INFORMATIQUE
Sciences
Université Paris Cité

Introduction à l'Auto-stabilisation

M2

Lélia Blin

lelia.blin@irif.fr

2024

Lélia Blin

Systèmes distribués

Réseaux téléphonique

Système biologique

IOT

BD

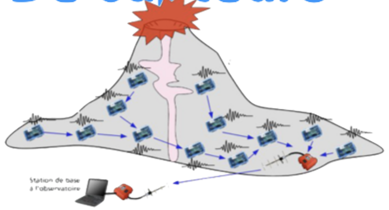
Cloud

P2P

Internet

Réseaux De capteurs

Colonie de robots



Caractérisation des systèmes distribués

- Entités informatiques autonomes interconnectées.
- Interaction par un moyen de communication.
- Absence de coordinateur.

Objectif du système distribué

Collaboration des entités pour réaliser des tâches communes:

- Structure couvrante (arbres, Steiner)
- Tables de routage
- Exclusion mutuelle
- ...

Vocabulaire

- Entités : processus ou **nodes**,
- Interconnecté : organisation **réseau**,
- Tâche globale : **algorithme distribué** (ou protocole)

Collaboration : échange d'informations

> Un nœud peut transmettre et recevoir des **informations** d'un autre nœud.

On suppose ici que les échanges d'informations sont bidirectionnels : un nœud v peut obtenir des informations du nœud u si et seulement si u peut obtenir des informations de v .

Lélia Blin

Modélisation du réseau

> Graphe non dirigé $G = (V, E)$
>
> - V est l'ensemble des nœuds
> - E est l'ensemble des arêtes

Si l'arête $\{v, u\}$ existe, alors v et u peuvent échanger directement des informations.

Caractérisation des nœuds

- Chaque nœud possède :
    - Un **identifiant**, ou non
        - Un réseau anonyme ou identifié
    - Sa propre **mémoire**
        - Mémoire non corruptible (identifiant, code, ...)
        - Une mémoire corruptible (contenu des variables)
    - Sa propre puissance de calcul
    - Sa propre horloge

Configurations

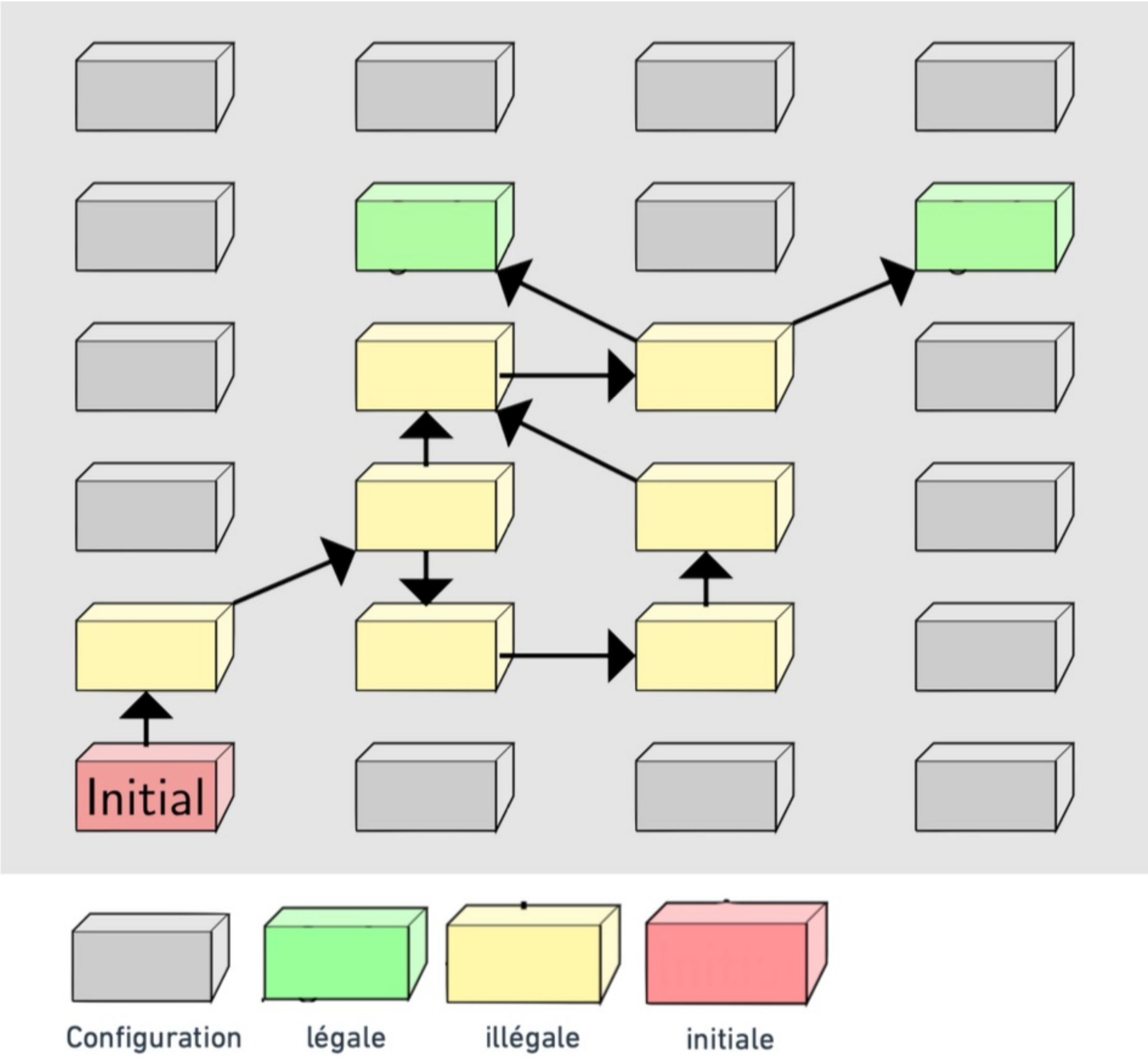
État du nœud : C'est l'ensemble contenant les valeurs de toutes ses variables.

> Pour un graphe $G(V, E)$, une **configuration** γ représente les états de tous les noeuds à un instant t.

L'ensemble de toutes les configurations est désigné par Γ.

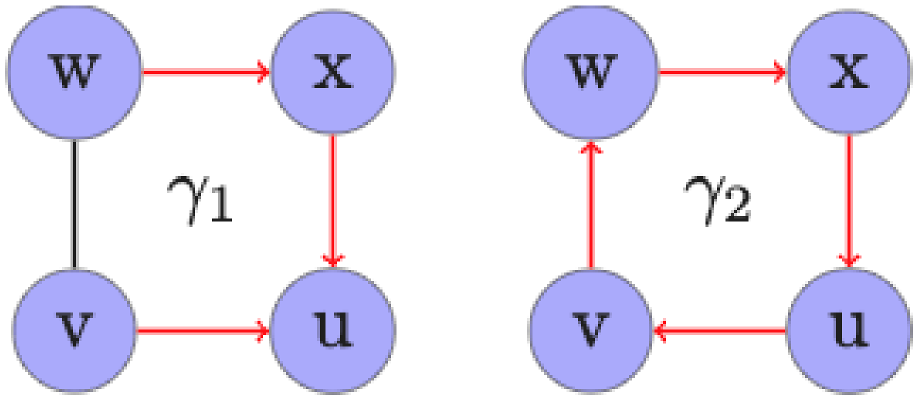
Configurations légitimes et illégitimes

- Soit $\mathcal{T}$ la tâche à résoudre par l'algorithme distribué $\mathcal{A}$.
  - Si la configuration γ correspond à $\mathcal{T}$, alors γ est appelée une configuration **légitime** ; sinon, elle est appelée **illegitime**.
- Note : Les termes "légal" et "illégal" sont également utilisés de manière interchangeable avec les termes "légitime" et "illégitime".


Initial
Configuration
légale
illégale
initiale

Exemple

- Arbre couvrant $\mathcal{T}$ (Spanning Tree)
- Variable locale p pour le parent
- γ_1 configuration légitime
- $\{p_u = \emptyset, p_v = u, p_w = x, p_x = u\}$
- γ_2 configuration illégitime
- $\{p_u = v, p_v = w, p_w = x, p_x = u\}$



Les systèmes distribués d'aujourd'hui

- Explosion combinatoire du nombre d'entités de calcul
- Hétérogénéité :
  - entités de calcul
  - support de communication

Conséquences

- Systèmes difficiles à initialiser
- Emergence de fautes

Type de défaillances

En général, on distingue trois types de défaillances possibles :.

1. **Défaillances transitoires** : Des défaillances de nature arbitraire peuvent frapper le système, mais il existe un moment dans l'exécution où ces défaillances n'apparaissent plus.
2. **Défaillances permanentes** : Des défaillances de nature arbitraire peuvent frapper le système, mais il existe un moment dans l'exécution où ces défaillances entraînent une incapacité permanente pour ceux qui en sont victimes.
3. **Défaillances intermittentes** : Des défaillances de nature arbitraire peuvent frapper le système à n'importe quel moment de l'exécution.

Tolérance aux pannes

Algorithmes robustes

- **Exploiter la redondance:** Utiliser plusieurs couches de redondance dans :
  - l'information (les noeuds)
  - les communications
- **Objectif** : garantir la sécurité de l'exécution du code en procédant à des vérifications croisées et à des validations approfondies.
- **Hypothèse sous-jacente** : le système est conçu pour résister à un nombre limité de défaillances et vise toujours à préserver une majorité d'éléments corrects, même en cas de défaillances plus graves.
- **Caractéristique** : de tels algorithmes sont typiquement des algorithmes masquants.

Algorithmes auto-stabilisants

- **Hypothèse:** Les défaillances sont transitoires (limitées dans le temps), mais il n'y a pas de contrainte sur l'étendue des défaillances qui peuvent affecter tous les éléments du système.
- **Un algorithme est considéré comme auto-stabilisant s'il peut atteindre une configuration légitime en un temps fini et rester dans des configurations légitimes, quelle que soit la configuration initiale.**
- **Caractéristique:** Typiquement non masquant. Entre la détection des fautes et la stabilisation du système vers un comportement correct, l'exécution peut être quelque peu erratique.

Algorithmes robustes

Avantages:

- S'alignent intuitivement sur la tolérance aux fautes.
- Redondance : Remplacer chaque élément par trois éléments identiques pour une meilleure fiabilité.
- Les actions sont décidées par consensus majoritaire pour un comportement fiable.

Algorithmes robustes

Inconvénients

- La triple redondance peut entraîner une inefficacité des ressources et une augmentation des coûts.
- Peut ne pas gérer les états arbitraires résultant de défaillances aussi efficacement que les algorithmes auto-stabilisants.

Algorithmes auto-stabilisants

Avantages:

- Lié au concept de convergence ; recherche un point fixe quelle que soit la position initiale.
- Peut démarrer à partir d'une configuration arbitraire.
- Capables de se comporter correctement en un temps fini, même s'ils partent d'une configuration inconnue.
- Utilisé dans de nombreux protocoles de réseaux informatiques.

