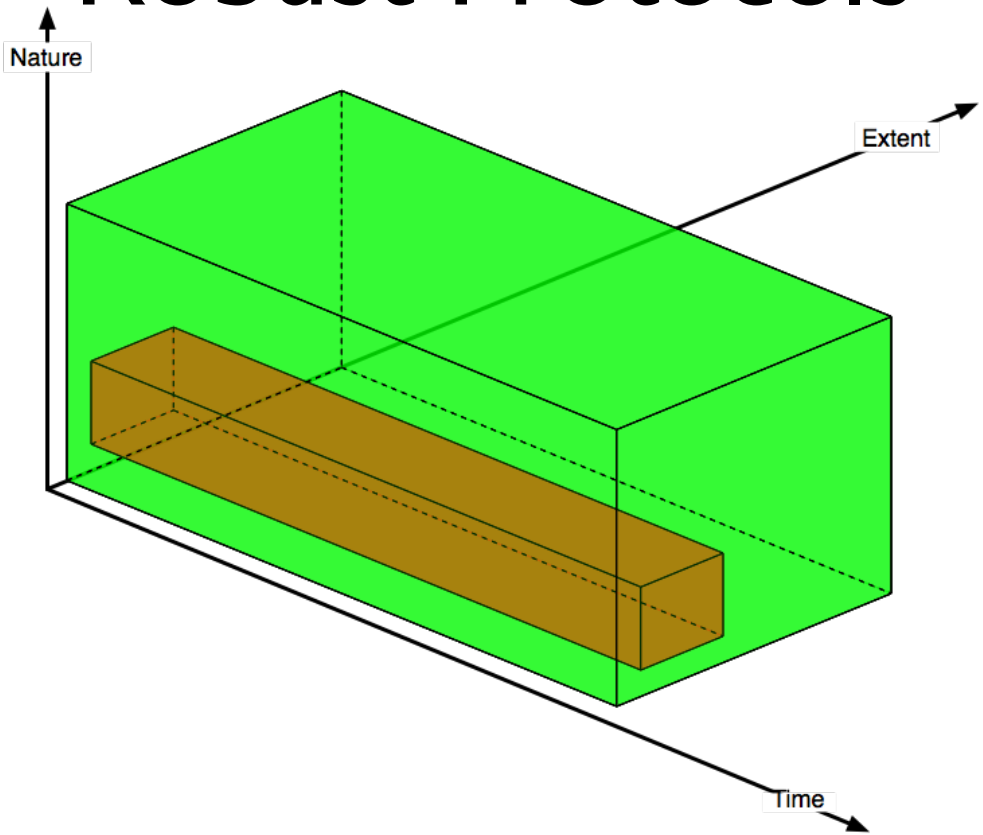
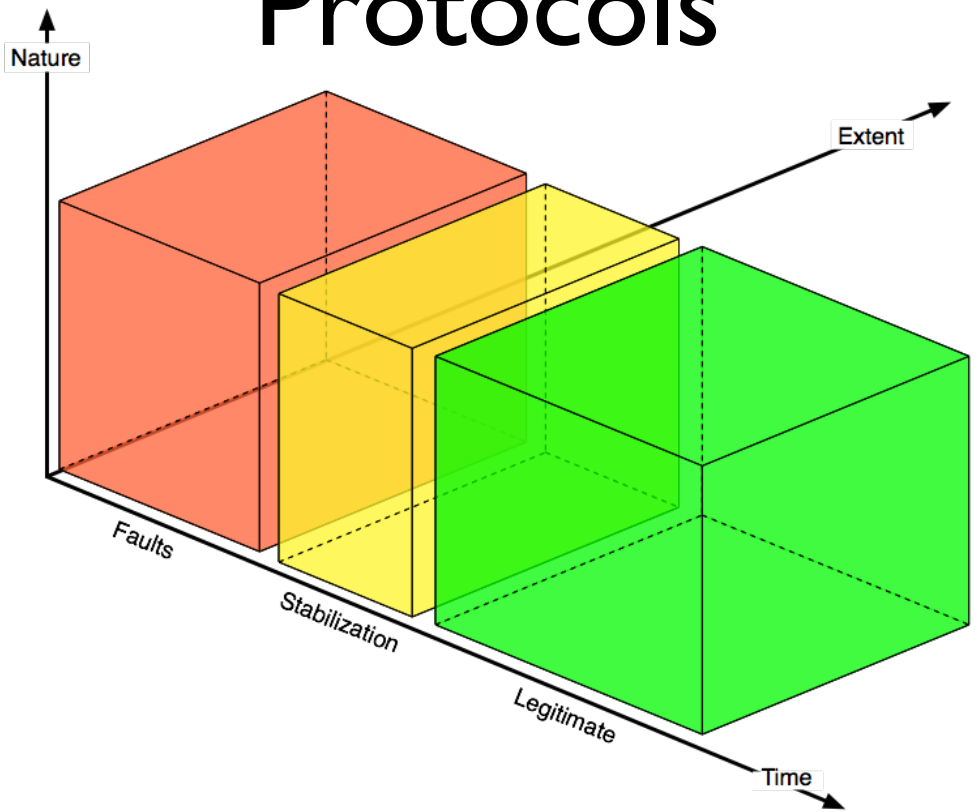
Snap-stabilization

Franck Petit

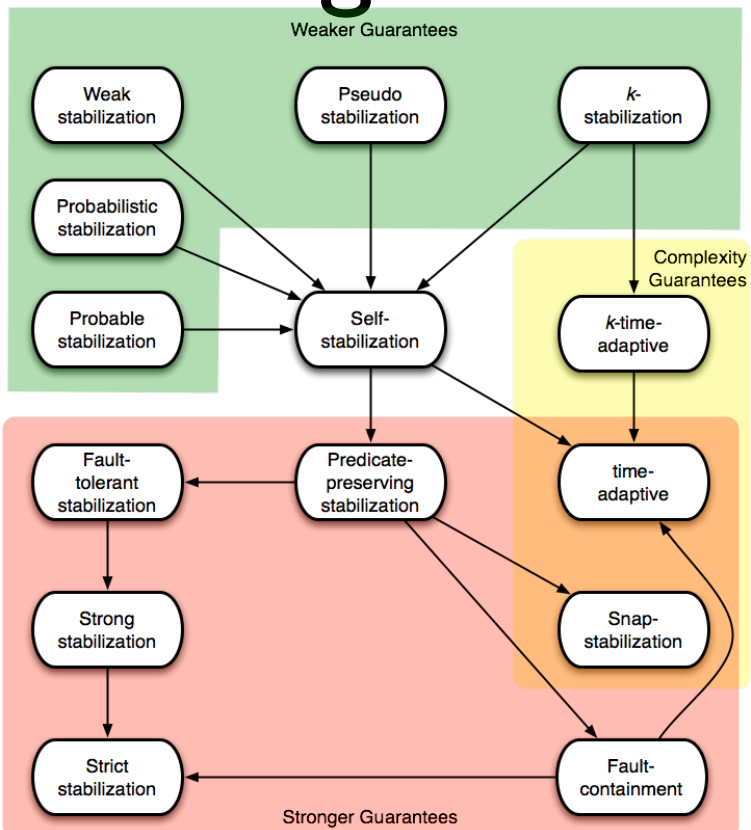
Robust Protocols



Self-stabilizing Protocols



The Big Picture

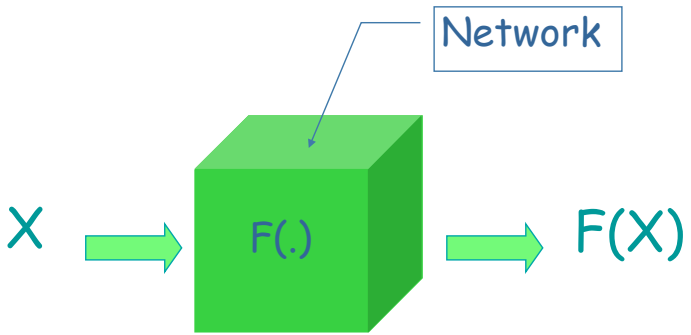


Snap-Stabilization

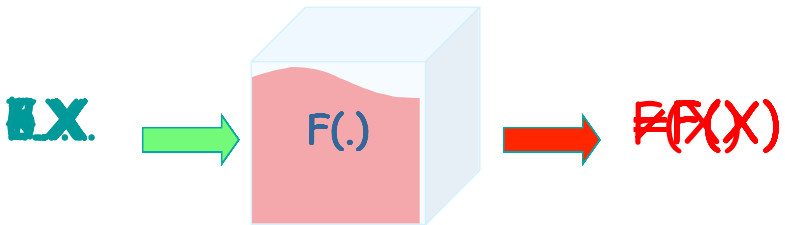
Snap-Stabilization

- **Specification**
- **Self-stabilization**: Starting from any initial configuration, the system eventually reaches a configuration from which its behavior is correct
- **Snap-stabilization**: Starting from any initial configuration, the system always behaves as its specification

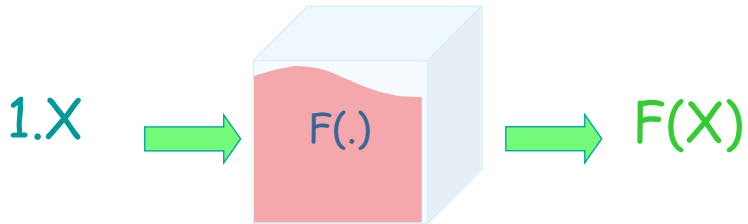
Non Self-*



Self-stabilization



Snap-stabilization



Related Works

- *Bui, Datta, Petit, Villain*. State-optimal snap-stabilizing PIF in tree networks. *WSS* 1999.
- Cournier, Datta, Petit, Villain. Enabling snap-stabilization. SSS 2003.
- *Cournier, Devismes, Villain*. From Self- to Snap- Stabilization. *SSS* 2006.
- *Bui, Datta, Petit, Villain*. Snap-stabilization and PIF in tree networks. *Distributed Computing* 2007.

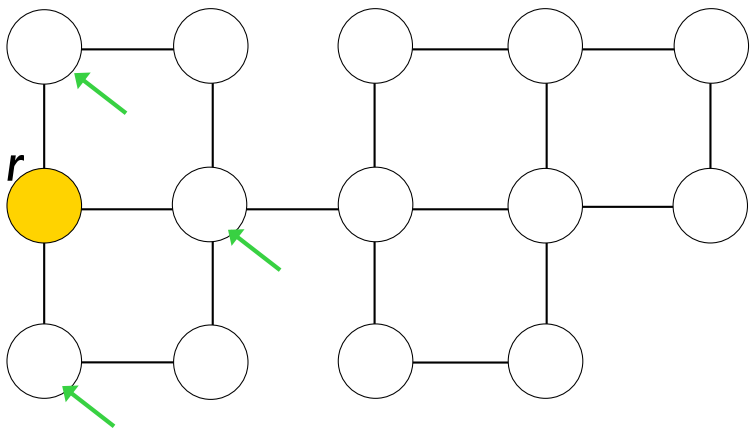
PIF

- Propagation of Information with Feedback

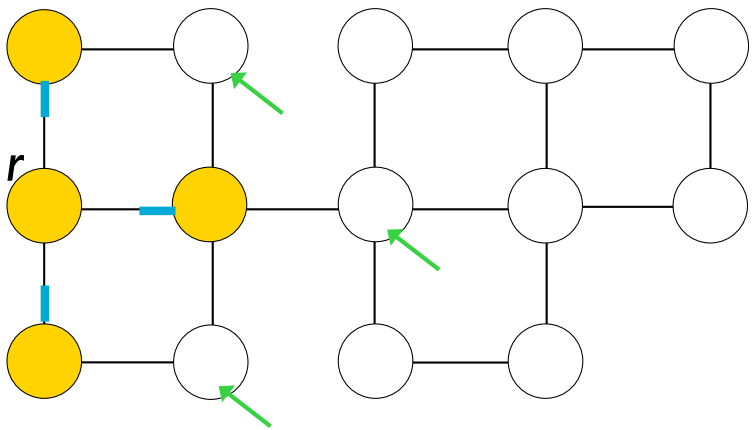
- **Distributed-Control Problems**
- Broadcast, Routing, Synchronization, Protocol, Leader Election, Resource Sharing and Allocation, Graph Algorithms, Termination Detection, Deadlock Detection, Reset, Distributed Ranking, Distributed Sorting...

PIF

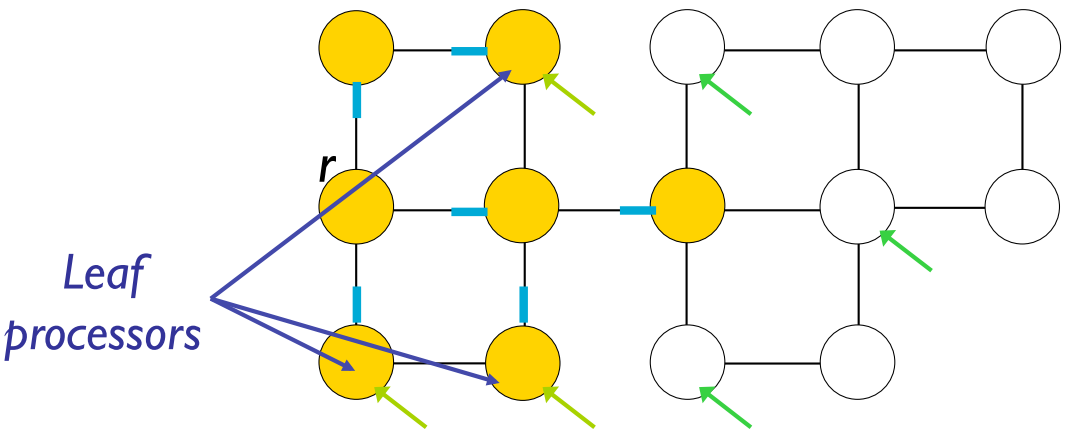
Propagation of Information with Feedback



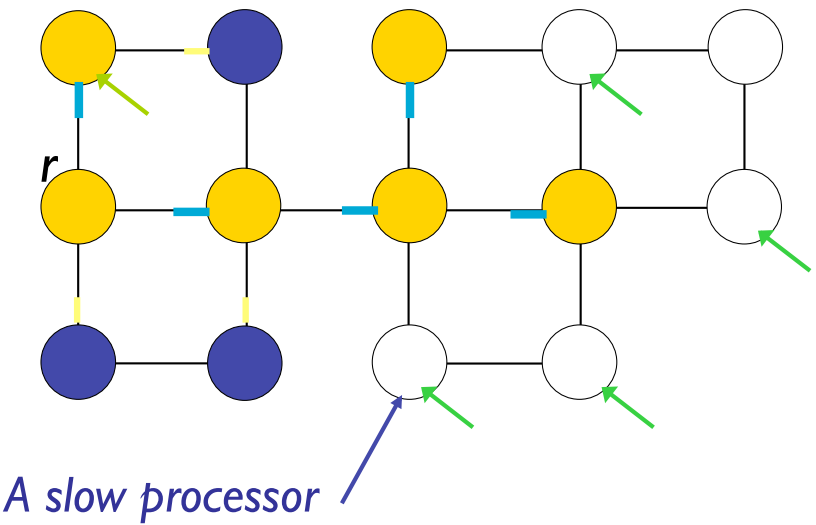
Propagation of Information with Feedback



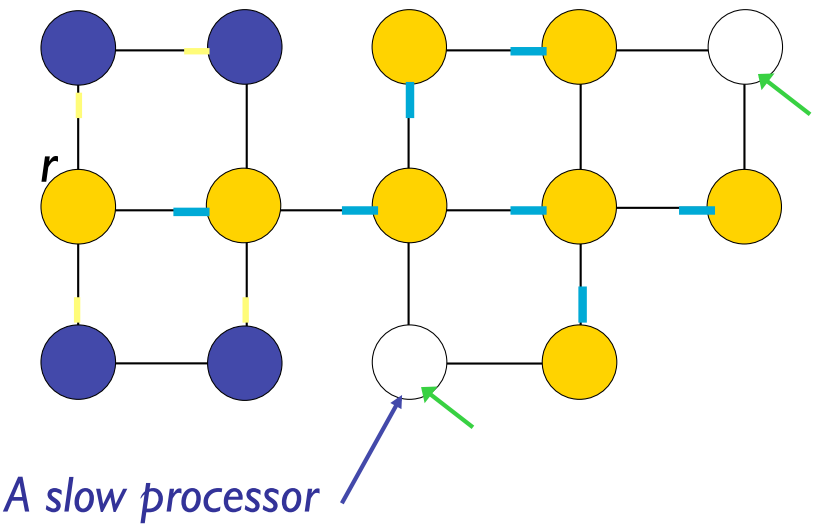
Propagation of Information with Feedback



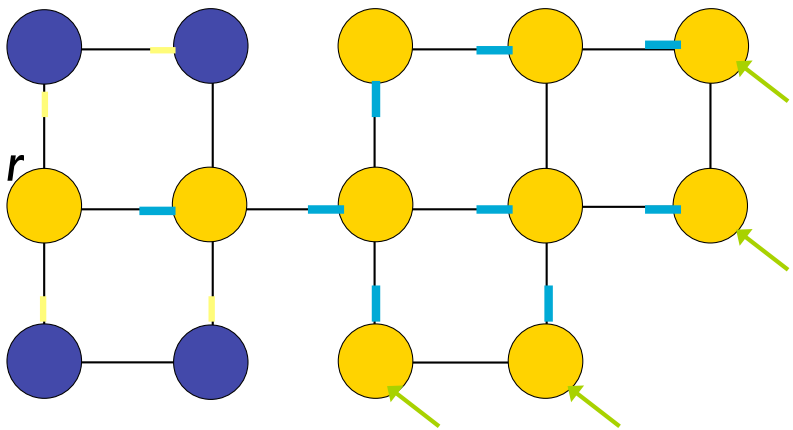
Propagation of Information with Feedback



