Cours ASPD
Principaux Algorithmes nRépartis
Telecom Nancy 2A IL

Dominique Méry
Telecom Nancy
Université de Lorraine

Année universitaire 2024-2025
5 mai 2025(1:47pm)

Summary

1. Auto-stabilisation
   - Généralités
   - Définition
   - Réseaux en anneau
   - Algorithme de Dijkstra
   - Algorithme de coloriage

Etat courant

Problème de l'élection
Problème de l'élection
Election dans un graphe acyclique

1 Auto-stabilisation
Généralités
Définition
Réseaux en anneau
Algorithme de Dijkstra
Algorithme de coloriage

Election du leader

- Un algorithme d'élection dun leader satisfait les propriétés suivantes :
  - Chaque processus ou site du réseau exécute le même algorithme.
  - L'algorithme est décentralisé : le calcul peut être initialisé par un sous-ensemble arbitraire de processus
  - L'algorithme atteint une configuration terminale pour chaque calcul et dans toute configuration terminale accessible, il n'y a qu'un seul processus dans l'état de savoir qu'il est leader et tous les autres savent qu'ils ne sont pas leader c'est-à-dire perdus.
- Algorithmes considérés
  - Algorithme de LeLann et Chang et Roberts (anneau)
  - Algorithme IEEE 1394 (arborescence)

Problème de l'élection d'un leader

- Les sites d'un réseau n'ont pas conscience des autres sites non connectés à eux.
- Chaque site a une connaissance locale
- Organiser des calculs dans un réseau nécessite de coordiner les calculs : besoin d'un leader.
- Calculs répartis par ondelettes

Algorithme de LeLann

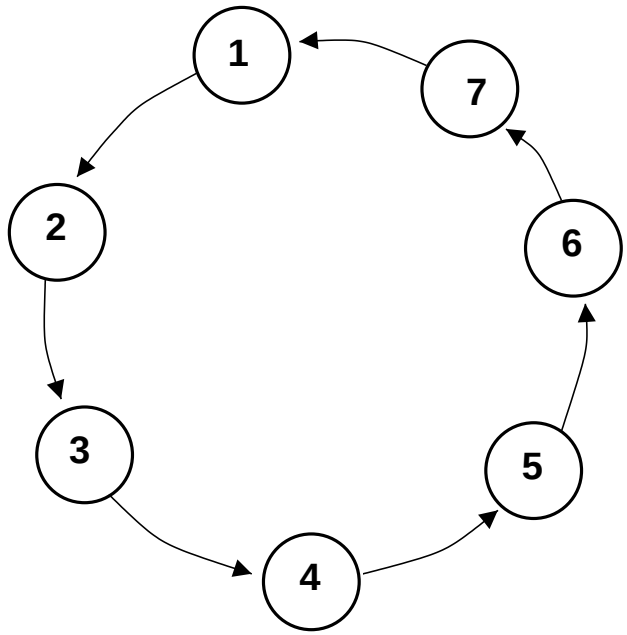
- Un ensemble de nœuds connectés en anneau
- Un sous-ensemble des nœuds est appelé l'ensemble des **candidats** à l'élection
- Chaque candidat maintient un ensemble de valeurs reçues des autres nœuds.
- Si un nœud reçoit le jeton d'un autre nœud et qu'il n'est pas candidat, il ne peut plus être candidat et passe le jeton à son voisin.
- Quand un nœud a reçu son jeton envoyé, il regarde s'il est le jeton de numéro le plus petit et il est élu.

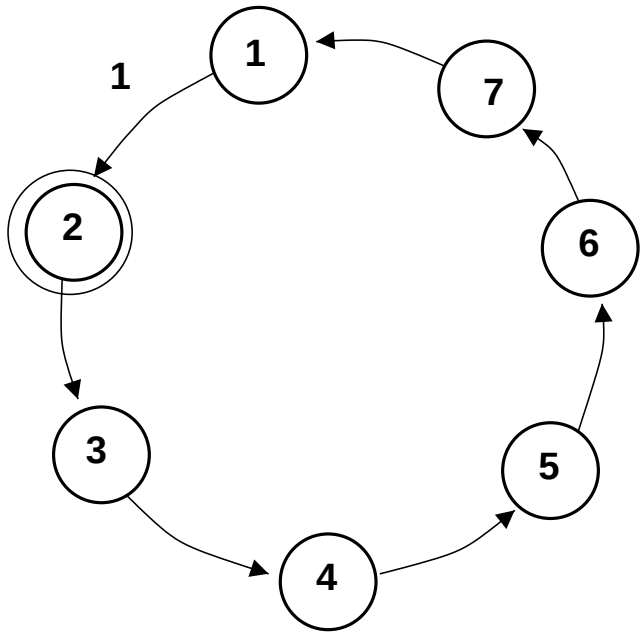
Algorithme de Chang et Roberts

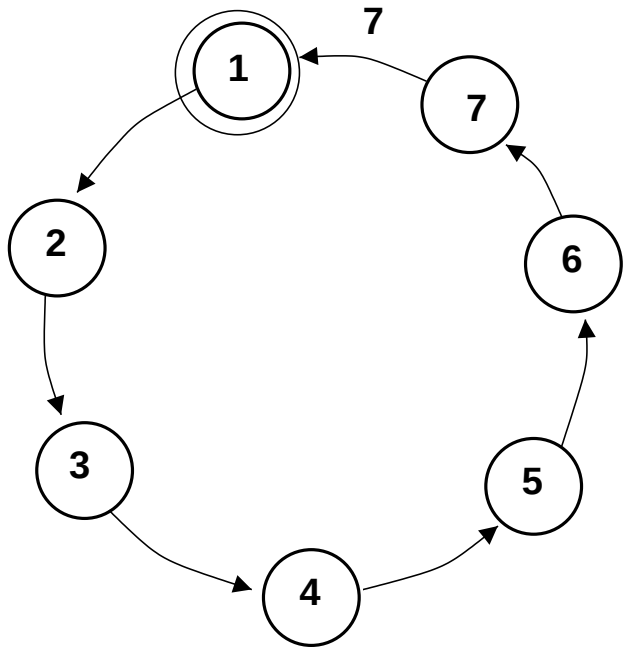
- Un ensemble de nœuds connectés en anneau
- Un sous-ensemble des nœuds est appelé l'ensemble des candidats à l'élection
- Si un nœud candidat reçoit un jeton plus petit que son jeton, il passe le jeton et se déclare pedu, sinon il garde le jeton.
- Si un nœud reçoit le jeton d'un autre nœud et qu'il n'est pas candidat, il ne peut plus être candidat et passe le jeton à son voisin.
- Quand un nœud a reçu son jeton envoyé, il est élu.

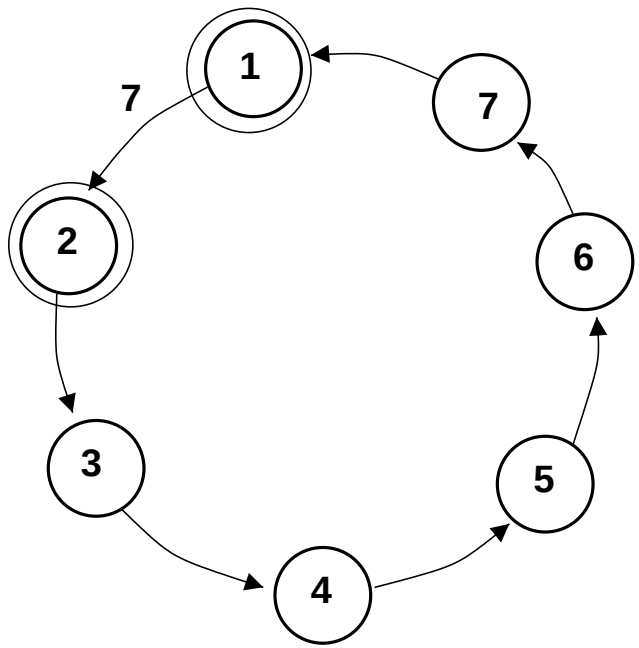
Algorithme LRC Chang et Roberts

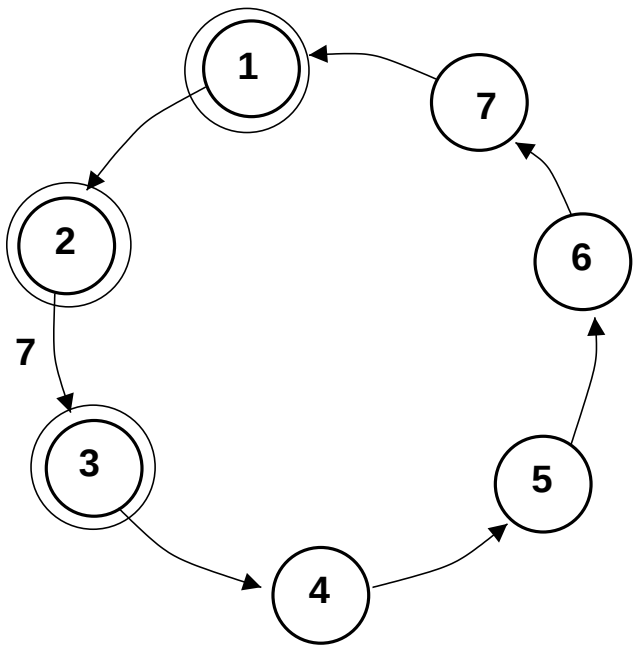
- Un réseau en anneau comporte des nœuds ayant un numéro différent.
- Un des nœuds ets le maximum du réseau
- Le processus d'élection transmet un jeton avec une valeur mise à jour à chaque nœud.

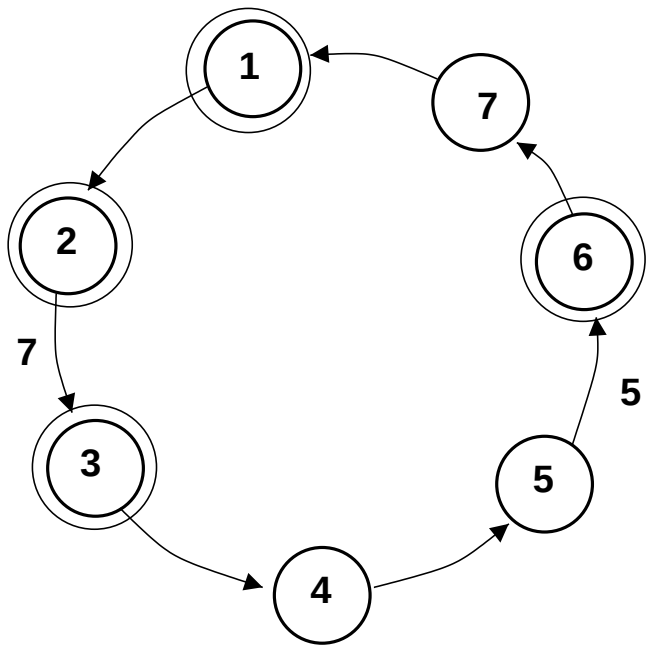

1
2
3
4
5
6
7

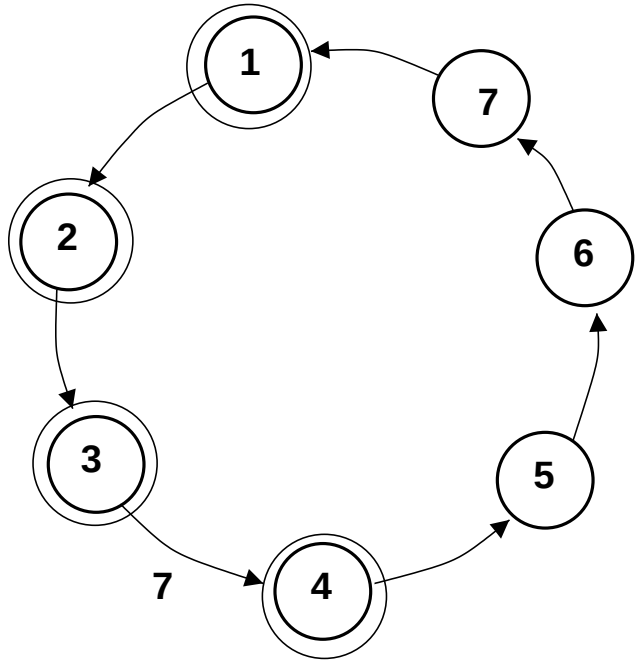

1
1
7
2
6
3
5
4

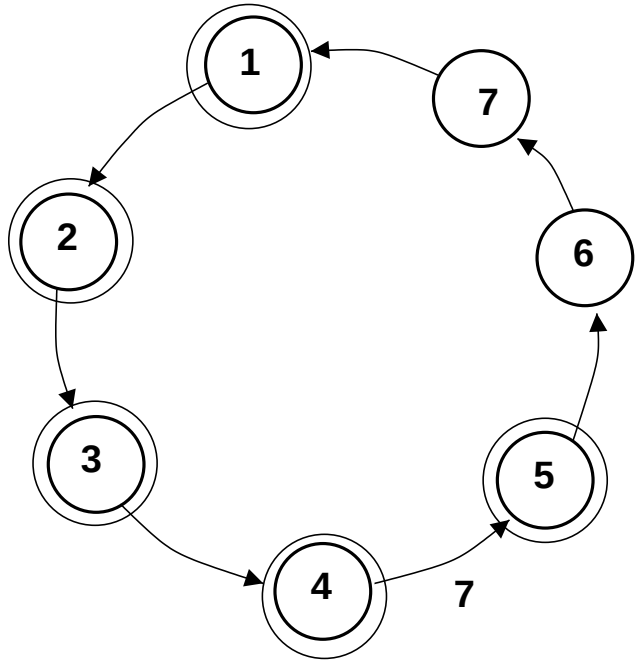

7
1
7
2
6
3
5
4


1
7
7
2
6
3
5
4


1
7
2
6
7
3
5
4


1
7
2
6
7
5
3
5
4


1
7
2
6
3
5
7
4


1
7
2
6
3
5
4
7

Algorithme LCR

- Seul le nœud de numéro maximal sera élu leader
- Les nœuds de numéro minimal restent dans l'état inconnu

Etat courant

Problème de l'élection

Problème de l'élection

Election dans un graphe acyclique

1 Auto-stabilisation

Généralités

Définition

Réseaux en anneau

Algorithme de Dijkstra

Algorithme de coloriage

Election du leader

- Un algorithme d'élection dun leader satisfait les propriétés suivantes :
  - Chaque processus ou site du réseau exécute le même algorithme.
  - L'algorithme est décentralisé : le calcul peut être initialisé par un sous-ensemble arbitraire de processus
  - L'algorithme atteint une configuration terminale pour chaque calcul et dans toute configuration terminale accessible, il n'y a qu'un seul processus dans l'état de savoir qu'il est leader et tous les autres savent qu'ils ne sont pas leader c'est-à-dire perdus.
- Algorithmes considérés
  - Algorithme de LeLann et Chang et Roberts (anneau)
  - Algorithme IEEE 1394 (arborescence)

Problème de l'élection d'un leader

- Les sites d'un réseau n'ont pas conscience des autres sites non connectés à eux.
- Chaque site a une connaissance locale
- Organiser des calculs dans un réseau nécessite de coordiner les calculs : besoin d'un leader.
- Calculs répartis par ondelettes

Algorithme de LeLann

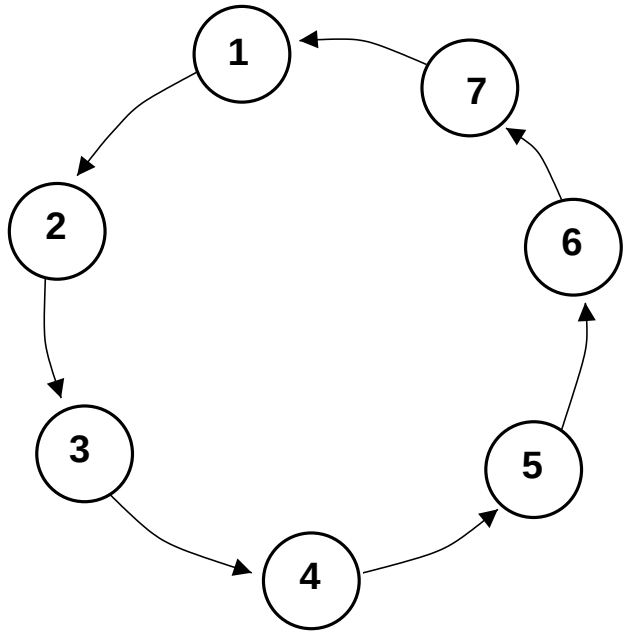
- Un ensemble de nœuds connectés en anneau
- Un sous-ensemble des nœuds est appelé l'ensemble des **candidats** à l'élection
- Chaque candidat maintient un ensemble de valeurs reçues des autres nœuds.
- Si un nœud reçoit le jeton d'un autre nœud et qu'il n'est pas candidat, il ne peut plus être candidat et passe le jeton à son voisin.
- Quand un nœud a reçu son jeton envoyé, il regarde s'il est le jeton de numéro le plus petit et il est élu.

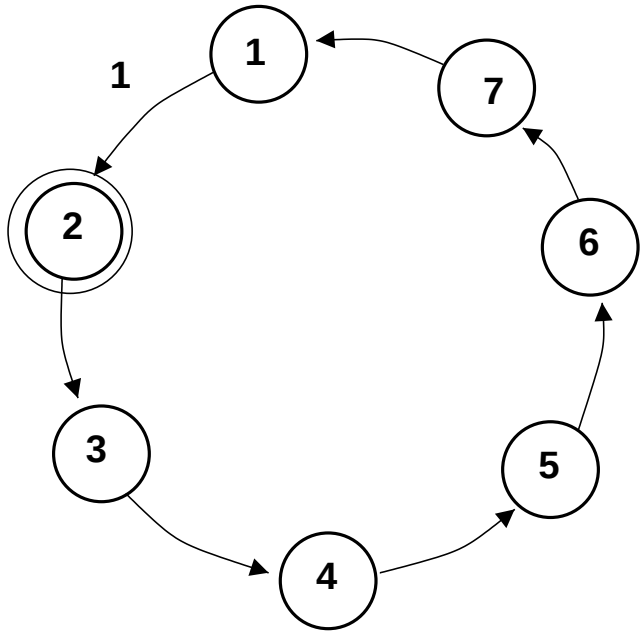
Algorithme de Chang et Roberts

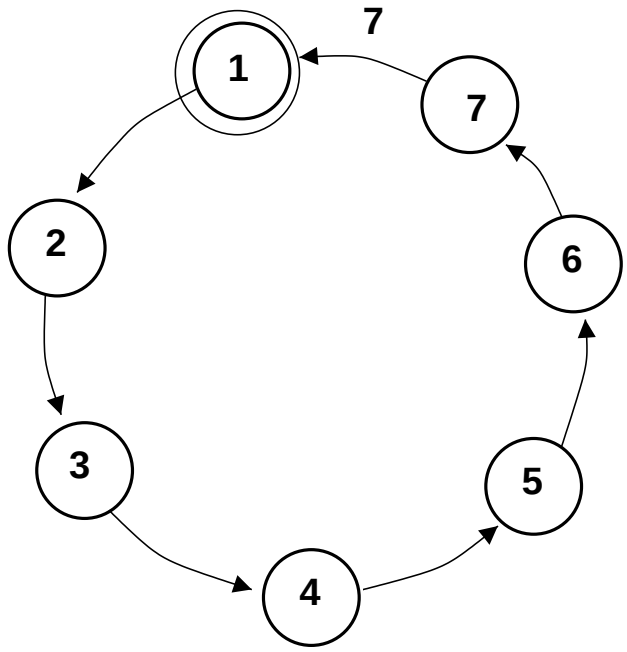
- Un ensemble de nœuds connectés en anneau
- Un sous-ensemble des nœuds est appelé l'ensemble des candidats à l'élection
- Si un nœud candidat reçoit un jeton plus petit que son jeton, il passe le jeton et se déclare pedu, sinon il garde le jeton.
- Si un nœud reçoit le jeton d'un autre nœud et qu'il n'est pas candidat, il ne peut plus être candidat et passe le jeton à son voisin.
- Quand un nœud a reçu son jeton envoyé, il est élu.

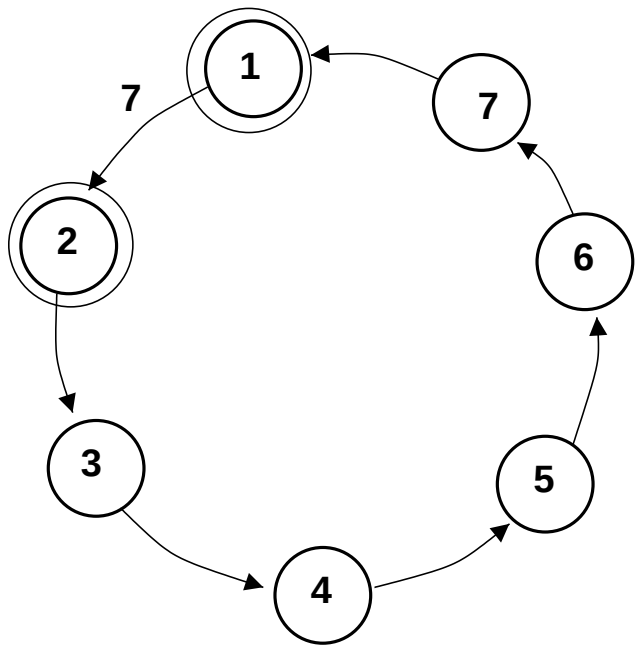
Algorithme LRC Chang et Roberts

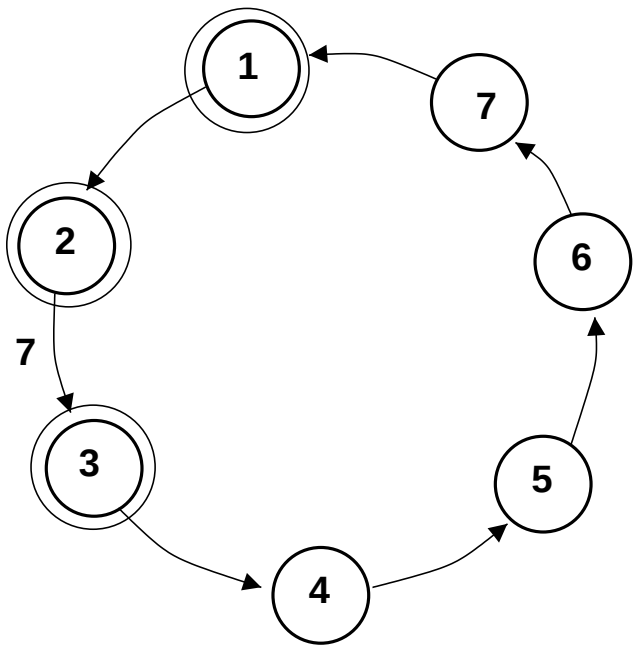
- Un réseau en anneau comporte des nœuds ayant un numéro différent.
- Un des nœuds ets le maximum du réseau
- Le processus d'élection transmet un jeton avec une valeur mise à jour à chaque nœud.

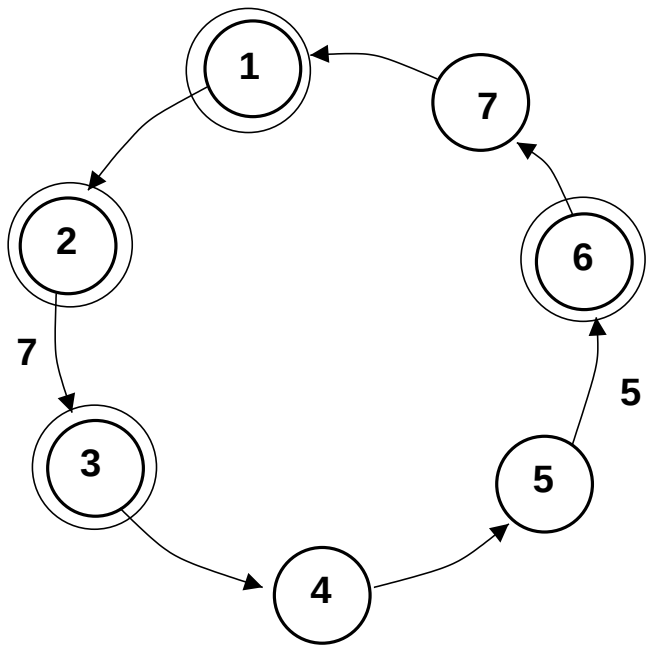

1
2
3
4
5
6
7

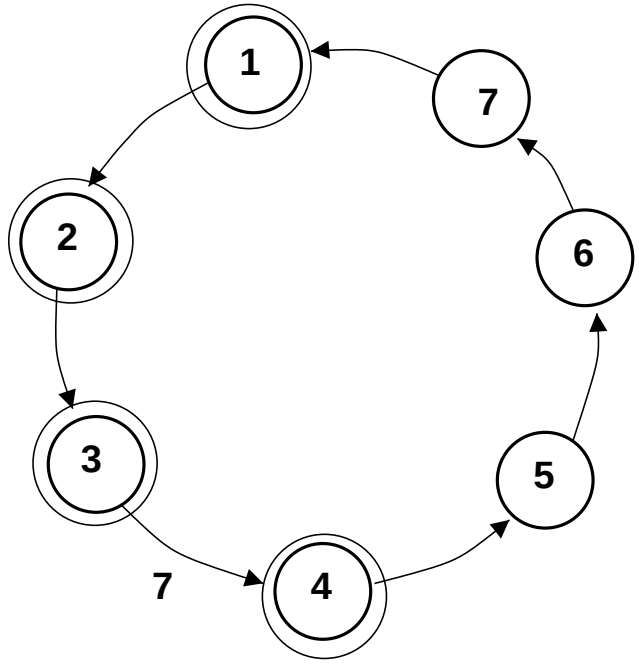

1
1
7
2
6
3
5
4

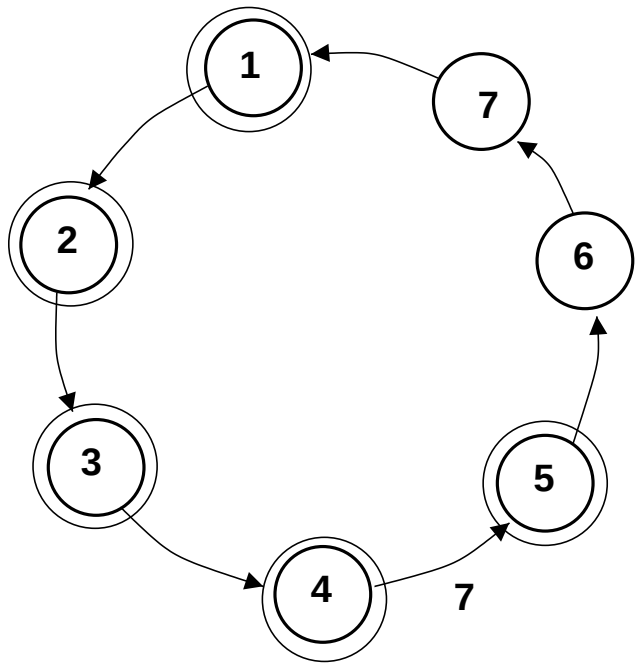

7
1
7
2
6
3
5
4


1
7
7
2
6
3
5
4


1
7
2
6
7
3
5
4


1
7
2
6
7
5
3
5
4


1
7
2
6
3
5
7
4


1
7
2
6
3
5
4
7

Algorithme LCR

- Seul le nœud de numéro maximal sera élu leader
- Les nœuds de numéro minimal restent dans l'état inconnu

Etat courant

Problème de l'élection
Problème de l'élection
Election dans un graphe acyclique

1 Auto-stabilisation
Généralités
Définition
Réseaux en anneau
Algorithme de Dijkstra
Algorithme de coloriage

(FireWire)

- standard international
- Intérêts commerciaux pour sa correction
- Sun, Apple, Philips, Microsoft, Sony, etc impliqués dans son développment
- Trois couches (physique, liaison, transaction)
- Le protocole d'étude est le Tree Identify Protocol
- Localisé dans la phase Bus Reset de la couche physique

Le Problème (1)

- Le bus est utilisé pour acheminer des siganux audio et vidéo digitalisés.
- Il est "hot-pluggable"
- Unités et périphériques peuvent être ajoutés ou retirés à tout instant.
- De tels changements sont suivis d'un bus reset
- L'élection du leader a lieu après un reset dans le réseau.
- Un leader doit être choisi pour agir en tant que manager du bus

Le Problème (2)

- Après un reset du bus tous les nœuds ont même statut.
- Tout nœud sait à quels nœuds il est directement connecté.
- Le réseau est connexe
- Le réseau est acyclique

Références (1)

BASE

- IEEE. *IEEE Standard for a High Performance Serial Bus. Std 1394-1995*. 1995
- IEEE. *IEEE Standard for a High Performance Serial Bus (supplement). Std 1394a-2000*. 2000

Références (2)

GENERAL

- N. Lynch. *Distributed Algorithms*. Morgan Kaufmann. 1996
- R. G. Gallager et al. *A Distributed Algorithm for Minimum Weight Spanning Trees*. IEEE Trans. on Prog. Lang. and Systems. 1983.