Algorithmes auto-stabilisants

Inconvénients

- Le fonctionnement à partir d'un état arbitraire peut être contre-intuitif dans certains contextes.
- Possibilité de comportement erratique avant d'atteindre la stabilisation.

Concept d'auto-stabilisation

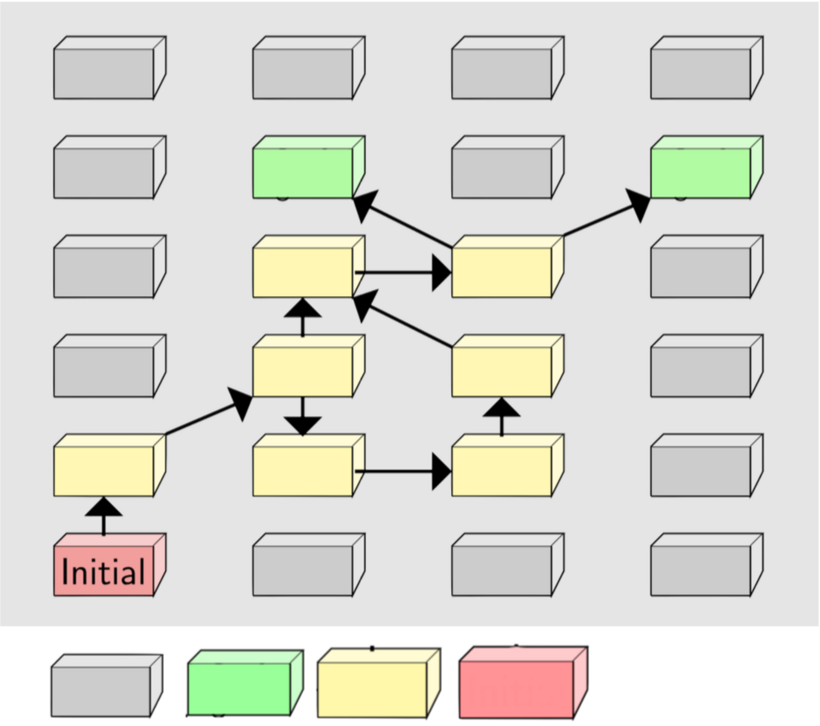
- Introduit par Dijkstra en 1974
- **Edsger Wybe Dijkstra**
  - 1930-2002
  - Informaticien
  - Prix Turing 1972
  - Prix Dijkstra

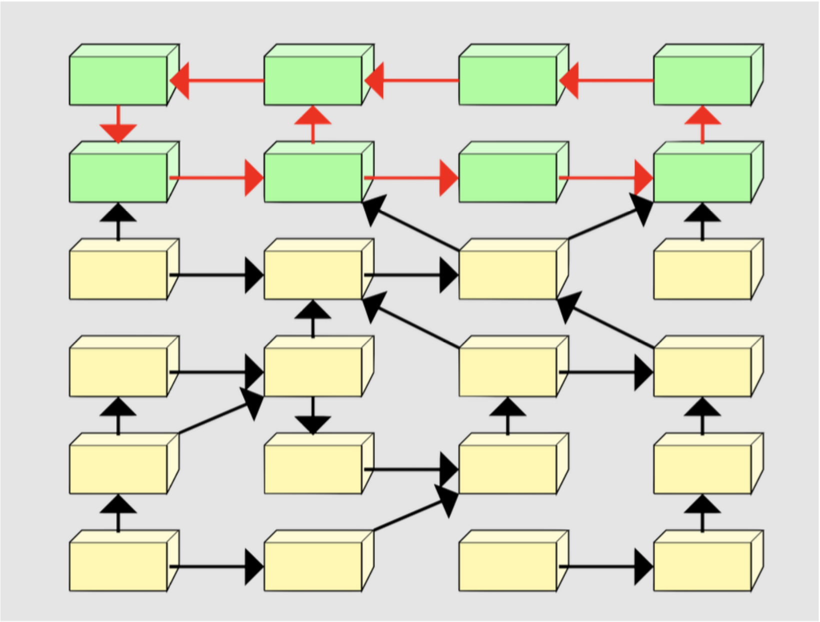
Définition

Un algorithme auto-stabilisant qui résout la tâche $\mathcal{T}$ est un algorithme distribué $\mathcal{A}$ satisfaisant :

1. **Convergence** : En partant d'une configuration arbitraire, le système atteint une configuration légitime en utilisant l'algorithme $\mathcal{A}$.
2. **Clôture** : A partir d'une configuration légitime, le système reste dans une configuration légitime.

Algorithms distribués classiques vs Auto-stabilisation





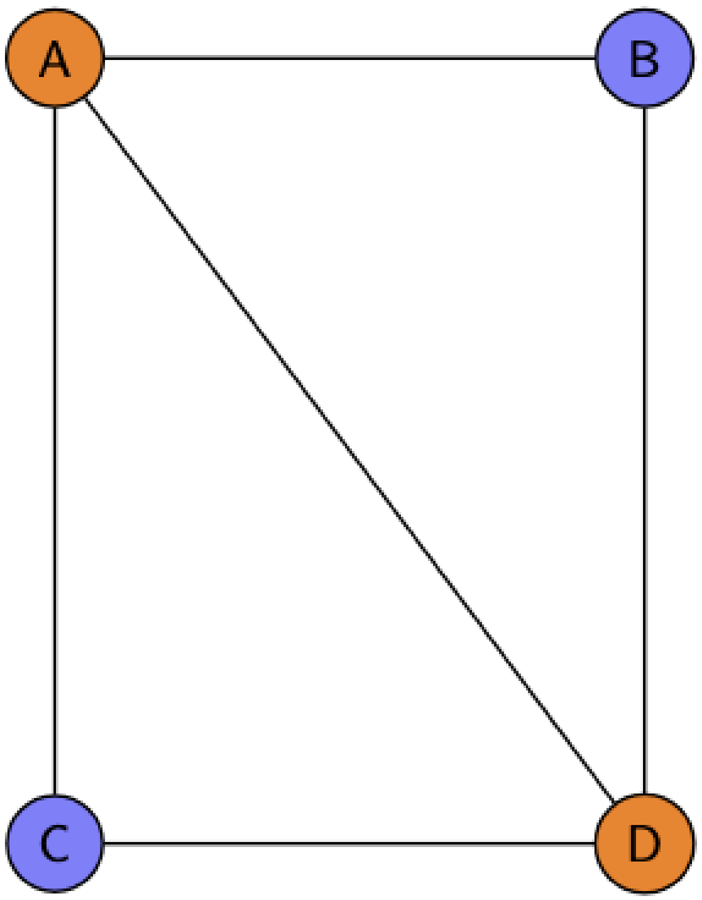
1. Détecter localement les configurations illégales/légales.
2. revenir à une configuration légale.

Détection locale des configurations illégitimes

Un algorithme auto-stabilisant doit non seulement **construire une solution** mais aussi fournir un mécanisme pour **vérifier** cette solution.

Détection locale des configurations illégitimes

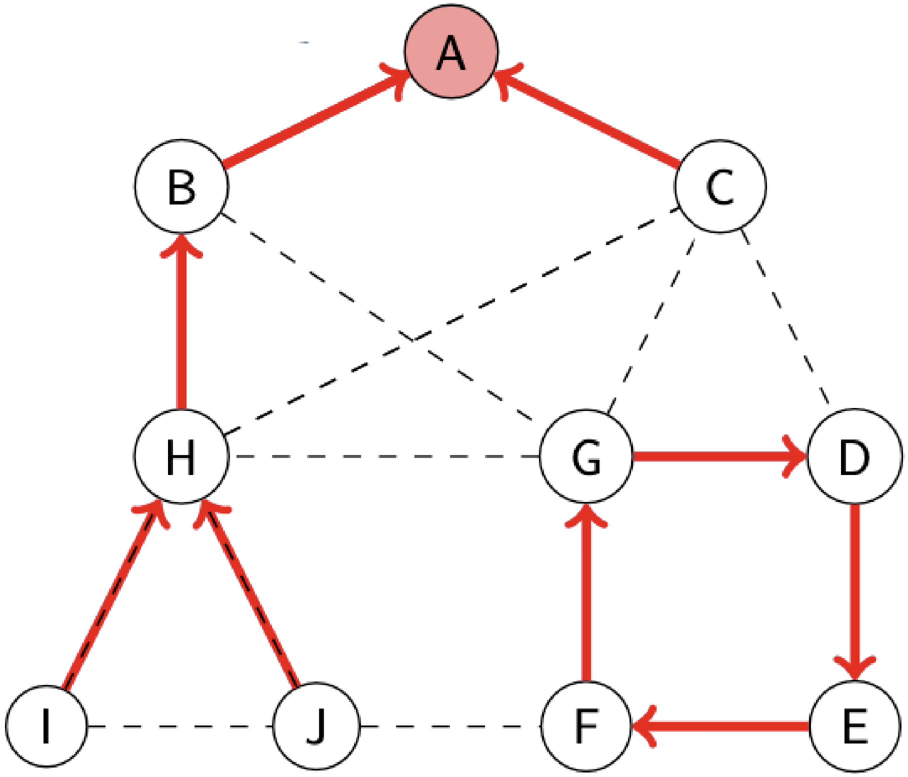
Un algorithme auto-stabilisant doit non seulement **construire une solution** mais aussi fournir un mécanisme pour **vérifier** cette solution.



Coloration

Détection locale des configurations illégitimes

Un algorithme auto-stabilisant doit non seulement **construire une solution** mais aussi fournir un mécanisme pour **vérifier** cette solution.



