Systèmes et Algorithmiques Répartis

Master 1
Informatique des Organisations - MIDO

Joyce EL HADDAD
elhaddad@lamsade.dauphine.fr

Chapitre 5 : Election

- Introduction
- Algorithme de Le Lann
- Algorithme de Chang et Roberts
- Algorithme de Franklin

Problématique

- Les sites d'une application répartie ont des fonctionnalités similaires
- Mais !!
- Certaines tâches spécifiques (ex. circulation de jeton) doivent être réalisées par une station unique
- Avant de démarrer l'application, les sites doivent se mettre d'accord sur l'identité de la station qui réalise la tâche
- Cette opération est appelée élection

Problématique

- Il est possible de fixer l'identité de ce site dans un fichier de configuration
- Inconvénients
  - En cas de changement de configuration du réseau, l'administrateur doit modifier le (ou les) fichier(s) de configuration
  - Un arrêt temporaire mais durable de la machine choisie nécessite une intervention manuelle
- Dans ce chapitre
  - Des solutions dynamiques où l'élection a lieu uniquement lorsqu'un site de l'application a besoin d'utiliser cette identité

Problématique

- Le service que nous réaliserons a une interface constituée d'une seule fonction Leader() qui renvoie l'identité du site élu
- La contrainte à vérifier : l'identité renvoyée est toujours la même quelque soit l'instant ou le site de l'appel
- Algorithmes selon le type de graphe de communication
  - anneau unidirectionnel (Le Lann77, Chang et Roberts79)
  - anneau bidirectionnel (Franklin82)
  - graphe quelconque

Problématique

- Propriétés communes
  - Avantage : un site ne connaît que ses voisins dans le graphe (l'insertion d'un nouveau site ne requiert aucune intervention au niveau des sites qui ne sont pas ses voisins --à condition que le type du graphe reste inchangé--)
  - Symétrie complète : chaque processus exécute le même algorithme
  - Algorithmes décentralisés : une exécution peut être commencée par un nombre quelconque de processus.
  - Les algorithmes atteignent une configuration terminale : il existe exactement un site « gagnant » et tous les autres sites sont « perdants »

Algorithme de Le Lann 1977

- Algorithme d'élection dans un anneau unidirectionnel : chaque site n'émet des messages que vers le site suivant sur l'anneau

- Hypothèses
  - Les sites « candidat » à l'élection sont les sites initiateurs
  - Les communications entre sites sont supposées fiables
  - Chaque site possède une identité unique et les identité sont ordonnées

Algorithme de Le Lann 1977

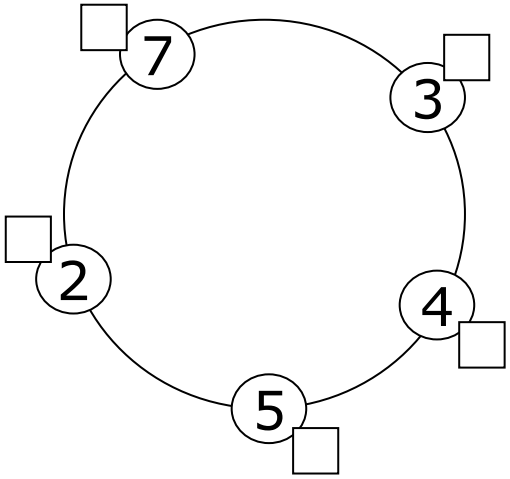
- Principe
  - Chaque initiateur envoie à son voisin de droite son identité
  - Un initiateur maintient une liste des identités des autres sites initiateurs
  - Dès qu'un initiateur reçoit sa propre identité, il sait qu'il a connaissance de tous les autres initiateurs alors il consulte sa liste:
    - s'il est l'initiateur dont l'identité est la plus petite alors il est élu et termine en envoyant un message indiquant à tous les autres sites qu'il a gagné

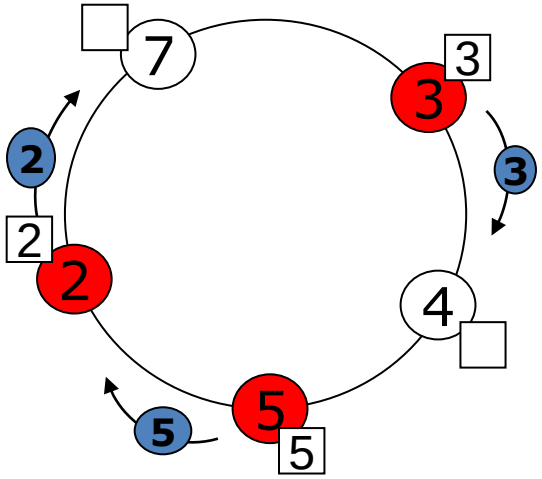
Algorithme de Le Lann 1977

- Lorsque l'application « réclame » l'identité du leader, trois cas se présentent :
  - Le processus d'élection est terminé et le service renvoie immédiatement l'identité du site élu
  - Le processus d'élection est en cours et connu du service. Dans ce cas, le service attend la terminaison du processus afin de renvoyer l'identité du leader
  - Le service n'est pas au courant d'un processus d'élection engagé. Dans ce cas, il envoie une requête circuler sur l'anneau afin de devenir le leader