Références (3)

MODEL CHECKING

- D.P.L. Simons et al. *Mechanical Verification of the IEE 1394a Root Contention Protocol using Uppaal2* Springer International Journal of Software Tools for Technology Transfer. 2001
- H. Toetenel et al. *Parametric verification of the IEEE 1394a Root Contention Protocol using LPMC* Proceedings of the 7th International Conference on Real-time Computing Systems and Applications. IEEE Computer Society Press. 2000

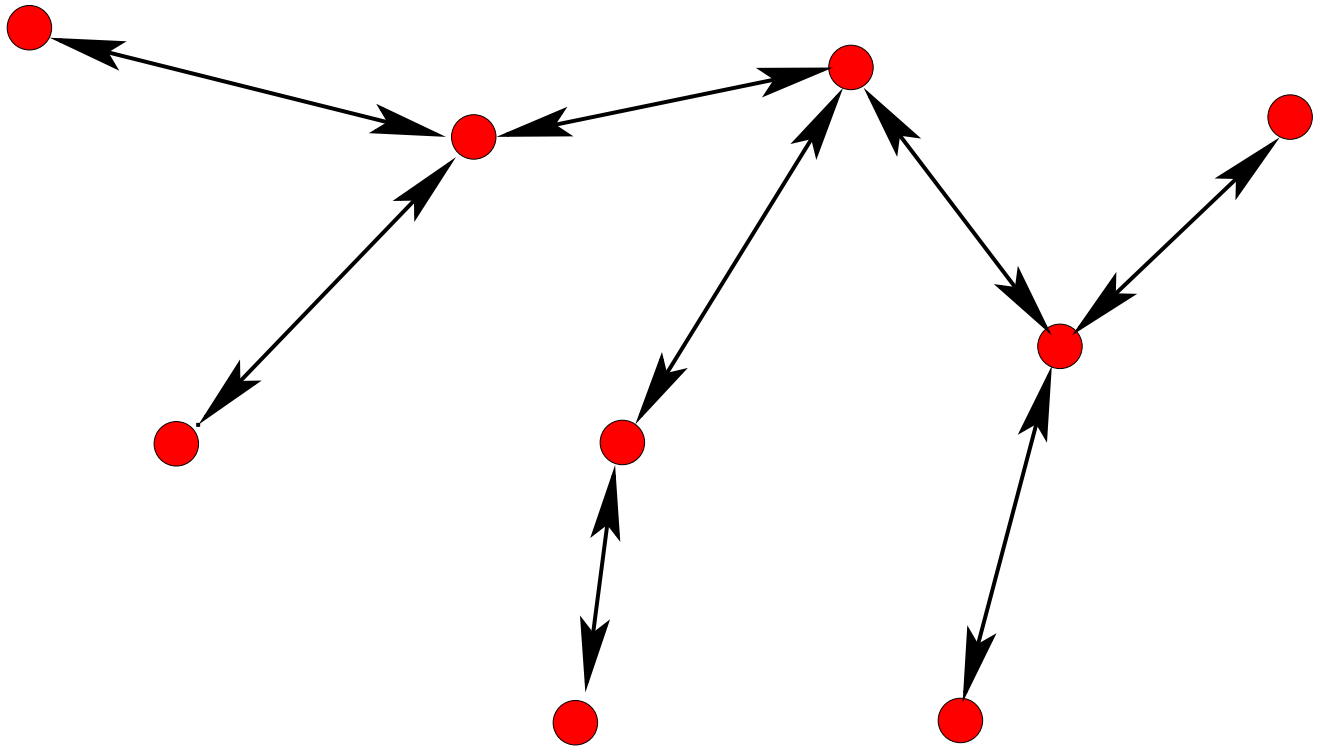
Références (4)

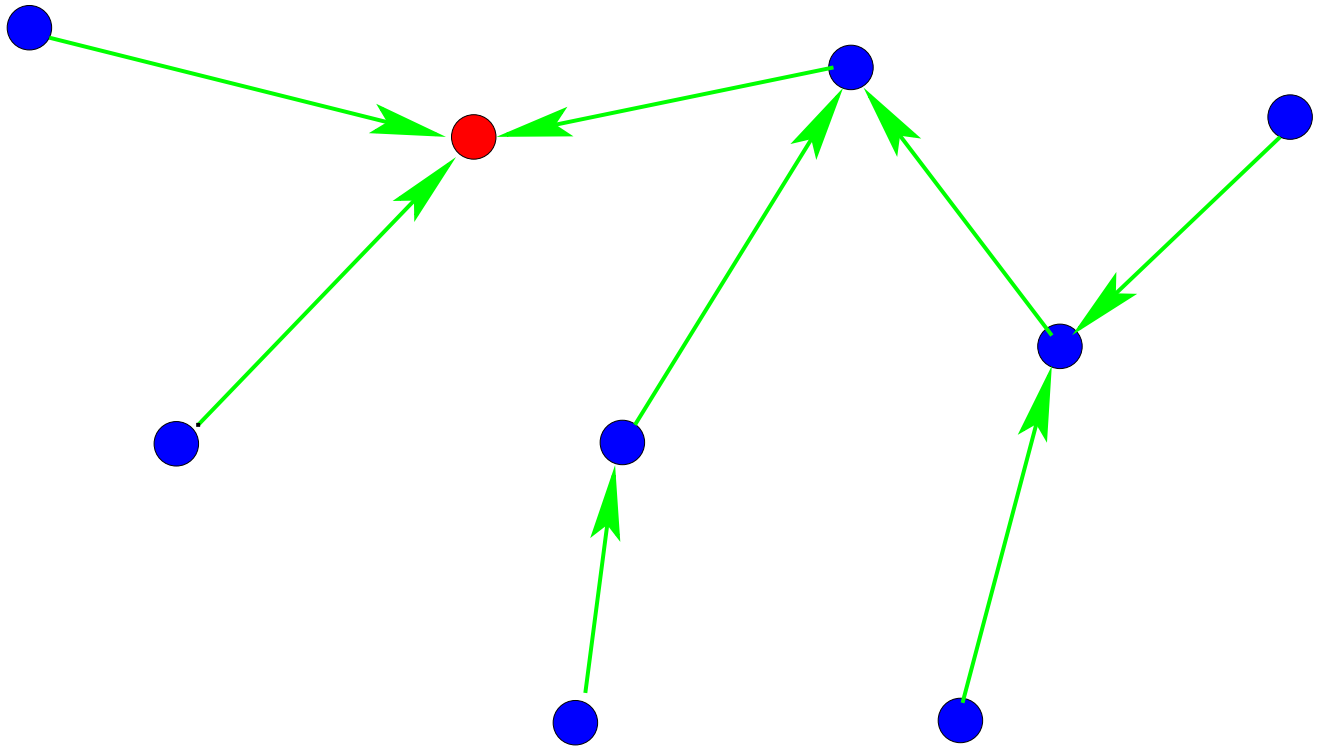
THEOREM PROVING

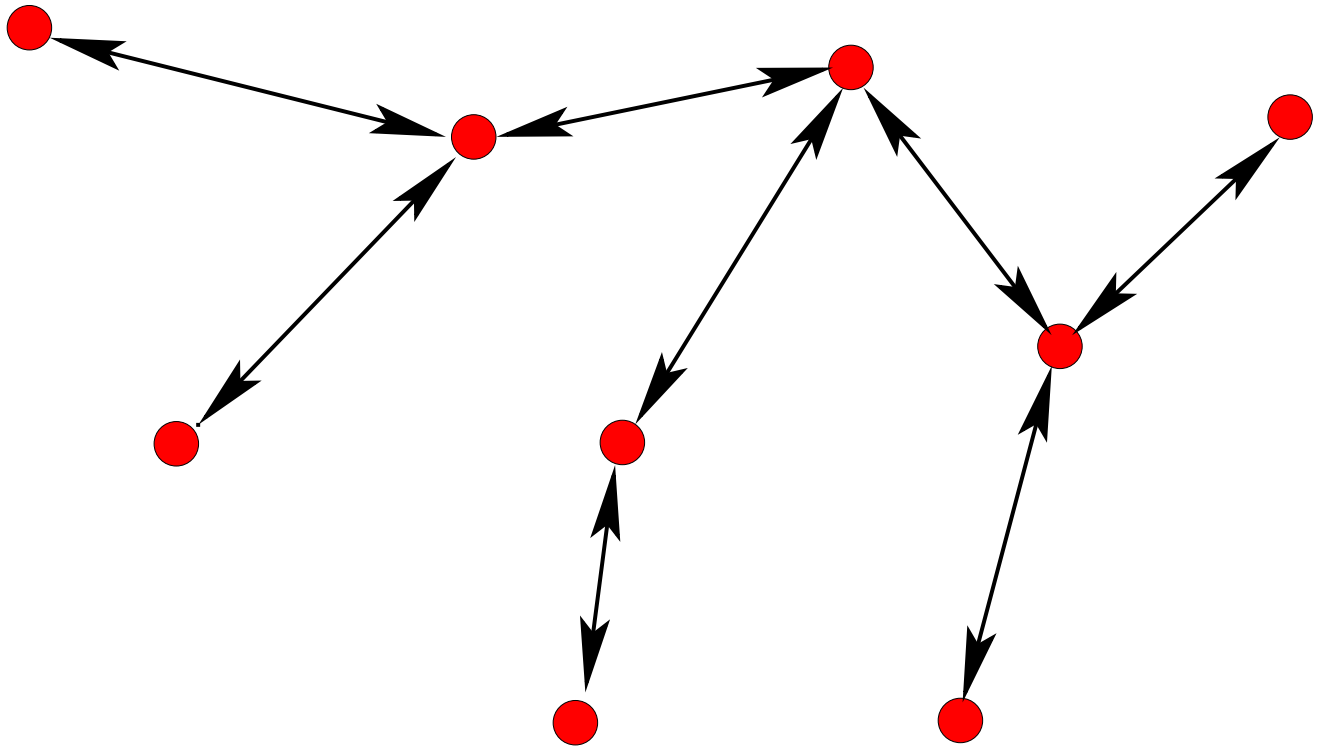
- M. Devillers et al. *Verification of the Leader Election : Formal Method Applied to IEEE 1394*. Formal Methods in System Design. 2000
- J.R. Abrial et al. *A Mechanically Proved and Incremental Development of IEEE 1394*. Formal Aspects of Computing, 2003.

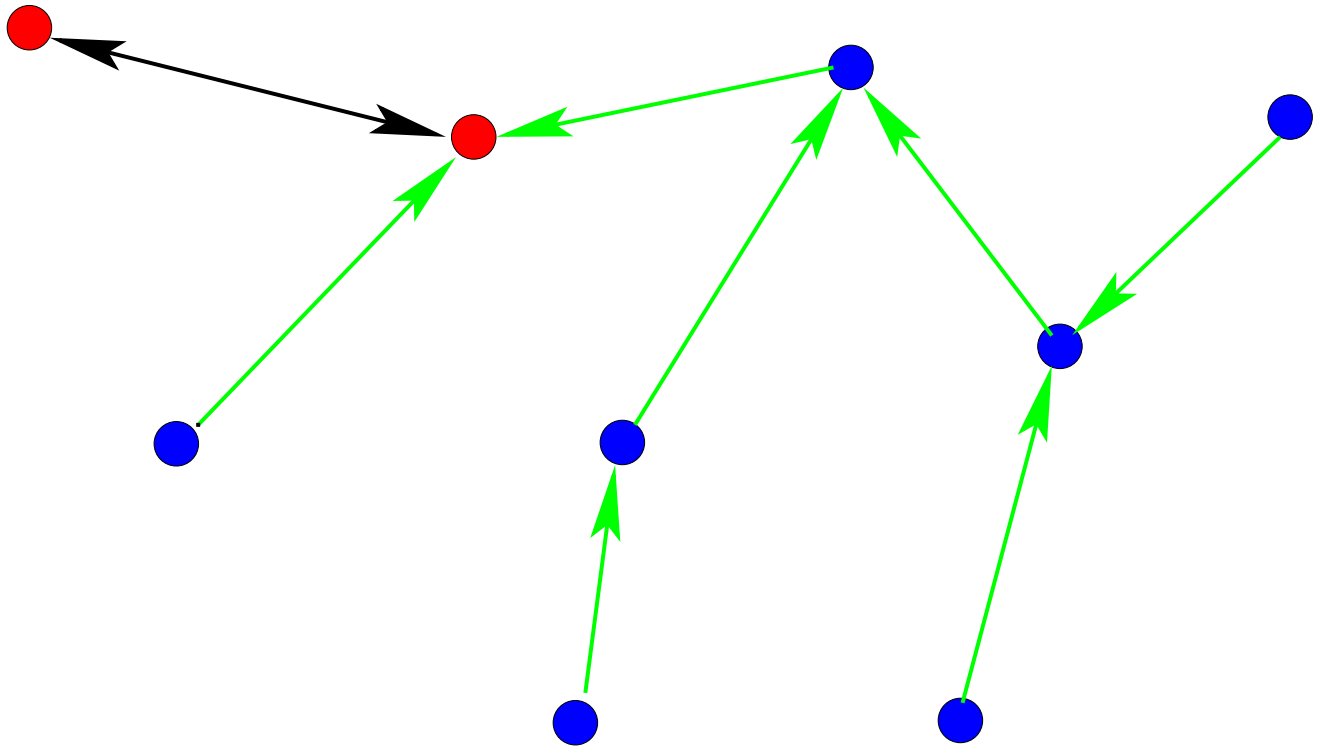
Propriétés informelles abstraites du protocole

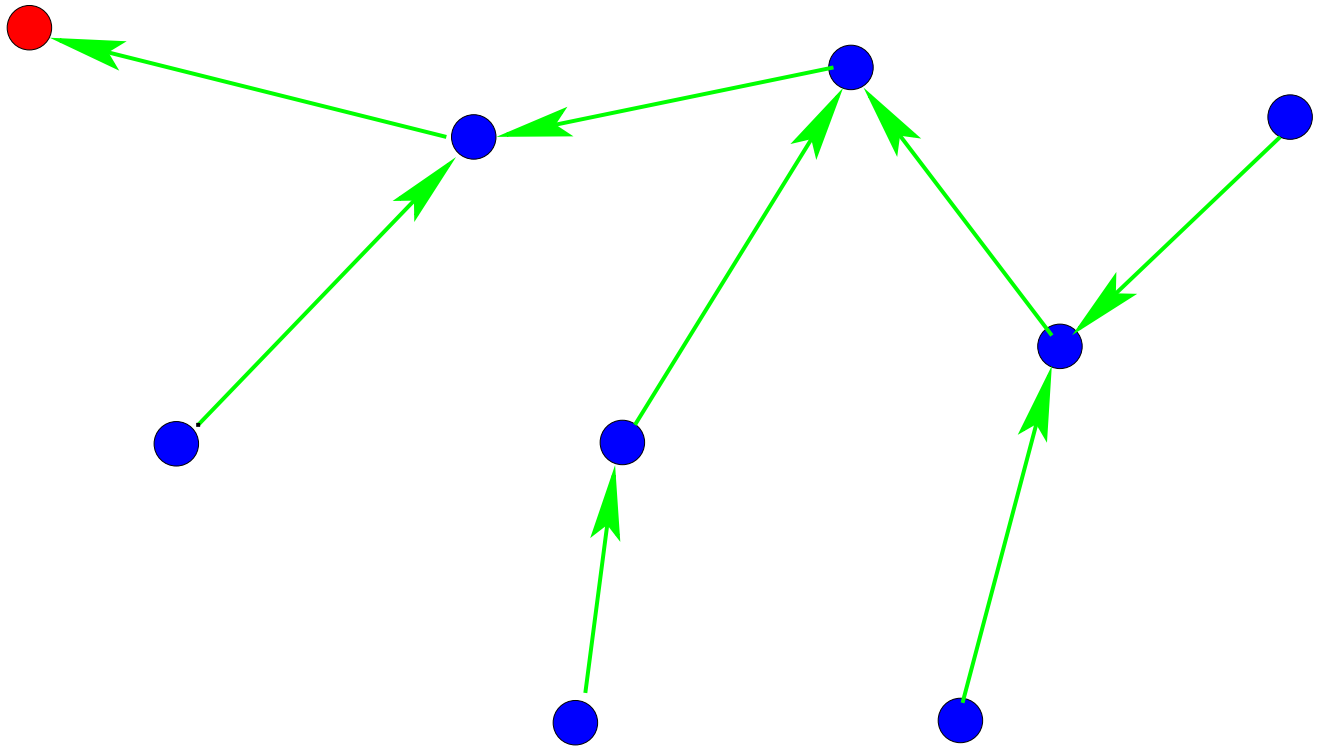
- Un réseau connexe et acyclique de nœuds.
- Nœuds reliés par des canaux bidirectionnels
- Election en un temps fini d'un leader.
- De manière répartie et non-déterministe.
- Un exemple d'animation du protocole.





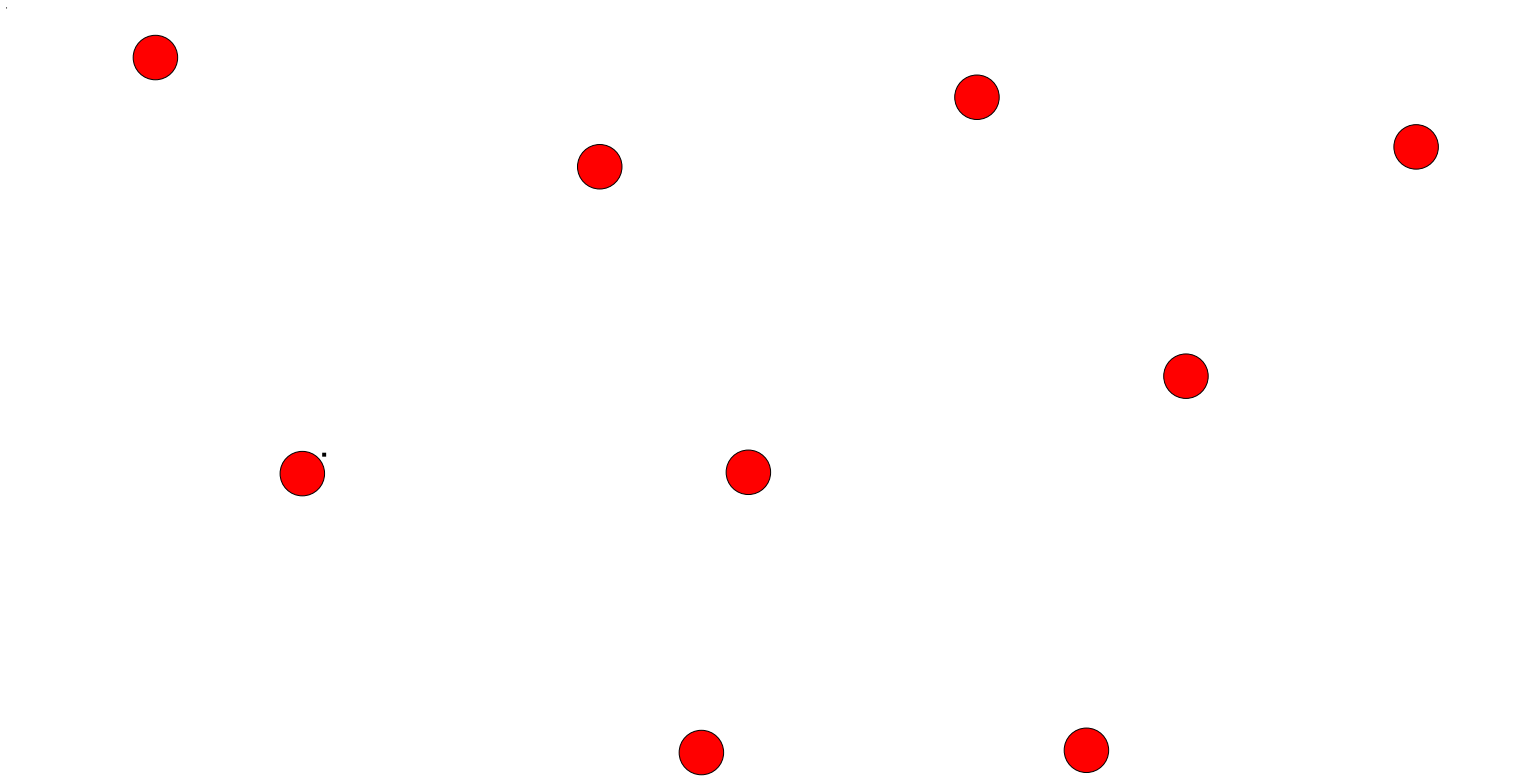




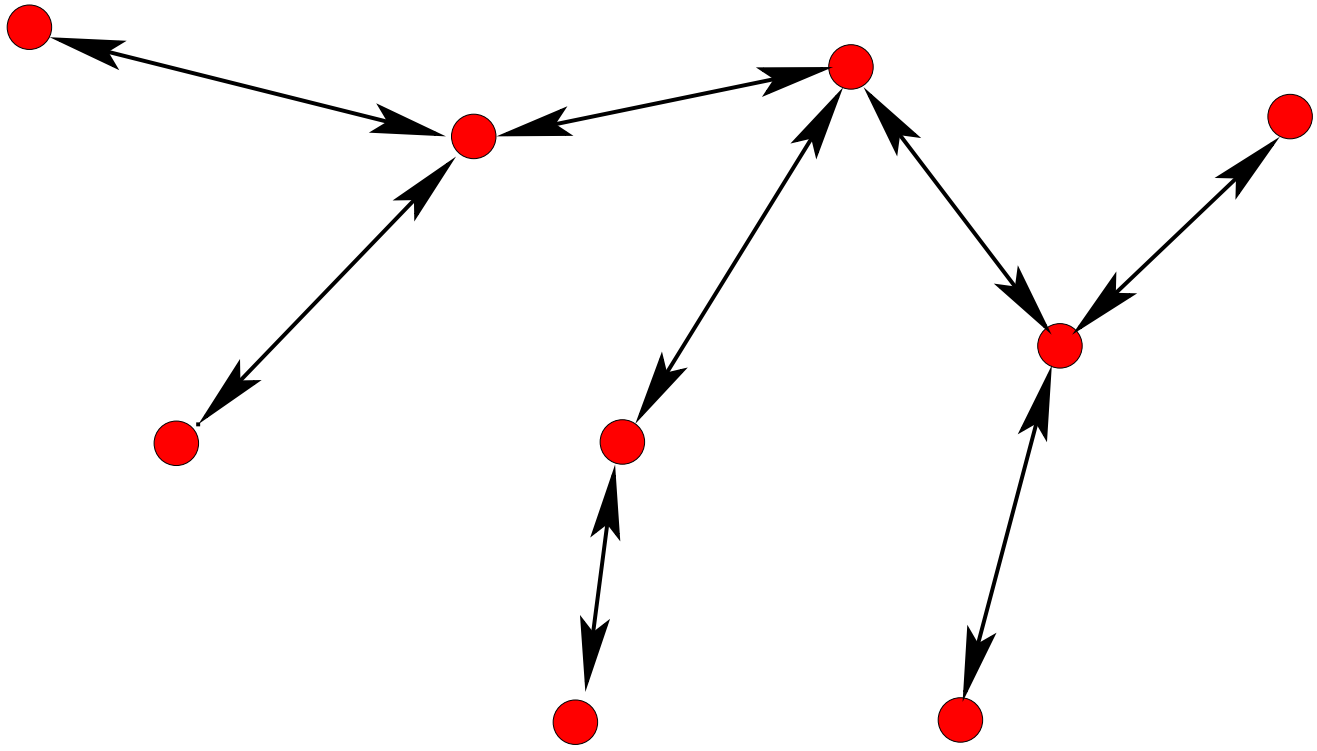


Le processus de développement

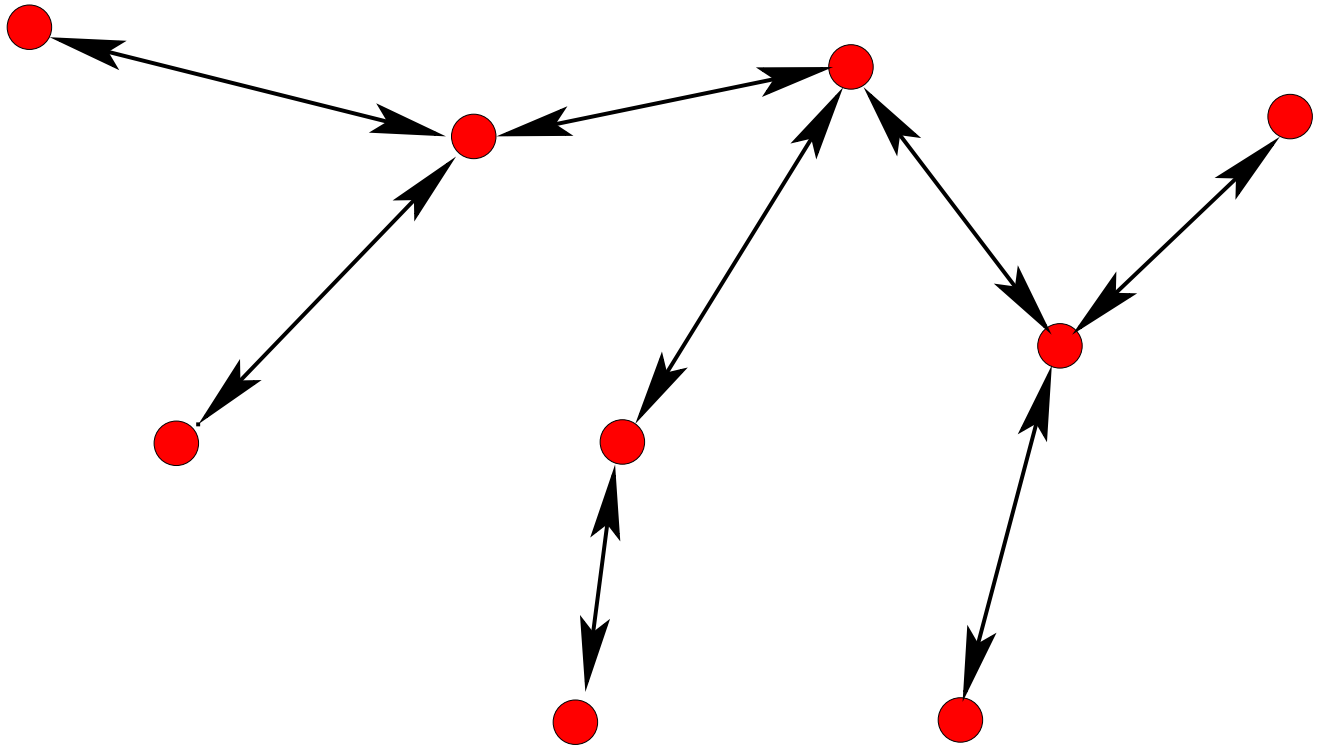
- Définition et propriétés du réseau dans un style formel
- Un modèle abstrait ¡¡one-shot¿¿ du protocole.
- Présentation d'une solution centralisée mais encore abstraite !
- Introduction du mécanisme de message passing entre les nœuds et des délais
- Modification de la structure de donnée pour répartir le protocole



ND un ensemble de noeuds (au moins 2 noeuds)



gr un graphe construit et défini sur ND



gr est un graphe symétrique et non réflexif

gr : un graphe construit sur ND $\quad$ $gr \subseteq ND \times ND$

gr : un graphe construit sur ND $\quad gr \subseteq ND \times ND$

gr est défini sur ND $\quad \mathsf{dom}\,(gr) = ND$

gr : un graphe construit sur ND $\quad gr \subseteq ND \times ND$

gr est défini sur ND $\quad \mathsf{dom}\,(gr) = ND$

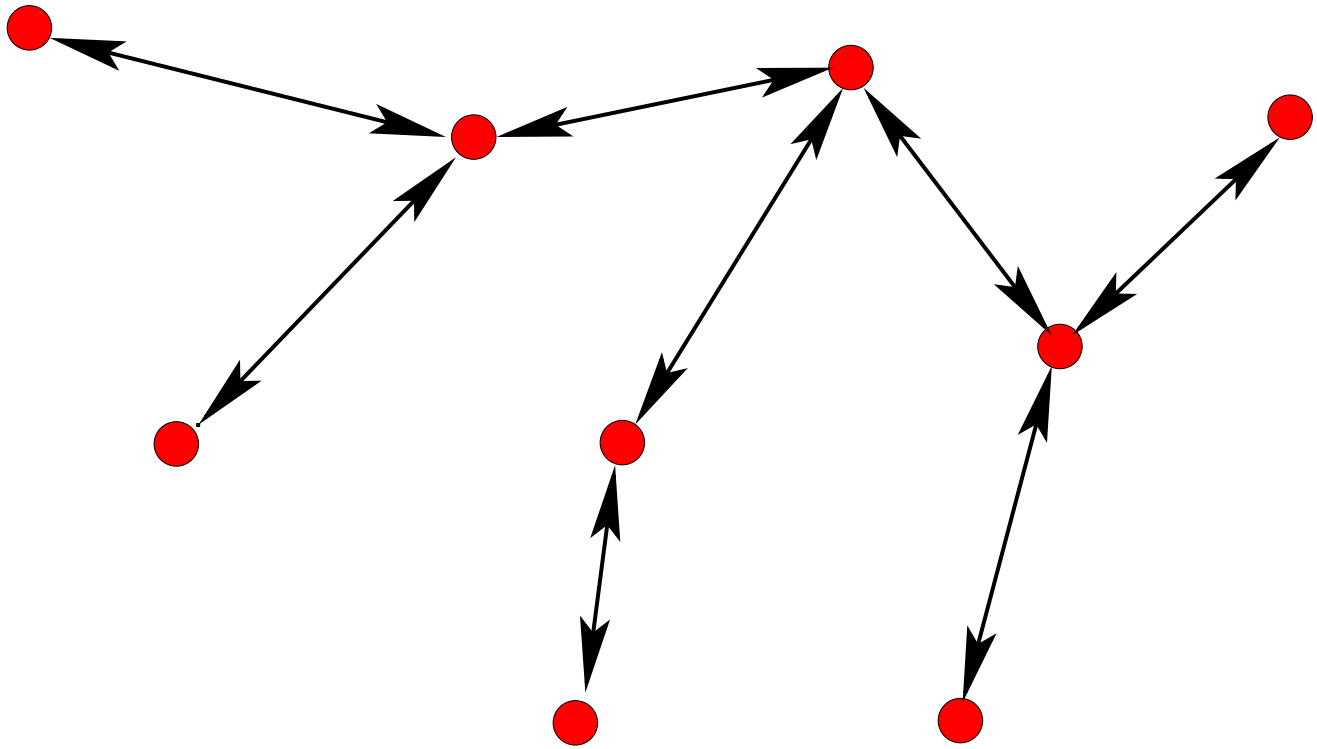
gr est symétrique $\quad gr = gr^{-1}$

gr : un graphe construit sur ND $gr \subseteq ND \times ND$

gr est défini sur ND $\mathsf{dom}\,(gr) = ND$

gr est symétrique $gr = gr^{-1}$

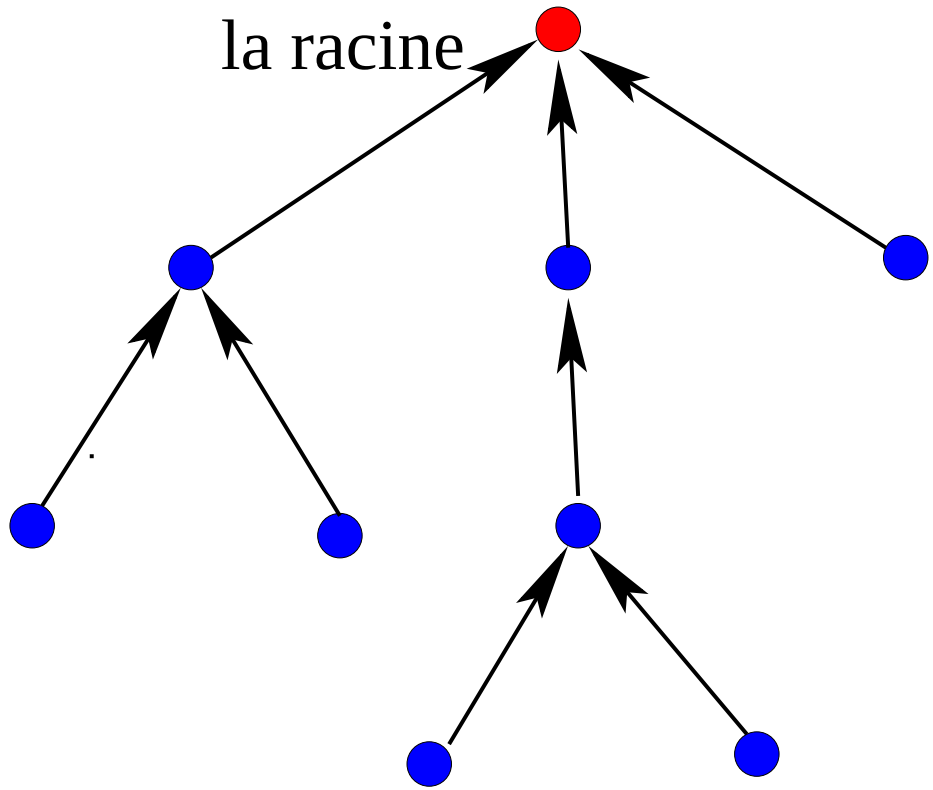
gr est non réflexif $\mathsf{id}\,(ND) \cap gr =$



