Modélisation des réseaux

Frédéric Vivien

e-mail: Frederic.Vivien@ens-lyon.fr

4 novembre 2005

Plan du cours

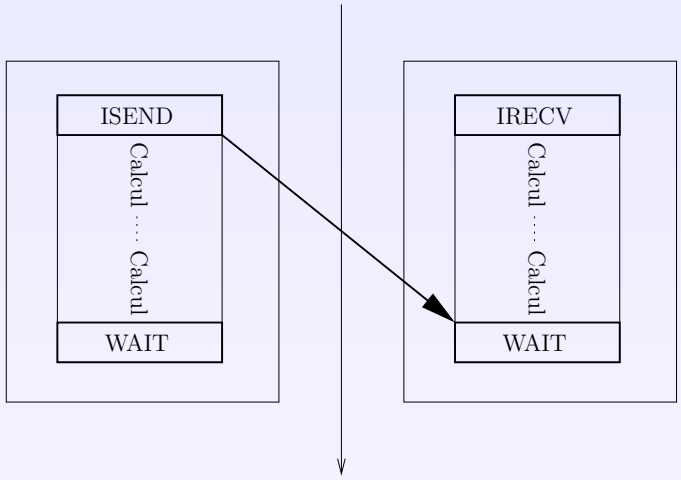
1. Recouvrement calcul/communications
2. Interférences entre calculs et communications
3. Modélisation des plates-formes distribuées
4. Allocation de bandes passantes

Plan du cours

Vision idyllique

- Pendant l'envoi et la réception de messages, on peut calculer.
- La puissance de calcul n'est aucunement affectée par les communications.
- Les communications sont réalisées au moyen de primitives non bloquantes.

Schéma expérimental



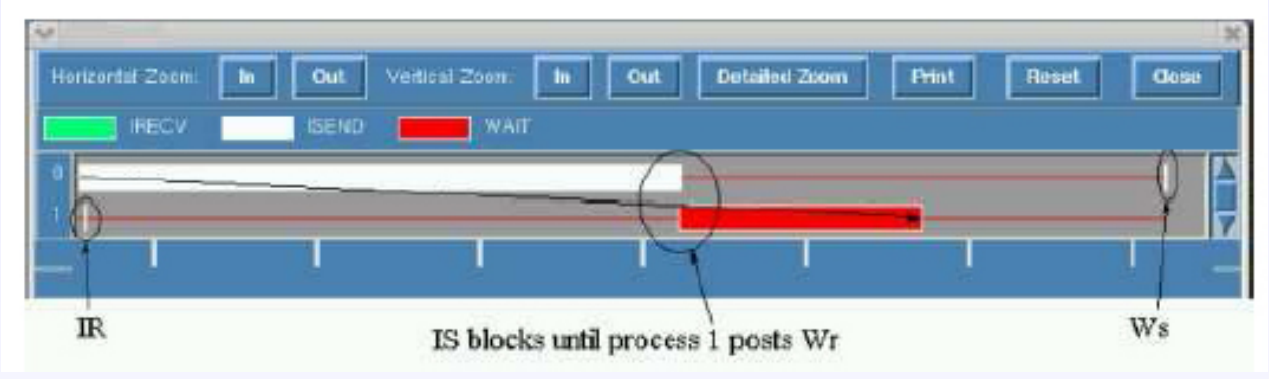
- Deux processus (sur deux processeurs différents) : le premier envoie un message au second.
- Les deux processus exécutent « en même temps » leurs commandes d'envoi et de réception asynchrone.
- Après une (longue) période de calcul, les deux processus se placent « en même temps » en attente de la terminaison des communications.

Comportement réel : message de 1k octet

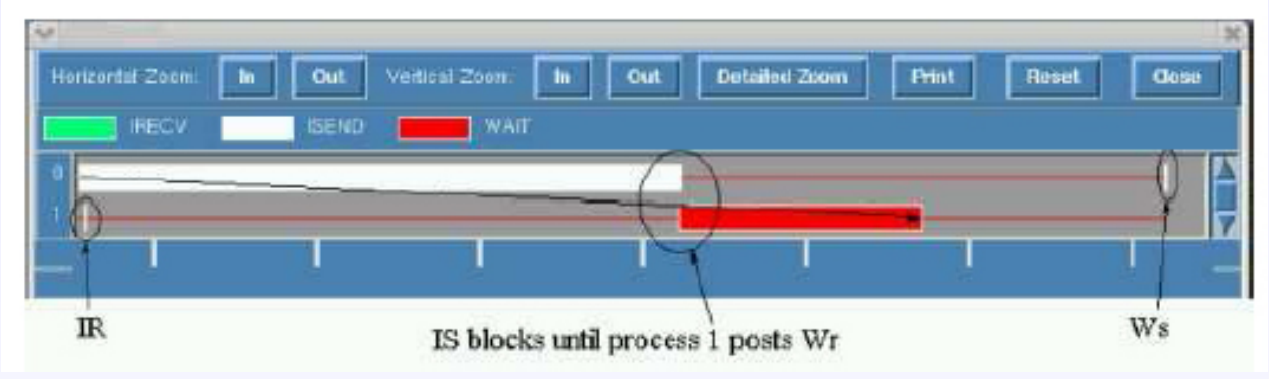
Comportement réel : message de 1k octet

Comportement attendu : les primitives ont un comportement apparemment non-bloquant.

Comportement réel : message de 60k octet (1)

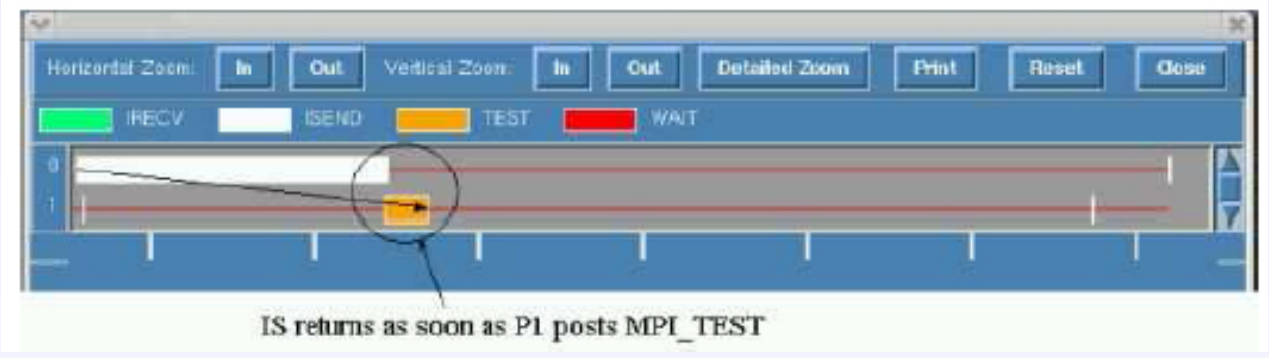


Comportement réel : message de 60k octet (1)



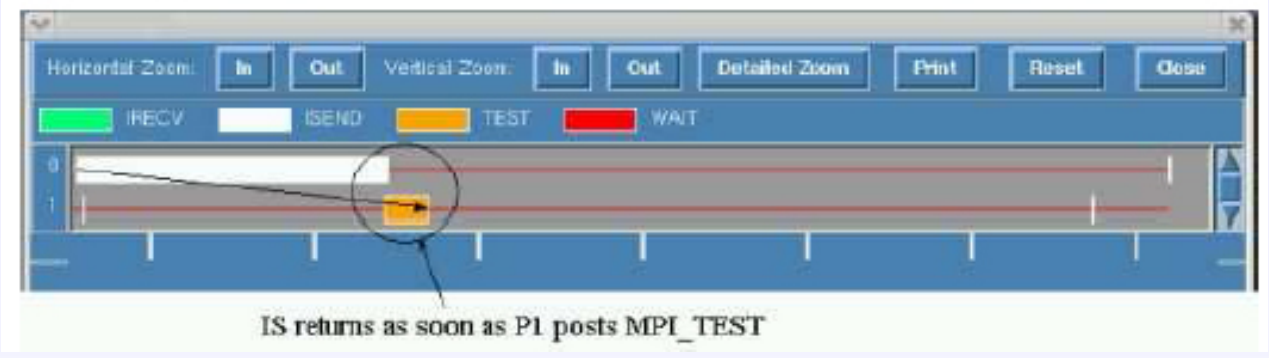
L'envoi censé être non-bloquant est en fait bloquant.

Comportement réel : message de 60k octet (2)



Message de 60k octet avec test non bloquant de terminaison de la communication au milieu de la phase de calcul du processus récepteur.

Comportement réel : message de 60k octet (2)



Message de 60k octet avec test non bloquant de terminaison de la communication au milieu de la phase de calcul du processus récepteur.

L'envoi se bloque jusqu'à l'émission par le récepteur du test de terminaison.

