Algorithmes de redistribution de données pour anneaux de processeurs hétérogènes

Hélène Renard, Yves Robert, Frédéric Vivien

projet *GRAAL* : CNRS, ENS Lyon et INRIA
Algorithmique et ordonnancement pour plates-formes hétérogènes distribuées

6 avril 2005

Plan de l'exposé

Plan de l'exposé

Contexte de redistribution

Plates-formes ciblées : plates-formes distribuées hétérogènes (réseau de stations de travail, clusters, clusters de clusters, grilles, etc.)

1. Différentes variations : ressources ou besoins de l'application.
2. Les données doivent être redistribuées afin d'obtenir un meilleur équilibrage de charge.
3. Pas de discussion du mécanisme d'équilibrage de charge : nous le considérons comme extérieur.

Pourquoi des anneaux ?

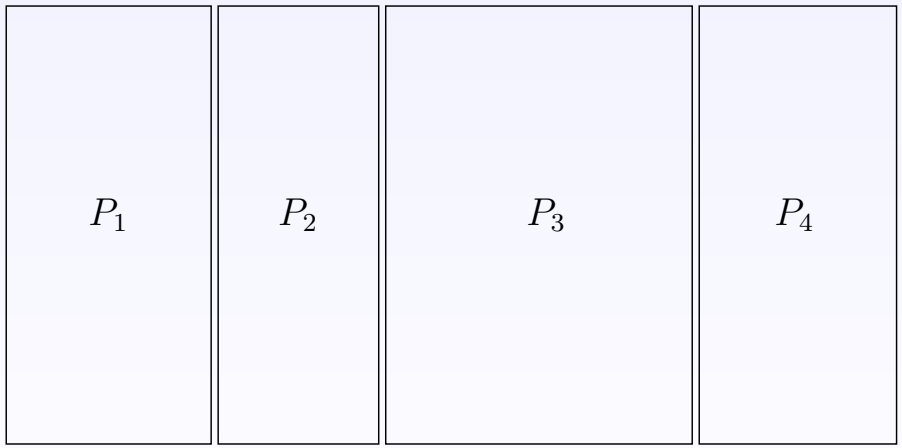
Les anneaux sont importants pour les applications traitant des données dont l'ordre doit être préservé.

Exemple : matrice de données pour laquelle chaque processeur traite une tranche de colonnes consécutives.

Pourquoi des anneaux ?

Les anneaux sont importants pour les applications traitant des données dont l'ordre doit être préservé.

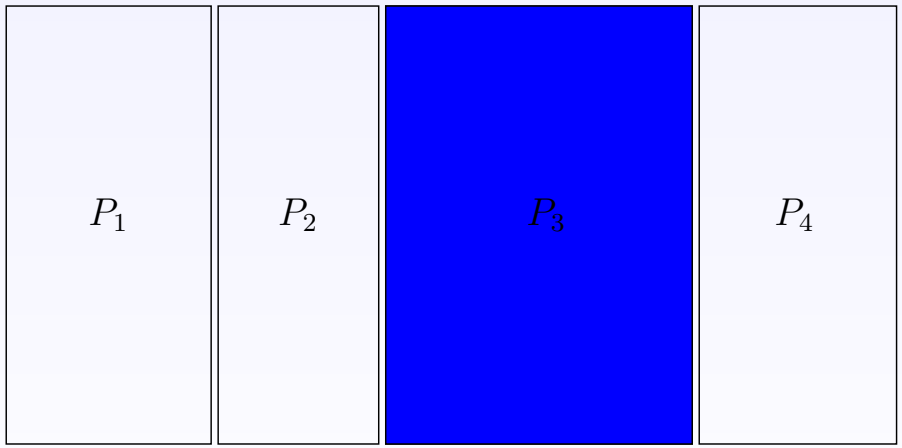
Exemple : matrice de données pour laquelle chaque processeur traite une tranche de colonnes consécutives.



Pourquoi des anneaux ?

Les anneaux sont importants pour les applications traitant des données dont l'ordre doit être préservé.

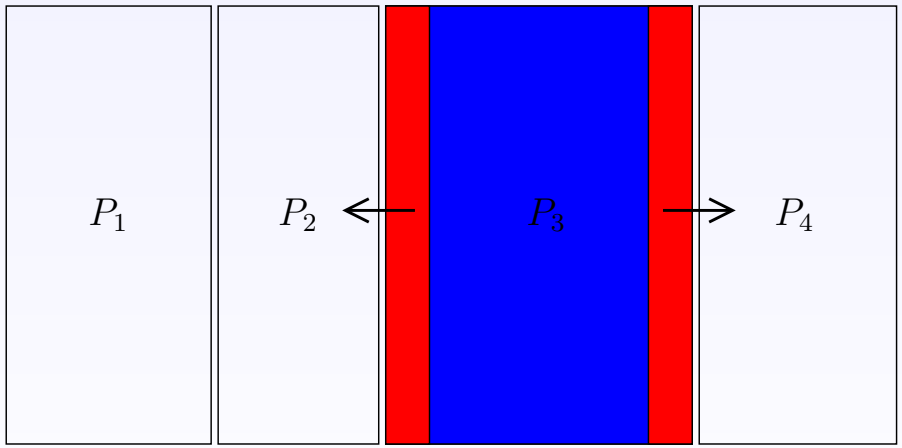
Exemple : matrice de données pour laquelle chaque processeur traite une tranche de colonnes consécutives.



Pourquoi des anneaux ?

Les anneaux sont importants pour les applications traitant des données dont l'ordre doit être préservé.

Exemple : matrice de données pour laquelle chaque processeur traite une tranche de colonnes consécutives.



Pourquoi des anneaux ?

Les anneaux sont importants pour les applications traitant des données dont l'ordre doit être préservé.

Exemple : matrice de données pour laquelle chaque processeur traite une tranche de colonnes consécutives.

Une distribution uni-dimensionnelle de données induit un arrangement uni-dimensionnel des processeurs.

Pourquoi des anneaux ?

Les anneaux sont importants pour les applications traitant des données dont l'ordre doit être préservé.

Exemple : matrice de données pour laquelle chaque processeur traite une tranche de colonnes consécutives.

Une distribution uni-dimensionnelle de données induit un arrangement uni-dimensionnel des processeurs.

Nous considérons des anneaux :

- uni et bi-directionnels.
- homogènes et hétérogènes.

Notations et hypothèses

- Processeurs : P_1, ..., P_n.
- Initialement, le processeur P_i possède L_i données.

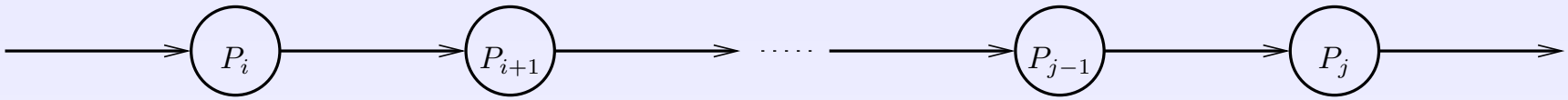
  δ_i est le déséquilibre de P_i : après redistribution, P_i possédera $L_i - \delta_i$ données.

  Loi de conservation des données : $\sum_i \delta_i = 0$

  Hypothèses concernant le déséquilibre :
  - Un processeur possède au moins une donnée avant la redistribution : $L_i \geq 1$.
  - Un processeur possède au moins une donnée après la redistribution : $L_i \geq 1 + \delta_i$.
- $c_{i,i+1}$: temps nécessaire pour l'envoi d'une donnée de P_i à P_{i+1}.
- Modèle 1-port : un processeur ne peut envoyer qu'à un seul processeur à la fois. (De même en réception)

Plan de l'exposé

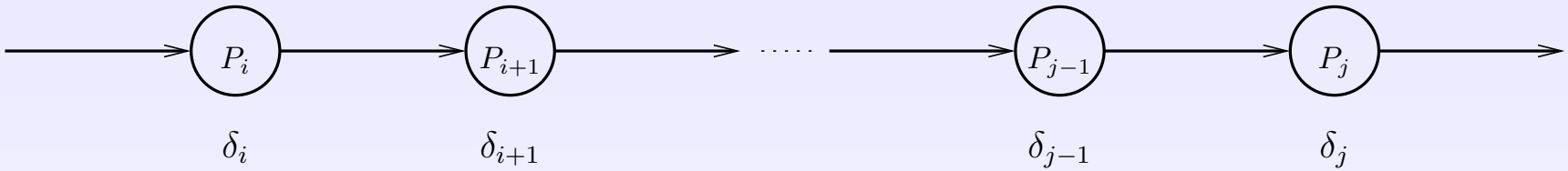
Cadre de travail



Capacité homogène des liens de communication : c.