Spanning Tree

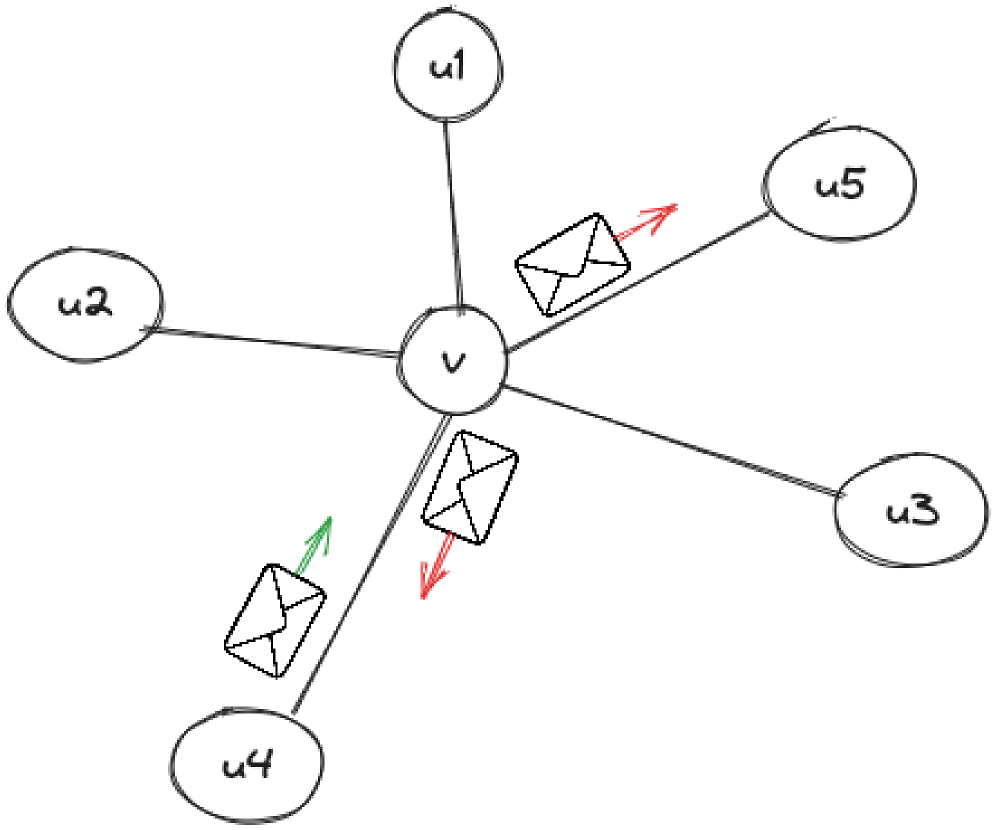
Modèle

Modèle informatique

- 3 modèles principaux de la littérature
- L'étape atomique : l'unité fondamentale

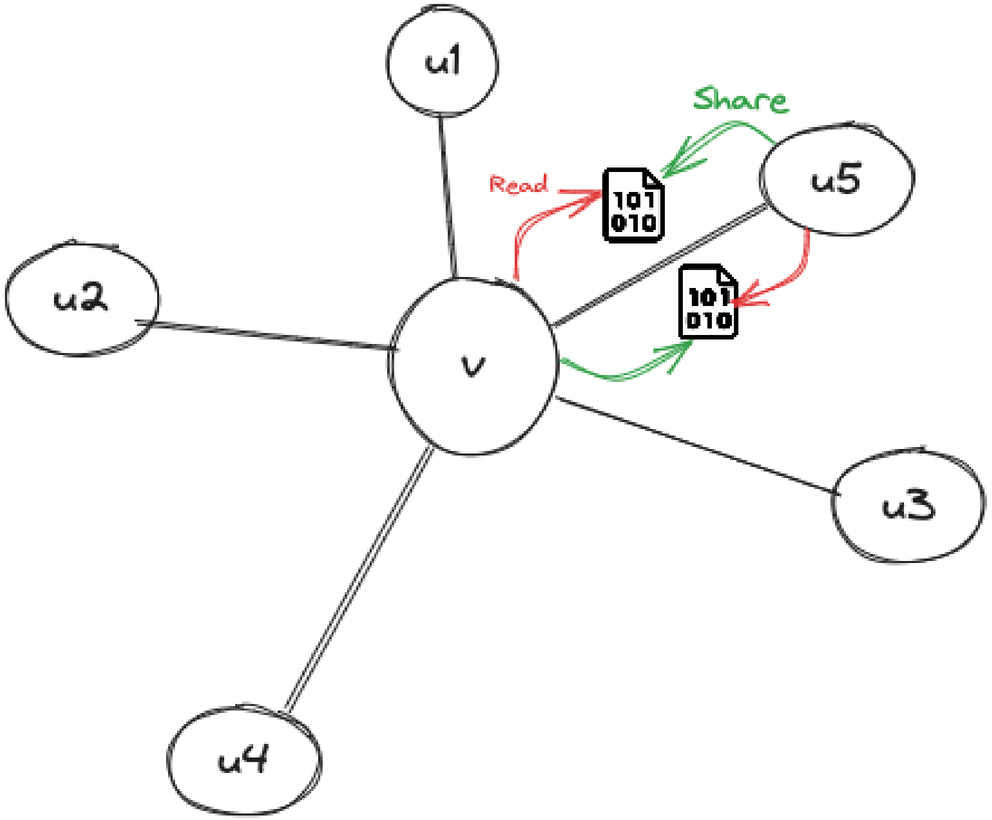
Passage de messages

En une seule étape atomique, un nœud envoie un message à un nœud voisin ou reçoit un message d'un nœud voisin, mais pas les deux simultanément.



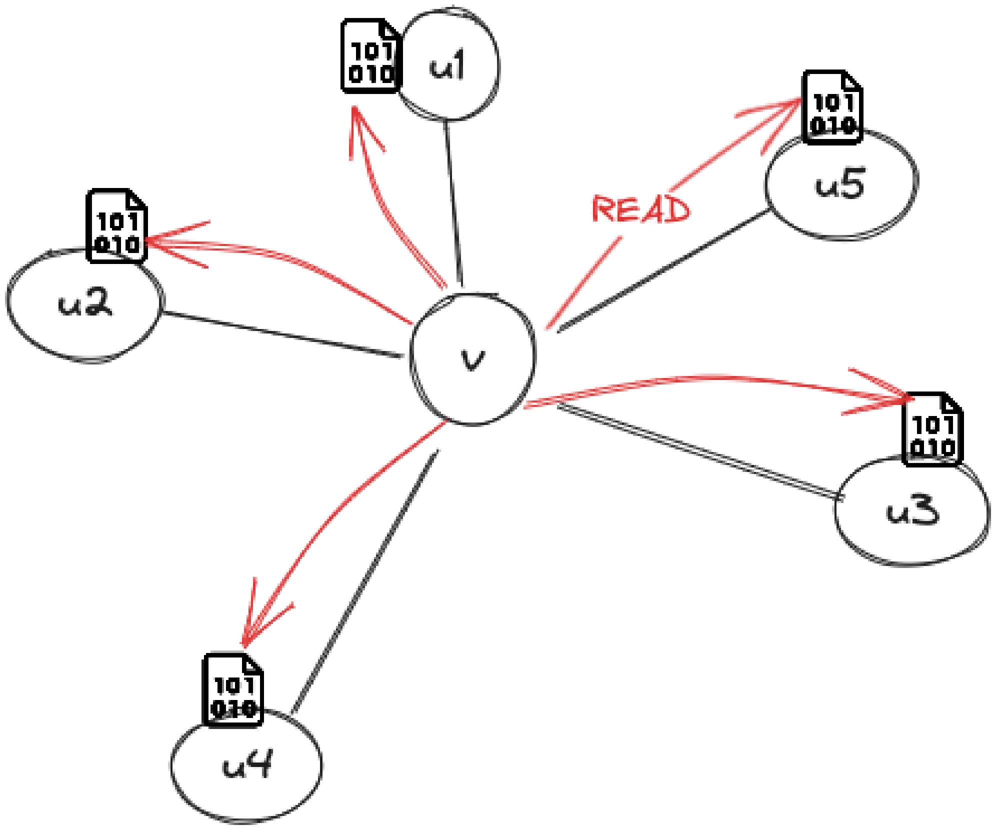
Registre partagé

En une seule étape atomique, un nœud peut soit lire l'état d'un nœud voisin, soit mettre à jour son propre état, mais pas les deux simultanément.



Mémoire partagée

En une seule étape atomique, un nœud peut lire l'état de chaque nœud voisin et mettre à jour son propre état.



Algorithme d'auto-stabilisation

$$\langle label \rangle : \langle guard \rangle \longrightarrow \langle command \rangle$$

- Les **étiquettes** ne sont utilisées que pour identifier les actions dans le raisonnement.
- Un **guard** est un prédicat booléen sur les variables du noeud.
- Une **commande** est un ensemble d'affectations de variables.

Nœuds activés

- Une action ne peut être exécutée que si sa garde est évaluée à true ;
- dans ce cas, on dit que l'action est **activée**.
- Par extension, un noeud est dit **activé** si au moins une de ses actions est activée.

Asynchronisme

- Les exécutions sont pilotées par un adversaire non déterministe qui modélise l'asynchronisme du système.
- Cet adversaire est appelé **daemon** ou **scheduler**.

Étape de calcul

- Notez qu'à tout moment, bien que l'ordonnanceur puisse choisir un sous-ensemble de nœuds activés, il doit en choisir au moins un.
- Après l'activation atomique de tous les nœuds activés choisis par l'ordonnanceur, une nouvelle configuration est obtenue.

Équité

- L'équité permet de réguler le taux d'exécution relatif des processus en prenant en compte les actions passées.
- Il s'agit d'une propriété de pérennité au sens d'Alpern et Schneider [AS85].

Hypothèses sur l'équité

Les trois hypothèses d'équité les plus populaires de la littérature.

Un ordonnanceur est

- **fortement équitable**, s'il active infiniment souvent tous les processus qui sont activés infiniment souvent.
- **faiblement équitable**, s'il active finalement tous les processus continuellement activés.
- **unfair** , n'a pas de contrainte d'équité
  - Il se peut qu'il ne sélectionne jamais un processus à moins qu'il ne soit le seul à être activé.

Complexité

Complexité temporelle

Les principales unités de mesure sont utilisées dans le modèle de l'état atomique :

- La complexité en **rondes** évalue le temps d'exécution en fonction de la vitesse des processus les plus lents
- La complexité en **mouvements** ou **étapes** saisit la quantité de calculs nécessaires à un algorithme.

Complexité de l'espace

Premier algorithme auto-stabilisant

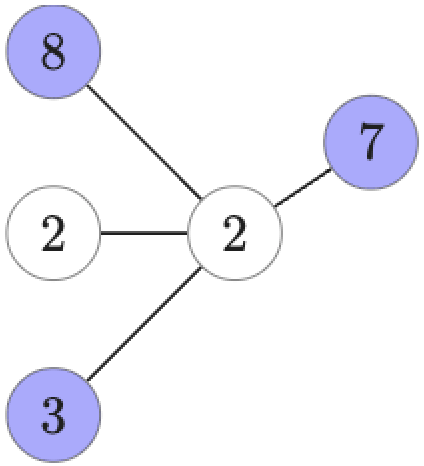
Exemple : Accord sur la même valeur

Accord sur la même valeur

Variable locale : val_v est un entier positif $\forall v \in V$

Algorithme

$\exists u \in N(v))|val_u < val_v \to val_v := \min\{val_u | u \in N(v)\}$



Preuve de validités

- Soit notée m la valeur minimale $m = \min\{val_v : \forall v \in V\}$.
- Soit ψ :$\Gamma \times V \to \mathbb{N}$ la fonction définie par :
  $\psi(\gamma, v) = val_v - m$.
- Soit Ψ : $\Gamma \to \mathbb{N}$ soit la fonction définie par :

$$\Psi(\gamma) = \sum_{v \in V} \psi(\gamma, v)$$

- Et définissons par γ_{ell} l'ensemble des configurations légales
  - $\Gamma_\ell = \{\gamma \in \Gamma : \Psi(\gamma) = 0\}$

Convergence du théorème : $true \triangleright \Gamma_\ell$

Preuve : Soit γ_0 une configuration illégale arbitraire, $\mathcal{E}(\gamma)$ l'ensemble des noeuds activés et $\mathcal{S}(\gamma)$ l'ensemble des noeuds activés choisis par l'ordonnanceur.

Pour tout v dans $\mathcal{S}(\gamma)$ après l'exécution de l'algorithme par le nœud v nous obtenons $val_v(\gamma') < val_v(\gamma_0)$ donc $\psi(\gamma', v) < \psi(\gamma_0, v)$ et $\Psi(\gamma) < \Psi(\gamma_0)$.