Algorithme de Le Lann 1977

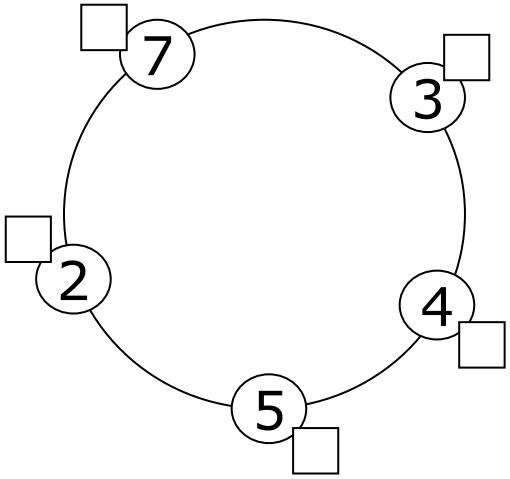
- Scénario d’une exécution

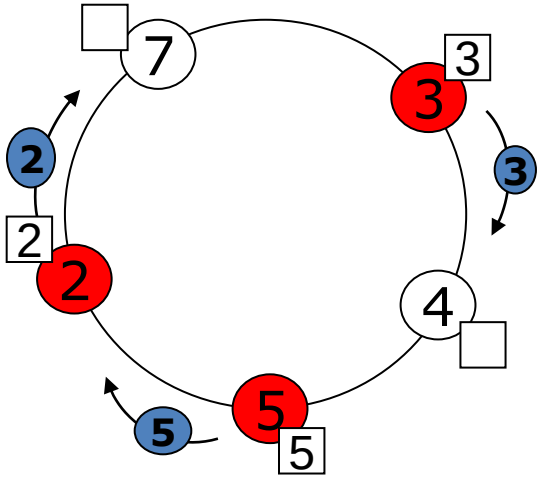


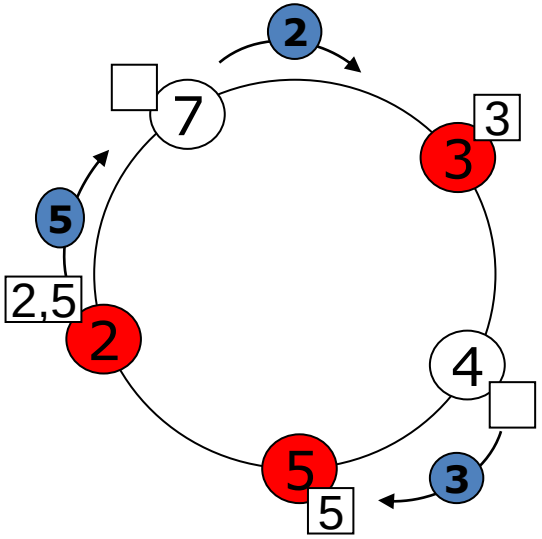


Algorithme de Le Lann 1977

- Scénario d'une exécution

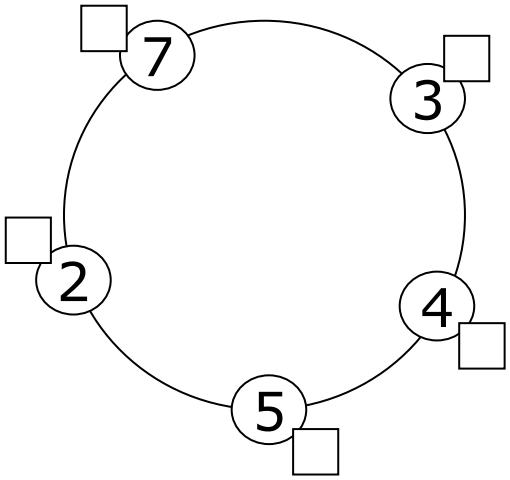


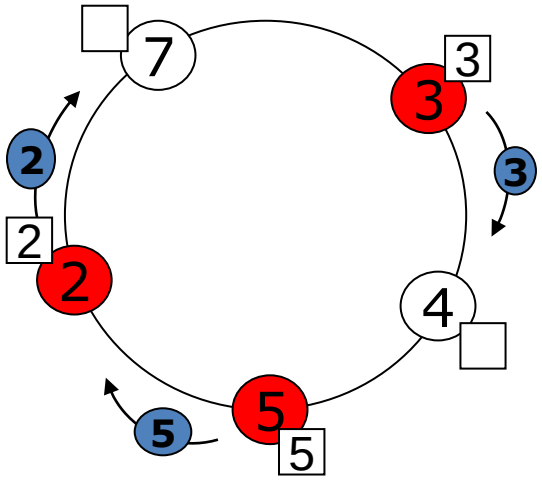


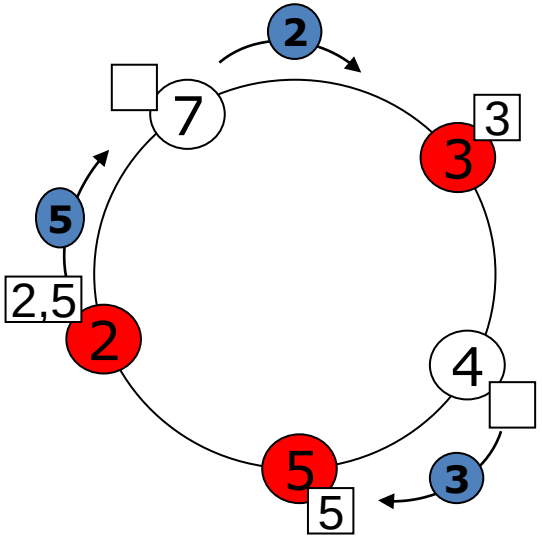


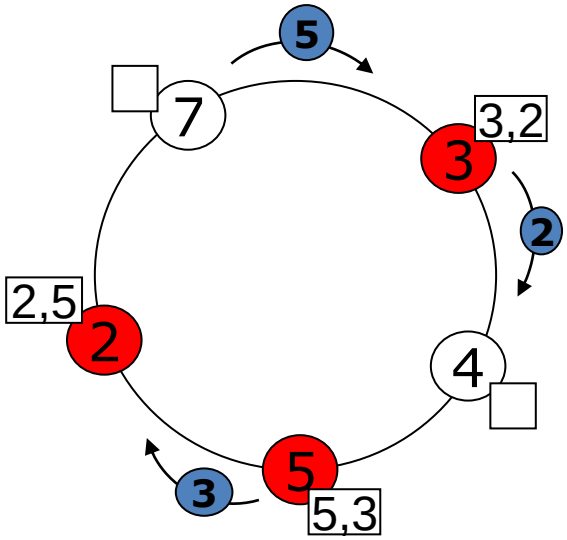
Algorithme de Le Lann 1977

- Scénario d'une exécution



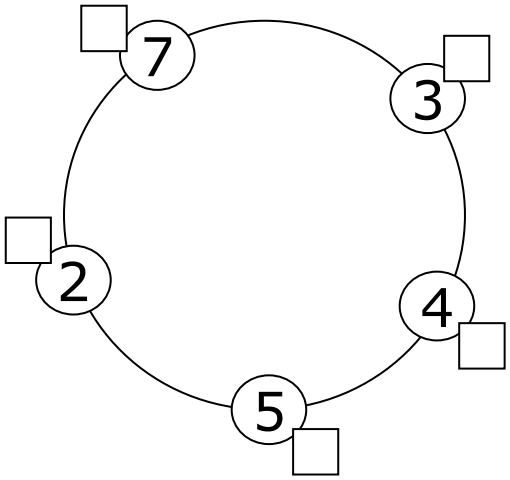


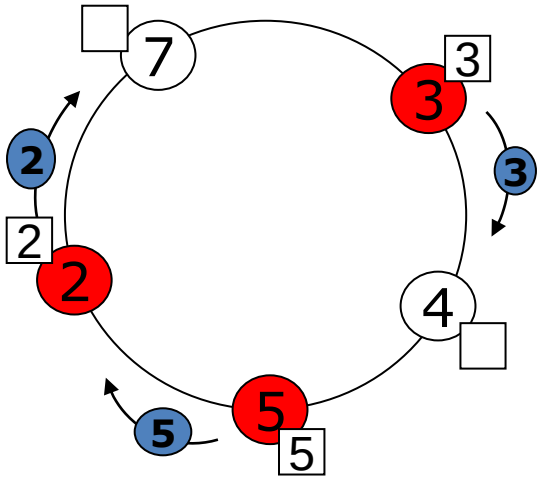


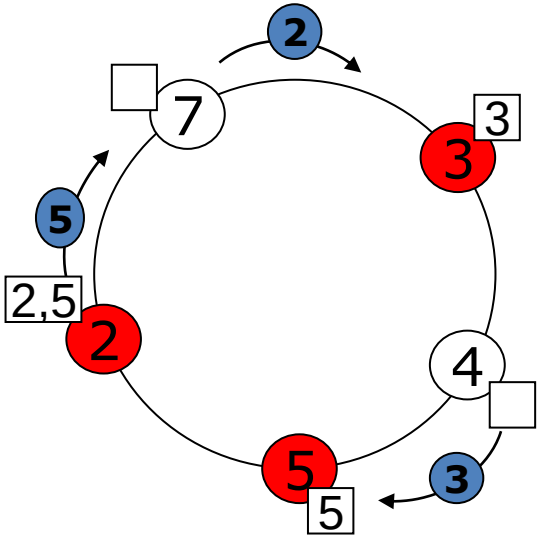


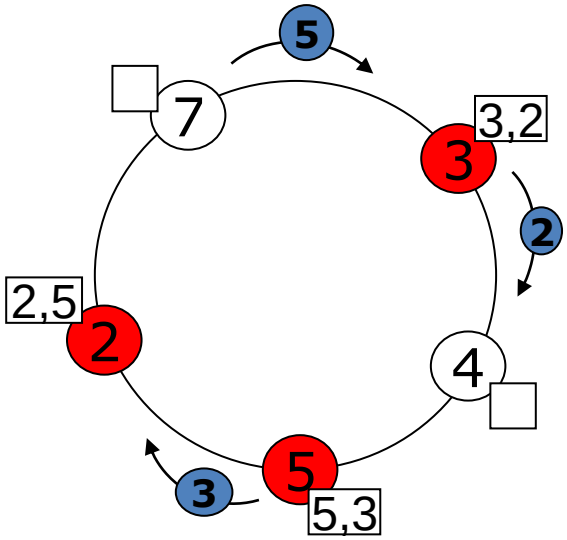
Algorithme de Le Lann 1977

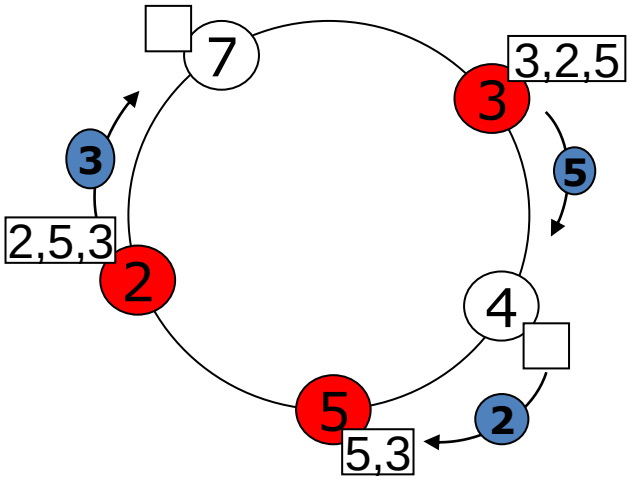
- Scénario d'une exécution





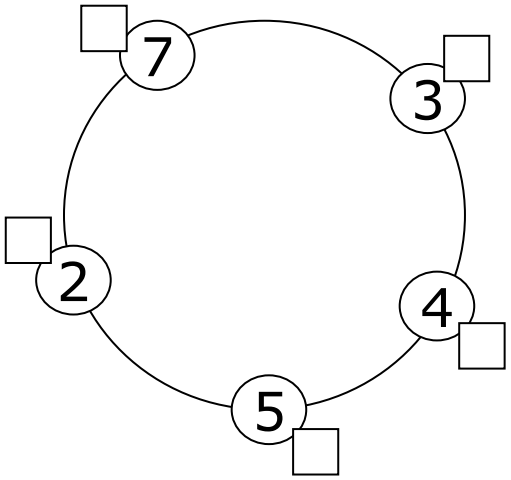


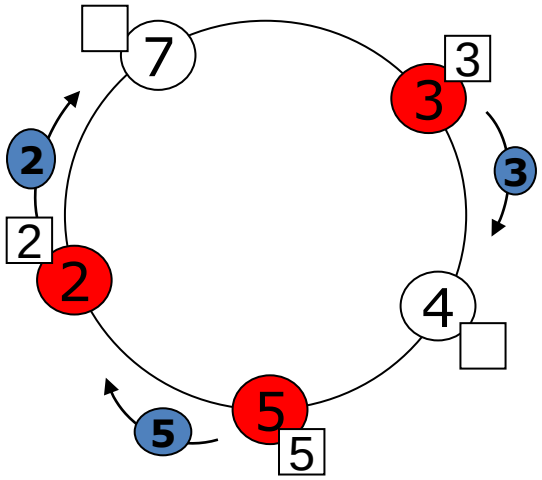


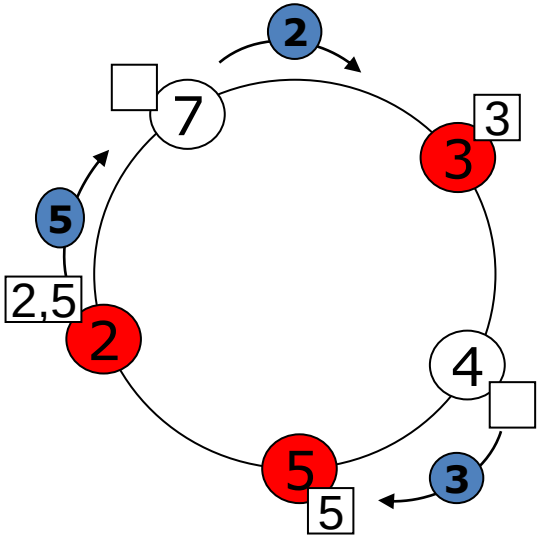


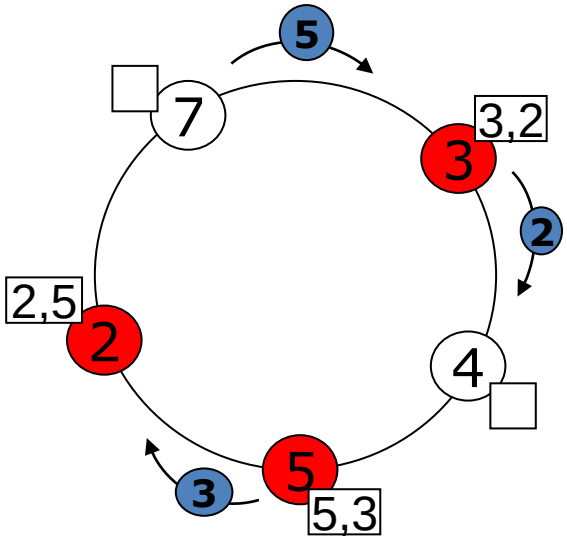
Algorithme de Le Lann 1977

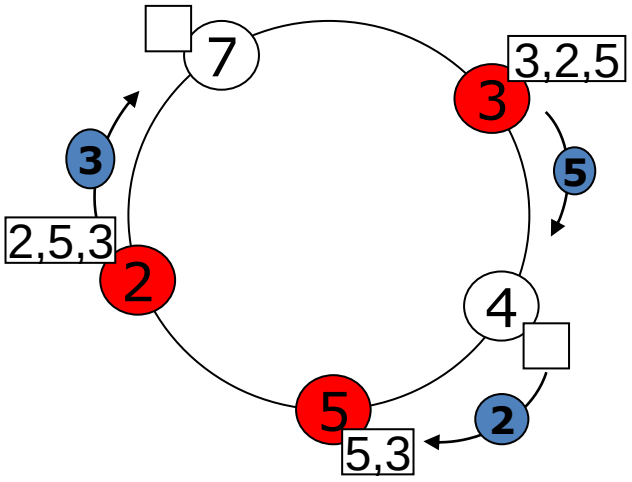
- Scénario d'une exécution

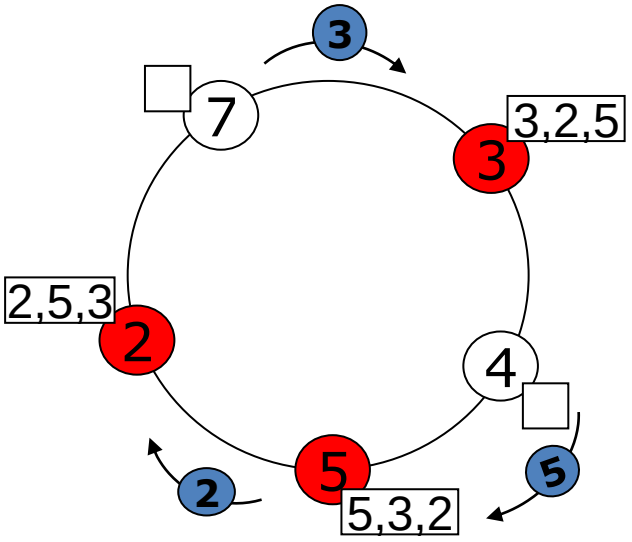






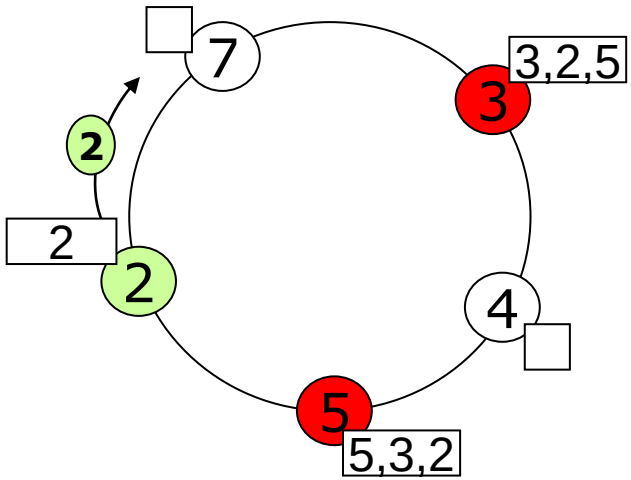


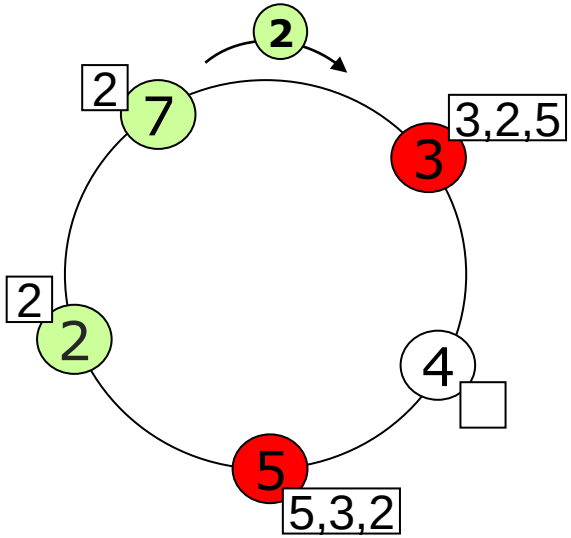


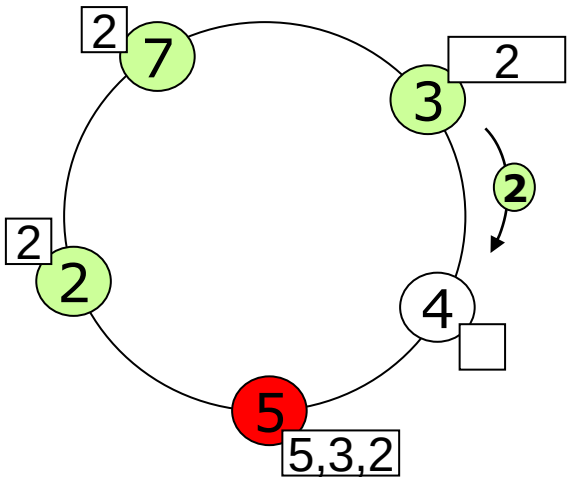


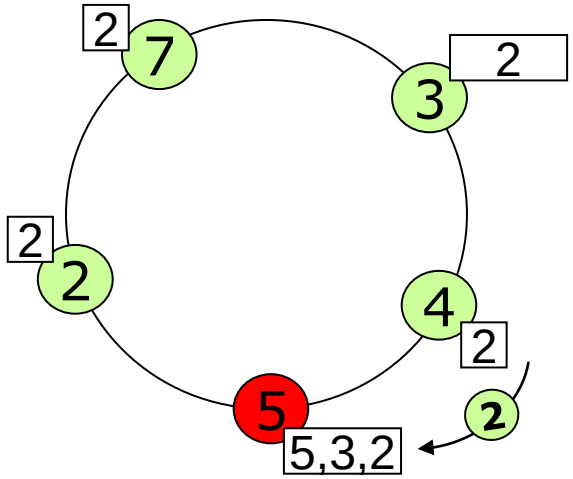
Algorithme de Le Lann 1977

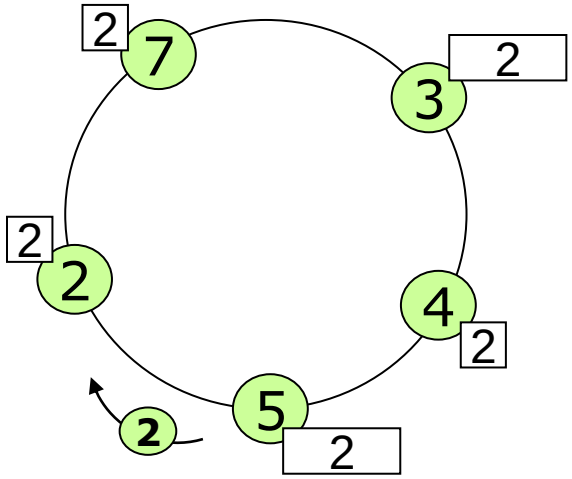
- Scénario d'une exécution

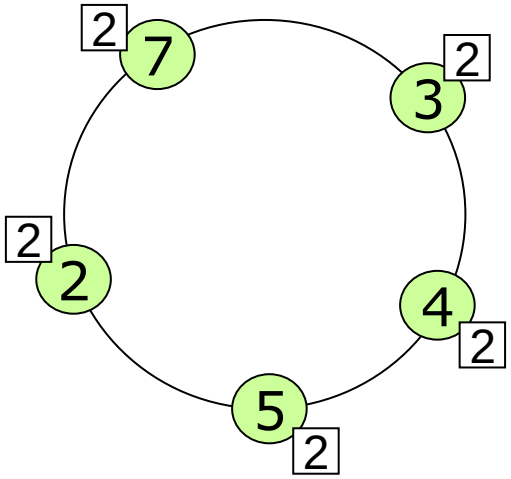












Algorithme de Le Lann 1977

- Les variables d'un site S_i
  - $suivant_i$: constante contenant l'identité du site successeur de S_i sur l'anneau.
  - $état_i$: état du site à valeur dans {repos, en_cours, terminé}. Elle est initialisée à {repos}.
  - $chef_i$: identité du site élu.
  - $liste_i$: liste des identités des sites candidats. Initialement vide.

Algorithme de Le Lann 1977

- Algorithme d'un site S_i

```
Leader( )
    Début
            liste_i=Ø;
            Si (état_i==repos et start _i = false)  Alors
                    état_i=en_cours;
                    liste_i=liste_i ⋃ {i};
                    envoyer_à(suivant_i,(req,i));
            Finsi
            Attendre(état_i==terminé);
            renvoyer(chef_i);
    Fin
```

Algorithme de Le Lann 1977

- Algorithme d'un site S_i

```
Sur_réception_de ( j, (req, k) )
    Début
        Si (état_i==repos) Alors
                envoyer_à(suivant_i,(req,k));
                start _i=true;
        Sinon  si (i!=k) Alors
                liste_i=liste_i ⋃ {k};
                envoyer_à(suivant_i,(req,k));
            sinon
                si (i==Min(liste_i)) Alors
                    état_i= terminé;
                    chef_i= i;
                   envoyer_à(suivant_i,(conf,i));
        finsi
            finsi
        Finsi
    Fin
```

Algorithme de Le Lann 1977

- Algorithme d'un site S_i

```
Sur_réception_de ( j, (conf, k) )
    Début
        Si (i!=k) Alors
                envoyer_à(suivant_i,(conf,k));
                état_i=terminé;
                chef_i=k;
        Finsi
    Fin
```

Algorithme de Le Lann 1977

- Preuve de l'algorithme