P_i ne peut envoyer des données qu'à P_{i+1}.

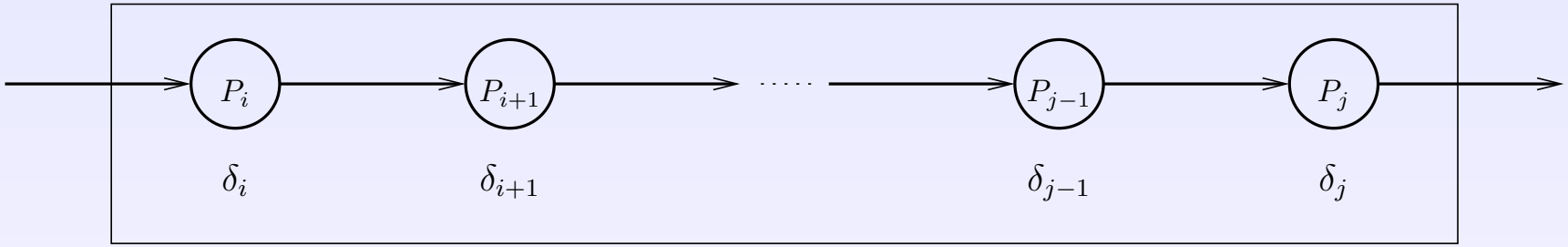
Borne inférieure du temps de redistribution



Capacité homogène des liens de communication : c.

P_i ne peut envoyer des données qu'à P_{i+1}.

Borne inférieure du temps de redistribution

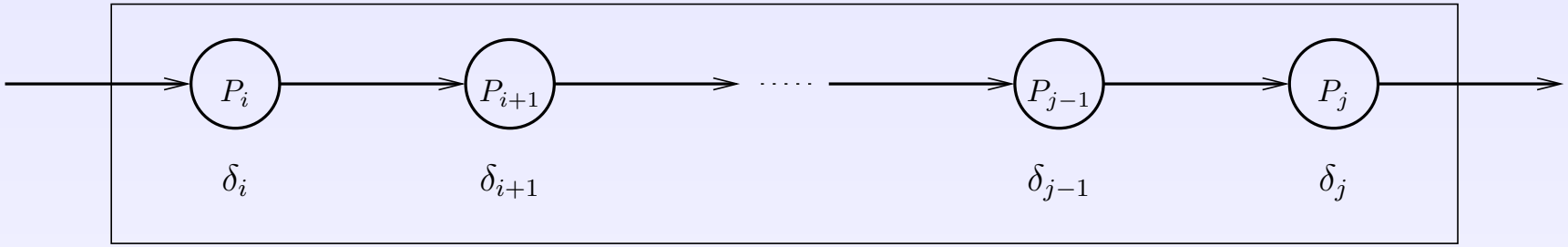


$$\delta_{i,j} = \delta_i + \delta_{i+1} + ... + \delta_{j-1} + \delta_j$$

Capacité homogène des liens de communication : c.

P_i ne peut envoyer des données qu'à P_{i+1}.

Borne inférieure du temps de redistribution



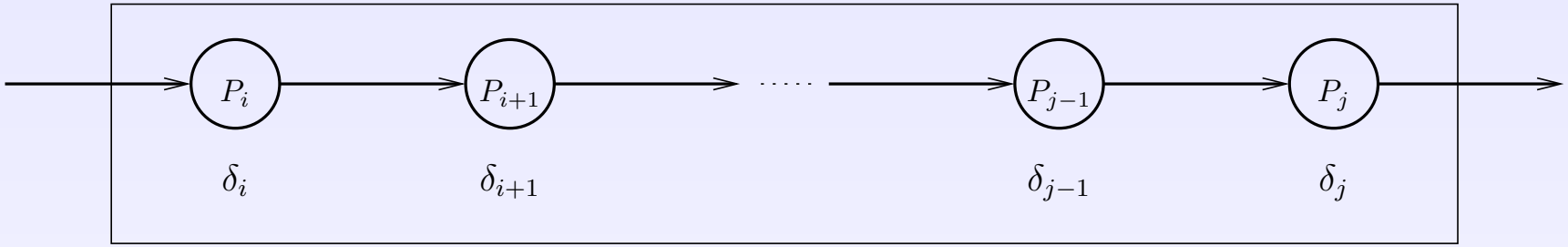
$$\delta_{i,j} = \delta_i + \delta_{i+1} + ... + \delta_{j-1} + \delta_j$$

Capacité homogène des liens de communication : c.

P_i ne peut envoyer des données qu'à P_{i+1}.

Cela prend à P_j au moins $\delta_{i,j} \times c$ unités de temps pour envoyer $\delta_{i,j}$ données (si $\delta_{i,j} > 0$).

Borne inférieure du temps de redistribution



$$\delta_{i,j} = \delta_i + \delta_{i+1} + ... + \delta_{j-1} + \delta_j$$

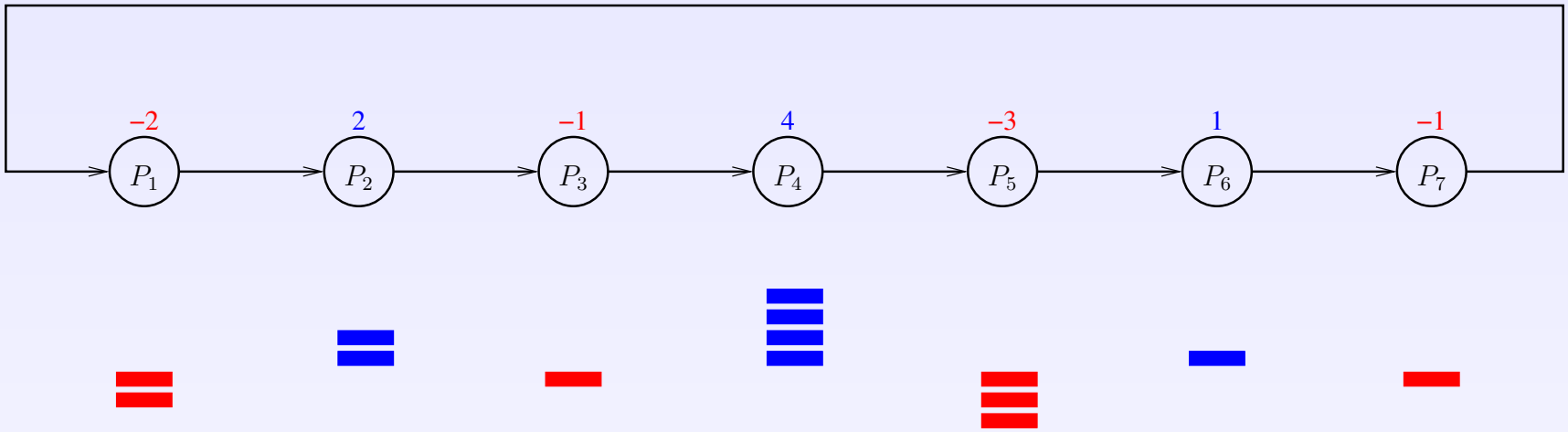
Capacité homogène des liens de communication : c.

P_i ne peut envoyer des données qu'à P_{i+1}.

