Loris MARCHAL
Laboratoire de l'Informatique du Parallélisme
Équipe Graal

Communications collectives et ordonnancement en régime permanent pour plates-formes hétérogènes

Thèse réalisée sous la direction d'Olivier BEAUMONT et d'Yves ROBERT

Parallélisme et nouvelles plates-formes de calcul

Pour répondre à une forte demande de calcul :

utilisation simultanée de plusieurs ressources de calcul

Évolution de plates-formes de calcul :

- super-calculateurs, grappes de calcul (cluster)
- moins coûteuses : réseau de stations
- grilles de calcul (grappes de grappes)

Parallélisme et nouvelles plates-formes de calcul

Pour répondre à une forte demande de calcul :

utilisation simultanée de plusieurs ressources de calcul

Évolution de plates-formes de calcul :

- super-calculateurs, grappes de calcul (cluster)
- moins coûteuses : réseau de stations
- grilles de calcul (grappes de grappes)

Parallélisme et nouvelles plates-formes de calcul

Pour répondre à une forte demande de calcul :

utilisation simultanée de plusieurs ressources de calcul

Évolution de plates-formes de calcul :

- super-calculateurs, grappes de calcul (cluster)
- moins coûteuses : réseau de stations
- grilles de calcul (grappes de grappes)

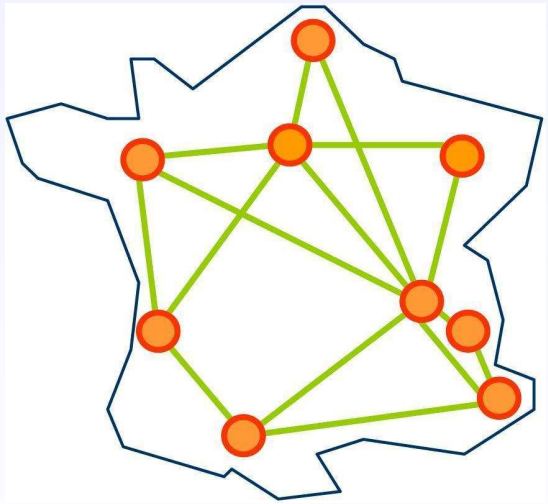
Parallélisme et nouvelles plates-formes de calcul

Pour répondre à une forte demande de calcul :

utilisation simultanée de plusieurs ressources de calcul

Évolution de plates-formes de calcul :

- super-calculateurs, grappes de calcul (cluster)
- moins coûteuses : réseau de stations
- grilles de calcul (grappes de grappes)



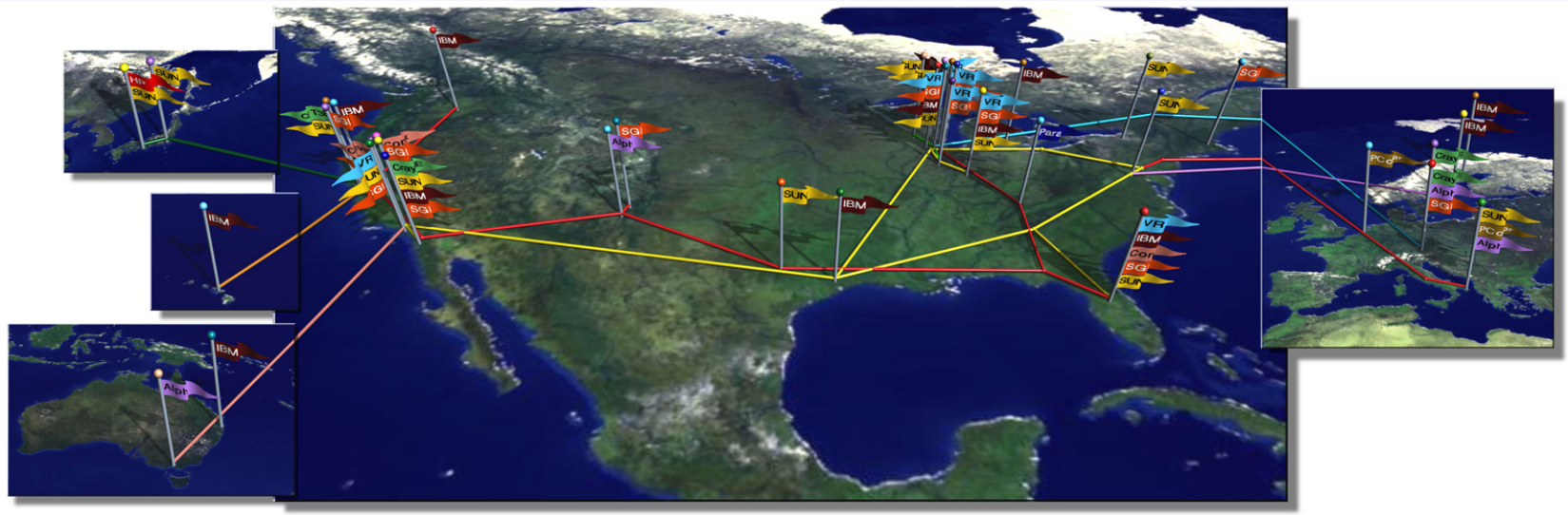
Parallélisme et nouvelles plates-formes de calcul

Pour répondre à une forte demande de calcul :

utilisation simultanée de plusieurs ressources de calcul

Évolution de plates-formes de calcul :

- super-calculateurs, grappes de calcul (cluster)
- moins coûteuses : réseau de stations
- grilles de calcul (grappes de grappes)



Parallélisme et nouvelles plates-formes de calcul

Pour répondre à une forte demande de calcul :

utilisation simultanée de plusieurs ressources de calcul

Évolution de plates-formes de calcul :

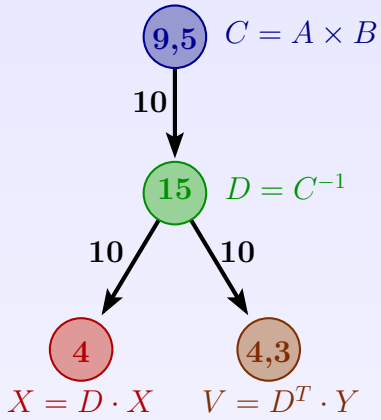
- super-calculateurs, grappes de calcul (cluster)
- moins coûteuses : réseau de stations
- grilles de calcul (grappes de grappes)

tendances : hétérogénéité et dynamicité

Techniques traditionnelles d'ordonnancement

Modélisation :

- Applications :
  graphe de tâches
- Plate-forme de calcul :
  processeurs homogènes ou hétérogènes
  réseau de communication avec ou sans congestion



Objectifs :

- allocation $\mathrm{alloc}(T)$: processeur traitant la tâche T
- ordonnancement $\sigma(T)$: date de début d'exécution de T.
- temps d'exécution total (*Makespan*) optimal

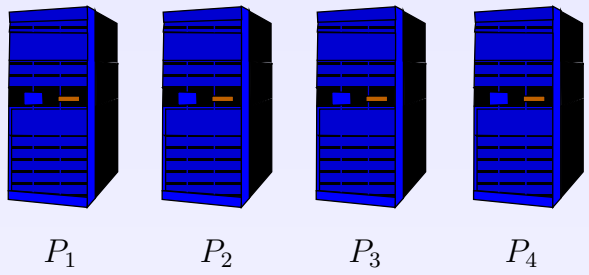
Résultats :

- problème NP-complet pour des tâches indépendantes
- algorithmes d'approximation (heuristiques de liste)

Techniques traditionnelles d'ordonnancement

Modélisation :

- Applications :
  graphe de tâches
- Plate-forme de calcul :
  processeurs homogènes ou hétérogènes
  réseau de communication avec ou sans congestion



Objectifs :

- allocation $\mathrm{alloc}(T)$: processeur traitant la tâche T
- ordonnancement $\sigma(T)$: date de début d'exécution de T.
- temps d'exécution total (*Makespan*) optimal

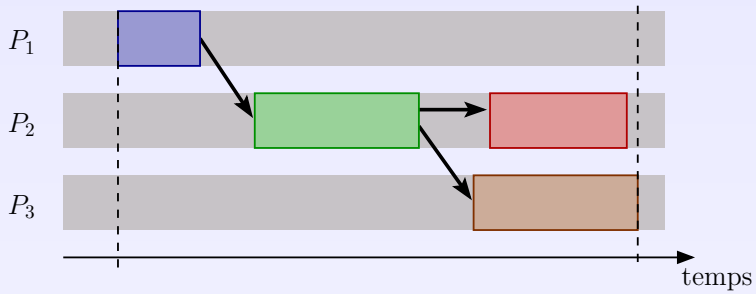
Résultats :

- problème NP-complet pour des tâches indépendantes
- algorithmes d'approximation (heuristiques de liste)

Techniques traditionnelles d'ordonnancement

Modélisation :

- Applications :
  graphe de tâches
- Plate-forme de calcul :
  processeurs homogènes ou hétérogènes
  réseau de communication avec ou sans congestion



Objectifs :

- allocation $\mathrm{alloc}(T)$: processeur traitant la tâche T
- ordonnancement $\sigma(T)$: date de début d'exécution de T.
- temps d'exécution total (*Makespan*) optimal

Résultats :

- problème NP-complet pour des tâches indépendantes
- algorithmes d'approximation (heuristiques de liste)

Techniques traditionnelles d'ordonnancement

Modélisation :

- Applications :
  graphe de tâches
- Plate-forme de calcul :
  processeurs homogènes ou hétérogènes
  réseau de communication avec ou sans congestion

Objectifs :

- allocation $\mathrm{alloc}(T)$: processeur traitant la tâche T
- ordonnancement $\sigma(T)$: date de début d'exécution de T.
- temps d'exécution total (*Makespan*) optimal

Résultats :

- problème NP-complet pour des tâches indépendantes
- algorithmes d'approximation (heuristiques de liste)

Conception d'ordonnancements efficaces

- Algorithme d'approximation
  exemple : 3/2-approximation
  optimal : 2 secondes $\rightsquigarrow$ 3 secondes
  optimal : 2 heures $\rightsquigarrow$ 3 heures
- Algorithme asymptotiquement optimal
  exemple : $T_{\text{opt}} + O(1)$
  optimal : 2 secondes $\rightsquigarrow$ 5 minutes + 2 secondes
  optimal : 2 heures $\rightsquigarrow$ 2 heures + 5 minutes

On se concentre sur un seul des problèmes soulevés :

concevoir des ordonnancements efficaces
pour les plates-formes hétérogènes à grande échelle.

On s'intéresse à des plates-formes statiques. La dynamicité des plates-formes émergentes reste à prendre en compte...

Conception d'ordonnancements efficaces

- Algorithme d'approximation
  exemple : 3/2-approximation
  optimal : 2 secondes $\leadsto$ 3 secondes
  optimal : 2 heures $\leadsto$ 3 heures
- Algorithme asymptotiquement optimal
  exemple : $T_{\text{opt}} + O(1)$
  optimal : 2 secondes $\leadsto$ 5 minutes + 2 secondes
  optimal : 2 heures $\leadsto$ 2 heures + 5 minutes

On se concentre sur un seul des problèmes soulevés :

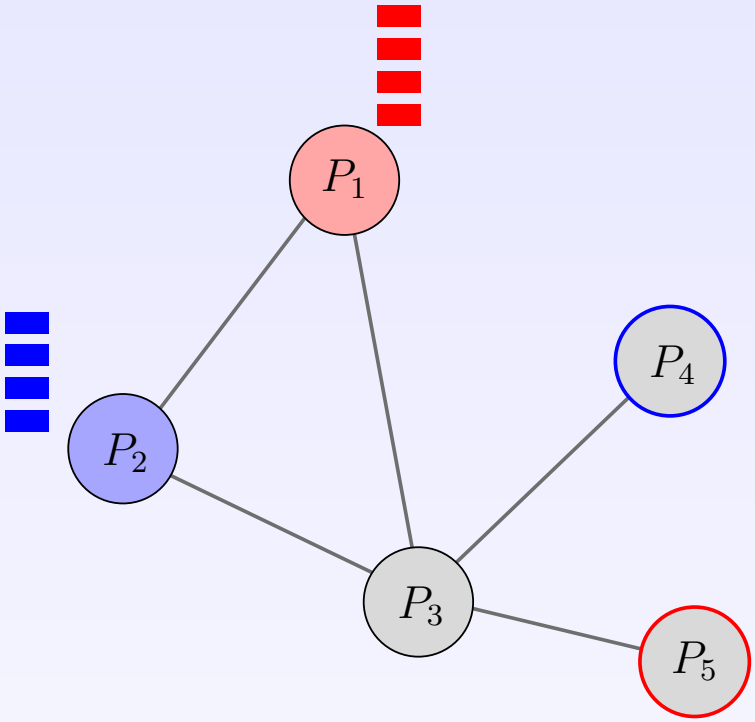
concevoir des ordonnancements efficaces
pour les plates-formes hétérogènes à grande échelle.

On s'intéresse à des plates-formes statiques. La dynamicité des plates-formes émergentes reste à prendre en compte...

Introduction au régime permanent

L'exemple du routage de paquets

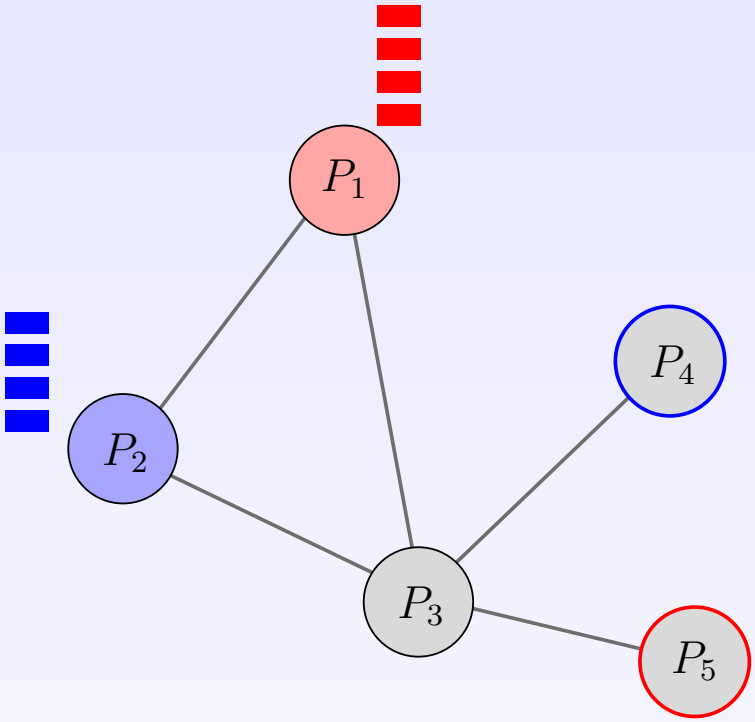
Routage de paquets – Problème



D. Bertsimas & D. Gamarnik,
« *Asymptotically optimal algorithm for job shop scheduling and packet routing* »
Journal of Algorithms, 1999.

- Plusieurs ensembles de paquets à acheminer depuis leur source jusqu'à leur destination
- Routage non fixé
- Les paquets d'un même ensemble peuvent emprunter des chemins différents
- Un paquet parcourt une arête en temps 1
- Une arête transporte un seul paquet à chaque instant

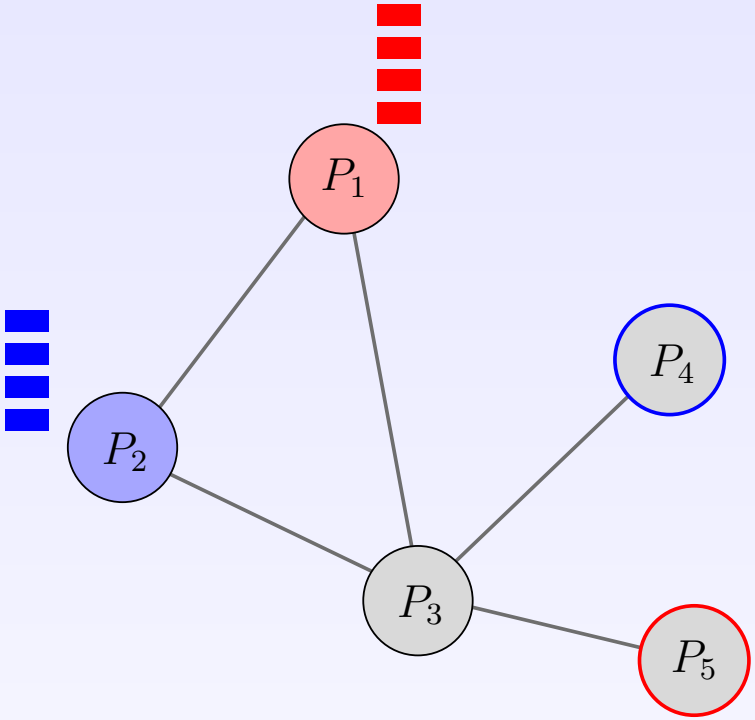
Routage de paquets – Problème

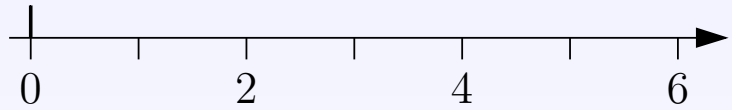


D. Bertsimas & D. Gamarnik,
« *Asymptotically optimal algorithm for job shop scheduling and packet routing* »
Journal of Algorithms, 1999.