L'algorithme LeLann77 résout le problème de l'élection pour n processus connectés par un anneau unidirectionnel avec connaissance réduite aux voisins

L'ordre des jetons est préservé sur l'anneau (propriété FIFO). Un initiateur Si envoie son jeton avant toute réception de jeton d'un autre site Sj. L'initiateur Sj reçoit le jeton de Si avant son propre jeton.

Donc, Sj termine avec Listej qui contient tous les autres sites initiateurs

Par conséquent, l'initiateur d'identité minimale est le seul a passé dans l'état terminé et a envoyé son message de confirmation

Algorithme de Le Lann 1977

- Complexité de l'algorithme

	Nombre de messages	Temps
Au mieux	$O(n)$	$O(n)$
Au pire	$O(n^2)$	$O(n)$
En moyenne	$O(n^2)$	$O(n)$

L'algorithme LeLann résout le problème de l'élection avec un nombre maximum de messages échangés (une majoration) $O(n^2)$ et en $O(n)$ unité de temps

Algorithme de Le Lann 1977

- Preuve

Notons d'abord que chaque site envoie au plus une requête.

Chaque requête donne lieu à exactement n envois de message.

Enfin, le message de confirmation donne lieu à exactement n envois de message.

En cumulant, nous obtenons au maximum (n^2+n) messages échangés.

Au plus tard, après n-1 unité de temps

- le premier initiateur a déjà lancé son jeton,
- chacun des autres initiateurs a déjà lancé sont jeton

Sachant que chaque initiateur recevra à nouveau son jeton dans n unité de temps et que le message de confirmation prendra n unité de temps, l'algorithme se terminera dans 3n-1 unité de temps.

Algorithme de Chang et Roberts [1979]

- Algorithme d'élection dans un anneau unidirectionnel
  - Chaque site n'émet des messages que vers le site suivant sur l'anneau

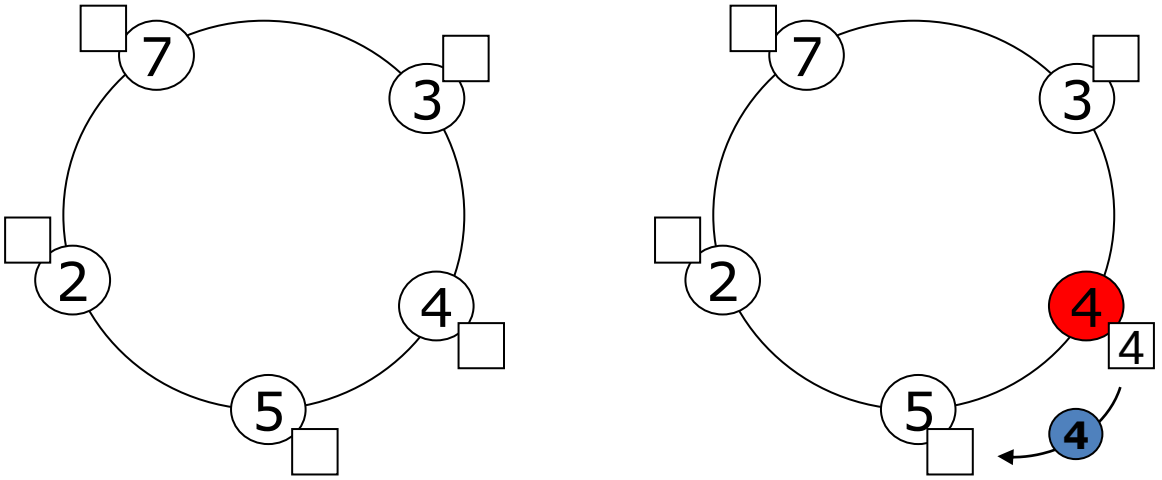
- Hypothèses
  - Les sites candidat à l'élection sont les sites initiateurs
  - Les communications entre sites sont supposées fiables
  - Chaque site possède une identité unique et les identité sont ordonnées

Algorithme de Chang et Roberts [1979]

- Principe
  - Chaque initiateur envoie à son voisin de droite son identité
  - Dès qu'un initiateur découvre qu'il n'est pas celui dont l'identité est la plus petite, il s'éteint et ne fait que relayer les messages qu'il reçoit à son voisin de droite
- Lorsqu'une requête parvient à un autre site, deux cas se présentent :
  - Le service du site était au repos ou avait provisoirement choisi un leader d'identité supérieure; il adopte l'initiateur de la requête comme nouveau leader potentiel et retransmet la requête au site suivant de l'anneau
  - Le service du site avait provisoirement choisi un leader d'identité inférieure; il ne donne pas suite à la requête
- Lorsqu'une requête revient à son initiateur, il conclut qu'il est l'élu et termine en envoyant un message de confirmation indiquant aux autres sites qu'il a gagné et que l'élection est terminée

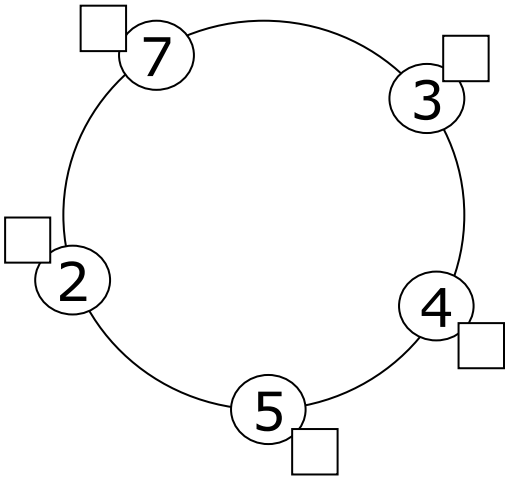
Algorithme de Chang et Roberts [1979]

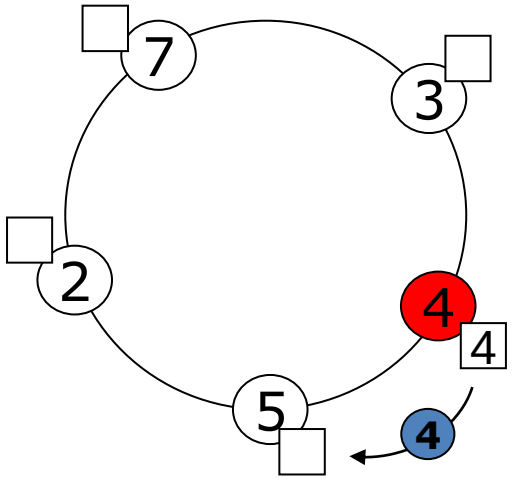
- Scénario d'une exécution

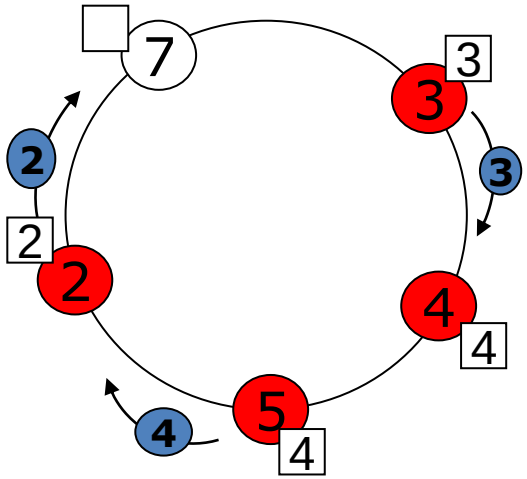


Algorithme de Chang et Roberts [1979]