Fermeture du théorème : Γ_ℓ est fermé

Preuve : Soit $\gamma \in \Gamma_\ell$, alors $\forall v \in V$ dans γ nous avons $\psi(\gamma, v) = 0$ et $val_v = m$ en conséquence $\mathcal{E}(\gamma) = \emptyset$.

Complexité temporelle

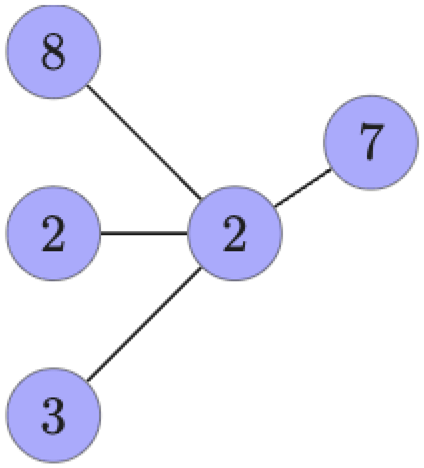
- $O(n)$ rounds (Quel ordonnanceur ?)
- $0(n^2)$ étapes (Quel ordonnanceur ?)

Variante : Accord sur la même valeur

Variable locale : val_v est un entier positif $\forall v \in V$

Algorithme

$\exists u \in N(v))|val_u \leq val_v \to val_v := \min\{val_u|u \in N(v)\}$



Propriété silencieuse

> Un algorithme auto-stabilisant est dit "silencieux" si, une fois qu'une configuration légale est atteinte, il reste dans cette même configuration légale.

Autostabilisation : Conclusion

Pour

- Le réseau n'a pas besoin d'être initialisé
- Lorsqu'un défaut est diagnostiqué, il suffit d'identifier, puis d'enlever ou de redémarrer les composants défectueux
- La propriété d'auto-stabilisation ne dépend pas de la nature du défaut
- La propriété d'auto-stabilisation ne dépend pas de l'étendue du défaut

Autostabilisation

Cons

- A priori, "éventuellement" ne donne pas de limite au temps de stabilisation
- A priori, les noeuds ne savent jamais si le système est stabilisé ou non
- Une seule défaillance peut déclencher une action corrective à chaque nœud du réseau
- Les défaillances doivent être suffisamment rares pour être considérées comme transitoires.

Ressources

- Auto-stabilisation** , Shlomi Dolev MIT Press 2000
- **Introduction to Distributed Self-Stabilizing Algorithms**, Karine Altisen, Séphane Devisme, Swan Dubois, Franck Petit, Synthesis Lectures on Distributed Computing Theory, Morgan & Claypool Publishers 2019
- https://pages.lip6.fr/Franck.Petit/enseign/master/HCMV/RAD/lectures/Self-Stabilization_4pages.pdf

La circulation de jeton dans l'anneau de Dijkstra

Le premier algorithme d'auto-stabilisation

Token circulation

Problème :

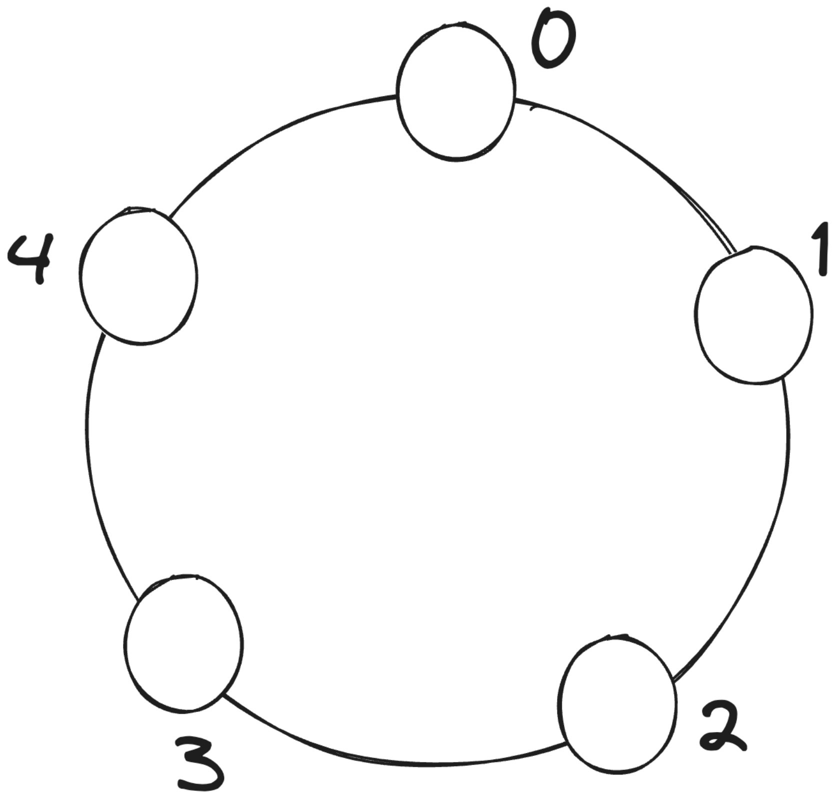
Il existe n nœuds dans le réseau dénotés par $v_0, \ldots, v_{n-1}$
Chaque noeud v_i peut détenir un jeton ou non

Objectif :

Jeton unique et le jeton visite chaque v_i infiniment souvent

Hypothèses

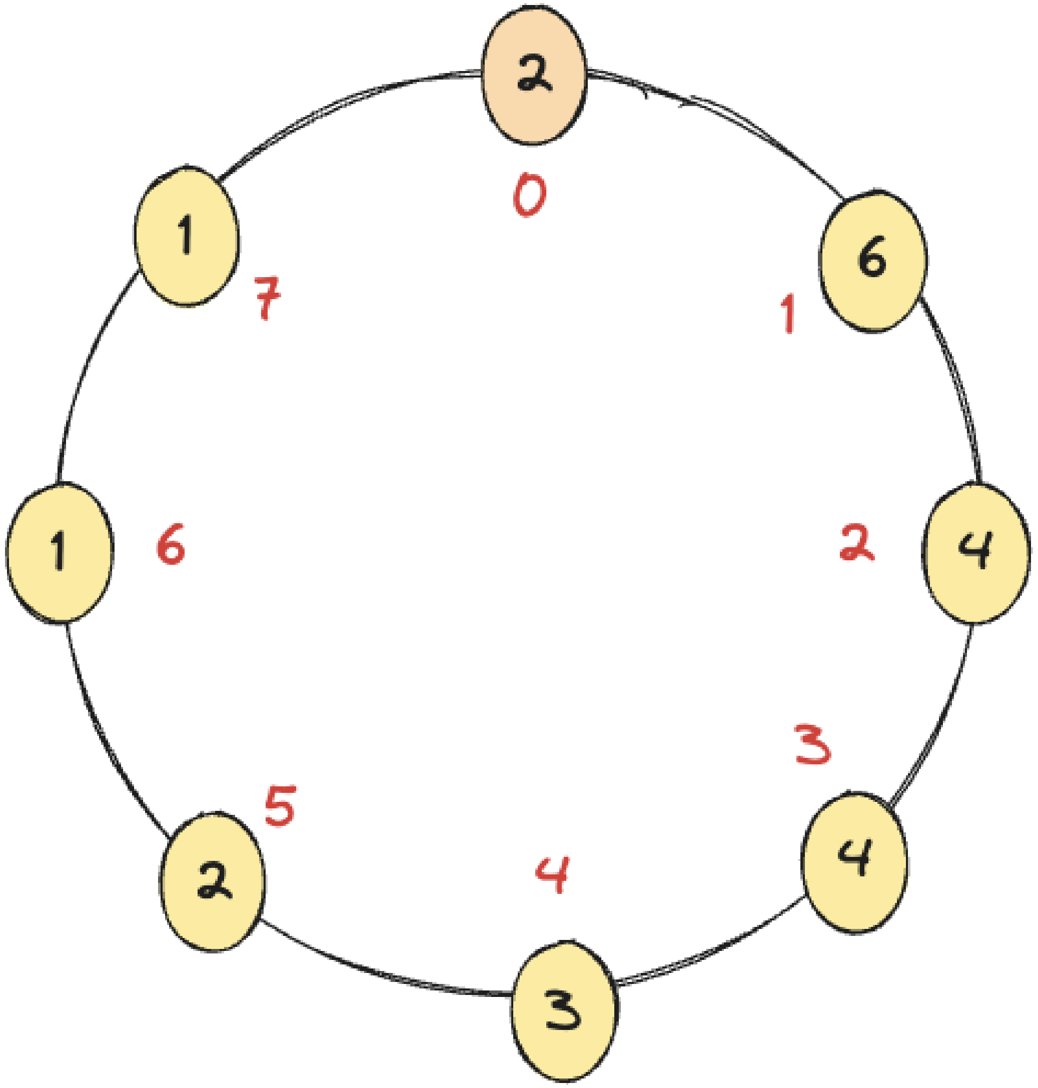
Les nœuds sont organisés en anneau.



État de chaque noeuds

> v state : $s_v \in 0, \ldots, n\}$

Remarque : l'état peut prendre $n+1$ valeurs différentes


2
0
6
1
4
2
4
3
3
4
2
5
1
6
1
7

Présence du jeton : vision globale

le nœud v_0 détient le jeton si $\forall v \in V : s_v = s_0$.