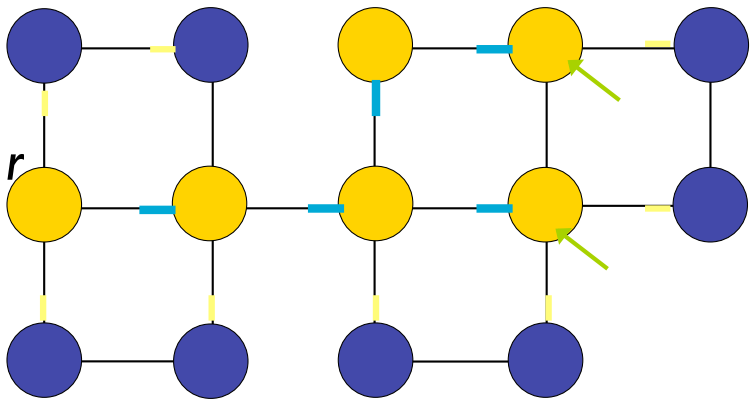
Propagation of Information with Feedback



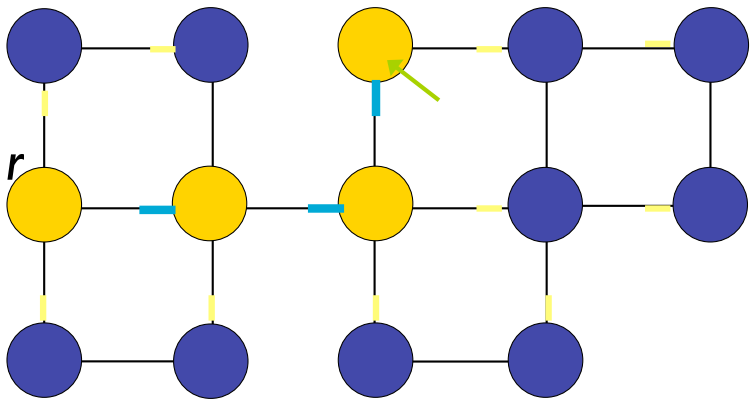
Propagation of Information with Feedback



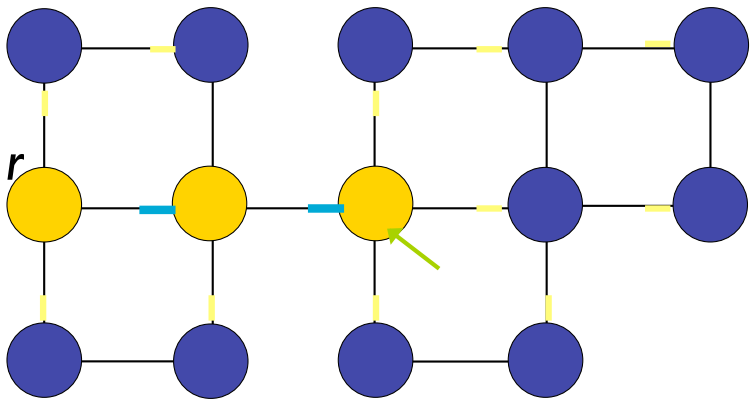
Propagation of Information with Feedback



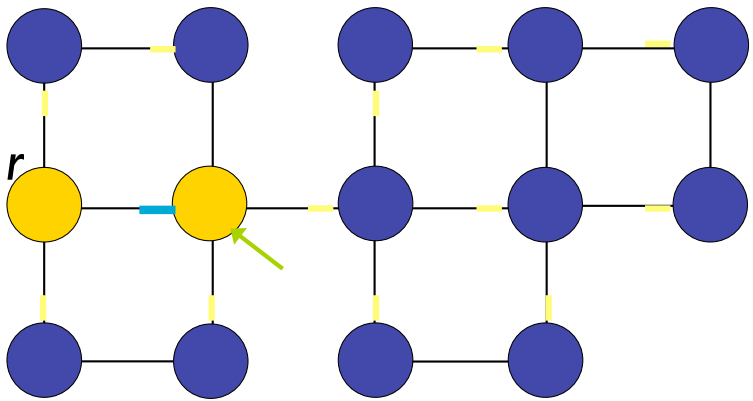
Propagation of Information with Feedback



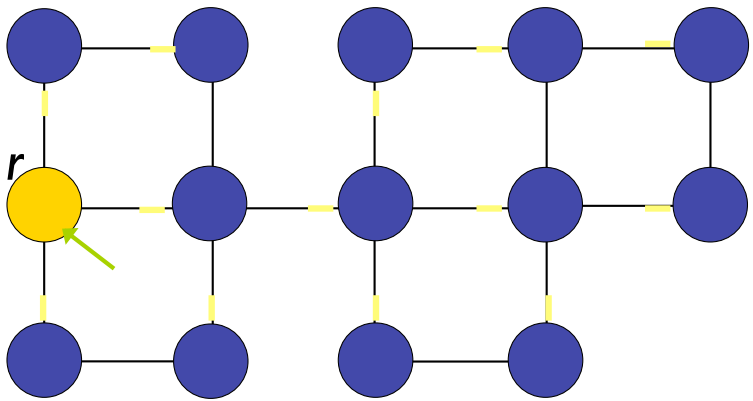
Propagation of Information with Feedback



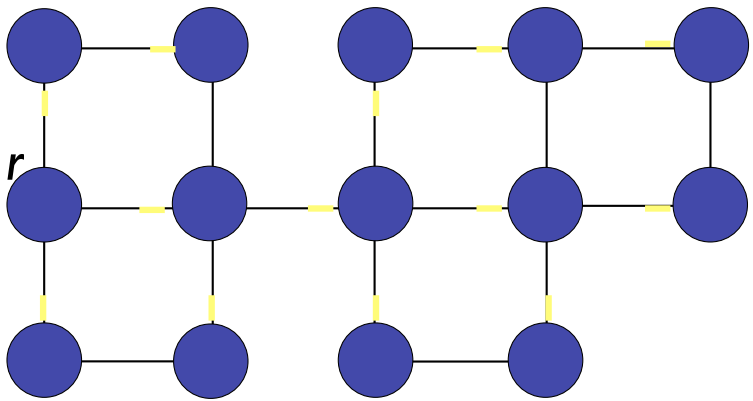
Propagation of Information with Feedback



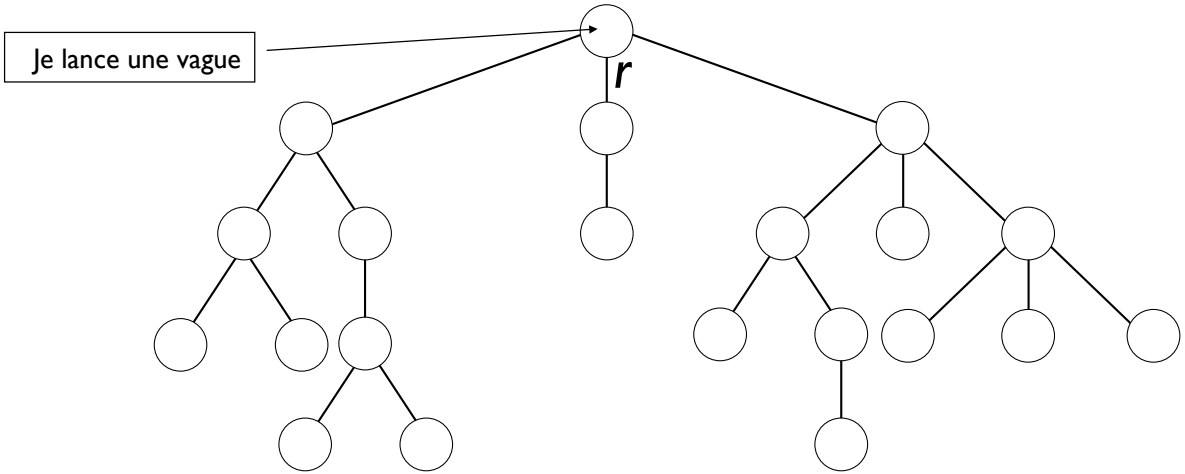
Propagation of Information with Feedback



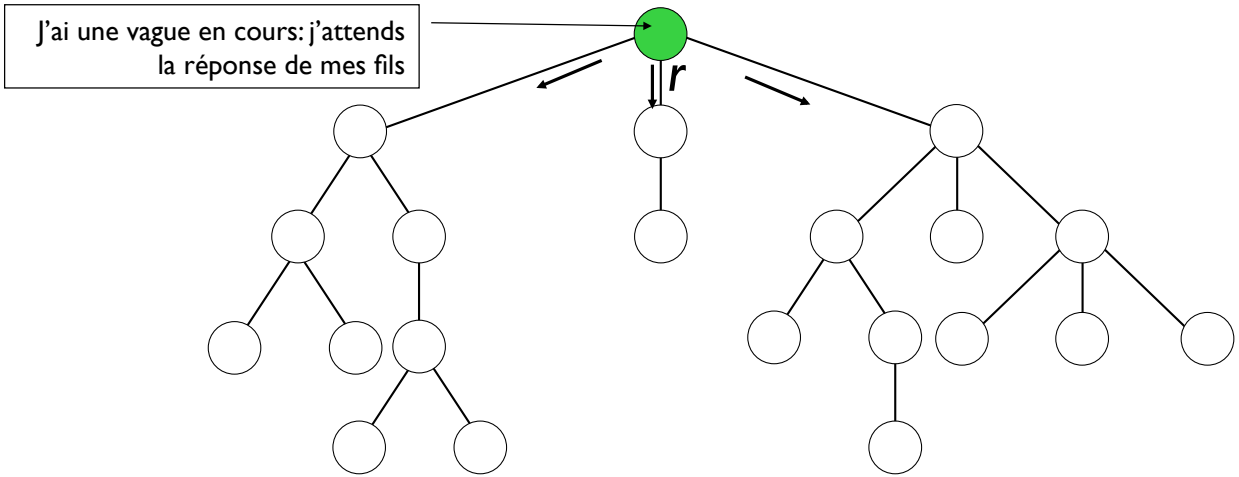
Propagation of Information with Feedback