- Plusieurs ensembles de paquets à acheminer depuis leur source jusqu'à leur destination
- Routage non fixé
- Les paquets d'un même ensemble peuvent emprunter des chemins différents
- Un paquet parcourt une arête en temps 1
- Une arête transporte un seul paquet à chaque instant

Routage de paquets



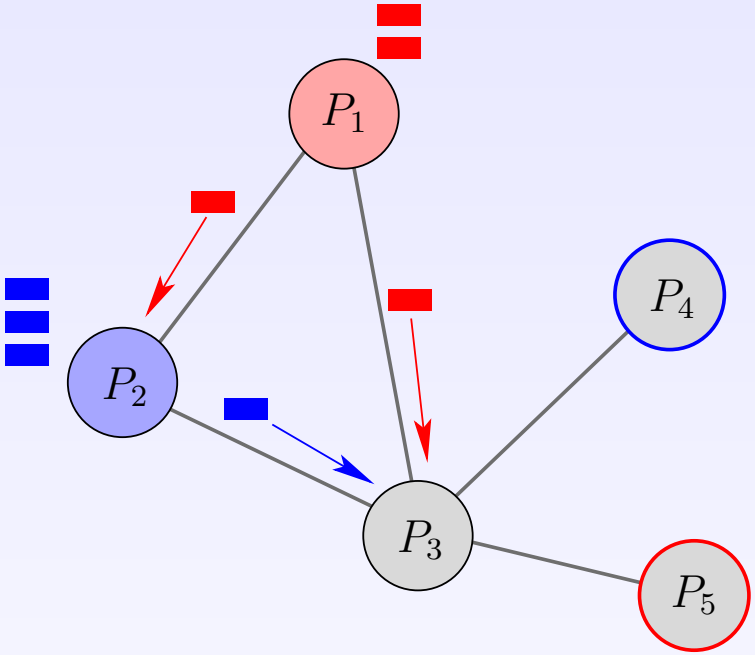


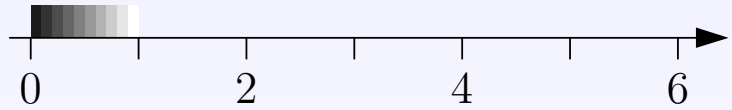
- $n_{i,j}^{k,l}$: nombre total de paquets acheminés de k à l, transmis par l'arête (i, j).
- Congestion :

$$C_{i,j} = \sum_{(k,l)|n^{k,l}>0} n_{i,j}^{k,l} \qquad C_{\max} = \max_{i,j} C_{i,j}$$

- $C_{\max}$: borne inférieure sur le temps total d'exécution

Routage de paquets



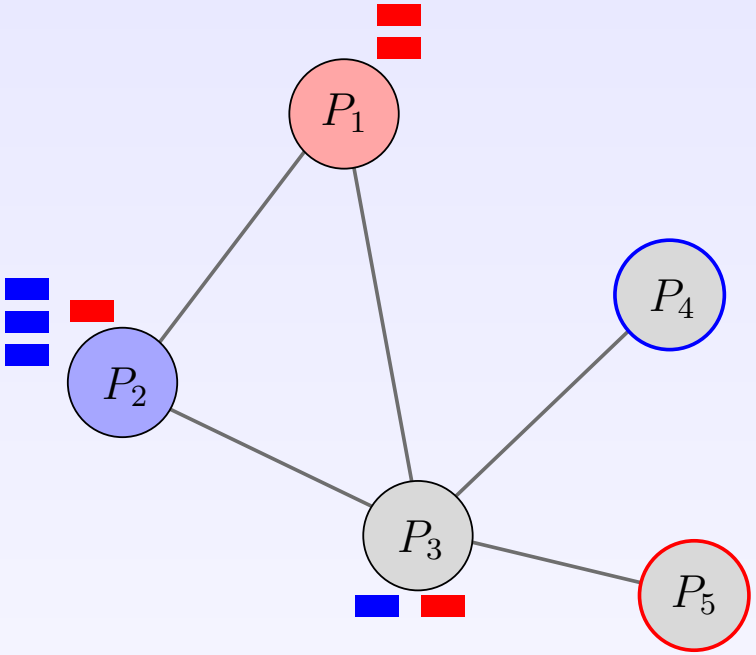


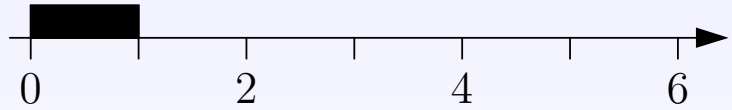
- $n_{i,j}^{k,l}$: nombre total de paquets acheminés de k à l, transmis par l'arête (i, j).
- Congestion :

$$C_{i,j} = \sum_{(k,l)|n^{k,l}>0} n_{i,j}^{k,l} \qquad C_{\max} = \max_{i,j} C_{i,j}$$

- $C_{\max}$: borne inférieure sur le temps total d'exécution

Routage de paquets



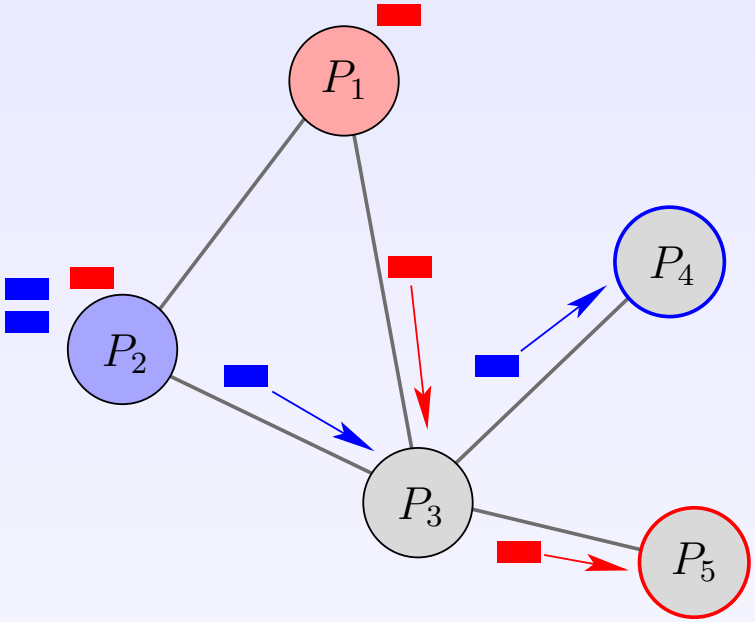


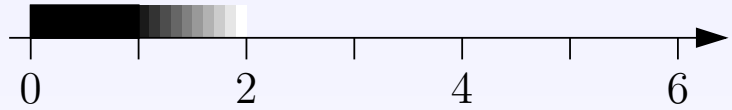
- $n_{i,j}^{k,l}$: nombre total de paquets acheminés de k à l, transmis par l'arête (i, j).
- Congestion :

$$C_{i,j} = \sum_{(k,l)|n^{k,l}>0} n_{i,j}^{k,l} \qquad C_{\max} = \max_{i,j} C_{i,j}$$

- $C_{\max}$: borne inférieure sur le temps total d'exécution

Routage de paquets



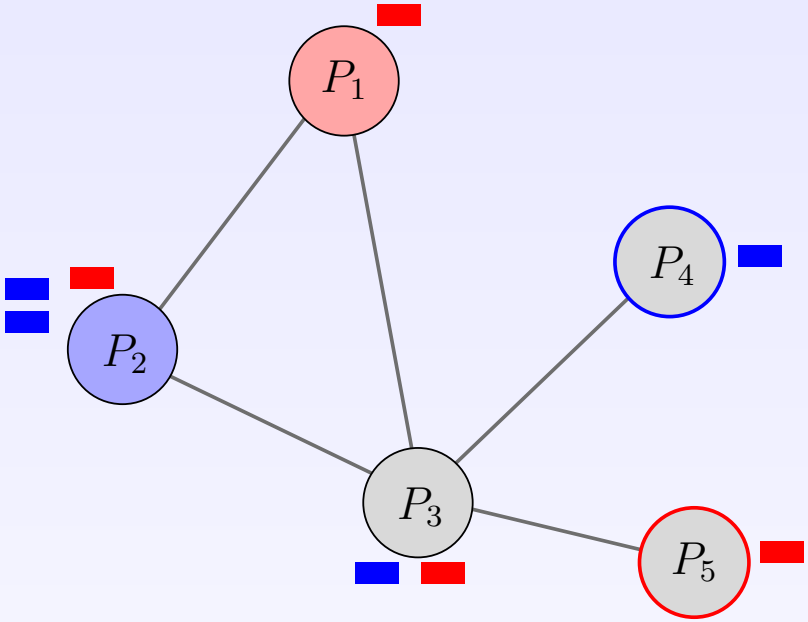


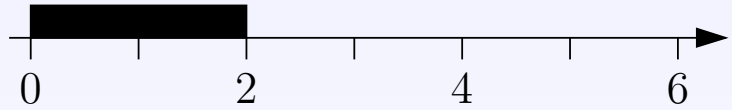
- $n_{i,j}^{k,l}$: nombre total de paquets acheminés de k à l, transmis par l'arête (i, j).
- Congestion :

$$C_{i,j} = \sum_{(k,l)|n^{k,l}>0} n_{i,j}^{k,l} \qquad C_{\max} = \max_{i,j} C_{i,j}$$

- $C_{\max}$: borne inférieure sur le temps total d'exécution

Routage de paquets



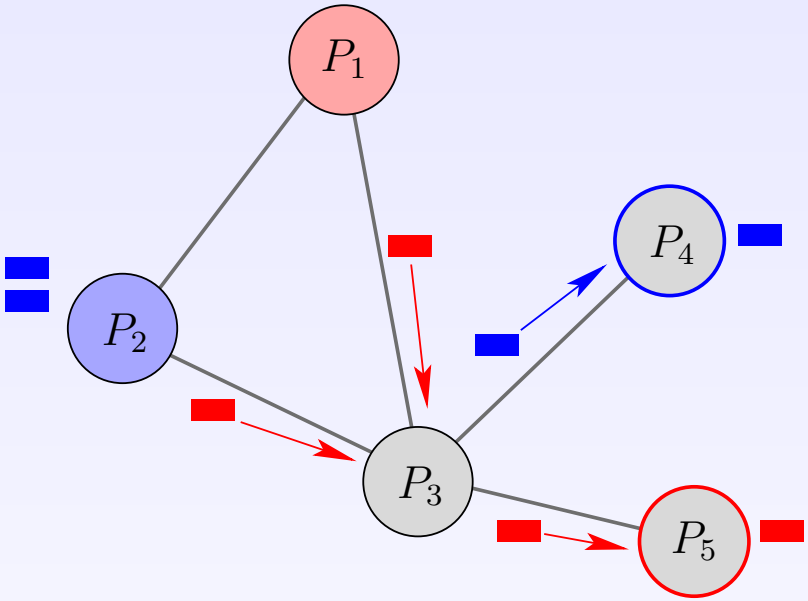


- $n_{i,j}^{k,l}$: nombre total de paquets acheminés de k à l, transmis par l'arête (i, j).
- Congestion :

$$C_{i,j} = \sum_{(k,l)|n^{k,l}>0} n_{i,j}^{k,l} \qquad C_{\max} = \max_{i,j} C_{i,j}$$

- $C_{\max}$: borne inférieure sur le temps total d'exécution

Routage de paquets

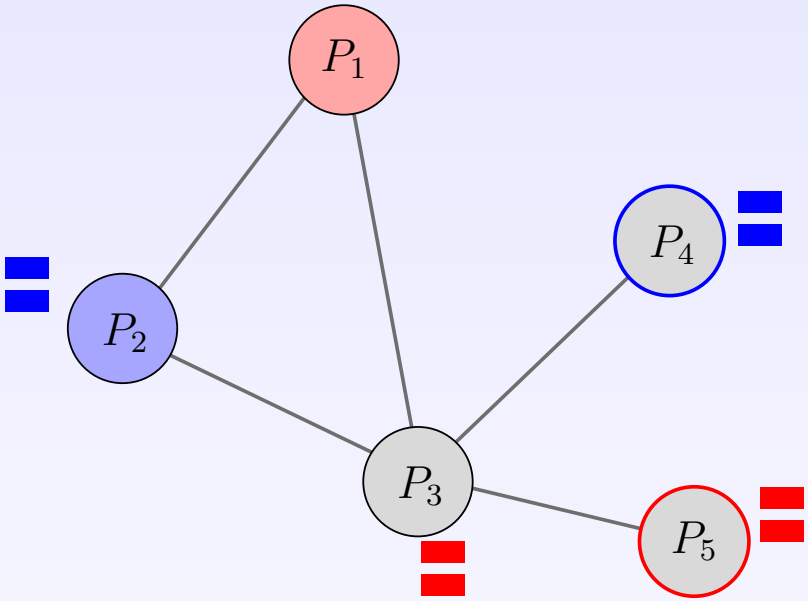


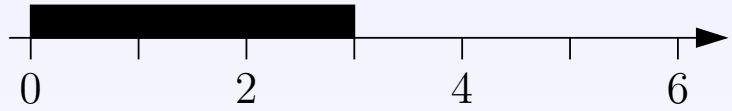
- $n_{i,j}^{k,l}$: nombre total de paquets acheminés de k à l, transmis par l'arête (i, j).
- Congestion :

$$C_{i,j} = \sum_{(k,l)|n^{k,l}>0} n_{i,j}^{k,l} \qquad C_{\max} = \max_{i,j} C_{i,j}$$

- $C_{\max}$: borne inférieure sur le temps total d'exécution

Routage de paquets



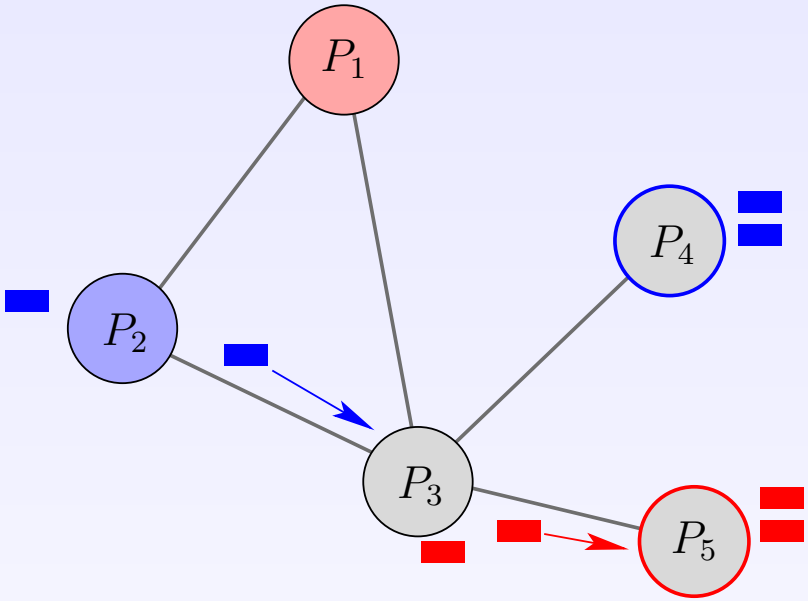


- $n_{i,j}^{k,l}$: nombre total de paquets acheminés de k à l, transmis par l'arête (i, j).
- Congestion :

$$C_{i,j} = \sum_{(k,l)|n^{k,l}>0} n_{i,j}^{k,l} \qquad C_{\max} = \max_{i,j} C_{i,j}$$

- $C_{\max}$: borne inférieure sur le temps total d'exécution

Routage de paquets

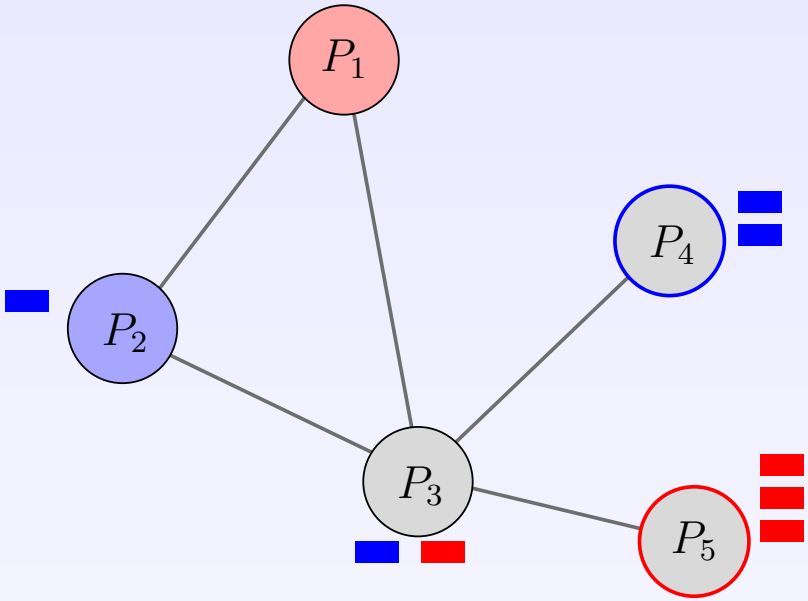


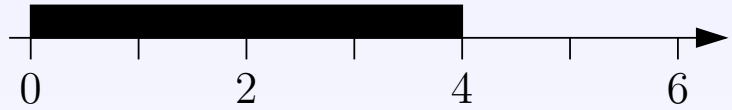
- $n_{i,j}^{k,l}$: nombre total de paquets acheminés de k à l, transmis par l'arête (i, j).
- Congestion :

$$C_{i,j} = \sum_{(k,l)|n^{k,l}>0} n_{i,j}^{k,l} \qquad C_{\max} = \max_{i,j} C_{i,j}$$

- $C_{\max}$: borne inférieure sur le temps total d'exécution

Routage de paquets



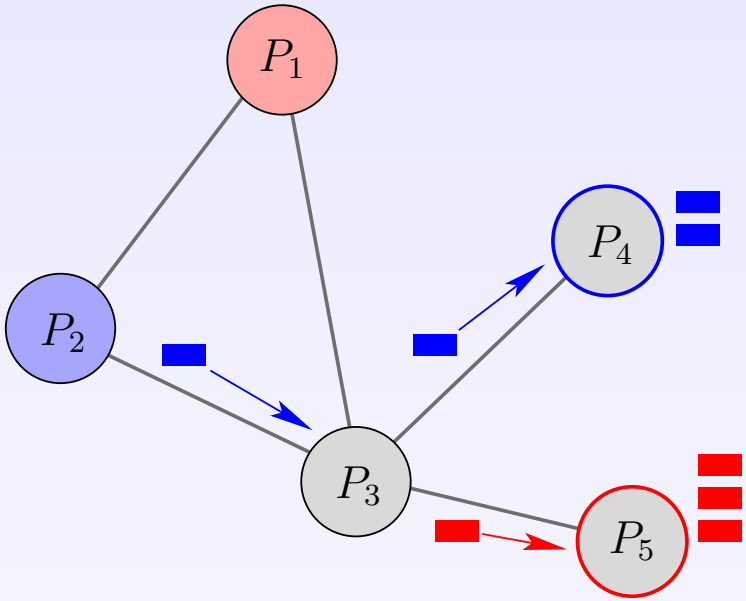


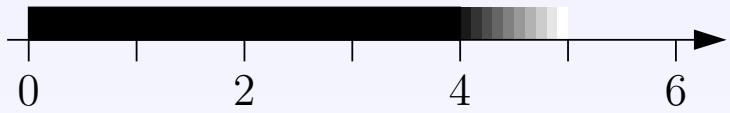
- $n_{i,j}^{k,l}$: nombre total de paquets acheminés de k à l, transmis par l'arête (i, j).
- Congestion :

$$C_{i,j} = \sum_{(k,l)|n^{k,l}>0} n_{i,j}^{k,l} \qquad C_{\max} = \max_{i,j} C_{i,j}$$

- $C_{\max}$: borne inférieure sur le temps total d'exécution

Routage de paquets



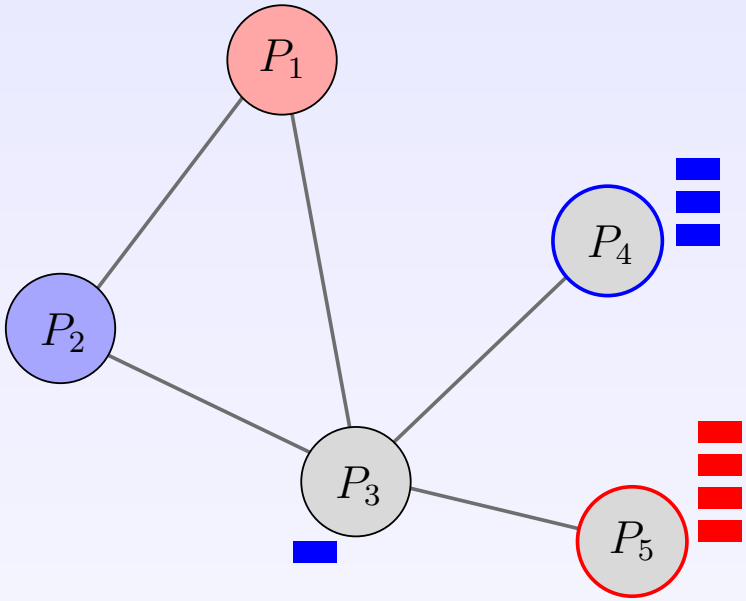


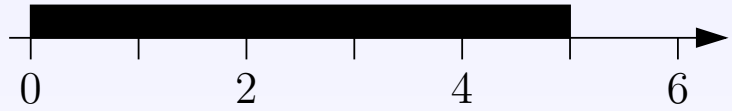
- $n_{i,j}^{k,l}$: nombre total de paquets acheminés de k à l, transmis par l'arête (i, j).
- Congestion :

$$C_{i,j} = \sum_{(k,l)|n^{k,l}>0} n_{i,j}^{k,l} \qquad C_{\max} = \max_{i,j} C_{i,j}$$

