçoit un message *Ack* de q :
 1. Il incrémente son compteur.
 2. Puis, il teste s'il a terminé sa participation au PIR.
- Dans ce cas,
- si p est l'initiateur, il décide,
 - sinon il envoie un accusé réception *Ack* à son père.

Trace exemple

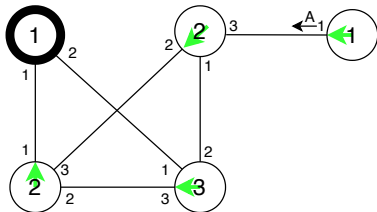


Vue locale :

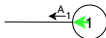


- Lorsqu'un processus p reçoit un message Brd de q :
 1. Il incrémente son compteur.
 2. Ensuite, si p est un suiveur et qu'il s'agit de sa toute première réception,
 - il affecte son pointeur $Père$ à q et
 - relaie le message à tous ses voisins sauf q .
 3. Enfin dans tous les cas, p teste s'il a terminé sa participation au PIR :
 si Cpt est supérieur ou égal à son degré δ_p .
 Dans ce cas,
 - si p est l'initiateur, il décide,
 - sinon il envoie un accusé réception Ack à son père (phase de retour).

Trace exemple



Vue locale :

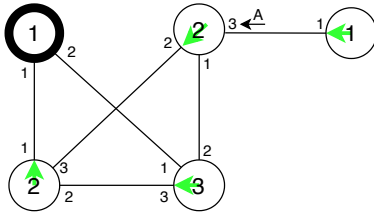


- Lorsqu'un processus p reçoit un message Brd de q :
 1. Il incrémente son compteur.
 2. Ensuite, si p est un suiveur et qu'il s'agit de sa toute première réception,
 - il affecte son pointeur $Père$ à q et
 - relaie le message à tous ses voisins sauf q .
 3. Enfin dans tous les cas, p teste s'il a terminé sa participation au PIR :
si Cpt est supérieur ou égal à son degré δ_p .

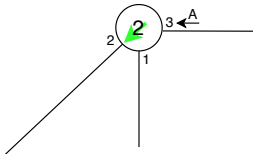
Dans ce cas,

 - si p est l'initiateur, il décide,
 - sinon il envoie un accusé réception Ack à son père (phase de retour).

Trace exemple

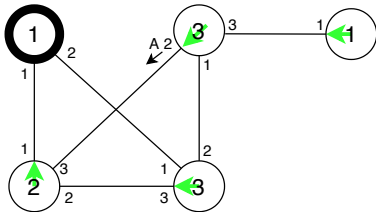


Vue locale :

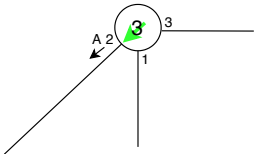


- Lorsqu'un processus p reçoit un message Ack de q :
 1. Il incrémente son compteur.
 2. Puis, il teste s'il a terminé sa participation au PIR.Dans ce cas,
 - si p est l'initiateur, il décide,
 - sinon il envoie un accusé réception Ack à son père.

Trace exemple

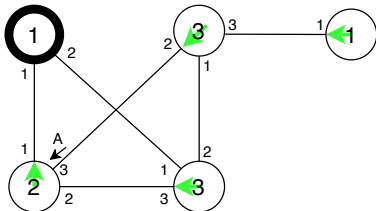


Vue locale :

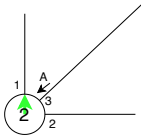


- Lorsqu'un processus p reçoit un message Ack de q :
 1. Il incrémente son compteur.
 2. Puis, il teste s'il a terminé sa participation au PIR.Dans ce cas,
 - si p est l'initiateur, il décide,
 - sinon il envoie un accusé réception Ack à son père.

Trace exemple

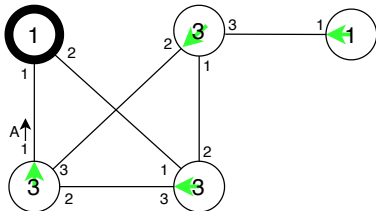


Vue locale :

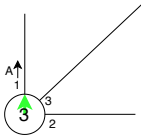


- Lorsqu'un processus p reçoit un message Ack de q :
 1. Il incrémente son compteur.
 2. Puis, il teste s'il a terminé sa participation au PIR.Dans ce cas,
 - si p est l'initiateur, il décide,
 - sinon il envoie un accusé réception Ack à son père.

Trace exemple

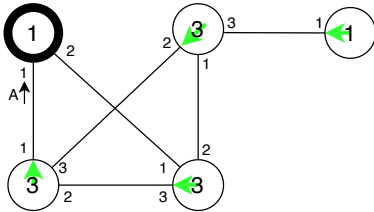


Vue locale :

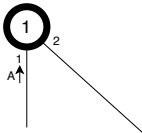


- Lorsqu'un processus p reçoit un message *Ack* de q :
 1. Il incrémente son compteur.
 2. Puis, il teste s'il a terminé sa participation au PIR.
- Dans ce cas,
- si p est l'initiateur, il décide,
 - sinon il envoie un accusé réception *Ack* à son père.

Trace exemple

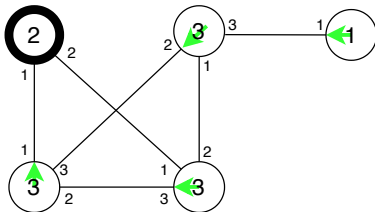


Vue locale :

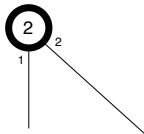


- Lorsqu'un processus p reçoit un message Ack de q :
 1. Il incrémente son compteur.
 2. Puis, il teste s'il a terminé sa participation au PIR.
- Dans ce cas,
- si p est l'initiateur, il décide,
 - sinon il envoie un accusé réception Ack à son père.

Trace exemple



Vue locale :



- Lorsqu'un processus p reçoit un message Ack de q :
 1. Il incrémente son compteur.
 2. Puis, il teste s'il a terminé sa participation au PIR.
- Dans ce cas,
- si p est l'initiateur, il décide,
 - sinon il envoie un accusé réception Ack à son père.

- Un pointeur de canal *Père*, initialisé à $\perp$.

L'initiateur devra fixer son pointeur à T .

Pour chaque suiveur, *Père* désignera l'arête incidente le reliant à son père dans l'arbre couvrant construit durant la première phase de l'algorithme (la phase de diffusion).

- Une variable *Cpt*, initialisée à 0.

Ce compteur contiendra le nombre de messages reçus par le processus.

Variables

1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$

2: $Cpt \in \mathbb{N}$ initialisé à 0

- Un pointeur de canal $Père$, initialisé à $\perp$.

L'initiateur devra fixer son pointeur à $\top$.

Pour chaque suiveur, $Père$ désignera l'arête incidente le reliant à son père dans l'arbre couvrant construit durant la première phase de l'algorithme (la phase de diffusion).

- Une variable Cpt , initialisée à 0.

Ce compteur contiendra le nombre de messages reçus par le processus.

Variables

1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$

2: $Cpt \in \mathbb{N}$ initialisé à 0

- L'initiateur démarre la vague de PIR lorsqu'il a une donnée d à diffuser.

Dans ce cas, il **initie la phase de diffusion** en envoyant un message Brd contenant d à tous ses voisins.

Variables

1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$

2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

3: $Père \leftarrow \top$

4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**

5: Envoyer $\langle Brd, d \rangle$ à q

6: **Fin Pour**

- L'initiateur démarre la vague de PIR lorsqu'il a une donnée d à diffuser.

Dans ce cas, il **initie la phase de diffusion** en envoyant un message Brd contenant d à tous ses voisins.

Variables

- 1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$
- 2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

- 3: $Père \leftarrow \top$
- 4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**
- 5: Envoyer $\langle Brd, d \rangle$ à q
- 6: **Fin Pour**

- Lorsqu'un processus p reçoit un message Brd de q :
 1. Il incrémente son compteur.
 2. Ensuite, si p est un suiveur et qu'il s'agit de sa toute première réception,
 - il affecte son pointeur $Père$ à q et
 - relaie le message à tous ses voisins sauf q .
 3. Enfin dans tous les cas, p teste s'il a terminé sa participation au PIR :
si Cpt est supérieur ou égal à son degré δ_p .
Dans ce cas,
 - si p est l'initiateur, il décide,
 - sinon il envoie un accusé réception Ack à son père (phase de retour).