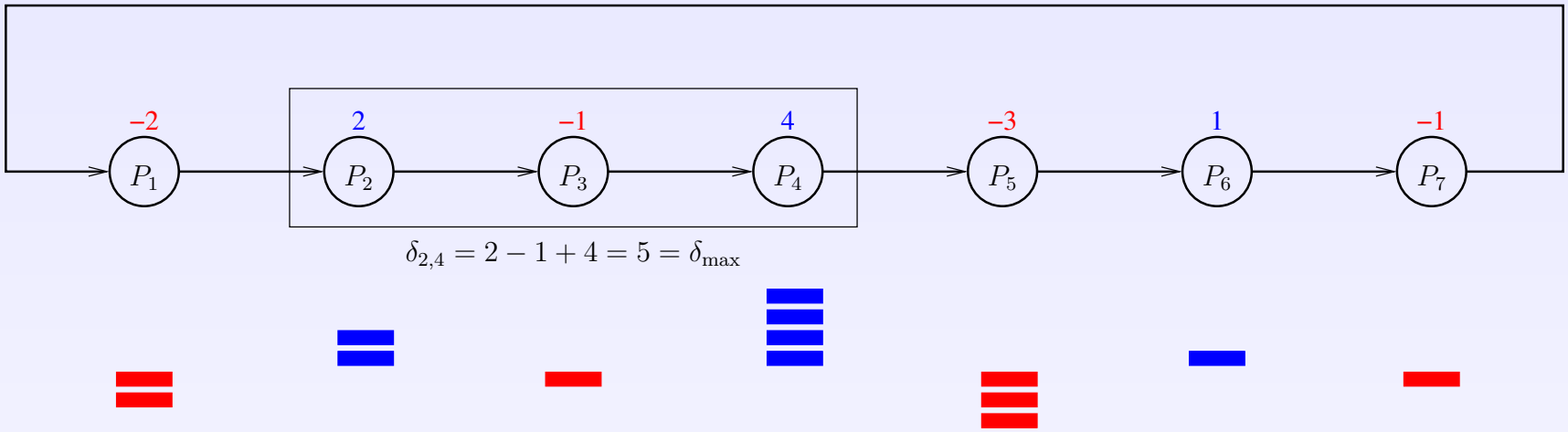
Cela prend à P_j au moins $\delta_{i,j} \times c$ unités de temps pour envoyer $\delta_{i,j}$ données (si $\delta_{i,j} > 0$).

Borne inférieure : $$\left(\max_{1 \leq i \leq n,\ 1 \leq j \leq n} \delta_{i,j} \right) \times c$$

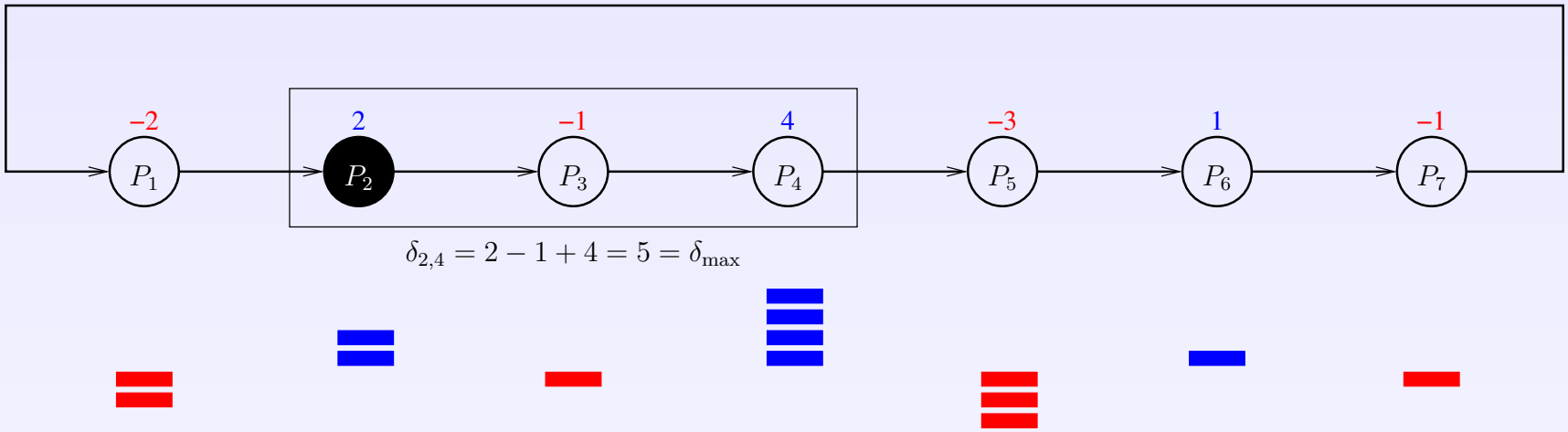
Algorithme de redistribution



Algorithme de redistribution



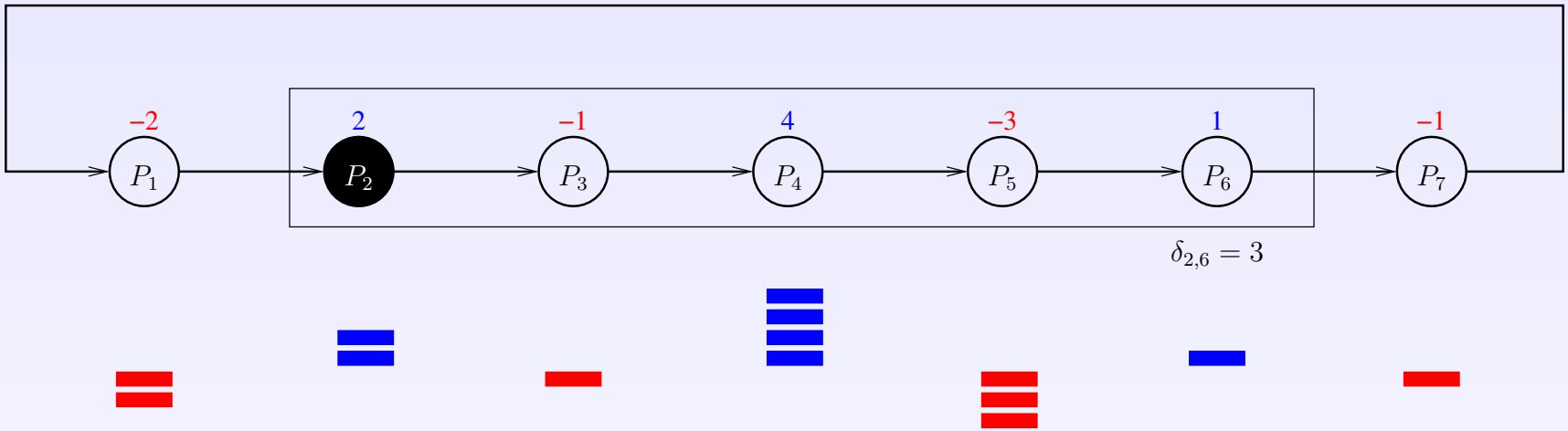
Algorithme de redistribution



$\delta_{\max} = 5$

L'algorithme complet de redistribution est défini par le premier processeur de déséquilibre maximal.

Algorithme de redistribution

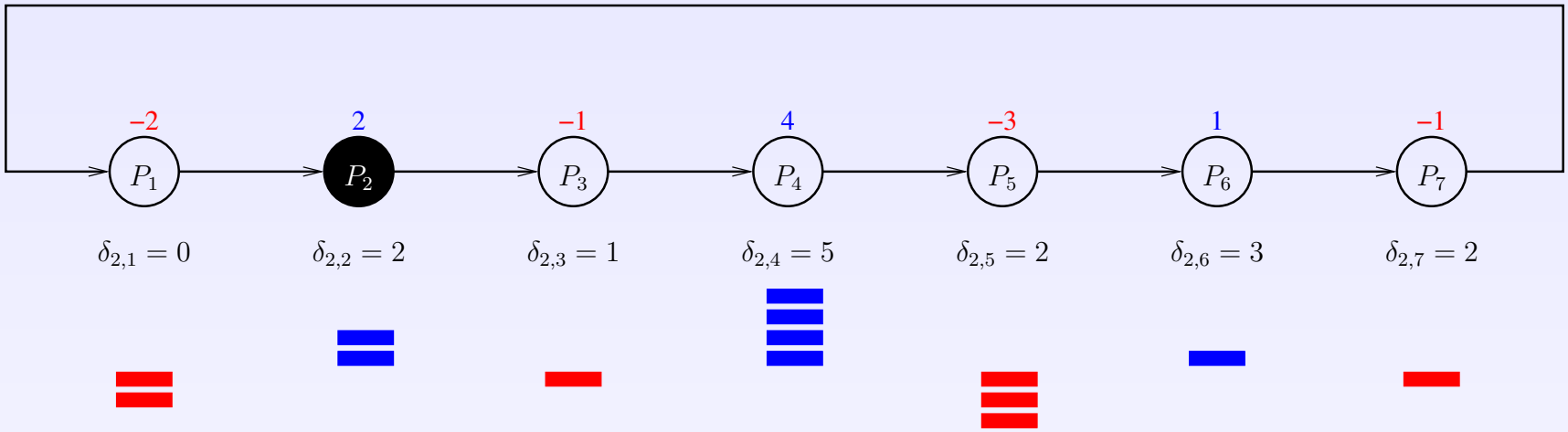


$\delta_{\text{max}} = 5$

L'algorithme complet de redistribution est défini par le premier processeur de déséquilibre maximal.