- Scénario d'une exécution

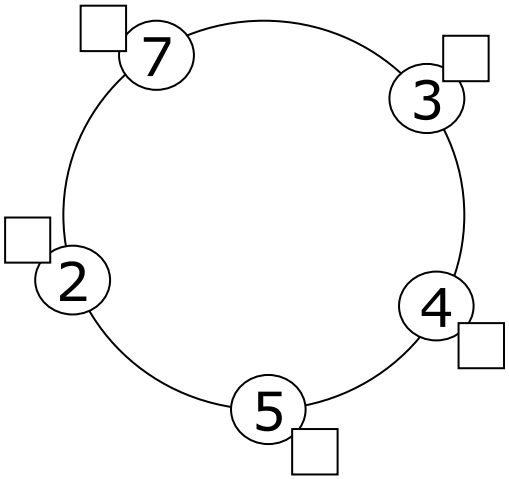


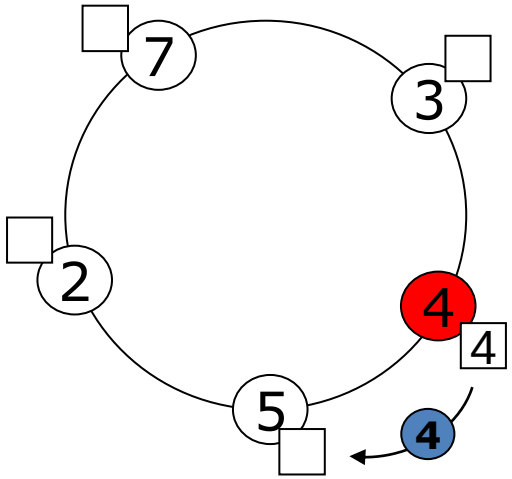


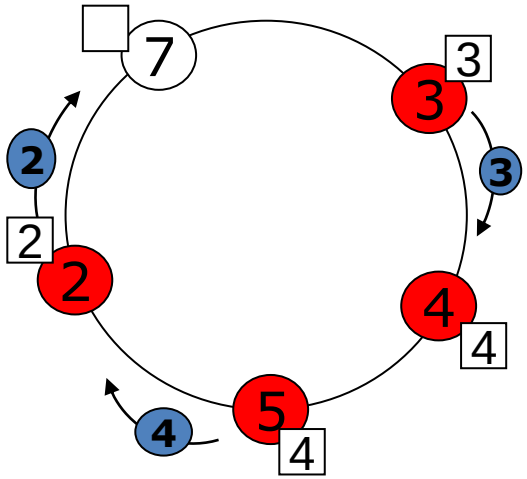


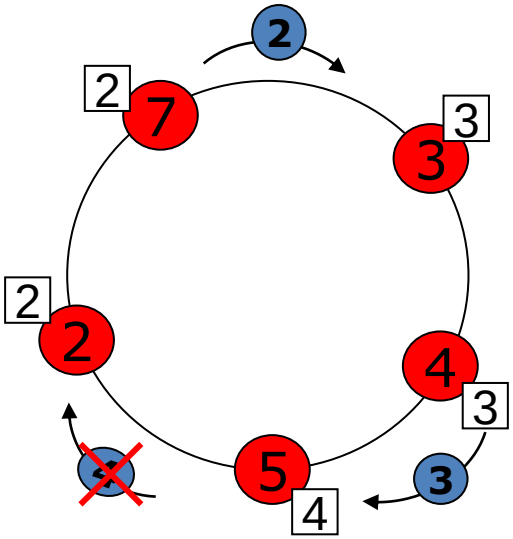
Algorithme de Chang et Roberts [1979]

- Scénario d'une exécution



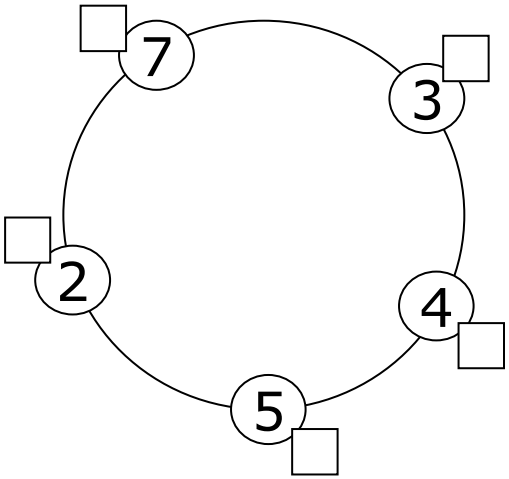


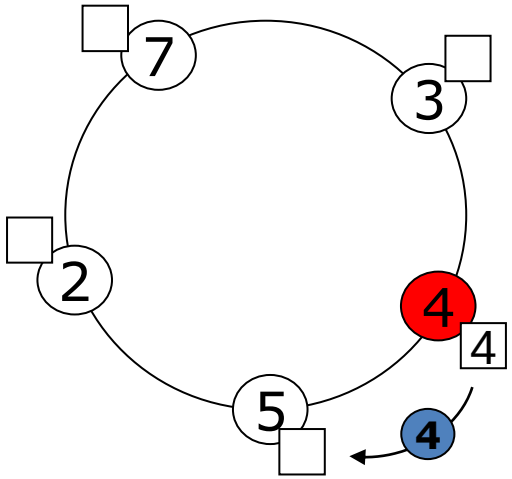


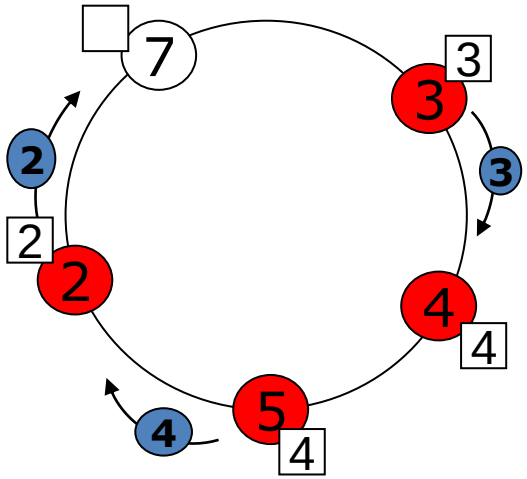


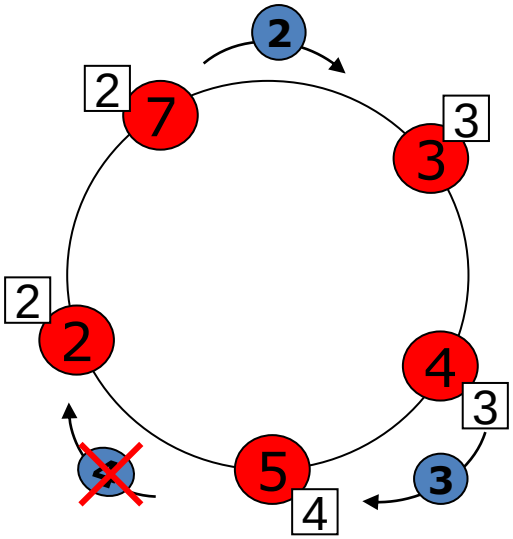
Algorithme de Chang et Roberts [1979]

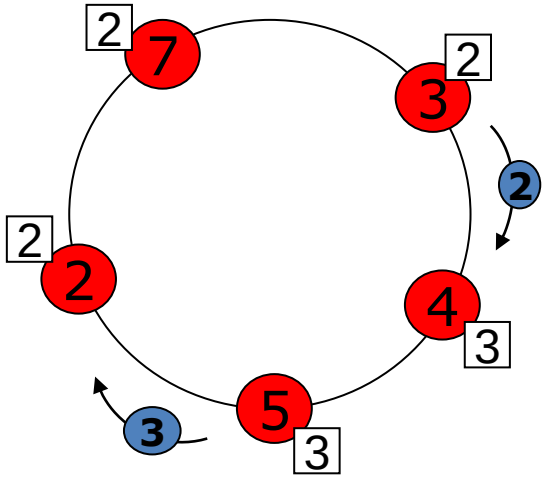
- Scénario d'une exécution





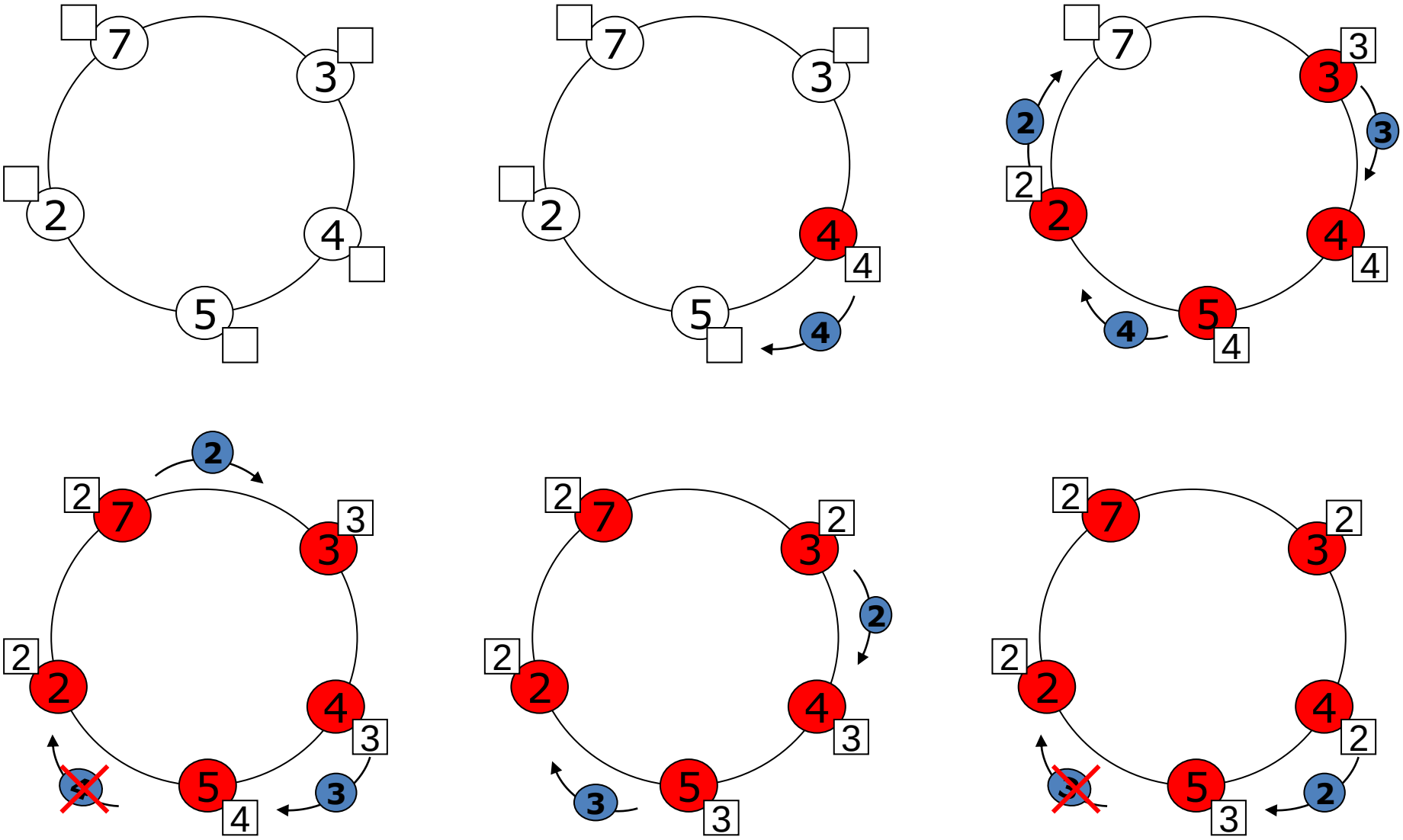






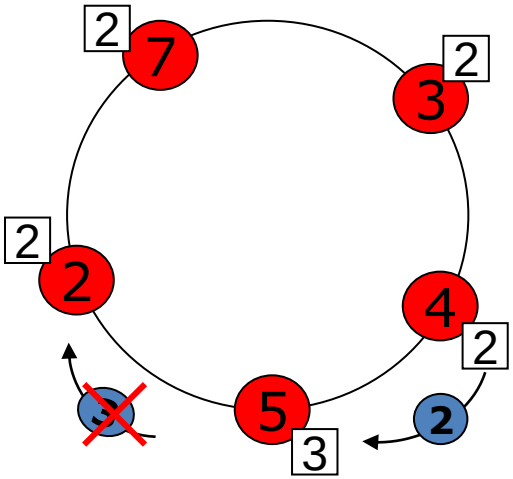
Algorithme de Chang et Roberts [1979]

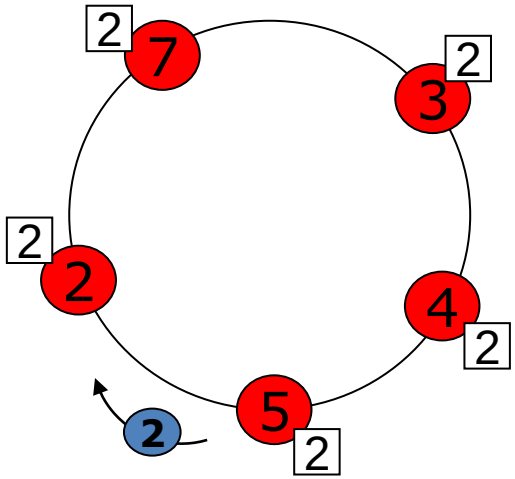
- Scénario d'une exécution



Algorithme de Chang et Roberts [1979]