Variables

- 1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$
- 2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

- 3: $Père \leftarrow \top$
- 4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**
- 5: Envoyer $\langle Brd, d \rangle$ à q
- 6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

- 7: $Cpt \leftarrow Cpt + 1$
- 8: **Si** $Père = \perp$ **alors**
- 9: $Père \leftarrow q$
- 10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**
- 11: Envoyer $\langle Brd, d \rangle$ à v
- 12: **Fin Pour**
- 13: **Fin Si**
- 14: Retour()

Fonction Retour()

- 17: **Si** $Cpt \geq \delta_p$ **alors**
- 18: **Si** $Père = \top$ **alors**
- 19: décide
- 20: **Sinon**
- 21: Envoyer $\langle Ack \rangle$ à $Père$
- 22: **Fin Si**
- 23: **Fin Si**

● Lorsqu'un processus p reçoit un message Brd de q :

1. Il incrémente son compteur.
2. Ensuite, si p est un suiveur et qu'il s'agit de sa toute première réception,
 - il affecte son pointeur $Père$ à q et
 - relaie le message à tous ses voisins sauf q .
3. Enfin dans tous les cas, p teste s'il a terminé sa participation au PIR :
si Cpt est supérieur ou égal à son degré δ_p .

Dans ce cas,

- si p est l'initiateur, il décide,
- sinon il envoie un accusé réception Ack à son père (phase de retour).

Variables

- 1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$
- 2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

- 3: $Père \leftarrow \top$
- 4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**
- 5: Envoyer $\langle Brd, d \rangle$ à q
- 6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

- 7: $Cpt \leftarrow Cpt + 1$
- 8: **Si** $Père = \perp$ **alors**
- 9: $Père \leftarrow q$
- 10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**
- 11: Envoyer $\langle Brd, d \rangle$ à v
- 12: **Fin Pour**
- 13: **Fin Si**
- 14: Retour()

Fonction Retour()

- 17: **Si** $Cpt \geq \delta_p$ **alors**
- 18: **Si** $Père = \top$ **alors**
- 19: décide
- 20: **Sinon**
- 21: Envoyer $\langle Ack \rangle$ à $Père$
- 22: **Fin Si**
- 23: **Fin Si**

- Lorsqu'un processus p reçoit un message *Ack* de q :

1. Il incrémente son compteur.
2. Puis, il teste s'il a terminé sa participation au PIR.

Dans ce cas,

- si p est l'initiateur, il décide,
- sinon il envoie un accusé réception *Ack* à son père.

Variables

- 1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$
- 2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

- 3: $Père \leftarrow \top$
- 4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**
- 5: Envoyer $\langle Brd, d \rangle$ à q
- 6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

- 7: $Cpt \leftarrow Cpt + 1$
- 8: **Si** $Père = \perp$ **alors**
- 9: $Père \leftarrow q$
- 10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**
- 11: Envoyer $\langle Brd, d \rangle$ à v
- 12: **Fin Pour**
- 13: **Fin Si**
- 14: Retour()

Réception de $\langle Ack \rangle$ de q

- 15: $Cpt \leftarrow Cpt + 1$
- 16: Retour()

Fonction Retour()

- 17: **Si** $Cpt \geq \delta_p$ **alors**
- 18: **Si** $Père = \top$ **alors**
- 19: décide
- 20: **Sinon**
- 21: Envoyer $\langle Ack \rangle$ à $Père$
- 22: **Fin Si**
- 23: **Fin Si**

- Lorsqu'un processus p reçoit un message Ack de q :

1. Il incrémente son compteur.
2. Puis, il teste s'il a terminé sa participation au PIR.

Dans ce cas,

- si p est l'initiateur, il décide,
- sinon il envoie un accusé réception Ack à son père.

Plan

- 1 Modèle
- 2 Principe
- 3 Spécification
- 4 Algorithme
- 5 Correction et complexité de l'algorithme

Question 1. Justifiez pourquoi tous les processus suivants reçoivent la donnée d .

Variables

- 1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$
- 2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

- 3: $Père \leftarrow \top$
- 4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**
- 5: Envoyer $\langle Brd, d \rangle$ à q
- 6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

- 7: $Cpt \leftarrow Cpt + 1$
- 8: **Si** $Père = \perp$ **alors**
- 9: $Père \leftarrow q$
- 10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**
- 11: Envoyer $\langle Brd, d \rangle$ à v
- 12: **Fin Pour**
- 13: **Fin Si**
- 14: Retour ()

Réception de $\langle Ack \rangle$ de q

- 15: $Cpt \leftarrow Cpt + 1$
- 16: Retour ()

Fonction Retour ()

- 17: **Si** $Cpt \geq \delta_p$ **alors**
- 18: **Si** $Père = \top$ **alors**
- 19: décide
- 20: **Sinon**
- 21: Envoyer $\langle Ack \rangle$ à $Père$
- 22: **Fin Si**
- 23: **Fin Si**

Question 1. Justifiez pourquoi tous les processus suiveurs reçoivent la donnée d .

Supposons, par contradiction, qu'un processus suiveur ne reçoive aucun message Brd .

Variables

- 1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$
- 2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

- 3: $Père \leftarrow \top$
- 4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**
- 5: Envoyer $\langle Brd, d \rangle$ à q
- 6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

- 7: $Cpt \leftarrow Cpt + 1$
- 8: **Si** $Père = \perp$ **alors**
- 9: $Père \leftarrow q$
- 10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**
- 11: Envoyer $\langle Brd, d \rangle$ à v
- 12: **Fin Pour**
- 13: **Fin Si**
- 14: Retour ()

Réception de $\langle Ack \rangle$ de q

- 15: $Cpt \leftarrow Cpt + 1$
- 16: Retour ()

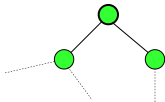
Fonction Retour ()

- 17: **Si** $Cpt \geq \delta_p$ **alors**
- 18: **Si** $Père = \top$ **alors**
- 19: décide
- 20: **Sinon**
- 21: Envoyer $\langle Ack \rangle$ à $Père$
- 22: **Fin Si**
- 23: **Fin Si**

Question 1. Justifiez pourquoi tous les processus suiveurs reçoivent la donnée d .

Supposons, par contradiction, qu'un processus suiveur ne reçoive aucun message Brd .

A l'initialisation, l'initiateur envoie Brd à tous ses voisins (qui sont suiveurs). Donc, tous les voisins de l'initiateur reçoivent Brd .



Variables

- 1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$
- 2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

- 3: $Père \leftarrow \top$
- 4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**
- 5: Envoyer $\langle Brd, d \rangle$ à q
- 6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

- 7: $Cpt \leftarrow Cpt + 1$
- 8: **Si** $Père = \perp$ **alors**
- 9: $Père \leftarrow q$
- 10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**
- 11: Envoyer $\langle Brd, d \rangle$ à v
- 12: **Fin Pour**
- 13: **Fin Si**
- 14: **Retour** ()

Réception de $\langle Ack \rangle$ de q

- 15: $Cpt \leftarrow Cpt + 1$
- 16: **Retour** ()

Fonction **Retour** ()

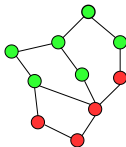
- 17: **Si** $Cpt \geq \delta_p$ **alors**
- 18: **Si** $Père = \top$ **alors**
- 19: décide
- 20: **Sinon**
- 21: Envoyer $\langle Ack \rangle$ à $Père$
- 22: **Fin Si**
- 23: **Fin Si**

Question 1. Justifiez pourquoi tous les processus suiveurs reçoivent la donnée d .

Supposons, par contradiction, qu'un processus suiveur ne reçoive aucun message Brd .

A l'initialisation, l'initiateur envoie Brd à tous ses voisins (qui sont suiveurs). Donc, tous les voisins de l'initiateur reçoivent Brd .