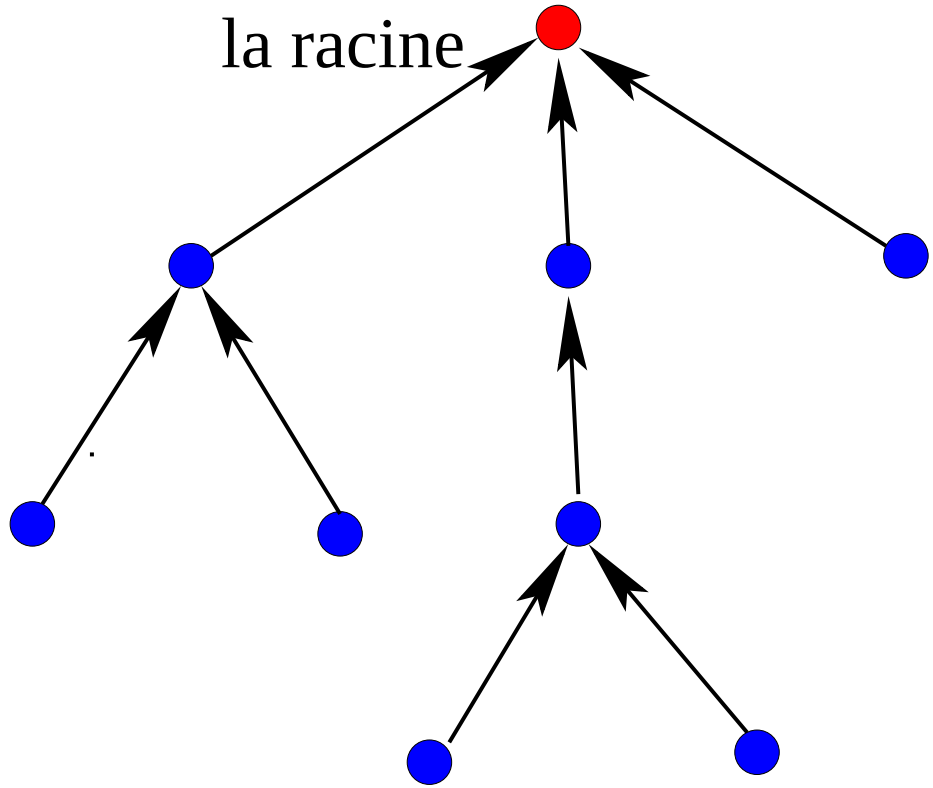
gr est connexe et acyclique

Un détour par les Arbres

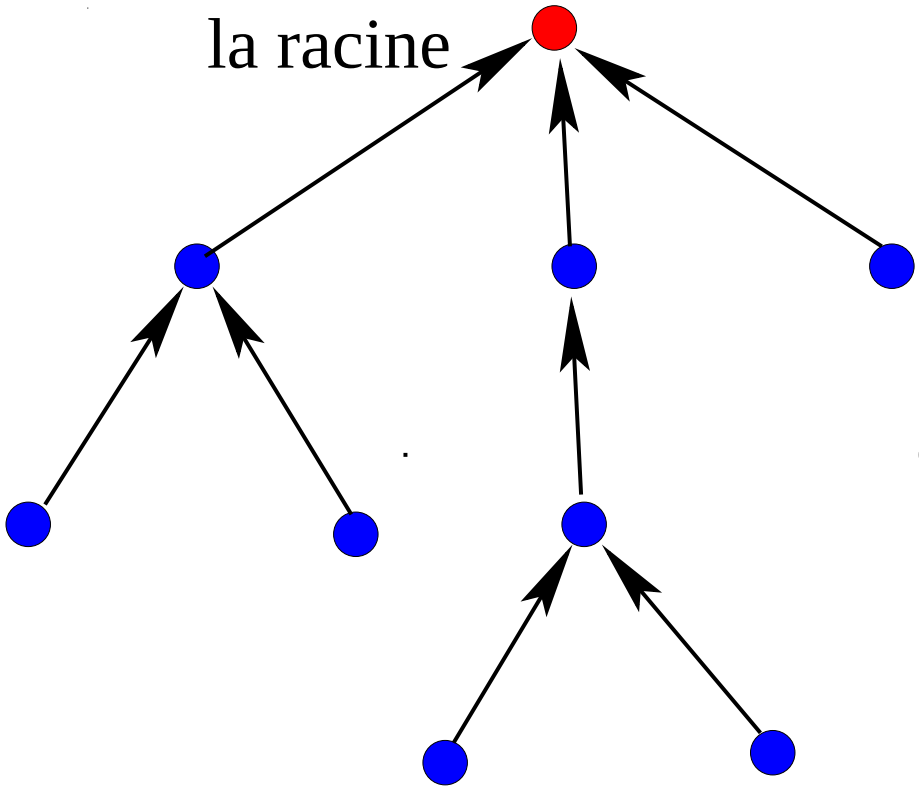
- Un arbre est un graphe particulier
- Un arbre a une racine
- Un arbre a une fonction parentale
- Un arbre est acyclique
- Un arbre est connexe à partir de la racine



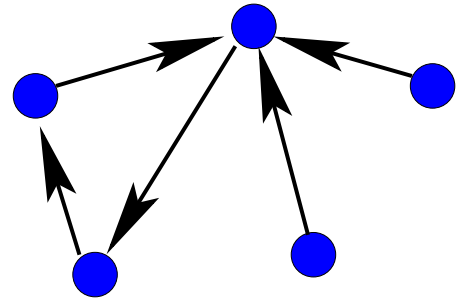
Un arbre t construit sur les noeuds



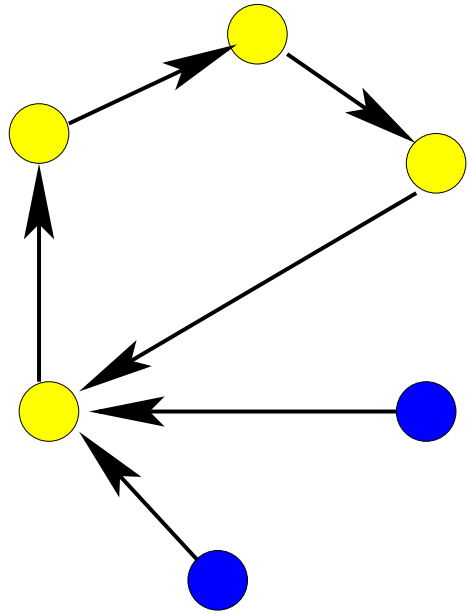
t est une fonction définie sur ND sauf pour la racine!



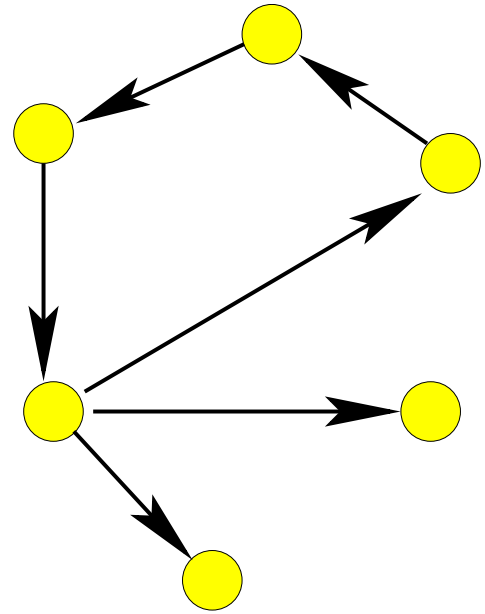
Eviter des cycles



Mauvais exemple!



Un cycle



Son image inverse

Les noeuds d'un cycle sont inclus
dans leur image inverse

Le prédicat tree (r, t)

Le prédicat tree (r, t)

r est un élément de ND $r \in ND$

Le prédicat tree (r, t)

r est un élément de ND $\quad r \in ND$

t est une fonction $\qquad\qquad t \in ND - \{r\} \rightarrow ND$

Le prédicat tree (r, t)

r est un élément de ND $\quad r \in ND$

t est une fonction $\quad t \in ND - \{r\} \rightarrow ND$

t est acyclique

$$\forall p \cdot \left(\begin{array}{l} p \subseteq ND \;\wedge \\ p \subseteq t^{-1}[p] \\ \Longrightarrow \\ p = \varnothing \end{array} \right)$$

t est acyclique : formulations équivalentes

$$\forall p \cdot \left(\begin{array}{l} p \subseteq ND \ \wedge \\ p \subseteq t^{-1}[p] \\ \Longrightarrow \\ p = \varnothing \end{array} \right) \quad \Leftrightarrow \quad \forall q \cdot \left(\begin{array}{l} q \subseteq ND \ \wedge \\ r \in q \ \wedge \\ t^{-1}[q] \subseteq q \\ \Longrightarrow \\ ND \subseteq q \end{array} \right)$$

On obtient une Règle d'Induction

$$\forall q \cdot \left(\begin{array}{l} q \subseteq ND \ \wedge \\ r \in q \ \wedge \\ \forall x \cdot (x \in ND - \{r\} \ \wedge \ t(x) \in q \implies x \in q) \\ \implies \\ ND \subseteq q \end{array} \right)$$

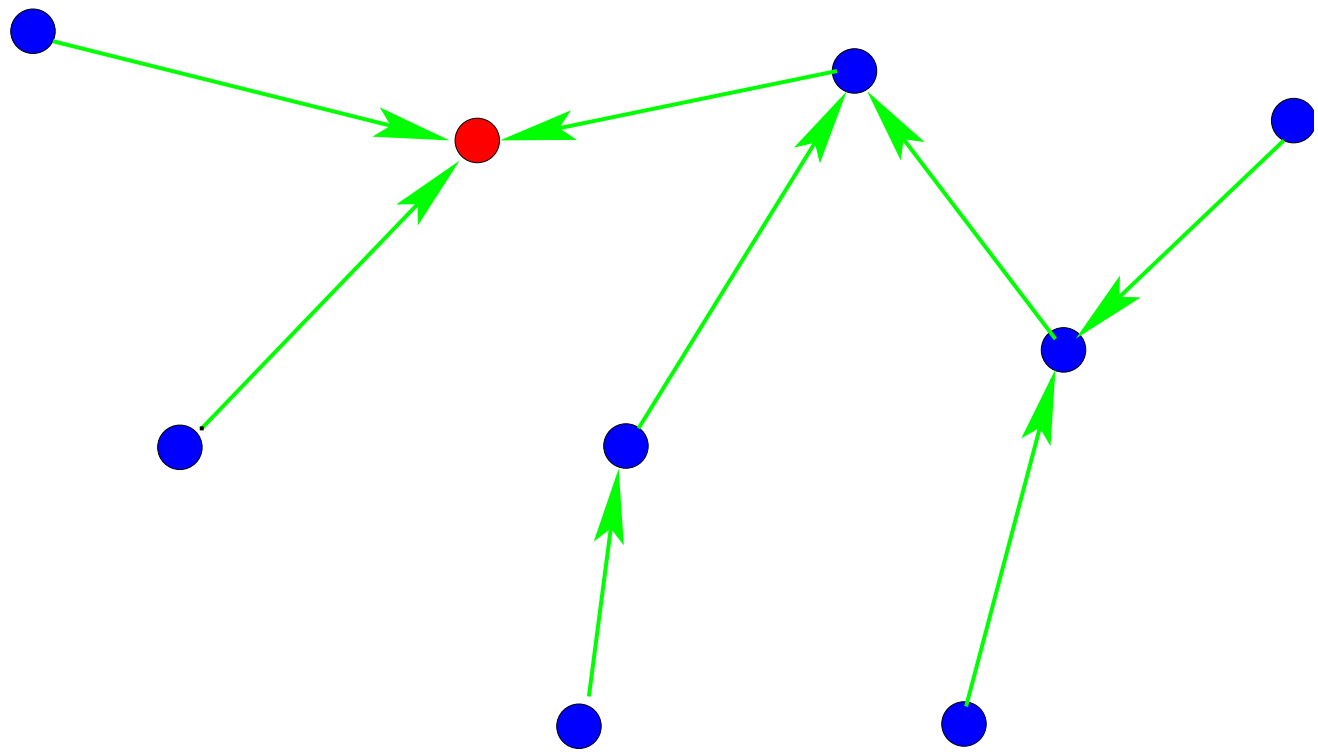
Le prédicat tree (r, t)

r est un élément de ND $\quad r \in ND$

t est une fonction $\qquad t \in ND \mathrel{-\!\!\!\rightarrow} \{r\} \rightarrow ND$

t est acyclique

$$\forall q \cdot \left(\begin{array}{l} q \subseteq ND \ \wedge \\ r \in q \ \wedge \\ t^{-1}[q] \subseteq q \\ \Longrightarrow \\ ND \subseteq q \end{array} \right)$$



Un arbre de recouvrement t de gr

Le prédicat $\text{spanning}\,(r, t, gr)$

r, t est un arbre	$\text{tree}\,(r, t)$
t est inclus dans gr	$t \subseteq gr$

Le graphe gr est connexe et acyclique (1)

- Définir une relation fn liant un noeud aux possibles spanning trees de gr ayant ce noeud comme racine :

$$fn \subseteq ND \times (ND \pfun ND)$$

$$\forall (r,t) \cdot \left(\begin{array}{l} r \in ND \ \land \\ t \in ND \pfun ND \\ \Longrightarrow \\ (r,t) \in fn \ \Leftrightarrow \ \mathsf{spanning}\,(r,t,gr) \end{array} \right)$$

Le graphe gr est connexe et acyclique (2)

Totalité de la relation $fn \implies$ Connectivité of gr

Fonctionnalité d'une relation $fn \implies$ Acyclicité de gr

Point sur les constantes gr and fn

$$gr \subseteq ND \times ND$$
$$\mathsf{dom}\,(gr) = ND$$
$$gr = gr^{-1}$$
$$\mathsf{id}\,(ND) \cap gr = \varnothing$$

$$fn \in ND \rightarrow (ND \nrightarrow ND)$$
$$\forall (r,t) \cdot \left(\begin{array}{l} r \in ND \ \wedge \\ t \in ND \nrightarrow ND \\ \Longrightarrow \\ t = fn(r) \ \Leftrightarrow \ \mathsf{spanning}\,(r,t,gr) \end{array} \right)$$

Tree

- Variables rt et ts

$$rt \in ND$$
$$ts \in ND \leftrightarrow ND$$

EVENT elect $\widehat{=}$
BEGIN
rt, ts : spanning (rt, ts, gr)
END

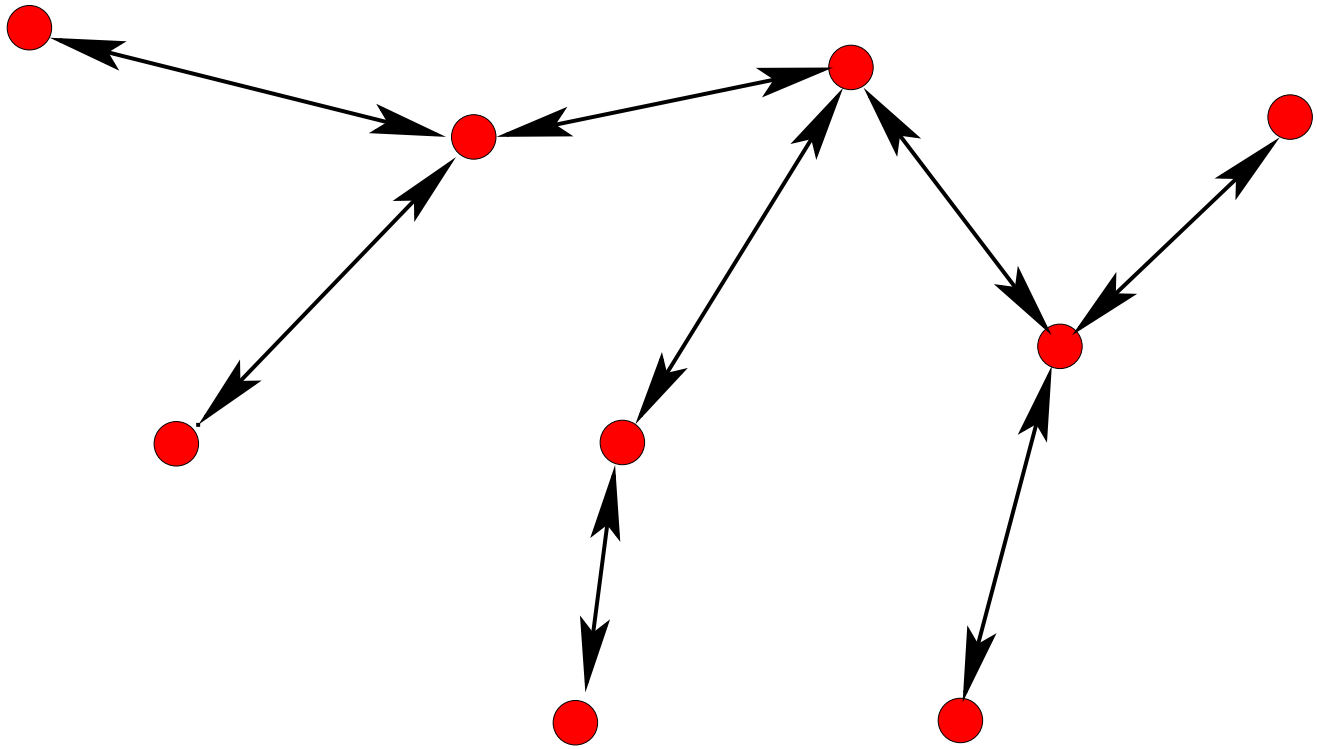
Premier raffinement (1)

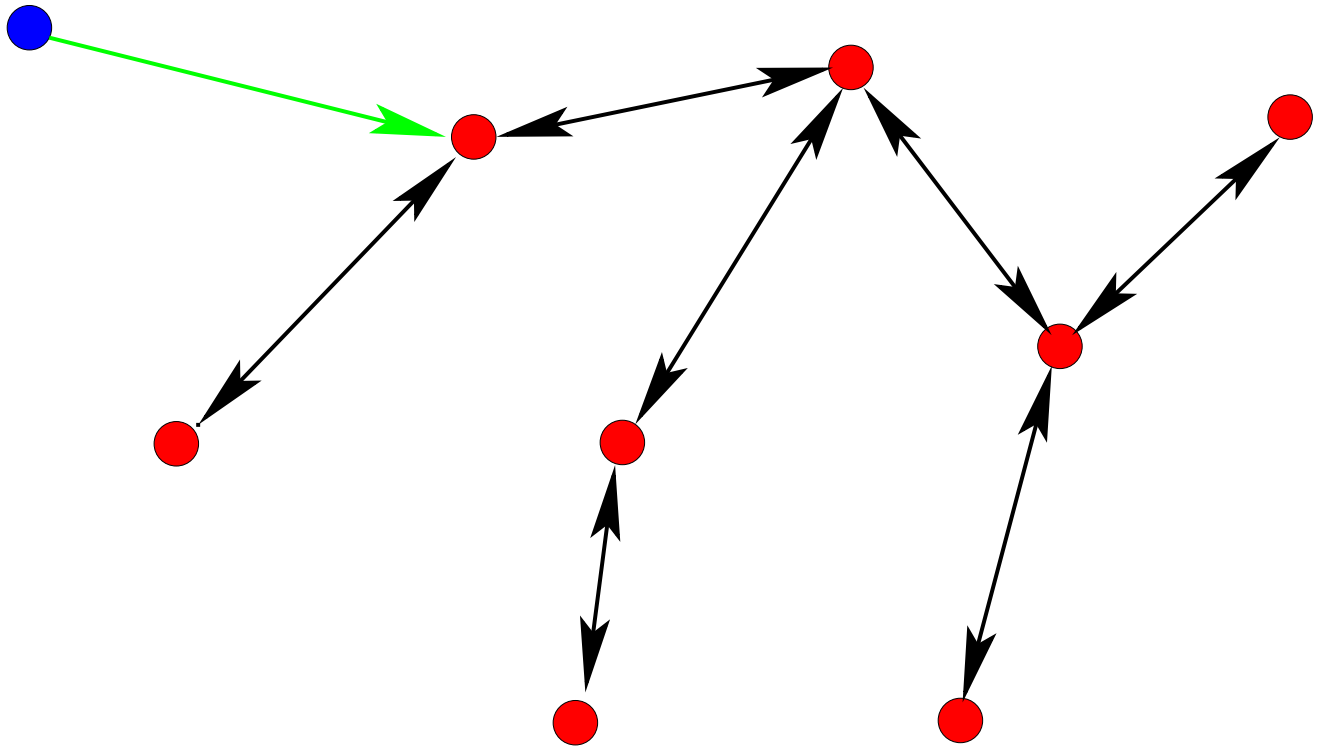
- Introduire une nouvelle variable, tr, correspondant à l'
  "arbre" en construction

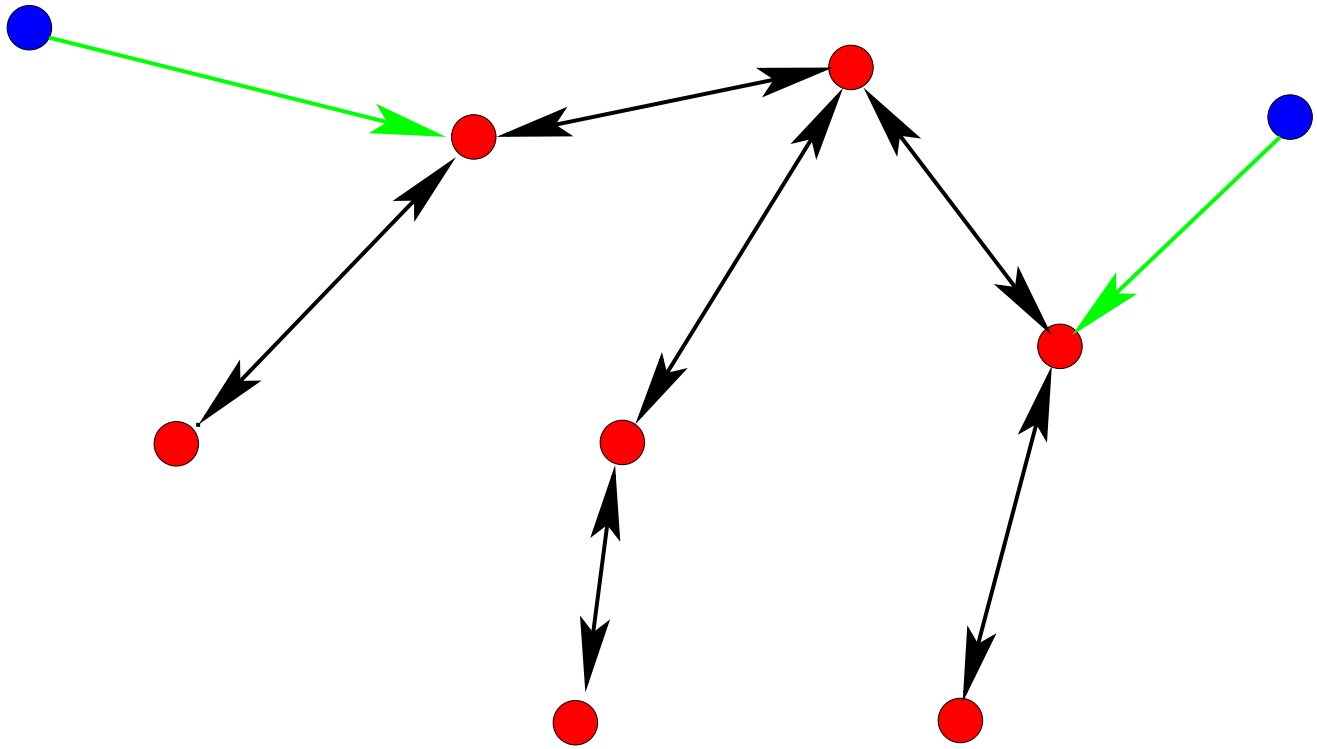
- Introduire un nouvel événement : progression

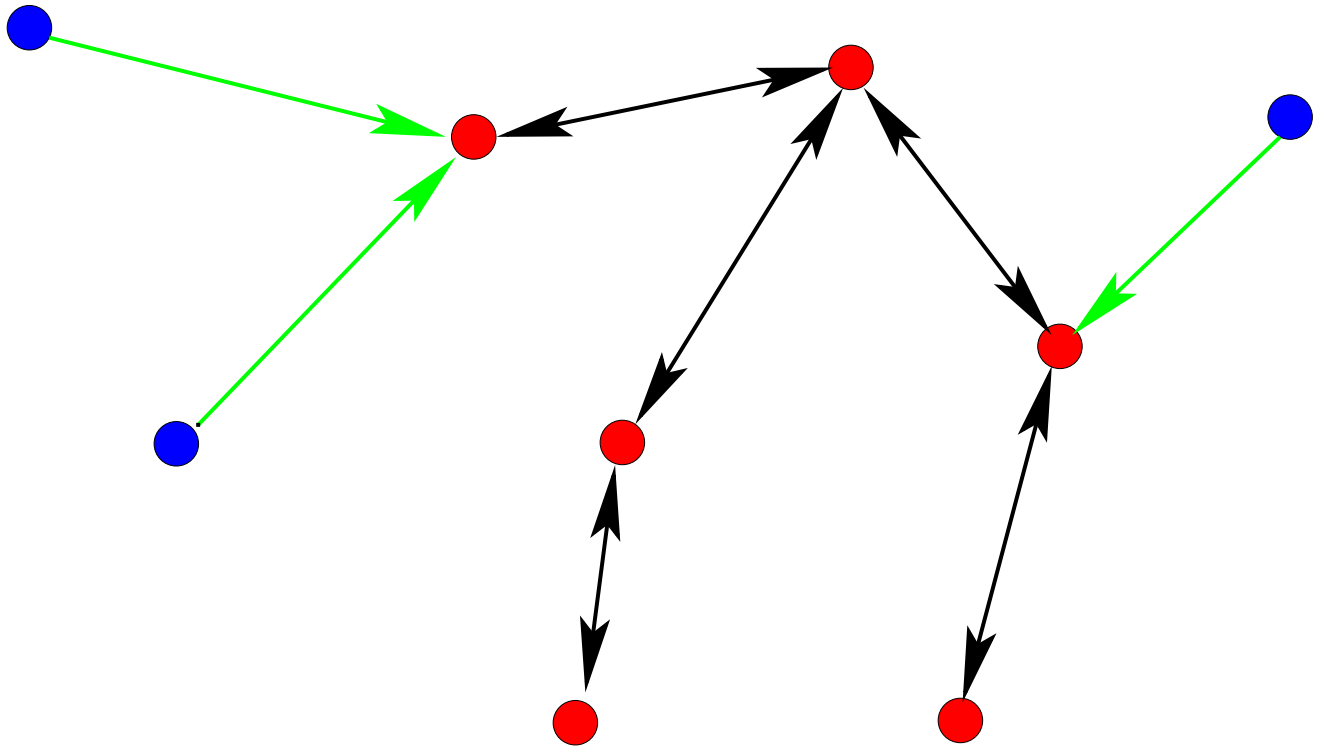
- Définir l'invariant

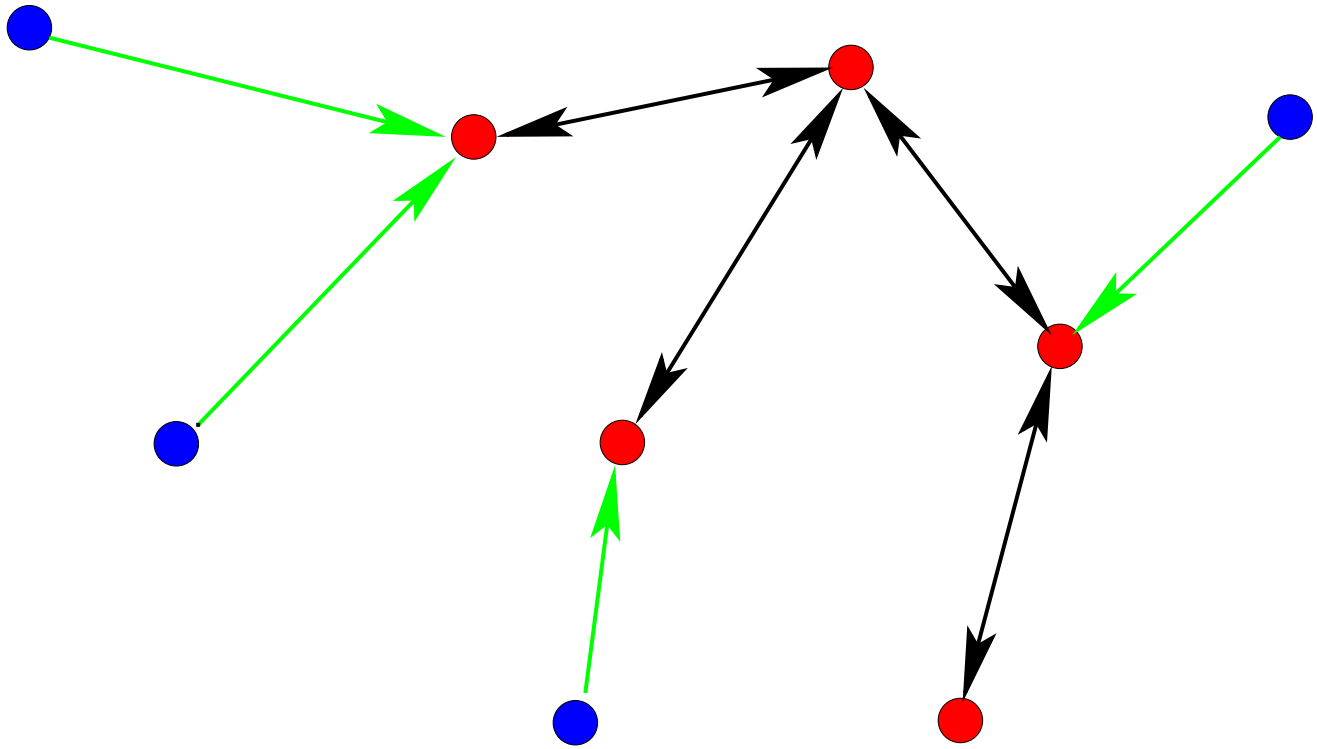
- Retour à l'animation : Observez la construction de l'arbre