Comportement réel : autres résultats

- Les symptômes sont les mêmes que l'on envoie un message de 60 k octet ou 3 messages de 20 k octet.
- Quand on augmente la taille du buffer socket de TCP/IP à 64 ko, l'envoi a un comportement non-bloquant.

Interprétation

- Passée une certaine taille (cumulée) de messages, les tampons (*buffers*) de système ne permettent plus de recopier localement le message. MPI passe alors dans un protocole de rendez-vous qui nécessite la synchronisation des deux processus censés communiquer.
- Ce comportement n'est pas un bug. Il est inhérent aux limitations des architectures matérielles utilisées.

Plan du cours

Hypothèses classiques

- On peut calculer et communiquer simultanément (au moyen de threads différents et/ou de communications asynchrones).
- Les calculs et les communications n'ont aucune influence l'un sur l'autre.

Hypothèses classiques

- On peut calculer et communiquer simultanément (au moyen de threads différents et/ou de communications asynchrones).
- Les calculs et les communications n'ont aucune influence l'un sur l'autre.

Que se passe-t-il dans la pratique ?

Hypothèses classiques

- On peut calculer et communiquer simultanément (au moyen de threads différents et/ou de communications asynchrones).
- Les calculs et les communications n'ont aucune influence l'un sur l'autre.

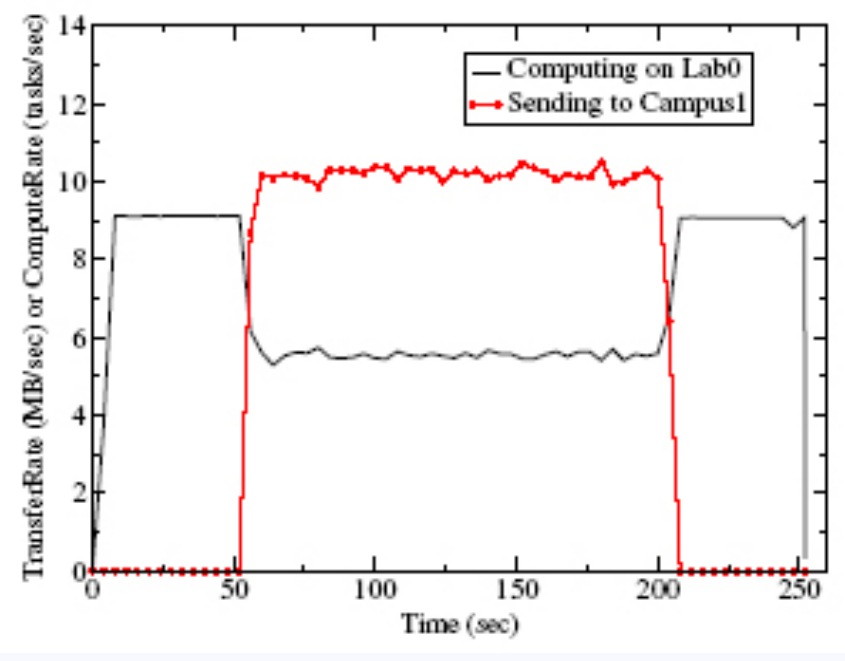
Que se passe-t-il dans la pratique ?

Les interférences existantes sont-elles négligeables ?

Contexte expérimental

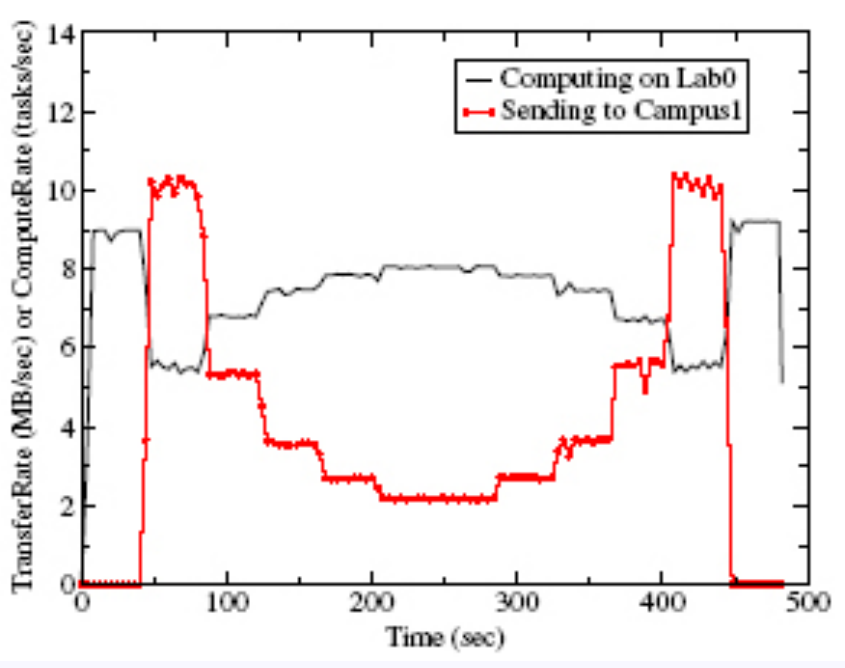
- Programme écrit en Java utilisant des *native threads*.
- Architecture des processeurs : Intel.
- Systèmes d'exploitation : FreeBSD, Linux 2.4 (Debian, RedHat) et Solaris (SunOS).
- Machines du laboratoire (Grail/UCSD), du campus (UCSD), de UCSB, et machines éloignées (Tennessee, Brésil, France).

Calcul et envoi simultanés (1)



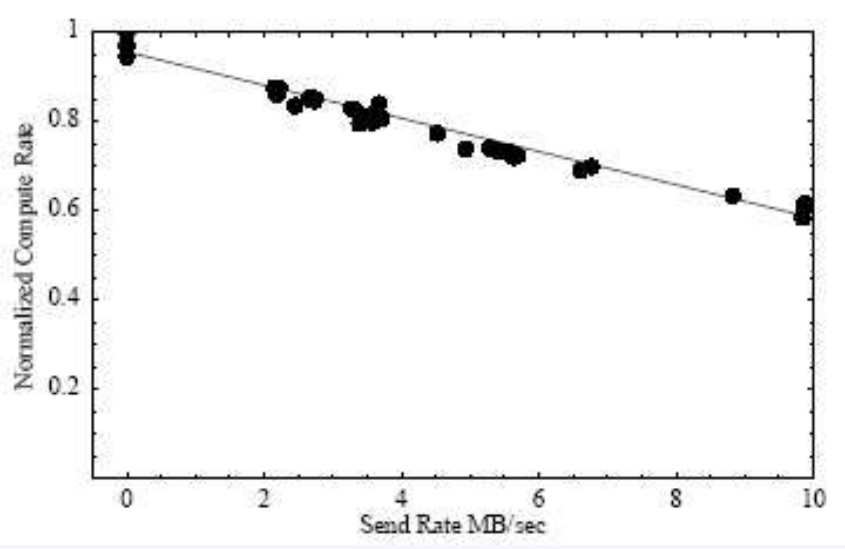
Un processeur calcule et envoie simultanément un message (débit constant).

Calcul et envoi simultanés (2)



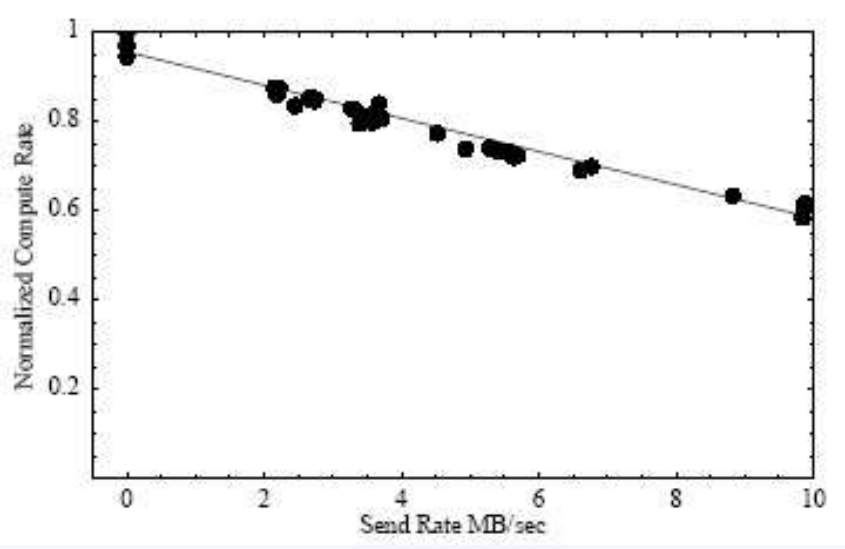
Un processeur calcule et envoie simultanément un message (débit variable).
(le débit est réglé par l'introduction de contention sur le destinataire)

Calcul et envoi simultanés (3)



Un processeur calcule et envoie simultanément un message : moyennes.