Comme le réseau est connexe, il existe au moins un processus suiveur p qui reçoit Brd et qui est voisin d'un processus suiveur q qui ne le reçoit pas.



Variables

- 1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, T\}$ initialisé à $\perp$
- 2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

- 3: $Père \leftarrow T$
- 4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**
- 5: Envoyer $\langle Brd, d \rangle$ à q
- 6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

- 7: $Cpt \leftarrow Cpt + 1$
- 8: **Si** $Père = \perp$ **alors**
- 9: $Père \leftarrow q$
- 10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**
- 11: Envoyer $\langle Brd, d \rangle$ à v
- 12: **Fin Pour**
- 13: **Fin Si**
- 14: Retour ()

Réception de $\langle Ack \rangle$ de q

- 15: $Cpt \leftarrow Cpt + 1$
- 16: Retour ()

Fonction Retour ()

- 17: **Si** $Cpt \geq \delta_p$ **alors**
- 18: **Si** $Père = T$ **alors**
- 19: décide
- 20: **Sinon**
- 21: Envoyer $\langle Ack \rangle$ à $Père$
- 22: **Fin Si**
- 23: **Fin Si**

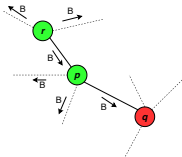
Question 1. Justifiez pourquoi tous les processus suiveurs reçoivent la donnée d .

Supposons, par contradiction, qu'un processus suiveur ne reçoive aucun message Brd .

A l'initialisation, l'initiateur envoie Brd à tous ses voisins (qui sont suiveurs). Donc, tous les voisins de l'initiateur reçoivent Brd .

Comme le réseau est connexe, il existe au moins un processus suiveur p qui reçoit Brd et qui est voisin d'un processus suiveur q qui ne le reçoit pas.

Soit r le processus émetteur du message Brd que p reçoit en premier. Soit r est l'initiateur, soit r a reçu Brd au préalable. Donc $r \neq q$. Or, lorsque p reçoit pour la première fois Brd , p envoie Brd à tous ses voisins sauf r . Donc q finit par recevoir Brd , contradiction.



Variables

- 1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$
- 2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

- 3: $Père \leftarrow \top$
- 4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**
- 5: Envoyer $\langle Brd, d \rangle$ à q
- 6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

- 7: $Cpt \leftarrow Cpt + 1$
- 8: **Si** $Père = \perp$ **alors**
- 9: $Père \leftarrow q$
- 10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**
- 11: Envoyer $\langle Brd, d \rangle$ à v
- 12: **Fin Pour**
- 13: **Fin Si**
- 14: **Retour** ()

□

Réception de $\langle Ack \rangle$ de q

- 15: $Cpt \leftarrow Cpt + 1$
- 16: **Retour** ()

Fonction **Retour** ()

- 17: **Si** $Cpt \geq \delta_p$ **alors**
- 18: **Si** $Père = \top$ **alors**
- 19: décide
- 20: **Sinon**
- 21: Envoyer $\langle Ack \rangle$ à $Père$
- 22: **Fin Si**
- 23: **Fin Si**

D'après la question précédente, tous les processus suiveurs finissent par affecter définitivement leur variable *Père* à un numéro de canal.

De plus, la variable *Père* de l'initiateur ne contient jamais un numéro de canal.

D'après la question précédente, tous les processus suiveurs finissent par affecter définitivement leur variable *Père* à un numéro de canal.

De plus, la variable *Père* de l'initiateur ne contient jamais un numéro de canal.

Dans la suite, on dira que *p désigne q comme père* si la variable *Père* de *p* pointe le canal reliant *p* à *q*.

D'après la question précédente, tous les processus suiveurs finissent par affecter définitivement leur variable *Père* à un numéro de canal.

De plus, la variable *Père* de l'initiateur ne contient jamais un numéro de canal.

Dans la suite, on dira que *p désigne q comme père* si la variable *Père* de *p* pointe le canal reliant *p* à *q*.

Soit $T = (V, E)$ un graphe orienté où

- V est l'ensemble des processus et
- E est l'ensemble des arcs tels que $(p, q) \in E$ si et seulement si $p, q \in V$ et p finit par désigner q comme père.

PIR et arbre couvrant

D'après la question précédente, tous les processus suiveurs finissent par affecter définitivement leur variable *Père* à un numéro de canal.

De plus, la variable *Père* de l'initiateur ne contient jamais un numéro de canal.

Dans la suite, on dira que *p désigne q comme père* si la variable *Père* de *p* pointe le canal reliant *p* à *q*.

Soit $T = (V, E)$ un graphe orienté où

- V est l'ensemble des processus et
- E est l'ensemble des arcs tels que $(p, q) \in E$ si et seulement si *p*, *q* $\in V$ et *p* finit par désigner *q* comme père.

Nous avons les deux propriétés suivantes :

- 1 $|E| = n - 1$ (par définition).
- 2 Pour tout processus *p*, il existe un chemin (orienté) dans T de *p* vers l'initiateur.

D'où

T est un arbre couvrant orienté enraciné à l'initiateur.

PIR et arbre couvrant

D'après la question précédente, tous les processus suiveurs finissent par affecter définitivement leur variable *Père* à un numéro de canal.

De plus, la variable *Père* de l'initiateur ne contient jamais un numéro de canal.

Dans la suite, on dira que *p désigne q comme père* si la variable *Père* de *p* pointe le canal reliant *p* à *q*.

Soit $T = (V, E)$ un graphe orienté où

- V est l'ensemble des processus et
- E est l'ensemble des arcs tels que $(p, q) \in E$ si et seulement si *p*, *q* $\in V$ et *p* finit par désigner *q* comme père.

Nous avons les deux propriétés suivantes :

- 1 $|E| = n - 1$ (par définition).
- 2 Pour tout processus *p*, il existe un chemin (orienté) dans T de *p* vers l'initiateur.

D'où

T est un arbre couvrant orienté enraciné à l'initiateur.

Démontrons la deuxième propriété par récurrence.

Pour tout processus p , il existe un chemin (orienté) dans T de p à l'initiateur.

Soit $p_1, \dots, p_n$ un ordre total sur les processus tel que p_1 est l'initiateur et pour tout suiveurs p_i et p_j , on a $i < j$ si p_i a reçu pour la première fois un message *Brd* avant p_j .

D'après la question précédente, cet ordre est bien défini.

Pour tout processus p , il existe un chemin (orienté) dans T de p à l'initiateur.

Soit $p_1, \dots, p_n$ un ordre total sur les processus tel que p_1 est l'initiateur et pour tout suiveurs p_i et p_j , on a $i < j$ si p_i a reçu pour la première fois un message *Brd* avant p_j .

D'après la question précédente, cet ordre est bien défini.

Prouvons par *récurrence sur i* que pour tout processus p_i avec $i \in [1..n]$, *il existe un chemin (orienté) dans T de p_i à l'initiateur.*

Pour tout processus p , il existe un chemin (orienté) dans T de p à l'initiateur.

Soit $p_1, \dots, p_n$ un ordre total sur les processus tel que p_1 est l'initiateur et pour tout suiveurs p_i et p_j , on a $i < j$ si p_i a reçu pour la première fois un message *Brd* avant p_j .

D'après la question précédente, cet ordre est bien défini.

Prouvons par *récurrence sur i* que pour tout processus p_i avec $i \in [1..n]$, *il existe un chemin (orienté) dans T de p_i à l'initiateur.*

- Le cas de base $i = 1$ est trivial : le chemin vide relie l'initiateur p_1 à lui même (dans T).

Pour tout processus p , il existe un chemin (orienté) dans T de p à l'initiateur.

Soit $p_1, \dots, p_n$ un ordre total sur les processus tel que p_1 est l'initiateur et pour tout suiveurs p_i et p_j , on a $i < j$ si p_i a reçu pour la première fois un message *Brd* avant p_j .

D'après la question précédente, cet ordre est bien défini.

Prouvons par *récurrence sur i* que pour tout processus p_i avec $i \in [1..n]$, *il existe un chemin (orienté) dans T de p_i à l'initiateur.*

- Le cas de base $i = 1$ est trivial : le chemin vide relie l'initiateur p_1 à lui même (dans T).
- Soit $i > 1$. Puisque p_i est un suiveur, posons p_j *le père de p_i* .

