

Ordonnancement des évènements

- [Ordonnancement par séquenceur](#)
 - [Le privilège](#)
 - [Séquenceur circulant - Anneau virtuel](#)
 - [Séquenceur circulant - Variables d'état](#)
 - [Séquenceur circulant - Panne d'un processus](#)
 - [Séquenceur circulant - Réinsertion d'un processus](#)
 - [Séquenceur circulant - Le jeton](#)
 - [Séquenceur sur une voie à diffusion](#)
 - [En cas de panne du site \$i\$](#)
-

Ordonnancement par séquenceur

Le séquenceur évite les trous dans la numérotation.

Un séquenceur S délivre une valeur entière ≥ 0 .

Fonction $ticket(S)$.

Deux propriétés :

P1 : si a et b deux opérations $ticket(S)$ alors $a > b$ ou $b > a$.

P2 : si a est une exécution de $t = ticket(S)$ alors t est le nombre d'opérations $ticket(S)$ qui ont précédé a .

P1 est l'EM sur $ticket(S)$ et P2 est l'absence de trous.

Le privilège

Mise en oeuvre générale d'un séquenceur = faire circuler un **privilège** entre les sites.

Privilège = valeur courante du séquenceur.

Quand site i possède le privilège, il peut faire $ticket(S)$.

⇒ Séquenceur circulant.

⇒ Séquenceur sur une voie à diffusion.

Séquenceur circulant - Anneau virtuel

Définir un itinéraire pour le parcours du privilège.

Méthode simple = chaque site communique avec 2 sites voisins.

Un numéro unique est attribué à chaque site.

Chaque site i a un voisin successeur $suiv(i)$ et un prédécesseur $pred(i)$.

⇒ anneau virtuel.

$$suiv[i]=i+1 \text{ mod } N$$

$$pred[i]=i-1 \text{ mod } N$$

Le privilège tourne toujours dans le même sens.

En cas de panne d'un site \Rightarrow reconstruction de l'anneau \Rightarrow **reconfiguration** (mise à jour des variables *suiv* et *pred*).

Site défaillant remis en route \Rightarrow réinsertion.

Séquenceur circulant - Variables d'état

1- Fonctionnement sans pannes.

Soit K ($K > 1$).

Sur chaque site i , P_i assure la circulation du privilège.

Chaque P_i a une variable $S[i]$ ($0 \leq S[i] \leq K-1$).

$S[i]$ peut être modifiée par P_i et lue par $P_{suiv(i)}$.

P_i possède le privilège quand :

$$S[i] \neq S[i-1] \text{ pour } i \neq 0$$

$$S[0] = S[N-1] \text{ pour } i = 0$$

Lorsque P_i possède le privilège il doit l'abandonner au bout d'un temps fini :

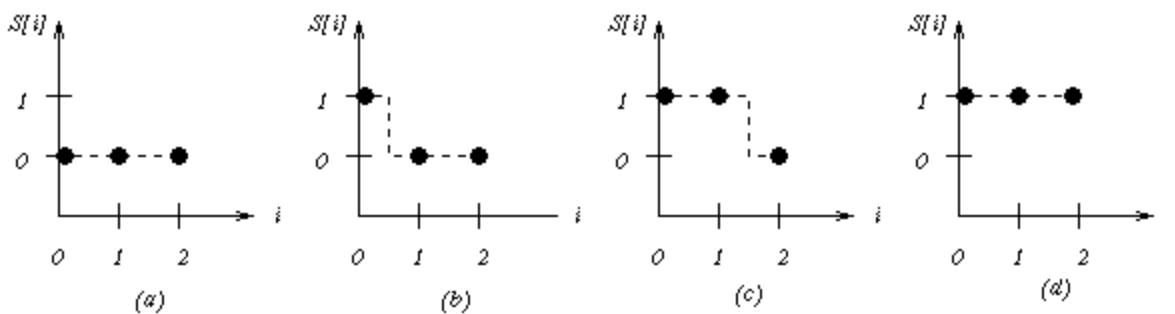
$$S[i] := S[i-1] \text{ pour } i \neq 0$$

$$S[0] := S[0] + 1 \text{ mod } K \text{ pour } i = 0$$

Équité = chaque processus reçoit le privilège à tour de rôle.

Exemple avec $N=3$, $K=2$ et $S=000$.