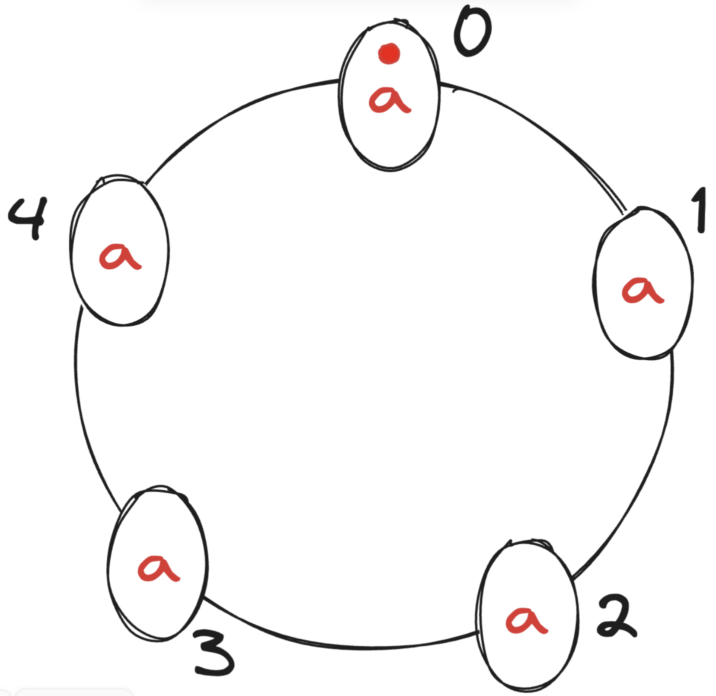
le noeud $v_i \in V \setminus \{v_0\}$ détient le jeton si

- $\forall j$ avec $j > i : s_j = s_i$
- $\forall j$ avec $j < i : s_j = s_0$

Configuration légale (L_0)

v_0 détient le jeton dans la configuration γ, si tous les noeuds ont le même état

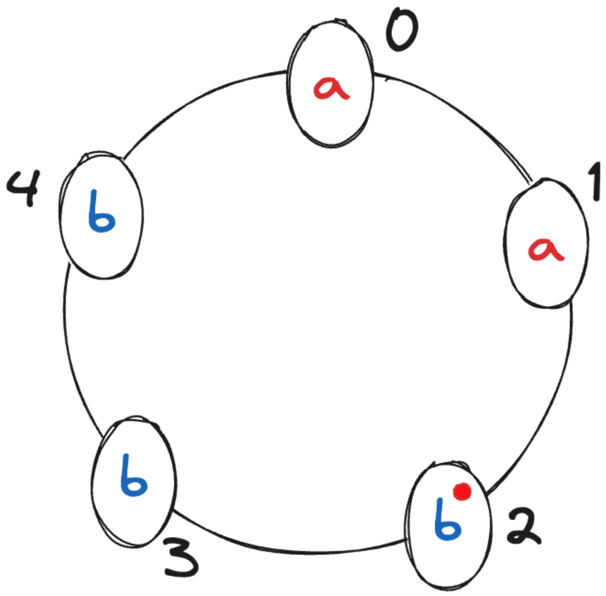
$$L_0(\gamma) \equiv (\forall i \neq 0 : s_i = s_0)$$



Configuration légale (L_i)

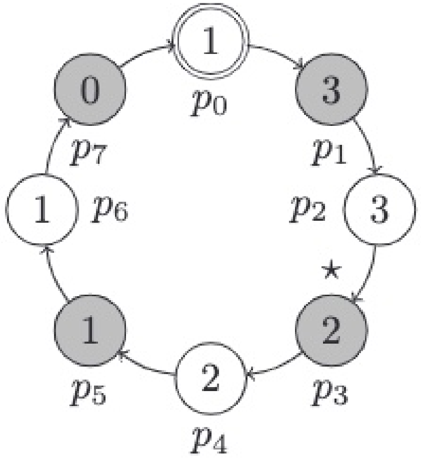
v_i avec $i \neq 0$ est un nœud unique avec le jeton dans la configuration γ, si

$$L_i(\gamma) \equiv \forall j \in \{1, n-1\} : (j < i : s_j = s_0) \wedge (j > i : s_j = s_i)$$



Présence du jeton : vision locale

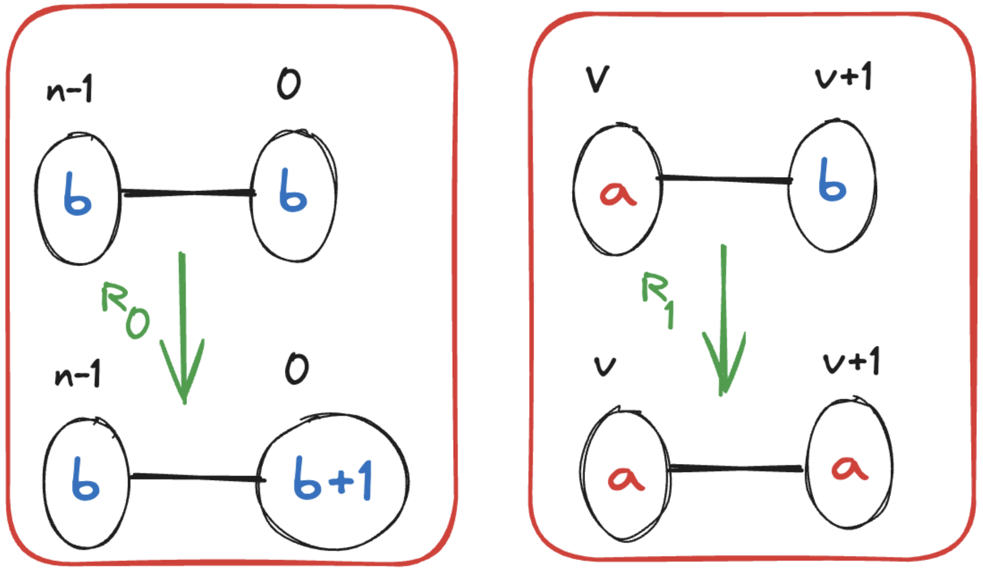
- Un noeud considère qu'il a le jeton si son successeur dans l'anneau à la même valeur que lui
- Exception: le dernier noeud sur l'anneau considère qu'il a le jeton si il a une valeur différente du premier noeud.



Configuration (i)

Algorithme

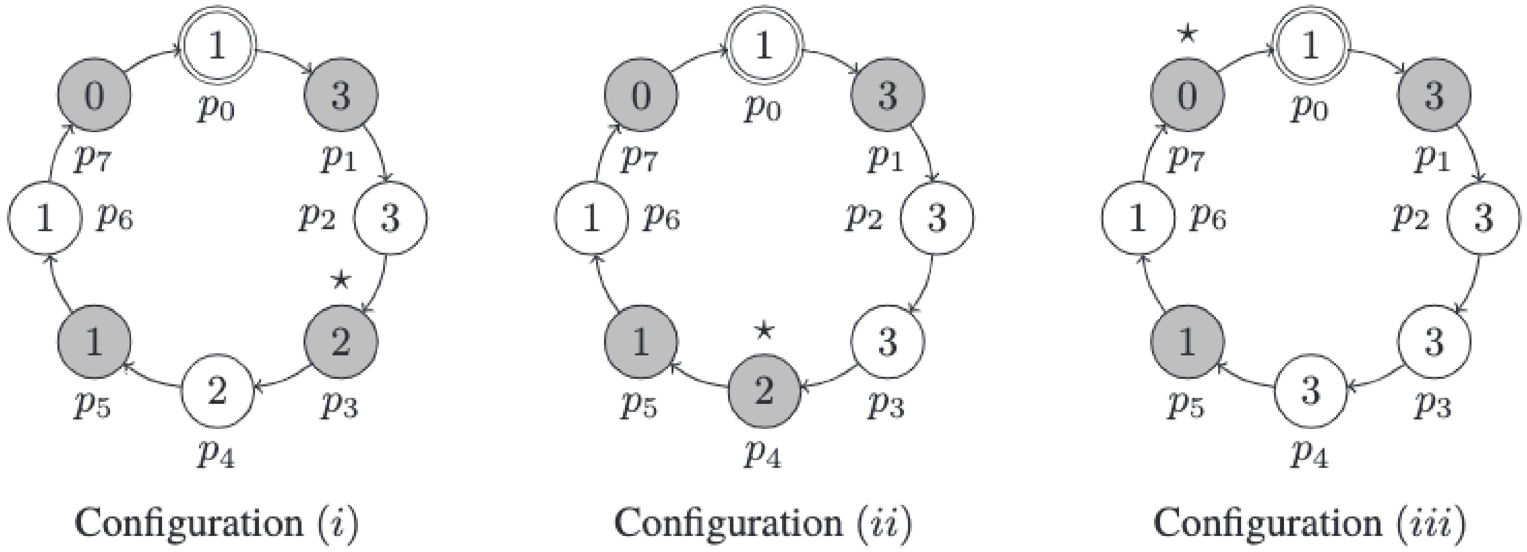
- $R_0 : (s_{n-1} = s_0) \longrightarrow s_0 := s_0 + 1 \mod n+1$
- $R_1 : (s_i \neq s_{i-1}) \wedge (i \neq 0) \longrightarrow s_i := s_{i-1}$

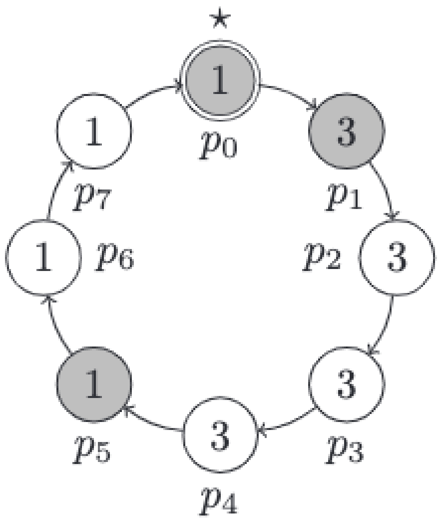


Exemple

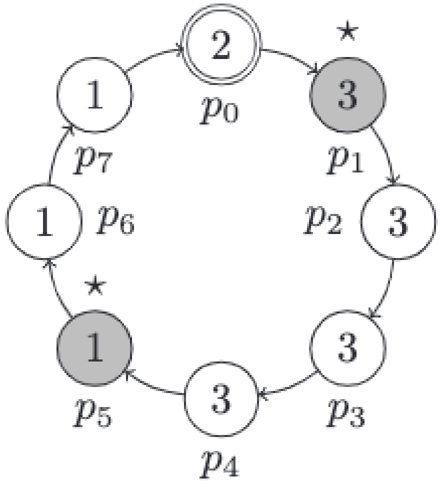
Les nœuds en gris ont un jeton

- $R_0 : (s_{n-1} = s_0) \longrightarrow s_0 := s_0 + 1 \mod n+1$
- $R_1 : (s_i \neq s_{i-1}) \wedge (i \neq 0) \longrightarrow s_i := s_{i-1}$