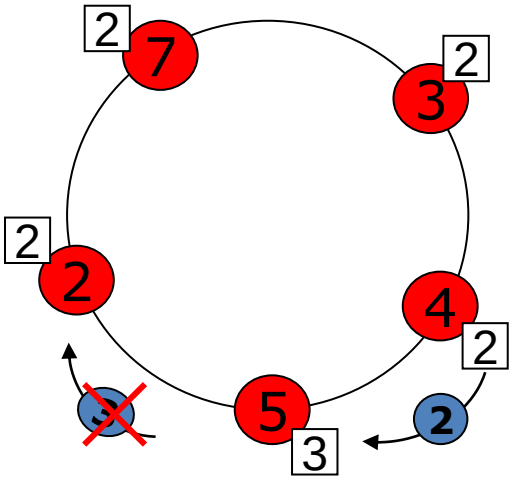
- Scénario d'une exécution

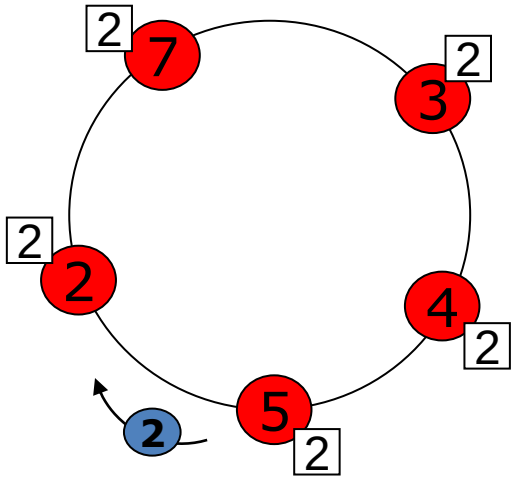


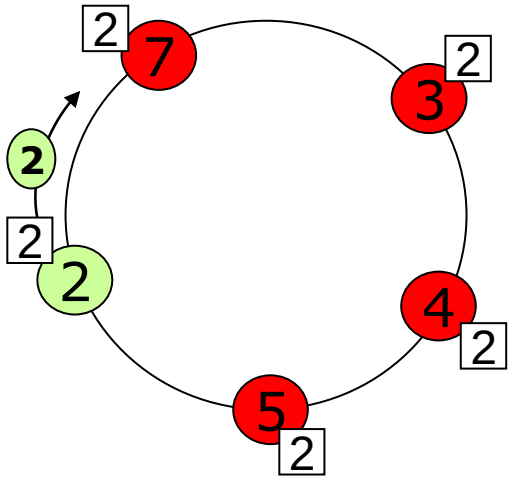


Algorithme de Chang et Roberts [1979]

- Scénario d'une exécution

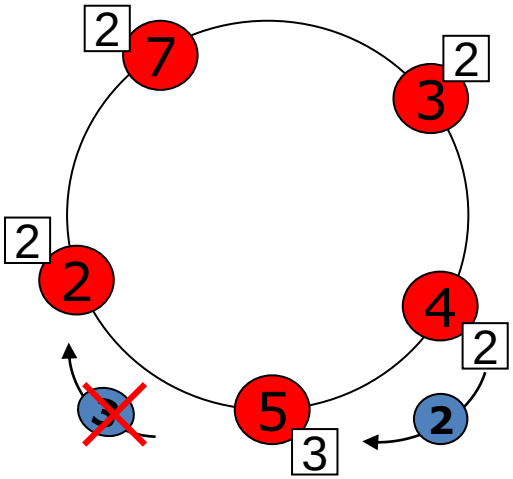


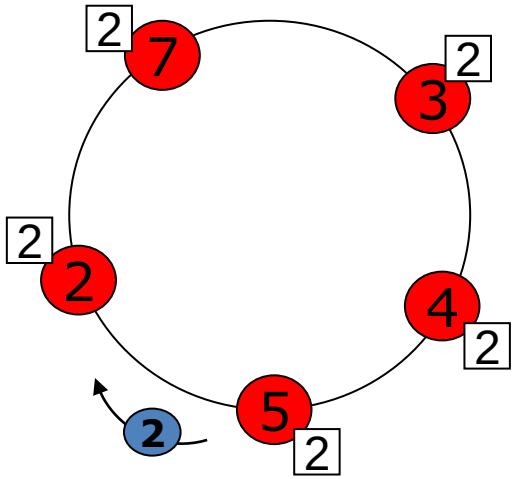


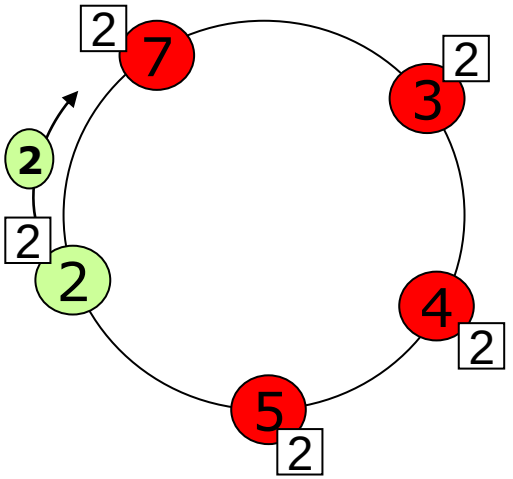


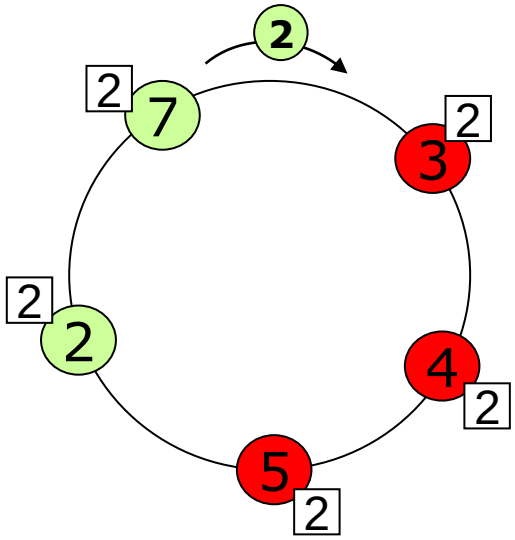
Algorithme de Chang et Roberts [1979]

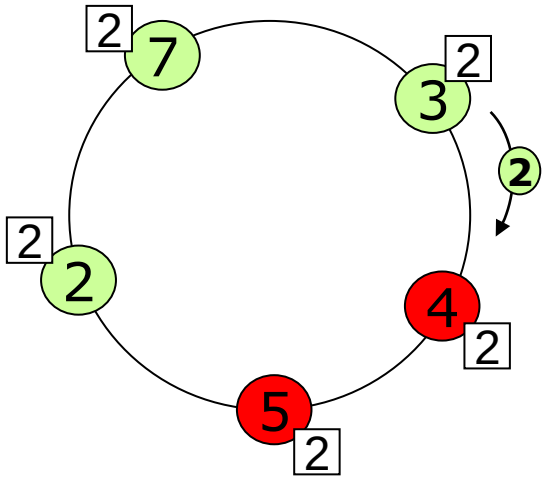
- Scénario d'une exécution

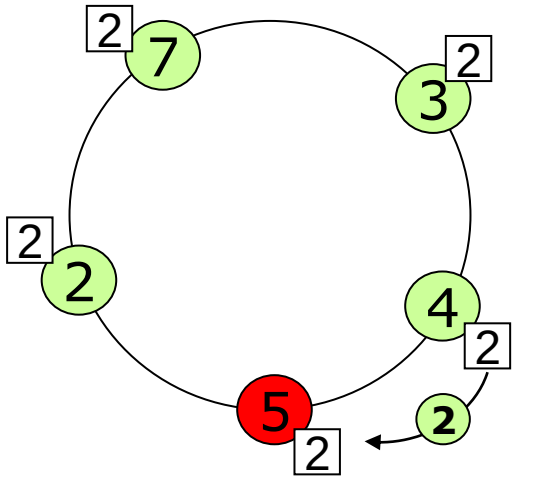






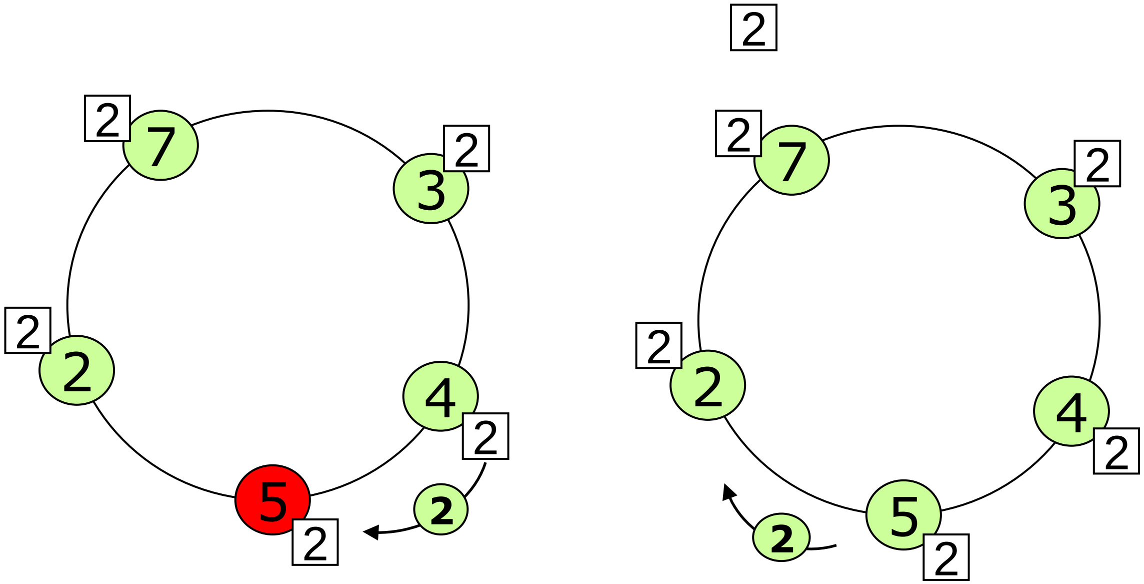






Algorithme de Chang et Roberts [1979]

- Scénario d'une exécution



Algorithme de Chang et Roberts [1979]

Variables du site S_i

1. $suivant_i$: constante contenant l'identité du site successeur de S_i sur l'anneau
2. $état_i$: état du site à valeur dans {repos, en_cours, terminé}. Elle est initialisée à repos
3. $chef_i$: identité du site élu

Algorithme du site S_i

Si aucun processus d'élection n'a atteint un site S_i, alors le service initialise un tel processus

Le service d'un site attend que le processus d'élection soit terminé pour renvoyer l'identité du site élu

Chang et Roberts 1979

```
Leader()
Début
   Si (état_i==repos)  Alors
       état_i=en_cours;
       chef_i=i;
       envoyer_à(suivant_i,(req,i));
   Finsi
   Attendre(état_i==terminé);
   renvoyer(chef_i);
Fin
```

Si le site est au repos, la réception d'une requête provoque le changement d'état et la retransmission de la requête

Dans le cas où le processus reçoit une "meilleure" requête, il en tient compte et retransmet la requête

Enfin si une requête parvient à son initiateur, ce site est élu et il avertit les autres sites

Chang et Roberts 1979

```
Sur_réception_de(j,(req,k))
Début
   Si (état_i==repos || k < chef_i) Alors
       état_i=en_cours;
       chef_i=k;
       envoyer_à(suivant_i,(req,k));
   Sinon
       si (i==k) Alors
          état_i=terminé;
          envoyer_à(suivant_i,(conf,i));
       fsi
   Finsi
Fin
```

Chang et Roberts 1979

```
Sur_réception_de(j,(conf,k))
Début
   Si (i!=k) Alors
    envoyer_à(suivant_i,(conf,k));
    état_i=terminé;
   Finsi
Fin
```

Cette primitive permet de remettre l'état des sites non élus à « terminé »

Chang et Roberts 1979

Preuve de l'algorithme

L'algorithme de Chang et Roberts79 résout le problème de l'élection pour n processus connectés par un anneau unidirectionnel avec connaissance réduite aux voisins

Soit S_0 l'initiateur d'identité minimale, la requête (req,S_0) circule de bout en bout et atteint S_0. Le site S_0 passera donc à l'état terminé, enverra son message de confirmation qui entraînera à sa réception le passage de tous les autres sites à ce même état et l'adoption de S_0 comme leader.