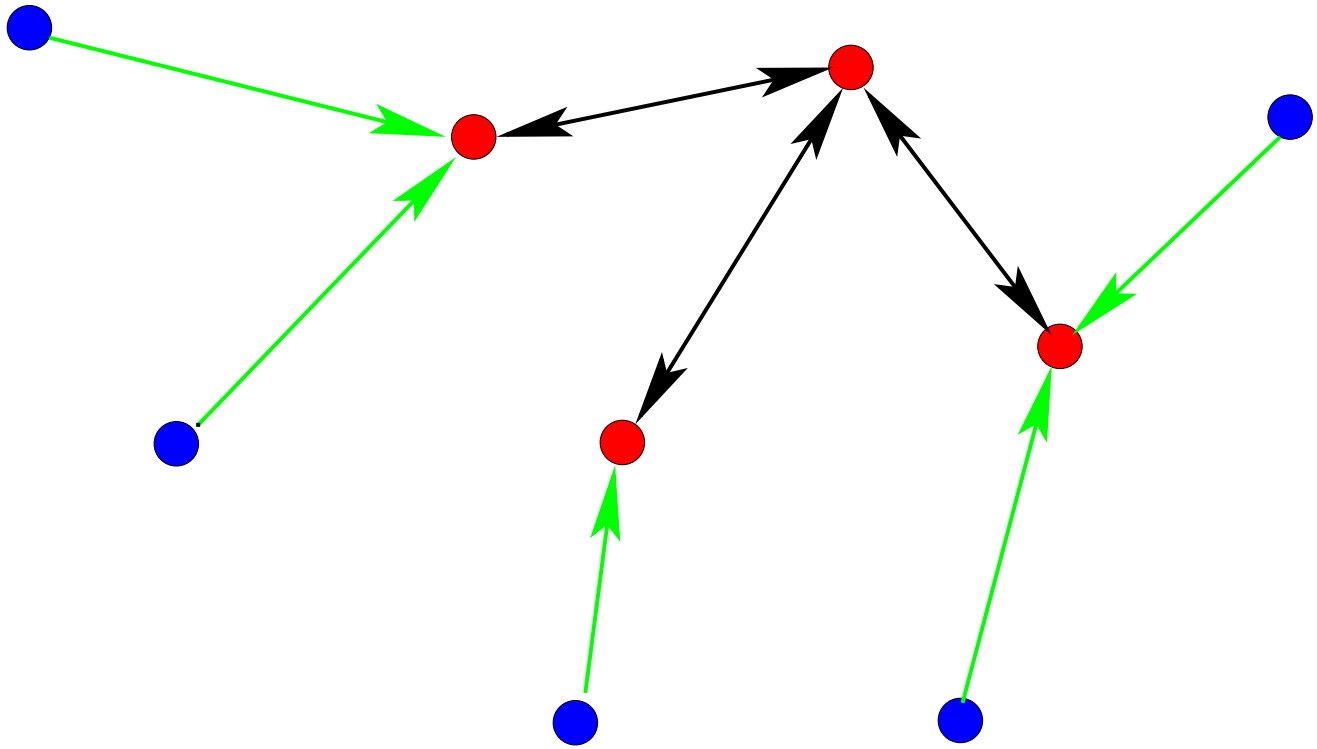


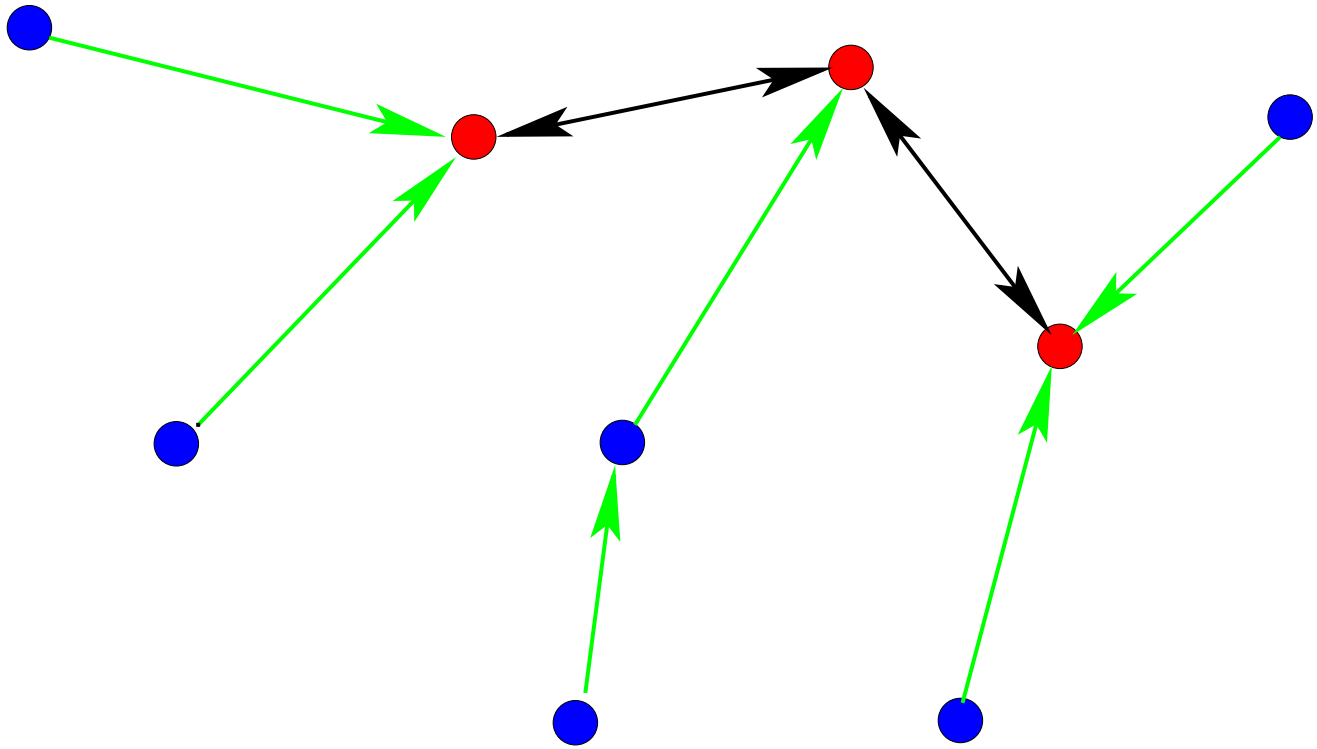


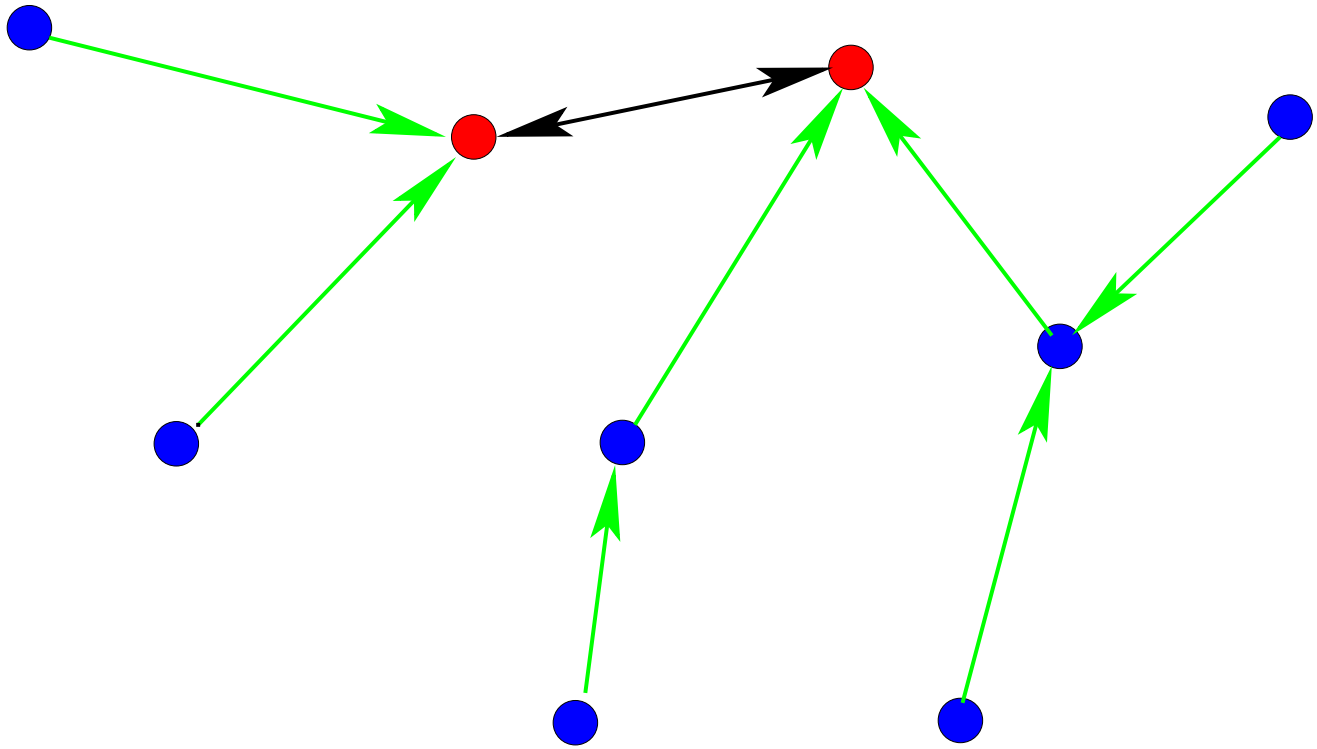


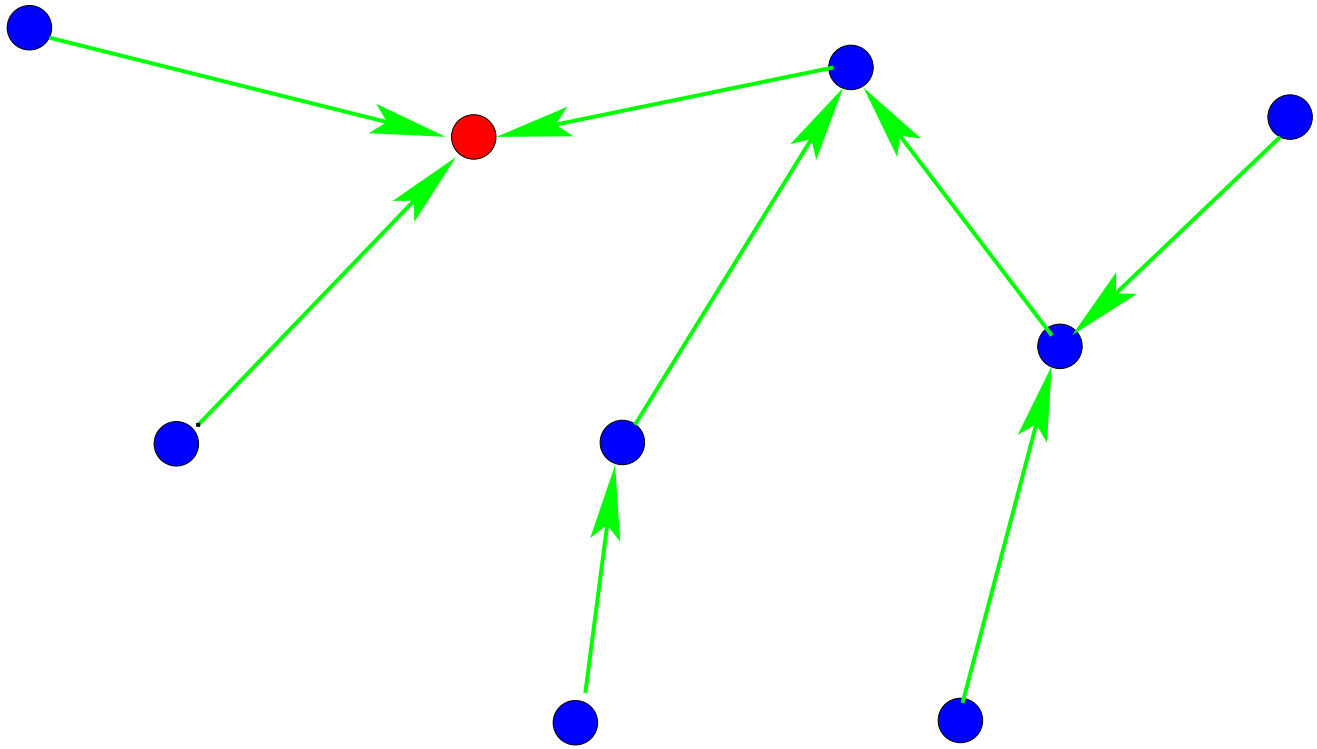












- Les flèches vertes correspondent à la fonction tr

- Les nœuds bleus sont le domaine de tr

- La fonction tr est une forêt sur les nœuds

- Les nœuds rouges sont les racines de ce arbres

Le prédicat invariant (tr)

$$tr \in ND \nrightarrow ND$$

Le prédicat invariant (tr)

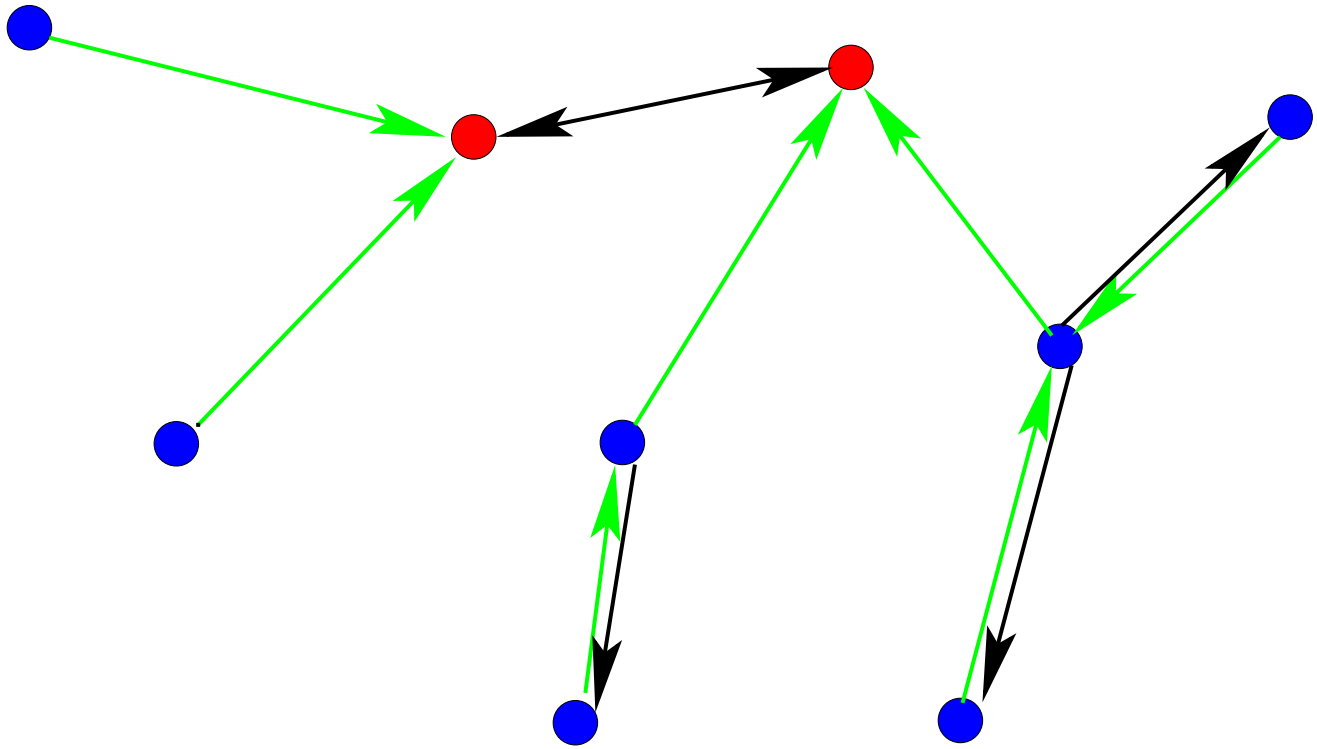
$$tr \in ND \nrightarrow ND$$

$$\forall p \cdot \left(\begin{array}{l} p \subseteq ND \quad \wedge \\ ND - \mathsf{dom}\,(tr) \subseteq p \quad \wedge \\ tr^{-1}\,[p] \subseteq p \\ \Longrightarrow \\ ND \subseteq p \end{array} \right)$$

Le prédicat invariant (tr)

$$tr \in ND \nrightarrow ND$$

$$\text{dom}\,(tr) \lhd (tr \cup tr^{-1}) = \text{dom}\,(tr) \lhd gr$$

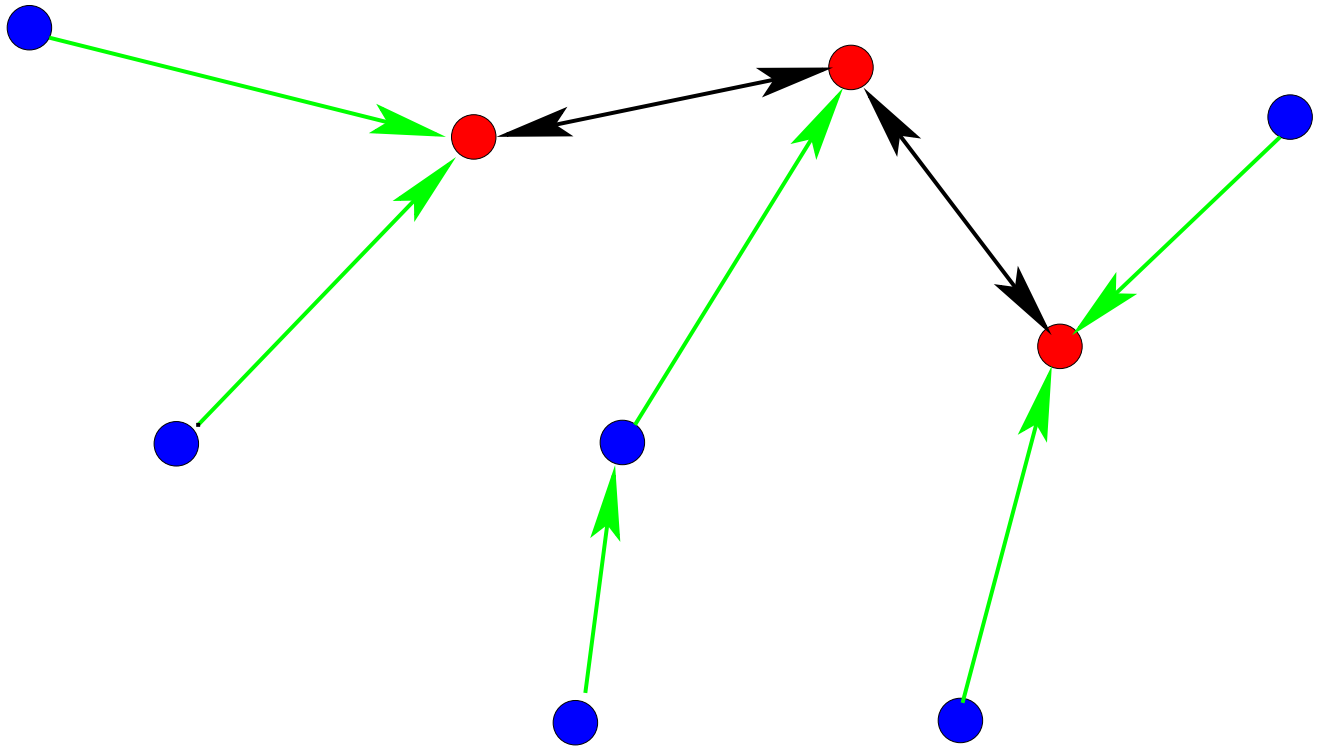


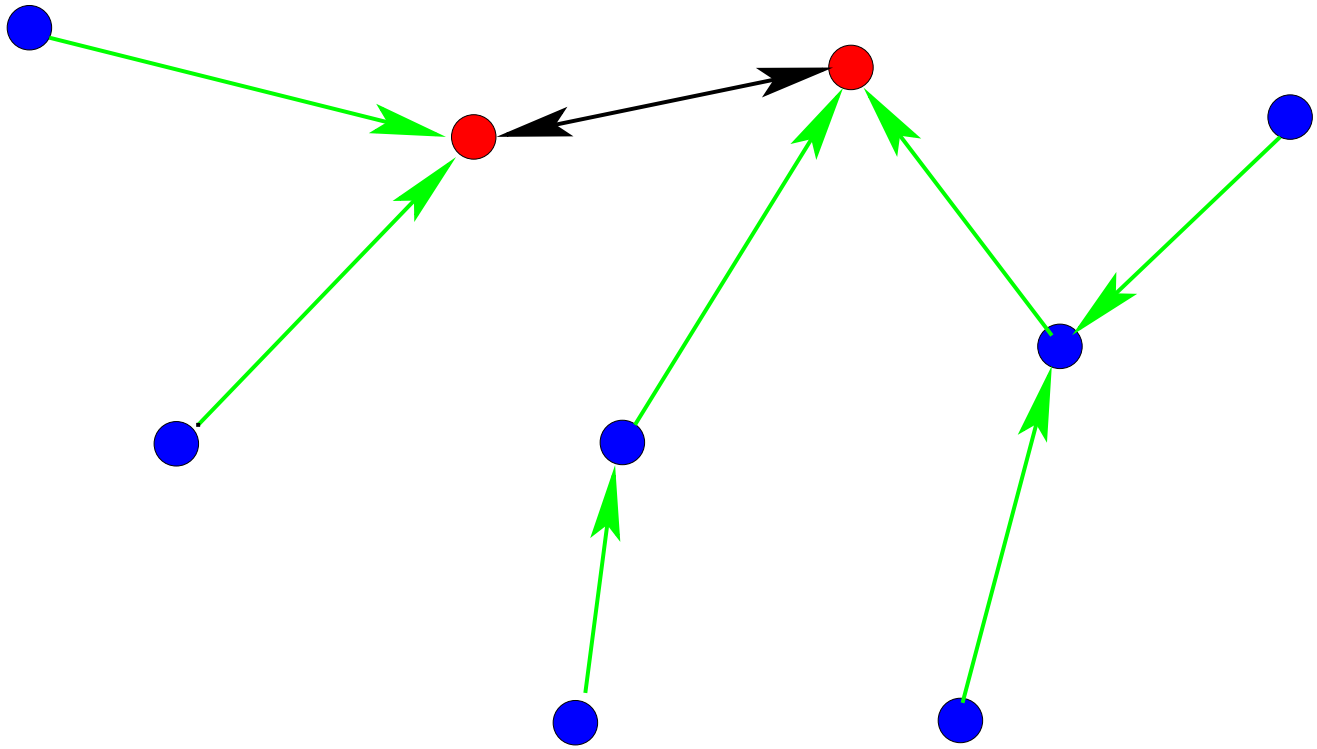
Premier raffinement (2)

- Introduire le nouvel événement "progress"

- Raffiner l'événement abstrait "elect"

- Retour à l'animation : Observez la "garde" de progress



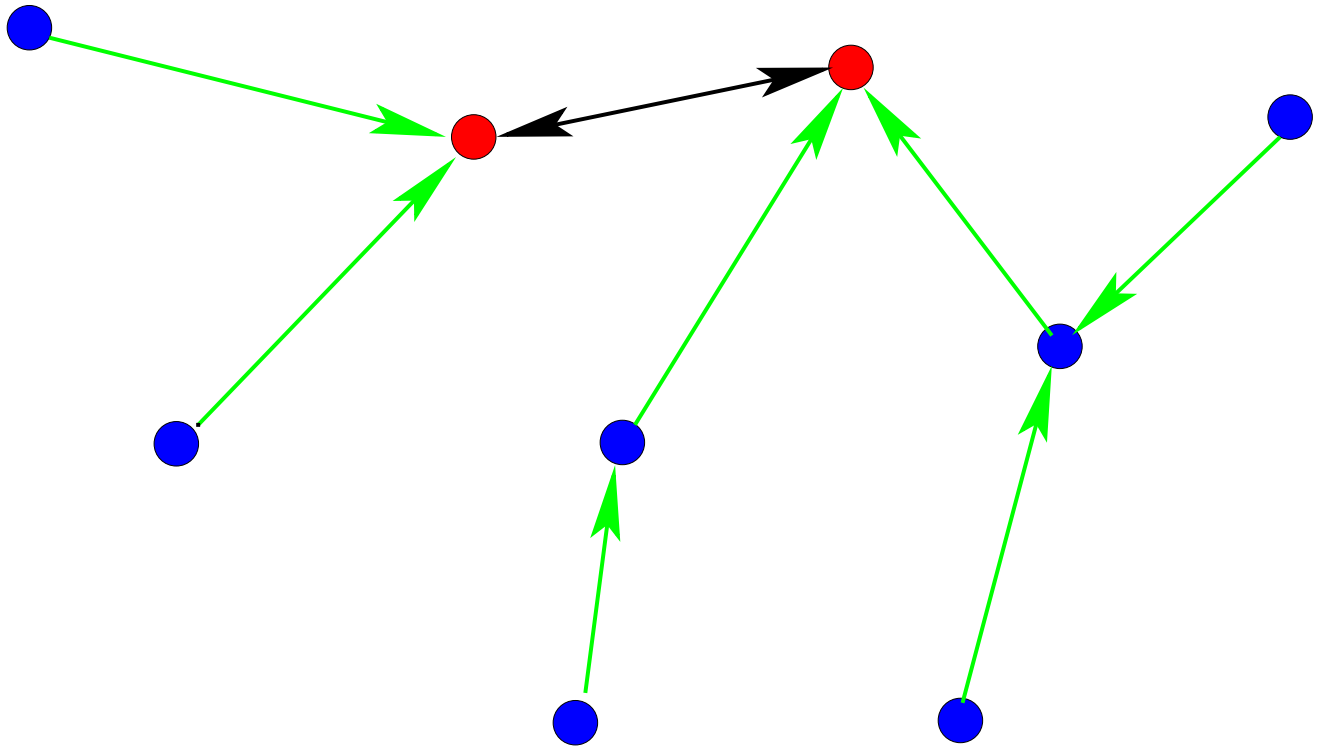


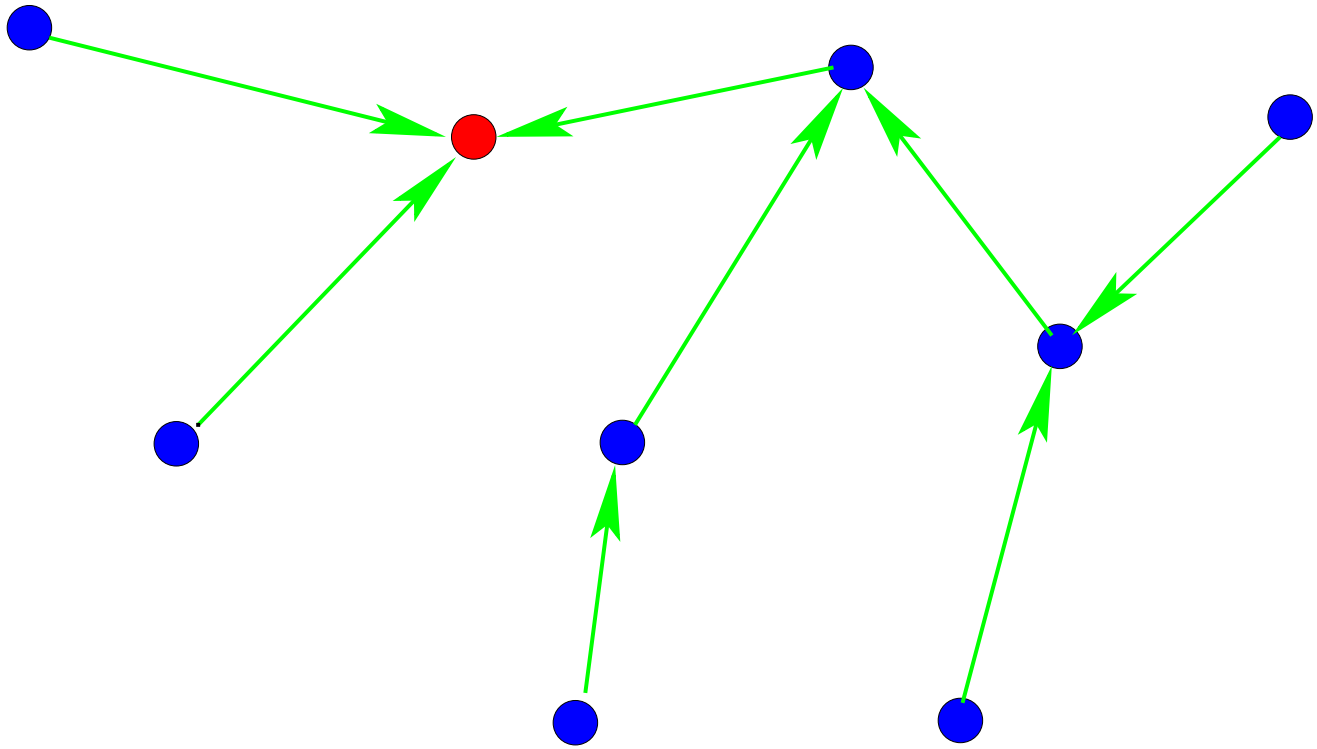
Quand un nœud rouge x est connecté à un autre nœud rouge y alors l'événement "progress" peut apparaître

$$
\begin{array}{l}
\text{EVENT progress} \ \widehat{=} \\
\quad \textbf{ANY } x, y \textbf{ WHERE} \\
\qquad x, y \in gr \ \wedge \\
\qquad x \notin \text{dom}\,(tr) \ \wedge \\
\qquad y \notin \text{dom}\,(tr) \ \wedge \\
\qquad gr[\{x\}] = tr^{-1}[\{x\}] \cup \{y\} \\
\quad \textbf{THEN} \\
\qquad tr := tr \cup \{x \mapsto y\} \\
\quad \textbf{END}
\end{array}
$$

Il faut prouver :

$$
\begin{array}{l}
\mathsf{invariant}(tr) \quad \wedge \\
x, y \in gr \quad \wedge \\
x \notin tr \quad \wedge \\
y \notin tr \quad \wedge \\
gr[\{x\}] = tr^{-1}[\{x\}] \cup \{y\} \\
\Longrightarrow \\
\mathsf{invariant}(tr \cup \{x \mapsto y\})
\end{array}
$$





Quand un nœud rouge x est SEULEMENT connecté aux nœuds bleus alors l'événement "elect" peut apparaître :

$$\begin{array}{l}\text{EVENT elect} \;\widehat{=}\\ \quad \textbf{ANY } x \textbf{ WHERE}\\ \qquad x \in ND \;\wedge\\ \qquad gr[\{x\}] = tr^{-1}[\{x\}]\\ \quad \textbf{THEN}\\ \qquad rt, ts := x, tr\\ \quad \textbf{END}\end{array}$$

$$\text{EVENT elect} \;\widehat{=}$$
BEGIN
$$rt, ts : \text{spanning}\,(rt, ts, gr)$$
END

$$\text{EVENT elect} \;\widehat{=}$$
ANY x **WHERE**
$$x \in ND \;\wedge$$
$$gr[\{x\}] = tr^{-1}[\{x\}]$$
THEN
$$rt, ts := x, tr$$
END

Il faut prouver :

$$\begin{array}{l}
\mathsf{invariant}(tr) \quad \wedge \\
x \in ND \quad \wedge \\
gr[\{x\}] = tr^{-1}[\{x\}] \\
ts = tr \\
\Longrightarrow \\
\mathsf{spanning}(x, ts, gr)
\end{array}$$

Un lemme

$$\begin{array}{l} \mathsf{invariant}(tr) \quad \wedge \\ x \in ND \quad \wedge \\ gr[\{x\}] = tr^{-1}[\{x\}] \\ \Longrightarrow \\ tr = fn(x) \end{array}$$

Récapitulatif du premier raffinement

- 12 preuves

- Parmi lesquelles 5 interactives (une assez difficile !)