- $C_{\max}$: borne inférieure sur le temps total d'exécution

Routage de paquets



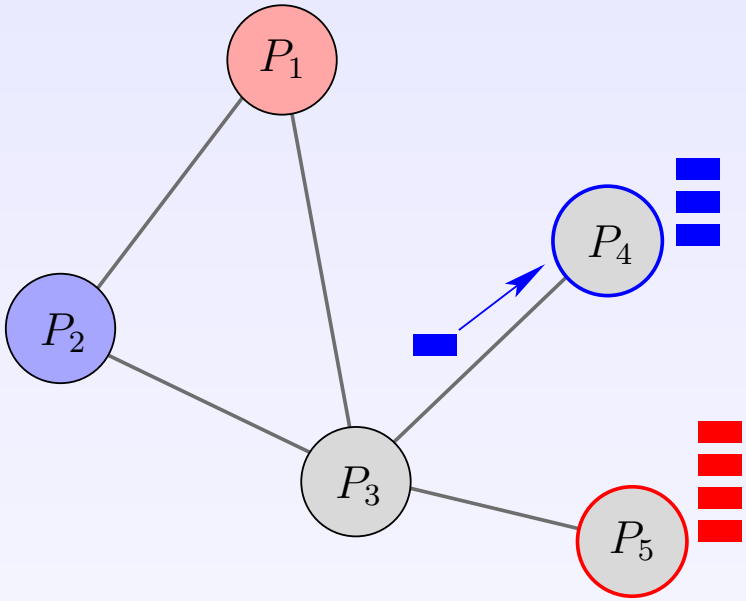


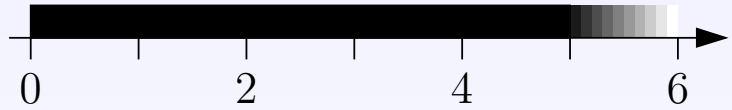
- $n_{i,j}^{k,l}$: nombre total de paquets acheminés de k à l, transmis par l'arête (i, j).
- Congestion :

$$C_{i,j} = \sum_{(k,l)|n^{k,l}>0} n_{i,j}^{k,l} \qquad C_{\max} = \max_{i,j} C_{i,j}$$

- $C_{\max}$: borne inférieure sur le temps total d'exécution

Routage de paquets



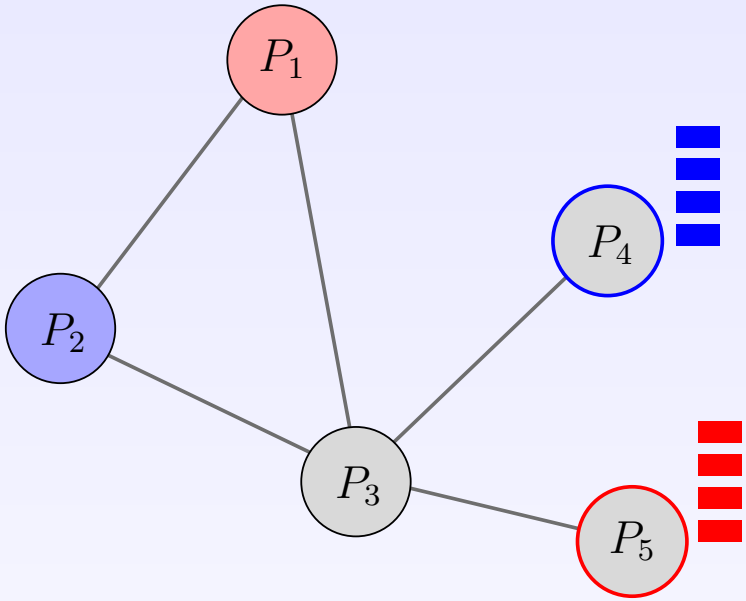


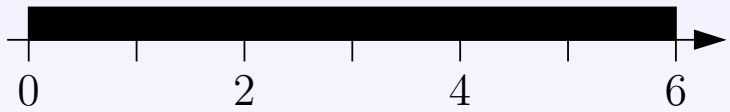
- $n_{i,j}^{k,l}$: nombre total de paquets acheminés de k à l, transmis par l'arête (i, j).
- Congestion :

$$C_{i,j} = \sum_{(k,l)|n^{k,l}>0} n_{i,j}^{k,l} \qquad C_{\max} = \max_{i,j} C_{i,j}$$

- $C_{\max}$: borne inférieure sur le temps total d'exécution

Routage de paquets



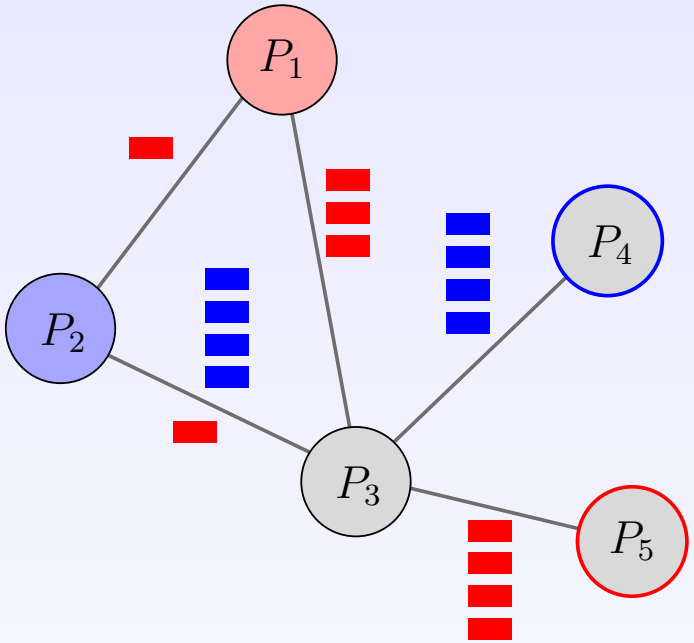


- $n_{i,j}^{k,l}$: nombre total de paquets acheminés de k à l, transmis par l'arête (i, j).
- Congestion :

$$C_{i,j} = \sum_{(k,l)|n^{k,l}>0} n_{i,j}^{k,l} \qquad C_{\max} = \max_{i,j} C_{i,j}$$

- $C_{\max}$: borne inférieure sur le temps total d'exécution

Routage de paquets

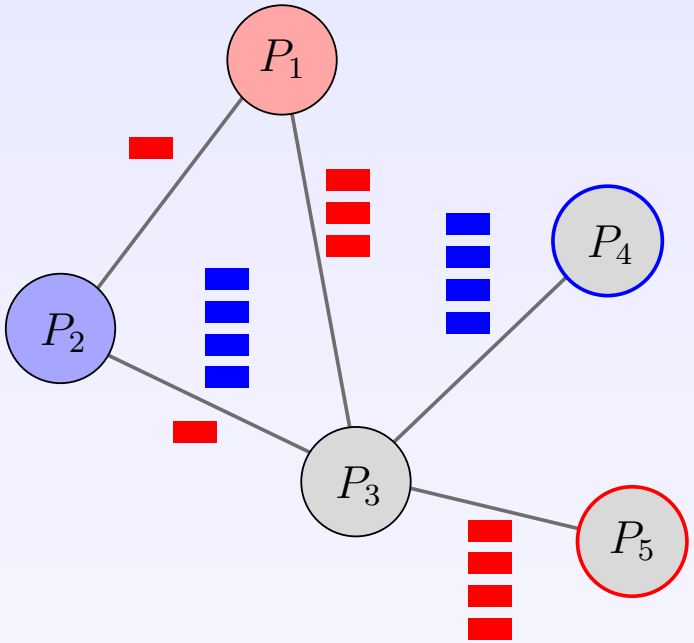


- $n_{i,j}^{k,l}$: nombre total de paquets acheminés de k à l, transmis par l'arête (i, j).
- Congestion :

$$C_{i,j} = \sum_{(k,l)|n^{k,l}>0} n_{i,j}^{k,l} \qquad C_{\max} = \max_{i,j} C_{i,j}$$

- $C_{\max}$: borne inférieure sur le temps total d'exécution

Routage de paquets

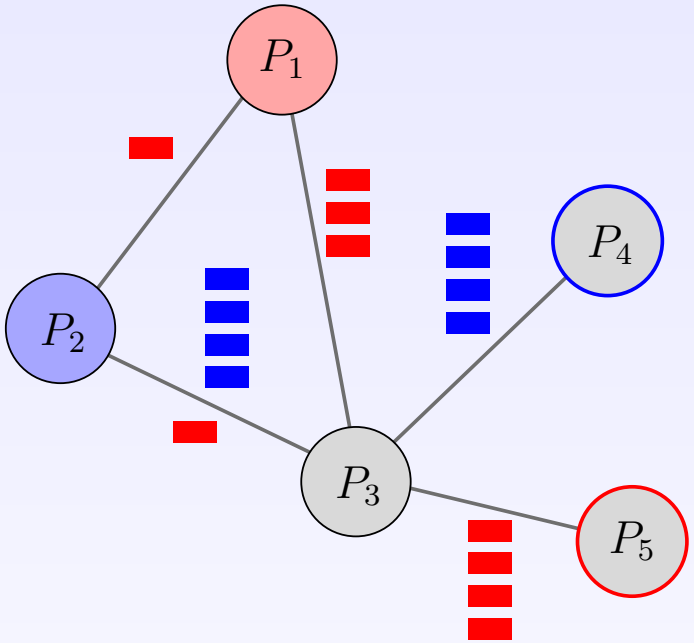


- $n_{i,j}^{k,l}$: nombre total de paquets acheminés de k à l, transmis par l'arête (i, j).
- Congestion :

$$C_{i,j} = \sum_{(k,l)|n^{k,l}>0} n_{i,j}^{k,l} \qquad C_{\max} = \max_{i,j} C_{i,j}$$

- $C_{\max}$: borne inférieure sur le temps total d'exécution

Routage de paquets

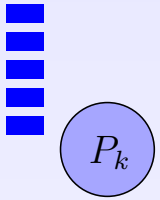


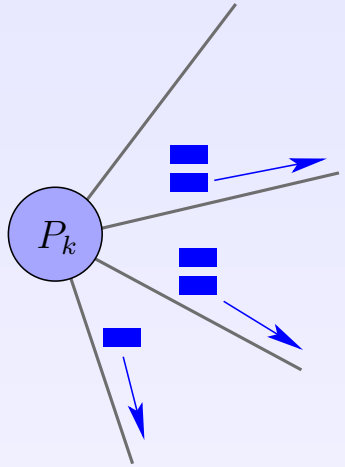
- $n_{i,j}^{k,l}$: nombre total de paquets acheminés de k à l, transmis par l'arête (i, j).
- Congestion :

$$C_{i,j} = \sum_{(k,l)|n^{k,l}>0} n_{i,j}^{k,l} \qquad C_{\max} = \max_{i,j} C_{i,j}$$

- $C_{\max}$: borne inférieure sur le temps total d'exécution

Routage de paquets – Équations 1/2

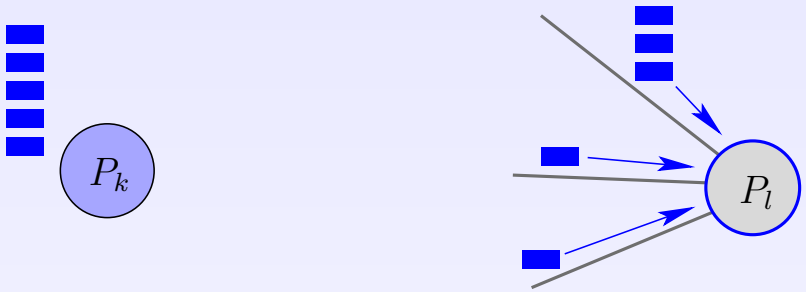




1. Paquets émis par une source : $\sum_{j|(k,j)\in A} n_{k,j}^{k,l} = n^{k,l}$
2. Paquets reçus par une destination : $\sum_{i|(i,l)\in A} n_{i,l}^{k,l} = n^{k,l}$
3. Loi de conservation (nœud relais i) :

$$\sum_{i|(i,j)\in A} n_{i,j}^{k,l} = \sum_{i|(j,i)\in A} n_{j,i}^{k,l} \quad \forall (k,l), j \neq k, j \neq l$$

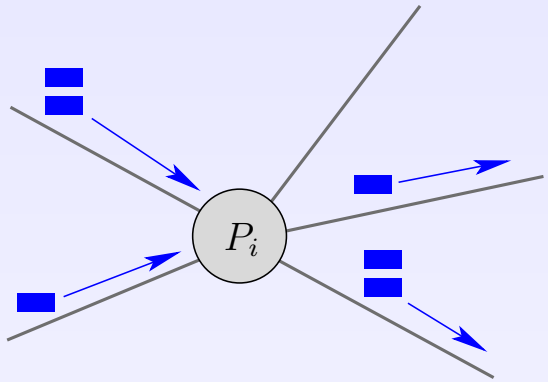
Routage de paquets – Équations 1/2



1. Paquets émis par une source : $\sum_{j|(k,j)\in A} n_{k,j}^{k,l} = n^{k,l}$
2. Paquets reçus par une destination : $\sum_{i|(i,l)\in A} n_{i,l}^{k,l} = n^{k,l}$
3. Loi de conservation (nœud relais i) :

$$\sum_{i|(i,j)\in A} n_{i,j}^{k,l} = \sum_{i|(j,i)\in A} n_{j,i}^{k,l} \quad \forall (k,l), j \neq k, j \neq l$$

Routage de paquets – Équations 1/2



1. Paquets émis par une source : $\sum_{j|(k,j)\in A} n_{k,j}^{k,l} = n^{k,l}$
2. Paquets reçus par une destination : $\sum_{i|(i,l)\in A} n_{i,l}^{k,l} = n^{k,l}$
3. Loi de conservation (nœud relais i) :

$$\sum_{i|(i,j)\in A} n_{i,j}^{k,l} = \sum_{i|(j,i)\in A} n_{j,i}^{k,l} \quad \forall (k,l), j \neq k, j \neq l$$

Routage de paquets – Équations 2/2

4. Congestion

$$C_{i,j} = \sum_{(k,l)|n^{k,l}>0} n_{i,j}^{k,l}$$

5. Fonction objective :

$$C_{\max} \geq C_{i,j}, \qquad \forall i, j$$

Minimiser $C_{\max}$

Programme linéaire en rationnels : résolu en temps polynomial

Algorithme de routage

1. Calculer une solution optimale $C_{\max}$, $n_{i,j}^{k,l}$ du programme linéaire
2. Ordonnancement périodique :
   - On pose $\Omega = \sqrt{C_{\max}}$
   - On utilise $\left\lceil \frac{C_{\max}}{\Omega} \right\rceil$ périodes de durée Ω
   - Pendant chaque période, l'arête (i, j) transmet (au plus)

$$m_{i,j}^{k,l} = \left\lfloor \frac{n_{i,j}^{k,l}\Omega}{C_{\max}} \right\rfloor \qquad \text{paquets de } k \text{ à } l$$

3. Terminaison : on traite séquentiellement les paquets résiduels

Ordonnancement valide, asymptotiquement optimal :

$$C_{\max} \leq C^* \leq C_{\max} + O(\sqrt{C_{\max}})$$

Étude du régime permanent

Principe :

- Relaxation de la fonction objective (temps d'exécution $\rightsquigarrow$ débit)
- Nombres rationnels de paquets dans le programme linéaire
- Construction d'un ordonnancement périodique
- Grand nombre de périodes $\Rightarrow$ obtention d'un régime permanent

Intérêt :

- Ordonnancement asymptotiquement optimal, adapté à :
  - grands volumes de données
  - applications constituées d'un grand nombre de tâches indépendantes
- Permet aussi d'obtenir une description plus compacte de l'ordonnancement :
  - période, décrite par intervalles
  - au lieu de la route empruntée par chaque paquet

Diffusion en régime permanent

Position du problème et bibliographie

Résolution efficace sous le modèle bidirectionnel

Complexité sous le modèle unidirectionnel

Heuristiques à un seul arbre

Expérimentations

Diffusion en régime permanent

Position du problème et bibliographie

Résolution efficace sous le modèle bidirectionnel

Complexité sous le modèle unidirectionnel

Heuristiques à un seul arbre

Expérimentations

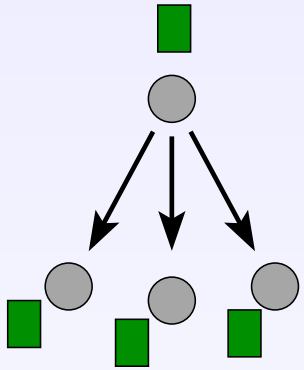
Communications collectives en régime permanent

Communication collective : opération de communication faisant intervenir plusieurs interlocuteurs

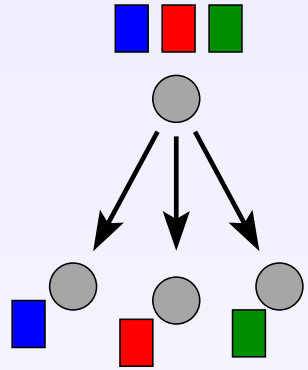
Hypothèse de régime permanent : communications pipelinées

$\rightsquigarrow$ un grand nombre de messages suivant le même schéma

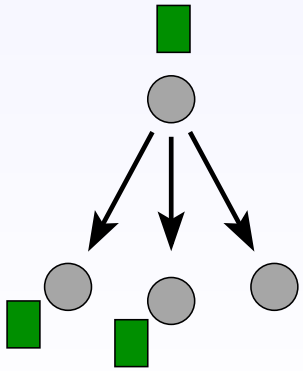
diffusion :



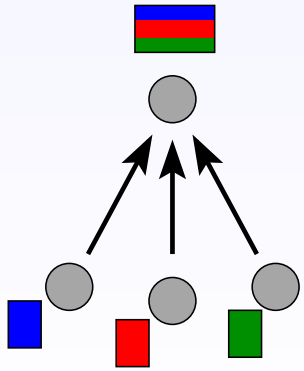
distribution :



multicast :



réduction :



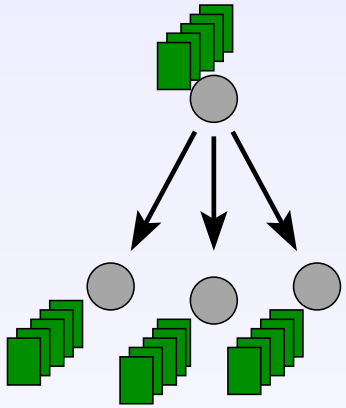
Communications collectives en régime permanent

Communication collective : opération de communication faisant intervenir plusieurs interlocuteurs

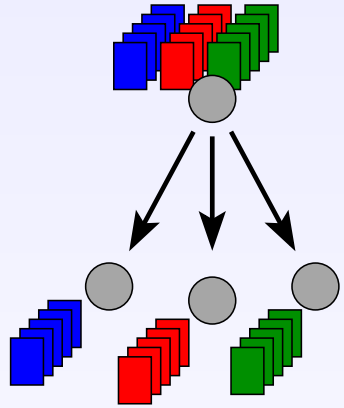
Hypothèse de régime permanent : communications pipelinées

$\rightsquigarrow$ un grand nombre de messages suivant le même schéma

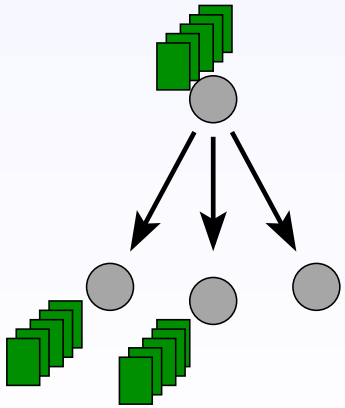
diffusion :



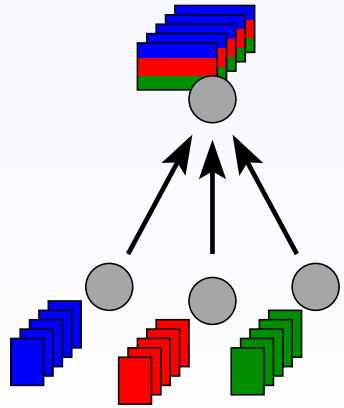
distribution :



multicast :



réduction :



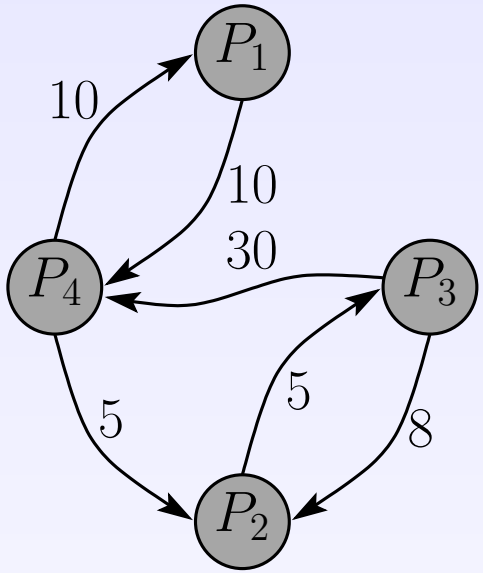
Bibliographie

- Diffusion sur des architectures spécialisées homogènes (utilisation d'arbres de diffusion concurrents)
  - grille, tore
  - hypercube
  - graphe de De Bruijn
  - réseaux pair-à-pair
- Topologies hétérogènes :
  - optimisation d'une diffusion
  - bibliothèques de communication pour environnements hétérogènes (ECO, MagPIe, MPICH-G2,...)