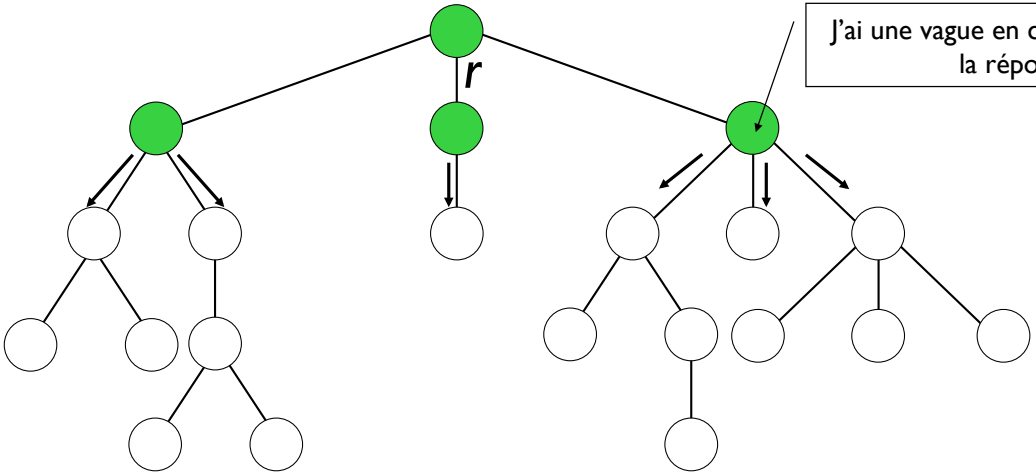
PIF sur l'arbre



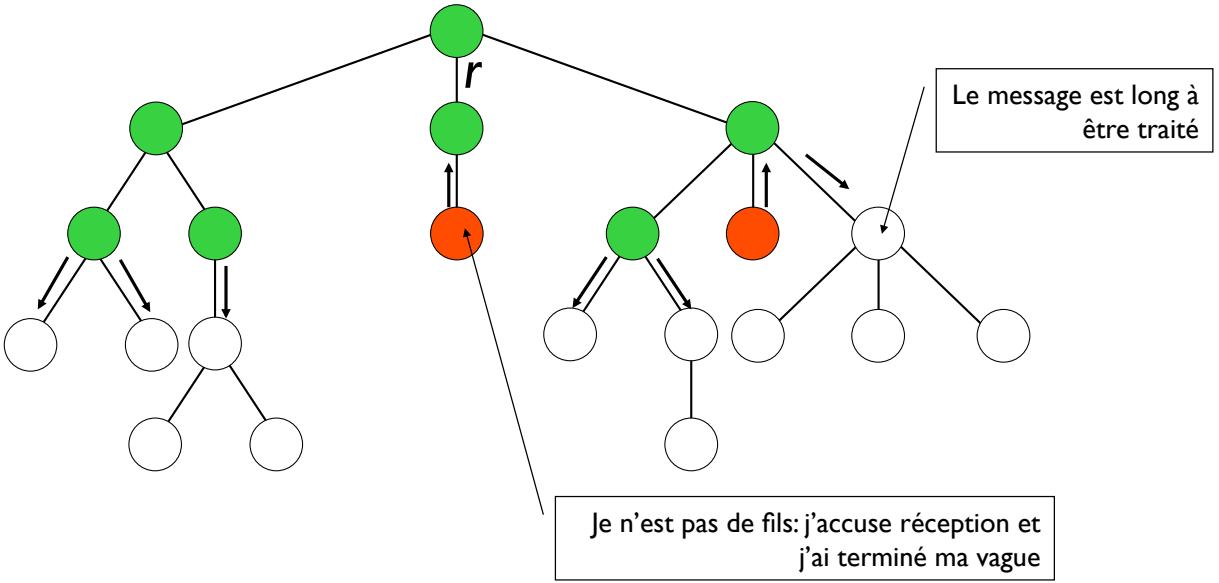
PIF sur l'arbre



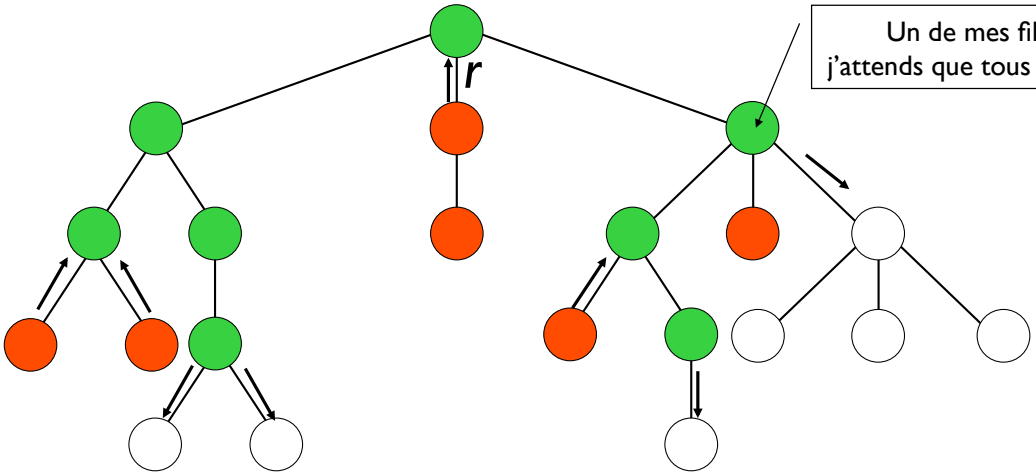
PIF sur l'arbre



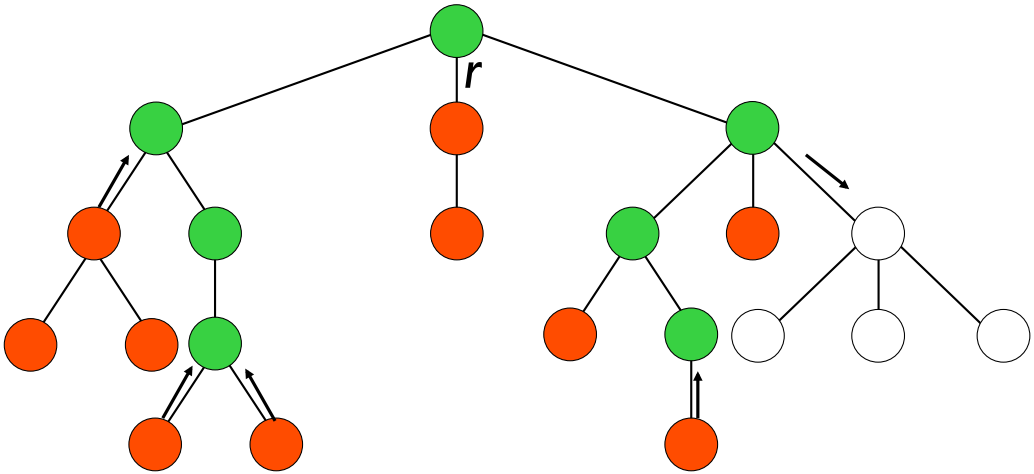
PIF sur l'arbre



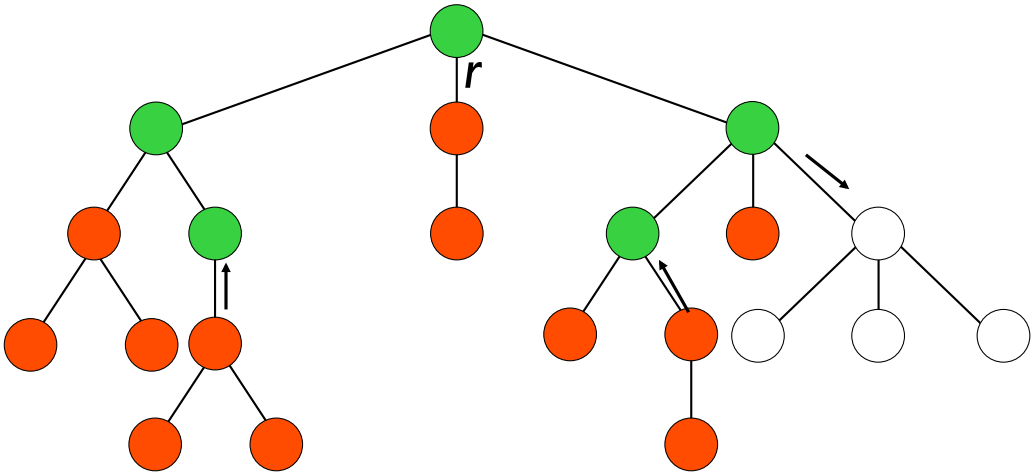
PIF sur l'arbre



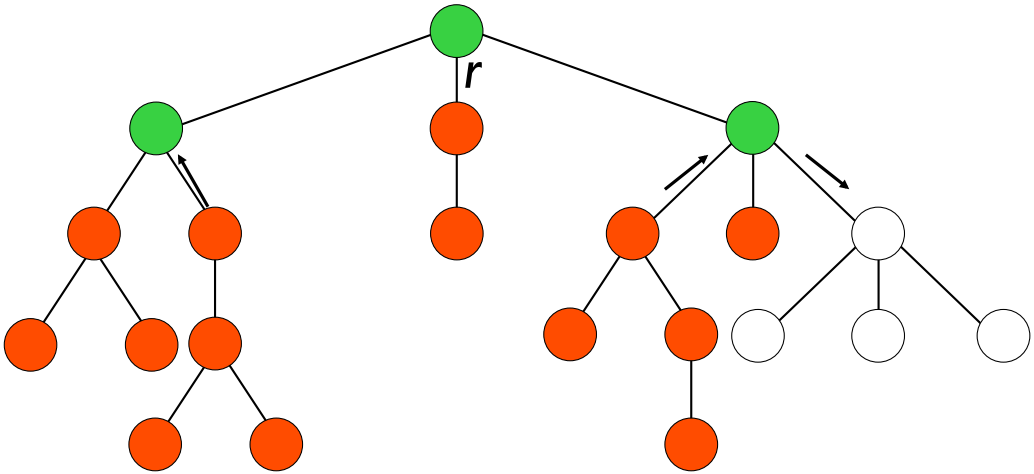
PIF sur l'arbre



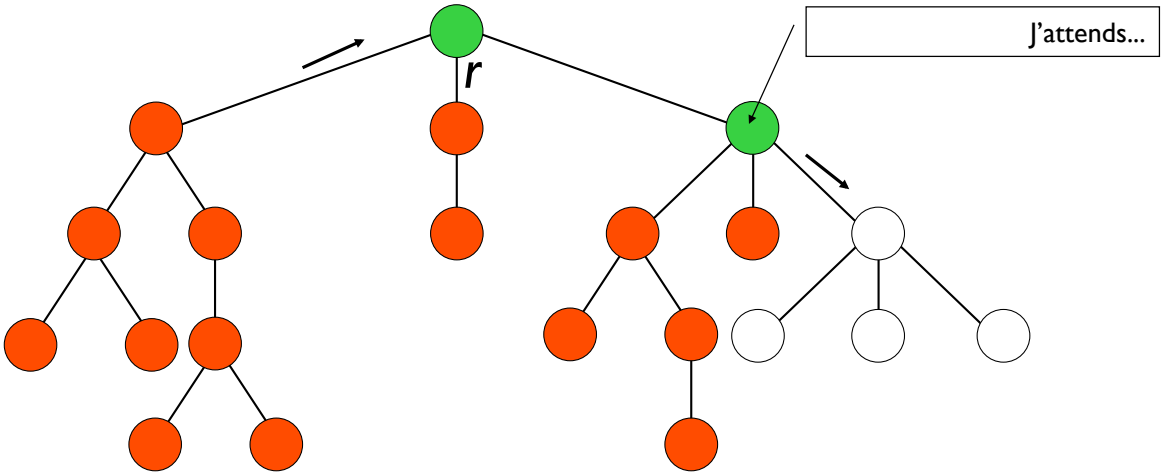
PIF sur l'arbre



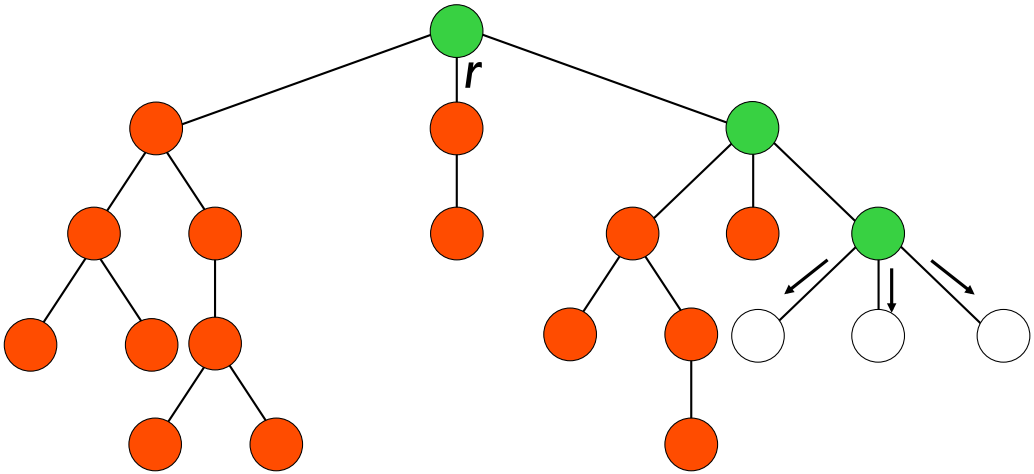
PIF sur l'arbre



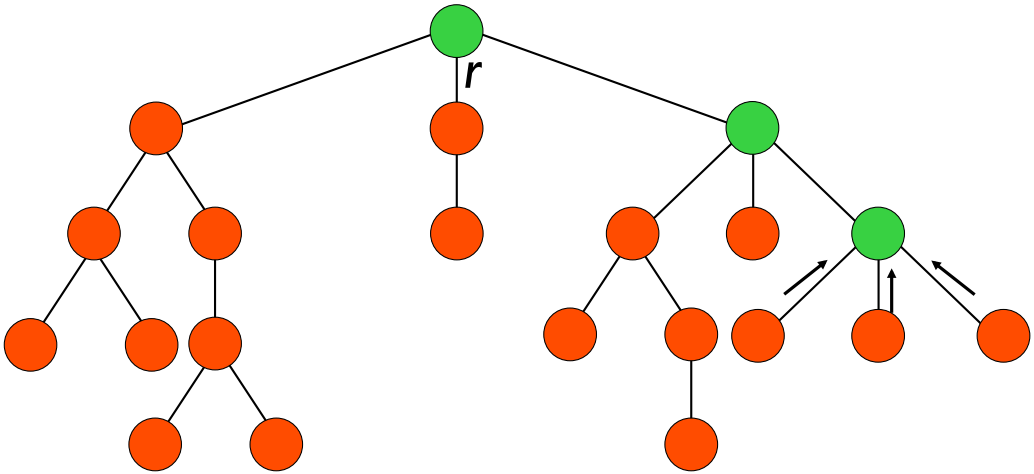
PIF sur l'arbre



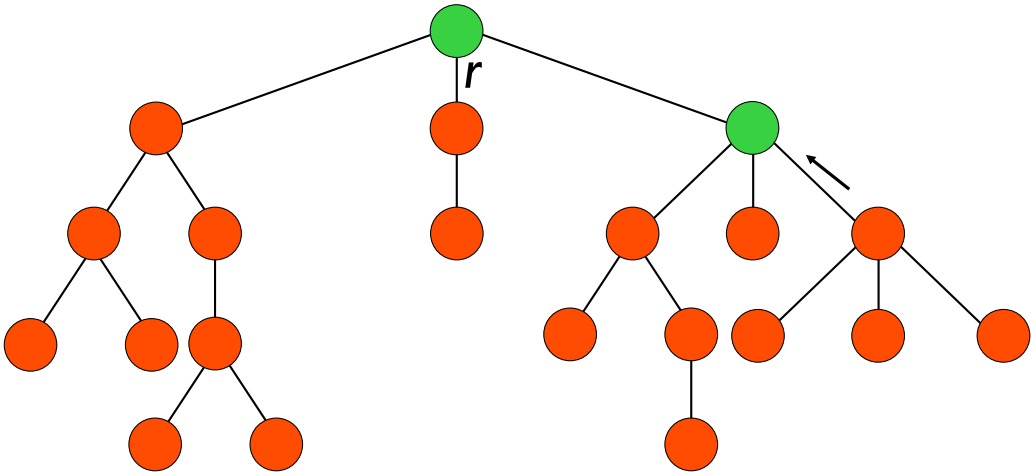
PIF sur l'arbre



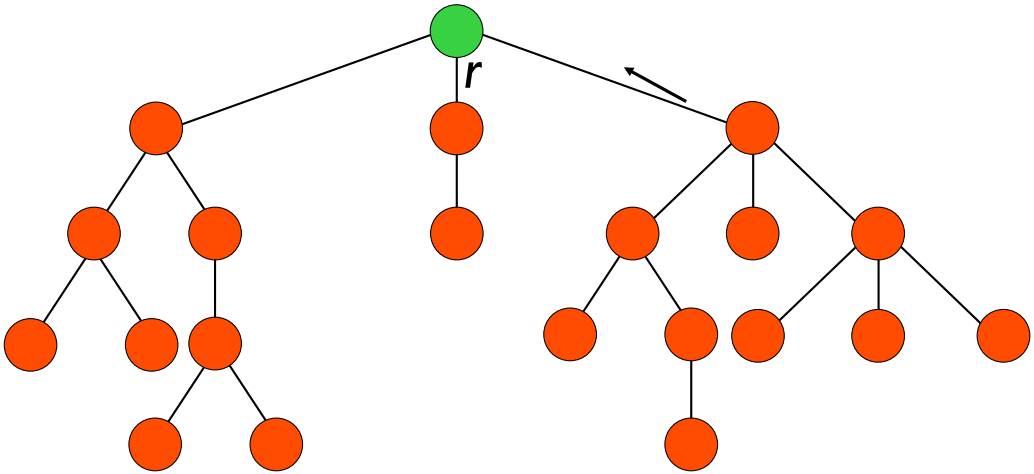
PIF sur l'arbre



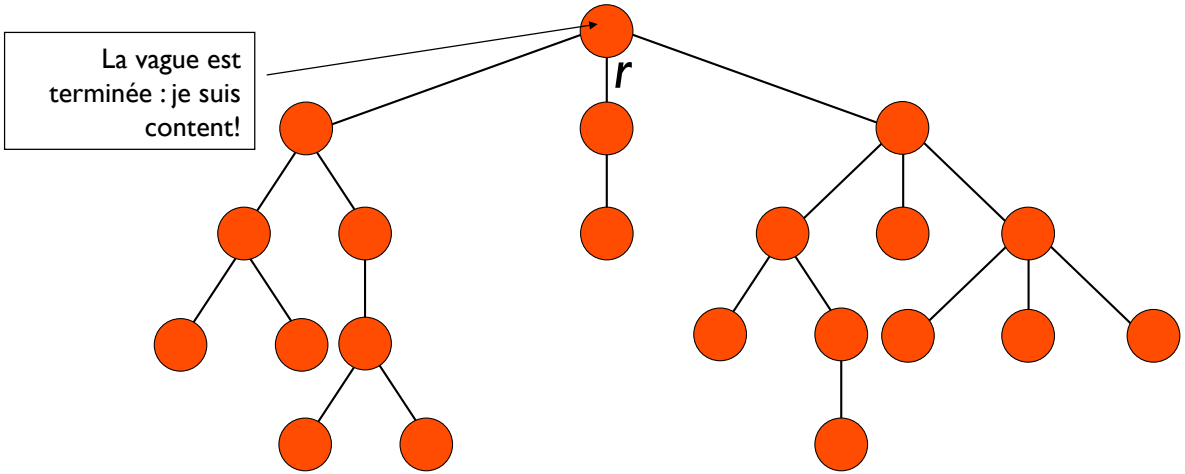
PIF sur l'arbre



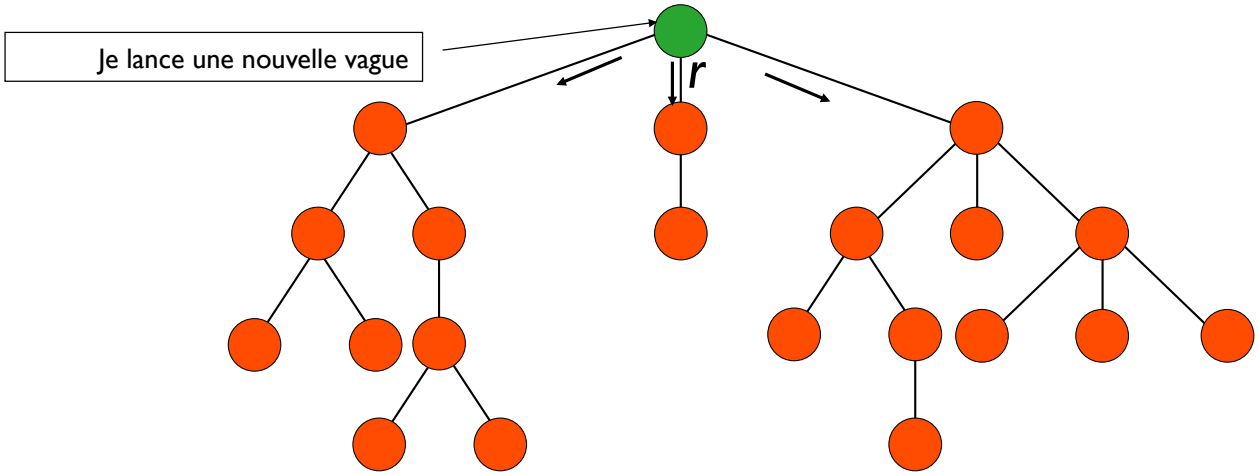
PIF sur l'arbre



PIF sur l'arbre



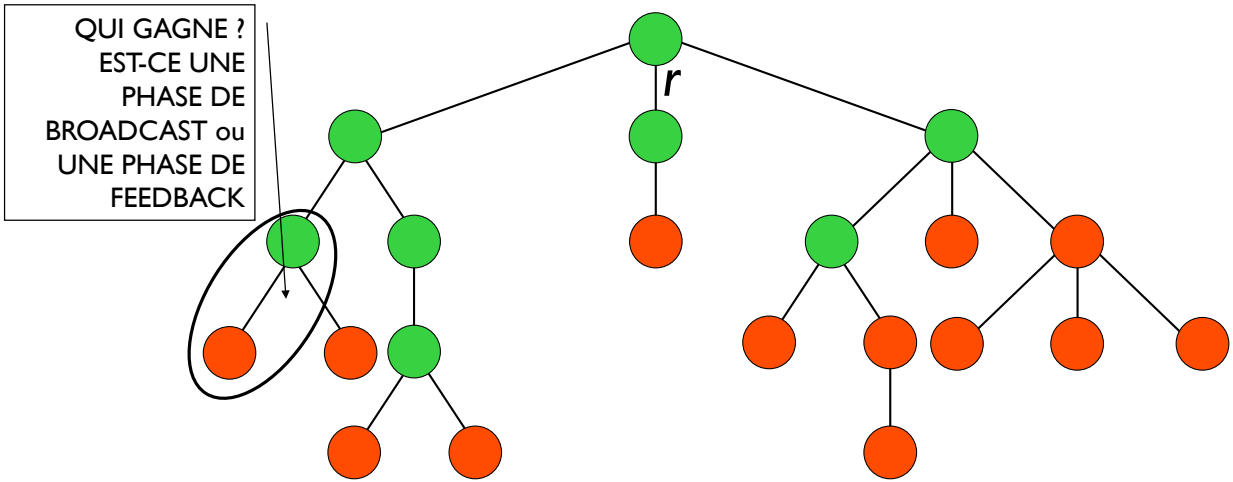
PIF sur l'arbre



Nombre d'états par processeur = 4

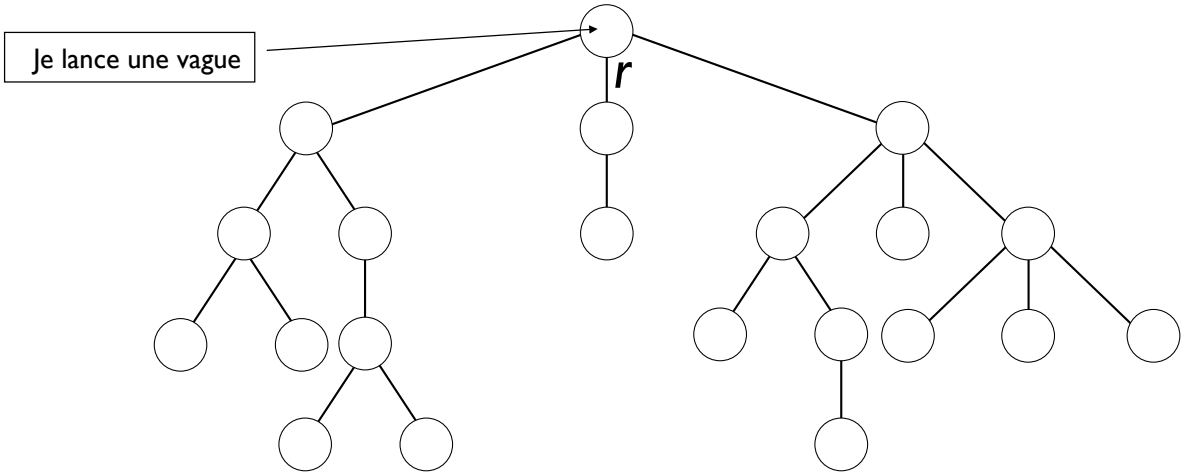
QUESTION : Est-il possible d'implémenter le PIF avec seulement 2 états ?