Pendant l'exécution de l'algorithme, le processeur P_i envoie $\delta_{2,i}$ données.

Algorithme de redistribution

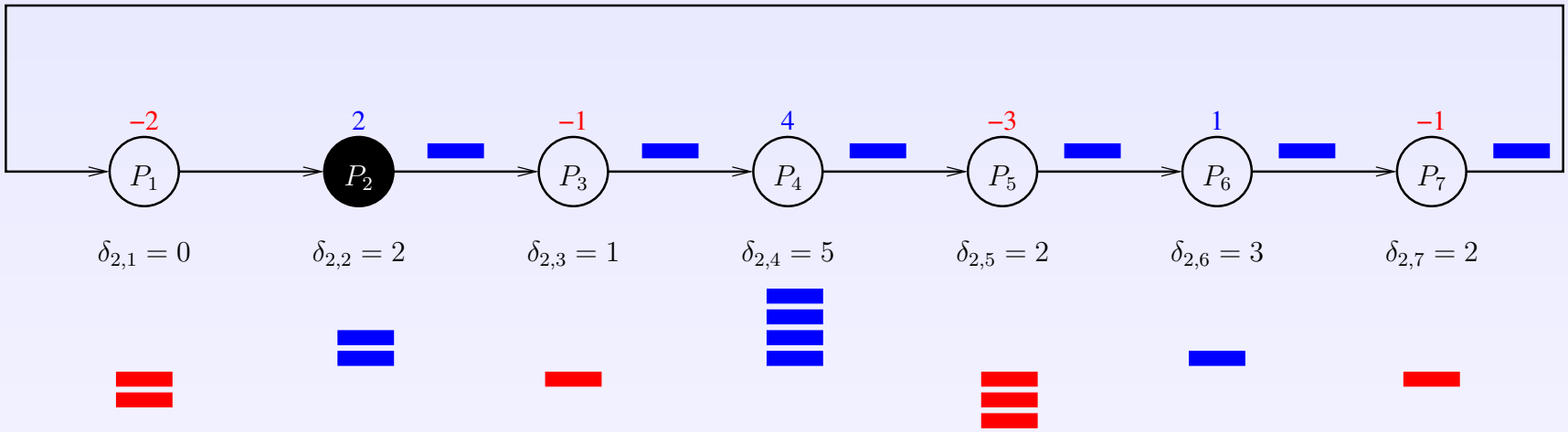


$\delta_{\text{max}} = 5$

L'algorithme complet de redistribution est défini par le premier processeur de déséquilibre maximal.

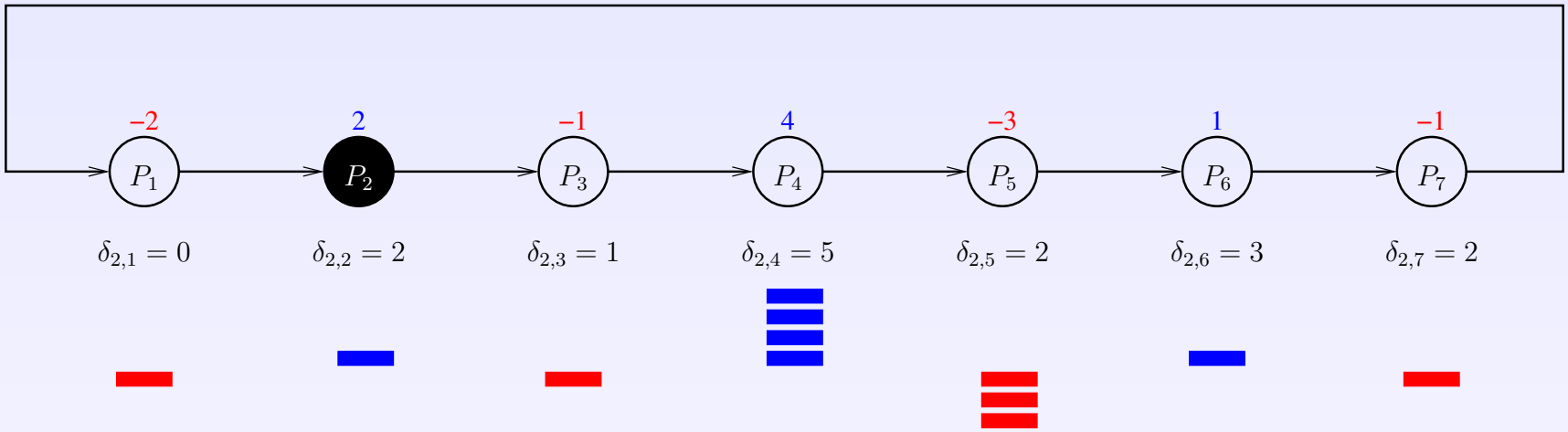
Pendant l'exécution de l'algorithme, le processeur P_i envoie $\delta_{2,i}$ données.

Algorithme de redistribution



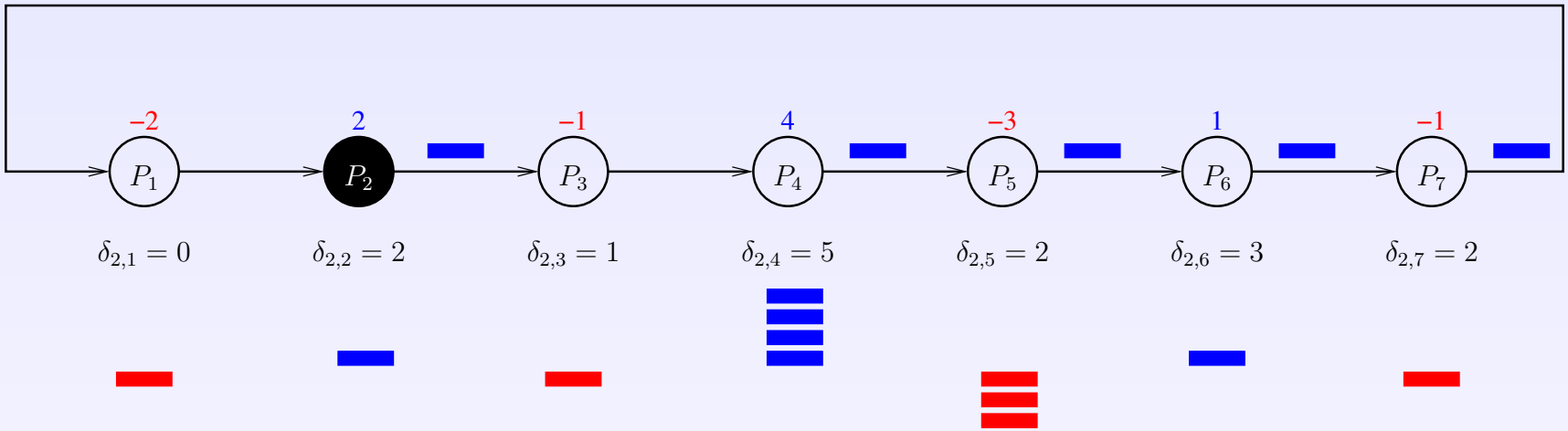
À l'étape 1, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 1$

Algorithme de redistribution



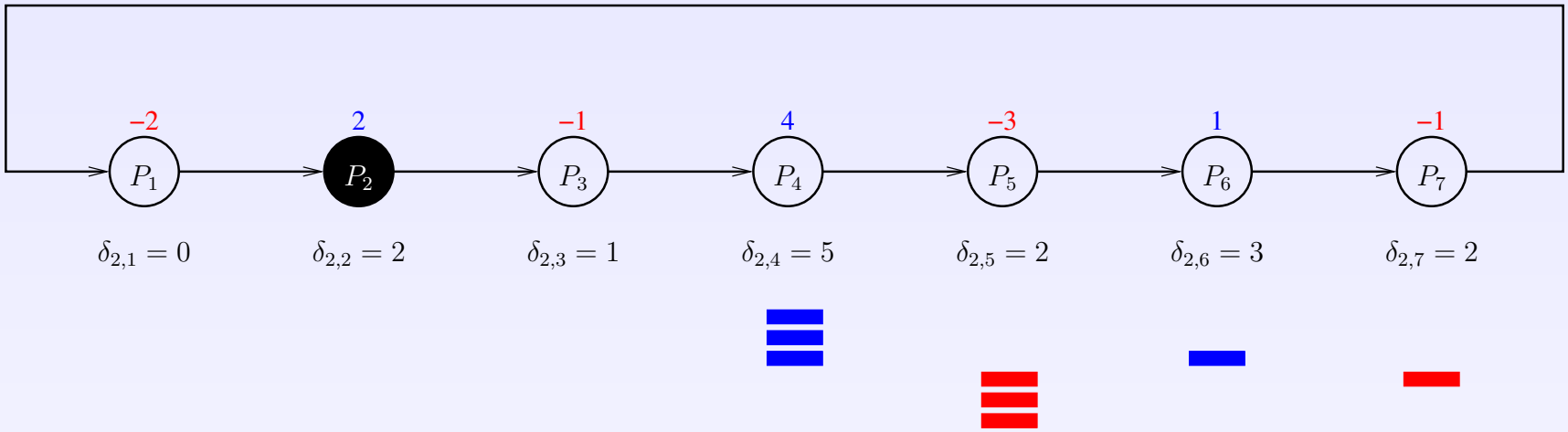
À l'étape 1, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 1$