NB : en optimisant le débit d'une série de diffusions, on résout de façon asymptotique le problème de la diffusion d'un message de grande taille

Modèle de communications

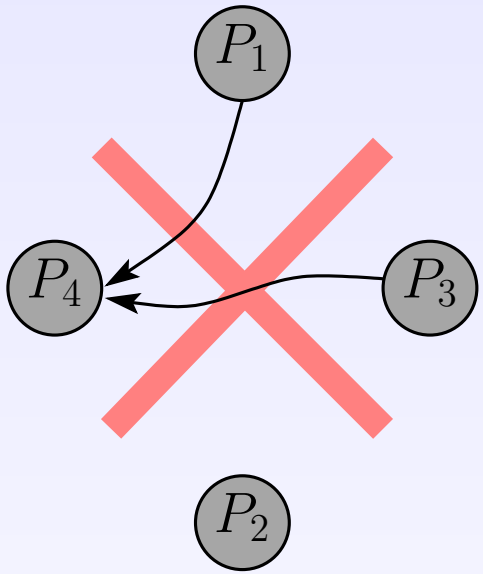
- plate-forme : graphe $G = (V, E, c)$
- $P_1, P_2, \ldots, P_n$: machines (processeurs)
- P_{source} (possède les données à diffuser)
- $(P_j, P_k) \in E$: lien de communication enter
- pondération sur les arêtes :
  $c(j, k)$= temps d'un transfert d'un message de taille 1 de P_j à P_j



- congestion au niveau d'une machine :
  modèle de communication un-port, avec 2 variantes :
  - bidirectionnel :
    un processeur peut émettre et reçoit un message à la fois
  - unidirectionnel :
    un processeur peut émettre ou reçoit un message à la fois

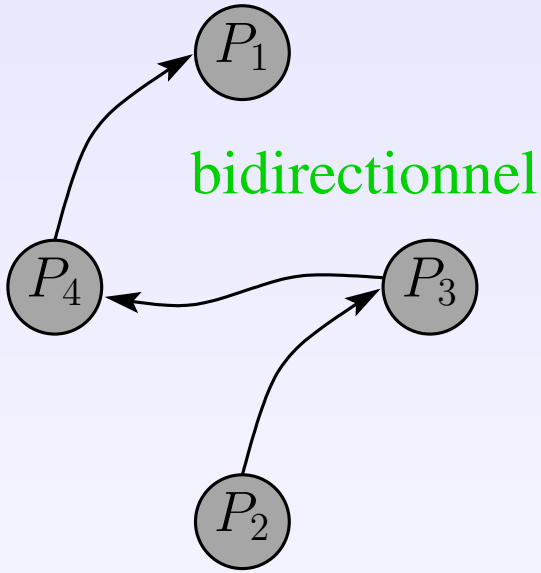
Modèle de communications

- plate-forme : graphe $G = (V, E, c)$
- $P_1, P_2, \ldots, P_n$: machines (processeurs)
- P_{source} (possède les données à diffuser)
- $(P_j, P_k) \in E$: lien de communication enter
- pondération sur les arêtes :
  $c(j,k)$= temps d'un transfert d'un message de taille 1 de P_j à P_j
- congestion au niveau d'une machine :
  modèle de communication un-port, avec 2 variantes :
  - bidirectionnel :
    un processeur peut émettre et reçoit un message à la fois
  - unidirectionnel :
    un processeur peut émettre ou reçoit un message à la fois



Modèle de communications

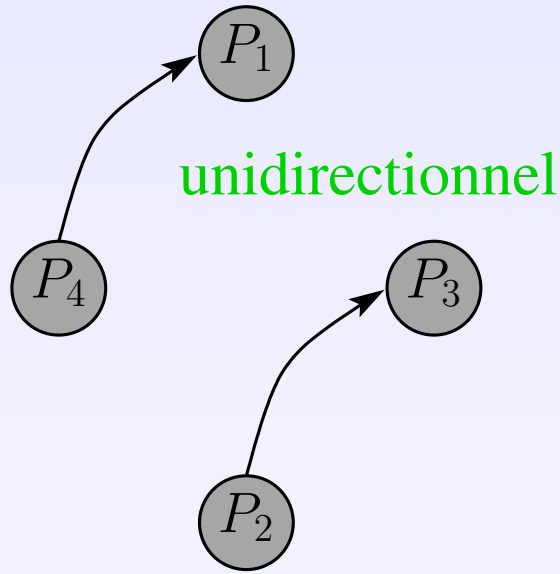
- plate-forme : graphe $G = (V, E, c)$
- $P_1, P_2, \ldots, P_n$: machines (processeurs)
- P_{source} (possède les données à diffuser)
- $(P_j, P_k) \in E$: lien de communication enter
- pondération sur les arêtes :
  $c(j,k)$= temps d'un transfert d'un message de taille 1 de P_j à P_j



- congestion au niveau d'une machine :
  modèle de communication un-port, avec 2 variantes :
  - bidirectionnel :
    un processeur peut émettre et reçoit un message à la fois
  - unidirectionnel :
    un processeur peut émettre ou reçoit un message à la fois

Modèle de communications

- plate-forme : graphe $G = (V, E, c)$
- $P_1, P_2, \ldots, P_n$: machines (processeurs)
- P_{source} (possède les données à diffuser)
- $(P_j, P_k) \in E$: lien de communication enter
- pondération sur les arêtes :
  $c(j,k)$= temps d'un transfert d'un message de taille 1 de P_j à P_j
- congestion au niveau d'une machine :
  modèle de communication un-port, avec 2 variantes :
  - bidirectionnel :
    un processeur peut émettre et reçoit un message à la fois
  - unidirectionnel :
    un processeur peut émettre ou reçoit un message à la fois



Diffusion en régime permanent

Position du problème et bibliographie

Résolution efficace sous le modèle bidirectionnel

Complexité sous le modèle unidirectionnel

Heuristiques à un seul arbre

Expérimentations

Approche par programmation linéaire

- Adaptation de la méthode de Bertsimas et Gamarnik
- Quantités moyennes de messages transmis par unité de temps
- Débit = quantité moyenne reçue par chaque destination, par unité de temps
- Une source / plusieurs destinations
- On distingue tout de même la destination des messages :

send$(P_i \to P_j, k)$: Nombre moyen de messages envoyés à destination de k, traversant l'arête (P_i, P_j) en une unité de temps.

Approche par programmation linéaire

- Adaptation de la méthode de Bertsimas et Gamarnik
- Quantités moyennes de messages transmis par unité de temps
- Débit = quantité moyenne reçue par chaque destination, par unité de temps
- Une source / plusieurs destinations
- On distingue tout de même la destination des messages :

send$(P_i \to P_j, k)$: Nombre moyen de messages envoyés à destination de k, traversant l'arête (P_i, P_j) en une unité de temps.

Contraintes

On peut écrire les mêmes contraintes que pour le routage de paquets :

- tous les messages sont émis par la source

$$\forall \text{ cible } P_k \qquad \sum_{(P_{\text{source}},P_j)\in E} \text{send}(P_{\text{source}} \to P_j, k) = \rho$$

- tous les messages sont reçus par une destination

$$\forall \text{ cible } P_k \qquad \sum_{(P_j,P_k)\in E} \text{send}(P_j \to k, k) = \rho$$

- conservation du nombre de messages

$$\forall \text{ cible } P_k, \forall P_i \neq P_k, P_{\text{source}} \qquad \sum_{(P_j,P_i)\in E} \text{send}(P_j \to P_i, k) = \sum_{(P_i,P_j)\in E} \text{send}(P_i \to P_j, k)$$

Contraintes

On peut écrire les mêmes contraintes que pour le routage de paquets :

- tous les messages sont émis par la source

$$\forall \text{ cible } P_k \qquad \sum_{(P_{\text{source}},P_j)\in E} \text{send}(P_{\text{source}} \to P_j, k) = \rho$$

- tous les messages sont reçus par une destination

$$\forall \text{ cible } P_k \qquad \sum_{(P_j,P_k)\in E} \text{send}(P_j \to k, k) = \rho$$

- conservation du nombre de messages

$$\forall \text{ cible } P_k, \forall P_i \neq P_k, P_{\text{source}} \qquad \sum_{(P_j,P_i)\in E} \text{send}(P_j \to P_i, k) = \sum_{(P_i,P_j)\in E} \text{send}(P_i \to P_j, k)$$

Contraintes

On peut écrire les mêmes contraintes que pour le routage de paquets :

- tous les messages sont émis par la source

$$\forall \text{ cible } P_k \qquad \sum_{(P_{\text{source}},P_j)\in E} \text{send}(P_{\text{source}} \to P_j, k) = \rho$$

- tous les messages sont reçus par une destination

$$\forall \text{ cible } P_k \qquad \sum_{(P_j,P_k)\in E} \text{send}(P_j \to k, k) = \rho$$

- conservation du nombre de messages

$$\forall \text{ cible } P_k, \forall P_i \neq P_k, P_{\text{source}}$$
$$\sum_{(P_j,P_i)\in E} \text{send}(P_j \to P_i, k) = \sum_{(P_i,P_j)\in E} \text{send}(P_i \to P_j, k)$$

Nombre de message sur une arête

- La distinction des messages par destination est “fictive”
- Quel est le nombre total de messages sur l'arête (i, j) ?

$$s(P_i \to P_j) = \sum_k \text{send}(P_i \to P_j, k) \qquad ?$$

- Certains messages à destination de k et k' sont en fait deux copies du même message : on ne veut pas les transmettre deux fois
- Hypothèse optimiste :

$$s(P_i \to P_j) = \max_k \{\text{send}(P_i \to P_j, k)\}$$

$\Leftrightarrow$ il existe une destination k_0 (réalisant le $\max_k$) telle que les messages envoyés vers toute autre destination k forment un sous-ensemble des messages envoyés à k_0

- Pas de justification de cette hypothèse ici : on cherche d'abord une majoration du débit, pas forcément réalisable

Nombre de message sur une arête

- La distinction des messages par destination est “fictive”
- Quel est le nombre total de messages sur l'arête (i, j) ?

$$s(P_i \to P_j) = \sum_k \text{send}(P_i \to P_j, k) \qquad \text{non}$$

- Certains messages à destination de k et k' sont en fait deux copies du même message : on ne veut pas les transmettre deux fois
- Hypothèse optimiste :

$$s(P_i \to P_j) = \max_k \{\text{send}(P_i \to P_j, k)\}$$

$\Leftrightarrow$ il existe une destination k_0 (réalisant le $\max_k$) telle que les messages envoyés vers toute autre destination k forment un sous-ensemble des messages envoyés à k_0

- Pas de justification de cette hypothèse ici : on cherche d'abord une majoration du débit, pas forcément réalisable

Nombre de message sur une arête

- La distinction des messages par destination est “fictive”
- Quel est le nombre total de messages sur l'arête (i, j) ?

$$s(P_i \to P_j) = \sum_k \text{send}(P_i \to P_j, k) \qquad \text{non}$$

- Certains messages à destination de k et k' sont en fait deux copies du même message : on ne veut pas les transmettre deux fois
- Hypothèse optimiste :

$$s(P_i \to P_j) = \max_k \{\text{send}(P_i \to P_j, k)\}$$

$\Leftrightarrow$ il existe une destination k_0 (réalisant le $\max_k$) telle que les messages envoyés vers toute autre destination k forment un sous-ensemble des messages envoyés à k_0

- Pas de justification de cette hypothèse ici : on cherche d'abord une majoration du débit, pas forcément réalisable

Nombre de message sur une arête

- La distinction des messages par destination est “fictive”
- Quel est le nombre total de messages sur l'arête (i, j) ?

$$s(P_i \to P_j) = \sum_k \text{send}(P_i \to P_j, k) \qquad \text{non}$$

- Certains messages à destination de k et k' sont en fait deux copies du même message : on ne veut pas les transmettre deux fois
- Hypothèse optimiste :

$$s(P_i \to P_j) = \max_k \{\text{send}(P_i \to P_j, k)\}$$

$\Leftrightarrow$ il existe une destination k_0 (réalisant le $\max_k$) telle que les messages envoyés vers toute autre destination k forment un sous-ensemble des messages envoyés à k_0

- Pas de justification de cette hypothèse ici : on cherche d'abord une majoration du débit, pas forcément réalisable

Contrainte du modèle un-port

- Temps moyen d'utilisation de l'arête (P_i, P_j) :

$$T(i,j) = s(i \rightarrow j) \times c_{i,j}$$

- Un processeur ne peut émettre deux message à la fois
  $\rightsquigarrow$ temps moyen d'émission par unité de temps borné :

$$\forall P_i \qquad \sum_{(P_i,P_j)\in E} T(i,j) \leq 1$$

- Un processeur ne peut recevoir deux message à la fois
  $\rightsquigarrow$ temps moyen de réception par unité de temps borné :

$$\forall P_i \qquad \sum_{(P_k,P_i)\in E} T(k,i) \leq 1$$

Contrainte du modèle un-port

- Temps moyen d'utilisation de l'arête (P_i, P_j) :

$$T(i,j) = s(i \to j) \times c_{i,j}$$

- Un processeur ne peut émettre deux message à la fois
  $\rightsquigarrow$ temps moyen d'émission par unité de temps borné :

$$\forall P_i \qquad \sum_{(P_i,P_j)\in E} T(i,j) \leq 1$$

- Un processeur ne peut recevoir deux message à la fois
  $\rightsquigarrow$ temps moyen de réception par unité de temps borné :

$$\forall P_i \qquad \sum_{(P_k,P_i)\in E} T(k,i) \leq 1$$

Contrainte du modèle un-port

- Temps moyen d'utilisation de l'arête (P_i, P_j) :

$$T(i,j) = s(i \rightarrow j) \times c_{i,j}$$

- Un processeur ne peut émettre deux message à la fois
  $\rightsquigarrow$ temps moyen d'émission par unité de temps borné :

$$\forall P_i \qquad \sum_{(P_i,P_j)\in E} T(i,j) \leq 1$$

- Un processeur ne peut recevoir deux message à la fois
  $\rightsquigarrow$ temps moyen de réception par unité de temps borné :

$$\forall P_i \qquad \sum_{(P_k,P_i)\in E} T(k,i) \leq 1$$

Programme linéaire obtenu

Maximiser ρ_{opt}

$$\forall P_k \in V_{\text{cibles}} \qquad \sum_{(P_{\text{source}}, P_j) \in E} \text{send}(P_{\text{source}} \to P_j, k) = \rho_{\text{opt}}$$

$$\forall P_k \in V_{\text{cibles}} \qquad \sum_{(P_j, P_k) \in E} \text{send}(P_j \to P_k, k) = \rho_{\text{opt}}$$

$$\begin{cases} \forall P_k \in V_{\text{cibles}}, \forall P_i \\ P_i \neq P_k, P_{\text{source}} \end{cases} \qquad \sum_{(P_j, P_i) \in E} \text{send}(P_j \to P_i, k) = \sum_{(P_i, P_j) \in E} \text{send}(P_i \to P_j, k)$$

$$\forall (i,j) \in E, \qquad \sum_{P_k \in V_{\text{cibles}}} \text{send}(P_i \to P_j, k) \cdot c_{i,j} \leq T_{i,j}$$

$$\forall P_i \in V, \qquad \sum_{(i,j) \in E} T_{i,j} \leq 1$$

$$\forall P_i \in V, \qquad \sum_{(j,i) \in E} T_{j,i} \leq 1$$

$$\forall P_i, P_j, P_k \in V \qquad \text{send}(P_i \to P_j, k) \geq 0$$

$O(n^2)$ contraintes (non triviales) pour $O(n^3)$ variables

Construction d'un ensemble d'arbres de débit optimal

On cherche à atteindre la borne ρ_{opt} calculé par le programme linéaire.

$G_s = (V, E, s)$: graphe dont les arêtes sont étiquetées par la valeur de $s(P_i \rightarrow P_j)$ dans une solution optimale du programme linéaire

Propriété

Dans G_s, il existe un flot de valeur ρ_{opt} de la source vers tout nœud.

ρ_{opt} est la valeur de la coupe minimale entre P_{source} et tout autre sommet P_i

Construction d'un ensemble d'arbres de débit optimal

Version pondérée du théorème d'Edmonds sur les arborescences

Étant donné le graphe dirigé pondéré G_s, on peut trouver un ensemble d'arbres $A_1, \ldots, A_k$ et un ensemble de poids $x_1, \ldots x_k$ tels que $\sum_k x_i T_i \leq G_s$ et

$$\sum_i x_i = \rho_{\text{opt}}$$

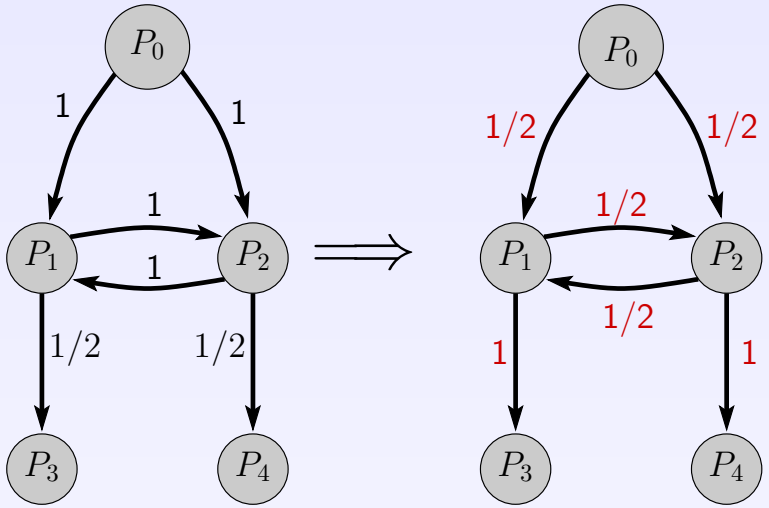
en temps fortement polynomial, et $k \leq |V|^3 + |E|$.