Calcul et envoi simultanés (3)

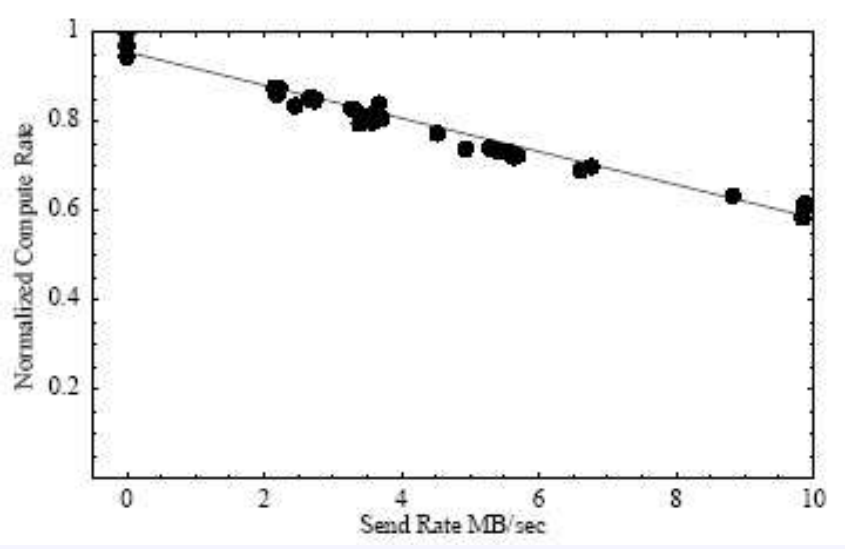


Un processeur calcule et envoie simultanément un message : moyennes.

Régression linéaire avec moindres carrés :

Compute rate $\approx$ -0.037 $\times$ Communication rate + 0.96.

Calcul et envoi simultanés (3)



Un processeur calcule et envoie simultanément un message : moyennes.

Régression linéaire avec moindres carrés :

Compute rate $\approx$ -0.037 $\times$ Communication rate + 0.96.

IR = 0.037 est le taux d'interférence (*Interference rate*)

Influence des réceptions sur les calculs

- Taux d'interférence moyen : 0.052.
  En moyenne quand le nœud reçoit des messages à 10 Mo/s, sa puissance de calcul diminue de 52%.

Influence des réceptions sur les calculs

- Taux d'interférence moyen : 0.052.
  En moyenne quand le nœud reçoit des messages à 10 Mo/s, sa puissance de calcul diminue de 52%.
- Quand une seule réception a lieu, la bande passante maximale atteinte est d'environ 85%-95% de la bande passante maximale atteignable (que l'on peut atteindre si l'on reçoit de plusieurs émetteurs simultanément).

Influence des réceptions sur les calculs

- Taux d'interférence moyen : 0.052.
  En moyenne quand le nœud reçoit des messages à 10 Mo/s, sa puissance de calcul diminue de 52%.
- Quand une seule réception a lieu, la bande passante maximale atteinte est d'environ 85%-95% de la bande passante maximale atteignable (que l'on peut atteindre si l'on reçoit de plusieurs émetteurs simultanément).
- Une fois le débit maximal atteint, l'impact sur la puissance de calcul disponible est maximisé, quel que soit le nombre de threads récepteurs.

Influence des réceptions sur les calculs

- Taux d'interférence moyen : 0.052.
  En moyenne quand le nœud reçoit des messages à 10 Mo/s, sa puissance de calcul diminue de 52%.
- Quand une seule réception a lieu, la bande passante maximale atteinte est d'environ 85%-95% de la bande passante maximale atteignable (que l'on peut atteindre si l'on reçoit de plusieurs émetteurs simultanément).
- Une fois le débit maximal atteint, l'impact sur la puissance de calcul disponible est maximisé, quel que soit le nombre de threads récepteurs.
- Estimation de l'impact d'un ensemble de réceptions :
  $1 - \sum_i \mathrm{IR}(i) \times \mathrm{TR}(i)$ où TR est le taux de transfert

Influence des envois sur les calculs

- Taux d'interférence moyen : 0.034.
  En moyenne quand le nœud envoie des messages à 10 Mo/s, sa puissance de calcul diminue de 34%.

Influence des envois sur les calculs

- Taux d'interférence moyen : 0.034.
  En moyenne quand le nœud envoie des messages à 10 Mo/s, sa puissance de calcul diminue de 34%.

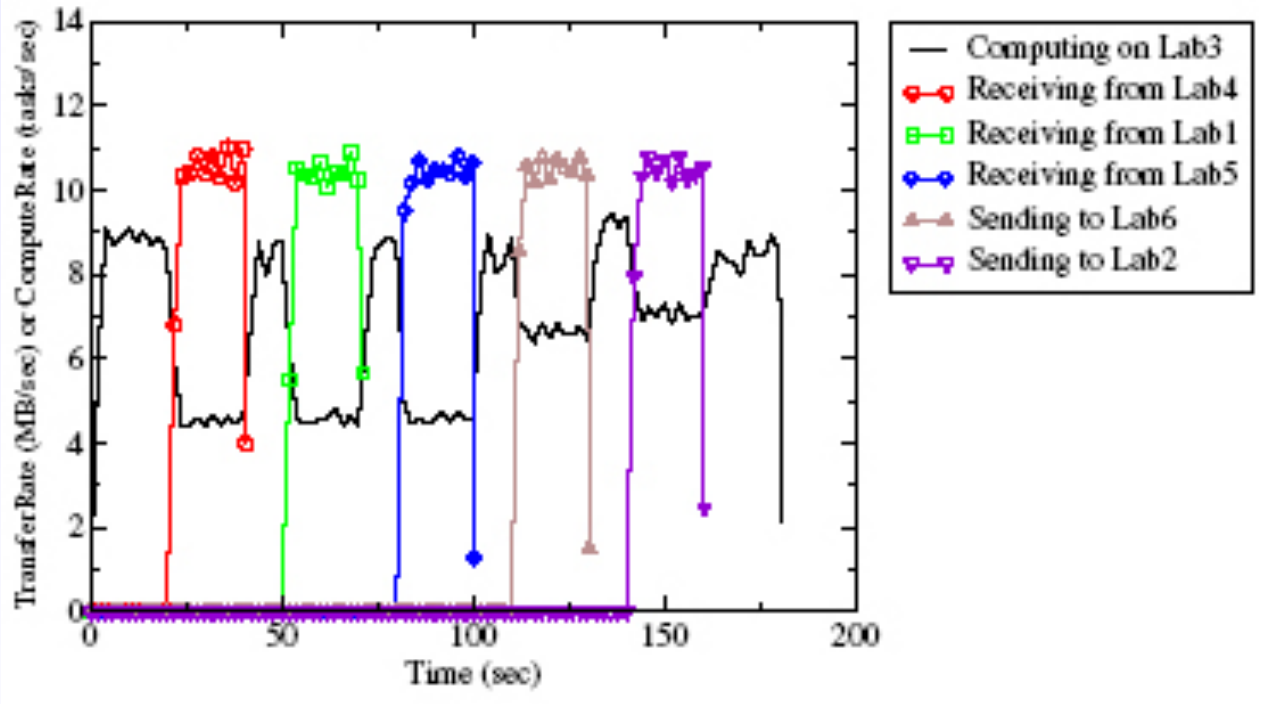
- Les réceptions influencent plus les calculs que les envois de message.

Influence des envois sur les calculs

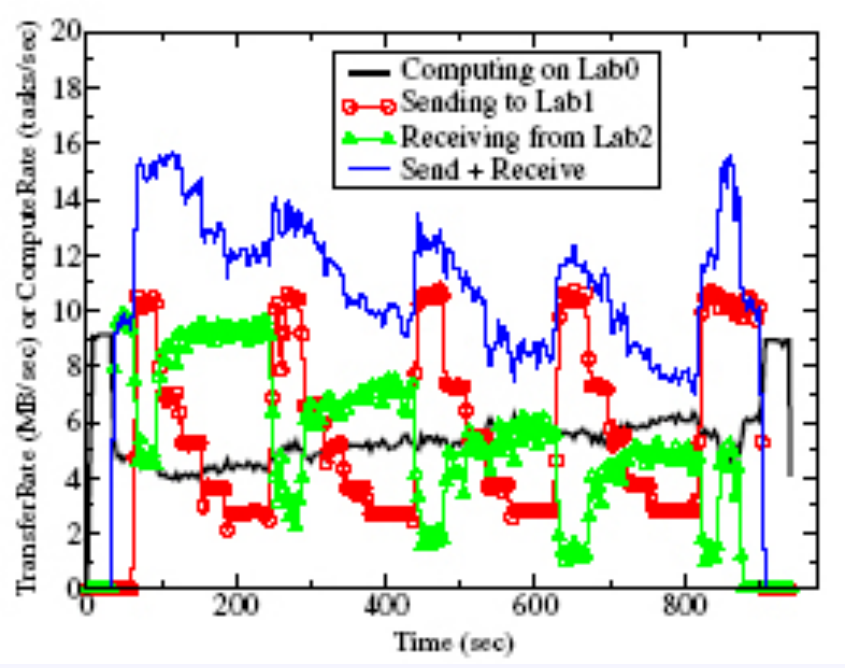
- Taux d'interférence moyen : 0.034.
  En moyenne quand le nœud envoie des messages à 10 Mo/s, sa puissance de calcul diminue de 34%.
- Les réceptions influencent plus les calculs que les envois de message.
- Estimation de l'impact d'un ensemble d'envois :

$$1 - \sum_i \mathrm{IR}(i) \times \mathrm{TR}(i) \qquad \text{où TR est le taux de transfert}$$