Pour tout processus p , il existe un chemin (orienté) dans T de p à l'initiateur.

Soit $p_1, \dots, p_n$ un ordre total sur les processus tel que p_1 est l'initiateur et pour tout suiveurs p_i et p_j , on a $i < j$ si p_i a reçu pour la première fois un message Brd avant p_j .

D'après la question précédente, cet ordre est bien défini.

Prouvons par **récurrence sur i** que pour tout processus p_i avec $i \in [1..n]$, **il existe un chemin (orienté) dans T de p_i à l'initiateur.**

- Le cas de base $i = 1$ est trivial : le chemin vide relie l'initiateur p_1 à lui même (dans T).
- Soit $i > 1$. Puisque p_i est un suiveur, posons p_j **le père de p_i** .

p_j est

- soit l'initiateur,
- soit un suiveur qui a reçu Brd pour la première fois avant p_i .

Pour tout processus p , il existe un chemin (orienté) dans T de p à l'initiateur.

Soit $p_1, \dots, p_n$ un ordre total sur les processus tel que p_1 est l'initiateur et pour tout suiveurs p_i et p_j , on a $i < j$ si p_i a reçu pour la première fois un message Brd avant p_j .

D'après la question précédente, cet ordre est bien défini.

Prouvons par **récurrence sur i** que pour tout processus p_i avec $i \in [1..n]$, **il existe un chemin (orienté) dans T de p_i à l'initiateur.**

- Le cas de base $i = 1$ est trivial : le chemin vide relie l'initiateur p_1 à lui même (dans T).
- Soit $i > 1$. Puisque p_i est un suiveur, posons p_j **le père de p_i** .

p_j est

- soit l'initiateur,
- soit un suiveur qui a reçu Brd pour la première fois avant p_i .

Par définition de l'ordre, on a $j < i$.

Pour tout processus p , il existe un chemin (orienté) dans T de p à l'initiateur.

Soit $p_1, \dots, p_n$ un ordre total sur les processus tel que p_1 est l'initiateur et pour tout suiveurs p_i et p_j , on a $i < j$ si p_i a reçu pour la première fois un message *Brd* avant p_j .

D'après la question précédente, cet ordre est bien défini.

Prouvons par **récurrence sur i** que pour tout processus p_i avec $i \in [1..n]$, **il existe un chemin (orienté) dans T de p_i à l'initiateur.**

- Le cas de base $i = 1$ est trivial : le chemin vide relie l'initiateur p_1 à lui même (dans T).
- Soit $i > 1$. Puisque p_i est un suiveur, posons p_j **le père de p_i** .

p_j est

- soit l'initiateur,
- soit un suiveur qui a reçu *Brd* pour la première fois avant p_i .

Par définition de l'ordre, on a $j < i$.

Donc, **par hypothèse de récurrence**, il existe **un chemin (orienté) P dans T de p_j à l'initiateur.**

Or, par définition, l'arc (p_i, p_j) appartient à E .

Donc, **p_i, P est un chemin (orienté) dans T de p_i à l'initiateur** et la récurrence est vérifiée.

□

Question 2. Justifiez pourquoi tout processus envoie au moins un message à chacun de ses voisins.

Variables

- 1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$
- 2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

- 3: $Père \leftarrow \top$
- 4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**
- 5: Envoyer $\langle Brd, d \rangle$ à q
- 6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

- 7: $Cpt \leftarrow Cpt + 1$
- 8: **Si** $Père = \perp$ **alors**
- 9: $Père \leftarrow q$
- 10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**
- 11: Envoyer $\langle Brd, d \rangle$ à v
- 12: **Fin Pour**
- 13: **Fin Si**
- 14: Retour ()

Réception de $\langle Ack \rangle$ de q

- 15: $Cpt \leftarrow Cpt + 1$
- 16: Retour ()

Fonction Retour ()

- 17: **Si** $Cpt \geq \delta_p$ **alors**
- 18: **Si** $Père = \top$ **alors**
- 19: décide
- 20: **Sinon**
- 21: Envoyer $\langle Ack \rangle$ à $Père$
- 22: **Fin Si**
- 23: **Fin Si**

Question 2. Justifiez pourquoi tout processus envoie au moins un message à chacun de ses voisins.

Supposons que p n'envoie pas de message à un voisin.

Variables

- 1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$
- 2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

- 3: $Père \leftarrow \top$
- 4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**
- 5: Envoyer $\langle Brd, d \rangle$ à q
- 6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

- 7: $Cpt \leftarrow Cpt + 1$
- 8: **Si** $Père = \perp$ **alors**
- 9: $Père \leftarrow q$
- 10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**
- 11: Envoyer $\langle Brd, d \rangle$ à v
- 12: **Fin Pour**
- 13: **Fin Si**
- 14: Retour ()

Réception de $\langle Ack \rangle$ de q

- 15: $Cpt \leftarrow Cpt + 1$
- 16: Retour ()

Fonction Retour ()

- 17: **Si** $Cpt \geq \delta_p$ **alors**
- 18: **Si** $Père = \top$ **alors**
- 19: décide
- 20: **Sinon**
- 21: Envoyer $\langle Ack \rangle$ à $Père$
- 22: **Fin Si**
- 23: **Fin Si**

Question 2. Justifiez pourquoi tout processus envoie au moins un message à chacun de ses voisins.

Supposons que p n'envoie pas de message à un voisin.
 p n'est pas l'initiateur.

Variables

- 1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$
- 2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

- 3: $Père \leftarrow \top$
- 4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**
- 5: Envoyer $\langle Brd, d \rangle$ à q
- 6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

- 7: $Cpt \leftarrow Cpt + 1$
- 8: **Si** $Père = \perp$ **alors**
- 9: $Père \leftarrow q$
- 10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**
- 11: Envoyer $\langle Brd, d \rangle$ à v
- 12: **Fin Pour**
- 13: **Fin Si**
- 14: Retour ()

Réception de $\langle Ack \rangle$ de q

- 15: $Cpt \leftarrow Cpt + 1$
- 16: Retour ()

Fonction Retour ()

- 17: **Si** $Cpt \geq \delta_p$ **alors**
- 18: **Si** $Père = \top$ **alors**
- 19: décide
- 20: **Sinon**
- 21: Envoyer $\langle Ack \rangle$ à $Père$
- 22: **Fin Si**
- 23: **Fin Si**

Question 2. Justifiez pourquoi tout processus envoie au moins un message à chacun de ses voisins.

Supposons que p n'envoie pas de message à un voisin.

p n'est pas l'initiateur. D'après la question 1, p finit par recevoir Brd . Lorsqu'il reçoit Brd pour la 1^{ère} fois, il le relaie à tous ses voisins, sauf son père. Donc p n'envoie aucun message à son père.

Variables

- 1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$
- 2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

- 3: $Père \leftarrow \top$
- 4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**
- 5: Envoyer $\langle Brd, d \rangle$ à q
- 6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

- 7: $Cpt \leftarrow Cpt + 1$
- 8: **Si** $Père = \perp$ **alors**
- 9: $Père \leftarrow q$
- 10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**
- 11: Envoyer $\langle Brd, d \rangle$ à v
- 12: **Fin Pour**
- 13: **Fin Si**
- 14: Retour ()

Réception de $\langle Ack \rangle$ de q

- 15: $Cpt \leftarrow Cpt + 1$
- 16: Retour ()

Fonction Retour ()

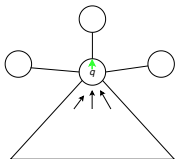
- 17: **Si** $Cpt \geq \delta_p$ **alors**
- 18: **Si** $Père = \top$ **alors**
- 19: décide
- 20: **Sinon**
- 21: Envoyer $\langle Ack \rangle$ à $Père$
- 22: **Fin Si**
- 23: **Fin Si**

Question 2. Justifiez pourquoi tout processus envoie au moins un message à chacun de ses voisins.