PIF sur l'arbre

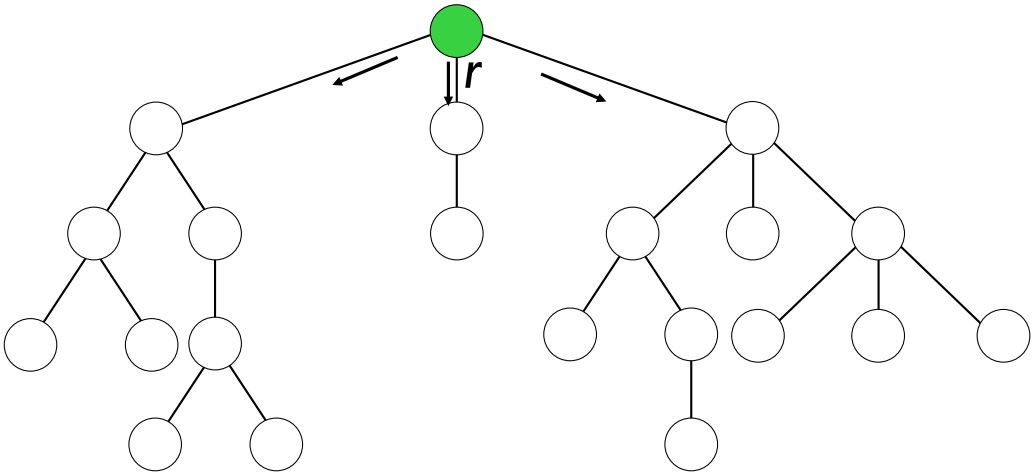


<u>REPONSE</u> : Non. Intuitivement, les états « *J'attend_les_réponses_de_mes_fils* » poussent les états « *J'ai_terminé_ma_vague* » de la vague précédente (ou état initial), et les états « *J'ai_terminé_ma_vague* » de la vague précédente poussent les états « *J'attend_les_réponses_de_mes_fils* ».