Algorithme de redistribution



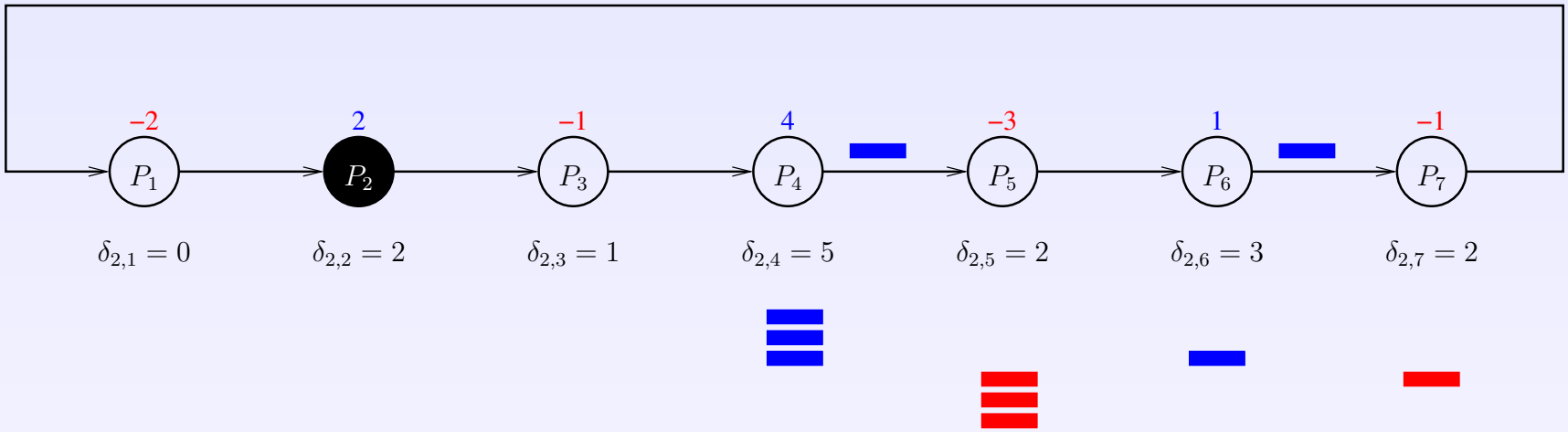
À l'étape 2, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 2$

Algorithme de redistribution



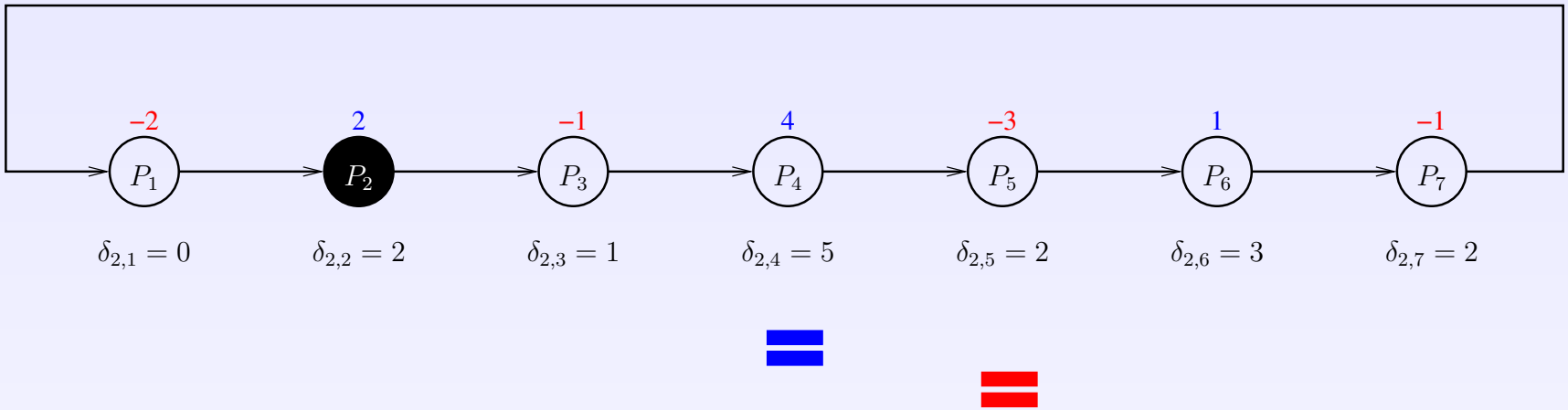
À l'étape 2, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 2$

Algorithme de redistribution



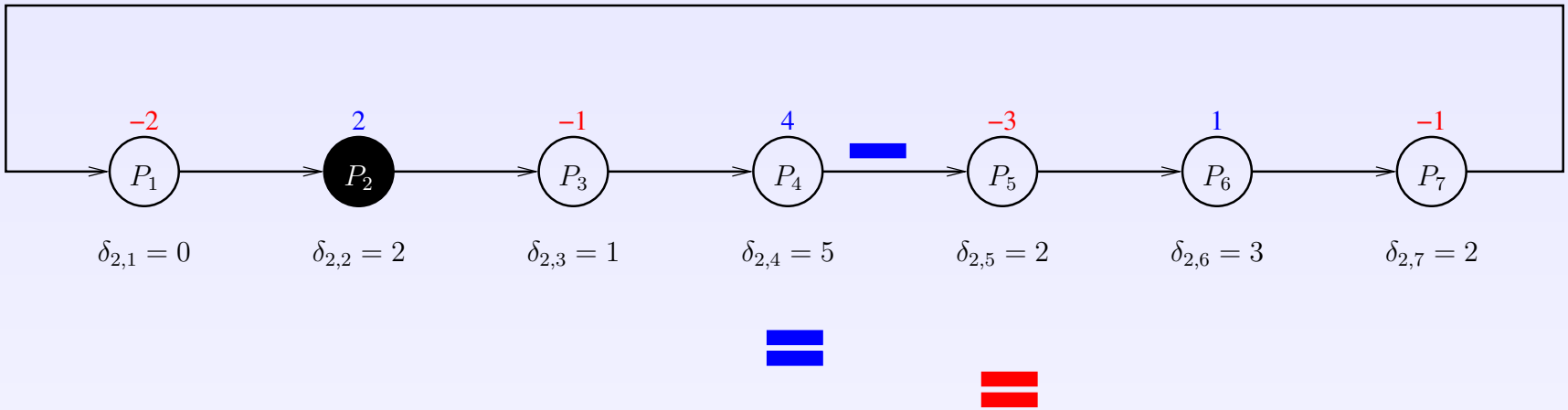
À l'étape 3, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 3$