Supposons que p n'envoie pas de message à un voisin.

p n'est pas l'initiateur. D'après la question 1, p finit par recevoir Brd . Lorsqu'il reçoit Brd pour la 1^{ère} fois, il le relaie à tous ses voisins, sauf son père. Donc p n'envoie aucun message à son père.

Il existe un suiveur q qui n'envoie aucun message à son père mais tel que tous ses fils dans T le font (il s'agit alors de messages Ack , d'après le code de l'algorithme).



Variables

- 1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$
- 2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

- 3: $Père \leftarrow \top$
- 4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**
- 5: Envoyer $\langle Brd, d \rangle$ à q
- 6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

- 7: $Cpt \leftarrow Cpt + 1$
- 8: **Si** $Père = \perp$ **alors**
- 9: $Père \leftarrow q$
- 10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**
- 11: Envoyer $\langle Brd, d \rangle$ à v
- 12: **Fin Pour**
- 13: **Fin Si**
- 14: **Retour** ()

Réception de $\langle Ack \rangle$ de q

- 15: $Cpt \leftarrow Cpt + 1$
- 16: **Retour** ()

Fonction **Retour** ()

- 17: **Si** $Cpt \geq \delta_p$ **alors**
- 18: **Si** $Père = \top$ **alors**
- 19: décide
- 20: **Sinon**
- 21: Envoyer $\langle Ack \rangle$ à $Père$
- 22: **Fin Si**
- 23: **Fin Si**

Question 2. Justifiez pourquoi tout processus envoie au moins un message à chacun de ses voisins.

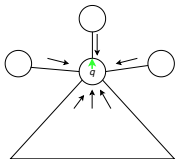
Supposons que p n'envoie pas de message à un voisin.

p n'est pas l'initiateur. D'après la question 1, p finit par recevoir Brd . Lorsqu'il reçoit Brd pour la 1^{ère} fois, il le relaie à tous ses voisins, sauf son père. Donc p n'envoie aucun message à son père.

Il existe un suiveur q qui n'envoie aucun message à son père mais tel que tous ses fils dans T le font (il s'agit alors de messages Ack , d'après le code de l'algorithme).

Tous les voisins de q qui ne sont pas ses fils dans T (dont son père) finissent par recevoir Brd (Question 1) et dans ce cas relaient Brd en particulier à q , puisqu'il n'est pas leur père.

Donc q reçoit au moins un message de chacun de ses voisins et ainsi finit par envoyer un message Ack à son père, contradiction.



Variables

1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, T\}$ initialisé à $\perp$

2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

3: $Père \leftarrow T$

4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**

5: Envoyer $\langle Brd, d \rangle$ à q

6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

7: $Cpt \leftarrow Cpt + 1$

8: **Si** $Père = \perp$ **alors**

9: $Père \leftarrow q$

10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**

11: Envoyer $\langle Brd, d \rangle$ à v

12: **Fin Pour**

13: **Fin Si**

14: **Retour** ()

Réception de $\langle Ack \rangle$ de q

15: $Cpt \leftarrow Cpt + 1$

□ 16: **Retour** ()

Fonction **Retour** ()

17: **Si** $Cpt \geq \delta_p$ **alors**

18: **Si** $Père = T$ **alors**

19: décide

20: **Sinon**

21: Envoyer $\langle Ack \rangle$ à $Père$

22: **Fin Si**

23: **Fin Si**

Question 3. Justifiez pourquoi l'initiateur finit par décider.

Variables

1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$

2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

3: $Père \leftarrow \top$

4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**

5: Envoyer $\langle Brd, d \rangle$ à q

6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

7: $Cpt \leftarrow Cpt + 1$

8: **Si** $Père = \perp$ **alors**

9: $Père \leftarrow q$

10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**

11: Envoyer $\langle Brd, d \rangle$ à v

12: **Fin Pour**

13: **Fin Si**

14: Retour()

Réception de $\langle Ack \rangle$ de q

15: $Cpt \leftarrow Cpt + 1$

16: Retour()

Fonction Retour()

17: **Si** $Cpt \geq \delta_p$ **alors**

18: **Si** $Père = \top$ **alors**

19: décide

20: **Sinon**

21: Envoyer $\langle Ack \rangle$ à $Père$

22: **Fin Si**

23: **Fin Si**

Question 3. Justifiez pourquoi l'initiateur finit par décider.

D'après la question précédente, l'initiateur i reçoit au moins δ_i messages. De plus, $\delta_i \geq 1$ (hypothèse du modèle).

Donc, d'après le code de l'algorithme, i finit par décider. $\square$

Variables

1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$

2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

3: $Père \leftarrow \top$

4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**

5: Envoyer $\langle Brd, d \rangle$ à q

6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

7: $Cpt \leftarrow Cpt + 1$

8: **Si** $Père = \perp$ **alors**

9: $Père \leftarrow q$

10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**

11: Envoyer $\langle Brd, d \rangle$ à v

12: **Fin Pour**

13: **Fin Si**

14: Retour()

Réception de $\langle Ack \rangle$ de q

15: $Cpt \leftarrow Cpt + 1$

16: Retour()

Fonction Retour()

17: **Si** $Cpt \geq \delta_p$ **alors**

18: **Si** $Père = \top$ **alors**

19: décide

20: **Sinon**

21: Envoyer $\langle Ack \rangle$ à $Père$

22: **Fin Si**

23: **Fin Si**

Question 4. Justifiez pourquoi tout processus suiveur accuse réception de d (Ack).

Variables

- 1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$
- 2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

- 3: $Père \leftarrow \top$
- 4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**
- 5: Envoyer $\langle Brd, d \rangle$ à q
- 6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

- 7: $Cpt \leftarrow Cpt + 1$
- 8: **Si** $Père = \perp$ **alors**
- 9: $Père \leftarrow q$
- 10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**
- 11: Envoyer $\langle Brd, d \rangle$ à v
- 12: **Fin Pour**
- 13: **Fin Si**
- 14: Retour ()

Réception de $\langle Ack \rangle$ de q

- 15: $Cpt \leftarrow Cpt + 1$
- 16: Retour ()

Fonction Retour ()

- 17: **Si** $Cpt \geq \delta_p$ **alors**
- 18: **Si** $Père = \top$ **alors**
- 19: décide
- 20: **Sinon**
- 21: Envoyer $\langle Ack \rangle$ à $Père$
- 22: **Fin Si**
- 23: **Fin Si**

Question 4. Justifiez pourquoi tout processus suiveur accuse réception de d (Ack).

D'après le code de l'algorithme, un processus suiveur n'envoie pas Brd à son père.

Or d'après la question 2, il envoie au moins un message à son père.

D'où, il envoie nécessairement Ack à son père.

Variables

- 1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$
- 2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

- 3: $Père \leftarrow \top$
- 4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**
- 5: Envoyer $\langle Brd, d \rangle$ à q
- 6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

- 7: $Cpt \leftarrow Cpt + 1$
- 8: **Si** $Père = \perp$ **alors**
- 9: $Père \leftarrow q$
- 10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**
- 11: Envoyer $\langle Brd, d \rangle$ à v
- 12: **Fin Pour**
- 13: **Fin Si**
- 14: Retour ()

Réception de $\langle Ack \rangle$ de q

- 15: $Cpt \leftarrow Cpt + 1$
- 16: Retour ()

Fonction Retour ()

- 17: **Si** $Cpt \geq \delta_p$ **alors**
- 18: **Si** $Père = \top$ **alors**
- 19: décide
- 20: **Sinon**
- 21: Envoyer $\langle Ack \rangle$ à $Père$
- 22: **Fin Si**
- 23: **Fin Si**

□

Question 5. Justifiez pourquoi l'exécution termine.

Variables

1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$

2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

3: $Père \leftarrow \top$

4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**

5: Envoyer $\langle Brd, d \rangle$ à q

6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

7: $Cpt \leftarrow Cpt + 1$

8: **Si** $Père = \perp$ **alors**

9: $Père \leftarrow q$

10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**

11: Envoyer $\langle Brd, d \rangle$ à v

12: **Fin Pour**

13: **Fin Si**

14: Retour()

Réception de $\langle Ack \rangle$ de q

15: $Cpt \leftarrow Cpt + 1$

16: Retour()

Fonction Retour()

17: **Si** $Cpt \geq \delta_p$ **alors**

18: **Si** $Père = \top$ **alors**

19: décide

20: **Sinon**

21: Envoyer $\langle Ack \rangle$ à $Père$

22: **Fin Si**

23: **Fin Si**

Question 5. Justifiez pourquoi l'exécution termine.