Ce résultat permet d'obtenir :

- un ensemble d'arbres réalisant ρ_{opt}, leur pondération
- le nombre d'arbres est borné par $|V|^3 + |E|$

Exemple d'extraction d'arbres

1. Résultat du programme linéaire :



$$\rho_{\text{opt}} = 1$$

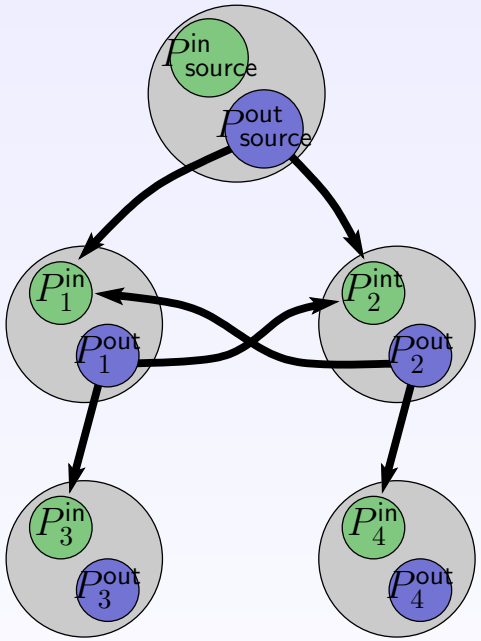
2. Découpage en arbres (théorème d'Edmonds) :

$$\Longrightarrow \frac{1}{2} \times \left(\right) + \frac{1}{2} \left(\right)$$

Organisation des communications

- Équations du modèle un-port $\rightsquigarrow$ organisation des communications
- processeur $P_i \rightarrow \begin{cases} \text{un processeur pour les émissions } P^{\text{out}} \\ \text{un processeur pour les réceptions } P^{\text{in}} \end{cases}$

G_B :



- Ensemble de communications pouvant se faire simultanément :

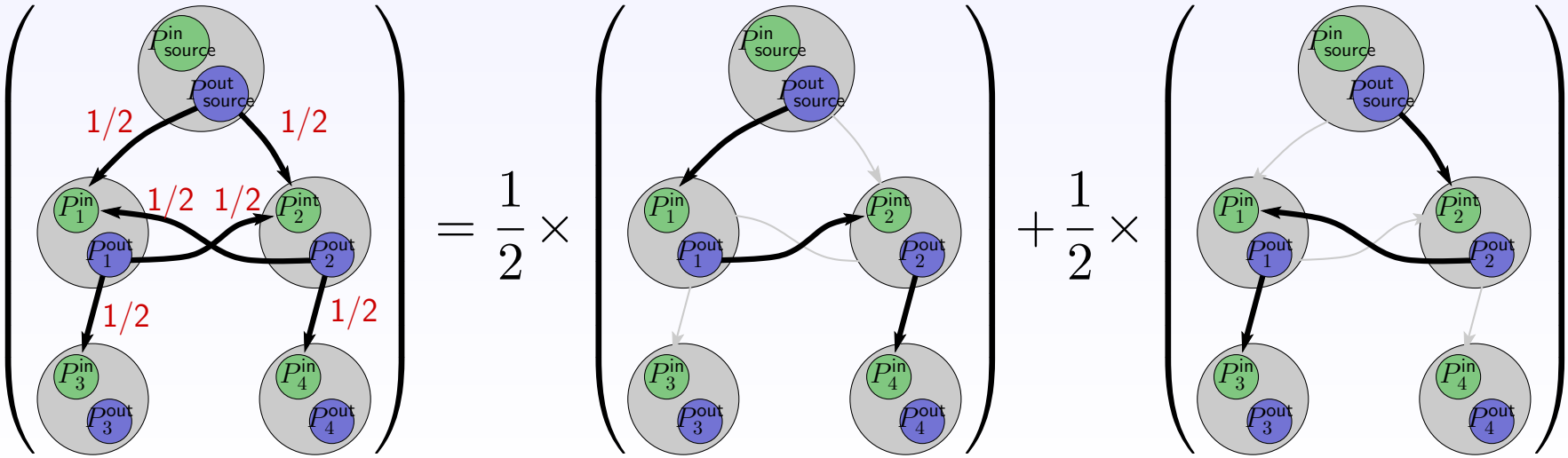
couplage dans le graphe biparti G_B

Version pondérée du théorème de König

Il est possible de décomposer G_B en une somme pondérée de couplages, telles que la somme des pondérations est au plus égale au degré maximal d'un nœud dans G_B.

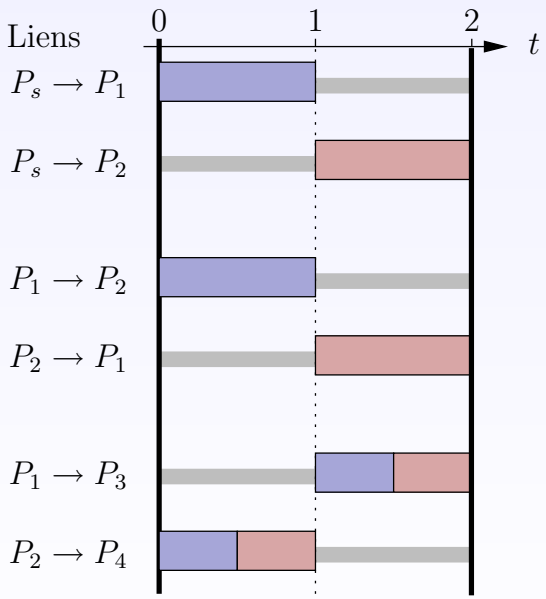
nombre de couplages borné par $|E|$

Le degré maximal d'un nœud dans le graphe biparti dont les arêtes sont étiquetées par la valeur $s(P_i \to P_j)$ vaut 1 (d'après les contraintes un-port).



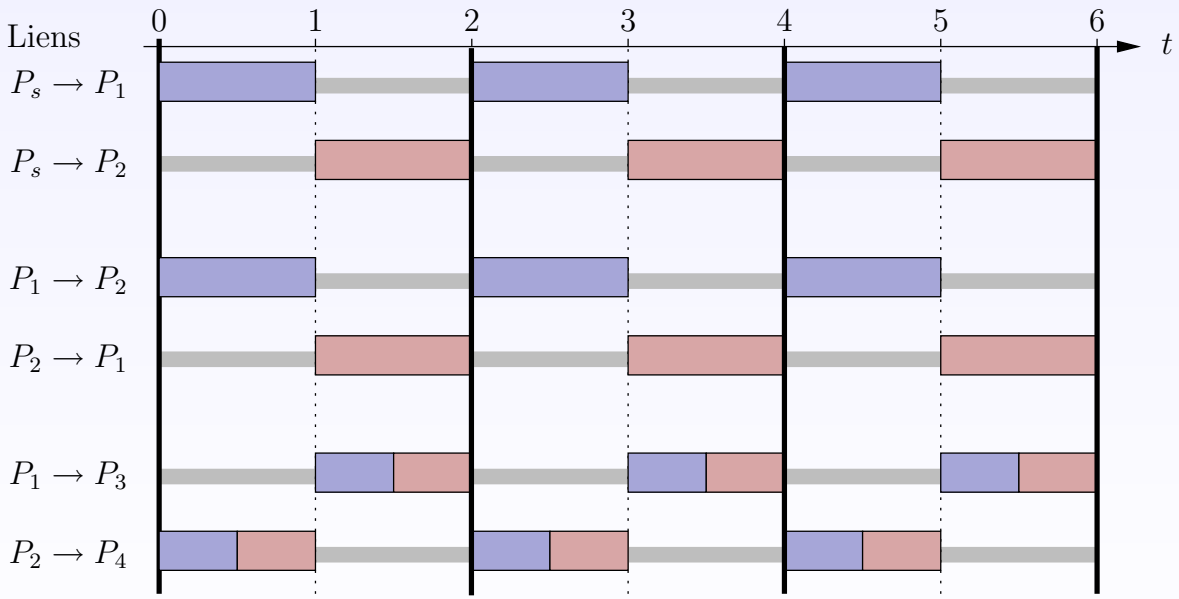
Reconstruction d'une période

- Quantités rationnelles de messages dans chaque couplage
- Durée de la période = PPCM des dénominateurs des quantités de messages dans chaque couplage
- Période découpée en intervalles représentant les couplages
- Affectation des transferts en respectant les dépendances (un processeur retransmet lors d'une période les messages reçus à la période précédente)



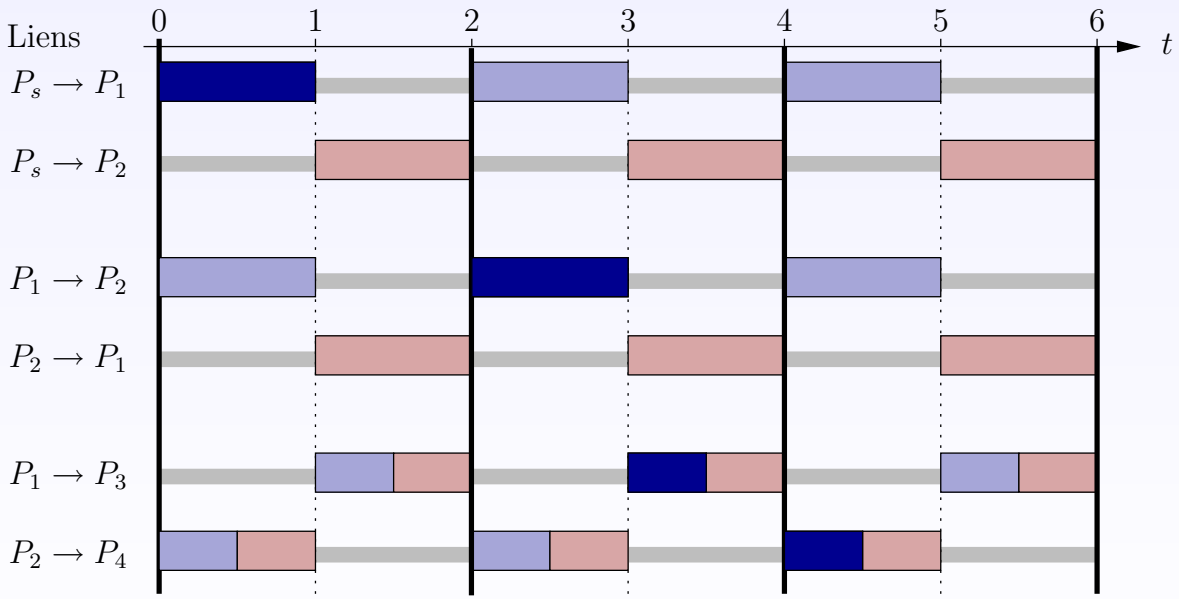
Reconstruction d'une période

- Quantités rationnelles de messages dans chaque couplage
- Durée de la période = PPCM des dénominateurs des quantités de messages dans chaque couplage
- Période découpée en intervalles représentant les couplages
- Affectation des transferts en respectant les dépendances (un processeur retransmet lors d'une période les messages reçus à la période précédente)



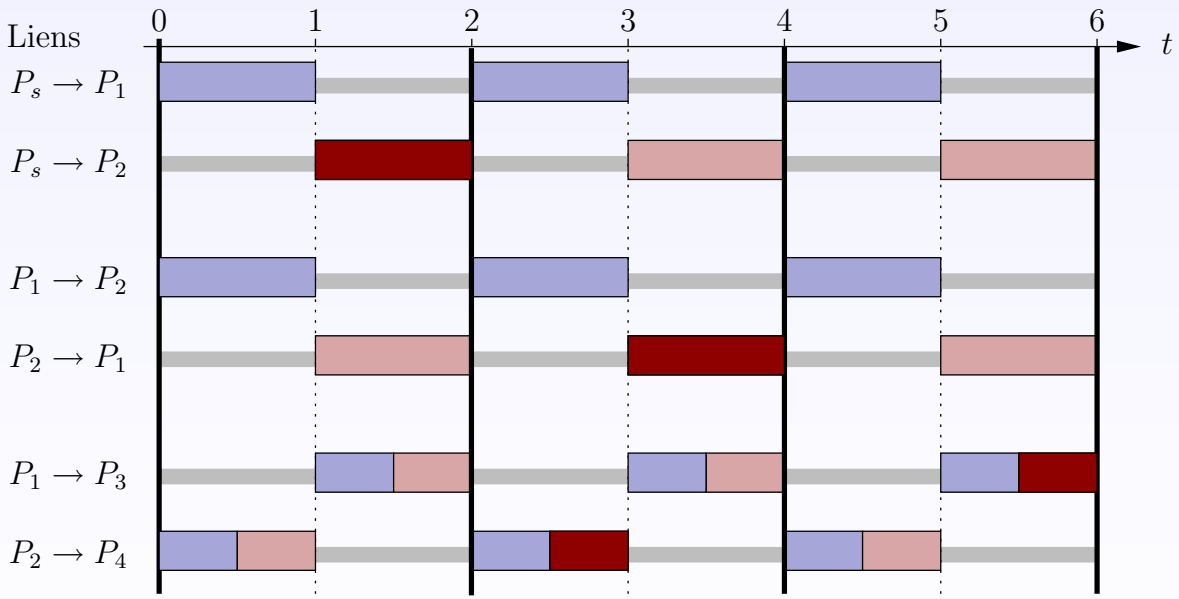
Reconstruction d'une période

- Quantités rationnelles de messages dans chaque couplage
- Durée de la période = PPCM des dénominateurs des quantités de messages dans chaque couplage
- Période découpée en intervalles représentant les couplages
- Affectation des transferts en respectant les dépendances
  (un processeur retransmet lors d'une période les messages reçus à la période précédente)



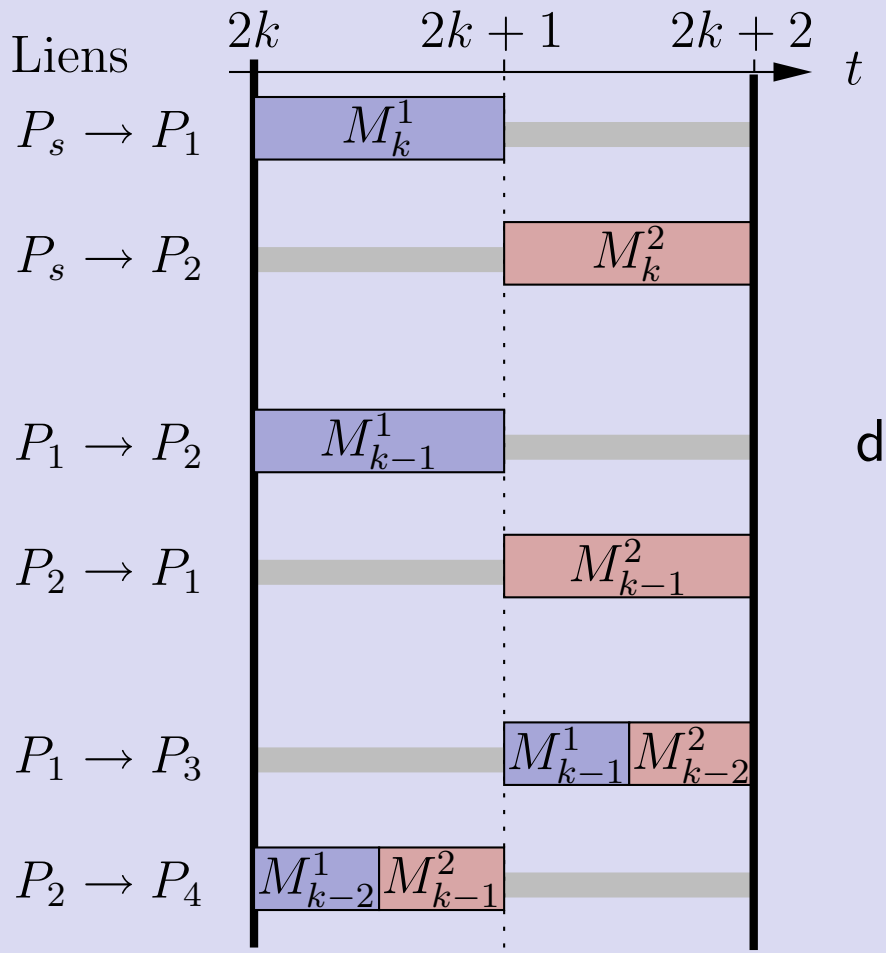
Reconstruction d'une période

- Quantités rationnelles de messages dans chaque couplage
- Durée de la période = PPCM des dénominateurs des quantités de messages dans chaque couplage
- Période découpée en intervalles représentant les couplages
- Affectation des transferts en respectant les dépendances
  (un processeur retransmet lors d'une période les messages reçus à la période précédente)



Reconstruction d'une période

Période de l'ordonnancement



de

description d'une période par intervalles

Résultats sous le modèle bidirectionnel

Pour la diffusion de messages

- ordonnancement périodique de débit optimal
- asymptotiquement optimal pour le temps d'exécution
- description compacte (période décrite par intervalles)

Extension à d'autres primitives de communications collectives :

- Étude similaire pour la réduction, la distribution
- Par contre le multicast (diffusion restreinte) est NP-complet, et non-APX

Diffusion en régime permanent

Position du problème et bibliographie

Résolution efficace sous le modèle bidirectionnel

Complexité sous le modèle unidirectionnel

Heuristiques à un seul arbre

Expérimentations

Spécificités du modèle unidirectionnel

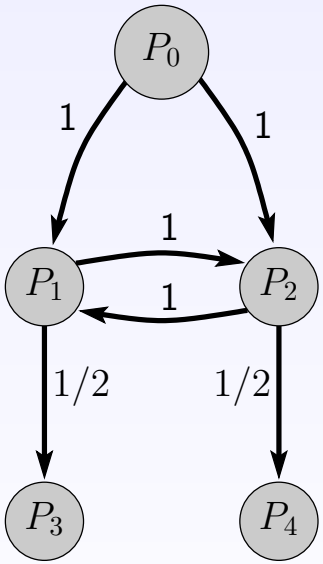
- Ensembles de communication pouvant avoir lieu simultanément : couplages dans le graphe G (non biparti)
- Pas de théorème de König pour les graphes quelconques
- Pas de lien direct entre le degré sortant d'un nœud et la somme des pondérations des couplages (= nombre chromatique)

$\Longrightarrow$ Autre méthode de résolution :

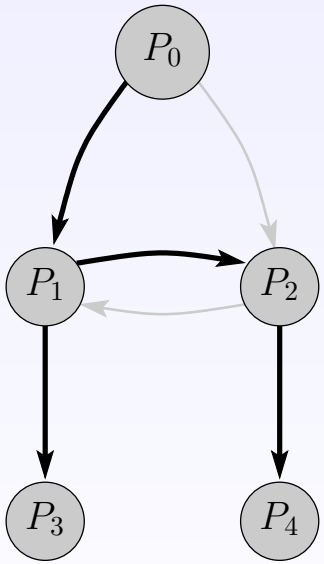
- Générale
  - pour toute primitive de communications collectives
  - pour tout modèle de communication
- Moins efficace, mais permet d'établir la complexité

Schémas d'allocation

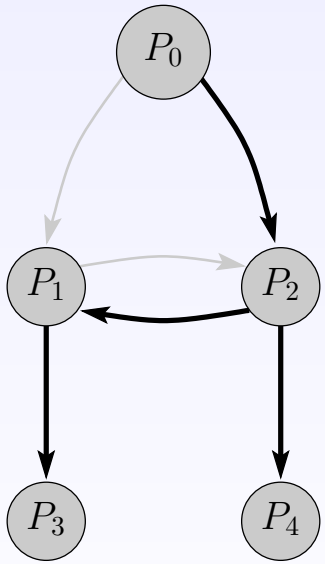
Ensemble des ressources utilisées pour la diffusion d'un message : arbre de diffusion.



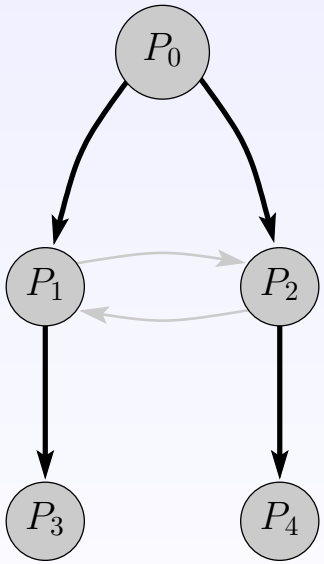
Topologie G



arbre A_1



arbre A_2



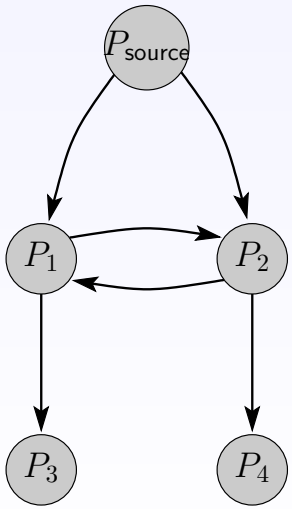
arbre A_3

Schémas de communications

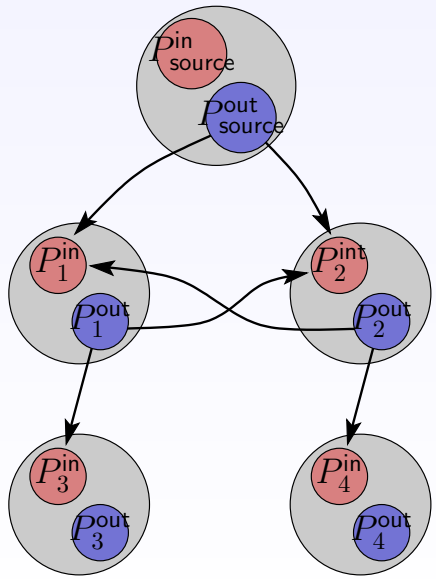
Pour un modèle de communication donné, ensemble de communications pouvant s'exécuter simultanément.