Second Raffinement

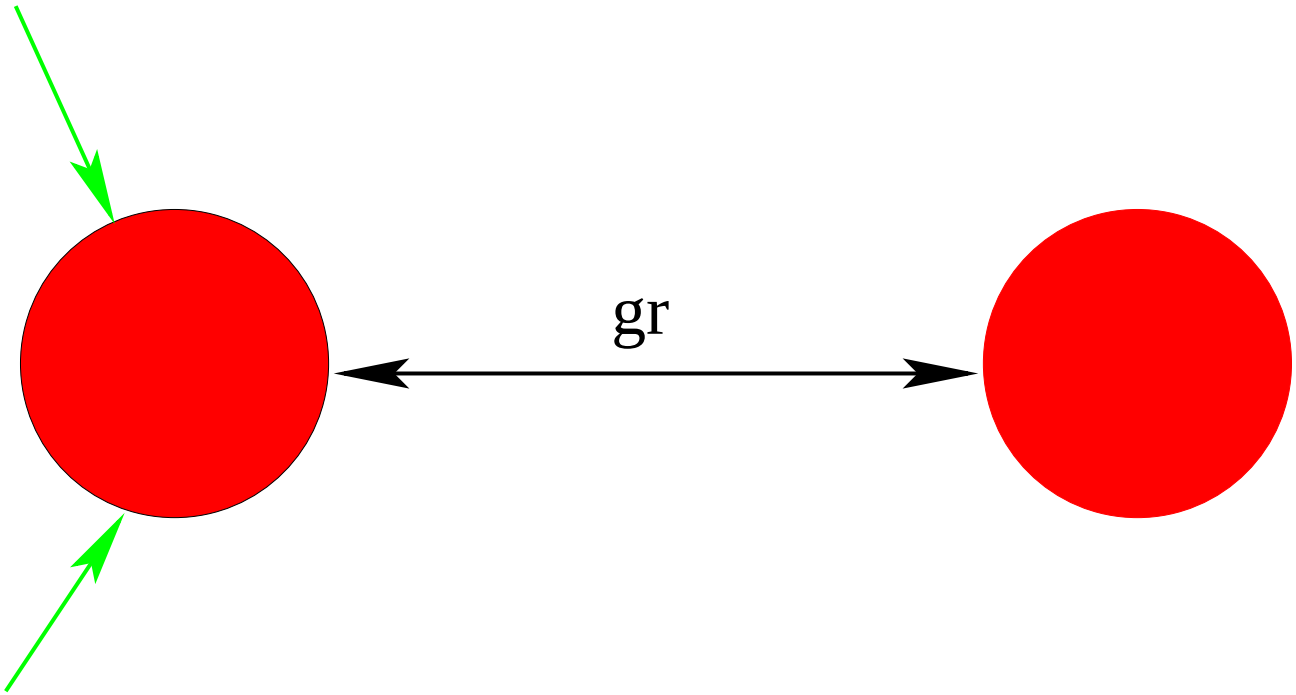
- Les nœuds communiquent avec leurs voisins

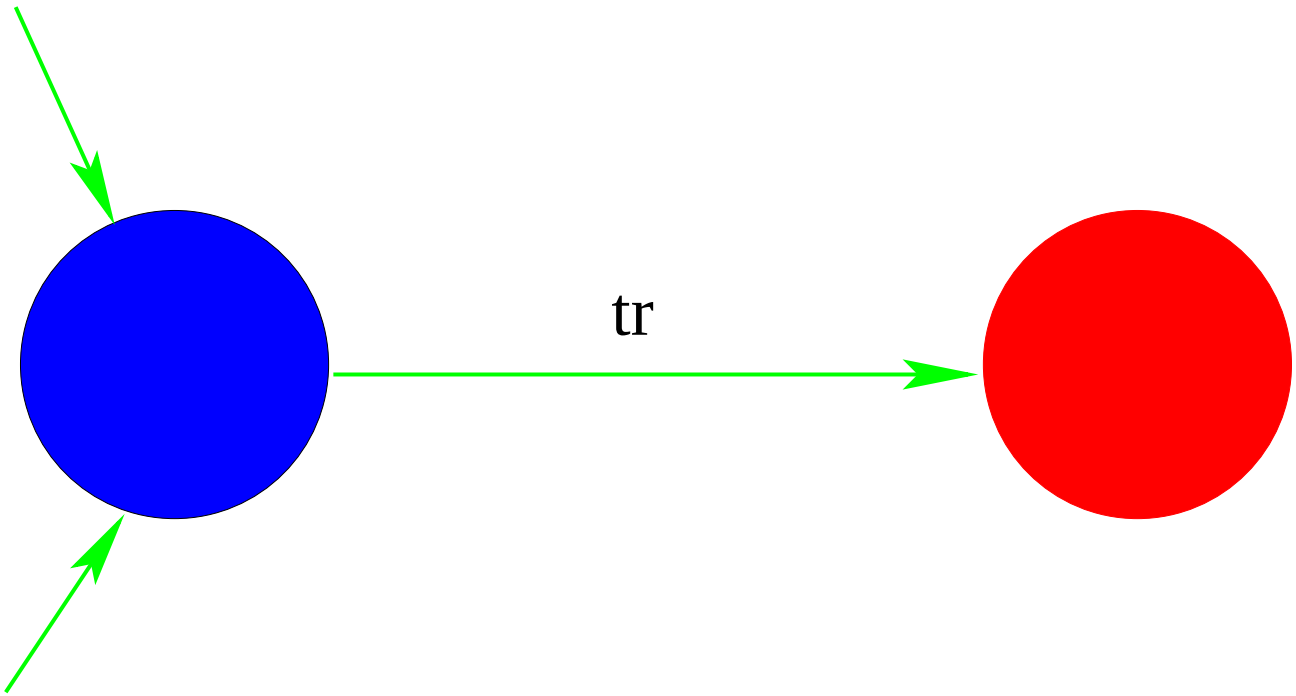
- On utilise des messages

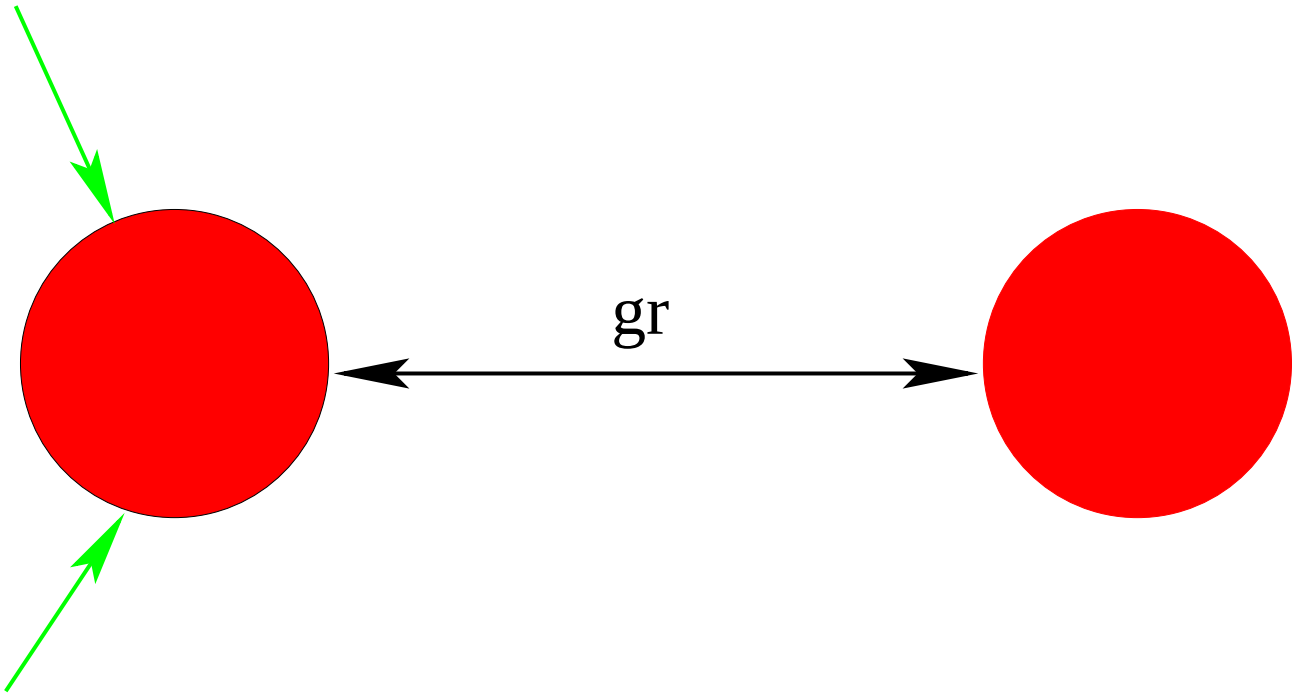
- Les messages ont des accusés de réception

- Les accusés de réception sont confirmés

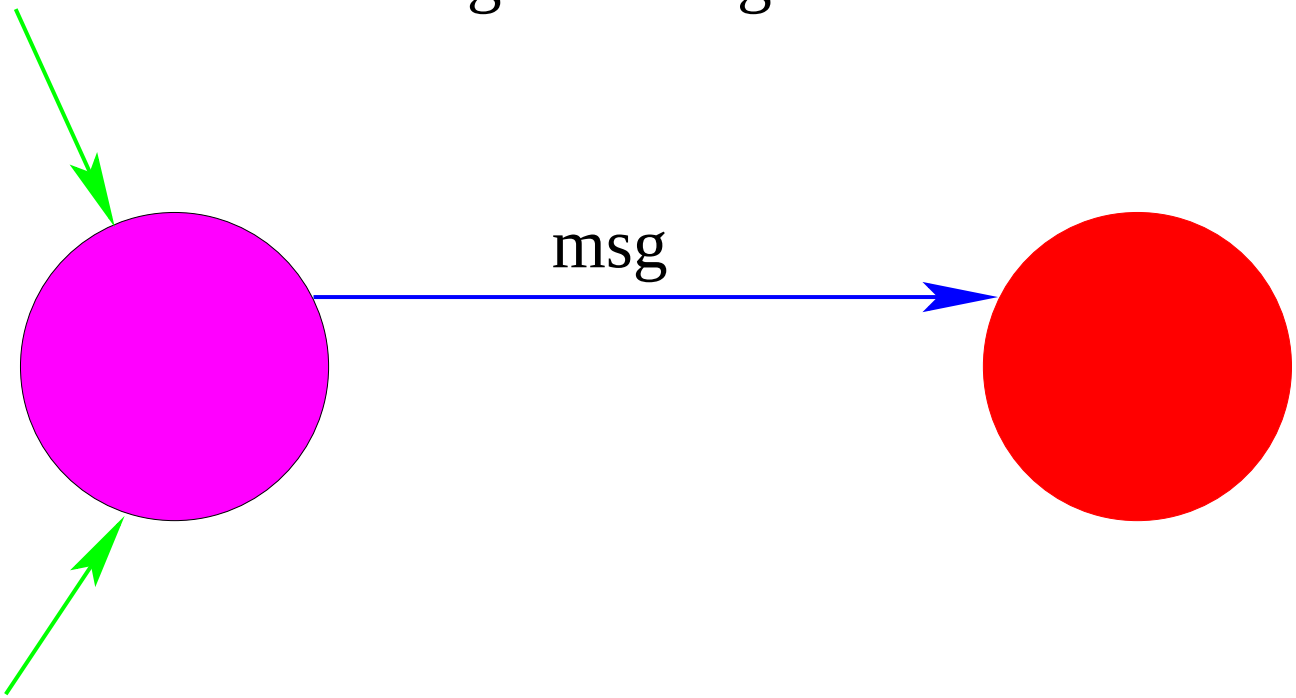
- Ensuite une animation locale


gr


tr

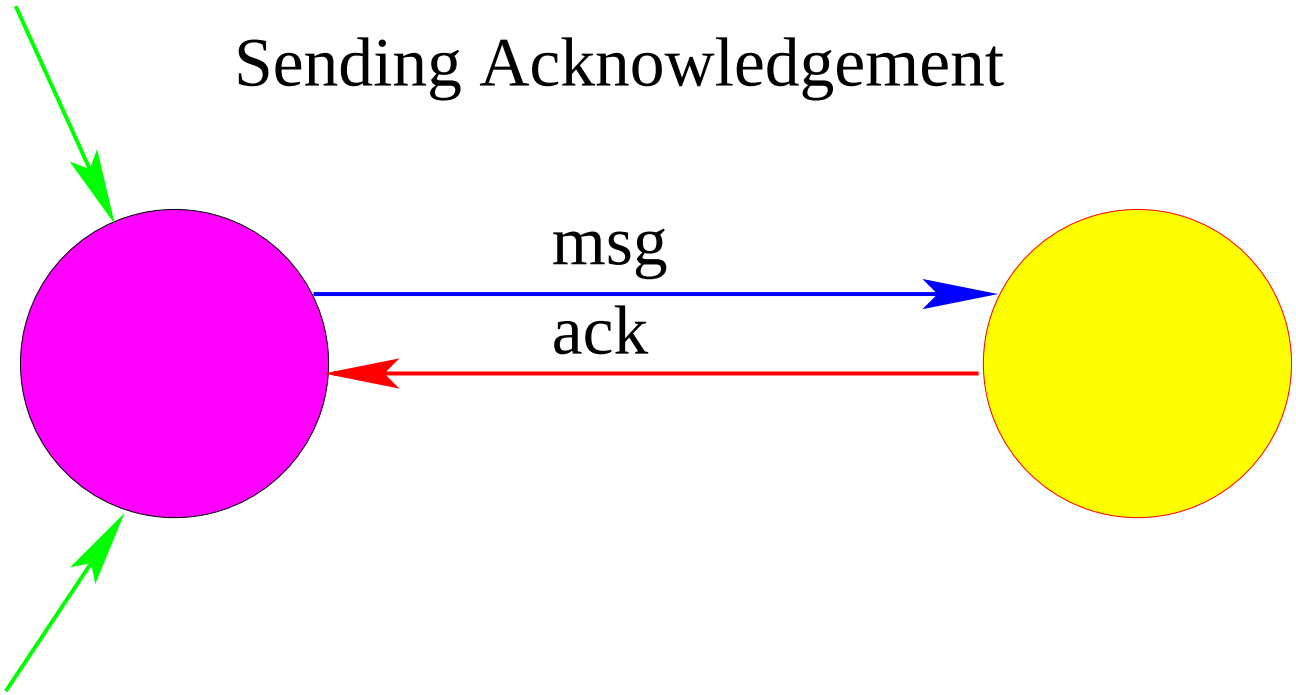

gr

Sending a message



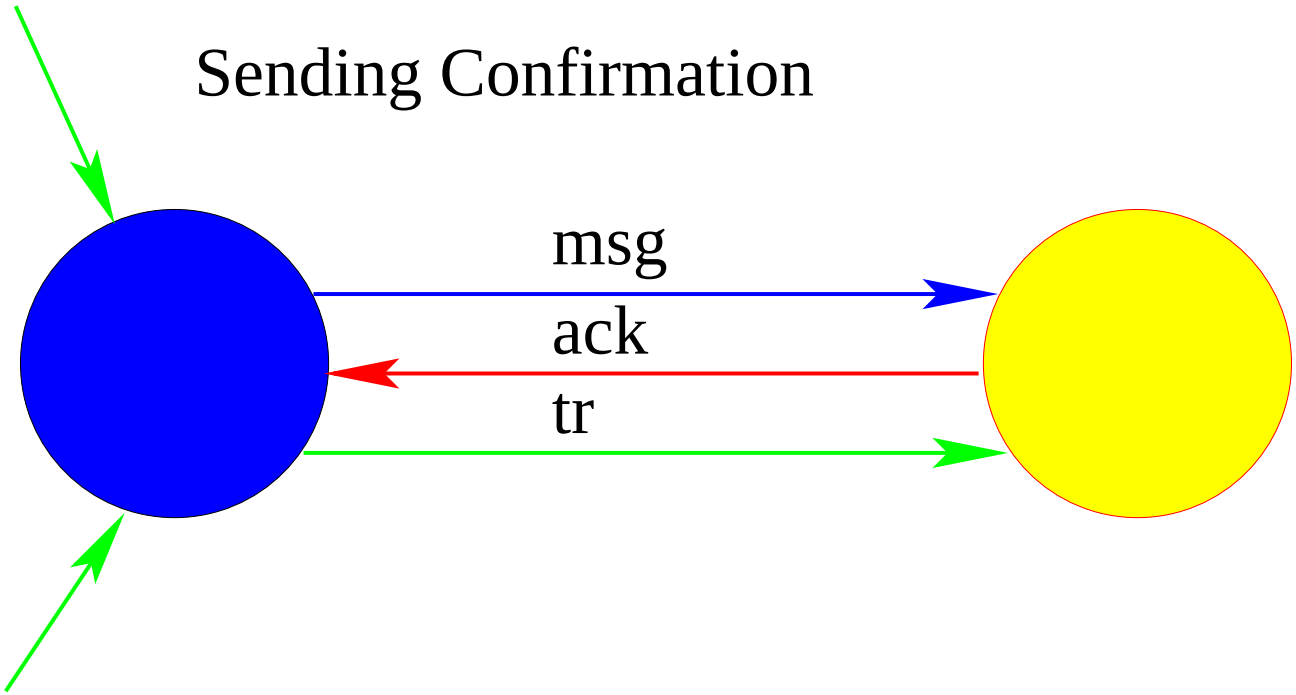
Receiving a message

Sending Acknowledgement

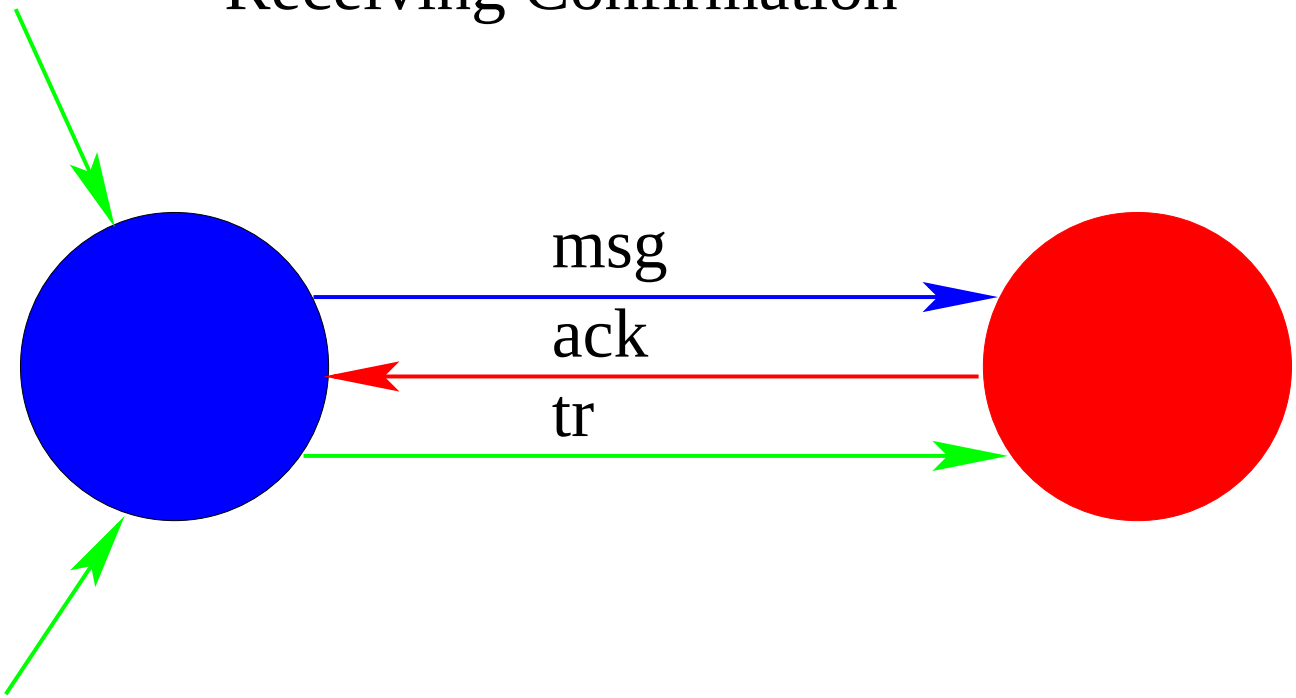


Receiving Acknowledgement

Sending Confirmation



Receiving Confirmation



Invariant (1)

$$msg \in ND \nrightarrow ND$$

$$ack \in ND \nrightarrow ND$$

$$tr \subseteq ack \subseteq msg \subseteq gr$$

Nœud x envoie un message au nœud y

$$
\begin{array}{l}
\text{EVENT send\_msg} \ \widehat{=} \\
\quad \textbf{ANY } x, y \textbf{ WHERE} \\
\qquad x, y \in gr \ \wedge \\
\qquad gr[\{x\}] = tr^{-1}[\{x\}] \cup \{y\} \ \wedge \\
\qquad y, x \notin ack \ \wedge \\
\qquad x \notin \text{dom}\,(msg) \\
\quad \textbf{THEN} \\
\qquad msg := msg \cup \{x \mapsto y\} \\
\quad \textbf{END}
\end{array}
$$

Nœud y envoie un acknowledgement au nœud x

$$
\begin{array}{l}
\text{EVENT send\_ack} \ \widehat{=} \\
\quad \textbf{ANY } x, y \textbf{ WHERE} \\
\qquad x, y \in msg{-}ack \ \wedge \\
\qquad y \notin \text{dom}\,(msg) \\
\quad \textbf{THEN} \\
\qquad ack := ack \cup \{x \mapsto y\} \\
\quad \textbf{END}
\end{array}
$$

Nœud x envoie une confirmation au nœud y

$$\begin{array}{l}\text{EVENT progress} \;\widehat{=}\\ \quad \textbf{ANY } x, y \textbf{ WHERE}\\ \qquad x, y \in ack \;\wedge\\ \qquad x \notin \text{dom}\,(tr)\\ \quad \textbf{THEN}\\ \qquad tr := tr \cup \{x \mapsto y\}\\ \quad \textbf{END}\end{array}$$

Invariant (2)

$$\forall\,(x,y)\cdot\left(\begin{array}{l} x,y\in msg{-}ack \\ \Longrightarrow \\ x,y\in gr \quad \wedge \\ x\notin \mathsf{dom}\,(tr)\ \wedge\ y\notin \mathsf{dom}\,(tr)\ \wedge \\ gr[\{x\}]\ =\ tr^{-1}[\{x\}]\cup\{y\} \end{array}\right)$$

$$\forall\,(x,y)\cdot\left(\begin{array}{l} x,y\in ack \quad \wedge \\ x\notin \mathsf{dom}\,(tr) \\ \Longrightarrow \\ x,y\in gr \quad \wedge \\ y\notin \mathsf{dom}\,(tr)\ \wedge \\ gr[\{x\}]\ =\ tr^{-1}[\{x\}]\cup\{y\} \end{array}\right)$$

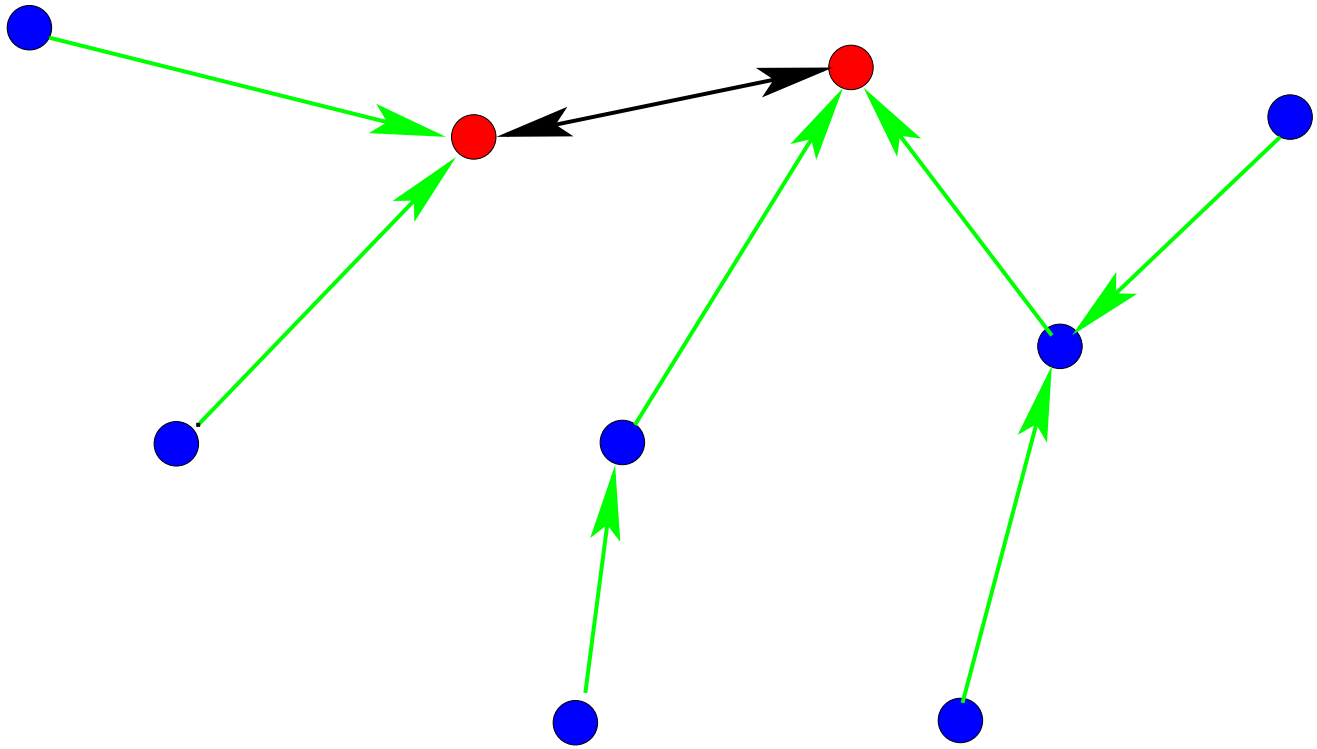
contention

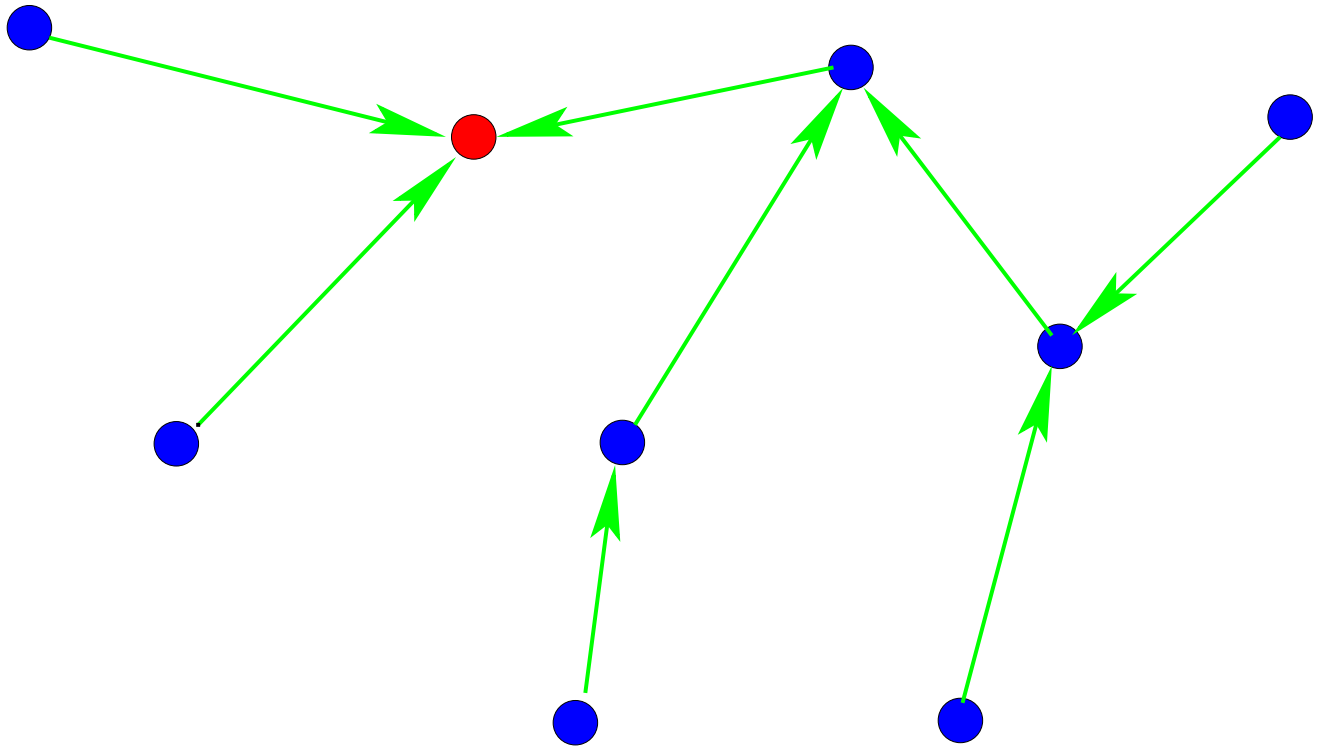
- Expliquer le problème

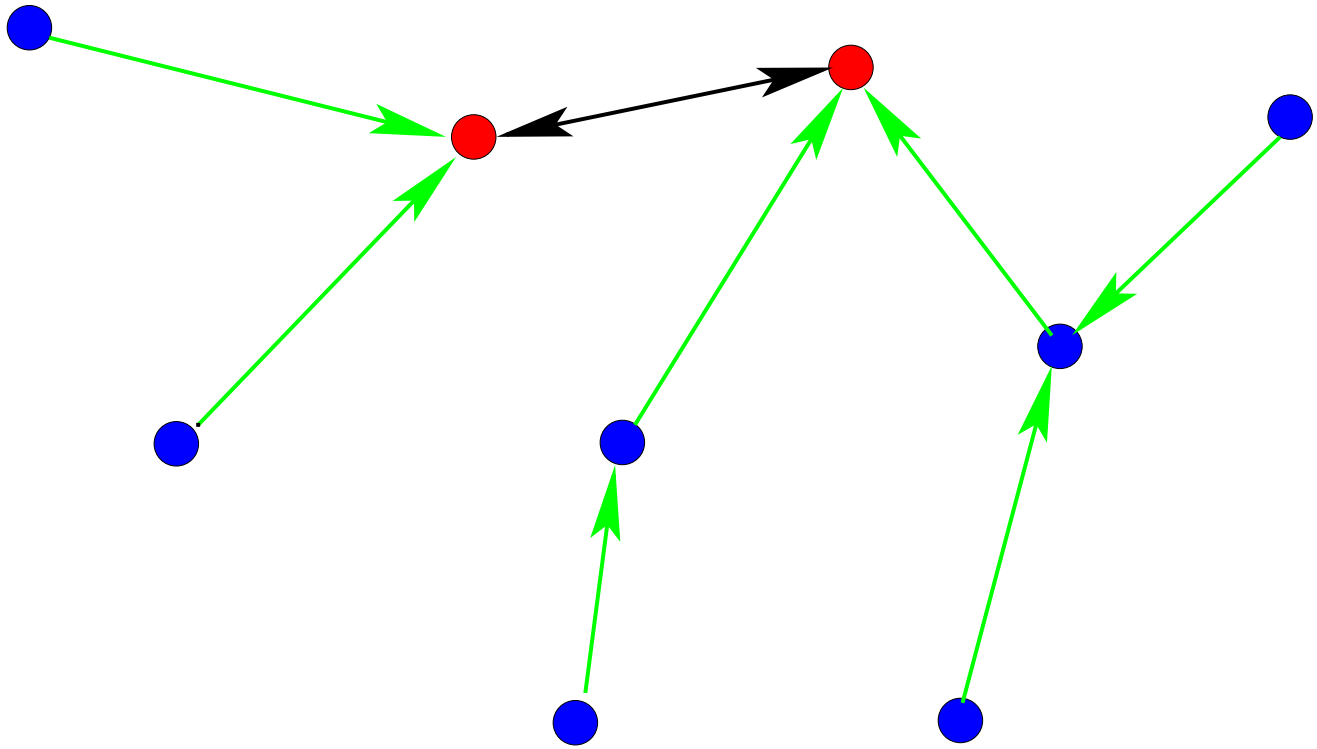
- Proposer une solution partielle.