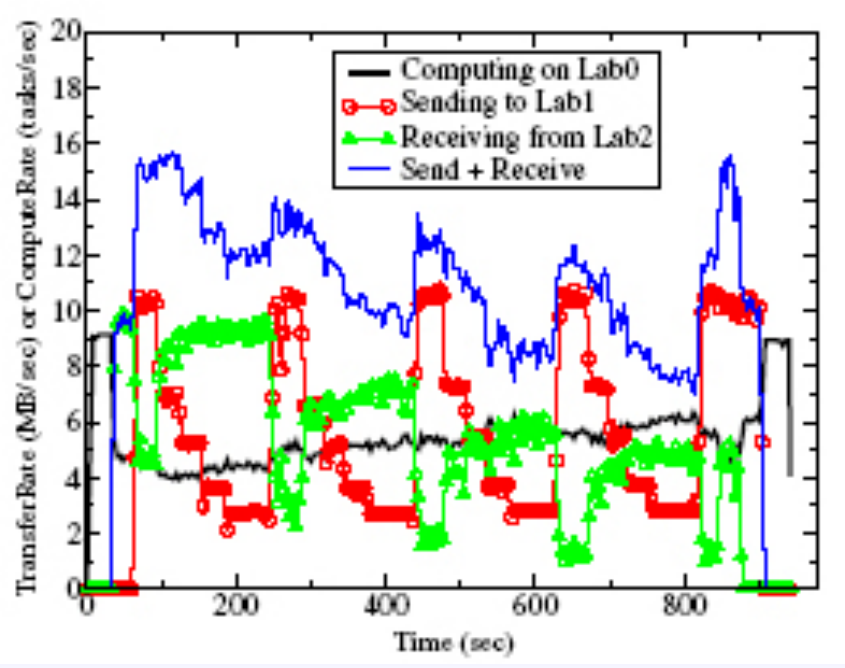
Les réceptions ont plus d'influence que les envois



Envois et réceptions simultanés



Envois et réceptions simultanés



L'estimation : $1 - \mathrm{IR}_s \times \mathrm{TR}_s - \mathrm{IR}_r \times \mathrm{TR}_r$

donne (s désignant les envois et r les réceptions) :

$\mathrm{IR}_r = 0.0427$ au lieu de 0.0502, $\mathrm{IR}_s = 0.0265$ au lieu de 0.0327

Il y a synergie entre les envois et les réceptions.

Autres observations

- Moins l'application utilise de mémoire moins les taux d'interférence semblent importants : les taux d'interférence dépendent des applications !
- La taille des messages n'a pas d'influence.
- Les processeurs distants ont des taux d'interférence nettement plus élevés (plus de 0.070 en réception), mais les bandes passantes maximales atteignables sont nettement plus faibles.

Conclusion

- Calculer et communiquer simultanément peut faire « perdre » plus de la moitié de la puissance de calcul.
- Une fois mesurés les taux d'interférence, pour une application donnée, entre chaque paire de processeurs, on peut en déduire une bonne approximation des puissances de calcul disponibles.
- Quid des autres plates-formes ? languages ? (même chose pour le C)

Conclusion

- Calculer et communiquer simultanément peut faire « perdre » plus de la moitié de la puissance de calcul.
- Une fois mesurés les taux d'interférence, pour une application donnée, entre chaque paire de processeurs, on peut en déduire une bonne approximation des puissances de calcul disponibles.
- Quid des autres plates-formes ? languages ? (même chose pour le C)

Est-ce que ça change quelque chose dans la pratique ?

Problème considéré

- Application n tâches identiques, indépendantes.

Problème considéré

- Application n tâches identiques, indépendantes.
- Plate-forme considérée : un arbre hétérogène de ressources de calcul (processeurs, clusters, etc.) reliées par des liens de communications de caractéristiques différentes.

Problème considéré

- Application n tâches identiques, indépendantes.
- Plate-forme considérée : un arbre hétérogène de ressources de calcul (processeurs, clusters, etc.) reliées par des liens de communications de caractéristiques différentes.
- Tous les fichiers de données sont initialement à la racine de l'arbre.

Problème considéré

- Application n tâches identiques, indépendantes.
- Plate-forme considérée : un arbre hétérogène de ressources de calcul (processeurs, clusters, etc.) reliées par des liens de communications de caractéristiques différentes.
- Tous les fichiers de données sont initialement à la racine de l'arbre.
- Le processeur racine décide quelles tâches il exécute lui-même et quelles tâches il délègue à ses fils. Chaque nœud interne procède de même.

Problème considéré

- Application n tâches identiques, indépendantes.
- Plate-forme considérée : un arbre hétérogène de ressources de calcul (processeurs, clusters, etc.) reliées par des liens de communications de caractéristiques différentes.
- Tous les fichiers de données sont initialement à la racine de l'arbre.
- Le processeur racine décide quelles tâches il exécute lui-même et quelles tâches il délègue à ses fils. Chaque nœud interne procède de même.
- Un nœud n'envoie de travail à un fils que lorsque celui-ci en demande. En cas de demandes simultanées, une *politique d'ordonnancement* résout les conflits.

Résultats connus

- Hypothèse : 1) pas d'interférence entre calculs et communications ou 2) communications et calculs sont mutuellement exclusifs.
- Solution *bandwidth centric* : s'il y a suffisamment de bande passante disponible, tous les fils travaillent ; sinon, les tâches sont envoyées aux fils qui ont des communications suffisamment rapides, la priorité étant donnée au fils avec lequel les communications sont les plus rapides.

Le modèle d'interférences à instancier

- Temps de calcul normalisé :

$$1 - \sum_i \mathsf{IR}_s(i) \times \mathsf{TR}_s(i) - \sum_i \mathsf{IR}_r(i) \times \mathsf{TR}_r(i)$$

- Obtenir une bonne approximation des différents taux d'interférence est compliqué et coûteux : mesure de la capacité de calcul pour différents débits d'émission et de réception.

Un modèle d'interférences instancié plus simplement (1)

Pour chaque nœud n,

- mesurer C^n : nombre de tâches exécutées par unité de temps ;

Un modèle d'interférences instancié plus simplement (1)

Pour chaque nœud n,

- mesurer C^n : nombre de tâches exécutées par unité de temps ;
- mesurer la bande passante maximale en réception MR^n et la puissance de calcul correspondante : C_r^n.

$$\mathrm{IR}_r = \frac{\frac{C^n - C_r^n}{C^n}}{\mathrm{MR}^n}$$

Un modèle d'interférences instancié plus simplement (1)

Pour chaque nœud n,

- mesurer C^n : nombre de tâches exécutées par unité de temps ;
- mesurer la bande passante maximale en réception MR^n et la puissance de calcul correspondante : C_r^n.

$$\mathrm{IR}_r = \frac{\frac{C^n - C_r^n}{C^n}}{\mathrm{MR}^n}$$

- Pour chaque fils i mesurer simultanément :

Un modèle d'interférences instancié plus simplement (1)

Pour chaque nœud n,

- mesurer C^n : nombre de tâches exécutées par unité de temps ;
- mesurer la bande passante maximale en réception MR^n et la puissance de calcul correspondante : C_r^n.

$$\mathrm{IR}_r = \frac{\frac{C^n - C_r^n}{C^n}}{\mathrm{MR}^n}$$

- Pour chaque fils i mesurer simultanément :
  1. le débit de n en émission au fils i : $\mathrm{SR}(i)$;

Un modèle d'interférences instancié plus simplement (1)

Pour chaque nœud n,

- mesurer C^n : nombre de tâches exécutées par unité de temps ;
- mesurer la bande passante maximale en réception MR^n et la puissance de calcul correspondante : C_r^n.

$$\mathrm{IR}_r = \frac{\frac{C^n - C_r^n}{C^n}}{\mathrm{MR}^n}$$

- Pour chaque fils i mesurer simultanément :
  1. le débit de n en émission au fils i : $\mathrm{SR}(i)$;
  2. la puissance de calcul du nœud n : $C_{sr}^n(i)$;

Un modèle d'interférences instancié plus simplement (1)

Pour chaque nœud n,

- mesurer C^n : nombre de tâches exécutées par unité de temps ;
- mesurer la bande passante maximale en réception MR^n et la puissance de calcul correspondante : C_r^n.

$$\mathrm{IR}_r = \frac{\frac{C^n - C_r^n}{C^n}}{\mathrm{MR}^n}$$

- Pour chaque fils i mesurer simultanément :
  1. le débit de n en émission au fils i : $\mathrm{SR}(i)$;
  2. la puissance de calcul du nœud n : $C_{sr}^n(i)$;
  3. le débit de n en réception de son père : $\mathrm{RR}(i)$.