- un-port unidirectionnel : couplages dans G
- un-port bidirectionnel : couplages dans le graphe biparti G_B. Chaque processeur P_i est transformé en :
  - un processeur pour les émissions P_i^{out}
  - un processeur pour les réceptions P_i^{in}

G :



G_B :



Formulation du problème

Optimiser le débit d'une série de diffusions :

- trouver un ensemble pondéré d'arbres de diffusion
- trouver un ensemble pondéré de couplages

tels que

- la somme des débits des arbres est le débit total
- toutes les communications induites par les arbres peuvent être effectuées par les couplages
- la somme des pondérations des couplages est inférieure à 1

Une fois obtenus ces ensembles pondérés, on peut reconstruire un ordonnancement périodique de débit optimal (comme précédemment).

Formalisation : programme linéaire

Soient

- (A_a, x_a) les arbres de diffusions et leur pondération
- (C_c, y_c) les couplages et leur pondération

Ce programme linéaire fournit le débit optimal :

Maximiser $\rho_{\text{opt}} = \sum_{A_a \in \mathcal{A}} x_a$

Sous les contraintes :

$$\sum_{C_c \in \mathcal{C}} y_c \leq 1$$

$$\forall (i,j) \in E, \quad \sum_{\substack{C_c \in \mathcal{C} \\ (i,j) \in C_c}} y_c \geq \sum_{\substack{A_a \in \mathcal{A} \\ (i,j) \in A_a}} x_a \cdot c_{i,j}$$

$$\forall A_a \in \mathcal{A}, \quad x_a \geq 0$$

$$\forall C_c \in \mathcal{C}, \quad y_c \geq 0$$

Problème : nombre de variables exponentiel

On ne va pas manipuler directement ce programme linéaire.

Formalisation : programme linéaire

Soient

- (A_a, x_a) les arbres de diffusions et leur pondération
- (C_c, y_c) les couplages et leur pondération

Ce programme linéaire fournit le débit optimal :

Maximiser $\rho_{\text{opt}} = \sum_{A_a \in \mathcal{A}} x_a$

Sous les contraintes :

$$\sum_{C_c \in \mathcal{C}} y_c \leq 1$$

$$\forall (i,j) \in E, \quad \sum_{\substack{C_c \in \mathcal{C} \\ (i,j) \in C_c}} y_c \geq \sum_{\substack{A_a \in \mathcal{A} \\ (i,j) \in A_a}} x_a \cdot c_{i,j}$$

$$\forall A_a \in \mathcal{A}, \quad x_a \geq 0$$

$$\forall C_c \in \mathcal{C}, \quad y_c \geq 0$$

Problème : nombre de variables exponentiel

On ne va pas manipuler directement ce programme linéaire.

Existence d'une solution compacte

Théorème

Il existe une solution au problème de la diffusion pipelinée, de débit optimal, décrite à l'aide d'un ensemble pondéré de N_1 arbres de diffusion et d'un ensemble pondéré de N_2 couplages de communications, tels que

- N_1 et N_2 sont polynomiaux en la taille des donnés,
- le codage des pondérations, des arbres et des couplages est polynomial en la taille des données.

Preuve :

propriété du programme linéaire : une solution est atteinte en un sommet du polyèdre.

- Ce résultat montre que le problème appartient à la classe NP.

Résolution du programme linéaire : Ellipsoïdes

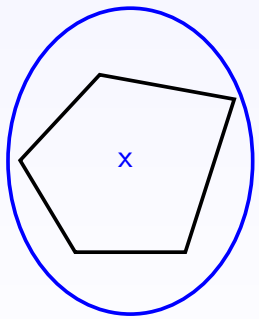
- **Optimisation forte** : Étant donné un polyèdre P et un vecteur C, trouver un vecteur qui maximise $C^T x$ ou prouver $P = \emptyset$.
- **Séparation forte** : Étant donné un vecteur x, décider si $x \in P$ et sinon, exhiber une contrainte non satisfaite par x.

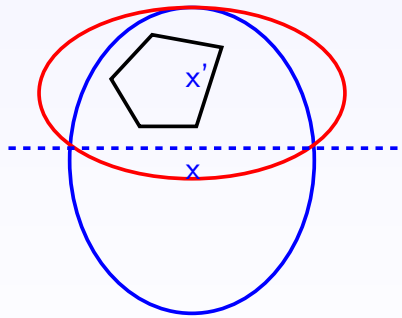
Théorème

S'il existe un oracle polynomial pour résoudre le problème de **Séparation forte** dans le **dual**, alors le problème d'**Optimisation forte** peut être résolu en temps polynomial dans le **primal**.

étape élémentaire

Étant donné un polyèdre P, trouver un point dans P ou prouver que P est vide.





Programme linéaire dual

Minimiser U_1

Sous les contraintes :

$$\forall A_a \in \mathcal{A} \quad \sum_{\substack{e_k=(i,j) \\ e_k \in A_a}} c_{i,j}\ U_{k+1} \geq 1$$

$$\forall C_c \in \mathcal{C} \quad \sum_{e_k \in C_l} U_{k+1} \leq U_1$$

$$U \geq 0$$

Construction d'un oracle de séparation
(Étant donné U, trouver une contrainte non satisfaite, ou vérifier que $U \in K$)

- immédiat
- $\Leftrightarrow$ trouver un couplage de poids maximum
- $\Leftrightarrow$ trouver un arbre de poids minimum

Programme linéaire dual

Minimiser U_1

Sous les contraintes :

$$\forall A_a \in \mathcal{A} \quad \sum_{\substack{e_k=(i,j) \\ e_k \in A_a}} c_{i,j}\ U_{k+1} \geq 1$$

$$\forall C_c \in \mathcal{C} \quad \sum_{e_k \in C_l} U_{k+1} \leq U_1$$

$$U \geq 0$$

Construction d'un oracle de séparation
(Étant donné U, trouver une contrainte non satisfaite, ou vérifier que $U \in K$)

- immédiat
- $\Leftrightarrow$ trouver un couplage de poids maximum
- $\Leftrightarrow$ trouver un arbre de poids minimum

Programme linéaire dual

Minimiser U_1

Sous les contraintes :

$$\forall A_a \in \mathcal{A} \quad \sum_{\substack{e_k=(i,j) \\ e_k \in A_a}} c_{i,j}\, U_{k+1} \geq 1$$

$$\forall C_c \in \mathcal{C} \quad \sum_{e_k \in C_l} U_{k+1} \leq U_1$$

$$U \geq 0$$

Construction d'un oracle de séparation
(Étant donné U, trouver une contrainte non satisfaite, ou vérifier que $U \in K$)

- immédiat
- $\Leftrightarrow$ trouver un couplage de poids maximum
- $\Leftrightarrow$ trouver un arbre de poids minimum

Programme linéaire dual

Minimiser U_1

Sous les contraintes :

$$\forall A_a \in \mathcal{A} \quad \sum_{\substack{e_k=(i,j)\\ e_k \in A_a}} c_{i,j}\, U_{k+1} \geq 1$$

$$\forall C_c \in \mathcal{C} \quad \sum_{e_k \in C_l} U_{k+1} \leq U_1$$

$$U \geq 0$$

Construction d'un oracle de séparation
(Étant donné U, trouver une contrainte non satisfaite, ou vérifier que $U \in K$)

- immédiat
- $\Leftrightarrow$ trouver un couplage de poids maximum
- $\Leftrightarrow$ trouver un arbre de poids minimum

Programme linéaire dual

Minimiser U_1

Sous les contraintes :

$$\forall A_a \in \mathcal{A} \quad \sum_{\substack{e_k=(i,j) \\ e_k \in A_a}} c_{i,j}\ U_{k+1} \geq 1$$

$$\forall C_c \in \mathcal{C} \quad \sum_{e_k \in C_l} U_{k+1} \leq U_1$$

$$U \geq 0$$

Construction d'un oracle de séparation
(Étant donné U, trouver une contrainte non satisfaite, ou vérifier que $U \in K$)

- immédiat
- $\Leftrightarrow$ trouver un couplage de poids maximum
- $\Leftrightarrow$ trouver un arbre de poids minimum

Résultat de complexité

Théorème

On peut déterminer en temps polynomial un ordonnancement de débit optimal.

Limitation :
Méthode des ellipsoïdes (pour traiter des programmes linéaires de taille exponentielle)

$\rightsquigarrow$ méthode peu utilisable en pratique

Résultat de complexité

Théorème

On peut déterminer en temps polynomial un ordonnancement de débit optimal.

Limitation :
Méthode des ellipsoïdes (pour traiter des programmes linéaires de taille exponentielle)

$\rightsquigarrow$ méthode peu utilisable en pratique

Diffusion en régime permanent

Position du problème et bibliographie

Résolution efficace sous le modèle bidirectionnel

Complexité sous le modèle unidirectionnel

Heuristiques à un seul arbre

Expérimentations

Heuristiques pour un arbre de diffusion

- Utiliser un ensemble pondéré d'arbres de diffusion engendre un contrôle assez lourd

  $\rightsquigarrow$ un arbre de diffusion efficace ?

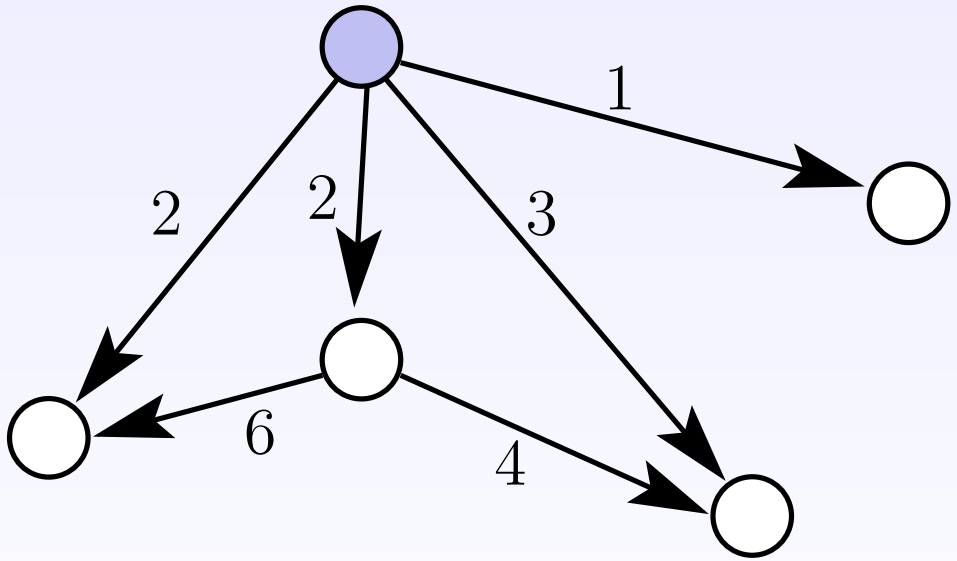
- Trouver l'arbre de meilleur débit est NP-complet

  DEGREE-CONSTRAINT-SPANNING-TREE

- Conception d'heuristiques :
  1. Heuristiques fondées sur des techniques de graphe
  2. Heuristiques s'inspirant d'une solution du programme linéaire (pour chacun des modèles : un-port ou multi-port)

Élagage simple de la plate-forme

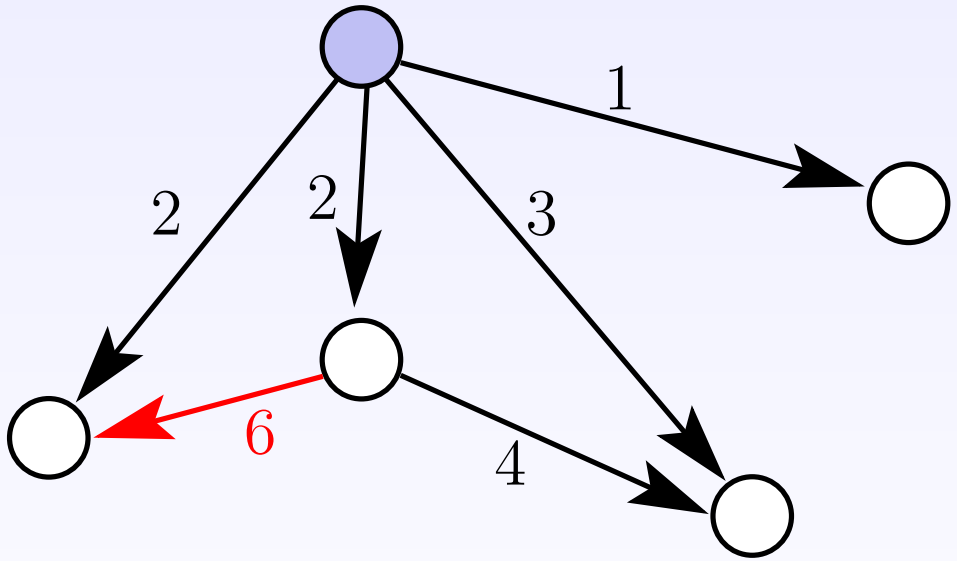
- Principe : supprimer les arêtes de poids maximum, jusqu'à obtenir un arbre
- Exemple :



Topologie, poids des arêtes $c_{i,j}$

Élagage simple de la plate-forme

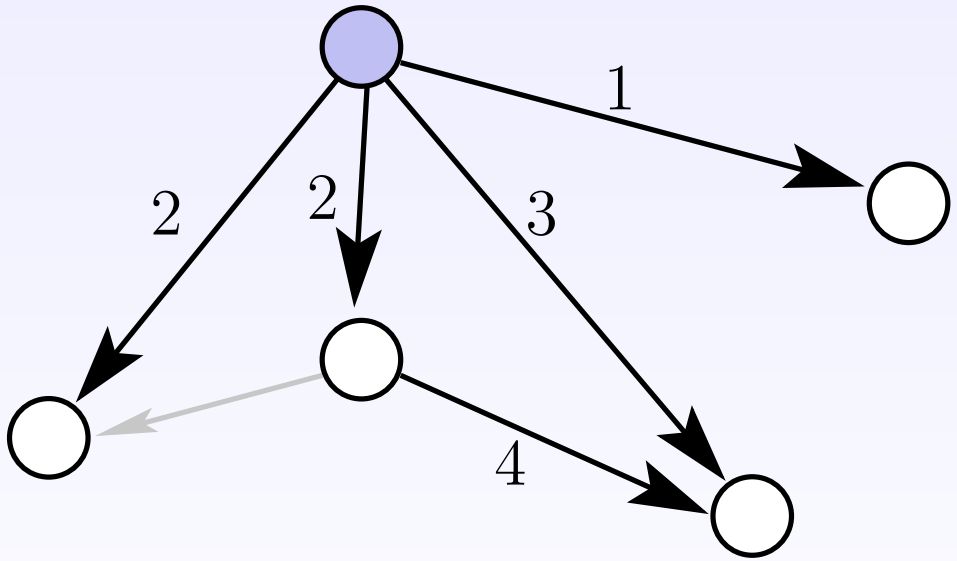
- Principe : supprimer les arêtes de poids maximum, jusqu'à obtenir un arbre
- Exemple :



On choisit puis on supprime l'arête de poids maximum

Élagage simple de la plate-forme

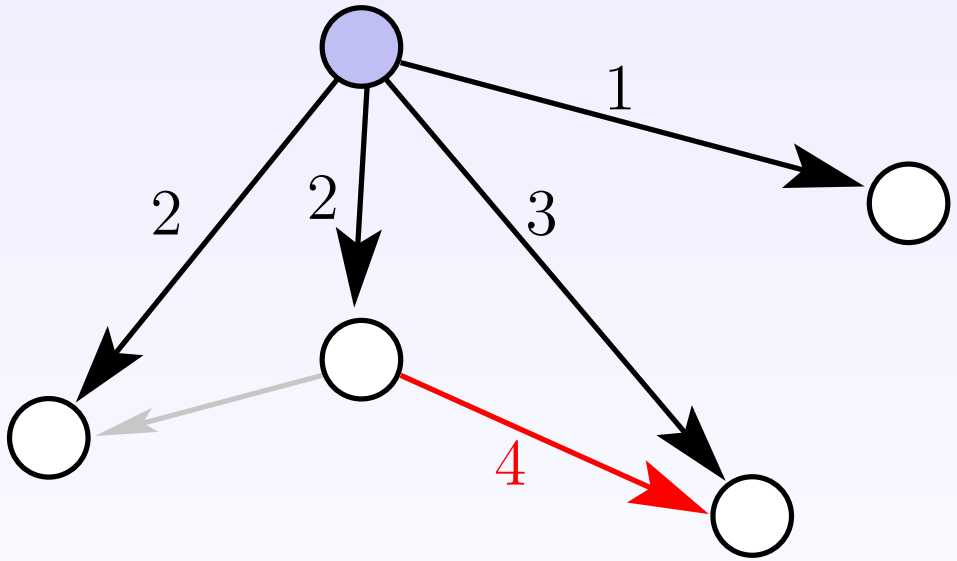
- Principe : supprimer les arêtes de poids maximum, jusqu'à obtenir un arbre
- Exemple :



On choisit puis on supprime l'arête de poids maximum

Élagage simple de la plate-forme

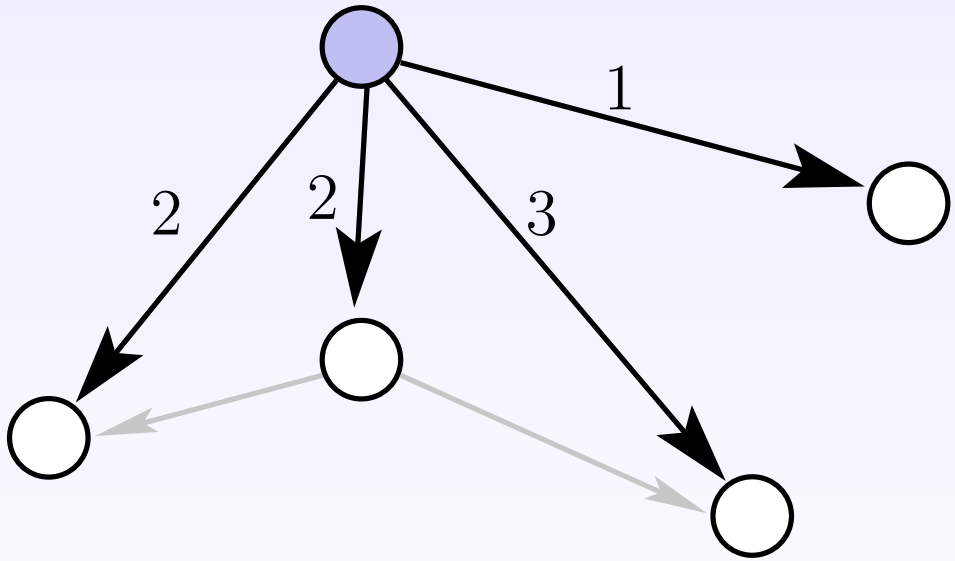
- Principe : supprimer les arêtes de poids maximum, jusqu'à obtenir un arbre
- Exemple :



On choisit puis on supprime l'arête de poids maximum

Élagage simple de la plate-forme

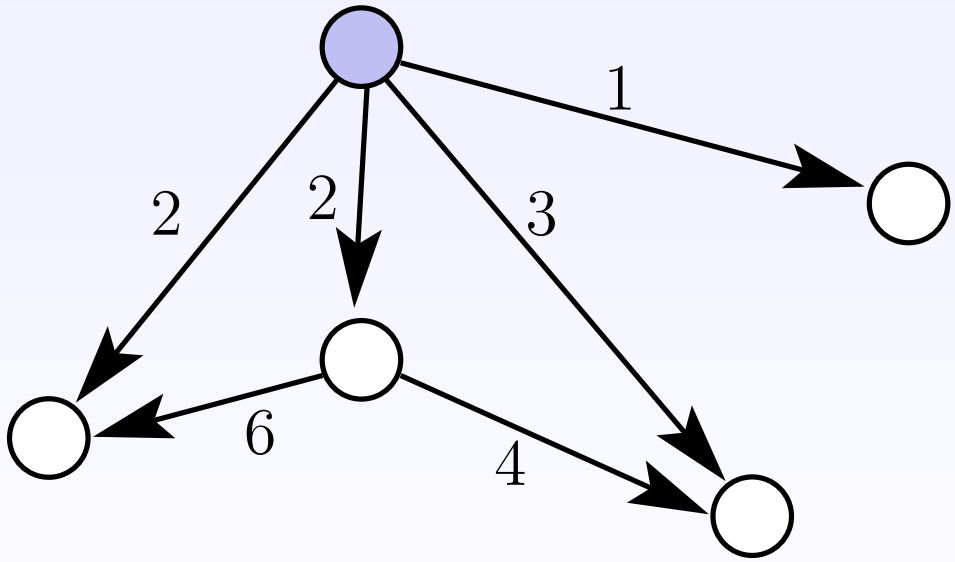
- Principe : supprimer les arêtes de poids maximum, jusqu'à obtenir un arbre
- Exemple :