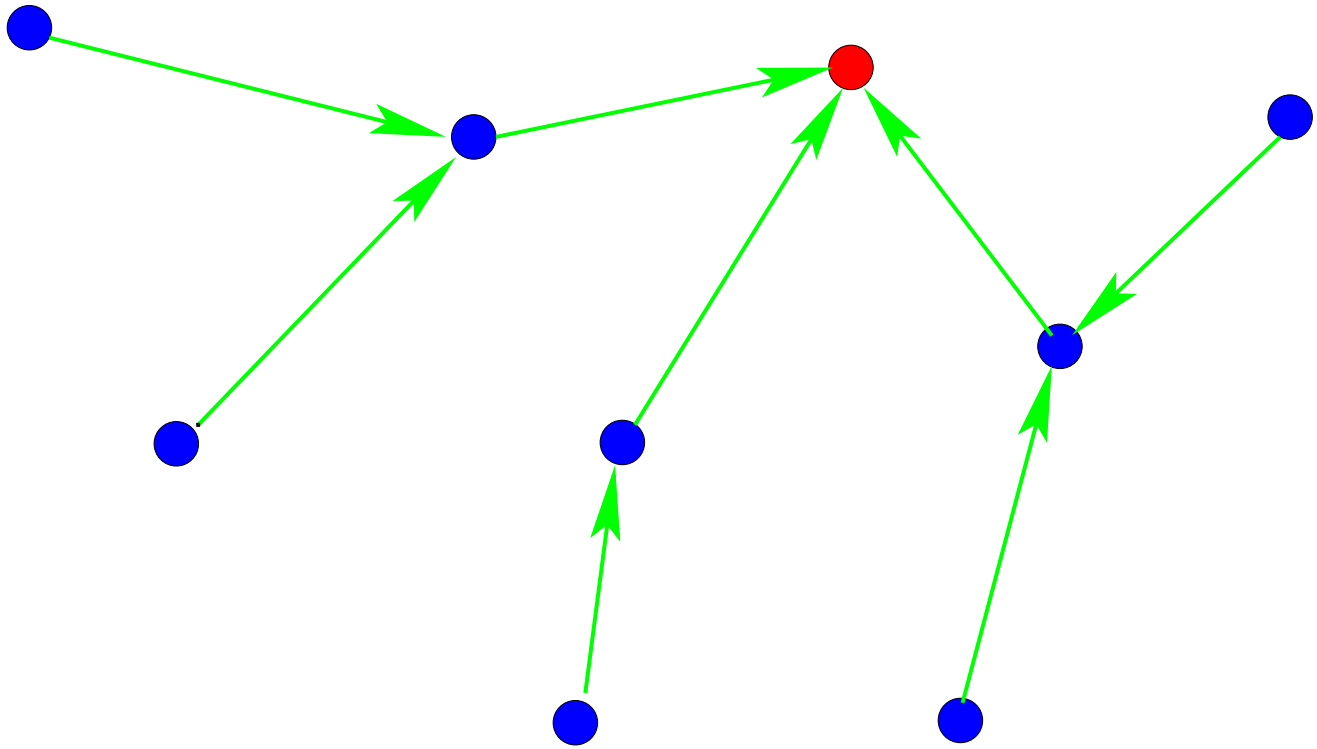
- Vers un meilleur traitement

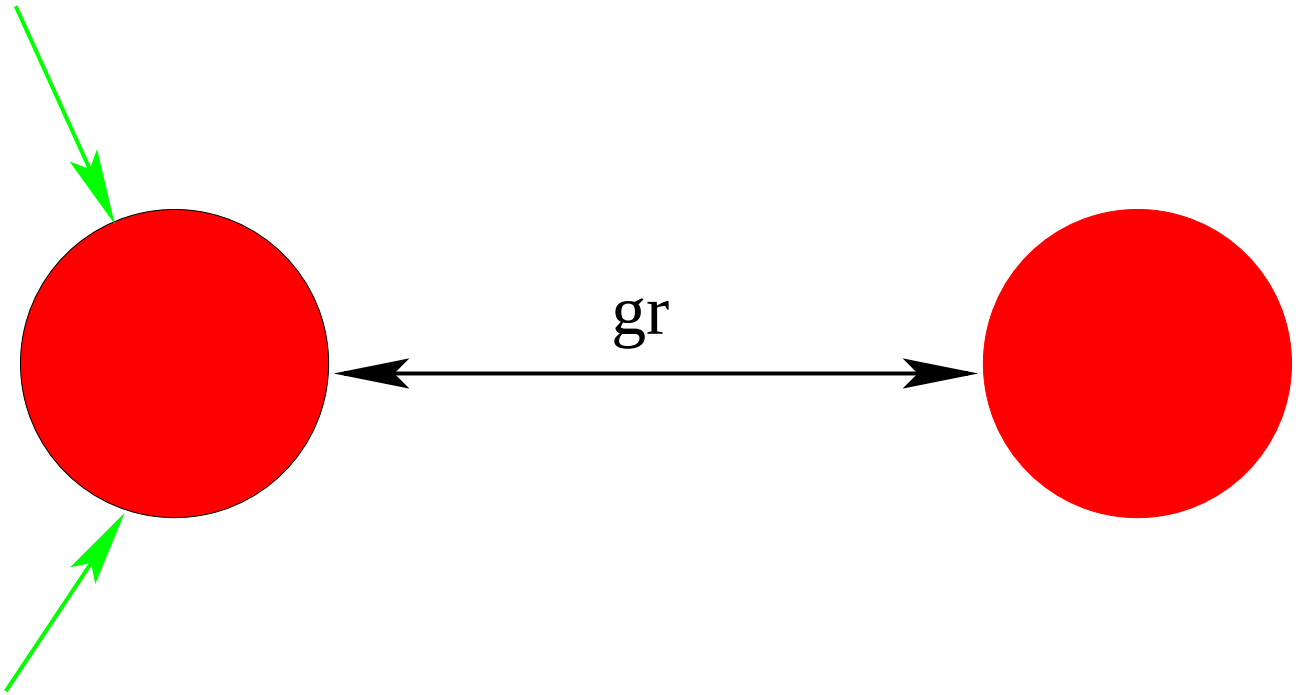
- Retour à l'animation locale.