Un modèle d'interférences instancié plus simplement (1)

Pour chaque nœud n,

- mesurer C^n : nombre de tâches exécutées par unité de temps ;
- mesurer la bande passante maximale en réception MR^n et la puissance de calcul correspondante : C_r^n.

$$\mathrm{IR}_r = \frac{\frac{C^n - C_r^n}{C^n}}{\mathrm{MR}^n}$$

- Pour chaque fils i mesurer simultanément :
  1. le débit de n en émission au fils i : $\mathrm{SR}(i)$;
  2. la puissance de calcul du nœud n : $C_{sr}^n(i)$;
  3. le débit de n en réception de son père : $\mathrm{RR}(i)$.

$$\mathrm{IR}_s(i) = \frac{\left(1 - \frac{\mathrm{RR}(i)}{\mathrm{MR}^n}\frac{C^n - C_r^n}{C^n} - \frac{C_{sr}^n(i)}{C^n}\right)}{\mathrm{SR}(i)}$$

Un modèle d'interférences instancié plus simplement (2)

- Une mesure pour le nœud n et une pour chacun de ses fils, au lieu d'un nombre potentiellement exponentiel de mesures avec un ensemble de valeurs de bandes passantes.
- Moins bonne précision globale, mais meilleure précision locale (à l'intérieur de l'enveloppe convexe des points mesurés).

Utilisation du modèle d'interférence

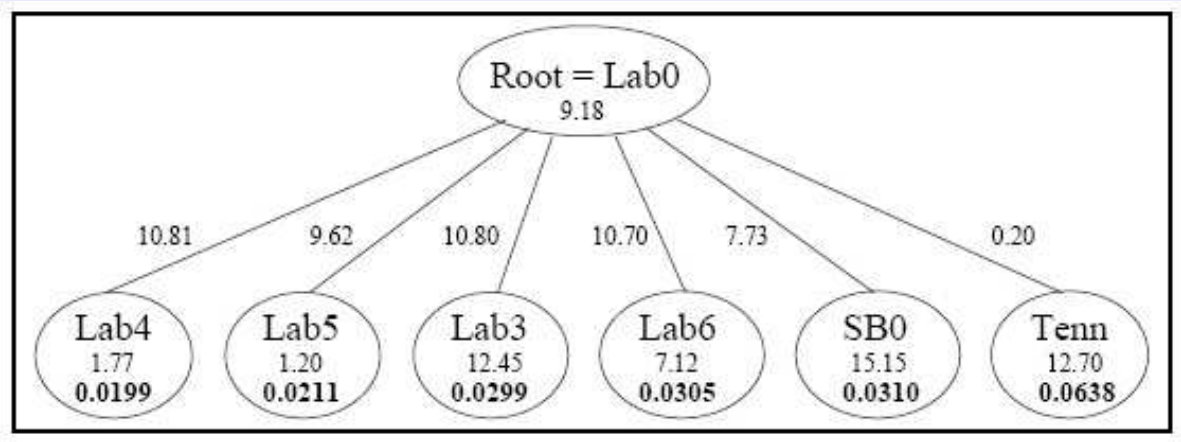
- Envois multi-ports : le débit de chaque nœud est maximisé quand la priorité est donnée aux fils de plus petit taux d'interférence.
  Ne pas donner une tâche à un fils pour lequel $\mathrm{IR}_s^n(i) ZC^n \geq 1$, où Z est la taille d'une tâche (envoyer une tâche à un fils coûte plus cher en temps de calcul que ça ne rapporte).
- Envois un-port : le débit de chaque nœud est maximisé quand la priorité est donnée aux fils de plus grande valeur pour : $B_r^i(1 - \mathrm{IR}_s^n ZC^n)$.

Vérification expérimentale

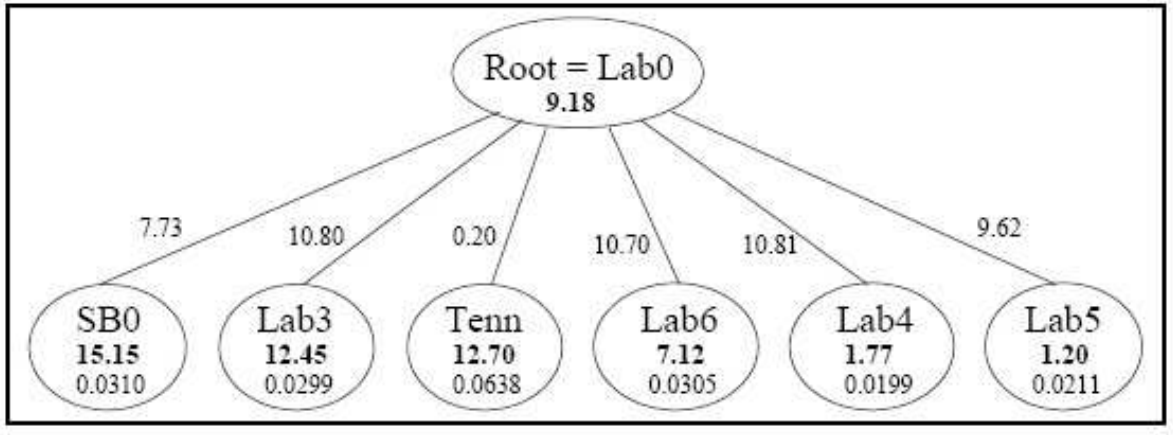
Politiques d'ordonnancement :

- IA :single : *interference aware, single port*, par valeurs de IR_s croissantes.
- IA :multi : *interference aware, multiple ports*, par valeurs de IR_s croissantes.
- FCFS : premier arrivé, premier servi.
- CompRate : par puissance de calcul décroissante.
- BWC : par bande passante décroissante.
- RootComputes : la racine fait tout le boulot.

Influence des politiques sur l'ordre (1)

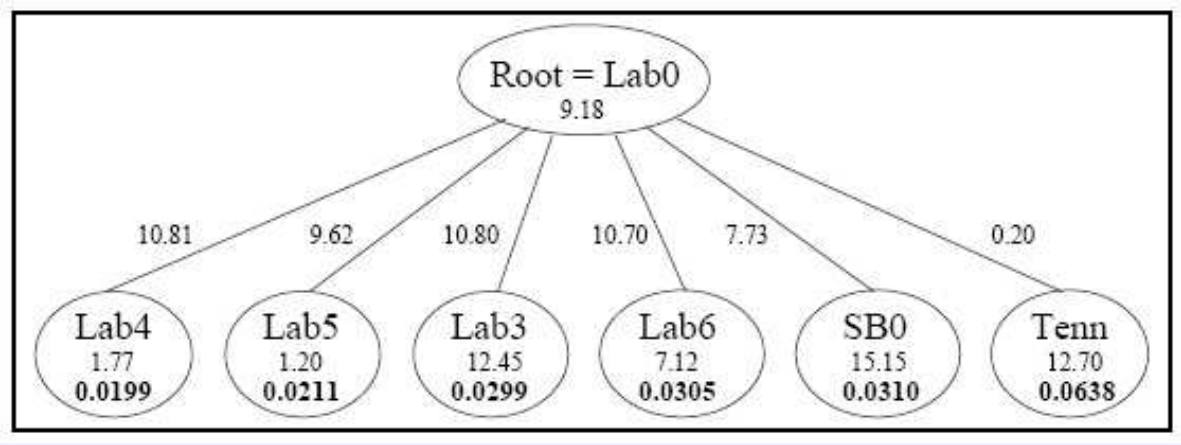


Ordre : taux d'interférences.

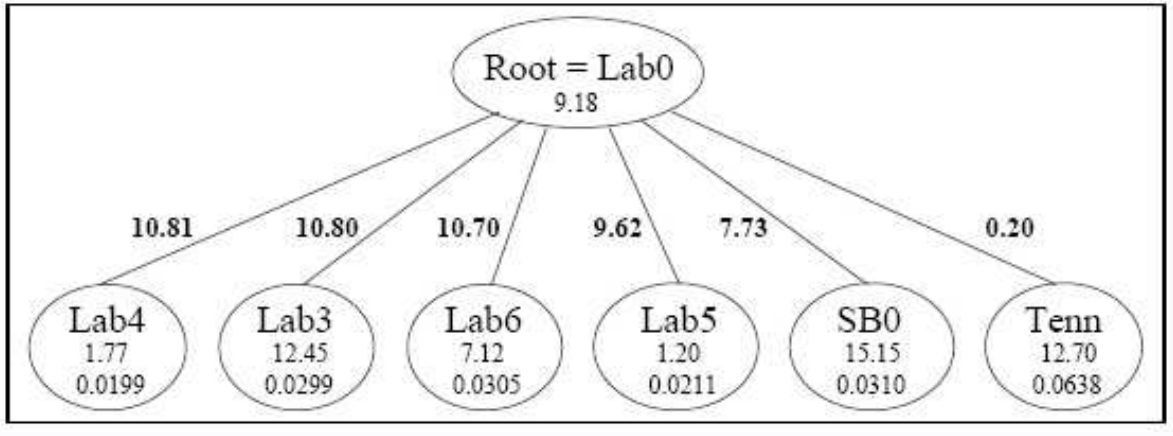


Ordre : puissance de calcul.