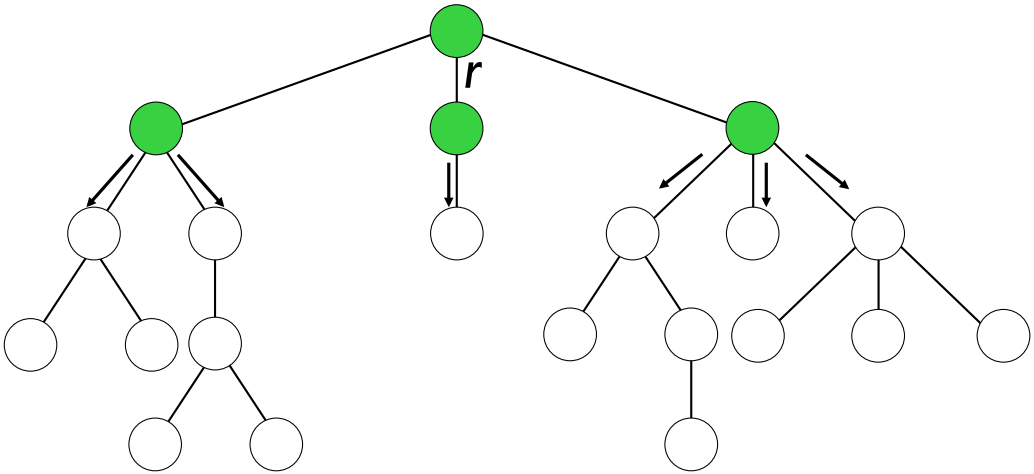
PIF sur l'arbre



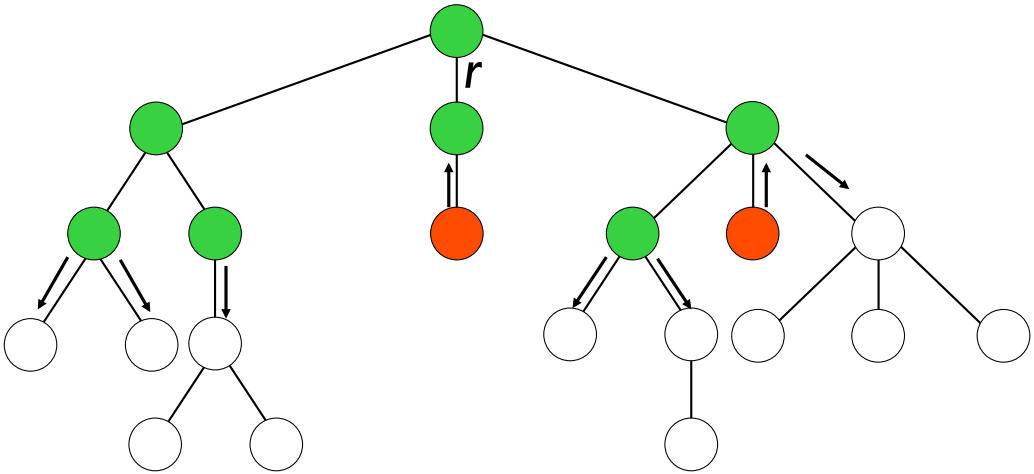
PIF sur l'arbre



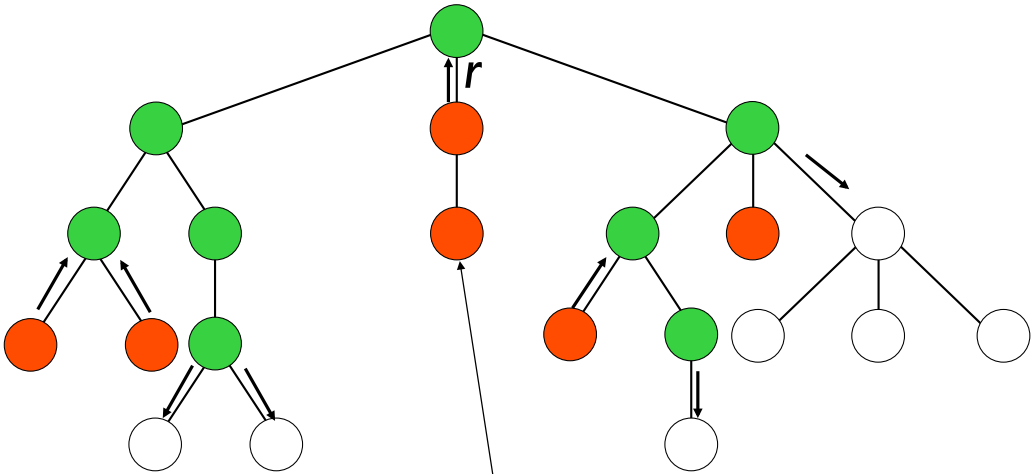
PIF sur l'arbre



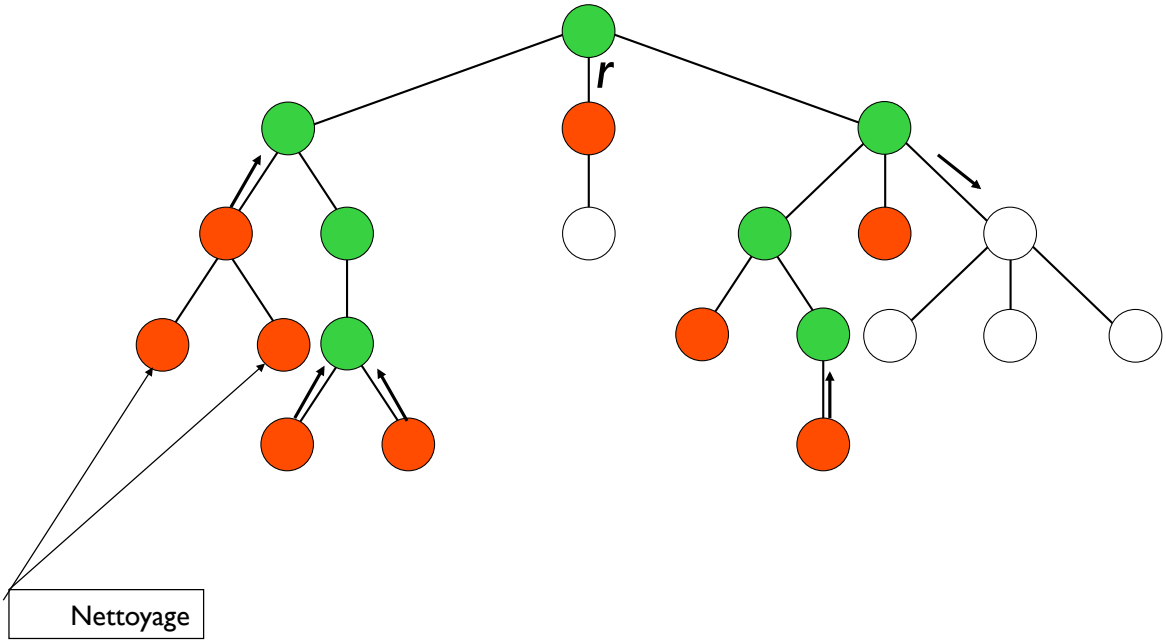
PIF sur l'arbre



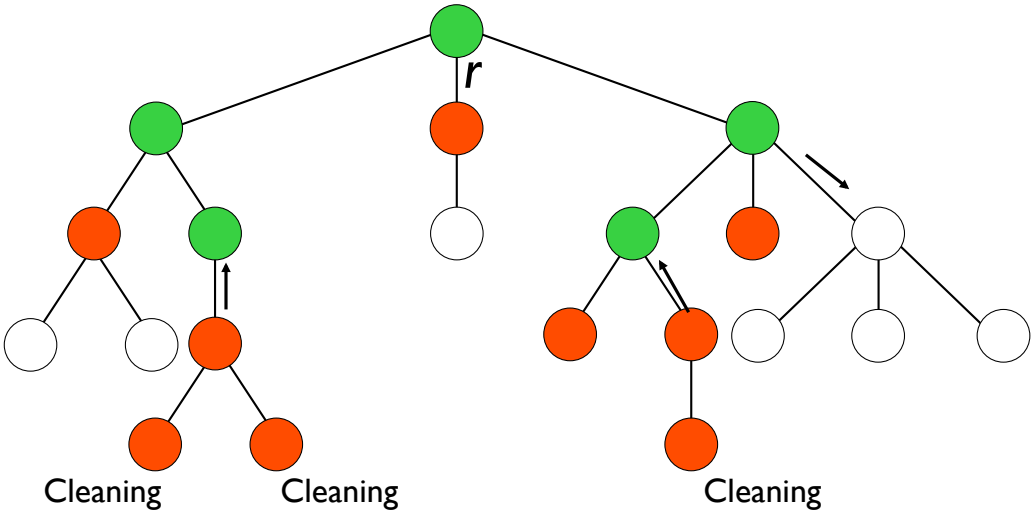
PIF sur l'arbre



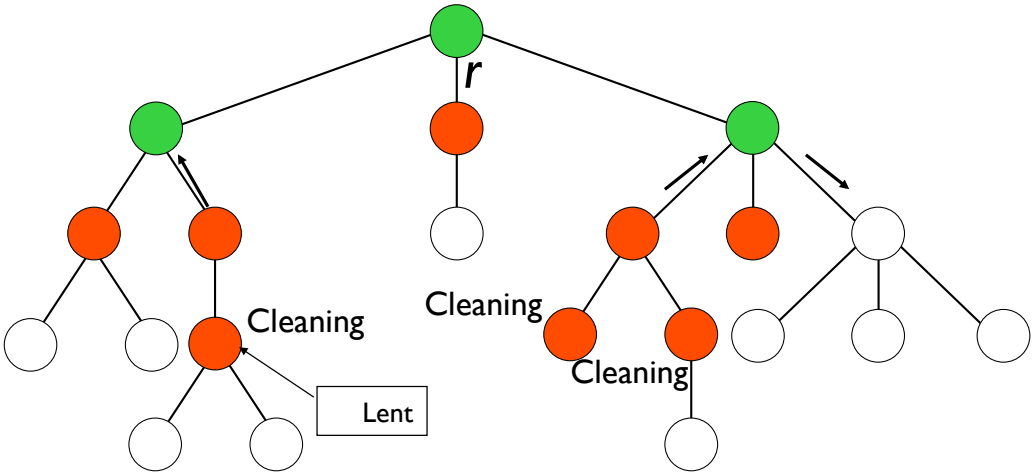
PIF sur l'arbre



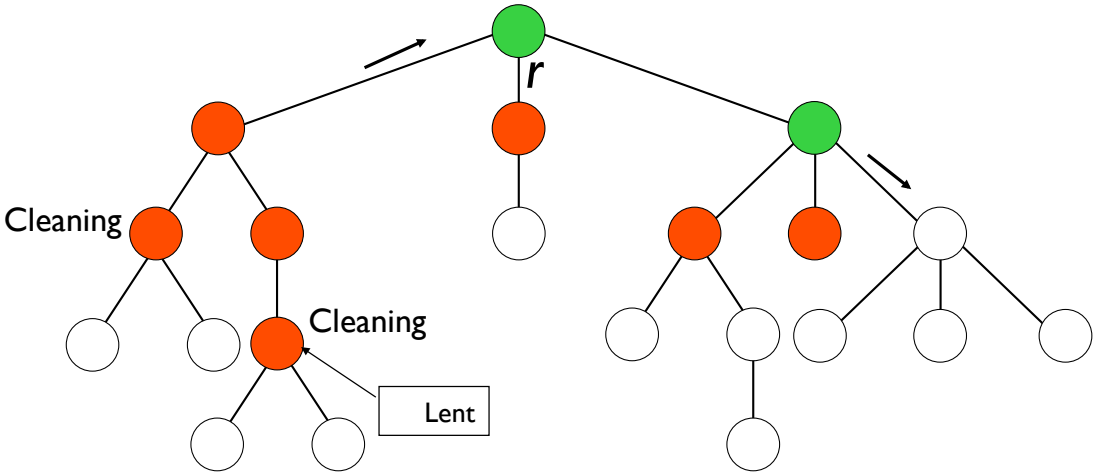
PIF sur l'arbre



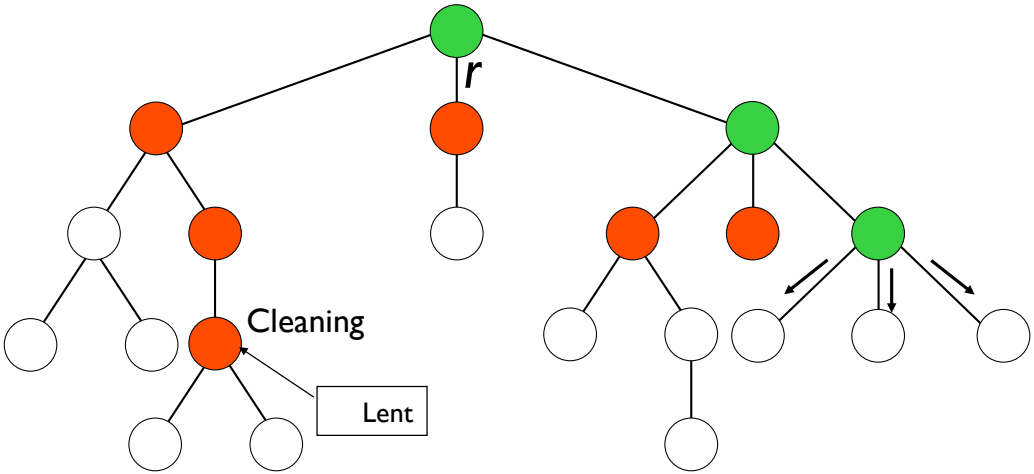
PIF sur l'arbre



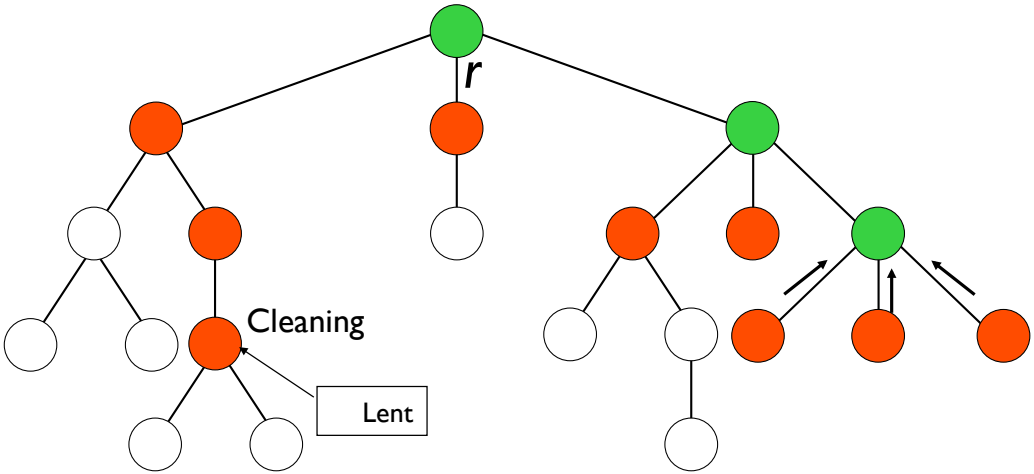
PIF sur l'arbre



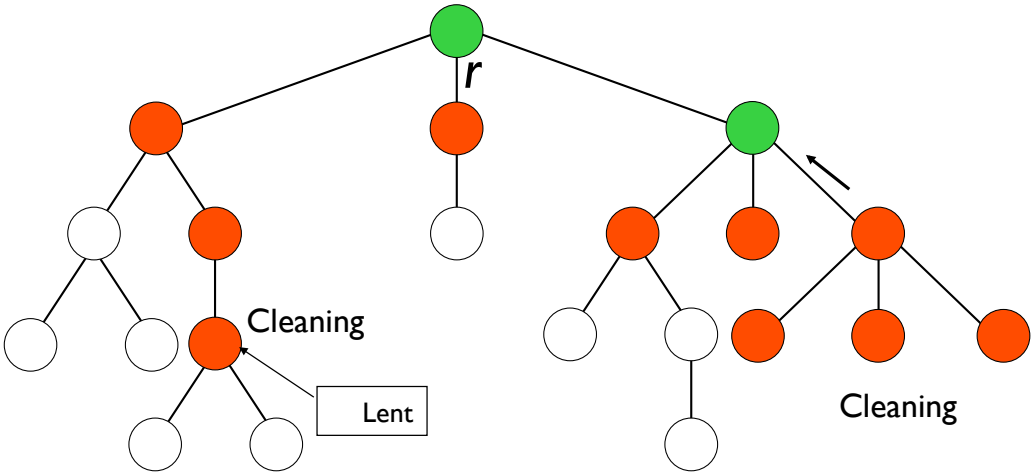
PIF sur l'arbre



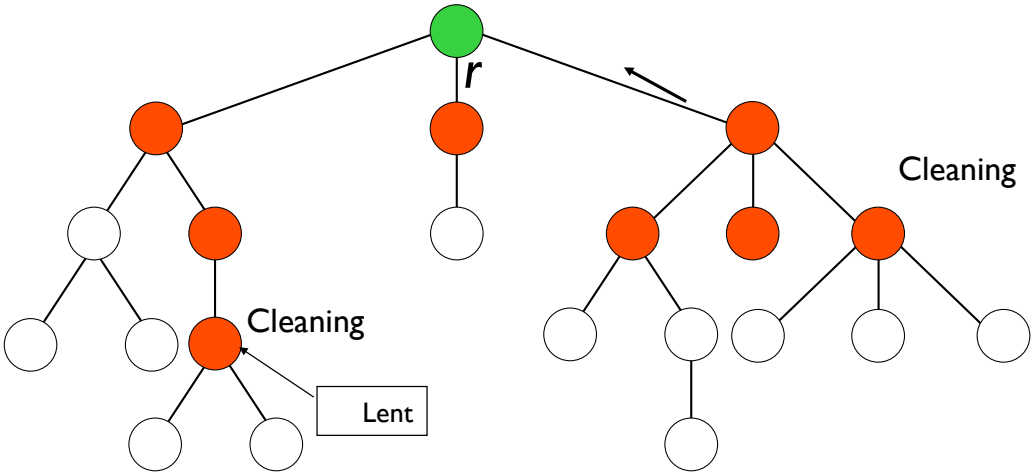
PIF sur l'arbre



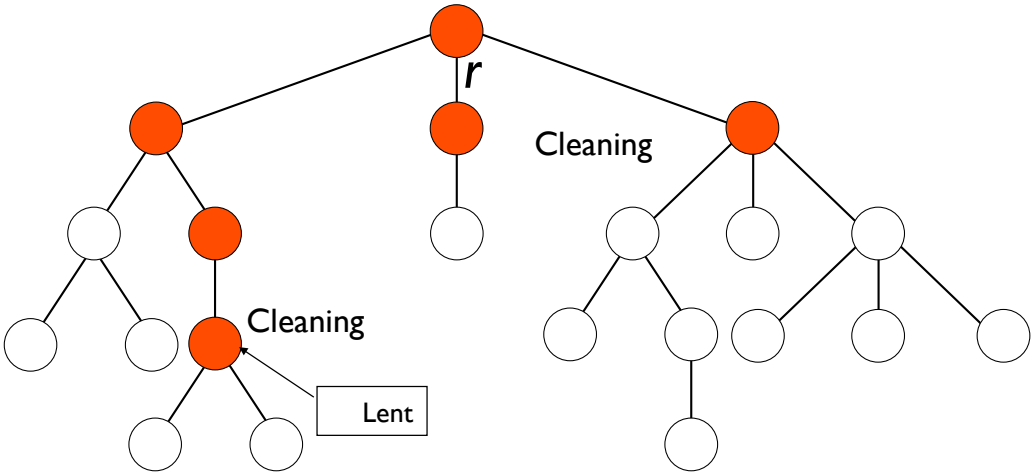
PIF sur l'arbre



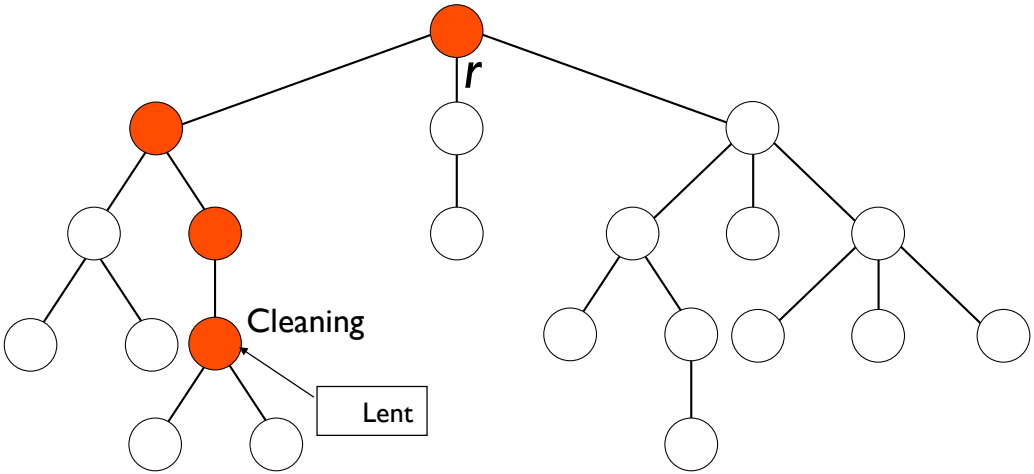
PIF sur l'arbre



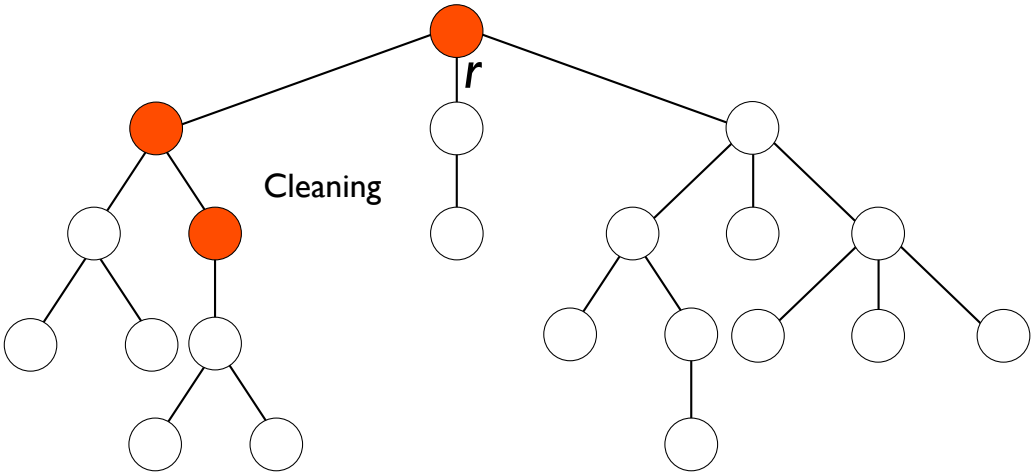
PIF sur l'arbre



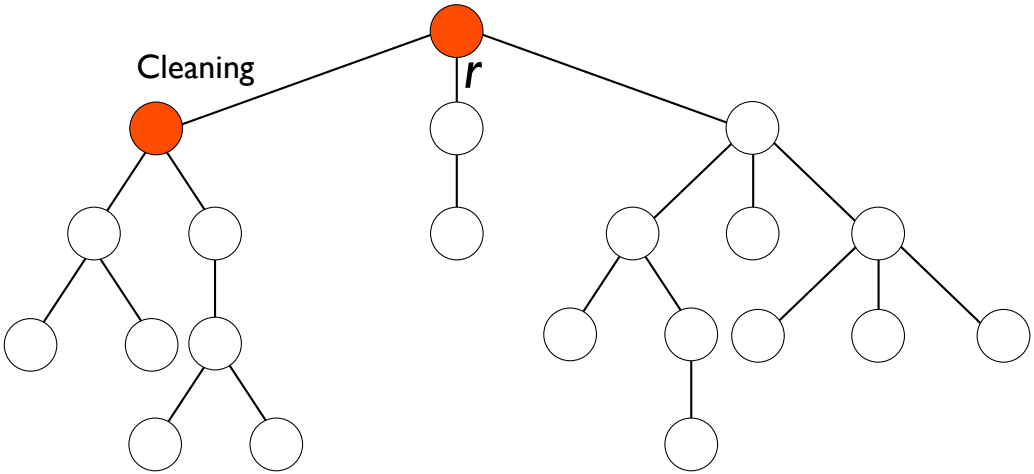
PIF sur l'arbre



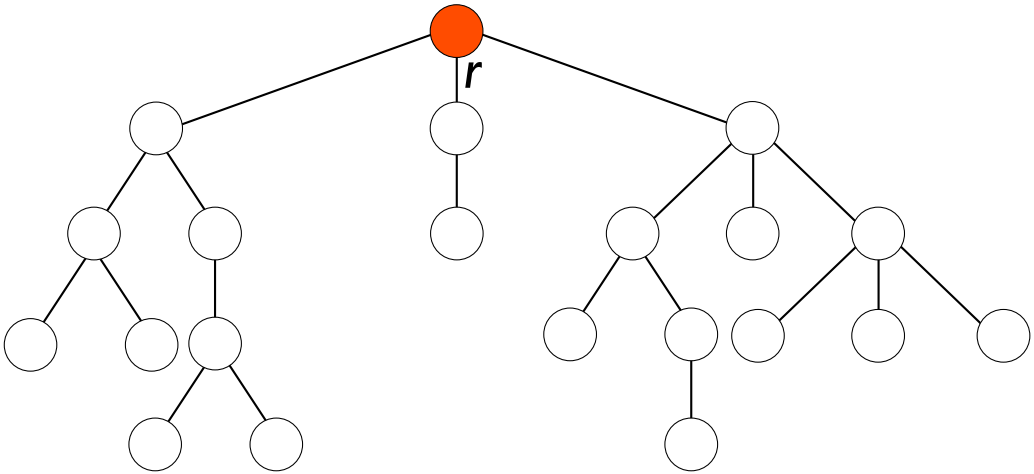
PIF sur l'arbre



PIF sur l'arbre



PIF sur l'arbre



Algorithm 1.1 ($\mathcal{PFCSD}$) PIF in Rooted Tree Networks.

Variable: S_p

$S_p \in \{B, F, C\}$ if p is an internal processor ($p \in I$).
$S_p \in \{B, C\}$ if p is the initiator ($p = \texttt{root}$).
$S_p \in \{F, C\}$ if p is a leaf processor ($p \in L$).

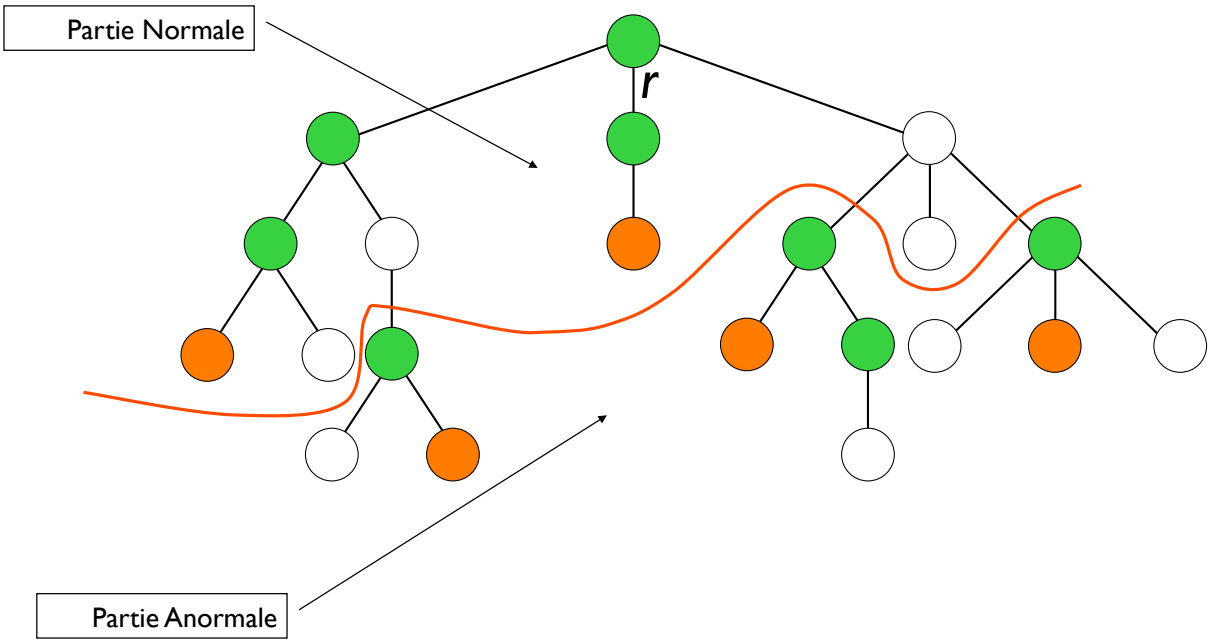
Notations:

P_p is the parent of p ($p \neq \texttt{root}$).
d is a descendant of p, i.e., $d \in D_p$.
q is a neighbor of p, i.e., $q \in N_p$.