Évolution du système = 000 - 1 00 - 11 0 - 111 - 0 11 - 00 1



Séquenceur circulant - Panne d'un processus

Processus $\neq P_0$ \Rightarrow

1- aucun nouveau front est créé.

2- disparition d'un front \Rightarrow régénération par P_0 .

Processus P_0 .

\Rightarrow impossible de régénérer le front.

Le processus qui joue le rôle de P_0 est le seul tel que $pred[i] > i$.

Chaque processus est programmé pour jouer le rôle de P_0 .

```
si (pred[i] > i)
alors
  << Comportement selon P_0 >>
sinon
  << Comportement selon P_i >>
```

Séquenceur circulant - Réinsertion d'un processus

1- Reconfigurer l'anneau virtuel.

2- réinitialiser $S[i]$.

Il faut que j ($j = suiv(i)$) n'autorise pas la lecture de $S[j]$ pendant la SC.

```
si (j < i) alors
  S[i] := S[j]-1 mod K
sinon
  S[i] := S[j]
```

Séquenceur circulant - Le jeton

Variable d'état = théorie.

Le privilège = 1 **jeton** qui circule sur l'anneau.

Le jeton porte la valeur v .

Avant la transmission du jeton :

```
S[j] := v pour j  $\neq$  0
v := v+1 mod K; S[j] := v pour j = 0
```

À chaque passage du jeton sur j , une horloge de garde est armée.

Si l'horloge se déclenche, j consulte $S[i]$ ($i = pred(j)$).

```
si (j > i) et (S[j]  $\neq$  S[i])
ou si (j < i) et (S[j] = S[i])
Alors le jeton est considéré comme perdu.
```

j le régénère avec $S[i]$ et réarme son horloge.

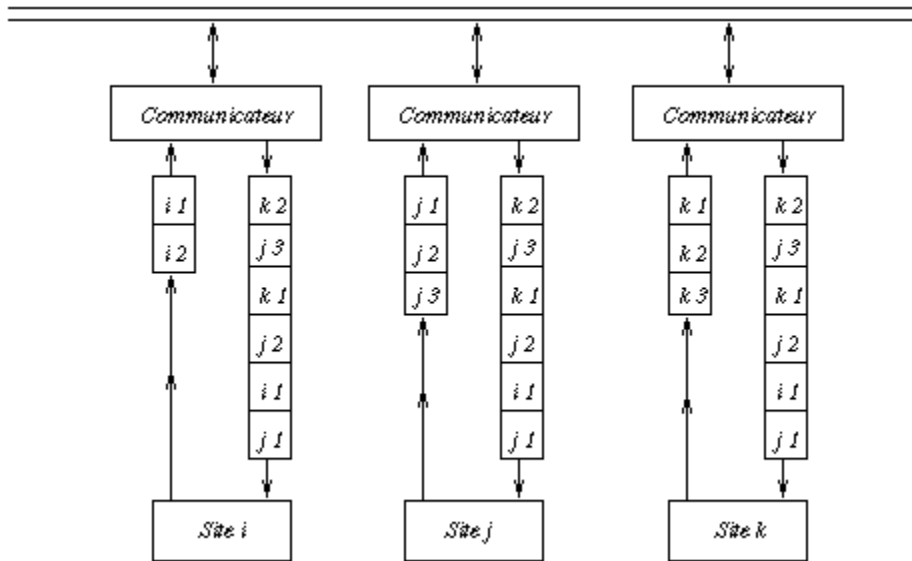
Séquenceur sur une voie à diffusion

1 site accède au médium grâce à un **communicateur**.

1 message envoyé par i est diffusé à tous les sites (y compris i).

Tous les communicateurs classent les messages dans le même ordre.

⇒ ordonnancement **global**.



Chaque site i a un compteur local cpt_i .

- $ticket(S)$ = requête dans file d'entrée.
 i attend que la requête apparaisse dans la file de sortie.
- le site i traite toutes les requêtes $ticket(S)$.
Si site = j alors cpt_i++ sinon utiliser ticket puis cpt_i++ .

En cas de panne du site i

Les autres sites continuent à fonctionner.

Pour se réinsérer, i envoie une requête de réinsertion à j .

j renvoie un message de synchronisation.

Quand le message apparaît dans la file de sortie :

- j envoie à i la valeur de cpt_j ;
- i :
 - ignore tous les messages $<$ synchro,
 - conserve tous les messages qui suivent (sans les traiter) jusqu'à la valeur de cpt_j ,
 - met à jour son compteur.



[Retour au sommaire.](#)