Algorithme de redistribution



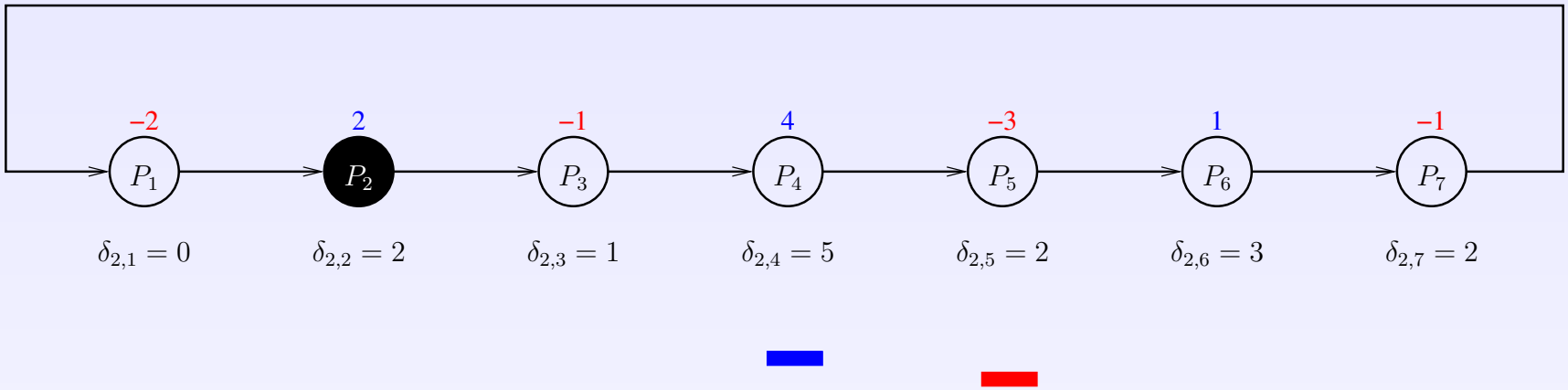
À l'étape 3, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 3$

Algorithme de redistribution



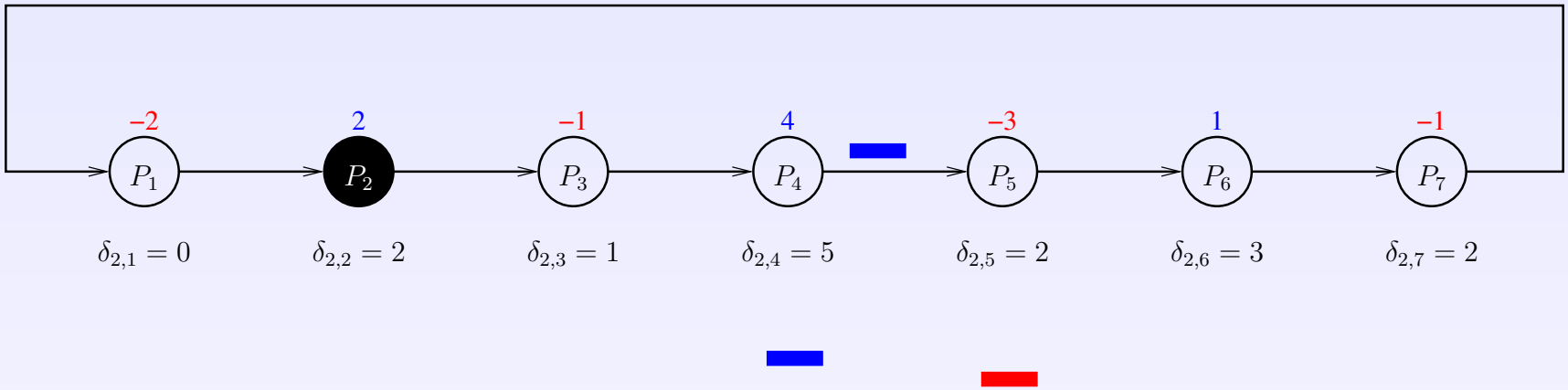
À l'étape 4, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 4$

Algorithme de redistribution



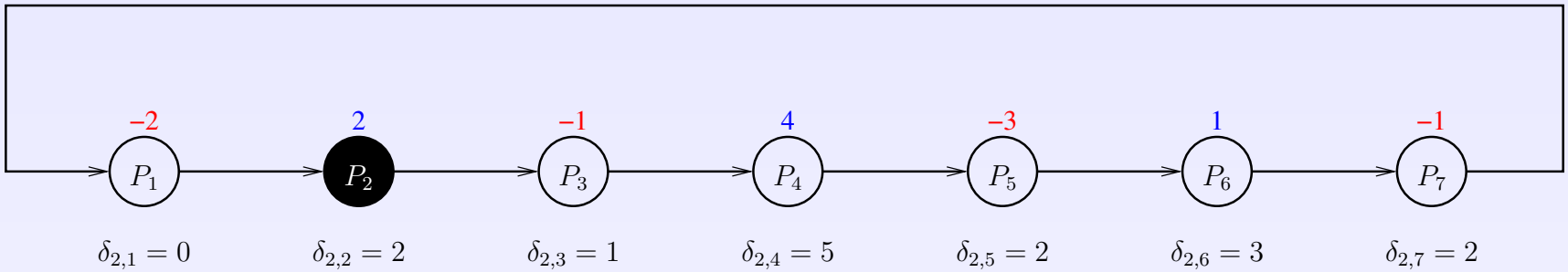
À l'étape 4, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 4$

Algorithme de redistribution



À l'étape 5, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 5$

Algorithme de redistribution



À l'étape 5, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 5$

L'algorithme

1: Soit $\delta_{\text{max}} = (\max_{1 \leq k \leq n, 0 \leq l \leq n-1} |\delta_{k,k+l}|)$
2: Soient `start` et `end` deux indices tels que la tranche $C_{\texttt{start,end}}$ est de déséquilibre maximal : $\delta_{\texttt{start,end}} = \delta_{\text{max}}$.
3: **Pour** $s = 1$ à δ_{max} **:**
4: **Pour tout** $l = 0$ à $n - 1$ **:**
5: **Si** $\delta_{\texttt{start,start}+l} \geq s$ **Alors**
6: $P_{\texttt{start}+l}$ envoie à $P_{\texttt{start}+l+1}$ une donnée pendant l'intervalle de temps $[(s-1) \times c, s \times c[$

Théorème

Cet algorithme de redistribution est optimal.

Plan de l'exposé

Borne inférieure du temps de redistribution

Le processeur P_i a besoin de $c_{i,i+1}$ unités de temps pour envoyer une donnée au processeur P_{i+1}.

Borne inférieure du temps de redistribution

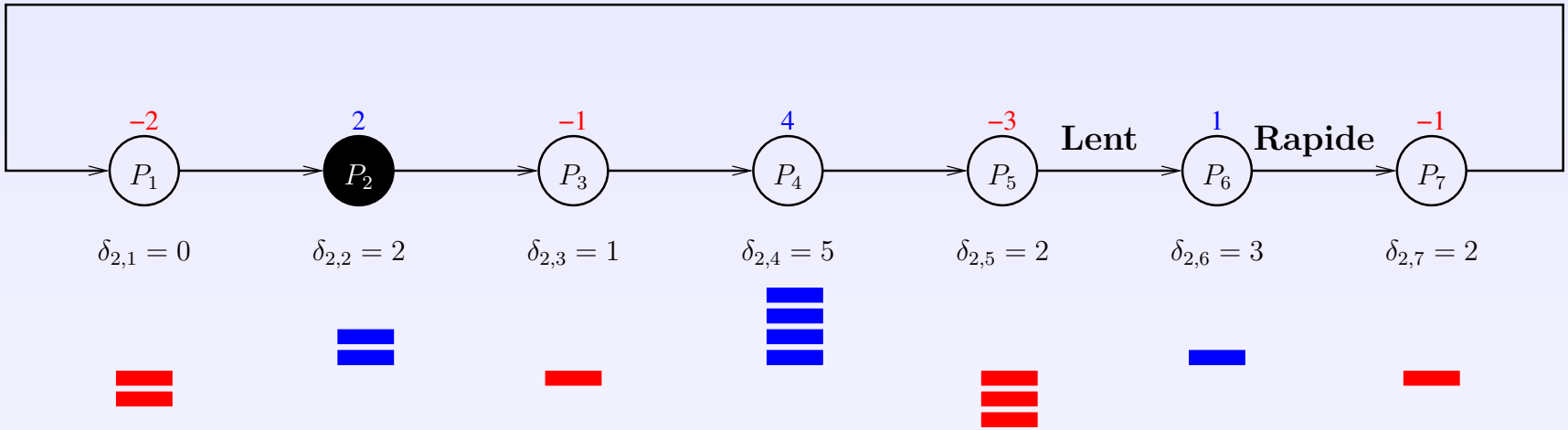
Le processeur P_i a besoin de $c_{i,i+1}$ unités de temps pour envoyer une donnée au processeur P_{i+1}.

La borne inférieure est définie comme dans le cas de l'anneau homogène unidirectionnel.

Cela prend à P_j au moins $\delta_{i,j} \times c_{j,j+1}$ unités de temps pour envoyer $\delta_{i,j}$ données (si $\delta_{i,j} > 0$).