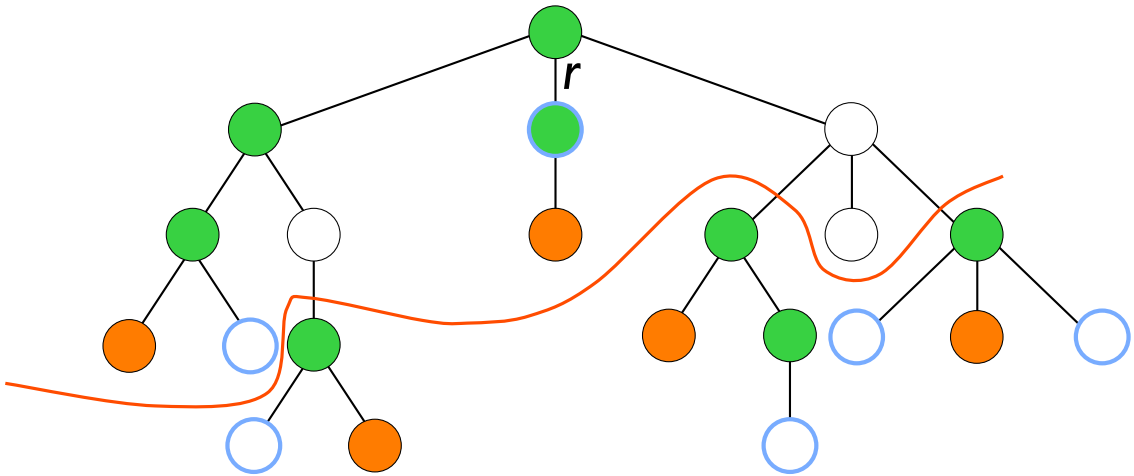
Actions:

		{For the internal processors}	
IB-action	::	$S_p = C \wedge S_{P_p} = B \wedge (\forall d \in D_p :: S_d = C)$	$\longrightarrow S_p := B;$
IF-action	::	$S_p = B \wedge (\forall d \in D_p :: S_d = F)$	$\longrightarrow S_p := F;$
IC-action	::	$S_p = F \wedge (\forall q \in N_p :: S_q \in \{F, C\})$	$\longrightarrow S_p := C;$
		{For root}	
rTB-action	::	$S_p = C \wedge (\forall q \in N_p :: S_q = C)$	$\longrightarrow S_p := B;$
rC-action	::	$S_p = B \wedge (\forall q \in N_p :: S_q = F)$	$\longrightarrow S_p := C;$
		{For the leaf processors}	
LF-action	::	$S_p = C \wedge S_{P_p} = B$	$\longrightarrow S_p := F$
LC-action	::	$S_p = F \wedge S_{P_p} \in \{F, C\}$	$\longrightarrow S_p := C;$