Il reste à démontrer qu'aucun autre message de confirmation ne circulera sur le réseau:
Soit S_1 un autre initiateur, si la requête de S_1 parvient à S_0, elle sera détruite puisque l'émission de la requête de S_0 précède cette réception (par définition d'un initiateur). En conséquence, aucun autre message de confirmation ne sera émis.

Chang et Roberts 1979

Complexité de l'algorithme

	Nombre de message	temps
Au mieux	$O(n)$	$O(n)$
Au pire	$\Theta(n^2)$	$O(n)$
En moyenne	$O(nlog(n))$	$O(n)$

L'algorithme Chang et Roberts résout le problème de l'élection avec un nombre maximum de messages échangés $\Theta(n^2)$ et en $O(n)$ unité de temps

Chang et Roberts 1979 : Complexité

Complexité au pire des cas

Notons Pire(n) = le nombre maximum de messages échangés durant une élection sur un anneau composé de n sites.

Chaque site envoie au plus une requête.

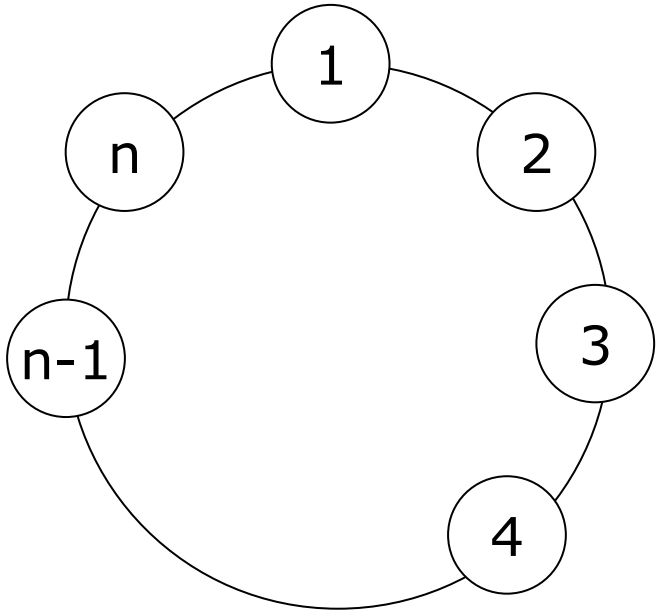
Chaque requête donne lieu à au plus n envois de message.

Le message de confirmation donne lieu à exactement n envois de message.

En cumulant ces bornes nous obtenons une majoration,

$$\text{Pire}(n) \leq n^2 + n$$

Chang et Roberts 1979 : Complexité



Examinons le cas défavorable où tous les sites sont initiateurs, la requête issue de S_i est envoyée exactement (n-i+1) fois. Ce qui nous donne pour cette exécution un nombre de messages (une minoration Ω) égal à :

$\Sigma_{i=1 \text{ à } n}\ (n-i+1) + n = \Sigma_{i=0 \text{ à } n-1}\ (n-i) + n = n(n+1)/2 + n \leq \text{Pire}(n)$

Cet encadrement nous fournit l'ordre exact de grandeur recherché :

$\text{Pire}(n) = \Theta(n^2)$

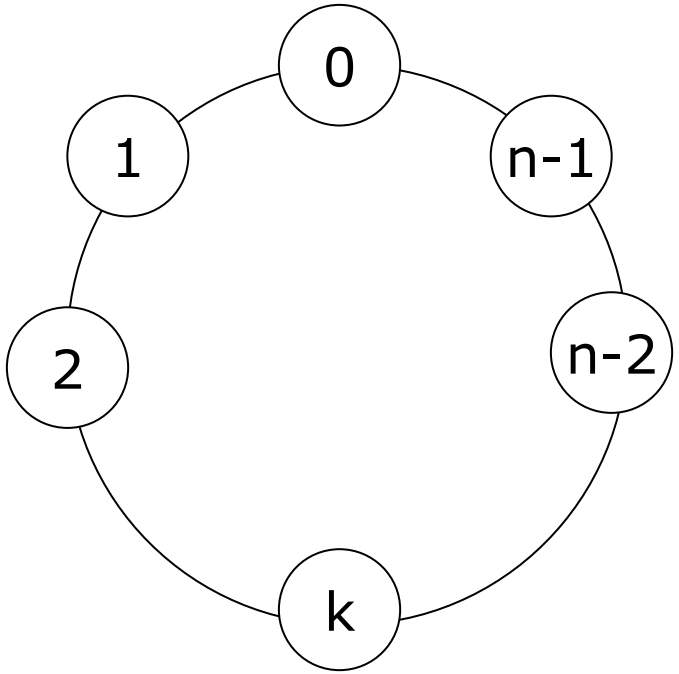
Chang et Roberts 1979 : Complexité

Complexité en moyenne

Notons Moyenne(n) = le nombre moyen de messages échangés sur l'ensemble des exécutions possibles.

Considérons la configuration suivante où tous les sites sont initiateurs et S_0 le site d'identité minimale.

Soit X_k la variable aléatoire représentant le nombre de messages issus de la requête de S_k (pour $k \neq 0$). Puisque cette requête ne pourra être relayée par S_0, nous avons : $Prob(X_k \geq k+1)=0$



Pour que cette requête soit relayée au moins t fois pour $t \leq k$, il faut et il suffit que S_k soit l'identité minimale dans l'ensemble des identités $\{S_k,...,S_{k-t+1}\}$.

Supposons que toutes les configurations sont équiprobables, donc cette minimalité est satisfaite avec une probabilité 1/t ⇨ $Prob(X_k \geq t)= 1/t$

Chang et Roberts 1979 : Complexité

Complexité en moyenne

Espérance est un paramètre de position, les valeurs possibles d'une variable aléatoire gravitent autour de cette valeur:

$E(X) =_{def} \sum_{t=1 \text{ à } \infty} t.Prob(X=t)$

$= \sum_{t=1 \text{ à } \infty} \sum_{s=1 \text{ à } t} Prob(X=t)$ *(remplacement du produit par la somme)*

$= \sum_{s=1 \text{ à } \infty} \sum_{t=s \text{ à } \infty} Prob(X=t)$ *(inversion des sommes)*

$= \sum_{s=1 \text{ à } \infty} Prob(X \geq s)$

D'où : $E(X_k) = \sum_{t=1 \text{ à } k} 1/t$ *(puisque $Prob(X_k \geq t) = 1/t$)*

En sommant ces différentes espérances et en ajoutant les deux tours (requête et confirmation) dûs à S_0 :

$Moyenne(n) = \sum_{k=1 \text{ à } n-1} \sum_{t=1 \text{ à } k} 1/t + 2.n$

$= \sum_{t=1 \text{ à } n-1} \sum_{k=t \text{ à } n-1} 1/t + 2.n$

$= \sum_{t=1 \text{ à } n-1} (n-t)/t + 2.n$ *(inversion des sommes et rempla-*

$= \sum_{t=1 \text{ à } n-1} n/t + \sum_{t=1 \text{ à } n-1} -t/t + 2.n$ *cement de la somme par le produit)*

$= n.\sum_{t=1 \text{ à } n-1} 1/t + n + 1$

$= n.\sum_{t=1 \text{ à } n-1} 1/t + n + n.(1/n)$

Chang et Roberts 1979 : Complexité

Complexité en moyenne

Finalement :

$$\text{Moyenne}(n) = n.\sum_{t=1 \text{ à } n} 1/t + n$$

Il nous reste à obtenir un ordre de grandeur de cette quantité.

Nous nous servons de l'encadrement immédiat :

$$\int_{t}^{t+1} 1/x \, dx \leq 1/t \leq \int_{t-1}^{t} 1/x \, dx \quad \text{pour } t \geq 2$$

ce qui nous donne par sommation :

$$\log(n+1) = \int_{1}^{n+1} 1/x \, dx \leq \sum_{t=1 \text{ à } n} 1/t \leq 1 + \int_{1}^{n} 1/x \, dx = 1+\log(n)$$

$$\log(n+1) \leq \sum_{t=1 \text{ à } n} 1/t \leq 1+\log(n)$$

$$n.\log(n+1) \leq n.\sum_{t=1 \text{ à } n} 1/t \leq n+n.\log(n)$$

$$n.\log(n+1)+n \leq n.\sum_{t=1 \text{ à } n} 1/t + n \leq n+n.\log(n)+n$$

$$n.\log(n+1)+n \leq \text{Moyenne}(n) \leq n.\log(n)+2.n$$

D'où :

$$\text{Moyenne}(n) = O(n.\log(n))$$

Franklin 1982

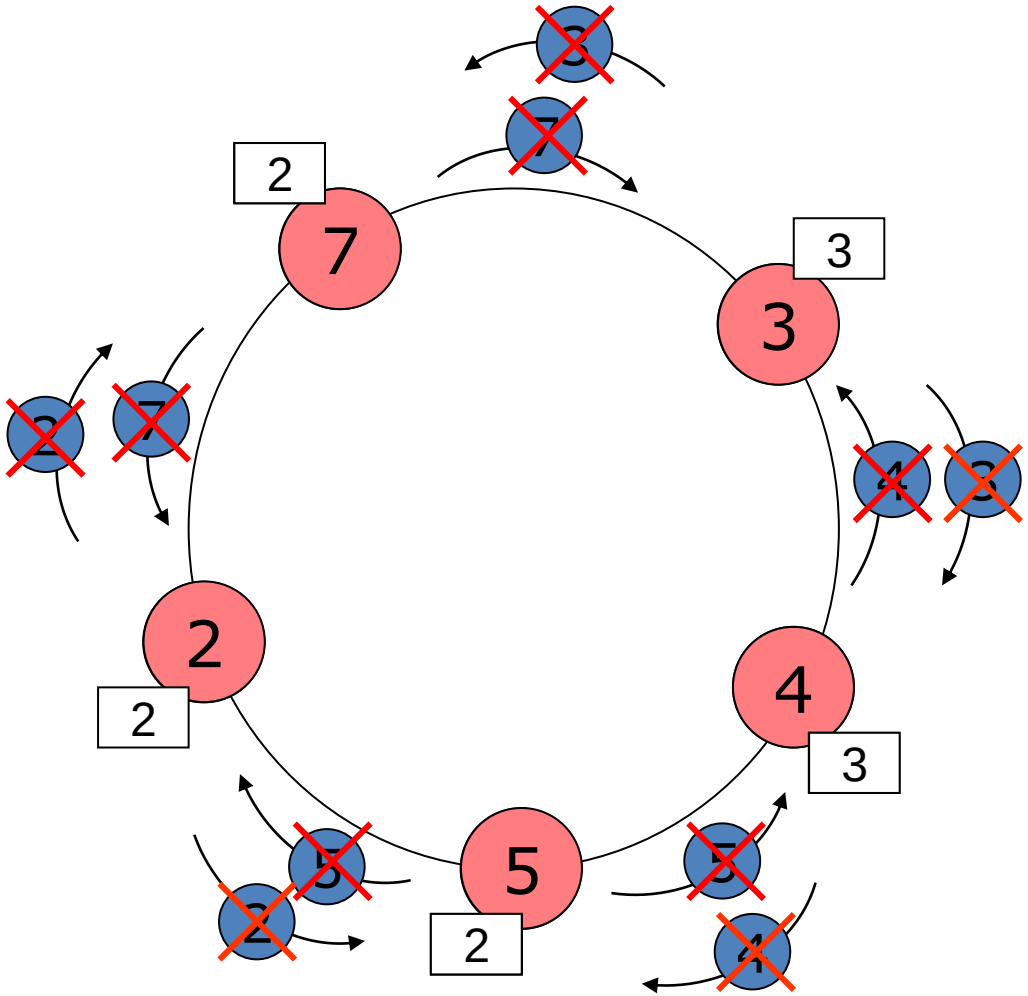
- Algorithme d'élection dans un anneau bidirectionnel
  - Un site peut émettre des messages vers le site *suivant* et *précédent* sur l'anneau

- Principe
  - Des "tours" où les compétiteurs sont les initiateurs
  - A chaque tour, un initiateur survivant envoie à ses voisins (à gauche et à droite) sa candidature. De ce fait, chaque initiateur reçoit les requêtes des initiateurs voisins gauche et droite et ne survit au tour suivant que s'il a la plus petite identité
  - Le tournoi s'achève quand un initiateur sait qu'il est ou sera au prochain tour le seul survivant. Il acquiert cette certitude soit en recevant l'un des messages qu'il a envoyé (ou éventuellement les deux) soit en recevant deux messages d'un même site
  - Les sites "éliminés" participent à l'algorithme en transmettant les messages qu'ils reçoivent
  - Un message de confirmation achève l'algorithme