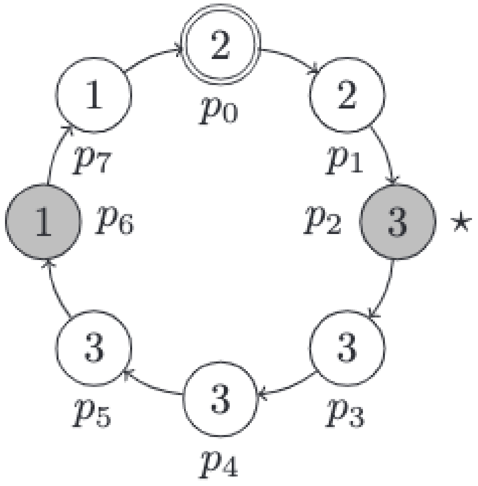




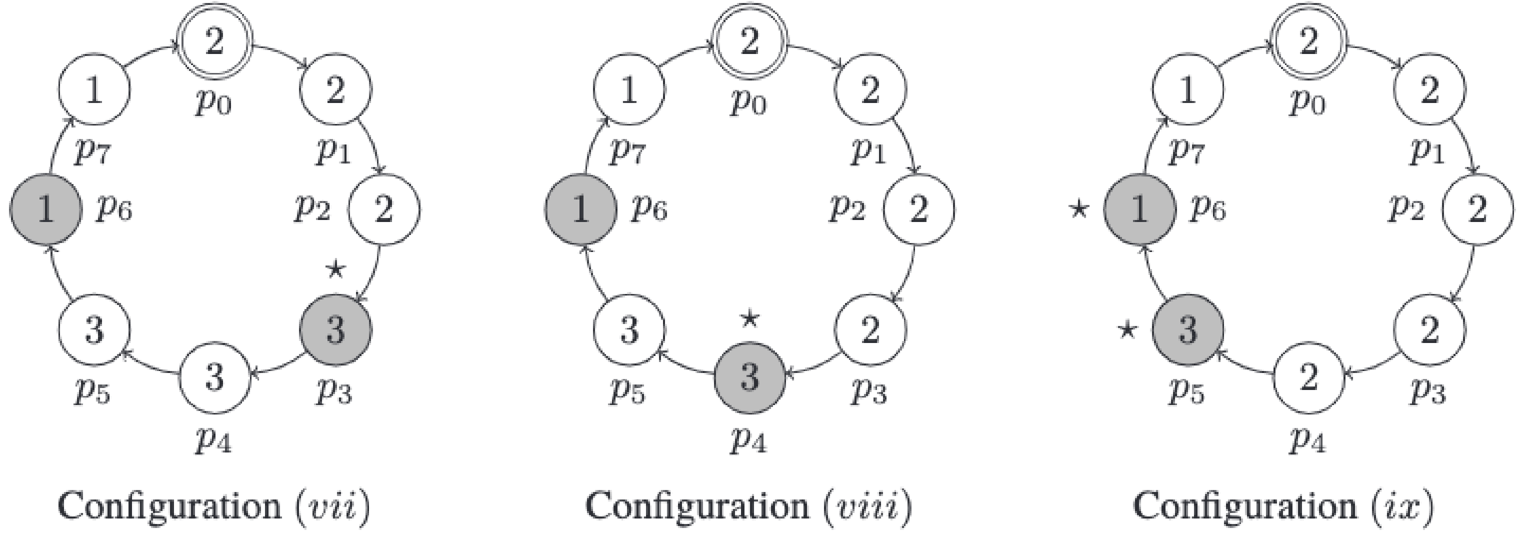
Configuration (iv)

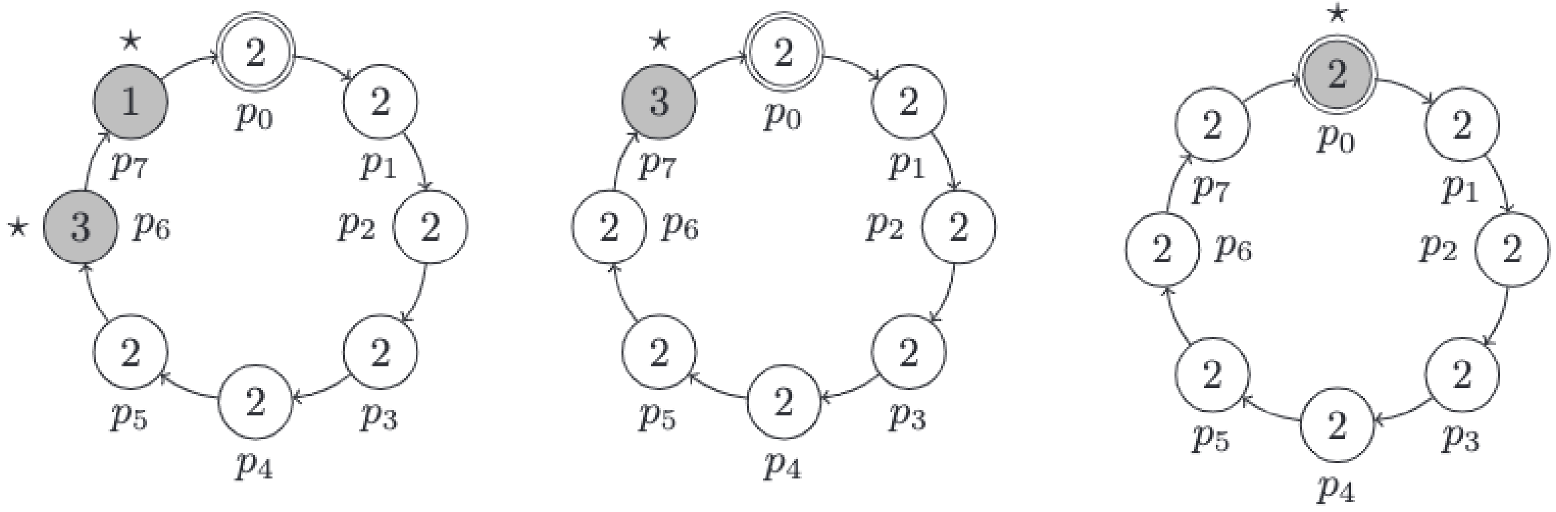


Configuration (v)



Configuration (vi)


2
p_0
1
p_7
2
p_1
1
p_6
p_2
2
★
3
p_5
3
p_4
3
p_3
Configuration (vii)
2
p_0
1
p_7
2
p_1
1
p_6
p_2
2
3
p_5
★
3
p_4
2
p_3
Configuration $(viii)$
2
p_0
1
p_7
2
p_1
★
1
p_6
p_2
2
★
3
p_5
2
p_4
2
p_3
Configuration (ix)



Configuration (x) Configuration (xi) Configuration (xii)

Clôture : preuve

1. Si v_i ne détient pas le jeton, alors v_i n'est pas activable et ne change pas d'état.
   → $\gamma' = \gamma$ et $\gamma' \in L$
2. Si v_i détient le jeton, alors v_i est activable :
   i. Si $i = 0$, v_0 augmente son état, et v_1 est le seul noeud à détenir le jeton, $\gamma' \in L$.
   ii. Si $i \neq 0$, v_i prend l'état de v_{i-1} , et v_i est le seul noeud à détenir le jeton, $\gamma' \in L$.

Convergence

> Théorème 2 : A partir d'une configuration $\gamma_0 \notin L$ le système converge vers une configuration $\gamma' \in L$

Notations

- $\mathcal{A}(\gamma)$ désigne l'ensemble des nœuds activables dans la configuration $\gamma \in \Gamma$.
- $\mathcal{A}^*(\gamma) \subset \mathcal{A}(\gamma)$ désigne l'ensemble des nœuds activés par l'ordonnanceur dans la configuration γ.
- $s_i(\gamma)$ l'état de v_i dans γ
- la configuration suivante : $\gamma \xrightarrow{\mathcal{A}^*(\gamma)} \gamma'$

Impasse (Deadlock)

lemma 1 : $\forall \gamma \in \Gamma : \mathcal{A}(\gamma)| \neq 0.$

Preuve par contradiction:

Supposons qu'il existe une configuration $\gamma \in \Gamma$ telle que $\mathcal{A}(\gamma)| = 0$, de sorte qu'aucun noeud ne peut exécuter une règle.

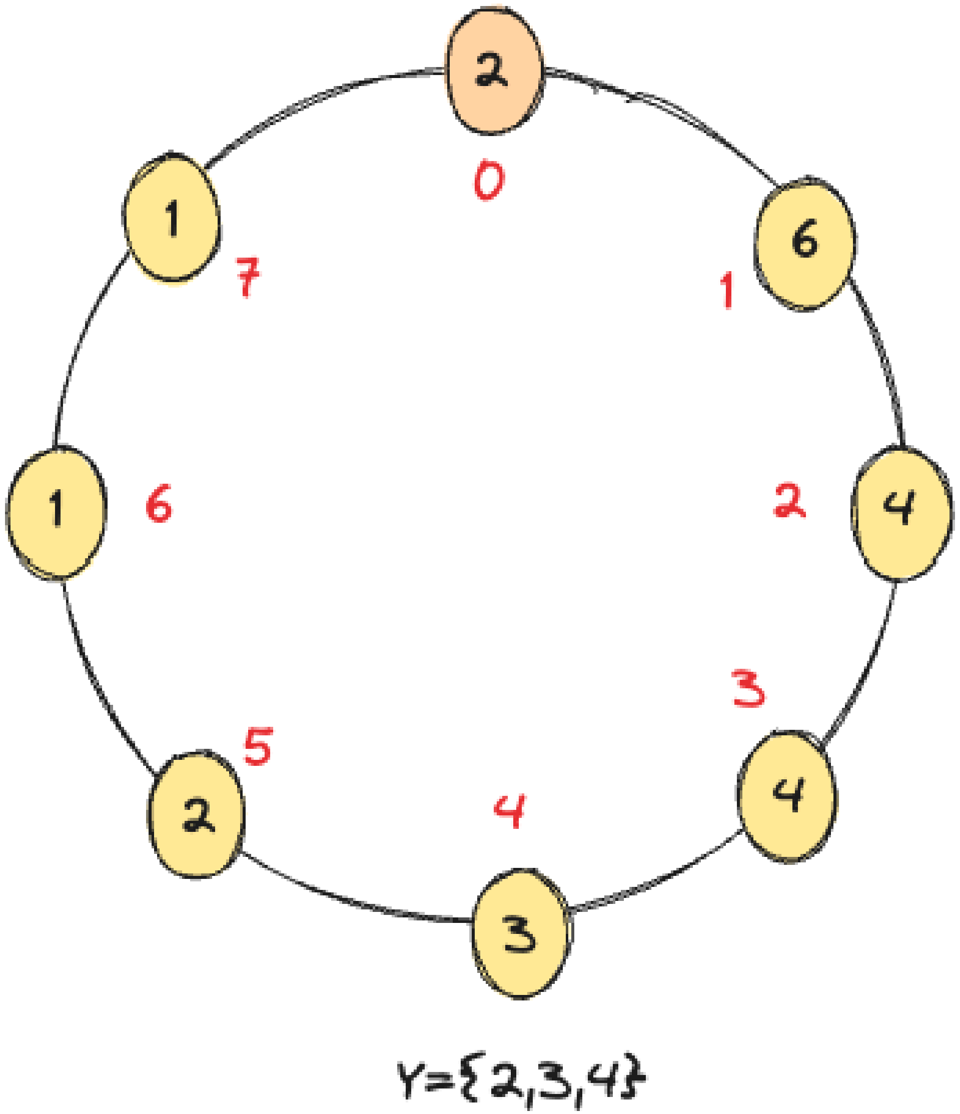
- Si $i \neq 0$, alors tous les noeuds ont la même valeur, y compris le noeud v_{n-1} et v_0 ; sinon, ils peuvent exécuter la règle R_1. Contradiction, v_0 peut exécuter la règle R_0.
- Si $i = 0$, v_0 ne doit pas avoir la même valeur que v_{n-1} ; sinon, v_0 peut exécuter la règle R_0, et $\forall i : 0 < i < n - 1, x_i = x_{n-1}$. Contradiction, v_1 peut exécuter la règle R_1.