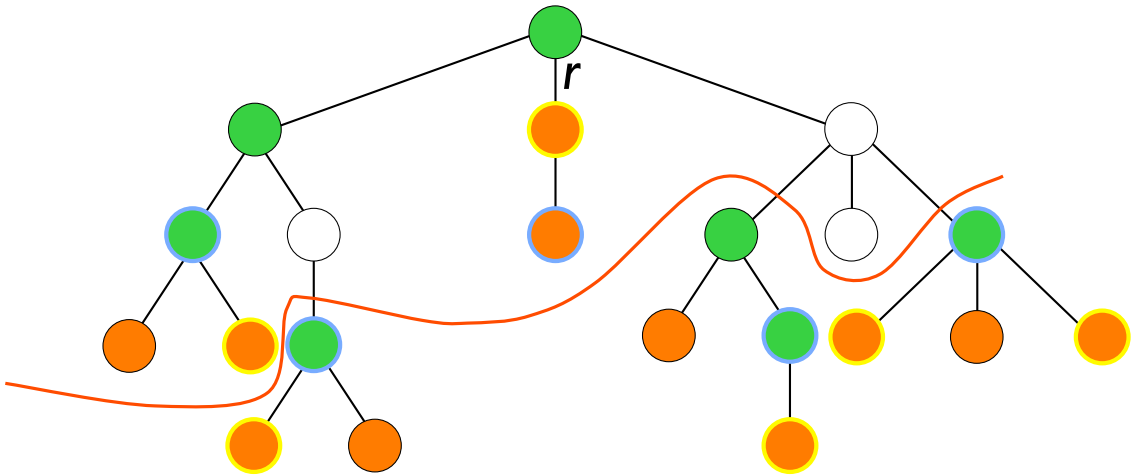
Temps de Stabilisation



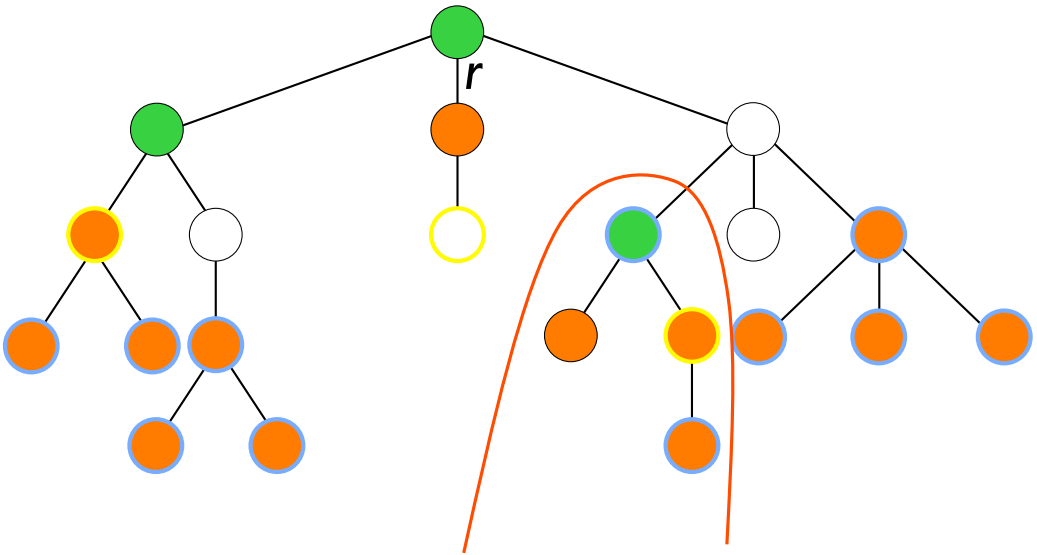
Temps de Stabilisation



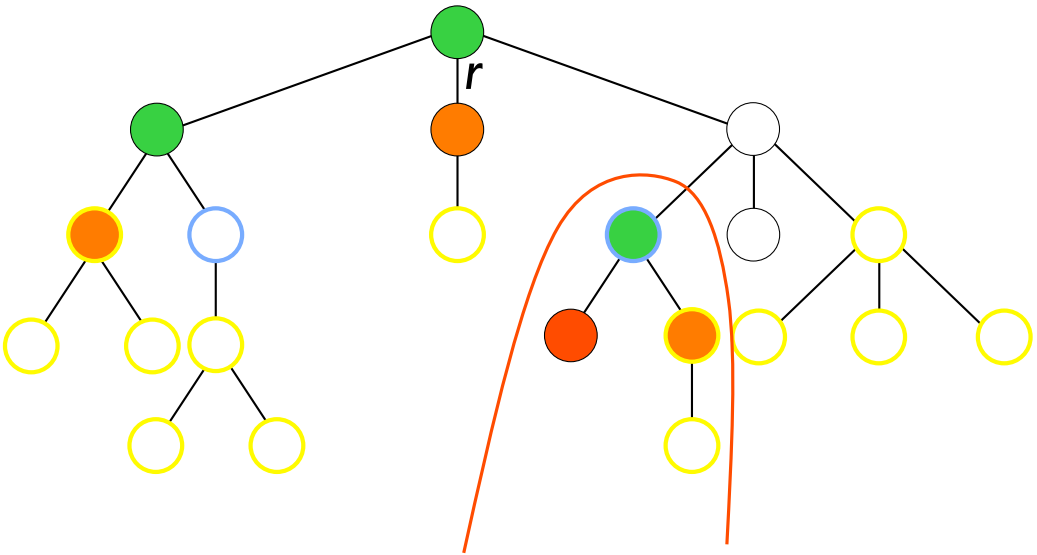
Temps de Stabilisation



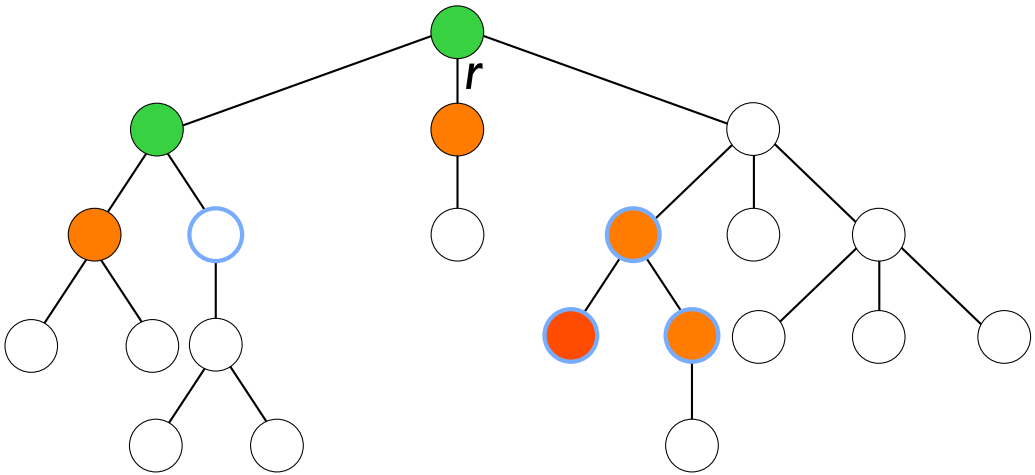
Temps de Stabilisation



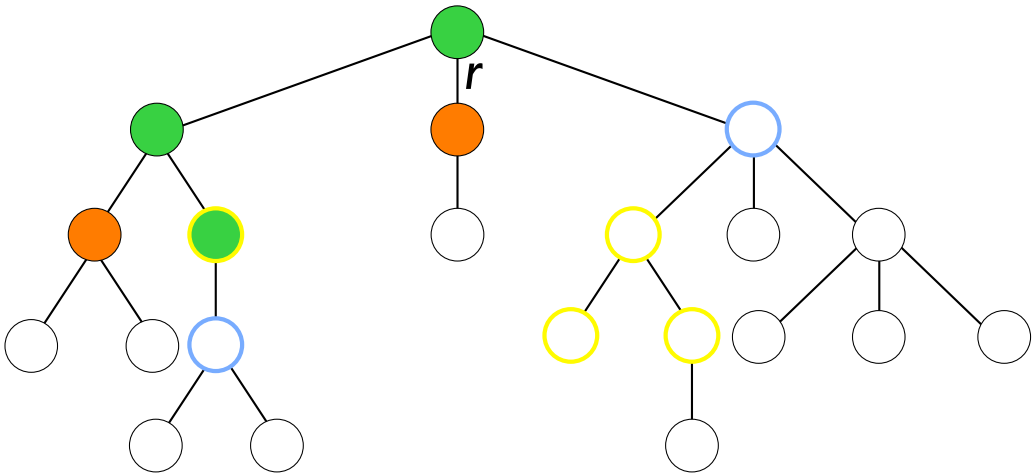
Temps de Stabilisation



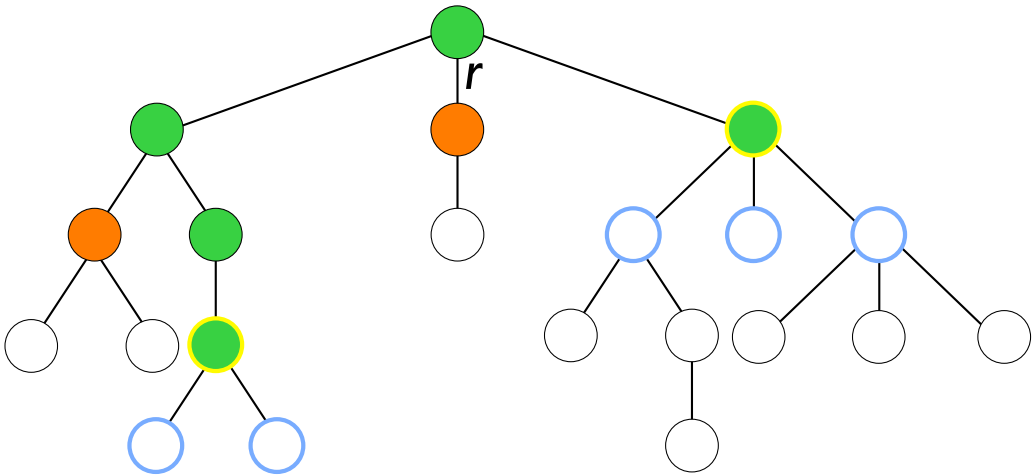
Temps de Stabilisation



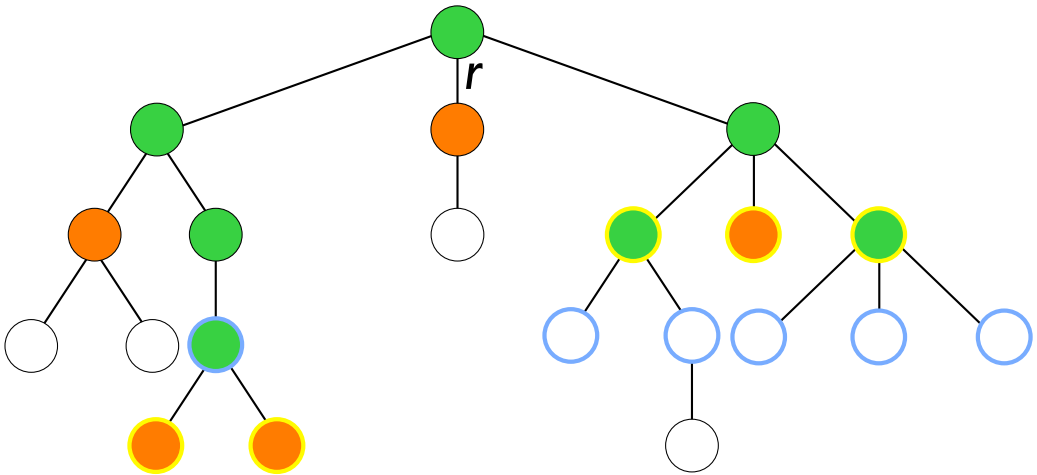
Temps de Stabilisation



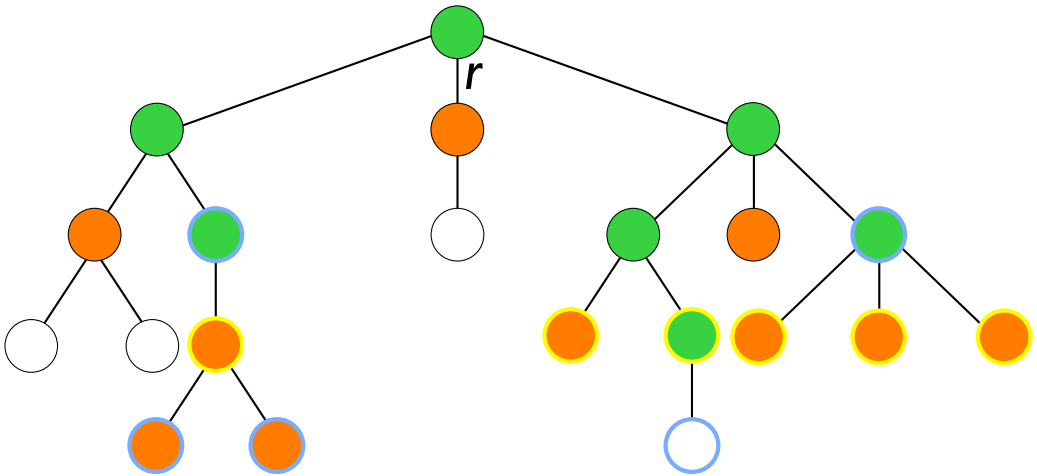
Temps de Stabilisation



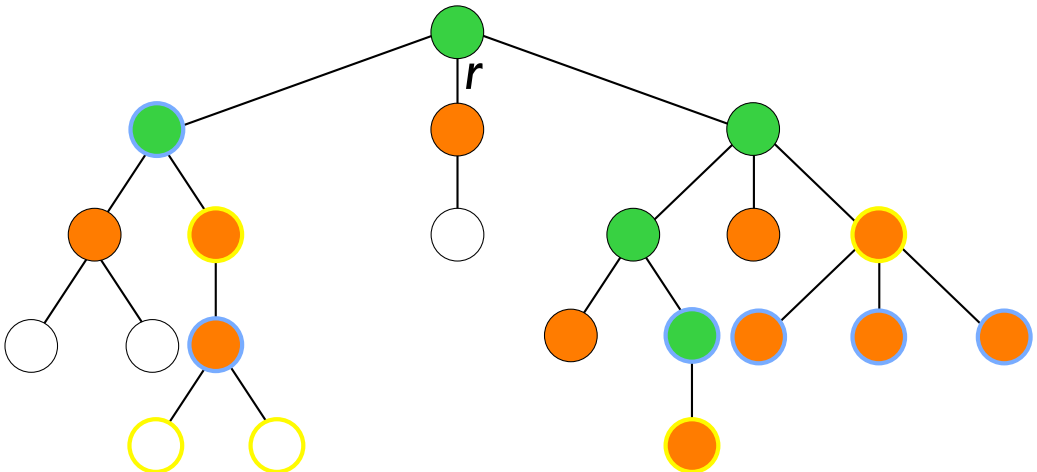
Temps de Stabilisation



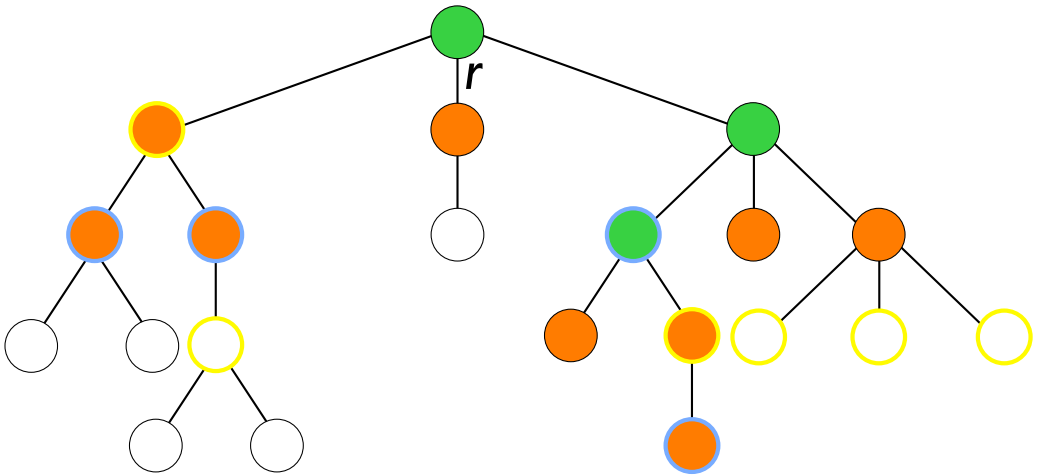
Temps de Stabilisation



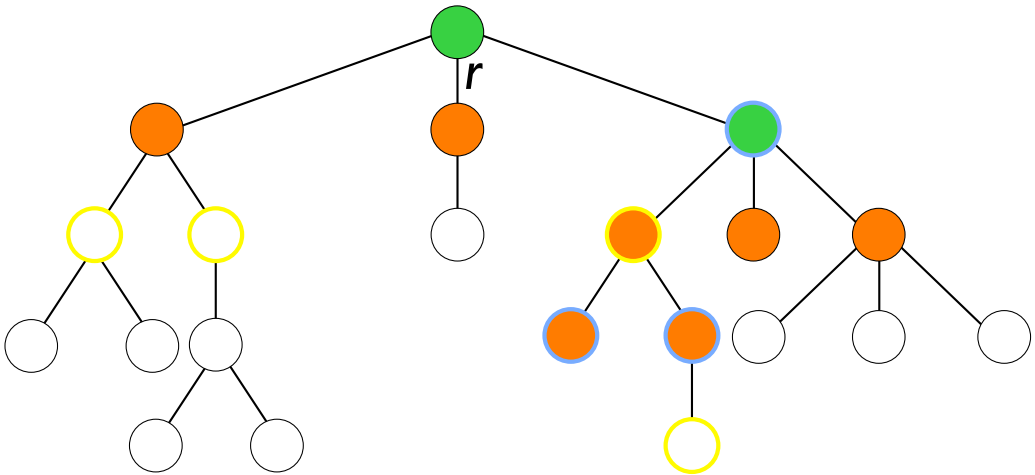
Temps de Stabilisation



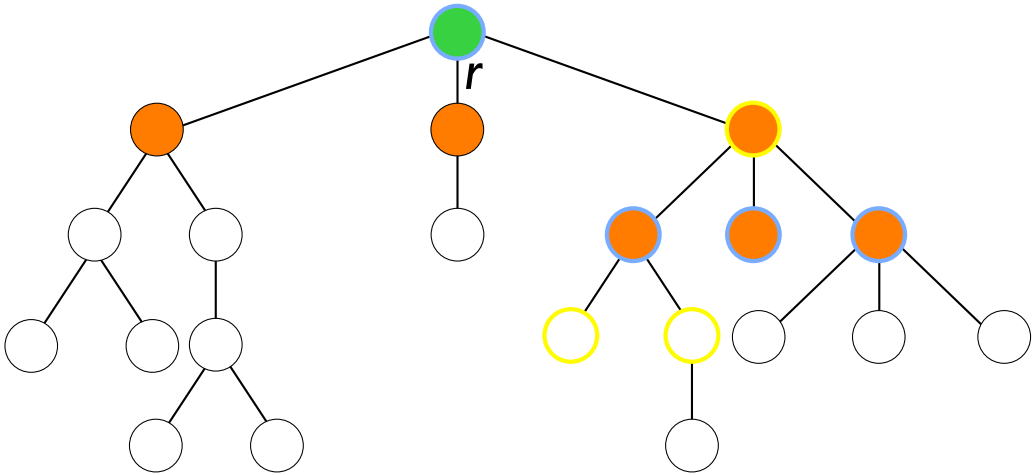
Temps de Stabilisation



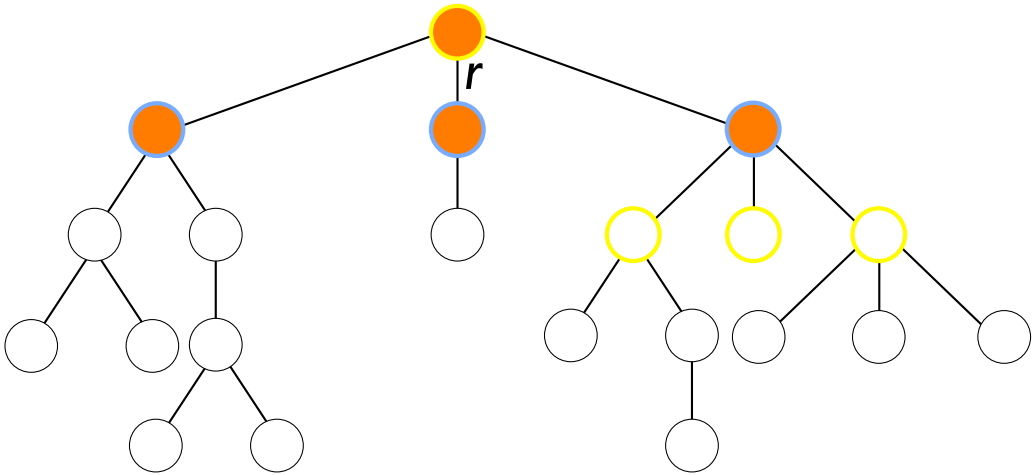
Temps de Stabilisation



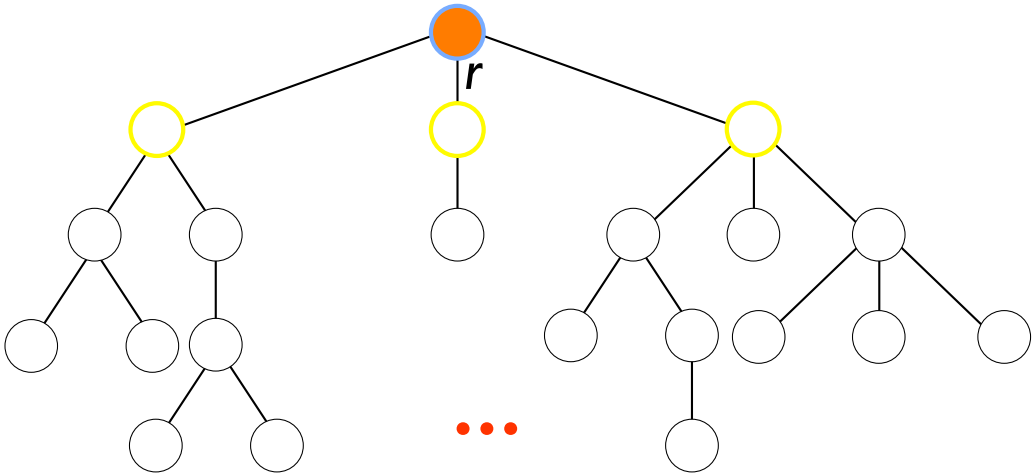
Temps de Stabilisation



Temps de Stabilisation



Temps de Stabilisation



Algorithm 1.2 (*fastPFCSD*) Fast Snap-Stabilizing PIF in Rooted Tree Networks.

Variable: S_p

$S_p \in \{B, F, C\}$ if p is an internal processor ($p \in I$).
$S_p \in \{B, C\}$ if p is the initiator ($p = \texttt{root}$).
$S_p \in \{F, C\}$ if p is a leaf processor ($p \in L$).

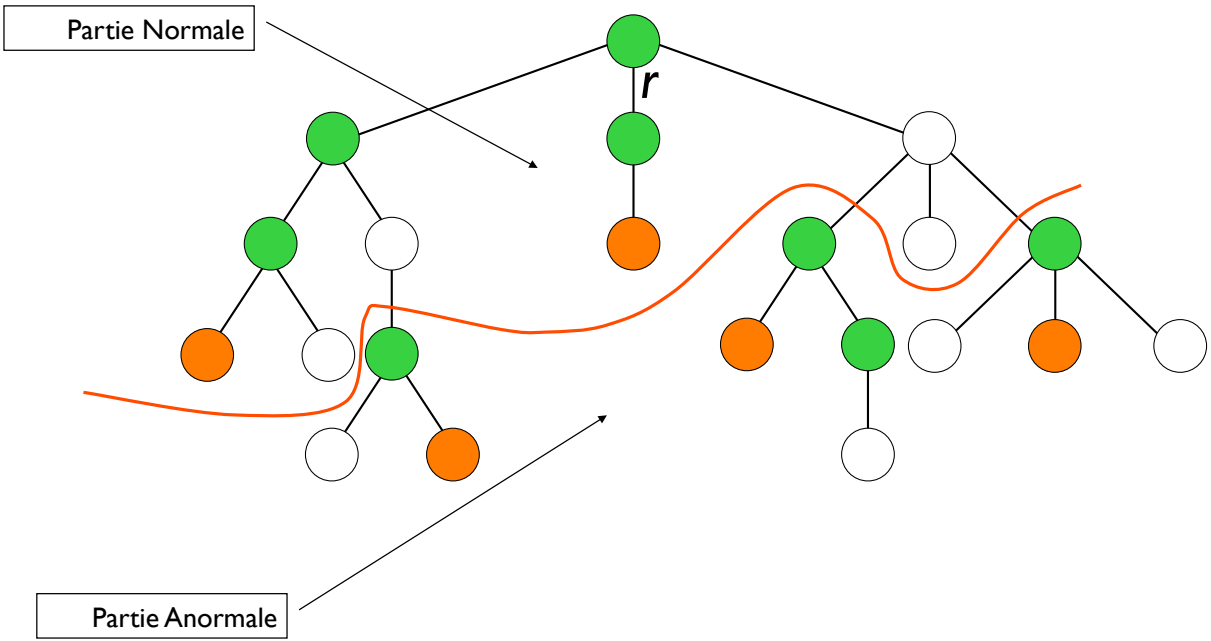
Notations:

P_p is the parent of p ($p \neq \texttt{root}$).
d is a descendant of p, i.e., $d \in D_p$.
q is a neighbor of p, i.e., $q \in N_p$.

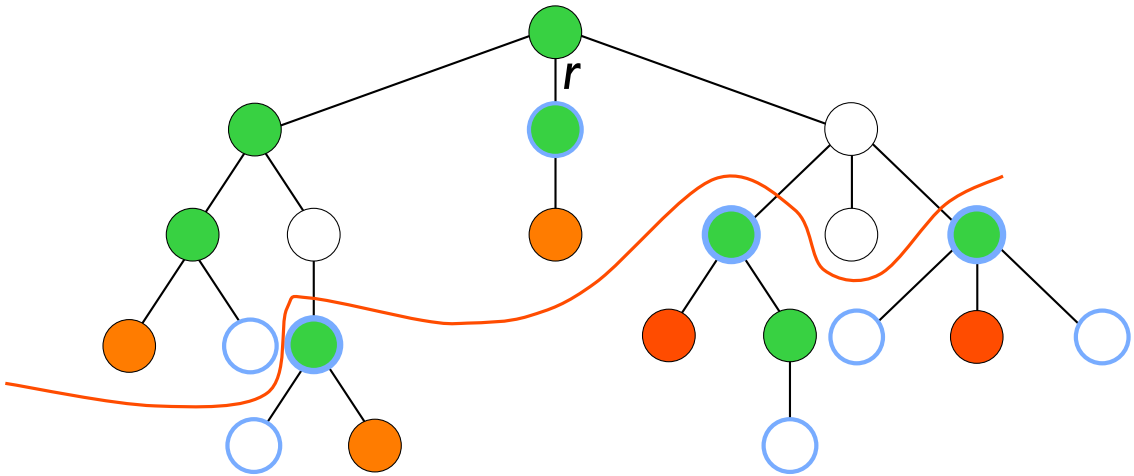
Actions:

		{For the internal processors}		
IB-action	::	$S_p = C \wedge S_{P_p} = B \wedge (\forall d \in D_p :: S_d = C)$	$\longrightarrow$	$S_p := B;$
IF-action	::	$S_p = B \wedge S_{P_p} = B \wedge (\forall d \in D_p :: S_d = F)$	$\longrightarrow$	$S_p := F;$
IC-action	::	$S_p = F \wedge (\forall q \in N_p :: S_q \in \{F, C\})$	$\longrightarrow$	$S_p := C;$
ICorrection	::	$S_p = B \wedge S_{P_p} \in \{F, C\}$	$\longrightarrow$	$S_p := C;$
		{For root}		
rTB-action	::	$S_p = C \wedge (\forall q \in N_p :: S_q = C)$	$\longrightarrow$	$S_p := B;$
rC-action	::	$S_p = B \wedge (\forall q \in N_p :: S_q = F)$	$\longrightarrow$	$S_p := C;$
		{For the leaf processors}		
LF-action	::	$S_p = C \wedge S_{P_p} = B$	$\longrightarrow$	$S_p := F$
LC-action	::	$S_p = F \wedge S_{P_p} \in \{F, C\}$	$\longrightarrow$	$S_p := C;$