Influence des politiques sur l'ordre (2)

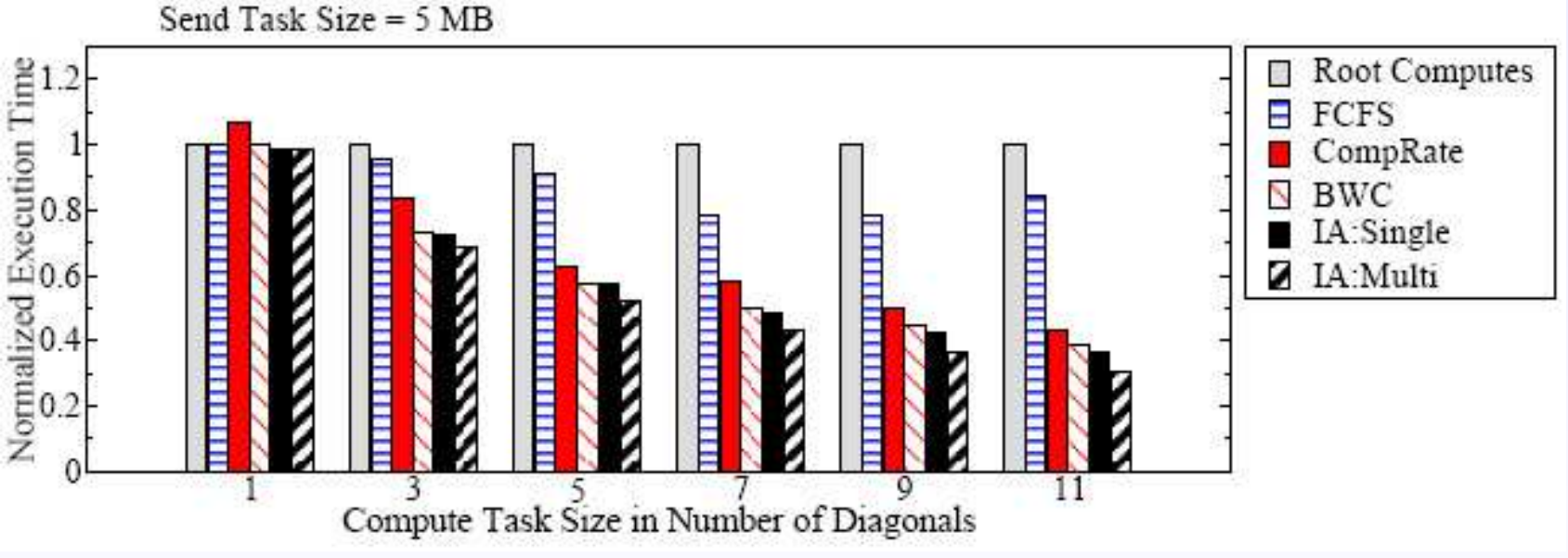


Ordre : taux d'interférences.



Ordre : bande passante.

Résultats


Send Task Size = 5 MB
Normalized Execution Time
0
0.2
0.4
0.6
0.8
1
1.2
1
3
5
7
9
11
Compute Task Size in Number of Diagonals
Root Computes
FCFS
CompRate
BWC
IA:Single
IA:Multi

Plan du cours

Problématique

Comment peut-on modéliser un réseau pour prédire les temps d'acheminement des messages ?

Latences des réseaux (1)

- Modélisation classique des temps de communications : $\alpha + \frac{x}{\beta}$

  où

  - x est la taille du message ;
  - α est la latence ;
  - β est la bande passante.

Latences des réseaux (1)

- Modélisation classique des temps de communications : $\alpha + \frac{x}{\beta}$

  où

  - x est la taille du message ;
  - α est la latence ;
  - β est la bande passante.
- De nombreux travaux négligent α.
  Justification : faciliter ou rendre possible la résolution.

Latences des réseaux (1)

- Modélisation classique des temps de communications : $\alpha + \frac{x}{\beta}$

  où

  - x est la taille du message ;
  - α est la latence ;
  - β est la bande passante.
- De nombreux travaux négligent α.
  Justification : faciliter ou rendre possible la résolution.
- Problème : la solution obtenue peut-être irréaliste car l'envoi d'un message infiniment petit n'est pas pénalisé.
  (Ex. : distribution en un nombre infini de tournées infiniment petites.)

Latences des réseaux (2)

Les latences sont-elles vraiment négligeables ?

Un exemple : TeraGrid entre le SDSC et le NCSA (USA)

- Latence : $\approx$ 100ms
- Bande passante : 40 Gb/s.
- Envoie d'un message de 1 G octet : plus du tiers du temps d'envoi est dû à la latence.

Conclusion ?

Latences des calculs

- Les programmes peuvent avoir un temps d'initialisation non négligeable.
- Le lancement de processus peut être long : sur Globus Toolkit 2.0, le lancement d'une tâche sans travail (*no-op job*) peut prendre 25 secondes (authentification, acquisition des ressources, création du processus, etc.).

Il peut être nécessaire de prendre en compte des latences de calcul même pour des programmes dont le temps *d'exécution* est linéaire en la taille des données.

Modélisation classique des réseaux

Modélisation classique

- Les processeurs sont reliés par des connexions point-à-point (pas de routeurs, de switch, etc.).
- Modèle multi-port : un processeur peut envoyer simultanément des messages à plusieurs autres processeurs.
- Modèle un-port : un processeur peut, à un instant donné, envoyer au plus un message à un autre processeur.
- Un seul message peut circuler sur un lien donné à un instant donné.
- Un graphe complet peut modéliser un switch, mais pas un réseau Ethernet.

1er problème de la modélisation classique des réseaux

Des liens de communications logiques différents peuvent partager des liens physiques.

Communications simultanées de A à B et de C à D. Logiquement : pas d'interférences, en pratique il faut connaître la topologie pour pouvoir décider

Mauvaise prédiction des contentions.

2^{e} problème de la modélisation classique des réseaux

Il peut être avantageux de partager l'utilisation des liens réseaux.
Exemple (caricatural)

- Un serveur reçoit des tâches et doit les répartir sur des processeurs de même puissance, en utilisant un même lien réseau.

2e problème de la modélisation classique des réseaux

Il peut être avantageux de partager l'utilisation des liens réseaux.
Exemple (caricatural)

- Un serveur reçoit des tâches et doit les répartir sur des processeurs de même puissance, en utilisant un même lien réseau.
- Au temps t arrive une tâche nécessitant 10 min. de communications et 10 min. de calcul.

2e problème de la modélisation classique des réseaux

Il peut être avantageux de partager l'utilisation des liens réseaux.
Exemple (caricatural)

- Un serveur reçoit des tâches et doit les répartir sur des processeurs de même puissance, en utilisant un même lien réseau.
- Au temps t arrive une tâche nécessitant 10 min. de communications et 10 min. de calcul.
- Au temps $t+5$ arrive une tâche nécessitant 1 min. de communications et 19 min. de calcul.

2e problème de la modélisation classique des réseaux

Il peut être avantageux de partager l'utilisation des liens réseaux.
Exemple (caricatural)

- Un serveur reçoit des tâches et doit les répartir sur des processeurs de même puissance, en utilisant un même lien réseau.
- Au temps t arrive une tâche nécessitant 10 min. de communications et 10 min. de calcul.
- Au temps $t+5$ arrive une tâche nécessitant 1 min. de communications et 19 min. de calcul.
- Objectif *stretch* moyen (ratio entre le temps passé par la tâche dans le système et le temps qu'elle aurait passé s'il n'y avait pas eu d'autre tâches)

2e problème de la modélisation classique des réseaux

Il peut être avantageux de partager l'utilisation des liens réseaux.
Exemple (caricatural)

- Un serveur reçoit des tâches et doit les répartir sur des processeurs de même puissance, en utilisant un même lien réseau.
- Au temps t arrive une tâche nécessitant 10 min. de communications et 10 min. de calcul.
- Au temps $t+5$ arrive une tâche nécessitant 1 min. de communications et 19 min. de calcul.
- Objectif *stretch* moyen (ratio entre le temps passé par la tâche dans le système et le temps qu'elle aurait passé s'il n'y avait pas eu d'autre tâches)
- Politique FIFO : $\frac{1}{2}(\frac{20}{20} + \frac{30}{20})) = 1,25$.