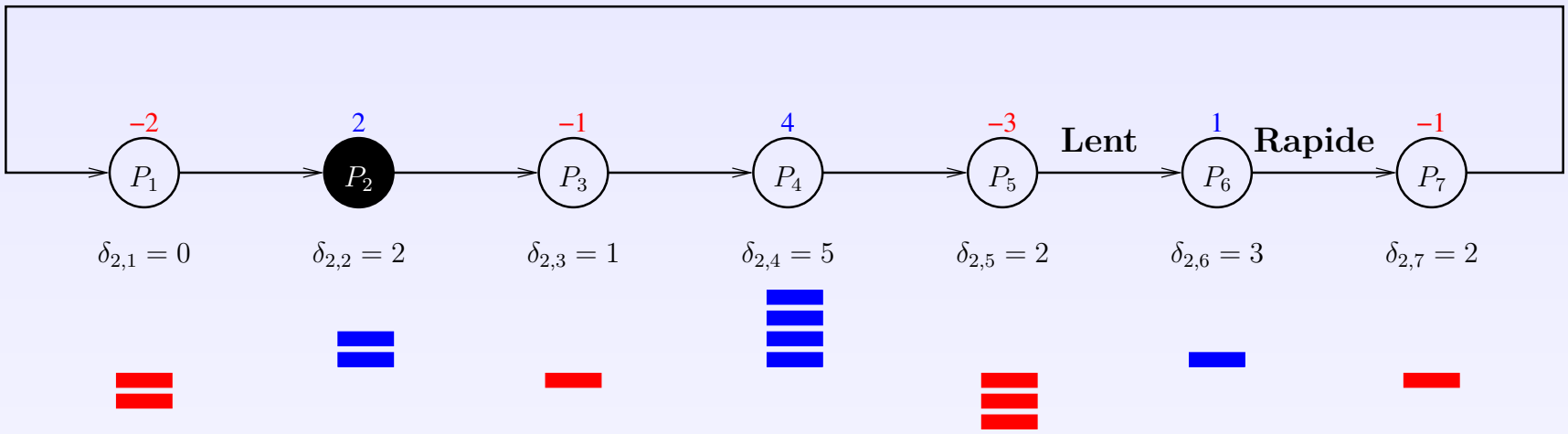
Borne inférieure : $$\max_{1 \leq i \leq n,\ 1 \leq j \leq n} \delta_{i,j} \times c_{j,j+1}$$

Conséquences de l'hétérogénéité des bandes passantes



P_6 peut avoir à attendre des données de P_5 afin d'envoyer toutes les données nécessaires à P_7.

Conséquences de l'hétérogénéité des bandes passantes



P_6 peut avoir à attendre des données de P_5 afin d'envoyer toutes les données nécessaires à P_7.

Le temps d'exécution de P_6 ne peut pas être exprimé avec une formule simple.

L'algorithme de redistribution est asynchrone.

L'algorithme de redistribution

C'est simplement une version asynchrone de l'algorithme précédent.

1: Soit $\delta_{\text{max}} = (\max_{1 \leq k \leq n, 0 \leq l \leq n-1} |\delta_{k,k+l}|)$
2: Soient `start` et `end` deux indices tels que la tranche $C_{\texttt{start},\texttt{end}}$ est de déséquilibre maximal : $\delta_{\texttt{start},\texttt{end}} = \delta_{\text{max}}$.
3: **Pour tout** $l = 0$ à $n - 1$ **:**
4: $P_{\texttt{start}+l}$ envoie $\delta_{\texttt{start},\texttt{start}+l}$ données une par une et dès que possible au processeur $P_{\texttt{start}+l+1}$

Optimalité

Lemme

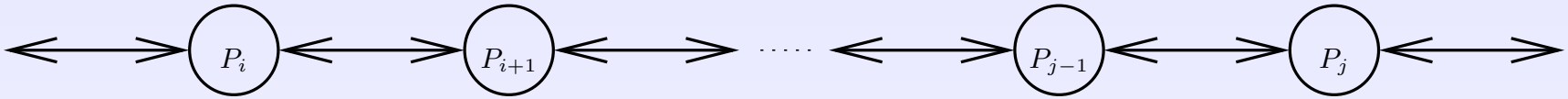
Le temps d'exécution de l'algorithme de redistribution est

$$\max_{1 \leq l \leq n} \delta_{start,l} \times c_{l,l+1}.$$

En d'autres termes, il n'y a aucun délai dû à l'attente, par un processeur, de la réception des données qu'il doit retransmettre.

Plan de l'exposé

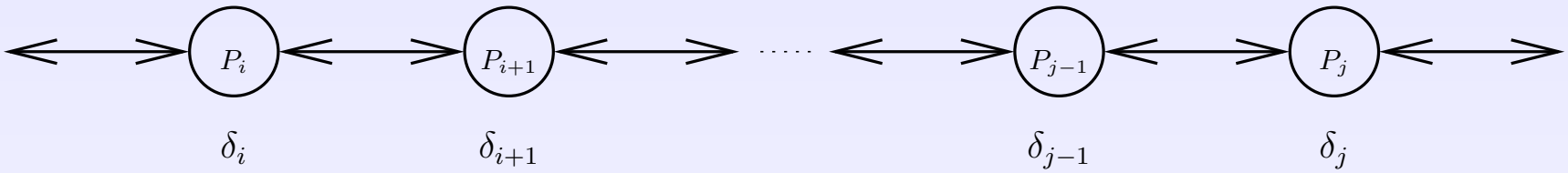
Cadre de travail



Capacité des liens de communication homogène : c.

Communications bi-directionnelles

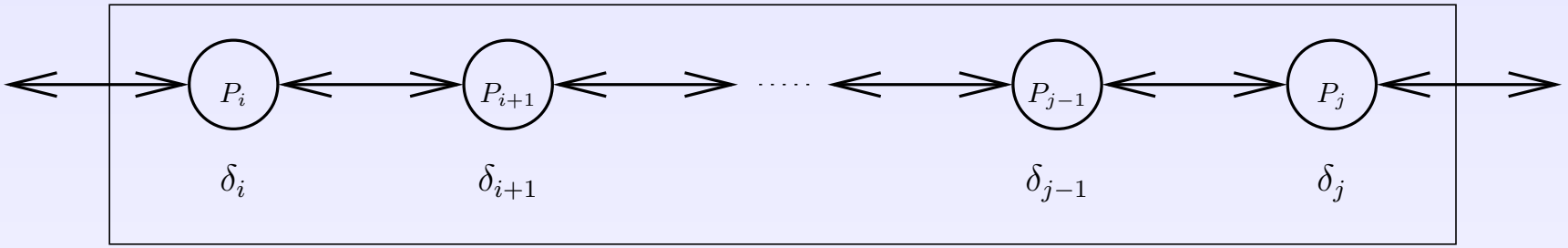
Borne inférieure du temps de redistribution



Capacité des liens de communication homogène : c.

Communications bi-directionnelles

Borne inférieure du temps de redistribution

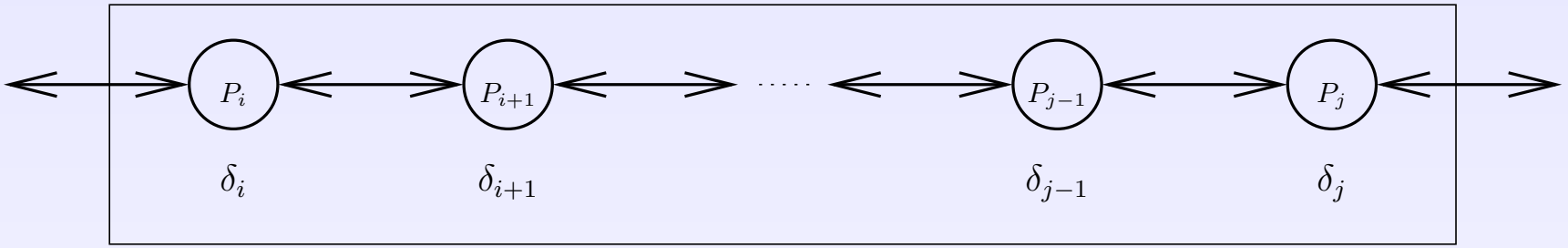


$$\delta_{i,j} = \delta_i + \delta_{i+1} + ... + \delta_{j-1} + \delta_j$$

Capacité des liens de communication homogène : c.