Franklin 1982

Scénario d'une exécution

1. Toutes les applications appellent la fonction `Leader()`
2. Chaque site envoie sa candidature à gauche et à droite
3. Tour 1 : seuls les sites 2 et 3 survivent et se retrouvent au deuxième tour
4. Tour 2 : les sites 2 et 3 envoient à nouveau leur candidature qui sont relayés par les sites éliminés



Franklin 1982

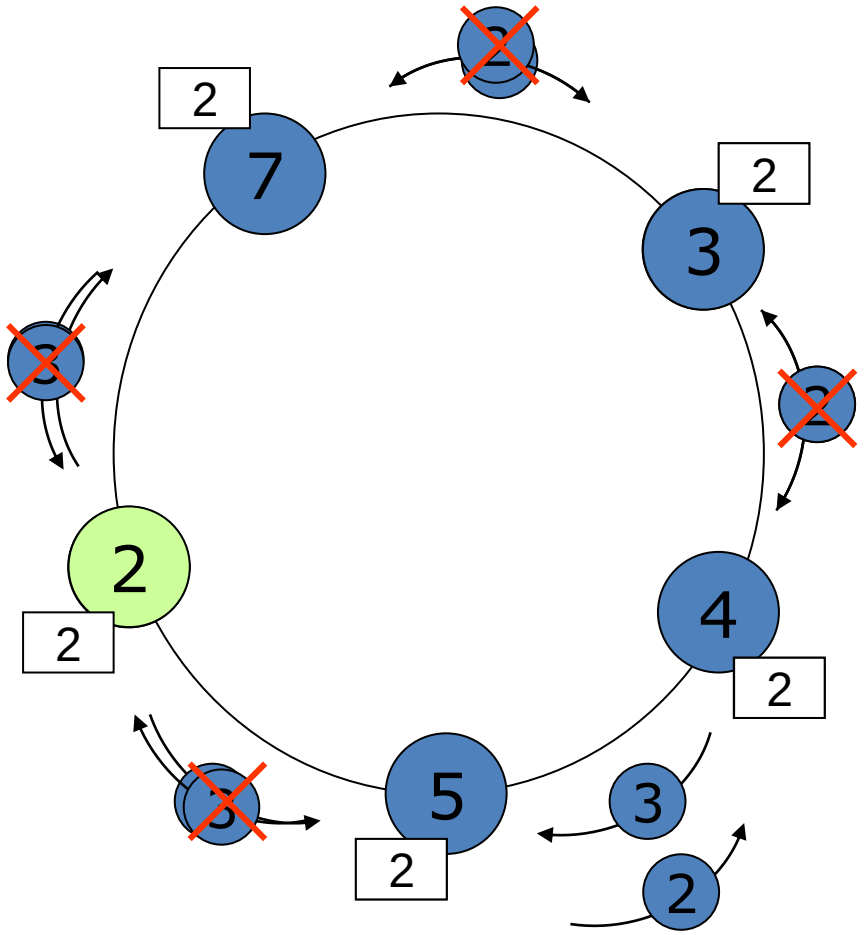
Scénario d'une exécution

1. Toutes les applications appellent la fonction `Leader()`

2. Chaque site envoie sa candidature à gauche et à droite

3. Tour 1 : seuls les sites 2 et 3 survivent et se retrouvent au deuxième tour

4. Tour 2 : les sites 2 et 3 envoient à nouveau leur candidature qui sont relayés par les sites éliminés



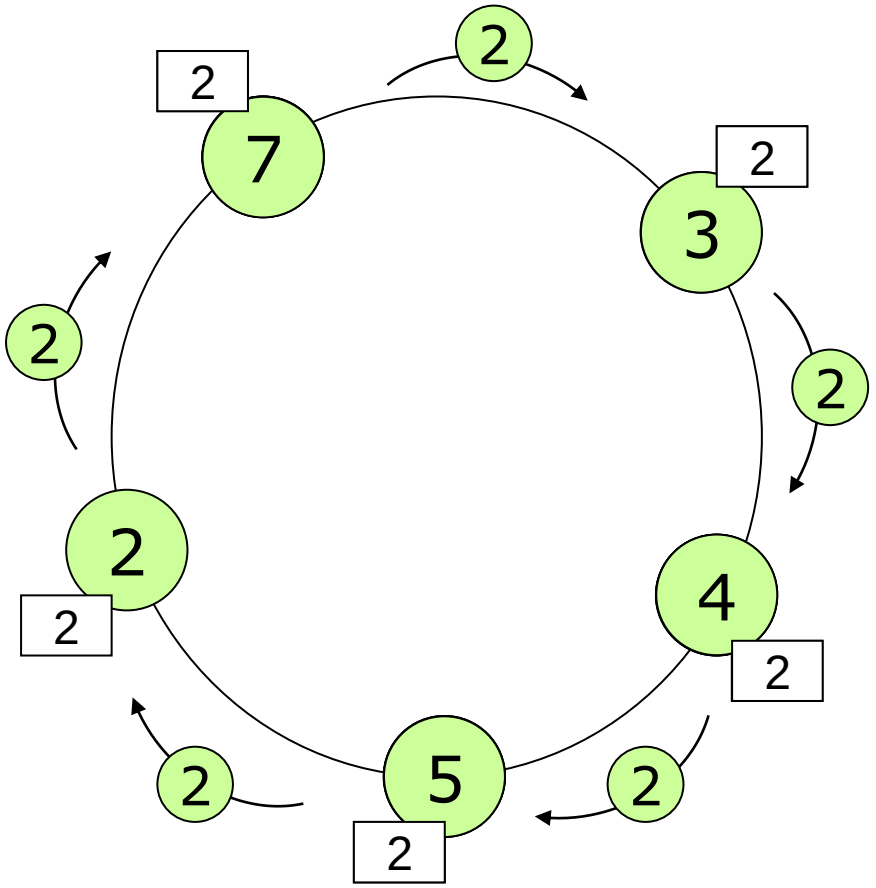
Franklin 1982

Scénario d'une exécution

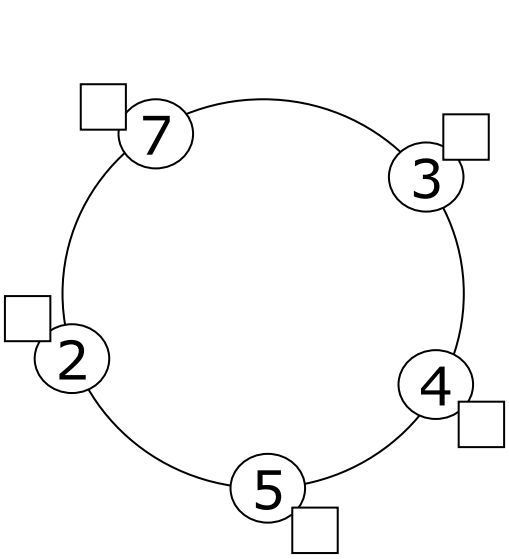
5. Tour 2 : lors de ce tour le site 2 est le seul survivant et de plus il le sait car il a reçu deux messages issus du site 3

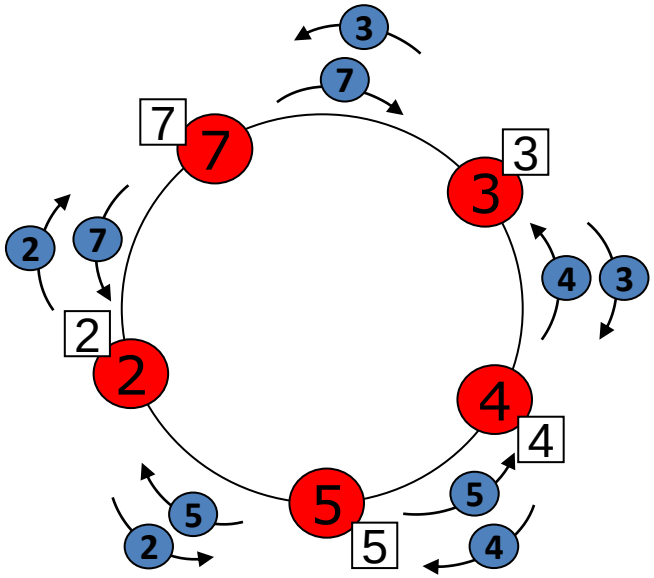
6. Tour 3 : le site 2 envoie donc un message de confirmation

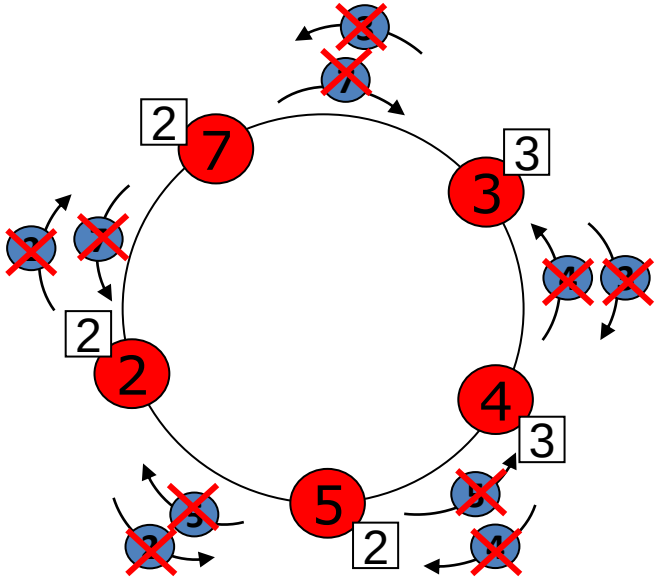
7. Lorsque ce message de confirmation a fait le tour de l'anneau, le processus d'élection est terminé.