Éléments de commutation

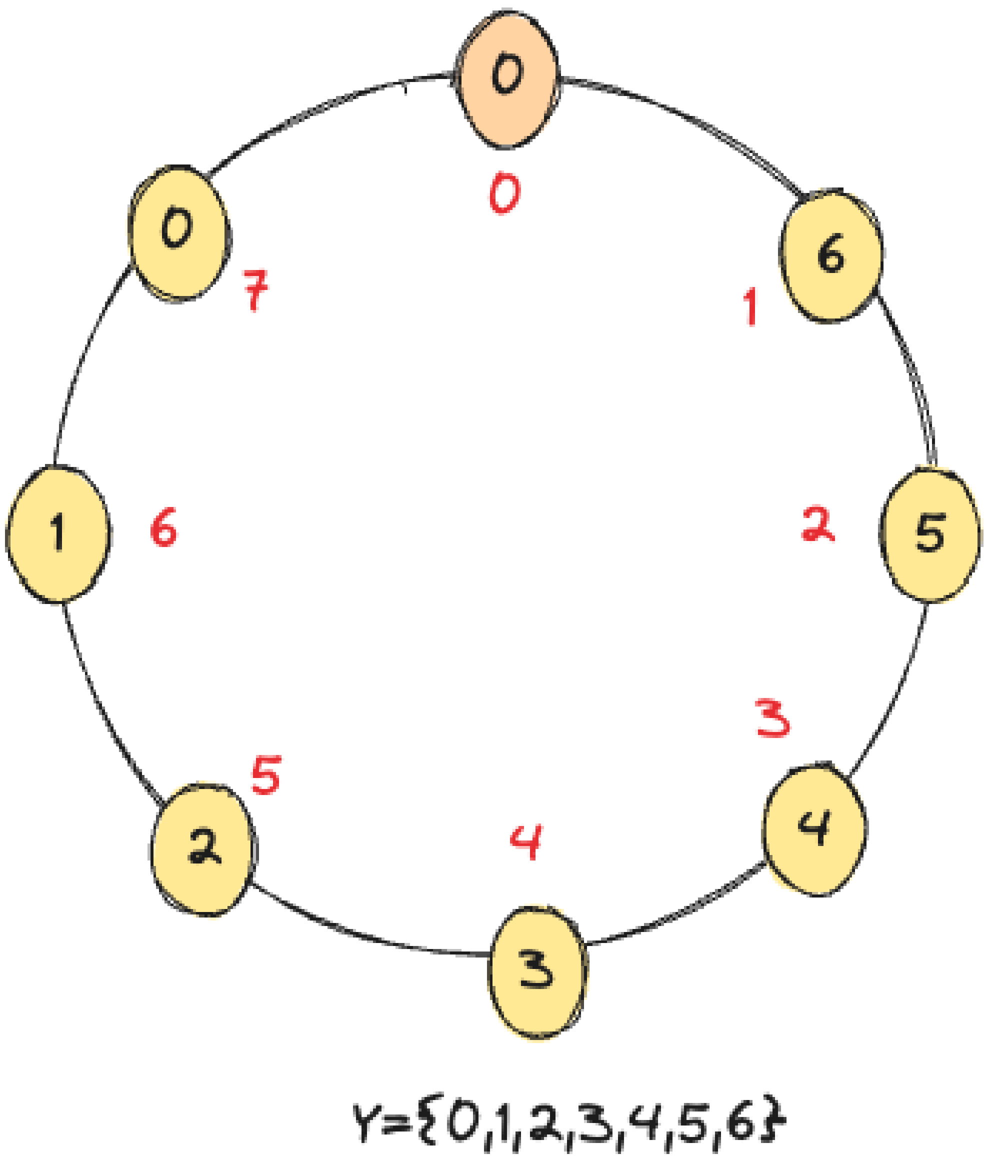
Nombre maximal d'activations de v_0.

> $x \in \{0, \ldots, n\}$ est valide si
>
> - $\exists i > 0 : s_i = x$ et
> - $(x = s_0)$ ou $\exists j > i | s_j = (x - 1 \mod n + 1)$ et s_j est valide

Désignons par $Y(\gamma)$ l'ensemble des éléments valides dans la configuration γ



- 2 est valide car $\exists i = 5 > 0 | s_5 = 2$ et $s_0 = 2$
- 3 (i=4) est valide car 2 (j=5) est valide et $j > i$
- 4 (i=3) est valide car 3 (j=4) est valide et $j > i$



Nombre maximum d'interrupteurs

lemma 2 : Si v_0 dans γ exécute R_0 alors $|Y(\gamma)| > |Y(\gamma')|$ avec $\gamma' > \gamma$

Proof :

- Si v_0 exécute R_0 on obtient $x_0(\gamma') = x_0(\gamma) + 1 \mod n + 1$.
- Par définition de l'élément valide $x_0(\gamma) \notin Y(\gamma')$
- Considérons $x_j(\gamma)\ dans Y(\gamma)$:

  i. soit $x_j(\gamma)$ disparaît par application de la règle R_1 par p_j.

  ii. soit $x_j(\gamma)$ demeure et reste valide.

  Donc $|Y(\gamma)| > |Y(\gamma')|$

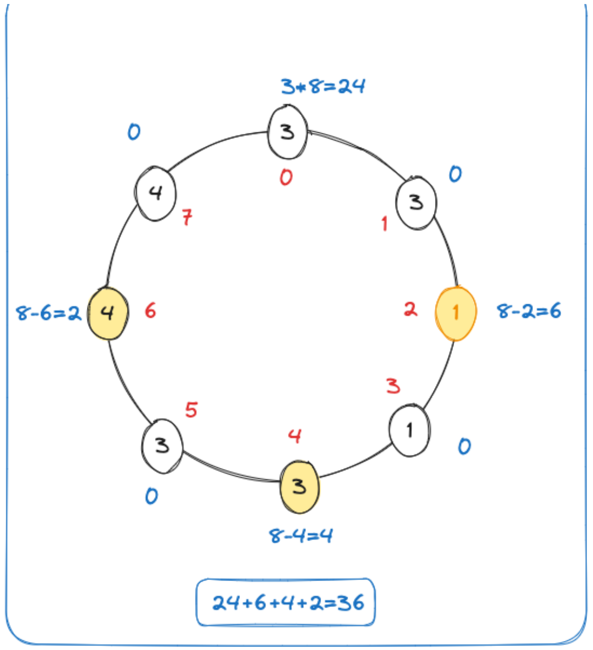
Fonction potentielle : Le poids d'un jeton

$\delta_i(\gamma) =$

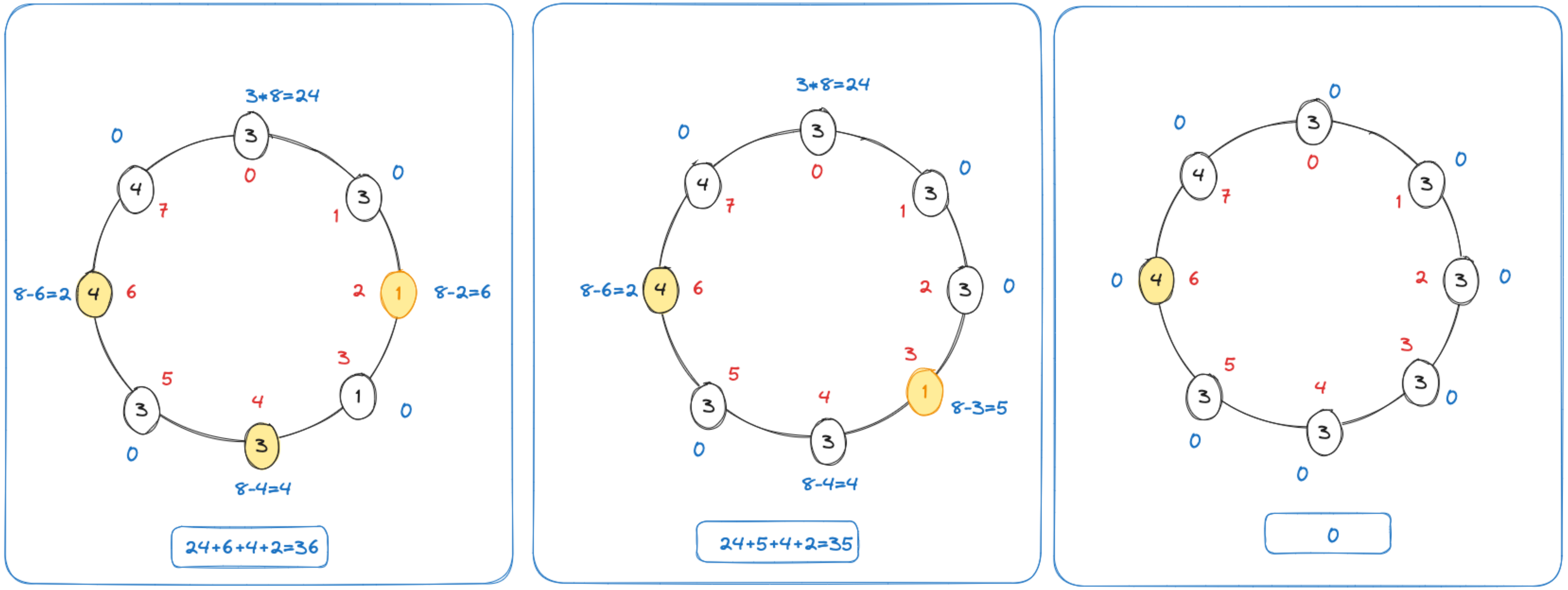
- $J * n$ si $i = 0 \wedge \neg L_0(\gamma) \wedge \neg L_i(\gamma)$ (Où J est le nombre de jetons)
- $n - i$ si $i \neq 0 \wedge \neg L_0(\gamma) \wedge \neg L_i(\gamma) \wedge (x_i \neq x_{i-1})$
- 0 sinon

$$\Delta(\gamma) = \sum_{i \in \{0,...,n-1\}} \delta_i(\gamma)$$

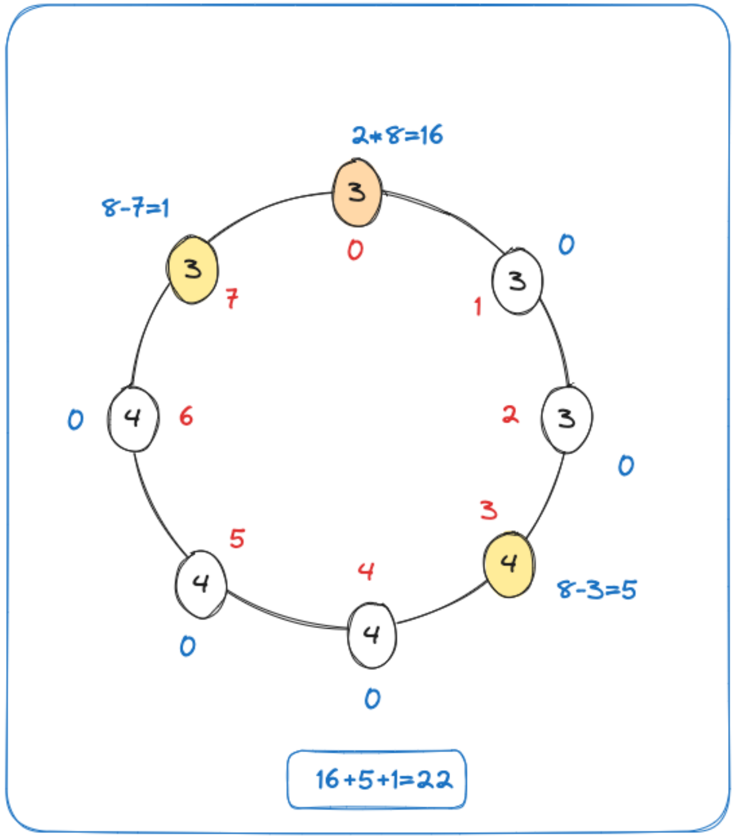
Remarque : Par construction, si $\gamma \in L$, nous avons $\Delta(\gamma) = 0$.