Communications bi-directionnelles

Borne inférieure du temps de redistribution

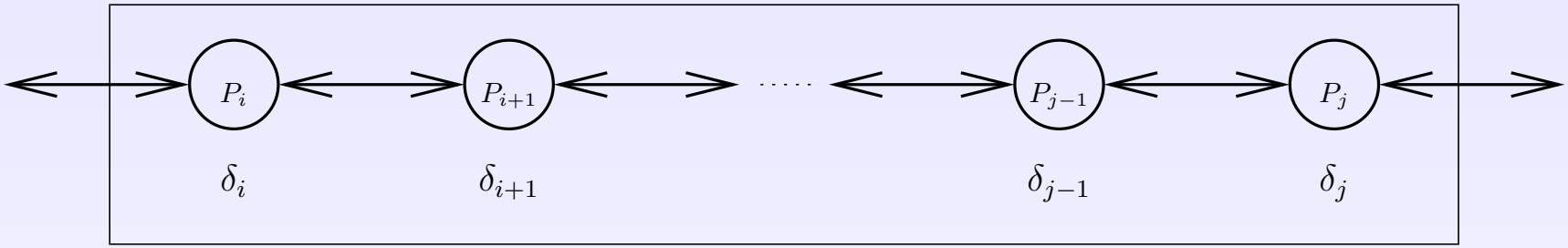


$$\delta_{i,j} = \delta_i + \delta_{i+1} + ... + \delta_{j-1} + \delta_j$$

Capacité des liens de communication homogène : c.

Cela prend au moins $\left\lceil \frac{\delta_{i,j}}{2} \right\rceil \times c$ unités de temps à un ensemble de processeurs $P_i, ..., P_j$ pour envoyer $\delta_{i,j}$ données (si $\delta_{i,j} > 0$).

Borne inférieure du temps de redistribution



$$\delta_{i,j} = \delta_i + \delta_{i+1} + ... + \delta_{j-1} + \delta_j$$

Capacité des liens de communication homogène : c.

Cela prend au moins $\left\lceil \frac{\delta_{i,j}}{2} \right\rceil \times c$ unités de temps à un ensemble de processeurs $P_i, ..., P_j$ pour envoyer $\delta_{i,j}$ données (si $\delta_{i,j} > 0$).

Borne inférieure : $$\max \left\{ \max_{1 \leq i \leq n} |\delta_i|, \max_{1 \leq i \leq n, 1 \leq j \leq n} \left\lceil \frac{|\delta_{i,j}|}{2} \right\rceil \right\} \times c$$

Intuitions de l'algorithme

1. Tout ensemble de processeurs contigus $P_i, ..., P_j$ tels que $\left\lceil \frac{|\delta_{i,j}|}{2} \right\rceil = \delta_{\text{max}}$ et $\delta_{i,j} \geq 0$ doit envoyer deux données à chaque étape de l'algorithme, une par chacune de ses extrémités.

2. Tout ensemble de processeurs contigus $P_i, ..., P_j$ tels que $\left\lceil \frac{|\delta_{i,j}|}{2} \right\rceil = \delta_{\text{max}}$ et $\delta_{i,j} \leq 0$ doit recevoir deux données à chaque étape de l'algorithme, une par chacune de ses extrémités.

3. Soit P_i tel que $|\delta_i| = \delta_{\max}$.
   Si P_i est déjà impliqué dans une communication (due aux cas précédents) : tout est arrangé.
   Sinon, nous avons la liberté de qui P_i recevra une donnée (cas $\delta_i \leq 0$), ou à qui P_i enverra une donnée (cas $\delta_i \geq 0$).
   De manière à simplifier l'algorithme, toutes les communications ont lieu dans le sens des indices croissants « de P_i vers P_{i+1} ».

Optimalité

Théorème

L'algorithme totalement décrit dans les actes de conférence est optimal.

La principale difficulté : manipulation de tous les cas spéciaux (effets de bord).

Plan de l'exposé

Borne inférieure

Même type de démonstration que précédemment.

$$\max \left\{ \begin{array}{l} \displaystyle\max_{1\leq k\leq n,\ \delta_k>0} \delta_k \min\{c_{k,k-1}, c_{k,k+1}\} \\ \\ \displaystyle\max_{1\leq k\leq n,\ \delta_k<0} -\delta_k \min\{c_{k-1,k}, c_{k+1,k}\} \\ \\ \displaystyle\max_{\substack{1\leq k\leq n,\\ 1\leq l\leq n-2,\\ \delta_{k,k+l}>0}} \ \min_{0\leq i\leq \delta_{k,k+l}} \max\{i\cdot c_{k,k-1}, (\delta_{k,k+l}-i)\cdot c_{k+l,k+l+1}\} \\ \\ \displaystyle\max_{\substack{1\leq k\leq n,\\ 1\leq l\leq n-2,\\ \delta_{k,k+l}<0}} \ \min_{0\leq i\leq -\delta_{k,k+l}} \max\{i\cdot c_{k-1,k}, -(\delta_{k,k+l}+i)\cdot c_{k+l+1,k+l}\} \end{array} \right.$$

Redistribution légère : définition

Définition

Une redistribution est légère si chaque processeur possède initialement toutes les données qu'il devra envoyer pendant l'exécution de l'algorithme.

$\mathcal{S}_{i,j}$: quantité de données envoyées par P_i à son voisin P_j pendant la redistribution.

Condition mathématique de la redistribution légère :

$$\mathcal{S}_{i,i+1} + \mathcal{S}_{i,i-1} \leq L_i.$$

Redistribution légère : résolution

MINIMISER τ, AVEC LES CONDITIONS :

$$\begin{cases} \mathcal{S}_{i,i+1} \geq 0 & 1 \leq i \leq n \\ \mathcal{S}_{i,i-1} \geq 0 & 1 \leq i \leq n \\ \mathcal{S}_{i,i+1} + \mathcal{S}_{i,i-1} - \mathcal{S}_{i+1,i} - \mathcal{S}_{i-1,i} = \delta_i & 1 \leq i \leq n \\ \mathcal{S}_{i,i+1} c_{i,i+1} + \mathcal{S}_{i,i-1} c_{i,i-1} \leq \tau & 1 \leq i \leq n \\ \mathcal{S}_{i+1,i} c_{i+1,i} + \mathcal{S}_{i-1,i} c_{i-1,i} \leq \tau & 1 \leq i \leq n \end{cases}$$

Redistribution légère : résolution

MINIMISER τ, AVEC LES CONDITIONS :

$$\begin{cases} \mathcal{S}_{i,i+1} \geq 0 & 1 \leq i \leq n \\ \mathcal{S}_{i,i-1} \geq 0 & 1 \leq i \leq n \\ \mathcal{S}_{i,i+1} + \mathcal{S}_{i,i-1} - \mathcal{S}_{i+1,i} - \mathcal{S}_{i-1,i} = \delta_i & 1 \leq i \leq n \\ \mathcal{S}_{i,i+1} c_{i,i+1} + \mathcal{S}_{i,i-1} c_{i,i-1} \leq \tau & 1 \leq i \leq n \\ \mathcal{S}_{i+1,i} c_{i+1,i} + \mathcal{S}_{i-1,i} c_{i-1,i} \leq \tau & 1 \leq i \leq n \end{cases}$$

Lemme

N'importe quelle solution du système précédent est faisable.

Redistribution légère : résolution

MINIMISER τ, AVEC LES CONDITIONS :

$$\begin{cases} \mathcal{S}_{i,i+1} \geq 0 & 1 \leq i \leq n \\ \mathcal{S}_{i,i-1} \geq 0 & 1 \leq i \leq n \\ \mathcal{S}_{i,i+1} + \mathcal{S}_{i,i-1} - \mathcal{S}_{i+1,i} - \mathcal{S}_{i-1,i} = \delta_i & 1 \leq i \leq n \\ \mathcal{S}_{i,i+1} c_{i,i+1} + \mathcal{S}_{i,i-1} c_{i,i-1} \leq \tau & 1 \leq i \leq n \\ \mathcal{S}_{i+1,i} c_{i+1,i} + \mathcal{S}_{i-1,i} c_{i-1,i} \leq \tau & 1 \leq i \leq n \end{cases}$$

Lemme

N'importe quelle solution du système précédent est faisable.

Lemme

L'une des deux approximations entières évidentes de la solution rationnelle du système précédent est une solution optimale entière.

Cas général

Ouvert.

Plan de l'exposé

Conclusion

Algorithmes optimaux pour les anneaux

1. Homogènes uni-directionnels.
2. Hétérogènes uni-directionnels.
3. Homogènes bi-directionnels.
4. Hétérogènes bi-directionnels (redistribution légère).

Les redistributions optimisées nécessitent des phases dynamiques d'équilibrage de charge.