Au final : la source doit émettre pendant un temps $2 + 2 + 3 + 1$
Débit réalisable : $1/8$

Élagage de plate-forme plus élaboré

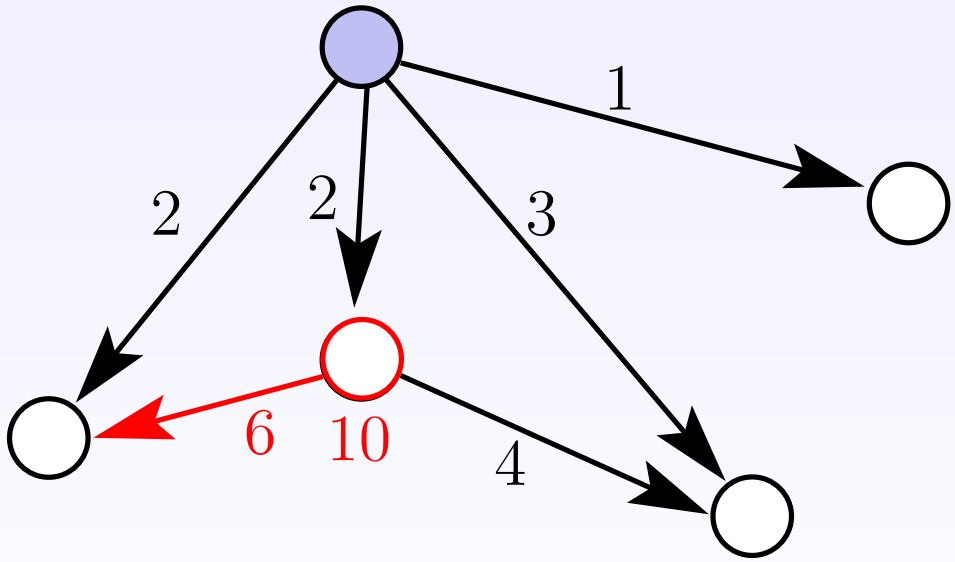
- Principe
  - à chaque étape, calculer le degré sortant pondéré de chaque nœud
  - supprimer une arête d'un nœud dont le degré sortant pondéré est maximum (en conservant la connexité du graphe)
- Exemple :



Topologie, poids des arêtes $c_{i,j}$

Élagage de plate-forme plus élaboré

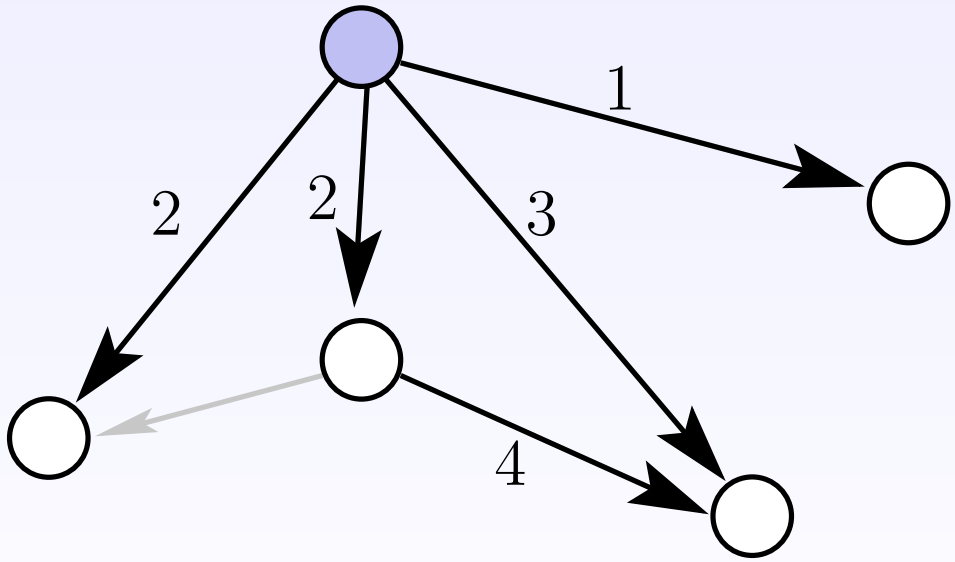
- Principe
  - à chaque étape, calculer le degré sortant pondéré de chaque nœud
  - supprimer une arête d'un nœud dont le degré sortant pondéré est maximum (en conservant la connexité du graphe)
- Exemple :



On choisit puis le nœud de degré sortant pondéré maximum, puis l'arête de poids maximum

Élagage de plate-forme plus élaboré

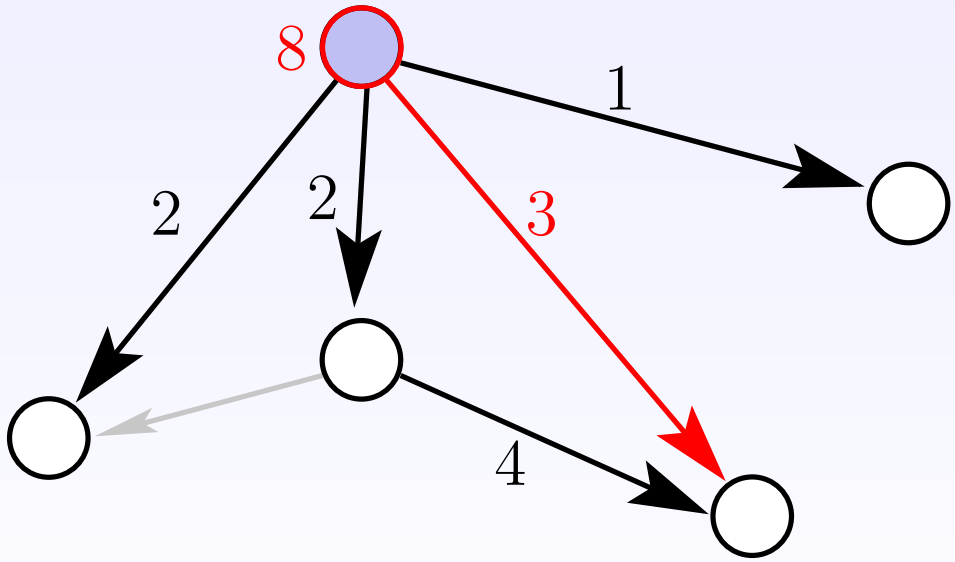
- Principe
  - à chaque étape, calculer le degré sortant pondéré de chaque nœud
  - supprimer une arête d'un nœud dont le degré sortant pondéré est maximum (en conservant la connexité du graphe)
- Exemple :



On choisit puis le nœud de degré sortant pondéré maximum, puis l'arête de poids maximum

Élagage de plate-forme plus élaboré

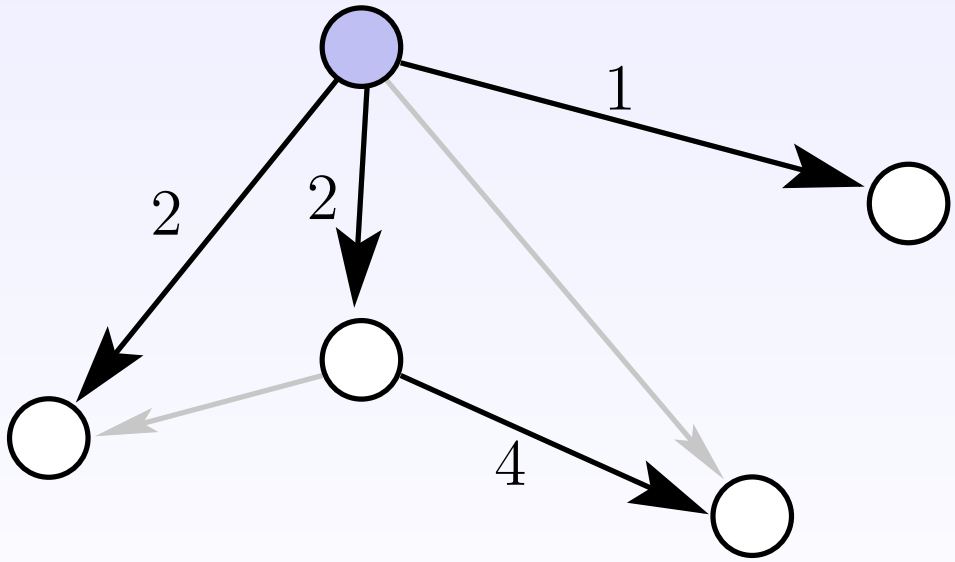
- Principe
  - à chaque étape, calculer le degré sortant pondéré de chaque nœud
  - supprimer une arête d'un nœud dont le degré sortant pondéré est maximum (en conservant la connexité du graphe)
- Exemple :



On choisit puis le nœud de degré sortant pondéré maximum, puis l'arête de poids maximum

Élagage de plate-forme plus élaboré

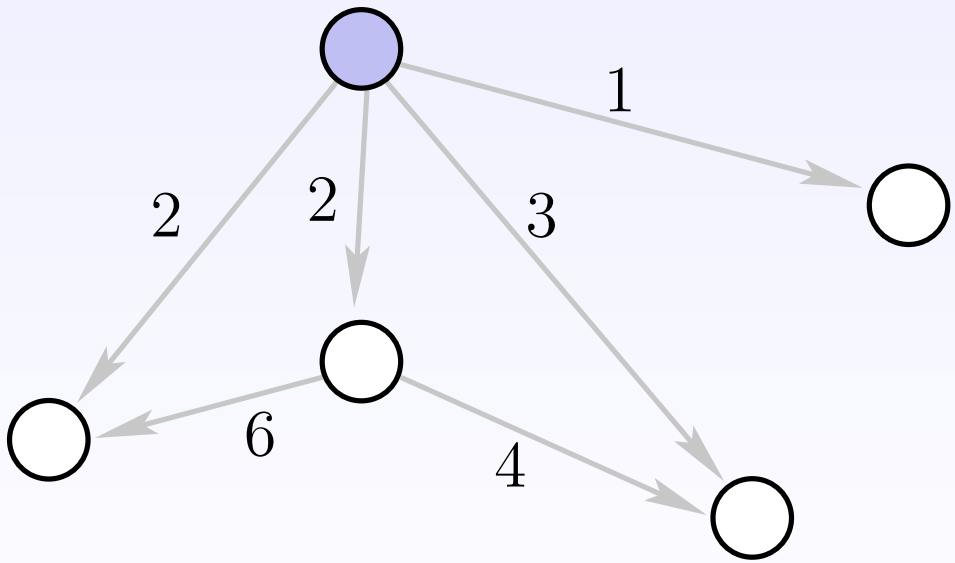
- Principe
  - à chaque étape, calculer le degré sortant pondéré de chaque nœud
  - supprimer une arête d'un nœud dont le degré sortant pondéré est maximum (en conservant la connexité du graphe)
- Exemple :



Débit réalisable : $1/5$

Faire grandir un arbre de degré sortant pondéré minimum

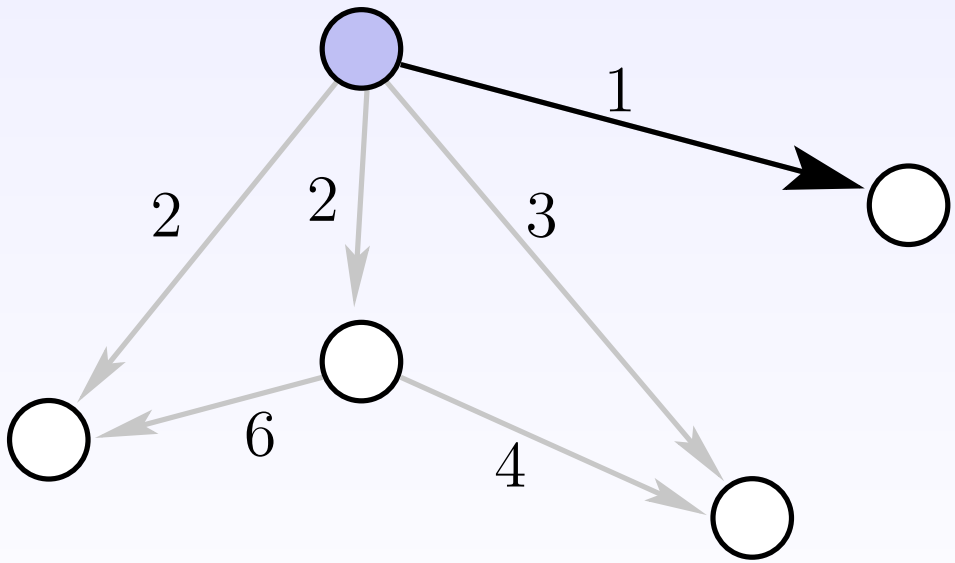
- Principe : construire un arbre comme le fait l'algorithme de Prim
- À chaque étape, rajouter une arête qui optimise une métrique
- Notre métrique :
  - minimiser le degré sortant pondéré de chaque nœud
- Exemple



Débit réalisable : 1/5

Faire grandir un arbre de degré sortant pondéré minimum

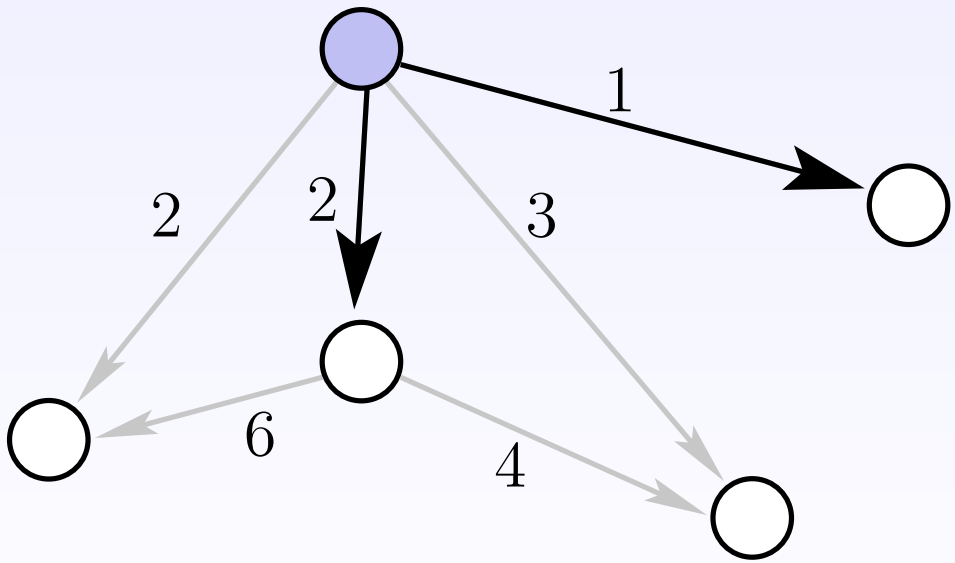
- Principe : construire un arbre comme le fait l'algorithme de Prim
- À chaque étape, rajouter une arête qui optimise une métrique
- Notre métrique :
  - minimiser le degré sortant pondéré de chaque nœud
- Exemple



Débit réalisable : 1/5

Faire grandir un arbre de degré sortant pondéré minimum

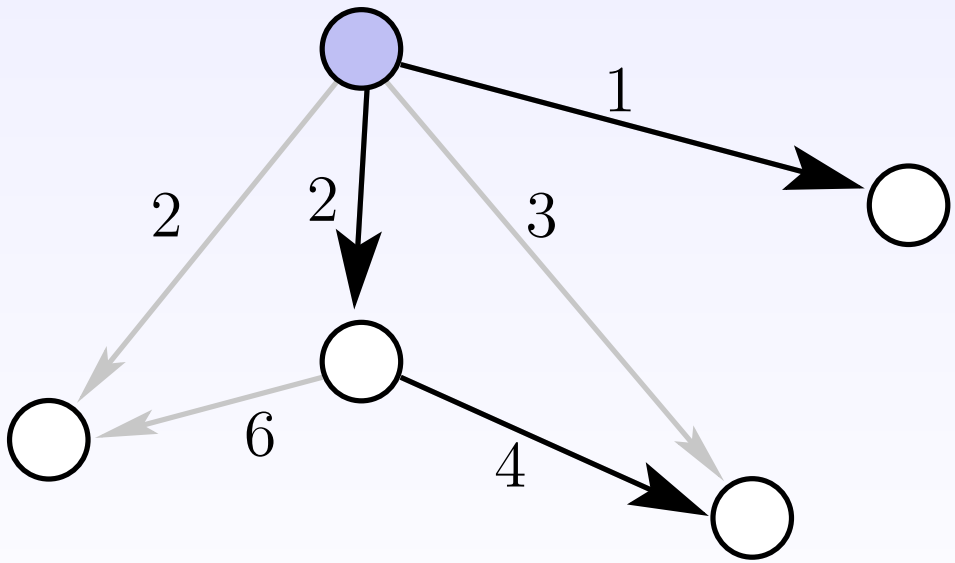
- Principe : construire un arbre comme le fait l'algorithme de Prim
- À chaque étape, rajouter une arête qui optimise une métrique
- Notre métrique :
  - minimiser le degré sortant pondéré de chaque nœud
- Exemple



Débit réalisable : 1/5

Faire grandir un arbre de degré sortant pondéré minimum

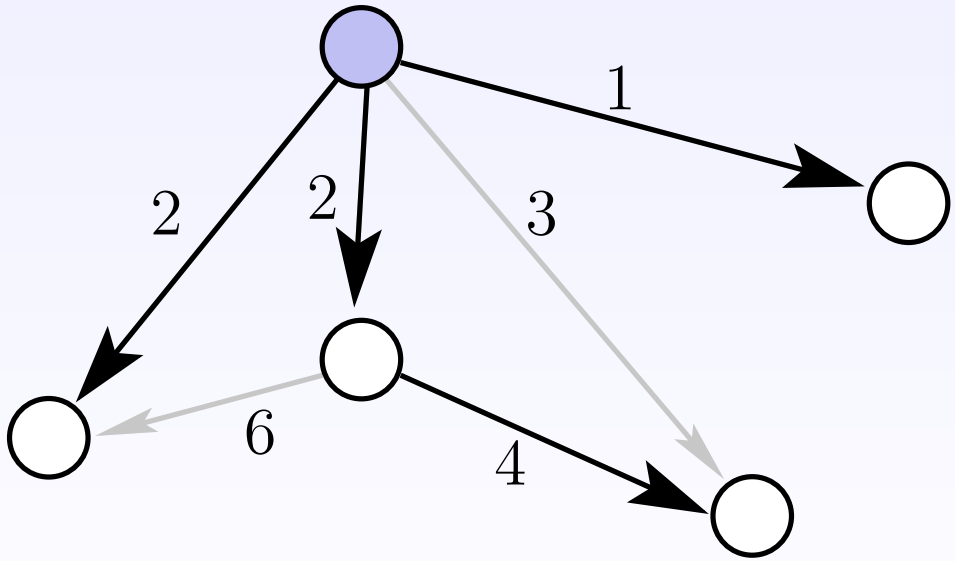
- Principe : construire un arbre comme le fait l'algorithme de Prim
- À chaque étape, rajouter une arête qui optimise une métrique
- Notre métrique :
  - minimiser le degré sortant pondéré de chaque nœud
- Exemple