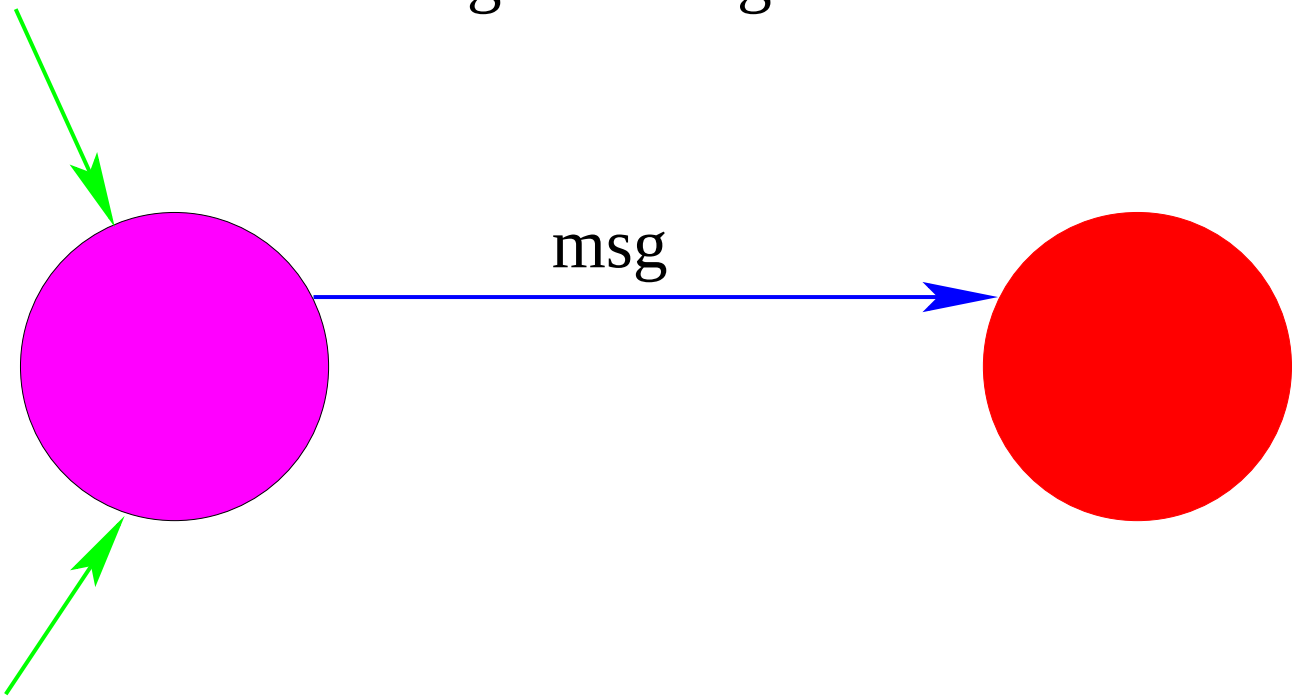




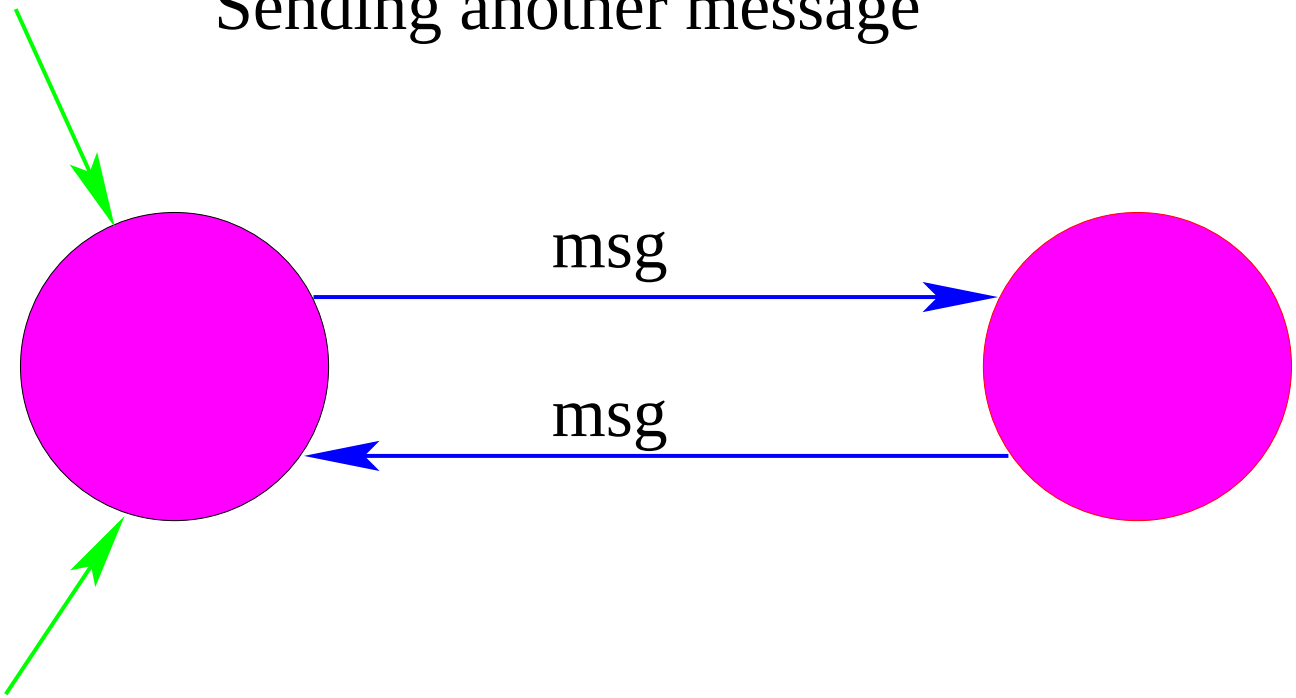



gr

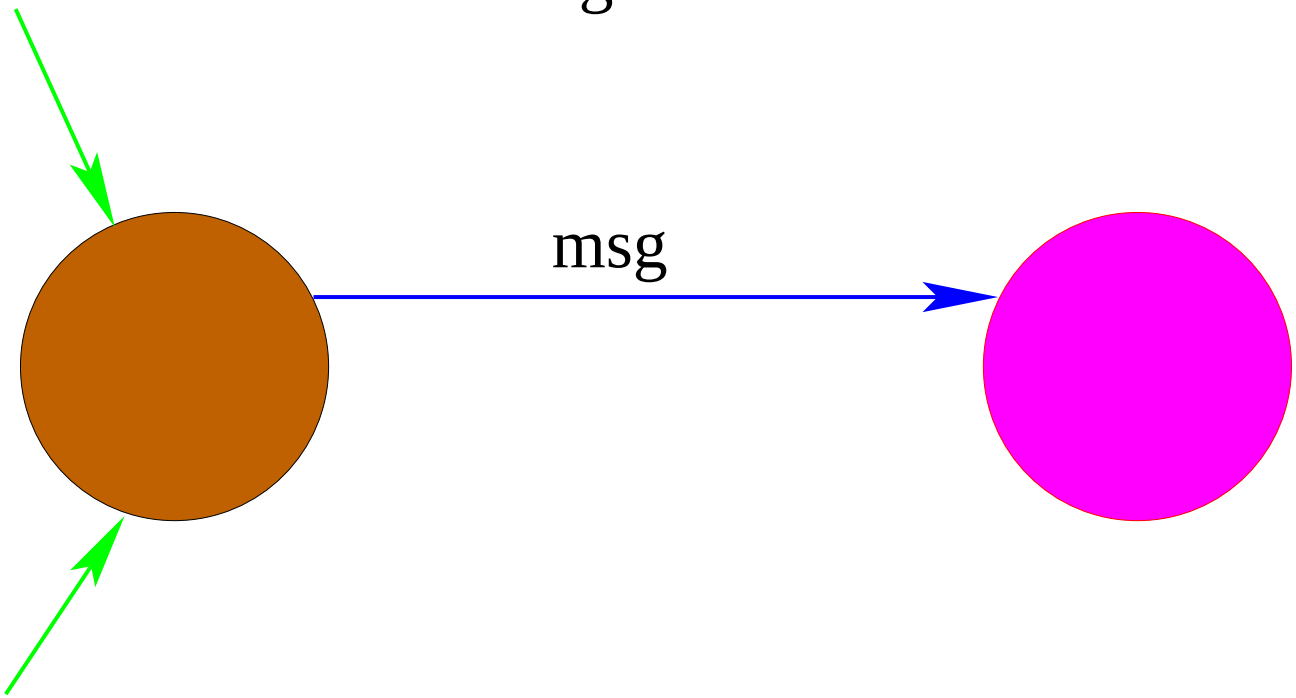
Sending a message



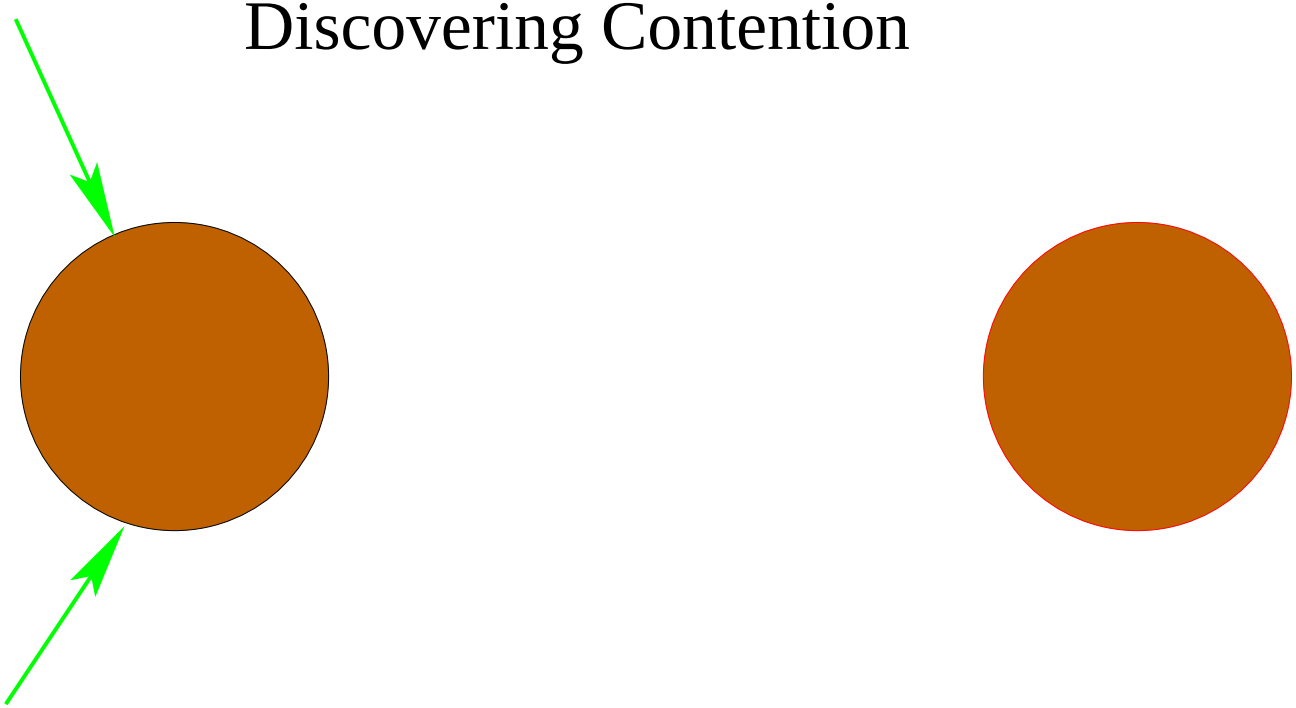
Sending another message



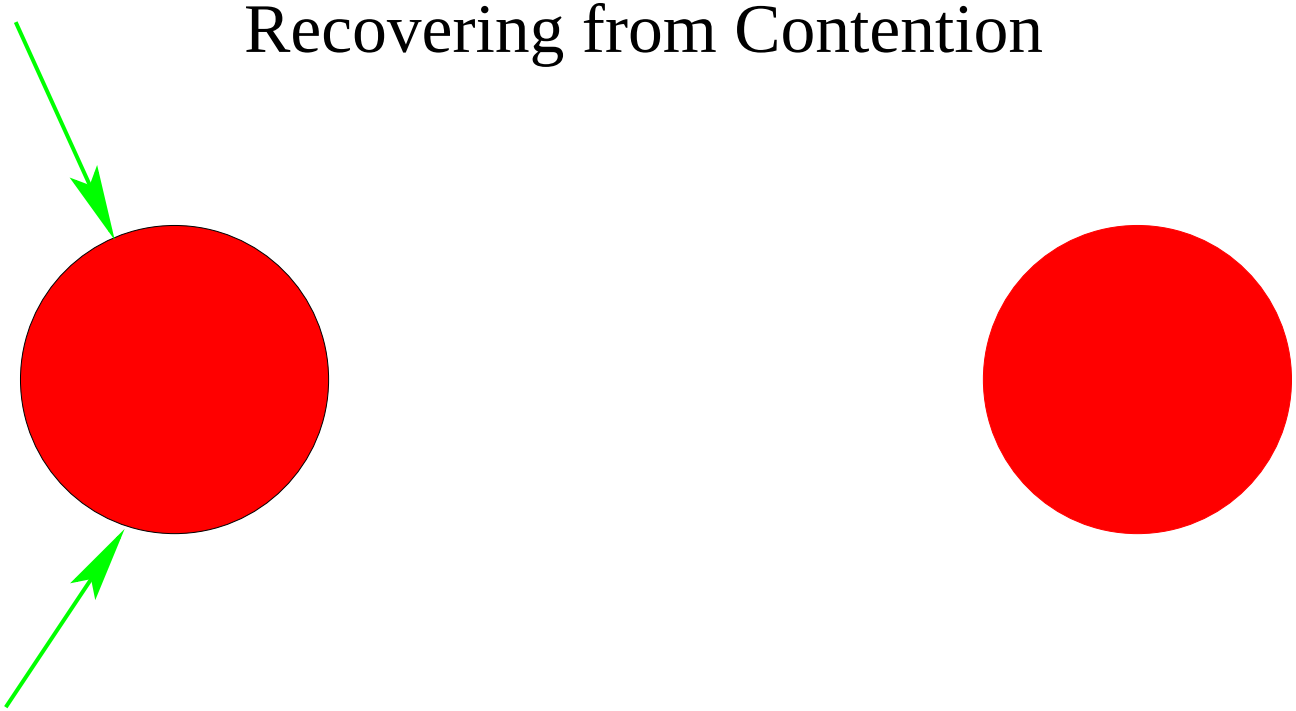
Discovering Contention



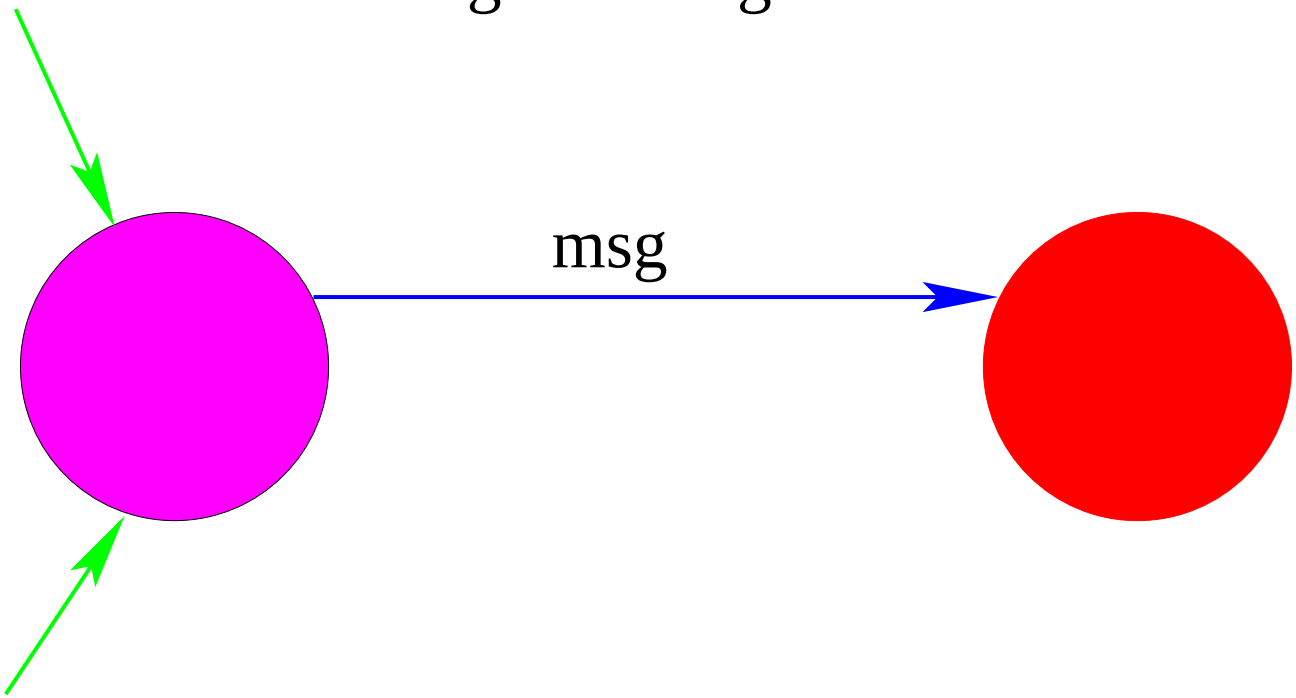
Discovering Contention



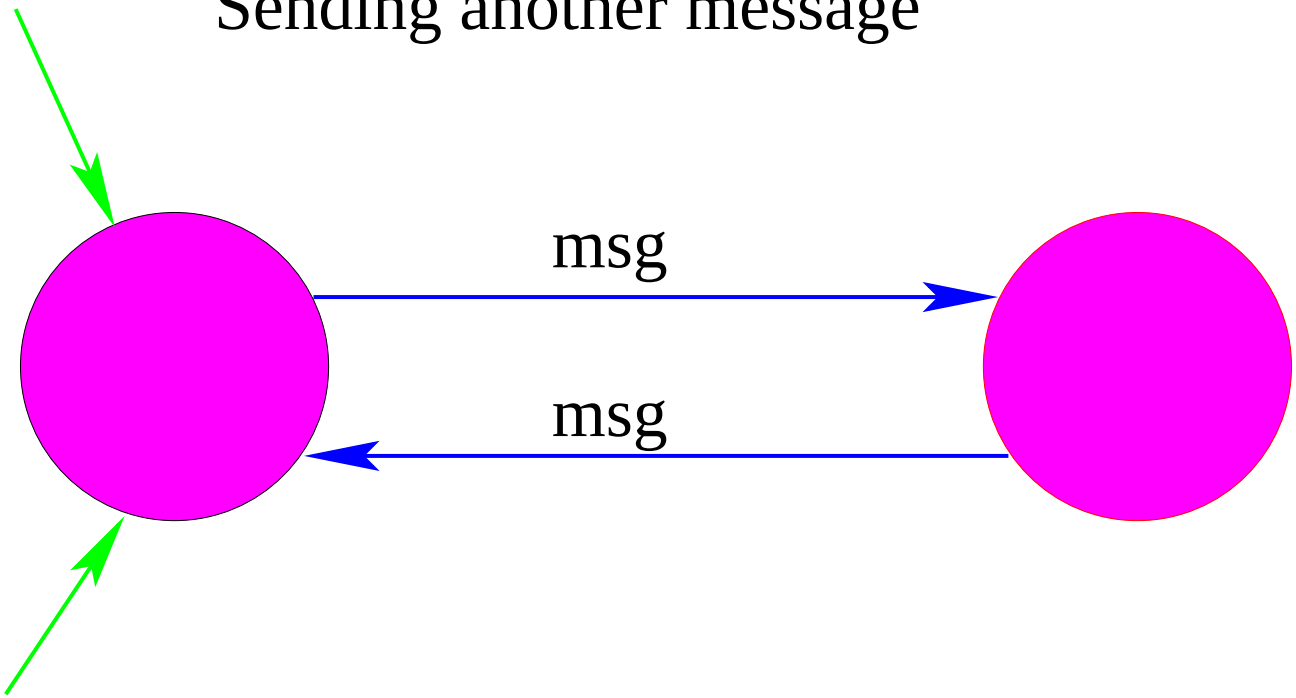
Recovering from Contention



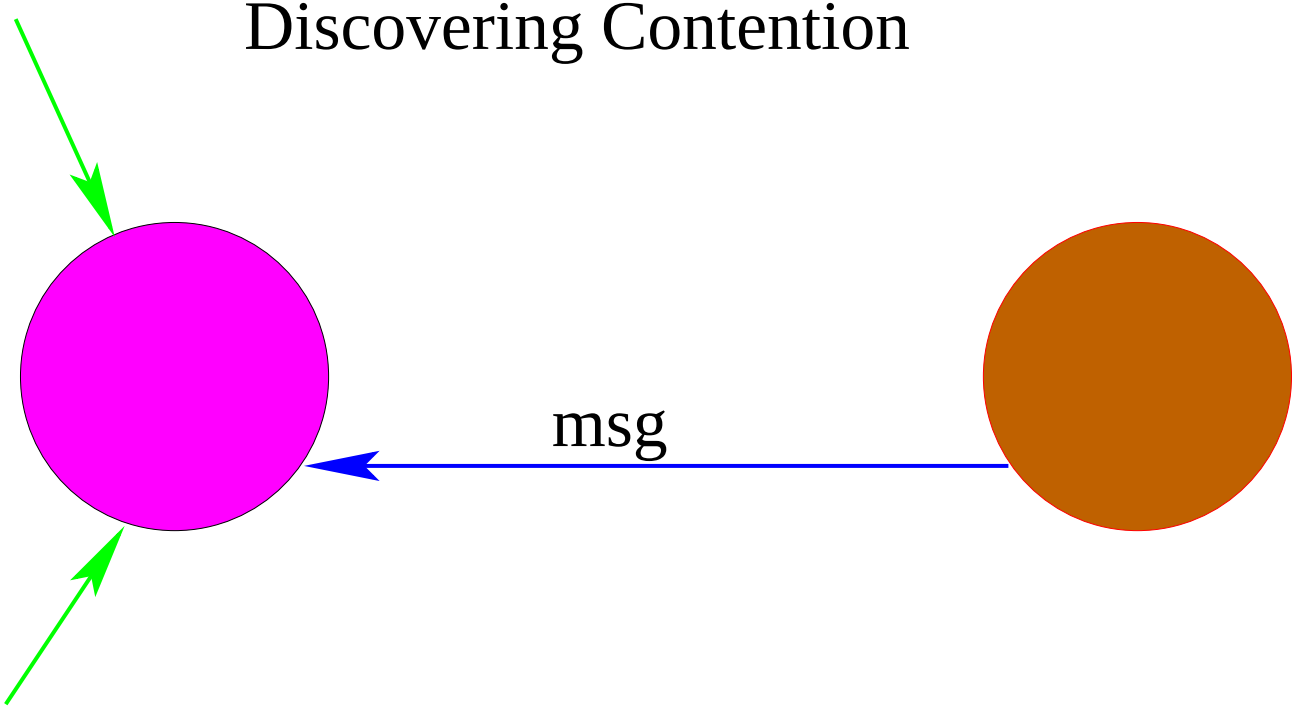
Sending a message



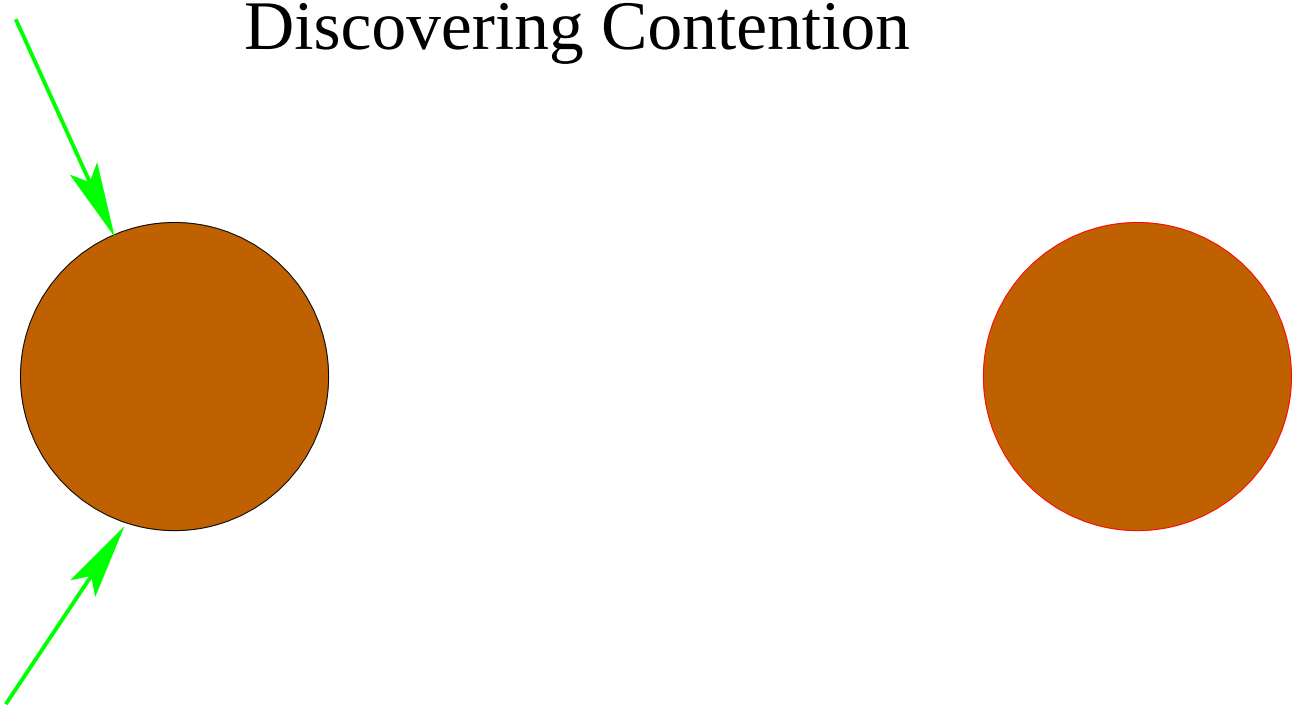
Sending another message



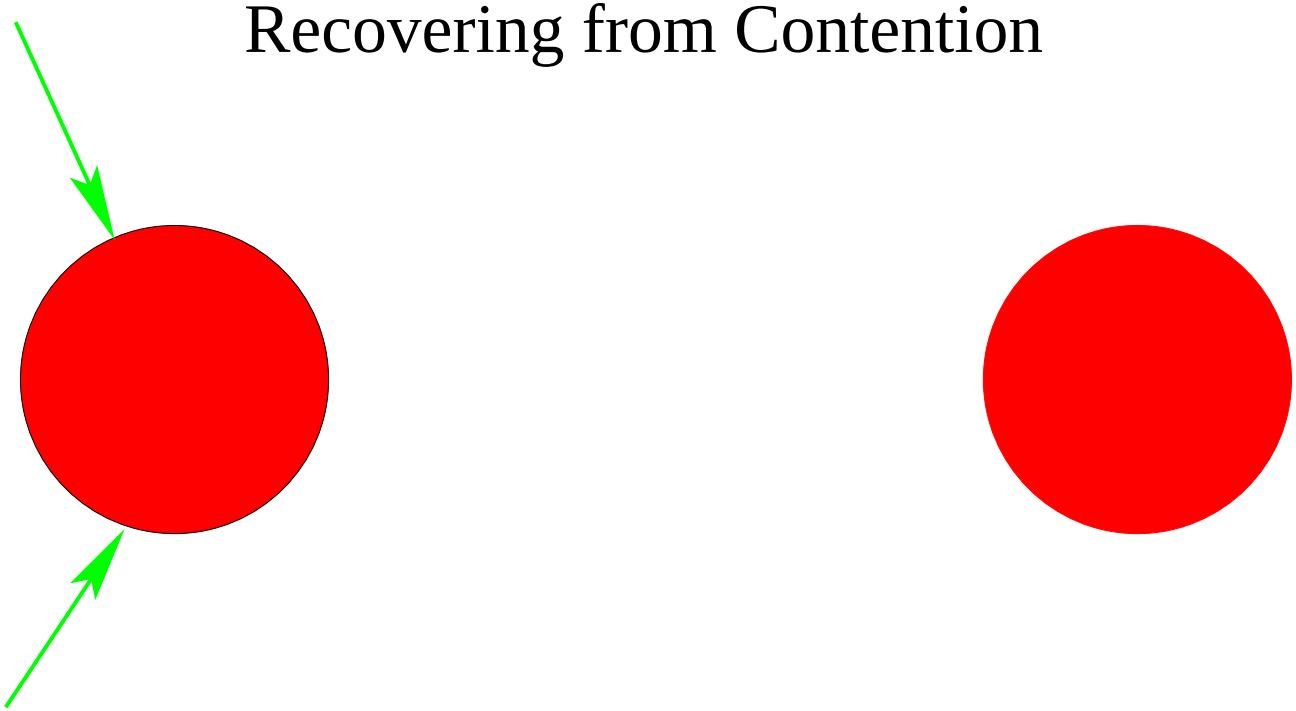
Discovering Contention

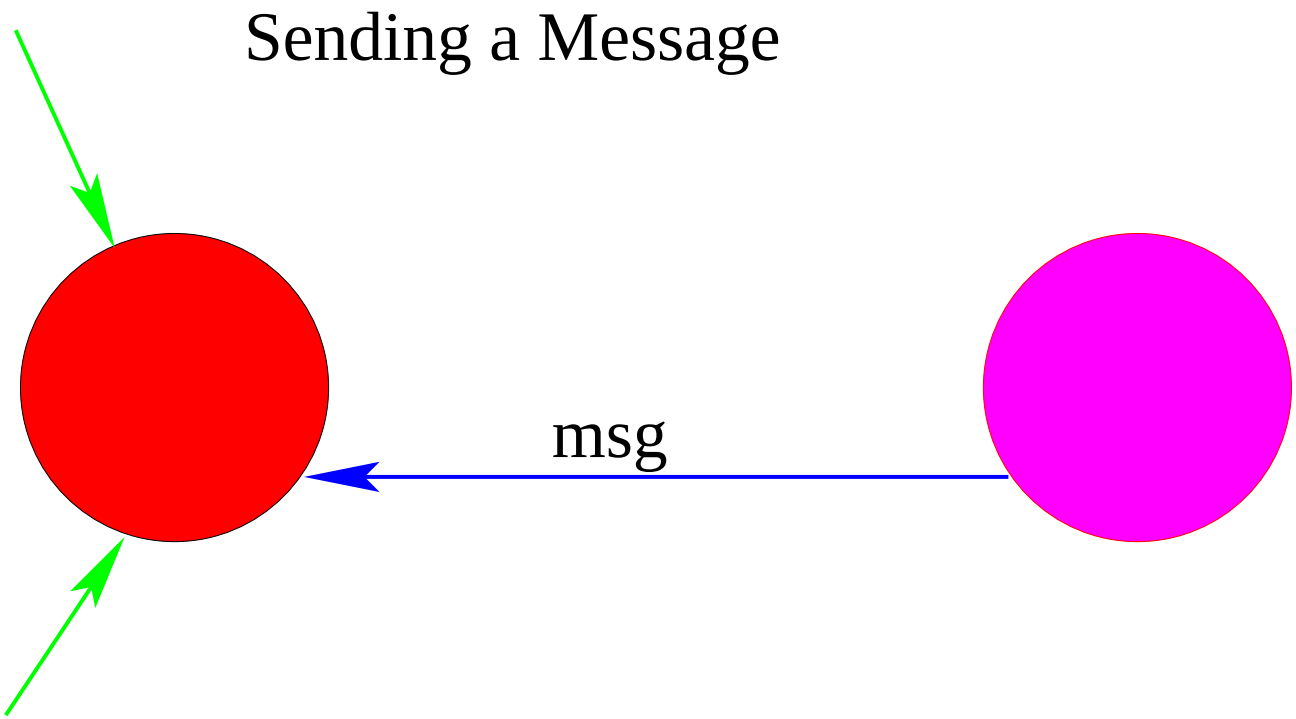


Discovering Contention

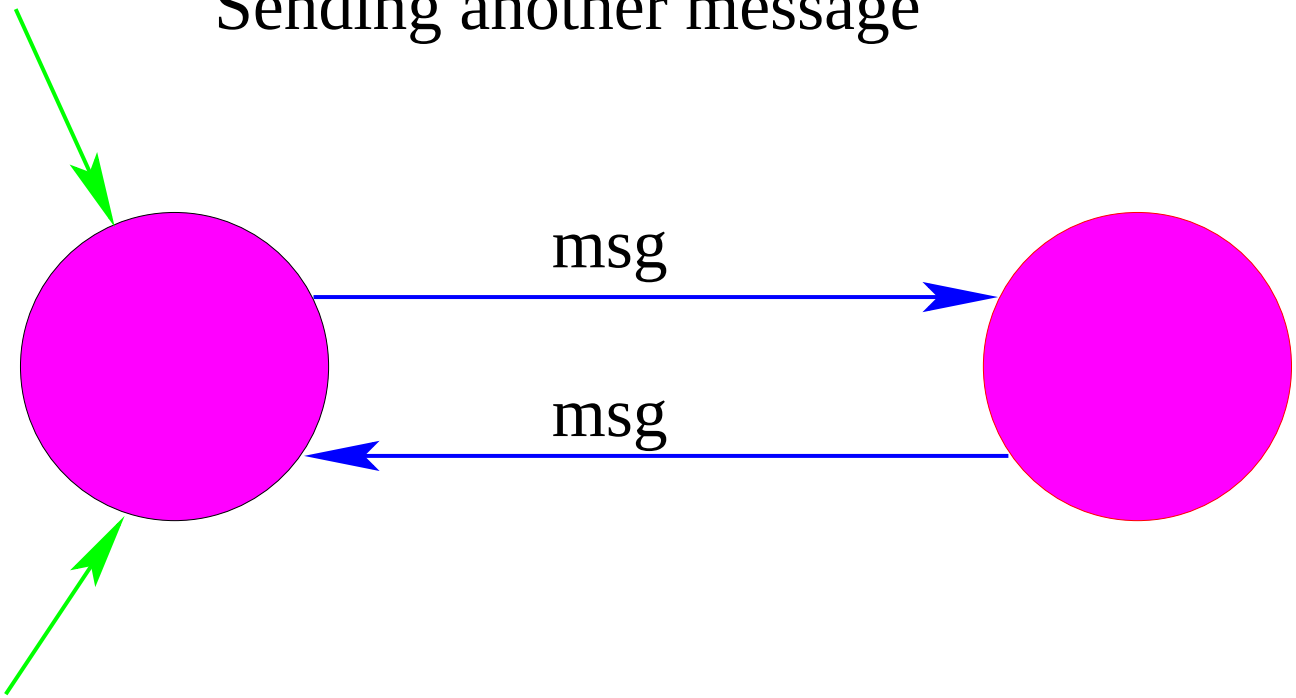


Recovering from Contention

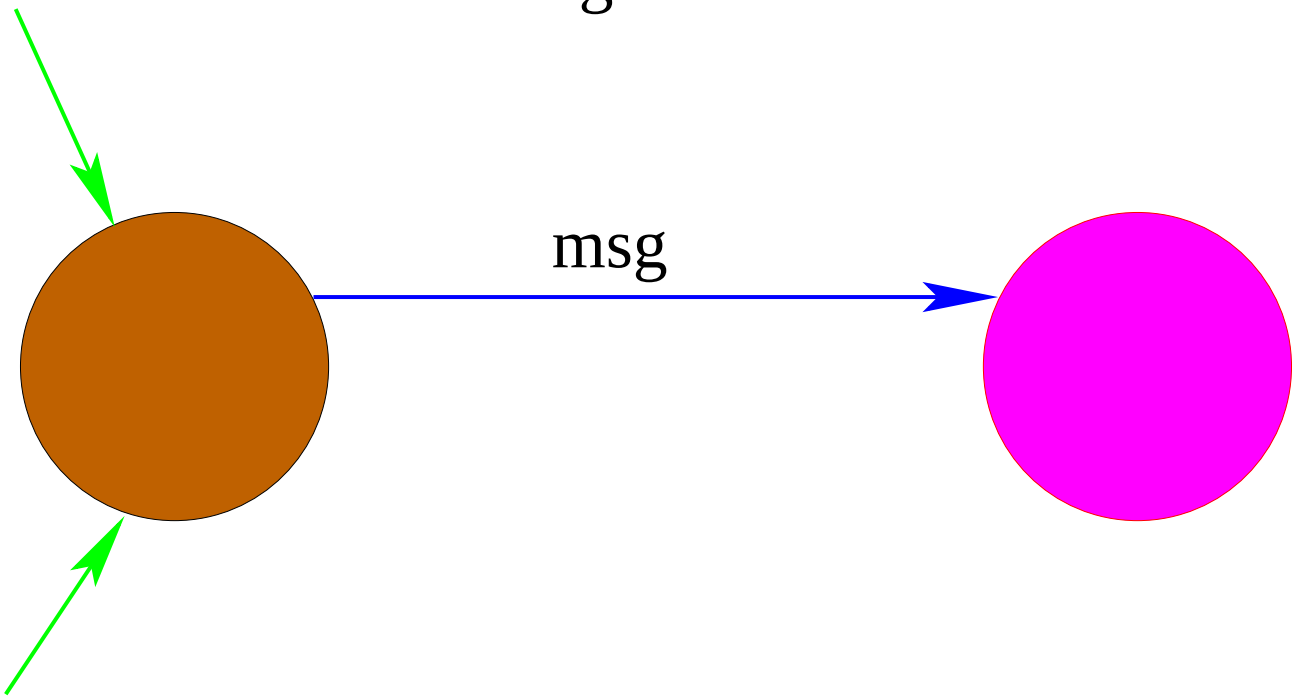



Sending a Message
msg

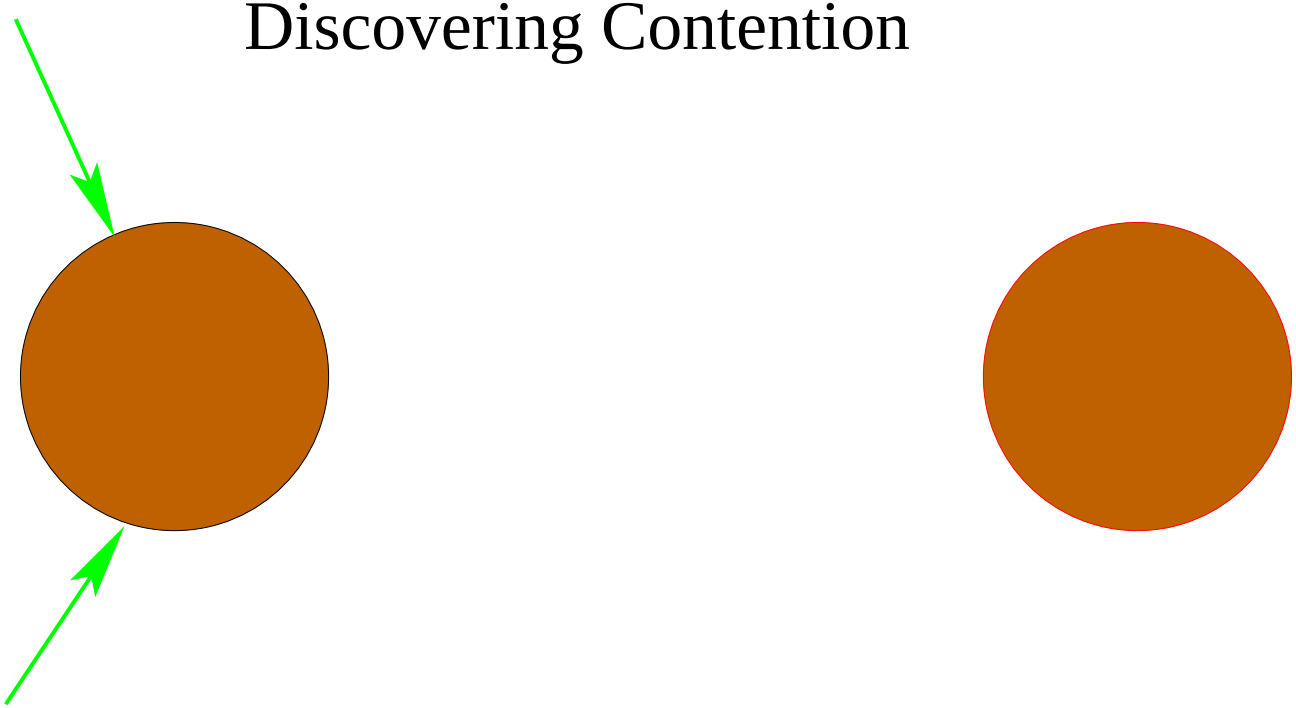
Sending another message



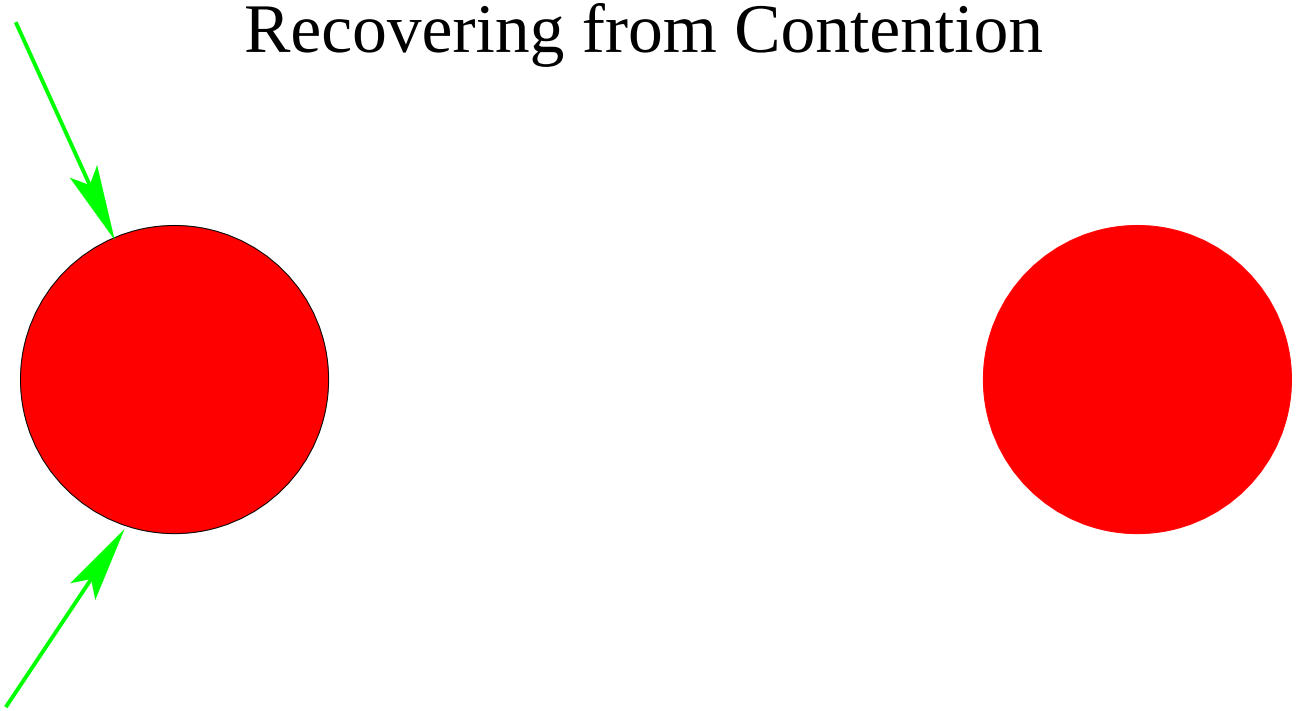
Discovering Contention



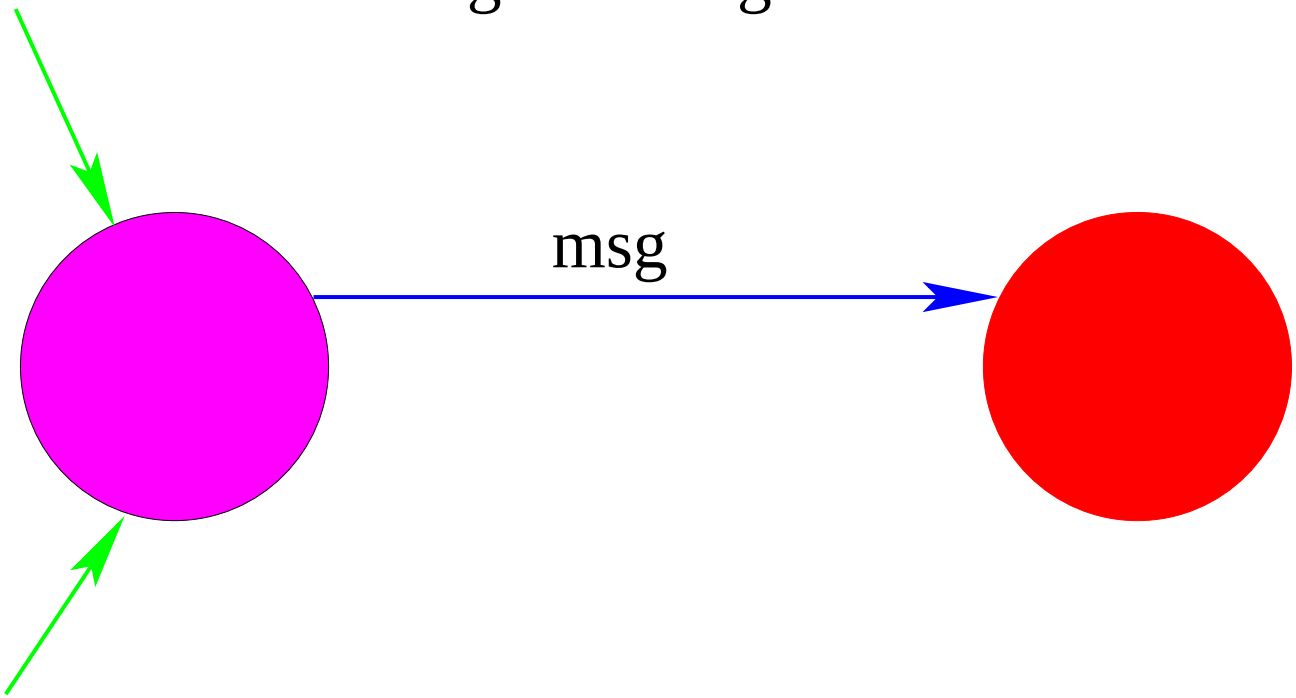
Discovering Contention



Recovering from Contention

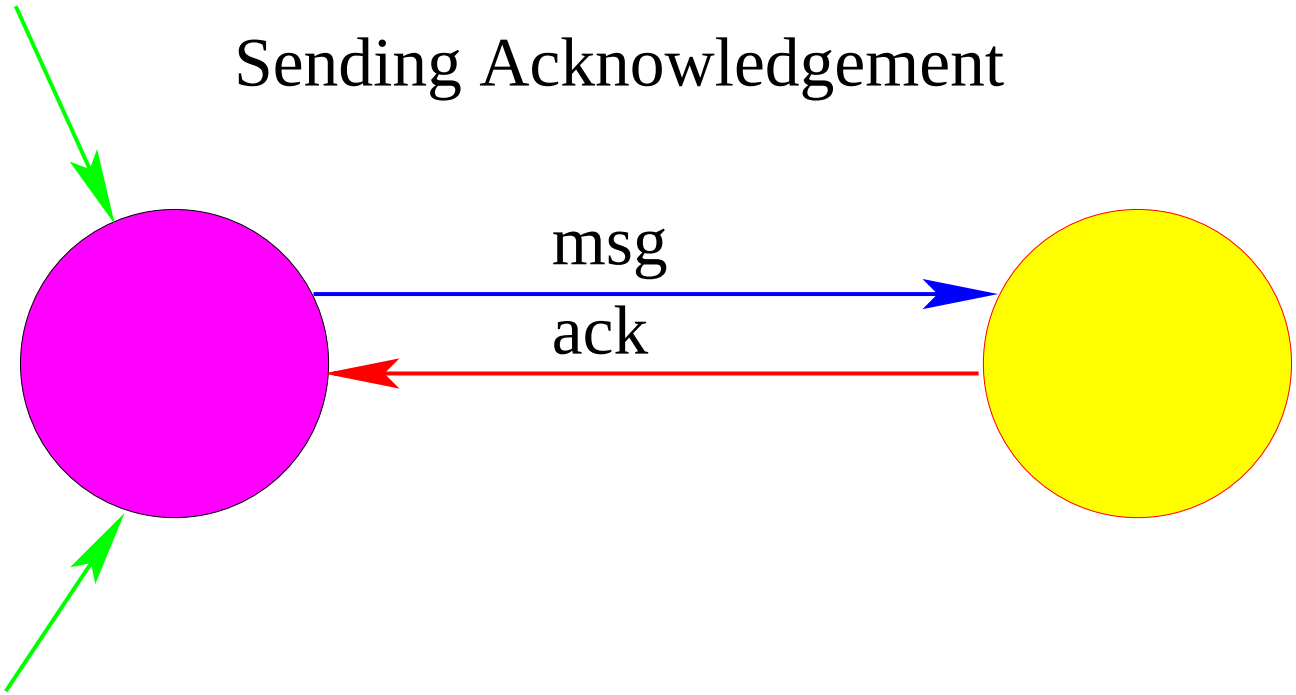


Sending a message



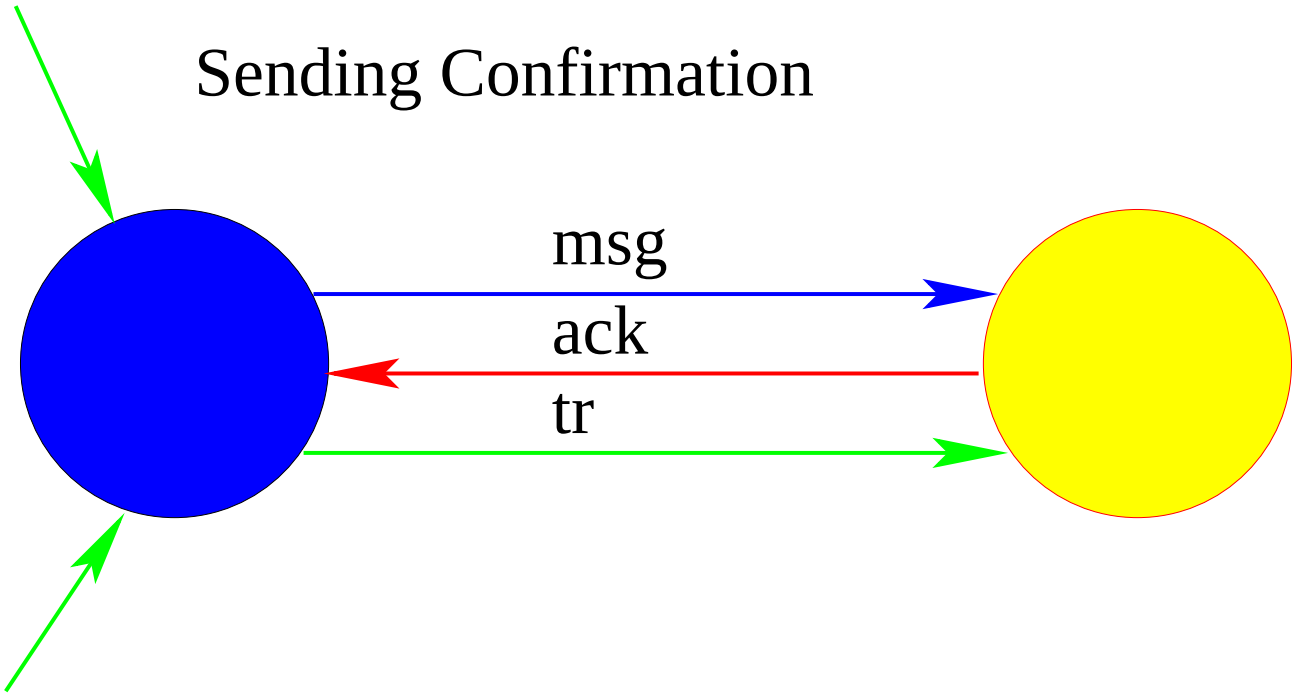
Receiving a message

Sending Acknowledgement

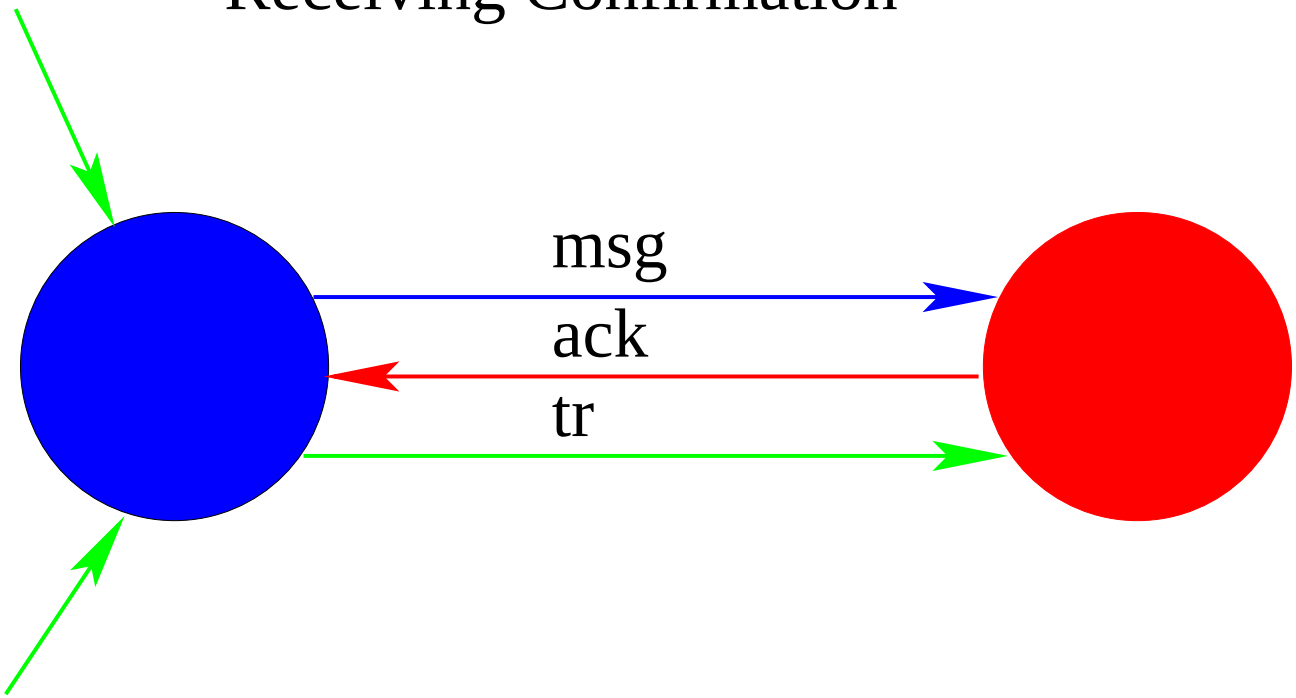


Receiving Acknowledgement

Sending Confirmation



Receiving Confirmation



Découverte de la contention (1)

- Nœud y découvre la contention avec x car :
  - Il a envoyé un message à x
  - Il n'a pas encore reçu une réponse du nœud x
  - Il reçoit à la place un message de x

Découverte de la contention (2)

- x découvre aussi la contention avec y

- Hypothèse : Le temps entre deux découvertes
  EST SUPPOSÉ BORNÉ
  PAR τ ms

- Le temps τ est le temps maximum de transmission entre 2 noeuds connectés

Une solution partielle

- Chaque noeud attend τ ms après sa découverte

- Après cela, chaque noeud sait que l'autre a aussi découvert la contention

- Chaque noeud essaie à nouveau immédiatement

- PROBLEME : Cela peut continuer indéfiniment

Une meilleure solution (1)

- Tout nœud attend τ ms après sa propre découverte

- Chaque nœud choisit avec équiprobabilité :
    - soit d'attendre un délai court
    - soit d'attendre un délai long

- Chaque nœud essaie à nouveau

nœud x

$$\text{EVENT send\_ack} \;\widehat{=}$$
ANY x, y **WHERE**
$x, y \in msg{-}ack \;\wedge$
$y \notin \text{dom}\,(msg)$
THEN
$ack := ack \cup \{x \mapsto y\}$
END

$$\text{EVENT contention} \;\widehat{=}$$
ANY x, y **WHERE**
$x, y \in msg{-}ack \;\wedge$
$y \in \text{dom}\,(msg)$
THEN
$cnt := cnt \cup \{x \mapsto y\}$
END

Introduire un canal de contention factice : cnt

Invariant (3)

$$\forall\,(x,y)\cdot\left(\begin{array}{l} x,y\in msg{-}ack\ \wedge \\ y\in \mathsf{dom}\,(msg) \\ \Longrightarrow \\ y,x\in msg{-}ack \end{array}\right)$$

$$\forall\,(x,y,z)\cdot\left(\begin{array}{ll} x,y\in msg & \wedge \\ z\in gr[\{x\}] & \wedge \\ z\neq y & \\ \Longrightarrow & \\ z,x\in tr & \end{array}\right)$$

Invariant (4)

$$\forall\,(x,y)\cdot\left(\begin{array}{ll} x,y\in msg{-}ack & \wedge \\ y\notin \mathsf{dom}\,(msg) & \\ \Longrightarrow & \\ x,y\notin cnt & \end{array}\right)$$

$$ack\cap ack^{-1}=\varnothing$$

$$ack\cap cnt=\varnothing$$

Résolution de la contention (simuler le délai τ)

EVENT solve_contention $\widehat{=}$

ANY x, y **WHERE**

$x, y \in cnt \cup cnt^{-1}$

THEN

$msg := msg - cnt \quad \|$

$cnt := \varnothing$

END

Récapitulatif du Second Raffinement

- 63 preuves

- Parmi lesquelles 31 interactives

développement

- Etablir le cadre mathématique
- Résoudre le problème mathématique en un coup.
- Résoudre le même problème progressivement.
- Inclure les communications par des messages.
- Localiser les structures de donnée
- Méthodologie fondée sur le raffinement et la preuve
- Autres développements : algorithmique répartie, algorithmique séquentielle, systèmes, SoCs, ...