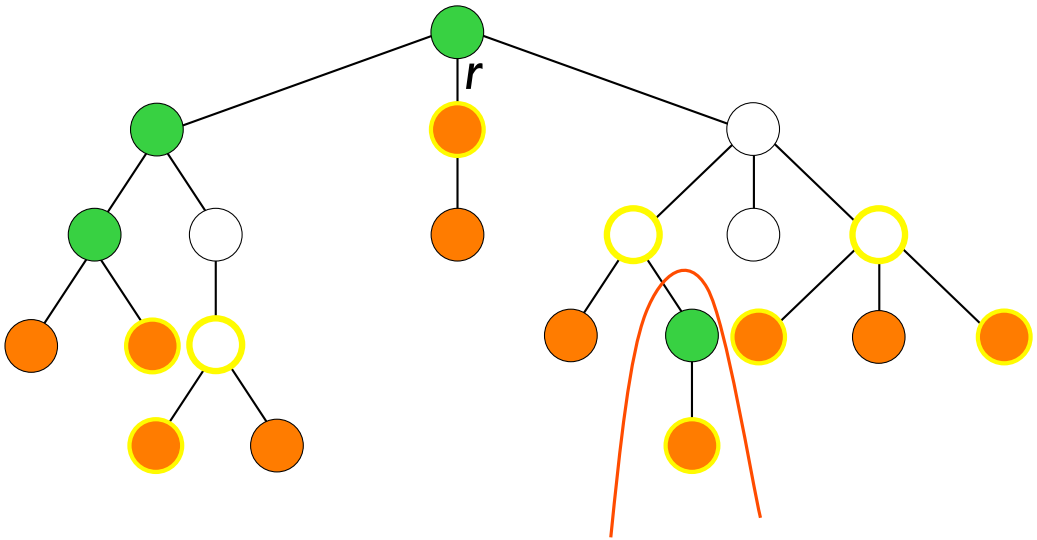
Temps de Stabilisation



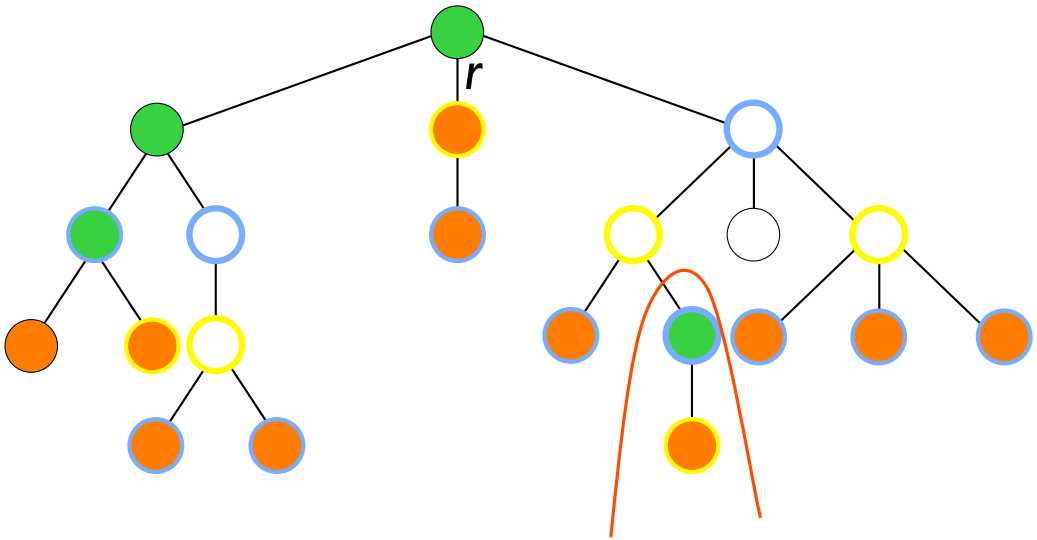
Temps de Stabilisation



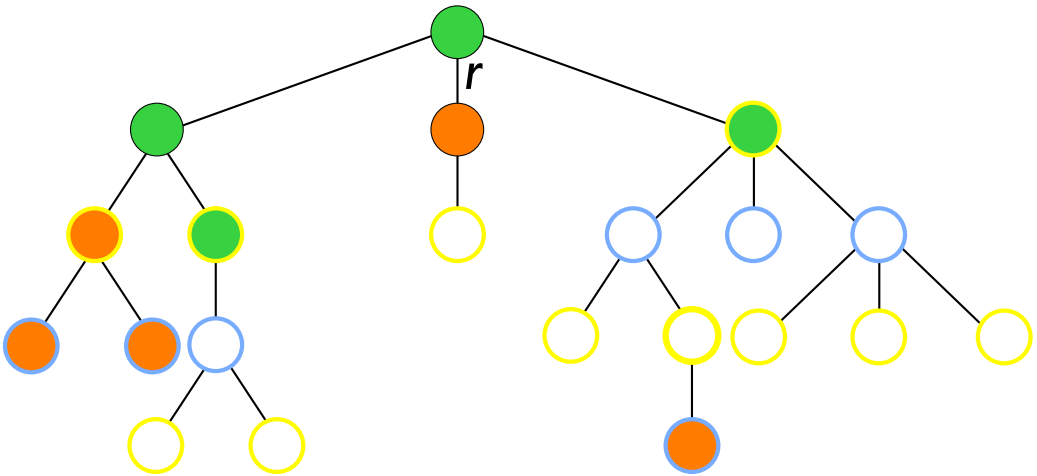
Temps de Stabilisation



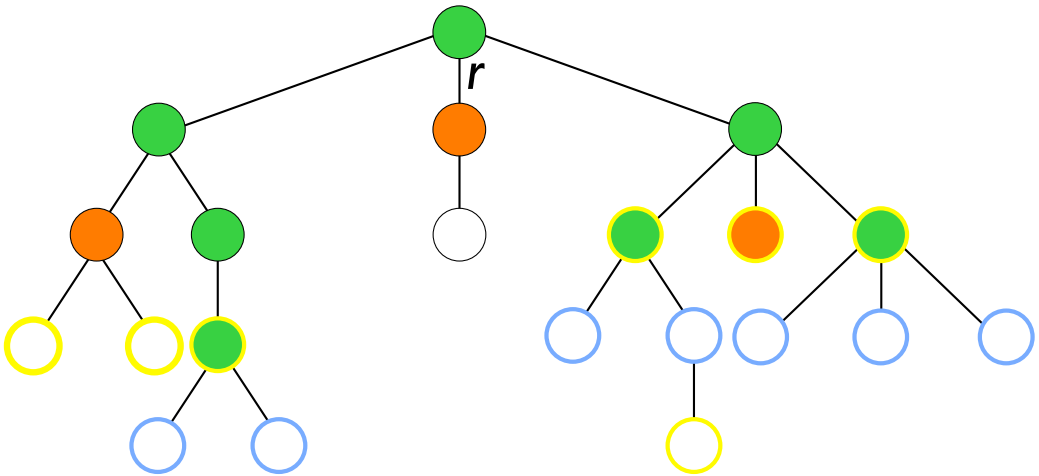
Temps de Stabilisation



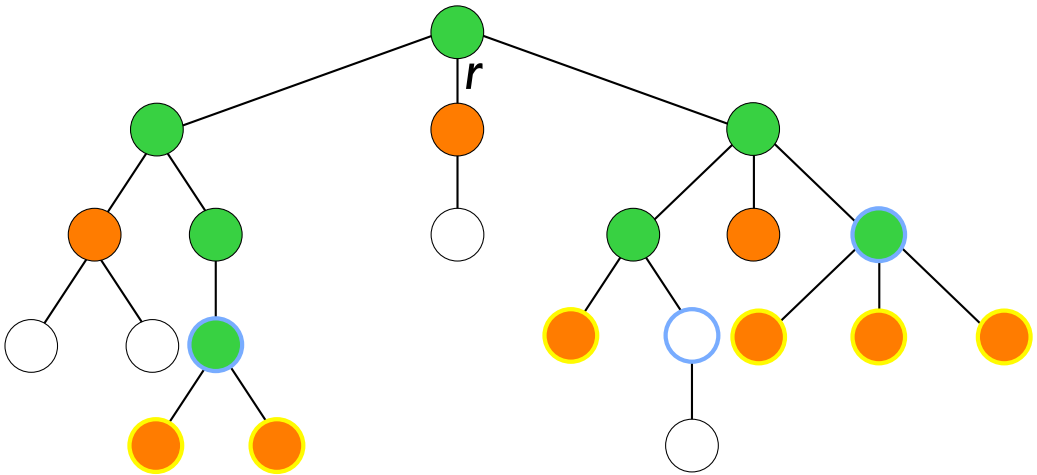
Temps de Stabilisation



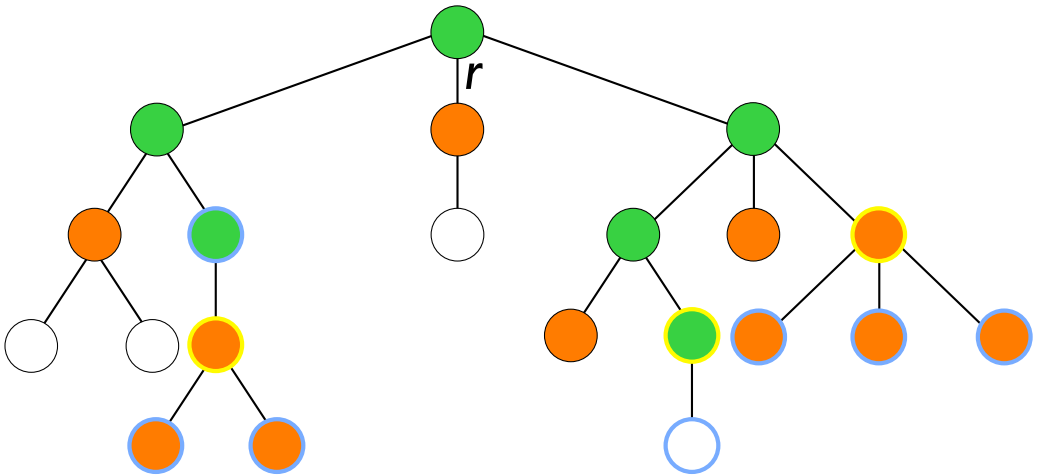
Temps de Stabilisation



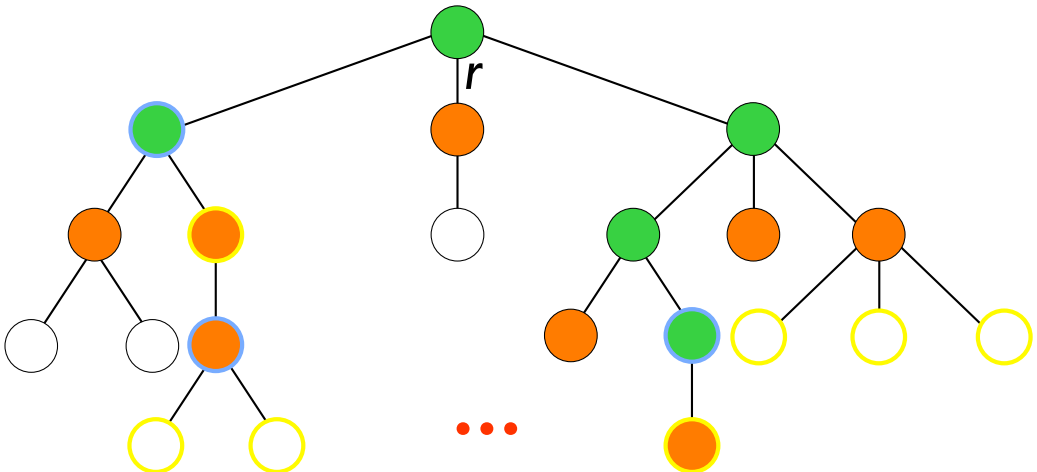
Temps de Stabilisation



Temps de Stabilisation



Temps de Stabilisation



Algorithme amélioré instantanément stabilisé

Peut être retardé d'au plus *l* étape !

Self-Stabilizing Compiler

- GOAL
- «Universal» Tool to Transform (Compile) any non self-stabilizing distributed algorithm into a self-stabilizing one
  [Katz and Perry 1993]

Snap-Stabilizing Compiler

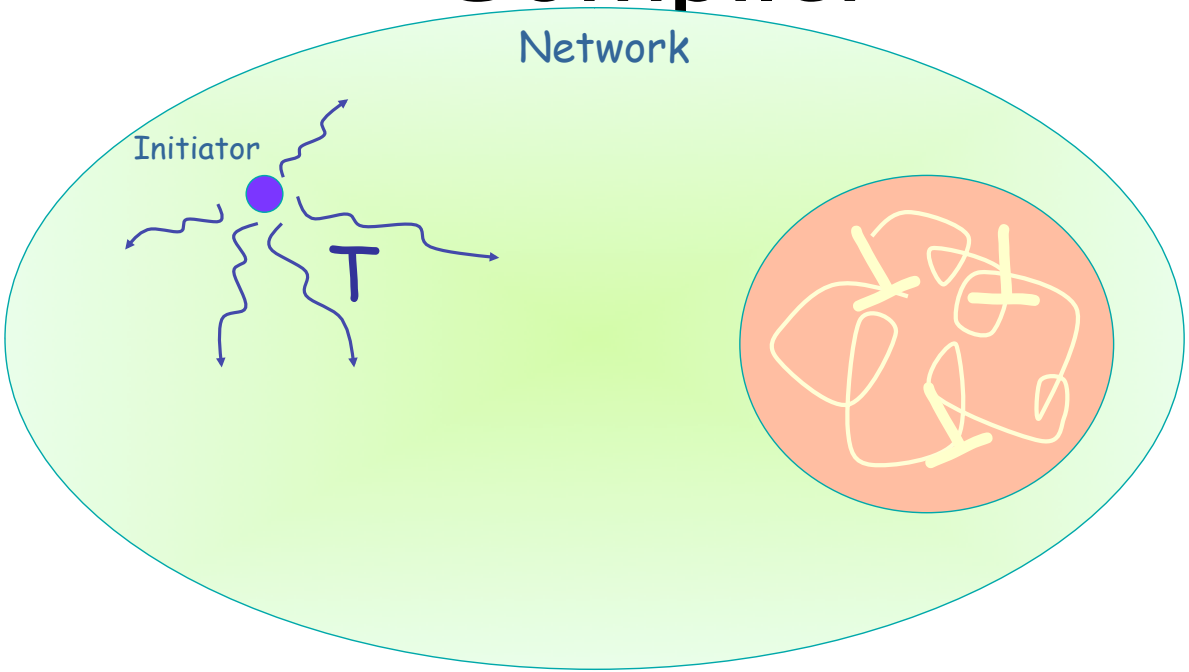
- GOAL
- «Universal» Tool to Transform (Compile) any non self-stabilizing distributed algorithm into a snap-stabilizing one
  [Cournier, Datta, Petit, Villain 2003]

Snap-Stabilizing Compiler

- IDEA
- Snap-stabilizing leader test
- Snap-stabilizing reset
- Snap-stabilizing snapshot
- Snap-stabilizing termination detection

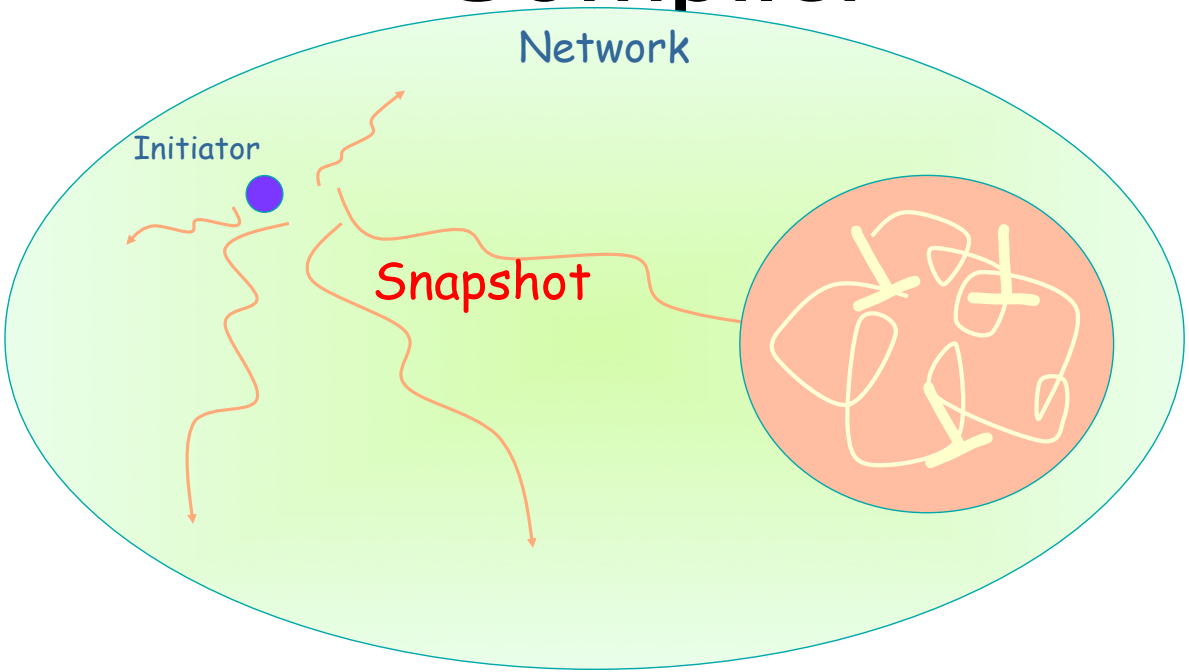
Snap-stabilizing PIF

Snap-Stabilizing Compiler



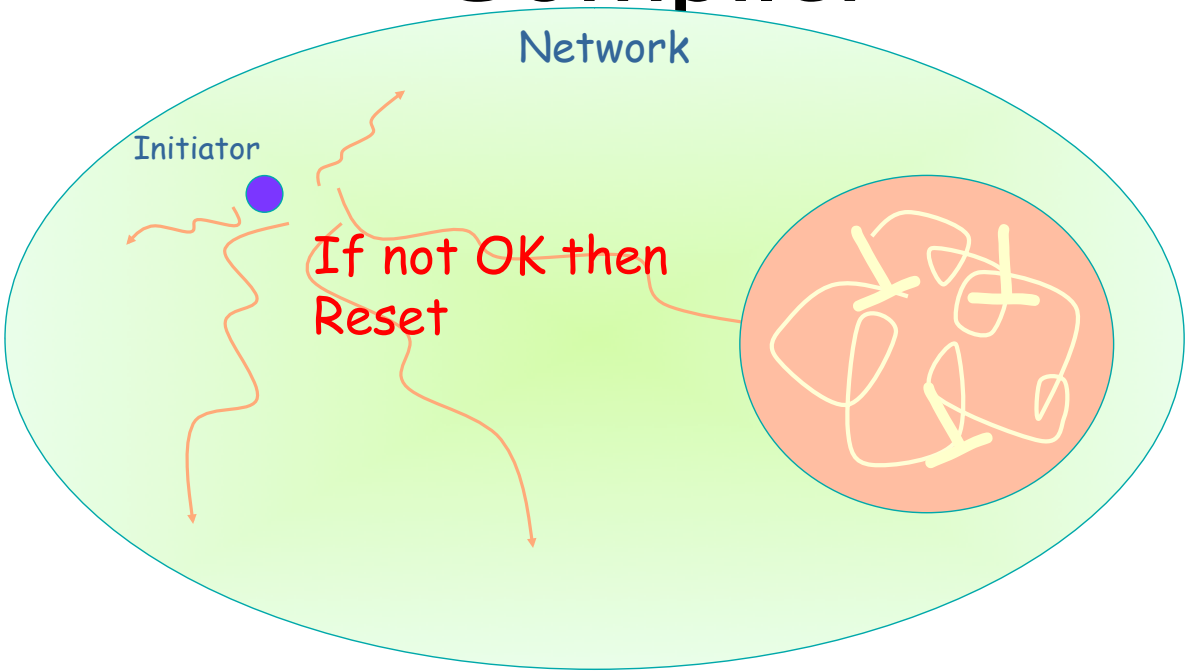
With a Single Initiator

Snap-Stabilizing Compiler



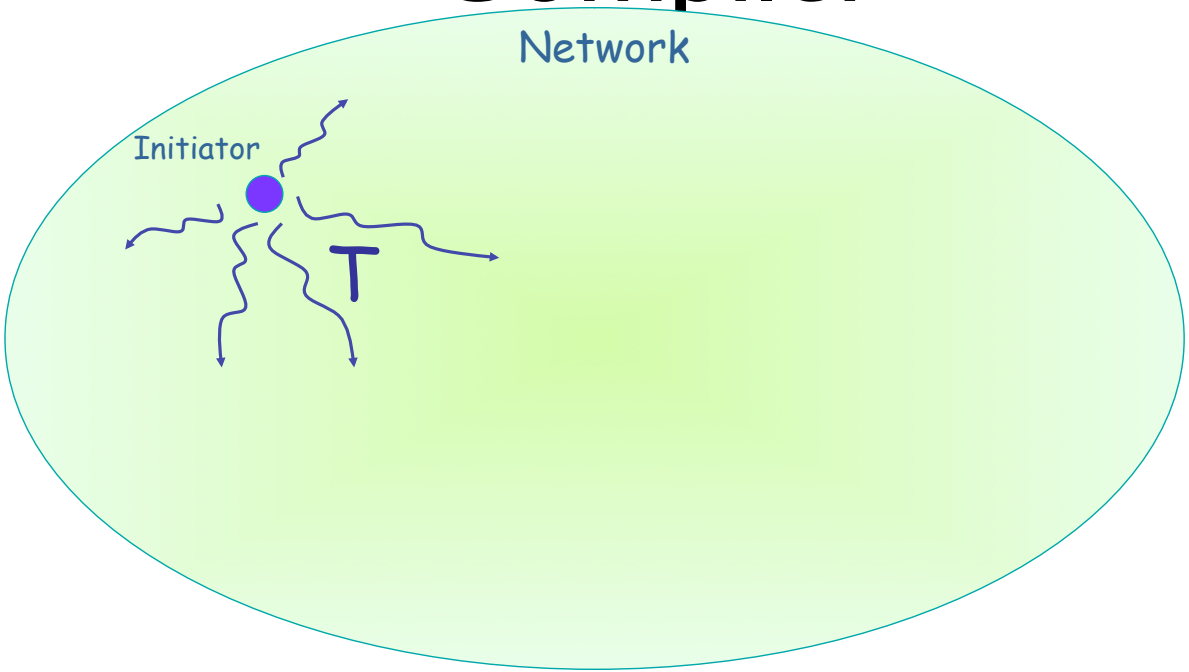
With a Single Initiator

Snap-Stabilizing Compiler



With a Single Initiator

Snap-Stabilizing Compiler



With a Single Initiator

Snap-Stabilizing Compiler

With a Multiple Initiators

- Same principle
- Snap-stabilizing leader test
- Snap-stabilizing leader election
- Snap-stabilizing reset
- Snap-stabilizing snapshot
- Snap-stabilizing termination detection