Tout processus envoie au plus un message à chacun de ses voisins.

Variables

1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$

2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

3: $Père \leftarrow \top$

4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**

5: Envoyer $\langle Brd, d \rangle$ à q

6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

7: $Cpt \leftarrow Cpt + 1$

8: **Si** $Père = \perp$ **alors**

9: $Père \leftarrow q$

10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**

11: Envoyer $\langle Brd, d \rangle$ à v

12: **Fin Pour**

13: **Fin Si**

14: Retour ()

Réception de $\langle Ack \rangle$ de q

15: $Cpt \leftarrow Cpt + 1$

16: Retour ()

Fonction Retour ()

17: **Si** $Cpt \geq \delta_p$ **alors**

18: **Si** $Père = \top$ **alors**

19: décide

20: **Sinon**

21: Envoyer $\langle Ack \rangle$ à $Père$

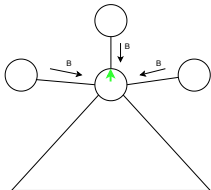
22: **Fin Si**

23: **Fin Si**

Question 5. Justifiez pourquoi l'exécution termine.

Tout processus envoie au plus un message à chacun de ses voisins.

1. L'initiateur envoie 1 message *Brd* à chacun de ses voisins.
Tout processus suiveur envoie 0 message *Brd* à son père dans *T* et 1 message *Brd* à ses autres voisins.



Variables

- 1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$
2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

- 3: $Père \leftarrow \top$
4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**
5: Envoyer $\langle Brd, d \rangle$ à q
6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

- 7: $Cpt \leftarrow Cpt + 1$
8: **Si** $Père = \perp$ **alors**
9: $Père \leftarrow q$
10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**
11: Envoyer $\langle Brd, d \rangle$ à v
12: **Fin Pour**
13: **Fin Si**
14: **Retour** ()

Réception de $\langle Ack \rangle$ de q

- 15: $Cpt \leftarrow Cpt + 1$
16: **Retour** ()

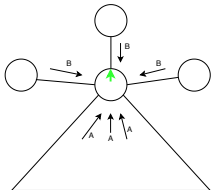
Fonction **Retour** ()

- 17: **Si** $Cpt \geq \delta_p$ **alors**
18: **Si** $Père = \top$ **alors**
19: décide
20: **Sinon**
21: Envoyer $\langle Ack \rangle$ à $Père$
22: **Fin Si**
23: **Fin Si**

Question 5. Justifiez pourquoi l'exécution termine.

Tout processus envoie au plus un message à chacun de ses voisins.

- 1 L'initiateur envoie 1 message *Brd* à chacun de ses voisins.
Tout processus suiveur envoie 0 message *Brd* à son père dans *T* et 1 message *Brd* à ses autres voisins.
- 2 Tout processus *p* envoie *Ack* uniquement à son père et seulement si *p* n'est pas initiateur.



Variables

- 1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$
- 2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

- 3: $Père \leftarrow \top$
- 4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**
- 5: Envoyer $\langle Brd, d \rangle$ à q
- 6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

- 7: $Cpt \leftarrow Cpt + 1$
- 8: **Si** $Père = \perp$ **alors**
- 9: $Père \leftarrow q$
- 10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**
- 11: Envoyer $\langle Brd, d \rangle$ à v
- 12: **Fin Pour**
- 13: **Fin Si**
- 14: Retour ()

Réception de $\langle Ack \rangle$ de q

- 15: $Cpt \leftarrow Cpt + 1$
- 16: Retour ()

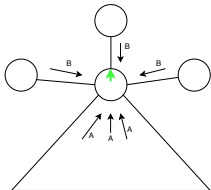
Fonction Retour ()

- 17: **Si** $Cpt \geq \delta_p$ **alors**
- 18: **Si** $Père = \top$ **alors**
- 19: décide
- 20: **Sinon**
- 21: Envoyer $\langle Ack \rangle$ à $Père$
- 22: **Fin Si**
- 23: **Fin Si**

Question 5. Justifiez pourquoi l'exécution termine.

Tout processus envoie au plus un message à chacun de ses voisins.

- 1 L'initiateur envoie 1 message *Brd* à chacun de ses voisins.
Tout processus suiveur envoie 0 message *Brd* à son père dans *T* et 1 message *Brd* à ses autres voisins.
- 2 Tout processus *p* envoie *Ack* uniquement à son père et seulement si *p* n'est pas initiateur.
- 3 Supposons, par contradiction, qu'un processus suiveur envoie *Ack* au moins deux fois à son père.



Variables

- 1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$
- 2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

- 3: $Père \leftarrow \top$
- 4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**
- 5: Envoyer $\langle Brd, d \rangle$ à q
- 6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

- 7: $Cpt \leftarrow Cpt + 1$
- 8: **Si** $Père = \perp$ **alors**
- 9: $Père \leftarrow q$
- 10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**
- 11: Envoyer $\langle Brd, d \rangle$ à v
- 12: **Fin Pour**
- 13: **Fin Si**
- 14: **Retour** ()

Réception de $\langle Ack \rangle$ de q

- 15: $Cpt \leftarrow Cpt + 1$
- 16: **Retour** ()

Fonction **Retour** ()

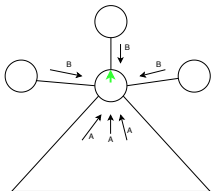
- 17: **Si** $Cpt \geq \delta_p$ **alors**
- 18: **Si** $Père = \top$ **alors**
- 19: décide
- 20: **Sinon**
- 21: Envoyer $\langle Ack \rangle$ à $Père$
- 22: **Fin Si**
- 23: **Fin Si**

Question 5. Justifiez pourquoi l'exécution termine.

Tout processus envoie au plus un message à chacun de ses voisins.

- 1 L'initiateur envoie 1 message *Brd* à chacun de ses voisins.
Tout processus suiveur envoie 0 message *Brd* à son père dans *T* et 1 message *Brd* à ses autres voisins.
- 2 Tout processus *p* envoie *Ack* uniquement à son père et seulement si *p* n'est pas initiateur.
- 3 Supposons, par contradiction, qu'un processus suiveur envoie *Ack* au moins deux fois à son père.

Il existe un processus suiveur *p* qui envoie deux fois *Ack* à son père, mais tel que chacun de ses fils lui envoie *Ack* au plus une fois.



Variables

- 1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$
- 2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

- 3: $Père \leftarrow \top$
- 4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**
- 5: Envoyer $\langle Brd, d \rangle$ à q
- 6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

- 7: $Cpt \leftarrow Cpt + 1$
- 8: **Si** $Père = \perp$ **alors**
- 9: $Père \leftarrow q$
- 10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**
- 11: Envoyer $\langle Brd, d \rangle$ à v
- 12: **Fin Pour**
- 13: **Fin Si**
- 14: Retour ()

Réception de $\langle Ack \rangle$ de q

- 15: $Cpt \leftarrow Cpt + 1$
- 16: Retour ()

Fonction Retour ()

- 17: **Si** $Cpt \geq \delta_p$ **alors**
- 18: **Si** $Père = \top$ **alors**
- 19: décide
- 20: **Sinon**
- 21: Envoyer $\langle Ack \rangle$ à $Père$
- 22: **Fin Si**
- 23: **Fin Si**

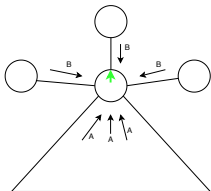
Question 5. Justifiez pourquoi l'exécution termine.

Tout processus envoie au plus un message à chacun de ses voisins.