Débit réalisable : 1/5

Faire grandir un arbre de degré sortant pondéré minimum

- Principe : construire un arbre comme le fait l'algorithme de Prim
- À chaque étape, rajouter une arête qui optimise une métrique
- Notre métrique :
  - minimiser le degré sortant pondéré de chaque nœud
- Exemple



Débit réalisable : $1/5$

Heuristiques fondées sur la programmation linéaire

1. Écrire et résoudre le programme linéaire correspondant

2. Construire le graphe étiqueté par le nombre total de messages par arêtes : $s(P_i \to P_j)$

3. Deux heuristiques pour construire un arbre à partir de ce graphe :
   1. Élagage du graphe, en supprimant les arêtes de poids minimal
   2. Faire grandir un arbre, en utilisant les arêtes de plus grand poids

Diffusion en régime permanent

Position du problème et bibliographie

Résolution efficace sous le modèle bidirectionnel

Complexité sous le modèle unidirectionnel

Heuristiques à un seul arbre

Expérimentations

Un autre modèle de communication

- Modèle un-port
  le plus utilisé, même sur des topologies irrégulières et hétérogènes
  nombre d'heuristiques pour la diffusion, en vue de minimiser le temps d'exécution
- Modèle multi-port
  - plusieurs transferts entrant ou sortant d'un nœud peuvent cohabiter
  - congestion : bande-passante limitée sur les liens et les interfaces réseau
  - modèle plus fluide : flots ininterrompus
  - proche de la problématique de régime permanent :
    Hong & Prasanna, IPDPS 2004 (flot de tâches indépendantes)

Peut-on adapter notre étude à d'autres modèles ?

oui ☺

- Adaptation de la la résolution efficace du un-port bidirectionnel pour le multi-port (et donc des heuristiques LP)

Un autre modèle de communication

- Modèle un-port
  le plus utilisé, même sur des topologies irrégulières et hétérogènes
  nombre d'heuristiques pour la diffusion, en vue de minimiser le temps d'exécution
- Modèle multi-port
  - plusieurs transferts entrant ou sortant d'un nœud peuvent cohabiter
  - congestion : bande-passante limitée sur les liens et les interfaces réseau
  - modèle plus fluide : flots ininterrompus
  - proche de la problématique de régime permanent :
    Hong & Prasanna, IPDPS 2004 (flot de tâches indépendantes)

Peut-on adapter notre étude à d'autres modèles ?

oui ☺

- Adaptation de la la résolution efficace du un-port bidirectionnel pour le multi-port (et donc des heuristiques LP)

Méthodologie des expérimentations

- Implantation distribuée générique
- Capable de suivre un scénario à un ou plusieurs arbres de diffusion
- Agents constitués de plusieurs *threads* : un par opérations (émission E, réception R, coordination C avec le maître)
- Mémoire tampon M de 20 messages
- Communication par socket TCP

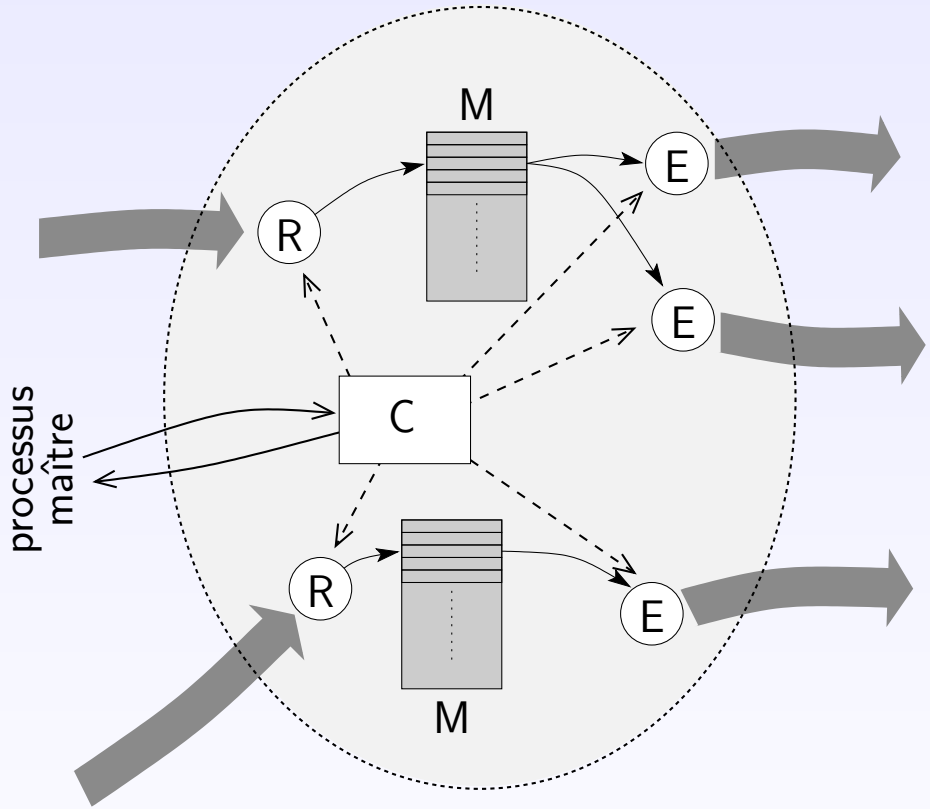
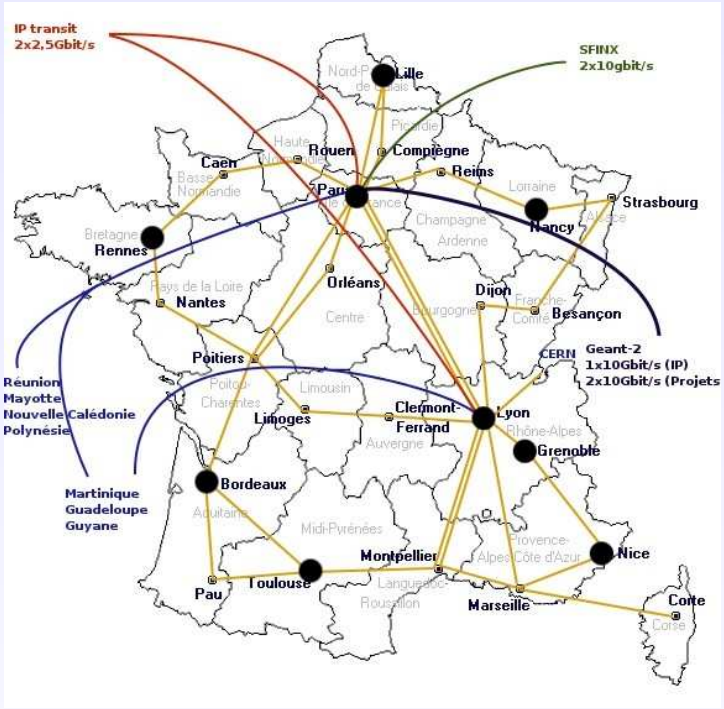
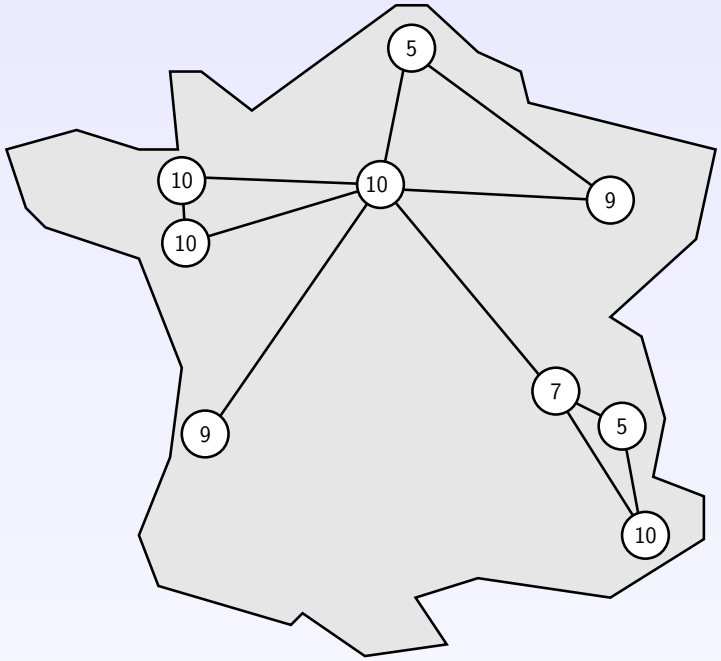


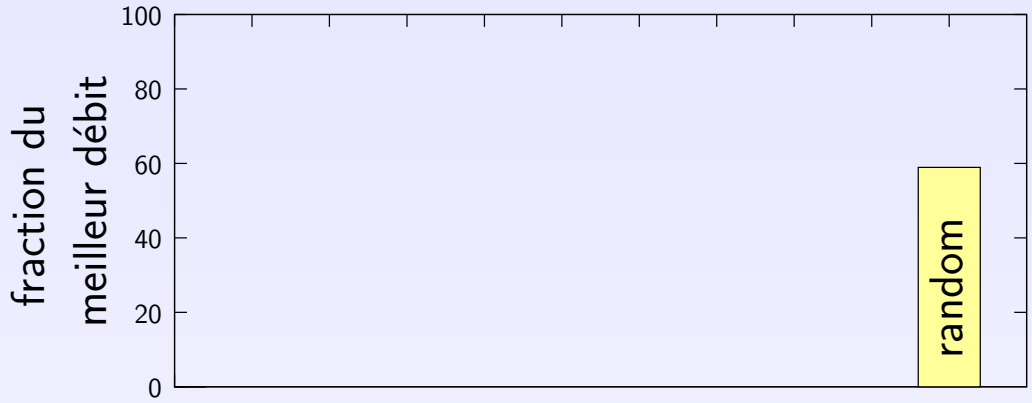
Plate-forme de test





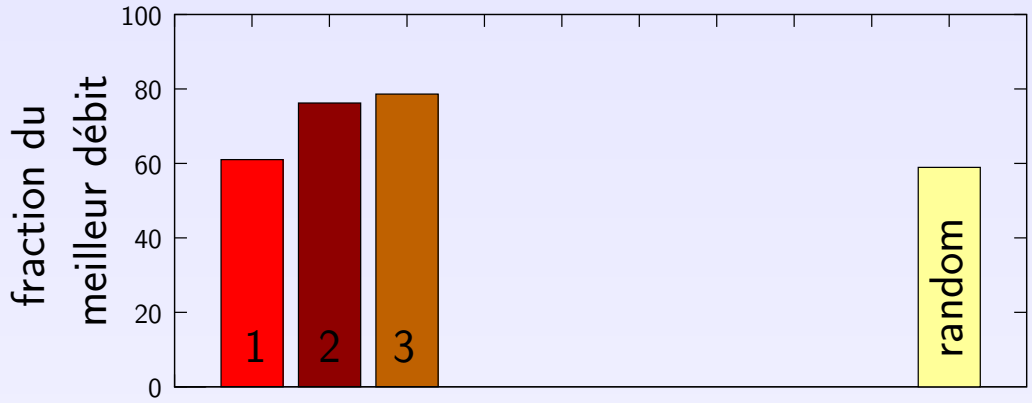
- Plate-forme de grille pour la recherche Grid5000
- 75 machines réparties sur 8 sites
- Mesures préalables de bande-passante

Résultats



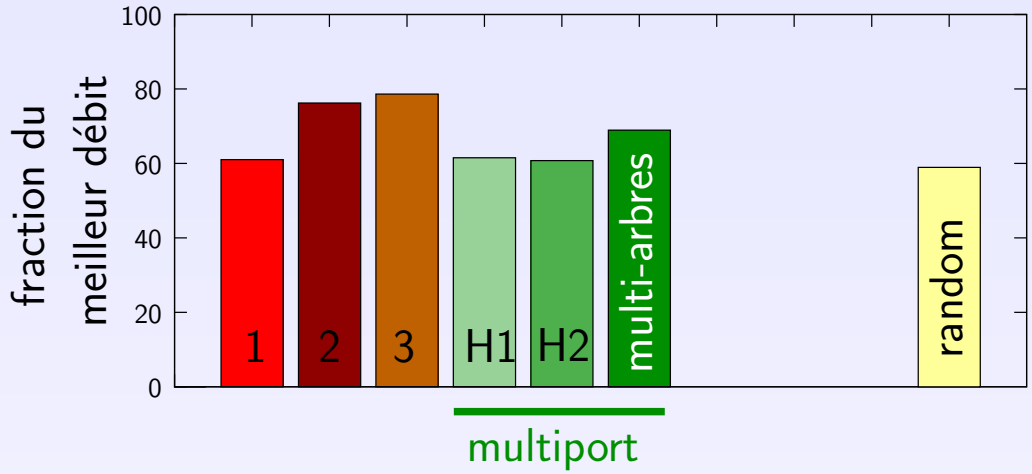
- Comparaison avec un arbre de diffusion aléatoire
- Heuristiques basées sur des techniques de graphe
  1. élagage simple
  2. élagage élaboré
  3. construction d'un arbre de degré sortant pondéré minimum
- Solution du PL sous le modèle multi-port
  1. H1, H2 : construction d'un arbre à partir de la solution du PL
  2. multi-arbres : décomposition en somme d'arbres
- Solution du PL sous le modèle un-port
  1. H1, H2 : construction d'un arbre à partir de la solution du PL
  2. multi-arbres : décomposition en somme d'arbres

Résultats



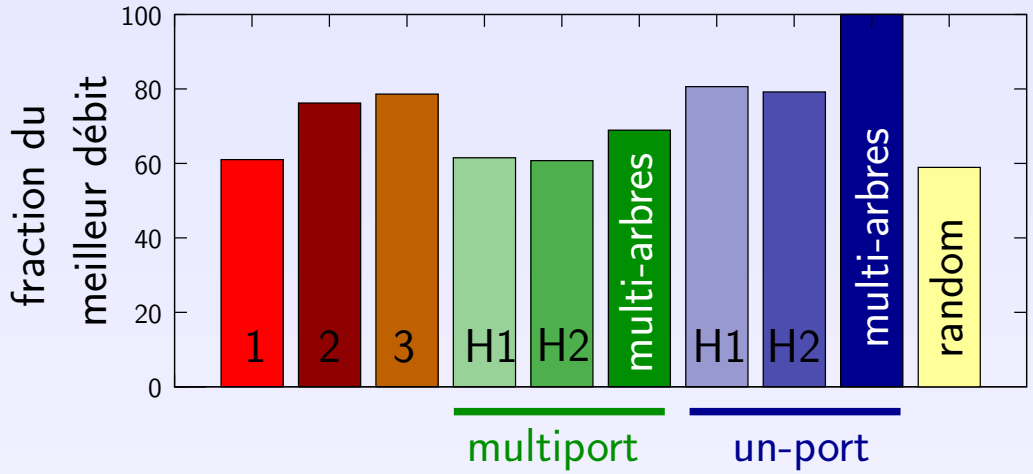
- Comparaison avec un arbre de diffusion aléatoire
- Heuristiques basées sur des techniques de graphe
  1. élagage simple
  2. élagage élaboré
  3. construction d'un arbre de degré sortant pondéré minimum
- Solution du PL sous le modèle multi-port
  1. H1, H2 : construction d'un arbre à partir de la solution du PL
  2. multi-arbres : décomposition en somme d'arbres
- Solution du PL sous le modèle un-port
  1. H1, H2 : construction d'un arbre à partir de la solution du PL
  2. multi-arbres : décomposition en somme d'arbres

Résultats



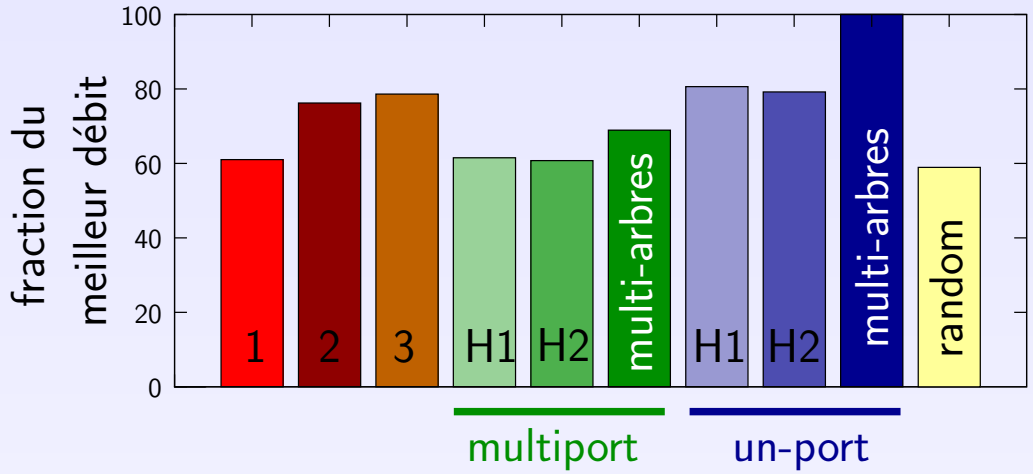
- Comparaison avec un arbre de diffusion aléatoire
- Heuristiques basées sur des techniques de graphe
  1. élagage simple
  2. élagage élaboré
  3. construction d'un arbre de degré sortant pondéré minimum
- Solution du PL sous le modèle multi-port
  1. H1, H2 : construction d'un arbre à partir de la solution du PL
  2. multi-arbres : décomposition en somme d'arbres
- Solution du PL sous le modèle un-port
  1. H1, H2 : construction d'un arbre à partir de la solution du PL
  2. multi-arbres : décomposition en somme d'arbres

Résultats



- Comparaison avec un arbre de diffusion aléatoire
- Heuristiques basées sur des techniques de graphe
  1. élagage simple
  2. élagage élaboré
  3. construction d'un arbre de degré sortant pondéré minimum
- Solution du PL sous le modèle multi-port
  1. H1, H2 : construction d'un arbre à partir de la solution du PL
  2. multi-arbres : décomposition en somme d'arbres
- Solution du PL sous le modèle un-port
  1. H1, H2 : construction d'un arbre à partir de la solution du PL
  2. multi-arbres : décomposition en somme d'arbres

Discussion



- Bon résultats de l'arbre aléatoire :
  - utilise la topologie construite
  - pas d'« allers-retours » entre les sites
- Meilleur résultat : décomposition en somme d'arbres de la solution du programme linéaire en un-port
- Résultats relativement mauvais des heuristiques multiport
  - modélisation inadéquate pour des flux TCP
  - mesure délicate de la bande-passante de sortie d'un nœud
- Débit de l'ordre de 5Mo/s (une seule connexion TCP par communication)

Conclusion et perspectives

Conclusion

- Cadre d'étude pour le régime permanent
- Adaptables à d'autres modèles de communication

- Complexité des communications collectives sous l'angle de l'optimisation du débit
- Résolution efficace dans certains modèles
- Implantation « grandeur nature » pour la diffusion

Autres travaux

- Communications collectives en régime permanent
  - Même étude pour la distribution, la réduction
  - Multicast et calcul des préfixes NP-complet
- Problèmes d'ordonnancement en régime permament
  - Pour des applications plus réalistes (pls. collections de tâches)
  - Pour des plates-formes plus réalistes (grille de calcul)
  - Impact des contraintes mémoires
- Collaborations
  - Modélisation de grilles (H. Casanova et Y. Yang)
  - Messages de retour pour les tâches divisibles (V. Rehn)
  - Requêtes dans les réseaux (P. Primet et J. Zheng)
  - VoroNet (A.-M. Kermarrec et E. Rivière)

Questions ouvertes

- Malgré la relaxation en régime permanent, le multicast (diffusion restreinte) reste NP-complet.
  $\rightsquigarrow$ randomisation *(network coding)* ?

- Contraintes mémoires importantes :
  grande période d'ordonnancement $\rightsquigarrow$ grande taille de mémoire nécessaire

- Modélisation de la plate-forme difficile à obtenir

Perspectives

- Modèles de topologies à grande échelle
  - topologies hiérarchiques (modélisation de grilles)
  - topologie virtuelle
- Ordonnancement dynamique et décentralisé
  - calcul de flot décentralisé
  - randomisation (network coding)
  - utilisation d'une topologie pair-à-pair