2e problème de la modélisation classique des réseaux

Il peut être avantageux de partager l'utilisation des liens réseaux.
Exemple (caricatural)

- Un serveur reçoit des tâches et doit les répartir sur des processeurs de même puissance, en utilisant un même lien réseau.
- Au temps t arrive une tâche nécessitant 10 min. de communications et 10 min. de calcul.
- Au temps $t+5$ arrive une tâche nécessitant 1 min. de communications et 19 min. de calcul.
- Objectif *stretch* moyen (ratio entre le temps passé par la tâche dans le système et le temps qu'elle aurait passé s'il n'y avait pas eu d'autre tâches)
- Politique FIFO : $\frac{1}{2}(\frac{20}{20} + \frac{30}{20})) = 1,25$.
- 2e tâche en premier : $\frac{1}{2}(\frac{26}{20} + \frac{20}{20}) = 1,15$.

2e problème de la modélisation classique des réseaux

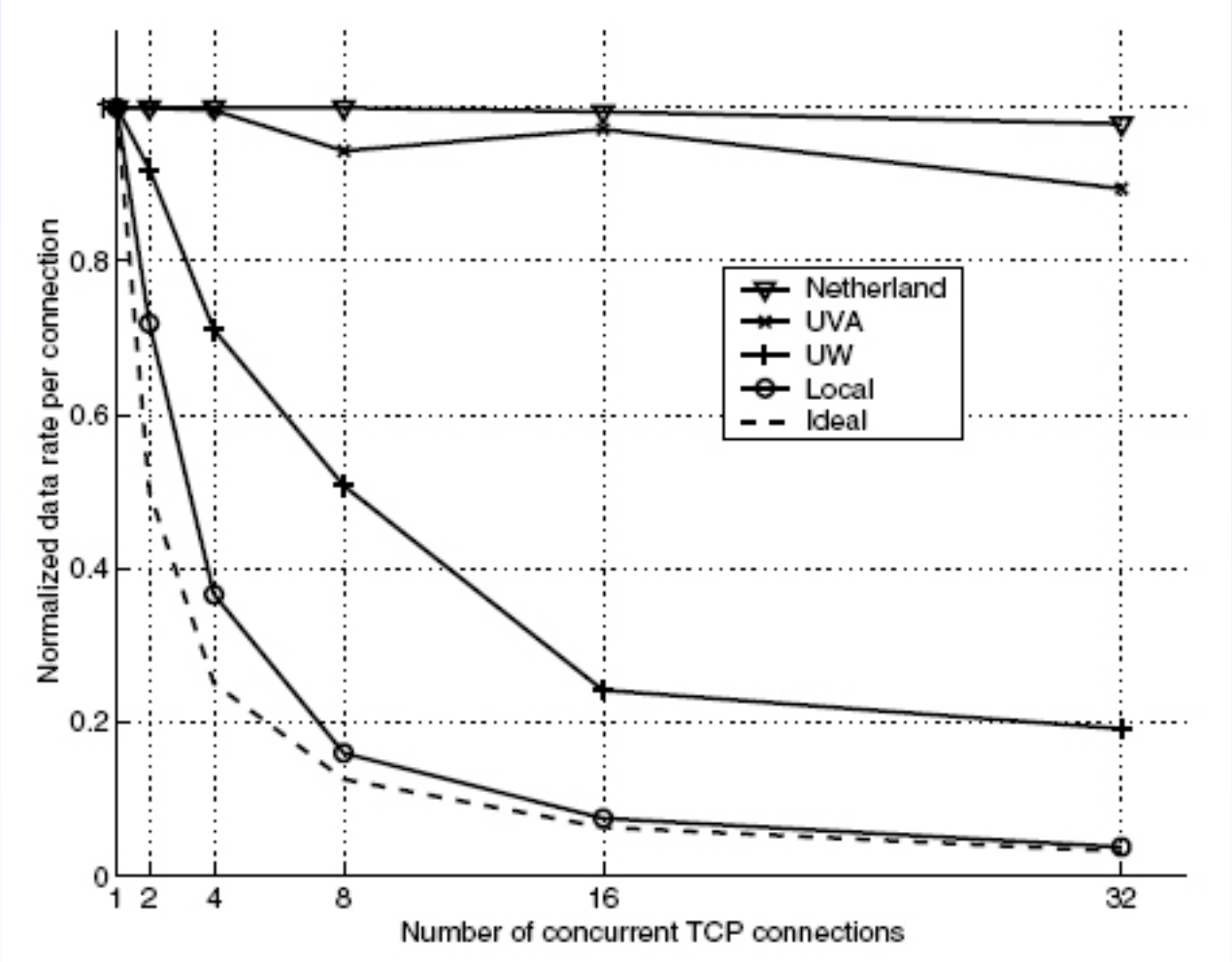
Il peut être avantageux de partager l'utilisation des liens réseaux.
Exemple (caricatural)

- Un serveur reçoit des tâches et doit les répartir sur des processeurs de même puissance, en utilisant un même lien réseau.
- Au temps t arrive une tâche nécessitant 10 min. de communications et 10 min. de calcul.
- Au temps $t+5$ arrive une tâche nécessitant 1 min. de communications et 19 min. de calcul.
- Objectif *stretch* moyen (ratio entre le temps passé par la tâche dans le système et le temps qu'elle aurait passé s'il n'y avait pas eu d'autre tâches)
- Politique FIFO : $\frac{1}{2}(\frac{20}{20} + \frac{30}{20})) = 1,25.$
- 2e tâche en premier : $\frac{1}{2}(\frac{26}{20} + \frac{20}{20}) = 1,15.$
- Partage équitable des bandes passantes : $\frac{1}{2}(\frac{21}{20} + \frac{21}{20}) = 1,05.$

Partage de la bande passante : modélisation classique

- Nous avons supposé que quand x communications partageaient un même lien de communication de bande passante B, chacune recevait une bande passante de $\frac{B}{x}$.
- C'est une modélisation classique.
- C'est (généralement) vrai sur un réseau local (LAN).

Partage de la bande passante : en pratique (1)



Partage de la bande passante : en pratique (2)

- La modélisation classique est invalide pour les réseaux longue distance (WAN).
- À très longue distance, toutes les connexions reçoivent la même bande passante que la première connexion ouverte !

  Par un *backbone* passent un très très grand nombre de communications : une de plus ou de moins ne change pas significativement la bande passante allouée à chacune des communications.

  Quelque soit la bande passante allouée sur un *backbone*, la portion que l'émetteur peut effectivement utiliser est limitée par sa fenêtre de congestion TCP.
- Généralisation du modèle classique : $\frac{B}{\alpha + x\beta}$.

Modélisation du réseau

- Les machines ne sont pas reliées par des *backbones* : la communication utilise un certain nombre de liens locaux avant d'atteindre le ou les *backbones* qui sont utilisés dans la communication longue distance.

Modélisation du réseau

- Les machines ne sont pas reliées par des *backbones* : la communication utilise un certain nombre de liens locaux avant d'atteindre le ou les *backbones* qui sont utilisés dans la communication longue distance.
- La bande passante pour une communication utilisant plusieurs liens est définie par le lien le plus « lent ».

Modélisation du réseau

- Les machines ne sont pas reliées par des *backbones* : la communication utilise un certain nombre de liens locaux avant d'atteindre le ou les *backbones* qui sont utilisés dans la communication longue distance.
- La bande passante pour une communication utilisant plusieurs liens est définie par le lien le plus « lent ».
- Le facteur limitant peut être la carte réseau de la machine ou le réseau local.