- 1 L'initiateur envoie 1 message *Brd* à chacun de ses voisins.
Tout processus suiveur envoie 0 message *Brd* à son père dans *T* et 1 message *Brd* à ses autres voisins.
- 2 Tout processus *p* envoie *Ack* uniquement à son père et seulement si *p* n'est pas initiateur.
- 3 Supposons, par contradiction, qu'un processus suiveur envoie *Ack* au moins deux fois à son père.

Il existe un processus suiveur *p* qui envoie deux fois *Ack* à son père, mais tel que chacun de ses fils lui envoie *Ack* au plus une fois.

Les fils de *p* ne lui envoient jamais *Brd* (Point 1), donc *p* reçoit au plus 1 message de chacun de ses fils. Ensuite, les voisins non fils de *p* ne lui envoient jamais *Ack* (Point 2) et au plus 1 message *Brd* (Point 1). Donc, *p* reçoit au plus 1 message de chacun de ses voisins. Ainsi, *p* envoie au plus une fois *Ack* à son père, contradiction.



Variables

- 1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$
- 2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

- 3: $Père \leftarrow \top$
- 4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**
- 5: Envoyer $\langle Brd, d \rangle$ à q
- 6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

- 7: $Cpt \leftarrow Cpt + 1$
- 8: **Si** $Père = \perp$ **alors**
- 9: $Père \leftarrow q$
- 10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**
- 11: Envoyer $\langle Brd, d \rangle$ à v
- 12: **Fin Pour**
- 13: **Fin Si**
- 14: Retour ()

Réception de $\langle Ack \rangle$ de q

□

- 15: $Cpt \leftarrow Cpt + 1$
- 16: Retour ()

Fonction Retour ()

- 17: **Si** $Cpt \geq \delta_p$ **alors**
- 18: **Si** $Père = \top$ **alors**
- 19: décide
- 20: **Sinon**
- 21: Envoyer $\langle Ack \rangle$ à $Père$
- 22: **Fin Si**
- 23: **Fin Si**

Question 6. Justifiez pourquoi au plus une décision est prise.

Variables

1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$

2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

3: $Père \leftarrow \top$

4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**

5: Envoyer $\langle Brd, d \rangle$ à q

6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

7: $Cpt \leftarrow Cpt + 1$

8: **Si** $Père = \perp$ **alors**

9: $Père \leftarrow q$

10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**

11: Envoyer $\langle Brd, d \rangle$ à v

12: **Fin Pour**

13: **Fin Si**

14: Retour()

Réception de $\langle Ack \rangle$ de q

15: $Cpt \leftarrow Cpt + 1$

16: Retour()

Fonction Retour()

17: **Si** $Cpt \geq \delta_p$ **alors**

18: **Si** $Père = \top$ **alors**

19: décide

20: **Sinon**

21: Envoyer $\langle Ack \rangle$ à $Père$

22: **Fin Si**

23: **Fin Si**

Question 6. Justifiez pourquoi au plus une décision est prise.

D'après le code de l'algorithme, seul l'initiateur peut décider.

De plus, d'après la réponse à la question précédente, il décide au plus une fois car il reçoit **au plus 1 message de chacun de ses voisins**.

Variables

1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$

2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

3: $Père \leftarrow \top$

4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**

5: Envoyer $\langle Brd, d \rangle$ à q

6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

7: $Cpt \leftarrow Cpt + 1$

8: **Si** $Père = \perp$ **alors**

9: $Père \leftarrow q$

10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**

11: Envoyer $\langle Brd, d \rangle$ à v

12: **Fin Pour**

13: **Fin Si**

14: Retour ()

□ Réception de $\langle Ack \rangle$ de q

15: $Cpt \leftarrow Cpt + 1$

16: Retour ()

Fonction Retour ()

17: **Si** $Cpt \geq \delta_p$ **alors**

18: **Si** $Père = \top$ **alors**

19: décide

20: **Sinon**

21: Envoyer $\langle Ack \rangle$ à $Père$

22: **Fin Si**

23: **Fin Si**

Question 7. Pourquoi la décision prise dépend causalement d'un accusé de réception de chaque processus ? (1/2)

Variables

- 1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$
- 2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

- 3: $Père \leftarrow \top$
- 4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**
- 5: Envoyer $\langle Brd, d \rangle$ à q
- 6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

- 7: $Cpt \leftarrow Cpt + 1$
- 8: **Si** $Père = \perp$ **alors**
- 9: $Père \leftarrow q$
- 10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**
- 11: Envoyer $\langle Brd, d \rangle$ à v
- 12: **Fin Pour**
- 13: **Fin Si**
- 14: Retour ()

Réception de $\langle Ack \rangle$ de q

- 15: $Cpt \leftarrow Cpt + 1$
- 16: Retour ()

Fonction Retour ()

- 17: **Si** $Cpt \geq \delta_p$ **alors**
- 18: **Si** $Père = \top$ **alors**
- 19: décide
- 20: **Sinon**
- 21: Envoyer $\langle Ack \rangle$ à $Père$
- 22: **Fin Si**
- 23: **Fin Si**

Question 7. Pourquoi la décision prise dépend causalement d'un accusé de réception de chaque processus ? (1/2)

L'initiateur a au moins un fils dans T (par hypothèse, il y a au moins 2 nœuds).

D'après les questions précédentes, l'initiateur décide après avoir reçu *Ack* de chacun de ses fils dans T .

Donc la décision de l'initiateur dépend causalement des accusés de réception de ses fils dans T .

Variables

- 1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$
- 2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

- 3: $Père \leftarrow \top$
- 4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**
- 5: Envoyer $\langle Brd, d \rangle$ à q
- 6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

- 7: $Cpt \leftarrow Cpt + 1$
- 8: **Si** $Père = \perp$ **alors**
- 9: $Père \leftarrow q$
- 10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**
- 11: Envoyer $\langle Brd, d \rangle$ à v
- 12: **Fin Pour**
- 13: **Fin Si**
- 14: Retour ()

Réception de $\langle Ack \rangle$ de q

- 15: $Cpt \leftarrow Cpt + 1$
- 16: Retour ()

Fonction Retour ()

- 17: **Si** $Cpt \geq \delta_p$ **alors**
- 18: **Si** $Père = \top$ **alors**
- 19: décide
- 20: **Sinon**
- 21: Envoyer $\langle Ack \rangle$ à $Père$
- 22: **Fin Si**
- 23: **Fin Si**

Question 7. Pourquoi la décision prise dépend causalement d'un accusé de réception de chaque processus ? (1/2)

L'initiateur a au moins un fils dans T (par hypothèse, il y a au moins 2 nœuds).

D'après les questions précédentes, l'initiateur décide après avoir reçu *Ack* de chacun de ses fils dans T .

Donc la décision de l'initiateur dépend causalement des accusés de réception de ses fils dans T .

Par récurrence sur la hauteur de p dans T , $h(p)$, nous pouvons prouver que le message *Ack* envoyé par tout suiveur à son père dépend causalement d'un accusé de réception de chacun des autres processus de son sous-arbre. Et ainsi nous pouvons conclure.

Variables

1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, T\}$ initialisé à $\perp$

2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

3: $Père \leftarrow T$

4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**

5: Envoyer $\langle Brd, d \rangle$ à q

6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

7: $Cpt \leftarrow Cpt + 1$

8: **Si** $Père = \perp$ **alors**

9: $Père \leftarrow q$

10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**

11: Envoyer $\langle Brd, d \rangle$ à v

12: **Fin Pour**

13: **Fin Si**

14: **Retour** ()

Réception de $\langle Ack \rangle$ de q

15: $Cpt \leftarrow Cpt + 1$

16: **Retour** ()

Fonction **Retour** ()

17: **Si** $Cpt \geq \delta_p$ **alors**

18: **Si** $Père = T$ **alors**

19: décide

20: **Sinon**

21: Envoyer $\langle Ack \rangle$ à $Père$

22: **Fin Si**

23: **Fin Si**

Question 7. Pourquoi la décision prise dépend causalement d'un accusé de réception de chaque processus ? (2/2)

Prouvons, par récurrence sur la hauteur de p , $h(p)$, que le message *Ack* envoyé par tout suiveur à son père dépend causalement d'un accusé de réception de chacun des autres processus de son sous-arbre.