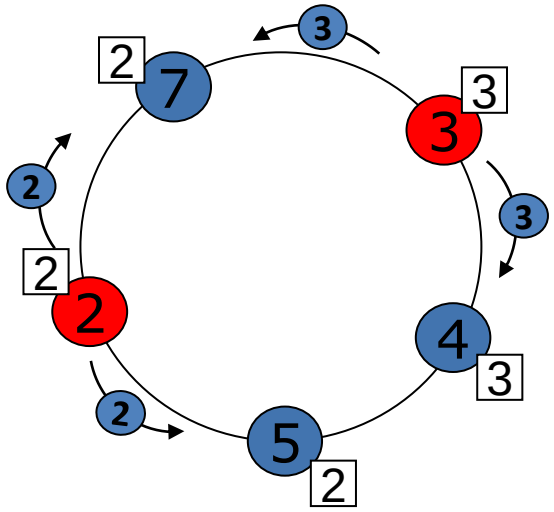


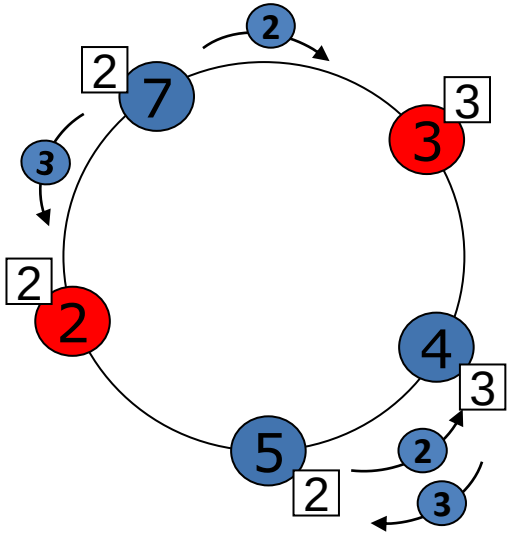
Franklin 1982

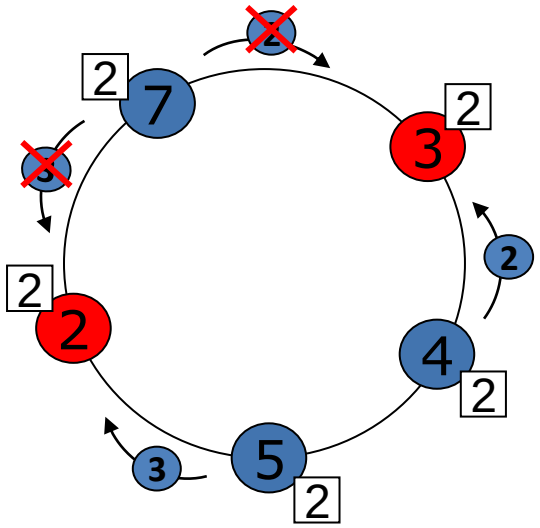




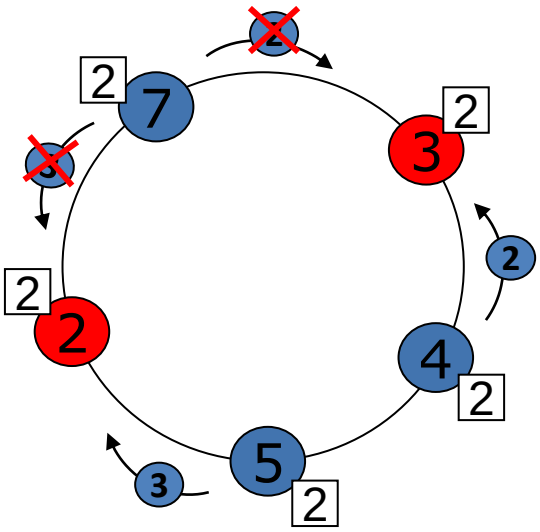


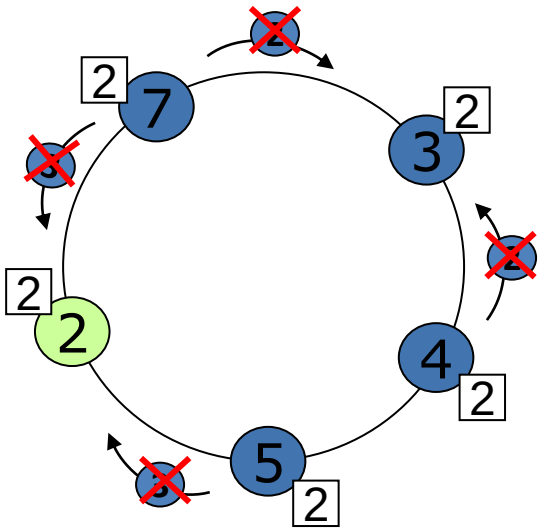


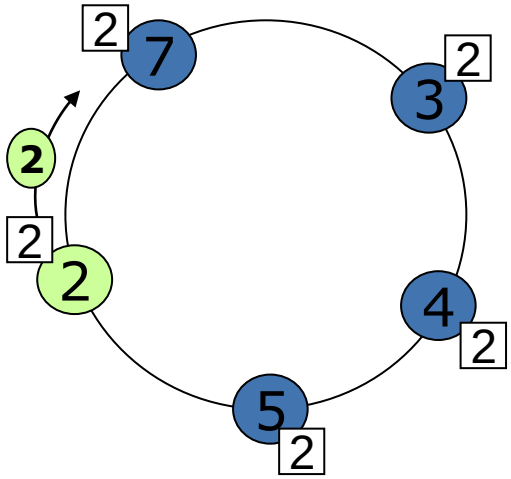


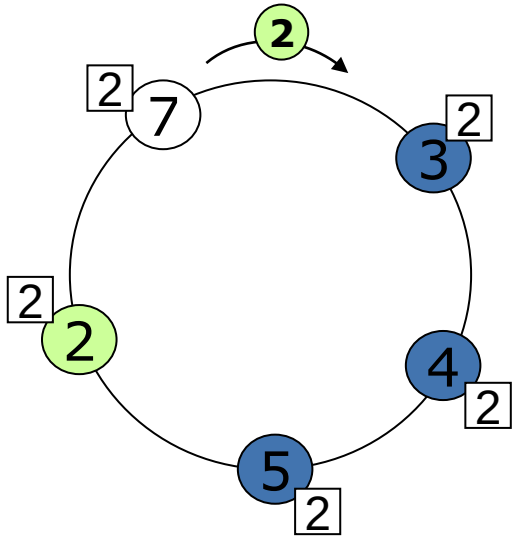


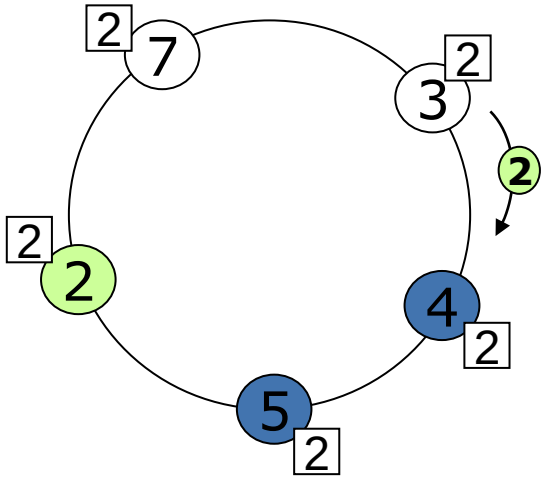
Franklin 1982

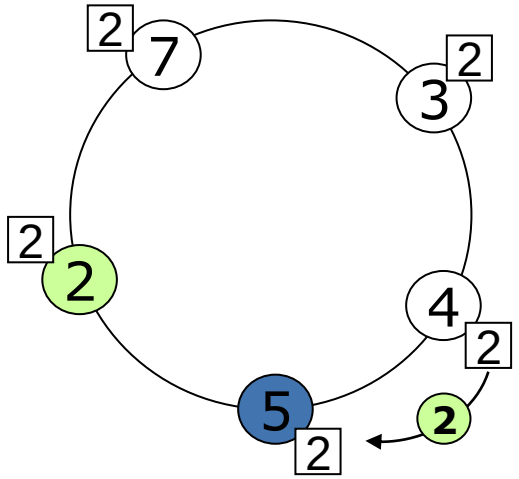




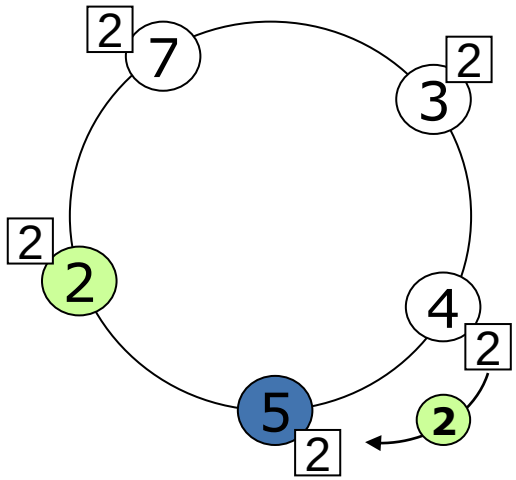


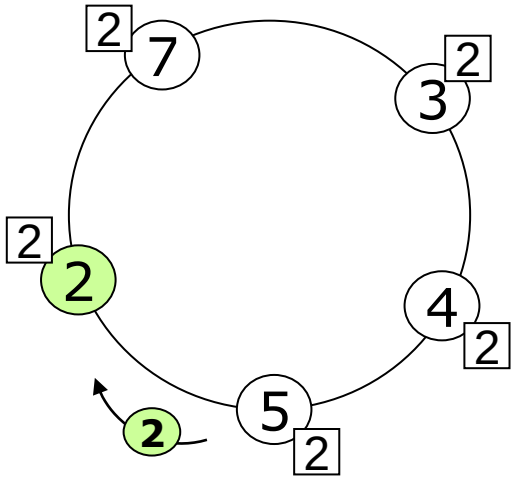


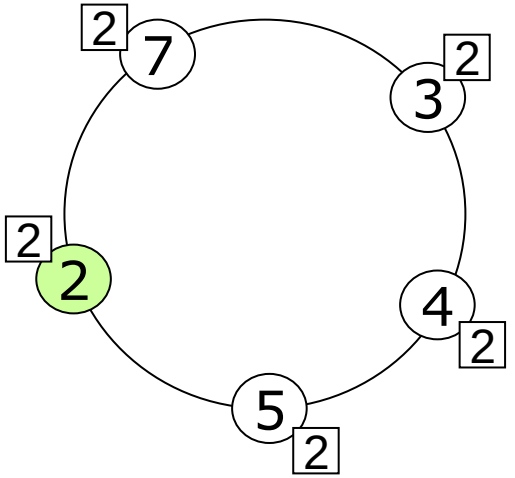




Franklin 1982







Franklin 1982

- Pb : Asynchronisme du réseau
  - Un site peut être en avance d'un tour sur un candidat voisin
    - Soit S_0, S_1, S_2, S_3 et S_4 des candidats "voisins" à un tour donné, S_3 peut avoir éliminé S_2 et S_4 et avoir envoyé ses messages du tour suivant alors que S_1 attend encore la réponse de S_0
    - Il faut que S_1 conserve ce message en avance pour le traiter au tour suivant
    - Un site ne peut recevoir qu'un message en avance et qu'il le reçoit dans la même direction que le premier message reçu

- Variables du site S_i

1. $suivant_i$: constante contenant l'identité du site successeur de S_i sur l'anneau
2. $précédent_i$: constante contenant l'identité du site prédécesseur de S_i sur l'anneau

Franklin 1982

3. $état_i$: état du service. Elle prend une valeur parmi (repos, en_cours, attente, terminé). Elle est initialisée à repos

4. $nbreq_i$: nombre de requêtes reçues au cours d'un tour

5. $conc_i$: identité du site dont on reçoit la première des deux requêtes du tour courant. Lorsqu'elle vaut S_i, cela signifie qu'aucune requête en avance n'est reçue.

6. dir_i : booléen indiquant la direction d'où vient la première des deux requêtes du tour courant

7. $concav_i$: identité du site dont on reçoit la première des deux requêtes du tour suivant

8. $chef_i$: identité du site (provisoirement) élu.

Franklin 1982

- Algorithme du site S_i

Si aucun processus d'élection n'a atteint le site S_i, le service initialise un tel processus

Au cours de ce processus, il propose sa candidature dans les deux sens de l'anneau tant qu'il n'est pas éliminé ou qu'il ne sait pas qu'il est le seul survivant

Dans tous les cas, il attend que le processus d'élection soit terminé pour renvoyer l'identité du site élu

A chaque changement de tour, il faut prendre garde à traiter l'éventuelle requête en avance.

Franklin 1982

```
Leader()
Début
   Si (état_i== repos) Alors
          état_i=en_cours;chef_i=i;concav_i=i;
          Répéter
              nbreq_i=0;
              Si concav_i!=i Alors
                     nbreq_i=1;
                     conc_i=concav_i;
                     Si (conc_i<chef_i)Alors
    chef_i=conc_i;
                     Finsi
                     concav_i=i;
              Finsi
              envoyer_à(suivant_i,(req,i));
              envoyer_à(précédent_i,(req,i));
              Attendre(nbreq_i==2);
          Jusqu'à(état_i != en_cours);
```

Franklin 1982

```
        Répéter
            .
            .
            .
        Jusqu'à(état_i != en_cours);
        Si concav_i!=i Alors
            Si (dir_i) Alors
                envoyer_à(suivant_i,(req,concav_i));
            Sinon
                envoyer_à(précédent_i,(req,concav_i));
            Finsi
        Finsi
    Finsi
    Attendre(état_i==terminé);
    renvoyer(chef_i);
Fin
```

Franklin 1982

A la réception d'une requête, on met à jour si nécessaire l'identité provisoire du leader

Dans le cas où on est un initiateur survivant (état à en_cours), on enregistre les deux requêtes voisines en prenant garde à mémoriser une requête en avance

Sur la réception de la deuxième, on teste si on a survécu puis si on est le seul survivant

Dans le cas où on est en attente, on retransmet les messages

```
sur_réception_de(j,(req,k))
Début
  Si (état_i == repos || k < chef_i ) Alors  chef_i = k;
  Finsi
```

Franklin 1982

```
Si (état_i == en_cours)   Alors
 Si (nbreq_i==0) Alors conc_i=k; nbreq_i=1; dir_i= (j== précédent_i);
 Sinon si ((dir_i && j==précédent_i)||(!diri && j!=précédent_i))
     Alors
         concav_i=k;
       sinon
         nbreq_i=2;
         si (chef_i<i) Alors état_i=attente;
                     sinon   si (conc_i==i || k==conc_i) Alors
                      état_i = terminé;
                      envoyer_à(suivant_i,(conf,i));
                         finsi
                     finsi
         finsi
        Finsi
Sinon
  état_i=attente;
  Si (j==précédent_i) Alors
      envoyer_à(suivant_i,(req,k));
  Sinon
      envoyer_à(précédent_i,(req,k));
  Finsi
  Finsi
Fin
```

Franklin 1982

La confirmation du site élu fait un tour de l'anneau

```
sur_réception_de(j,(conf,k))
Début
    Si (i!=k) Alors
          envoyer_à(suivant_i,(conf,k));
          état_i=terminé;
    Finsi
Fin
```

Complexité de l'algorithme

	Nombre de message	temps
Au mieux	*O(n)*	*O(n)*
Au pire	*O(nlog(n))*	*O(n)*
En moyenne	*O(nlog(n))*	*O(n)*