The predicate invariant (tr)

$$tr \in ND \nrightarrow ND$$

The predicate invariant (tr)

$$tr \in ND \nrightarrow ND$$

$$\text{dom}\,(tr) \lhd (tr \cup tr^{-1}) = \text{dom}\,(tr) \lhd gr$$

Invariant (3)

$$\forall\,(x,y)\cdot\left(\begin{array}{ll} x,y\in msg & \wedge \\ y,x\in msg & \\ \Longrightarrow & \\ y,x\notin ack & \end{array}\right)$$

$$\forall\,(x,y,z)\cdot\left(\begin{array}{ll} x,y\in msg & \wedge \\ z\in gr[\{x\}] & \wedge \\ z\neq y & \\ \Longrightarrow & \\ z,x\in tr & \end{array}\right)$$

Invariant (4)

$$\forall\,(x,y)\cdot\left(\begin{array}{ll} x,y\in cnt & \\ \Longrightarrow & \\ x,y\in msg{-}ack & \wedge \\ y\in \mathsf{dom}\,(msg) & \end{array}\right)$$

$$ack\cap ack^{-1}=\varnothing$$

$$ack\cap cnt=\varnothing$$

Third Refinement : Localization

- The representation of the graph gr is modified

- The representation of the tree tr is modified

- Other data structures are localized

Localization (1)

The graph gr and the tree tr are now localized

$$nb \in ND \to \mathbb{P}(ND)$$

$$\forall x \cdot (\, x \in ND \implies nb(x) = gr[\{x\}]\,)$$

$$sn \in ND \to \mathbb{P}(ND)$$

$$\forall x \cdot (\, x \in ND \implies sn(x) \subseteq tr^{-1}[\{x\}]\,)$$

Localization (2)

$$bm \subseteq ND$$

$$bm = \mathsf{dom}\,(msg)$$

$$bt \subseteq ND$$

$$bt = \mathsf{dom}\,(tr)$$

$$ba \in ND \to \mathbb{P}(ND)$$

$$\forall x \cdot (\, x \in ND \implies ba(x) = ack^{-1}[\{x\}]\,)$$

- Node x is elected the leader

$$\begin{array}{l}
\text{EVENT elect} \;\widehat{=} \\
\quad \textbf{ANY } x \textbf{ WHERE} \\
\qquad x \in ND \;\wedge \\
\qquad nb(x) = sn(x) \\
\quad \textbf{THEN} \\
\qquad rt := x \\
\quad \textbf{END}
\end{array}$$

- Node x sends a message to node y (y is unique)

$$
\begin{array}{l}
\text{EVENT send\_msg} \ \widehat{=} \\
\quad \textbf{ANY } x, y \textbf{ WHERE} \\
\qquad x \in ND{-}bm \ \wedge \\
\qquad y \in ND{-}ba(x) \ \wedge \\
\qquad nb(x) = sn(x) \cup \{y\} \\
\quad \textbf{THEN} \\
\qquad msg := msg \cup \{x \mapsto y\} \quad \| \\
\qquad bm := bm \cup \{x\} \\
\quad \textbf{END}
\end{array}
$$

- Node y sends an acknowledgement to node x

$$
\begin{array}{l}
\text{EVENT send\_ack} \ \widehat{=} \\
\quad \textbf{ANY } x, y \textbf{ WHERE} \\
\qquad x, y \in msg \ \wedge \\
\qquad x \notin ba(y) \ \wedge \\
\qquad y \notin bm \\
\quad \textbf{THEN} \\
\qquad ack := ack \cup \{x \mapsto y\} \quad \| \\
\qquad ba(y) := ba(y) \cup \{x\} \\
\quad \textbf{END}
\end{array}
$$

- Node x sends a confirmation to node y

$$
\begin{array}{l}
\text{EVENT progress} \ \widehat{=} \\
\quad \textbf{ANY } x, y \textbf{ WHERE} \\
\qquad x, y \in ack \ \wedge \\
\qquad x \notin bt \\
\quad \textbf{THEN} \\
\qquad tr := tr \cup \{x \mapsto y\} \quad \| \\
\qquad bt := bt \cup \{x\} \\
\quad \textbf{END}
\end{array}
$$

- Node y receives confirmation from node x

$$
\begin{array}{l}
\text{EVENT rcv\_cnf} \ \widehat{=} \\
\quad \textbf{ANY } x, y \textbf{ WHERE} \\
\qquad x, y \in tr \ \wedge \\
\qquad x \notin sn(y) \\
\quad \textbf{THEN} \\
\qquad sn(y) := sn(y) \cup \{x\} \\
\quad \textbf{END}
\end{array}
$$

$$\text{EVENT contention} \;\widehat{=}$$

ANY x, y **WHERE**

$$x, y \in cnt \cup cnt^{-1} \;\wedge$$
$$x \notin ba(y) \;\wedge$$
$$y \in bm$$

THEN

$$cnt := cnt \cup \{x \mapsto y\}$$

END

EVENT solve_contention $\widehat{=}$

ANY x, y **WHERE**

$x, y \in cnt \cup cnt^{-1}$

THEN

$msg := msg - cnt \quad \|$

$bm := bm - \mathsf{dom}\,(cnt) \quad \|$

$cnt := \varnothing$

END

Section Courante

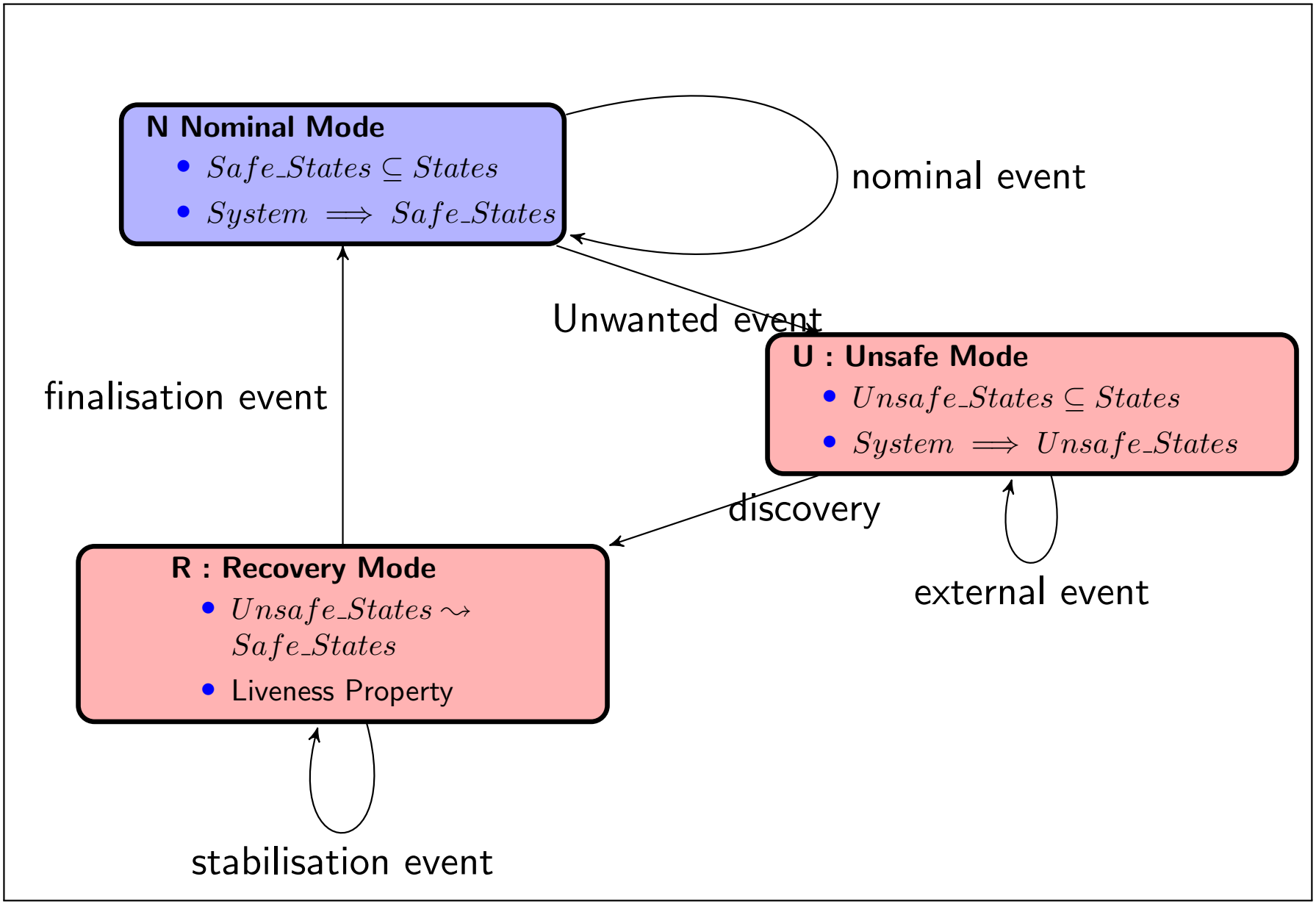
1 Auto-stabilisation]
- Généralités
- Définition
- Réseaux en anneau
- Algorithme de Dijkstra
- Algorithme de coloriage

Etat courant

Problème de l'élection
Problème de l'élection
Election dans un graphe acyclique

1 Auto-stabilisation
Généralités
Définition
Réseaux en anneau
Algorithme de Dijkstra
Algorithme de coloriage

Auto-stabilisation dans les systèmes répartis



Stabilisation

- Les systèmes peuvent être victimes de fautes transitoires
- Un certain nombre d'algorithmes permettent d'anticiper de tels problèmes.
- Les algorithmes auto-stabilisants constituent une alternative à la perte de l'état consistant d'un système donné à la suite d'une défaillance.
- Différents phénomènes peuvent apparaître lors de l'exécution d'un algorithme : en particulier peuvent se produire des changements dans le réseau (ajout ou disparition d'un sommet ou d'un lien) ou bien encore des altérations de messages ou de mémoires.
- Un algorithme est dit auto-stabilisant s'il se termine correctement en dépit de l'apparition de ces phénomènes.
- Un algorithme auto-stabilisant ne nécessite pas d'initialisation particulière.

Contexte

- Les systèmes répartis sont exposés à des risques de défaillances transitoires qui peuvent placer un système donné dans un état ou une configuration arbitraire.
- Cette configuration arbitraire peut être une configuration ne satisfaisant plus l'invariant du système.
- Dans ce cas, la question est de concevoir un algorithme réparti permettant de faire converger le système global d'une configuration arbitraire vers une configuration dite stable et satisfaisant l'invariant.

Etat courant

Problème de l'élection
Problème de l'élection
Election dans un graphe acyclique

1 Auto-stabilisation
Généralités
Définition
Réseaux en anneau
Algorithme de Dijkstra
Algorithme de coloriage

Modélisation d'un système auto-stabilisant

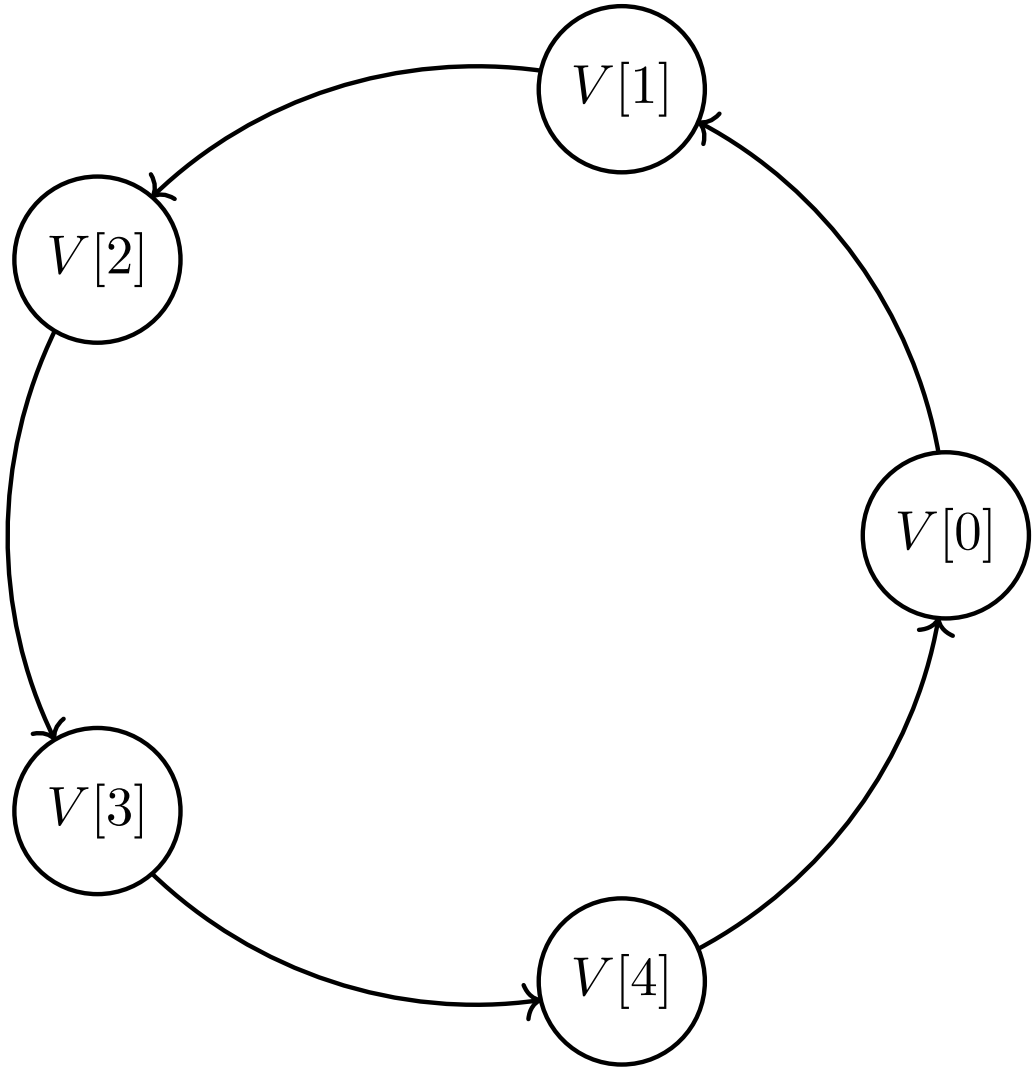
- Soit un système réparti $\mathcal{S}$ modélisé par un système de transition $(States, \longrightarrow)$ où $States$ est un ensemble d'états ou de configurations et où $\longrightarrow$ modélise la relation de transition simulant l'activité du système.
- Une exécution ou un calcul de $\mathcal{S}$ est une suite maximale $(\sigma_0, \sigma_1, \sigma_2, \ldots, \sigma_i, \ldots)$ engendrée par $(States, \longrightarrow)$.
- Tout préfixe d'une telle suite est un calcul puisqu'il n'y a pas d'états initiaux déterminés.
- Tout préfixe d'exécution est donc un calcul ou une exécution de $\mathcal{S}$ (pas de dépendance d'un état initial)

Etat courant

Problème de l'élection
Problème de l'élection
Election dans un graphe acyclique

1 Auto-stabilisation
Généralités
Définition
Réseaux en anneau
Algorithme de Dijkstra
Algorithme de coloriage

Réseau en anneau



Etat courant

Problème de l'élection
Problème de l'élection
Election dans un graphe acyclique

1 Auto-stabilisation
Généralités
Définition
Réseaux en anneau
Algorithme de Dijkstra
Algorithme de coloriage

Problème de l'exclusion mutuelle

Description

- Dans toute configuration, au plus un processus a le privilège.
- Chaque processus a le privilège infiniment souvent.

- Dans un réseau avec une topologie en anneau, des processus ont accès à une ressource.
- Le privilège est accordé à un processus à la fois.
- Le problème est que le privilège est géré par un jeton qui peut être perdu et donc être dans une configuration ne satisfaisant plus la propriété.

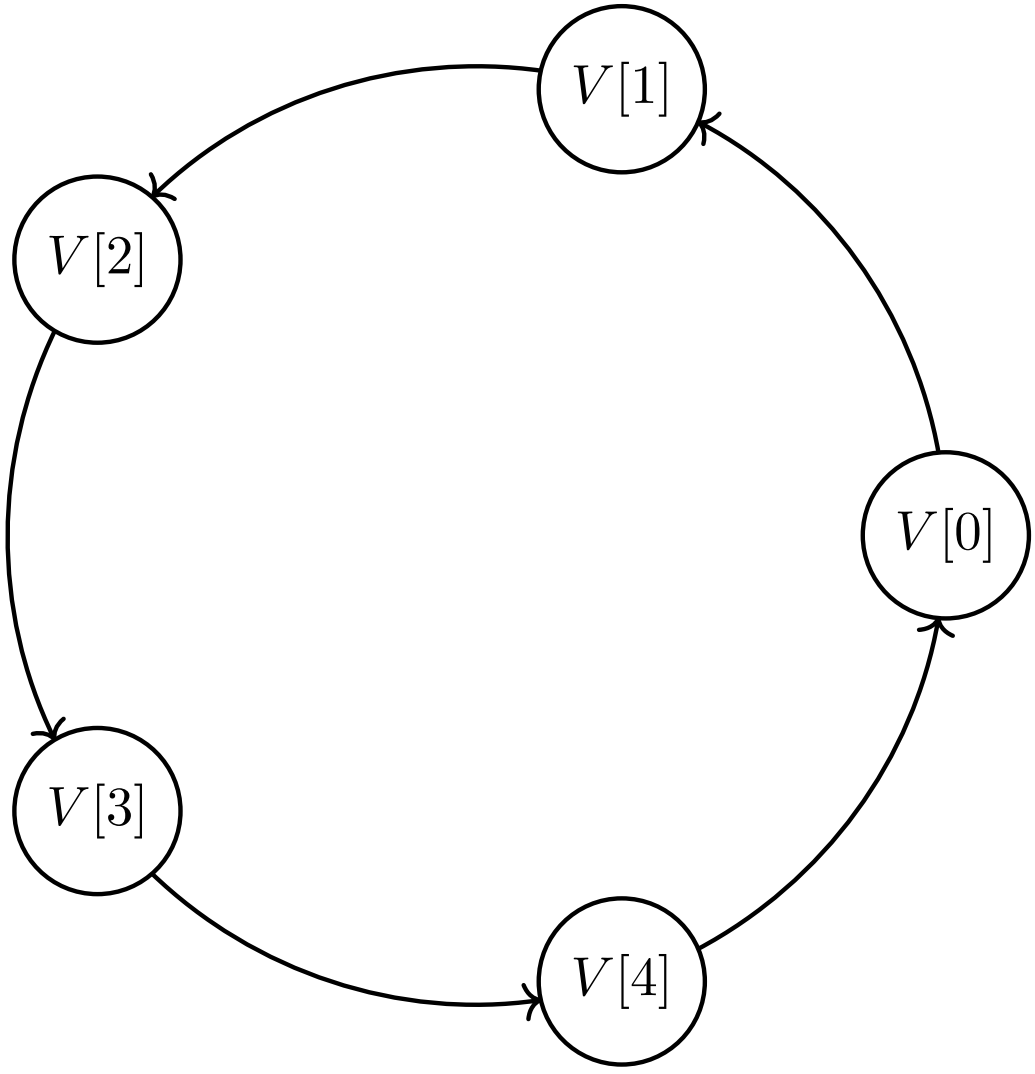
Algorithme de Dijkstra

$\text{DOMAINE} \triangleq 0..N$
$\text{IMAGE} \triangleq 0..M$

$\text{NToZero} \triangleq$
$\quad \wedge \quad V[0] = V[N]$
$\quad \wedge \quad V' = [V \ \textbf{EXCEPT} \ ![0] = (V[0]+1)\%(M+1)]$

$\text{Others(I)} \triangleq$
$\quad \wedge \quad I \in 0..N$
$\quad \wedge \quad I \neq N$
$\quad \wedge \quad V[I+1] \neq V[I]$
$\quad \wedge \quad V' = [V \ \textbf{EXCEPT} \ ![I+1] = V[I]]$

Réseau en anneau



Propriétés de stabilisation

```
Init == V = [i \in 0..N |-> (IF i # N THEN i  ELSE 0)]
Next == NToZero \/ (\E I \in 0..N-1:Others(I))

Prop1 ==    (V[0] = V[N])
              => (\A i \in 1..N-1: V[i+1] = V[i])

Prop2 ==
        \/   (\E i,j  \in 0..N-1: i # j
               /\ V[i+1] #  V[i] /\ V[j+1] #  V[j])
        \/ (V[0] = V[N] /\
              (\E i  \in 0..N-1:  V[i+1] #  V[i]))
```

Etat courant

Problème de l'élection
Problème de l'élection
Election dans un graphe acyclique

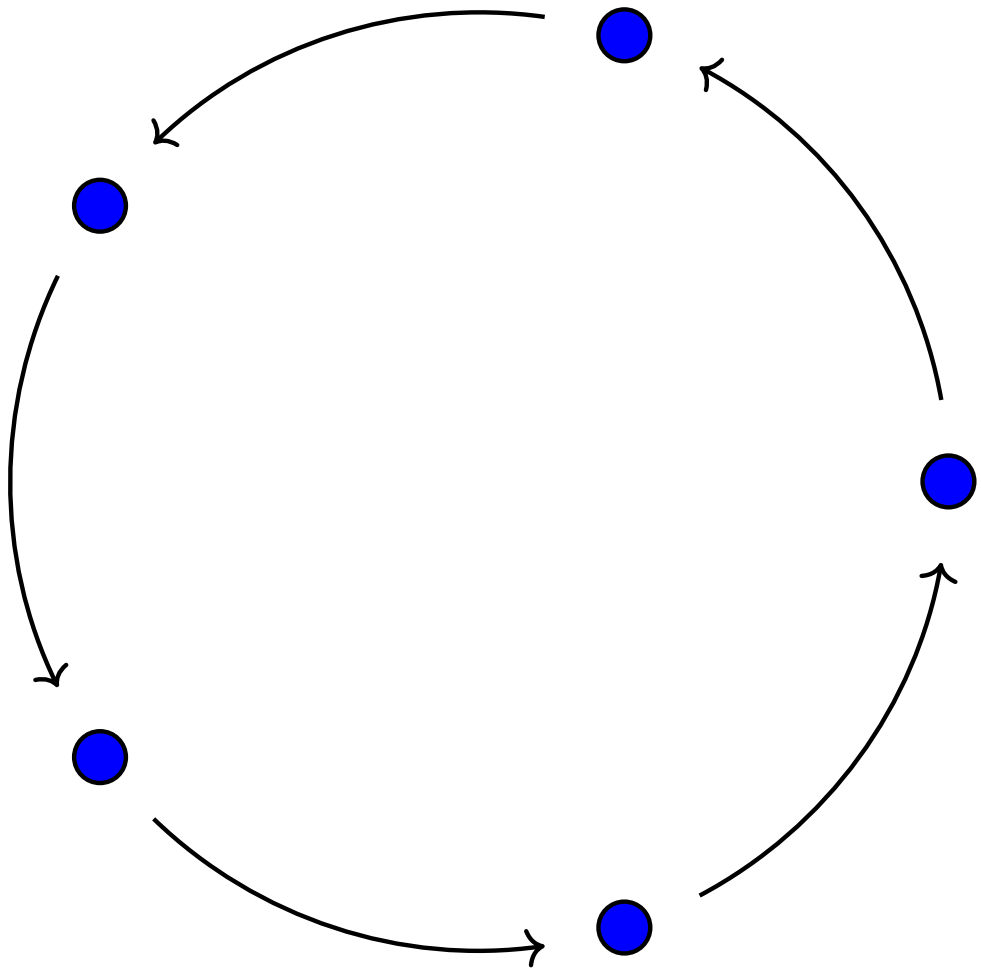
1 Auto-stabilisation
Généralités
Définition
Réseaux en anneau
Algorithme de Dijkstra
Algorithme de coloriage

Coloriage d'un anneau

Le problème du coloriage d'un anneau concerne l'existence d'une configuration dans laquelle trois nœuds consécutifs n'ont pas la même couleur et pose la question de définir un algorithme réparti permettant de l'atteindre à parrtir de toute configuration intiale de l'anneau.

- Modéliser un processus de coloration d'un anneau à partir d'une situation quelconque initiale pour atteindre une situation stable caractérisée par le fait que deux nœuds voisins n'ont pas la même couleur.
- La règle de changement de couleur : Si un nœud a la même couleur que l'un de ses vosins, il chosit une autre couleur que celle de ses voisins.
- La règle est applicable tant que deux nœuds ont la même couleur.
- La règle est non-applicable dès que les nœuds adjacents ont tous une couleur différente : le cas d'un anneau à un nœud est exclus en considérant qu'il y a au moins deux nœuds dans l'anneau.

Exemple d'anneau



CONTEXT $ring$
SETS
N
CONSTANTS
n
AXIOMS
$axm1 : n \in N \rightarrowtail\!\!\!\!\!\rightarrow N$
$axm2 : \forall s \cdot s \subseteq n[s] \Rightarrow N \subseteq s$
$axm3 : finite(N)$
END
MACHINE $one{-}shot$
SEES $ring, color$
VARIABLES
col
INVARIANTS
$inv1 : col \in N \rightarrow color$

EVENTS
EVENT INITIALISATION
$act1 : col :\in N \rightarrow color$
EVENT stable
ANY
f
WHERE
$grd1 : f \in N \rightarrow color$
$grd2 : \forall x \cdot f(x) \notin f[(n \cup n^{-1})[\{x\}]]$
THEN
$act1 : col := f$
END

Le modèle définit la solution à trouver par un événement abstrait exprimant le lien entre la configuration initiale et une configuration stable. Les régles données explique comment le calcul est effectué par les différents processus du réseau. Chaque règle suppose que leur application est possible quand trois nœuds consécutifs vérifient une condition donnée.

Chaque règle constitue un élément de calcul réparti et cet élément est appliqué sur la *boule* centrée sur un nœud donné et tient compte des couleurs des nœpériphériques.

MACHINE $rule$ **REFINES** $one{-}shot$
SEES $ring, color$
VARIABLES col, c
INVARIANTS
$inv1 : c \in N \to color$
THEOREMS
$thm1 :$

$$\left(\begin{array}{l} \forall x \cdot c(x) \neq c(n(x)) \\ \lor \\ \exists x, clr \cdot c(x) = c(n(x)) \land clr \neq c(x) \land clr \neq c(n(x))) \end{array} \right)$$

EVENTS

EVENT INITIALISATION

$$act1 : col, c : | \left(\begin{array}{l} col' \in N \to color \\ \wedge \\ c' \in N \to color \\ \wedge \\ col' = c' \end{array} \right)$$

EVENT stable REFINES stable

WHEN

$grd1 : \forall x \cdot c(x) \neq c(n(x))$

WITNESSES

$f : f = c$

THEN

$act1 : col := c$

END

EVENT rule

STATUS convergent

ANY

x, clr

WHERE

$grd1 : c(x) = c(n(x)) \vee c(x) = c(n^{-1}(x))$

$grd2 : clr \neq c(n(x))$

$grd3 : clr \neq c(n^{-1}(x))$

THEN

$act1 : c(x) := clr$

END

La terminaison de ce processus dépend de la condition appelée variant et qui doit décroître par application des règles.

VARIANT $card(\{x|c(x) = c(n(x))\})$

TLA$^+$

Les graphes VISIDIA sont symétriques ; le graphe g est défini à partir du graphe n et la règle $rule$ est raffinée en $visidia{-}rule$

CONTEXT $ringgr$
EXTENDS $ring1$
CONSTANTS
g
AXIOMS
$axm1 : g = n \cup n^{-1}$
END

EVENT INITIALISATION

$act1 : col, c : |(col' \in N \to color \wedge c' \in N \to color \wedge col' = c')$

EVENT stable REFINES stable

WHEN

$grd1 : \forall x \cdot c(x) \notin c[g[\{x\}]]$

THEN

$act1 : col := c$

END

EVENT rule

STATUS convergent

REFINES rule

ANY

x, clr

WHERE

$grd1 : c(x) \in c[g[\{x\}]]$

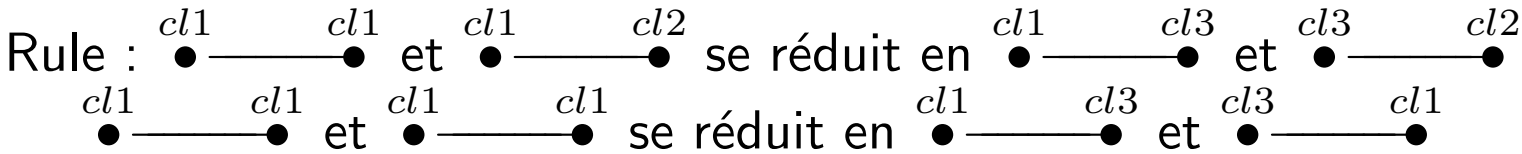
$grd2 : clr \notin c[g[\{x\}]]$

THEN

$act1 : c(x) := clr$

END

Dérivation d'un programme VISIDIA



Sommaire sur l'autostabilisation

- Définition d'algorithmes indépendants des états initiaux
- Algorithmes très complexes à vérifier en particulier *la terminaison*
- Version autostabilisante de beaucoup d'algorithmes

Terminaison de l'algorithme

- stabilité : un seul processus a la main ou un seul $i \in 0..N \wedge v[i] = v[i +_N 1]$.
- propriété 1 : il y a au moins un processus avec une garde franchissable :
  - tous les processus de 1 à N ont la même valeur (hypothèse)
  - nécessairement 0 et N ont la même valeur et 0 a la main.
- propriété 2 : toute configuration légale satisfait la propriété de clôture :
  - les processus de 0 à i-1 ont la même valeur
  - les processus de i à N ont la même valeur
  - $v[i] \neq v[i-1]$
- propriété 3 : à partir de toute configuration illégale, l'anneau atteint une configuration légale (le nombre de processus ayant la main ne peut pas croître et au mieux il peut stagner)
  - $0 \ldots i-1$: non franchissable
  - chaque processus maintient le nombre de processus franchissables : on construit à partir de p, une suite w1 ... wN différent deux à deux
  - cette suite ne peut pas rester indéfiniment stable pr propagation de la valeur et respect de la constnace.

Conclusion

- Algorithmes répartis : problème de l'expression locale de la globalité
- Algorithmes très complexes à vérifier
- Prise en compte de nombreux aspects de l'environnement comme les fautes, les erreurs, ...