Variables

1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$

2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

3: $Père \leftarrow \top$

4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**

5: Envoyer $\langle Brd, d \rangle$ à q

6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

7: $Cpt \leftarrow Cpt + 1$

8: **Si** $Père = \perp$ **alors**

9: $Père \leftarrow q$

10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**

11: Envoyer $\langle Brd, d \rangle$ à v

12: **Fin Pour**

13: **Fin Si**

14: Retour()

Réception de $\langle Ack \rangle$ de q

15: $Cpt \leftarrow Cpt + 1$

16: Retour()

Fonction Retour()

17: **Si** $Cpt \geq \delta_p$ **alors**

18: **Si** $Père = \top$ **alors**

19: décide

20: **Sinon**

21: Envoyer $\langle Ack \rangle$ à $Père$

22: **Fin Si**

23: **Fin Si**

Question 7. Pourquoi la décision prise dépend causalement d'un accusé de réception de chaque processus ? (2/2)

Prouvons, par récurrence sur la hauteur de p , $h(p)$, que le message *Ack* envoyé par tout suiveur à son père dépend causalement d'un accusé de réception de chacun des autres processus de son sous-arbre.

- Le cas de base $h(p) = 0$ correspond à une feuille de l'arbre, donc la récurrence est trivialement vérifiée puisqu'une feuille est l'unique nœud de son sous-arbre.

Variables

- 1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$
- 2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

- 3: $Père \leftarrow \top$
- 4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**
- 5: Envoyer $\langle Brd, d \rangle$ à q
- 6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

- 7: $Cpt \leftarrow Cpt + 1$
- 8: **Si** $Père = \perp$ **alors**
- 9: $Père \leftarrow q$
- 10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**
- 11: Envoyer $\langle Brd, d \rangle$ à v
- 12: **Fin Pour**
- 13: **Fin Si**
- 14: Retour()

Réception de $\langle Ack \rangle$ de q

- 15: $Cpt \leftarrow Cpt + 1$
- 16: Retour()

Fonction Retour()

- 17: **Si** $Cpt \geq \delta_p$ **alors**
- 18: **Si** $Père = \top$ **alors**
- 19: décide
- 20: **Sinon**
- 21: Envoyer $\langle Ack \rangle$ à $Père$
- 22: **Fin Si**
- 23: **Fin Si**

Question 7. Pourquoi la décision prise dépend causalement d'un accusé de réception de chaque processus ? (2/2)

Prouvons, par récurrence sur la hauteur de p , $h(p)$, que le message *Ack* envoyé par tout suiveur à son père dépend causalement d'un accusé de réception de chacun des autres processus de son sous-arbre.

- Le cas de base $h(p) = 0$ correspond à une feuille de l'arbre, donc la récurrence est trivialement vérifiée puisqu'une feuille est l'unique nœud de son sous-arbre.

- Soit p un processus suiveur tel que $h(p) > 0$.

p reçoit au plus un message de chacun de ses fils (Question 5). D'après l'algorithme, tout message reçu d'un de ses fils est un *Ack*.

Donc, d'après l'algorithme p envoie un *Ack* à son père après avoir reçu un *Ack* de chacun de ses fils.

Par **hypothèse de récurrence**, le *Ack* de chaque fils dépend causalement d'un accusé de réception de chacun des autres processus de son sous-arbre.

Ainsi, le *Ack* de p dépend causalement d'un accusé de réception de chacun des autres processus de son sous-arbre.

Variables

- 1: $Père \in \{1, \dots, \delta_p\} \cup \{\perp, \top\}$ initialisé à $\perp$
- 2: $Cpt \in \mathbb{N}$ initialisé à 0

Spontanément

- 3: $Père \leftarrow \top$
- 4: **Pour tout** $q \in \{1, \dots, \delta_p\}$ **faire**
- 5: Envoyer $\langle Brd, d \rangle$ à q
- 6: **Fin Pour**

Réception de $\langle Brd, d \rangle$ de q

- 7: $Cpt \leftarrow Cpt + 1$
- 8: **Si** $Père = \perp$ **alors**
- 9: $Père \leftarrow q$
- 10: **Pour tout** $v \in \{1, \dots, \delta_p\} \setminus \{q\}$ **faire**
- 11: Envoyer $\langle Brd, d \rangle$ à v
- 12: **Fin Pour**
- 13: **Fin Si**
- 14: Retour ()

Réception de $\langle Ack \rangle$ de q

- 15: $Cpt \leftarrow Cpt + 1$
- 16: Retour ()

Fonction Retour ()

- 17: **Si** $Cpt \geq \delta_p$ **alors**
- 18: **Si** $Père = \top$ **alors**
- 19: décide
- 20: **Sinon**
- 21: Envoyer $\langle Ack \rangle$ à $Père$
- 22: **Fin Si**
- 23: **Fin Si**

□

Conclusion

D'après l'ensemble des réponses aux questions, vous pouvez conclure :

Théorème

L'algorithme résout la propagation d'information avec retour dans un réseau quelconque.

Conclusion

D'après l'ensemble des réponses aux questions, vous pouvez conclure :

Théorème

L'algorithme résout la propagation d'information avec retour dans un réseau quelconque.

Remarque

Pour pouvoir répéter les vagues, il suffit de réinitialiser les variables après l'envoi de l'accusé de réception et après la décision.

Conclusion

D'après l'ensemble des réponses aux questions, vous pouvez conclure :

Théorème

L'algorithme résout la propagation d'information avec retour dans un réseau quelconque.

Remarque

Pour pouvoir répéter les vagues, il suffit de réinitialiser les variables après l'envoi de l'accusé de réception et après la décision.

Remarque

Pour faire du multi-initiateur, il faut dupliquer l'algorithme en n algorithmes (n étant le nombre de processus) et taguer les messages et variables avec les identités.

Question 8. Donnez la complexité en nombre de messages de l'algorithme.

Question 8. Donnez la complexité en nombre de messages de l'algorithme.

D'après les questions précédentes, chaque lien est traversé exactement une fois dans chaque sens, d'où, $2m$ messages, où m est le nombre d'arêtes.

Question 8. Donnez la complexité en nombre de messages de l'algorithme.

D'après les questions précédentes, chaque lien est traversé exactement une fois dans chaque sens, d'où, $2m$ messages, où m est le nombre d'arêtes.

Question 9. Donnez la complexité en temps de l'algorithme.

Question 8. Donnez la complexité en nombre de messages de l'algorithme.

D'après les questions précédentes, chaque lien est traversé exactement une fois dans chaque sens, d'où, $2m$ messages, où m est le nombre d'arêtes.

Question 9. Donnez la complexité en temps de l'algorithme.

- La phase de diffusion s'exécute en au plus $\mathcal{D}$ unités de temps, où $\mathcal{D}$ est le diamètre du réseau.

Question 8. Donnez la complexité en nombre de messages de l'algorithme.

D'après les questions précédentes, chaque lien est traversé exactement une fois dans chaque sens, d'où, $2m$ messages, où m est le nombre d'arêtes.

Question 9. Donnez la complexité en temps de l'algorithme.

- La phase de diffusion s'exécute en au plus $\mathcal{D}$ unités de temps, où $\mathcal{D}$ est le diamètre du réseau.
- La phase de retour consiste en une propagation de bas en haut de l'arbre.

La hauteur de l'arbre étant au maximum $n - 1$, cela se fait en au plus $n - 1$ unités de temps.

Question 8. Donnez la complexité en nombre de messages de l'algorithme.

D'après les questions précédentes, chaque lien est traversé exactement une fois dans chaque sens, d'où, $2m$ messages, où m est le nombre d'arêtes.

Question 9. Donnez la complexité en temps de l'algorithme.

- La phase de diffusion s'exécute en au plus $\mathcal{D}$ unités de temps, où $\mathcal{D}$ est le diamètre du réseau.
- La phase de retour consiste en une propagation de bas en haut de l'arbre.

La hauteur de l'arbre étant au maximum $n - 1$, cela se fait en au plus $n - 1$ unités de temps.

D'où, une complexité en temps de $\mathcal{D} + n - 1 \leq 2n - 2 \leq 2m$.