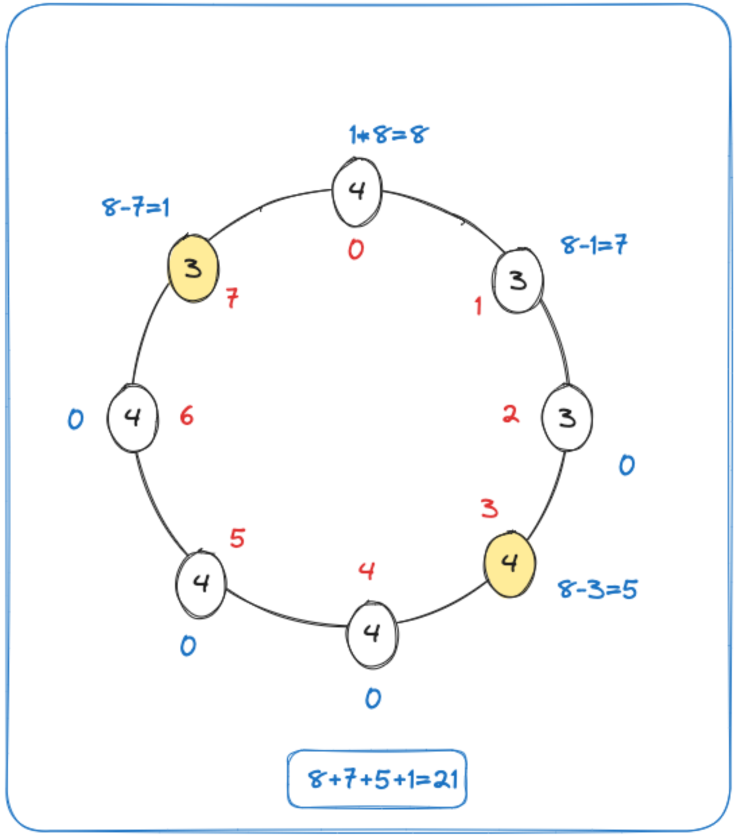


- $J * n$ si $i = 0 \wedge$
  - $\neg L_0(\gamma) \wedge \neg L_i(\gamma)$
- $n - i$ si $i \neq 0 \wedge$
  - $\neg L_0(\gamma) \wedge \neg L_i(\gamma) \wedge (x_i \neq x_{i-1})$
- 0 sinon


3∗8=24
8-6=2
8-2=6
8-4=4
24+6+4+2=36
3∗8=24
8-6=2
8-3=5
8-4=4
24+5+4+2=35
0

Focus $J * n$


2∗8=16
8-7=1
3
0
3
7
1
3
0
0
4
6
2
3
0
3
5
4
4
8-3=5
4
4
0
0
16+5+1=22


1∗8=8
8-7=1
4
0
8-1=7
3
7
1
3
0
4
6
2
3
0
3
5
4
4
8-3=5
4
4
0
0
8+7+5+1=21

Convergence

nous devons prouver que la fonction potentielle Δ diminue entre deux configurations illégitimes jusqu'à ce que Δ atteigne zéro

Le poids des jetons diminue : v_i

Lemma 3 : pour $i \neq 0$ lorsque v_i libère le jeton dans $\gamma \notin L$, le poids du jeton diminue.

La preuve :

Considérons le nœud v_i avec le jeton dans la configuration γ ainsi nous obtenons :

$$\delta_i(\gamma) = n - i \geq \delta_{i-1}(\gamma') = n - i - 1$$

Le poids des jetons diminue : v_0

Lemma 4 : Lorsque v_0 relâche le jeton, le poids du jeton diminue.

En d'autres termes : Si $v_0 \in \mathcal{A}^*(\gamma)$ et $\delta_0(\gamma) + \delta_1(\gamma) > \delta_0(\gamma') + \delta_1(\gamma')$

La preuve est faite : Désignons par k un entier inférieur à n grâce au lemme 2 on obtient :
$\delta_i(\gamma) = k * n > \delta_i(\gamma') \geq (k-1) * n$

$\delta_i(\gamma) + \delta_{i+1}(\gamma) = k * n + 0 > \delta_i(\gamma') + \delta_i(\gamma') \geq (k-1) * n + n - 1$
$kn > kn - n + n - 1 = kn - 1.$

Le nombre de jetons n'augmente pas

Lemma 5 : $|\mathcal{A}(\gamma)| \geq |\mathcal{A}(\gamma')|$$

Preuve : Si un noeud v_i avec un jeton dans la configuration γ exécute une règle, alors il change sa valeur x_i. Cette action n'affecte que son voisin v_{i+1} selon les règles R_0 et R_1. Si, après cela, v_i a le jeton dans γ', cela signifie que v_{i-1} avait le jeton dans γ.
Si un nœud v_i avec un jeton dans la configuration γ n'exécute pas de règle, alors il ne change pas sa valeur x_i. Ainsi, si son voisin v_{i-1} possède le jeton dans la configuration γ' et exécute une règle, soit les deux jetons fusionnent, soit les deux jetons disparaissent.

Rappel :

lemma 1 :$\forall \gamma \in \Gamma : |\mathcal{A}(\gamma)| \neq 0$

lemma 2 : Si v_0 dans $\gamma \notin L$ exécute R_0 alors $\Psi(\gamma) > \Psi(\gamma')$

Lemma 3 : Lorsque $v_i : i \neq 0$ libère le jeton dans $\gamma \notin L$, le poids du jeton diminue.

Lemma 4: Le poids du jeton diminue : Lorsque v_0 libère le jeton, le poids du jeton diminue.

Lemma 5 : $|\mathcal{A}(\gamma)| \geq |\mathcal{A}(\gamma')|$

Preuve du théorème 2

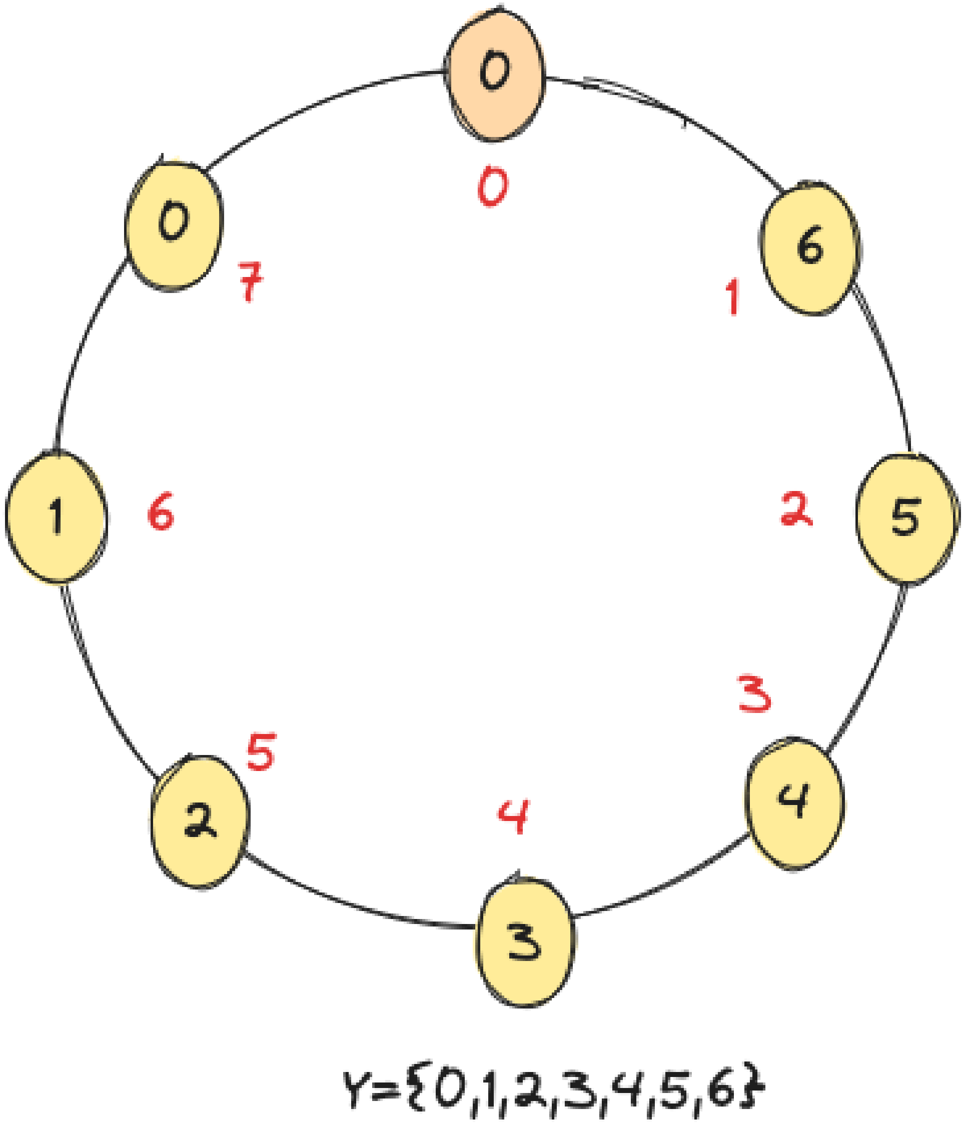
La preuve : Grâce au lemme 1 nous savons qu'il existe au moins un jeton dans un système, et grâce au lemme 5, nous avons vu que le nombre de jetons n'augmente pas. De plus, le poids d'un jeton diminue (voir les lemmes 3 et 4), donc comme $\Delta(\gamma)$ est la somme des poids des jetons dans la configuration γ, on obtient $\Delta(\gamma') < \Delta(\gamma)$.

Preuve de l'algorithme

Fermeture : Théorème 1

Convergence : Théorème 2

Qualité de la solution



Complexité temporelle : Pire cas

Complexité des étapes

- Décalages pour un nouveau jeton : $n * (n-2)$ étapes
- Placer le jeton à la fin de l'anneau : n étapes
- Supprimer les anciens jetons :
  - $(n-2) + (n-3) + (n-4) + \cdots + 1$
  - $\frac{(n-2)(n-1)}{2}$ étapes
- Total : $n^2 - 2n + n + \frac{(n-2)(n-1)}{2} = \frac{3n^2-5n+2}{2}$
  Complexité des étapes : $O(n^2)$

Complexité des arrondis

Remarque: à chaque configuration $\gamma \notin L$, tous les noeuds avec le jeton sont dans $\mathcal{A}(\gamma)$:

$O(n)$ rounds

Complexité de l'espace

Rappel : $s_v \in V$ une seule variable d'état : $s_v \in \{0, \ldots, n\}$.

> $O(\log n)$ bits de mémoire par noeud