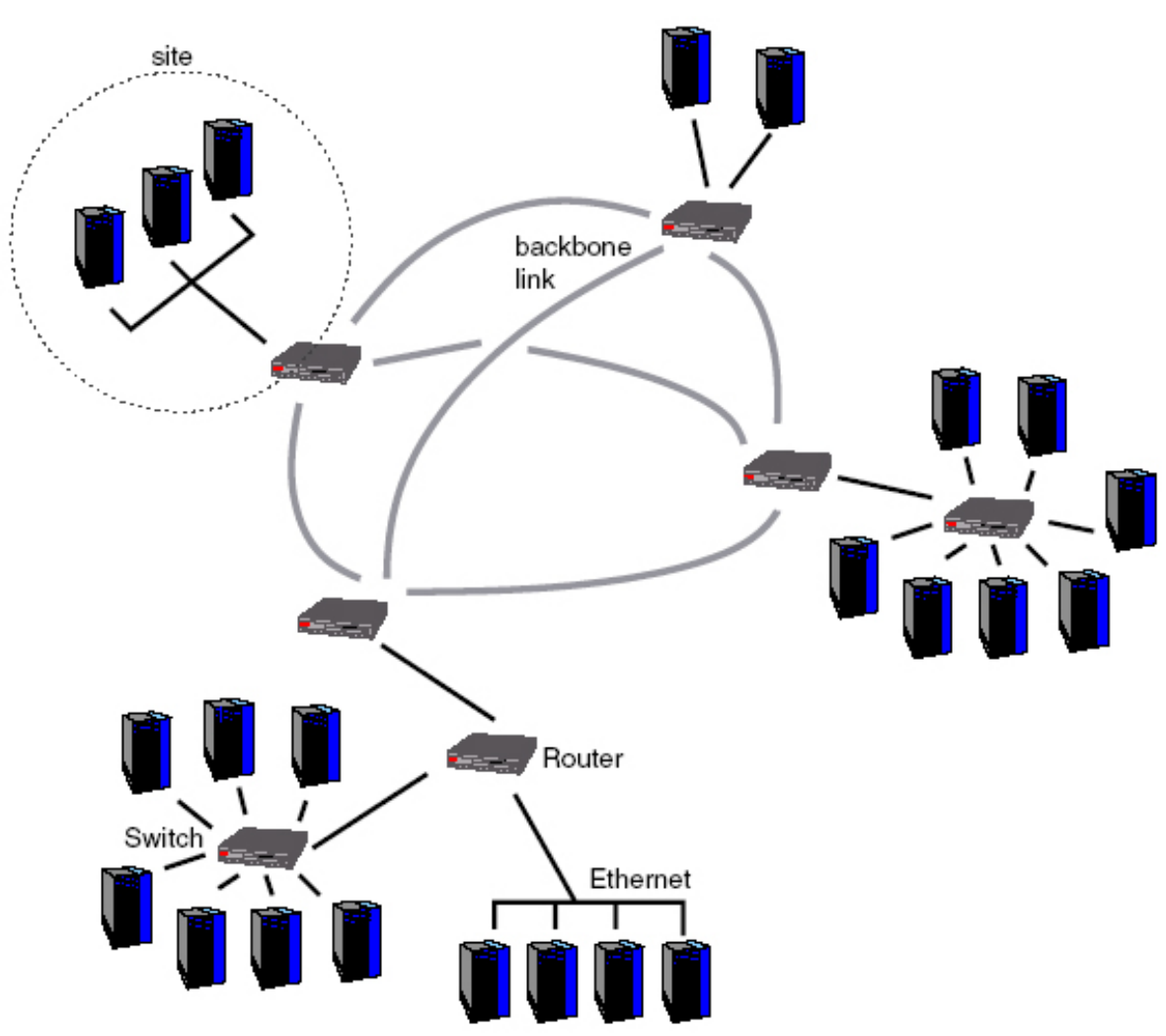
Modélisation du réseau

- Les machines ne sont pas reliées par des *backbones* : la communication utilise un certain nombre de liens locaux avant d'atteindre le ou les *backbones* qui sont utilisés dans la communication longue distance.
- La bande passante pour une communication utilisant plusieurs liens est définie par le lien le plus « lent ».
- Le facteur limitant peut être la carte réseau de la machine ou le réseau local.
- Il est nécessaire de considérer la topologie locale si plusieurs machines d'un même site risquent de communiquer simultanément.

Modélisation du réseau

- Les machines ne sont pas reliées par des *backbones* : la communication utilise un certain nombre de liens locaux avant d'atteindre le ou les *backbones* qui sont utilisés dans la communication longue distance.
- La bande passante pour une communication utilisant plusieurs liens est définie par le lien le plus « lent ».
- Le facteur limitant peut être la carte réseau de la machine ou le réseau local.
- Il est nécessaire de considérer la topologie locale si plusieurs machines d'un même site risquent de communiquer simultanément.
- Comportement de TCP : sur un lien congestionné, les différentes communications reçoivent des bandes passantes inversement proportionnelles à leur *round-trip-times*.

Modélisation du réseau



Plan du cours

Problématique

- Graphe $G = (S, A)$.
  L'arête $a \in A$ a une bande passante maximale $b(a)$.

- Ensemble $\mathcal{R}$ de chemins dans le graphe G.
  Notre problème est d'attribuer à chaque chemin $r \in \mathcal{R}$ une bande passante λ_r

- Respect des bandes passantes disponibles :

$$\forall a \in A, \sum_{r \in \mathcal{R}, a \in r} \lambda_r \leq b(a),$$

où $a \in r$ signifie que le chemin r emprunte l'arête a.

Problématique

- Graphe $G = (S, A)$.
  L'arête $a \in A$ a une bande passante maximale $b(a)$.

- Ensemble $\mathcal{R}$ de chemins dans le graphe G.
  Notre problème est d'attribuer à chaque chemin $r \in \mathcal{R}$ une bande passante λ_r

- Respect des bandes passantes disponibles :

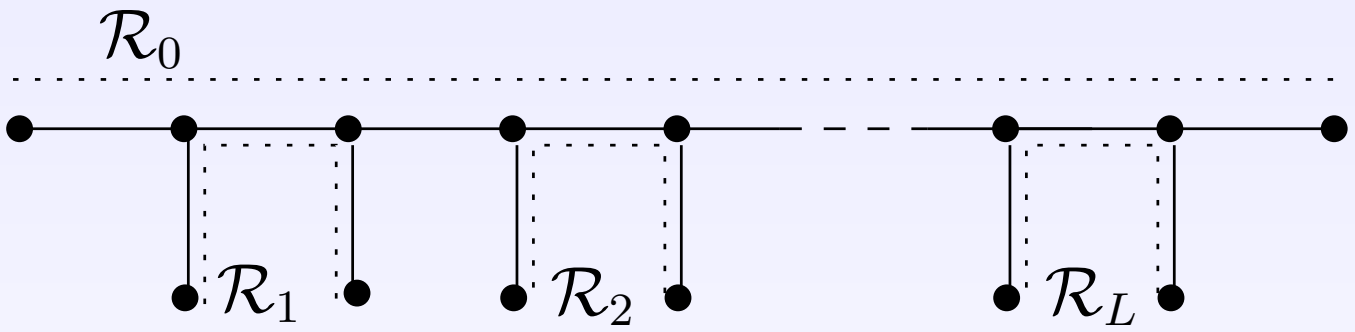
$$\forall a \in A, \sum_{r \in \mathcal{R}, a \in r} \lambda_r \leq b(a),$$

où $a \in r$ signifie que le chemin r emprunte l'arête a.

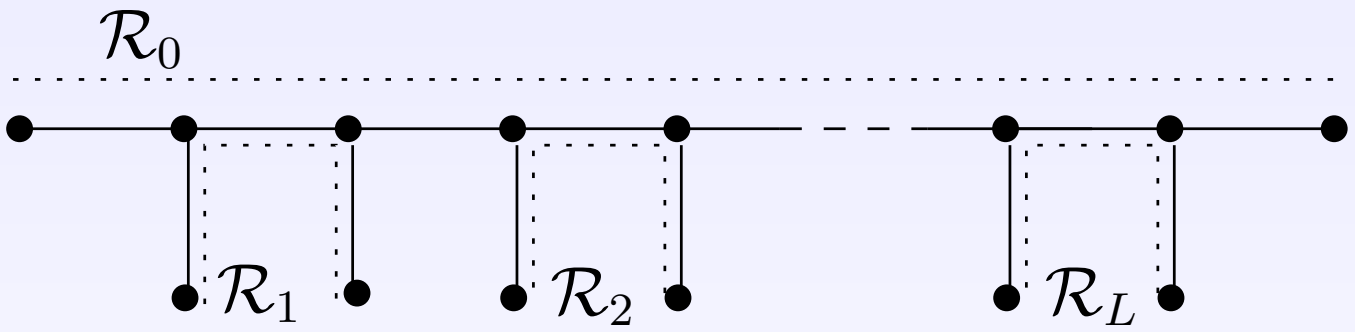
Comment allouer les bandes passantes ?

Maximisation du débit total

Maximisation du débit total



Maximisation du débit total



Dans une solution optimale : $\lambda_1 = \lambda_2 = ... = \lambda_L = 1 - \lambda_0$

Le débit total est égal à : $L - (L-1)\lambda_0$ et est maximal quand la route $\mathcal{R}_0$ se voit attribuer une bande passante nulle !

Équité *max-min* : définition et propriétés

- Définition : la plus petite bande passante allouée est maximale.

Équité *max-min* : définition et propriétés

- Définition : la plus petite bande passante allouée est maximale.
- Si aucune arête utilisé par le chemin $\mathcal{R}_i$ n'est saturée

$$\forall a \in \mathcal{R}_i, \sum_{r \in \mathcal{R}, a \in r} \lambda_r < b(a)$$

on peut strictement augmenter la bande passante allouée à $\mathcal{R}_i$.

Équité *max-min* : définition et propriétés

- Définition : la plus petite bande passante allouée est maximale.
- Si aucune arête utilisé par le chemin $\mathcal{R}_i$ n'est saturée

$$\forall a \in \mathcal{R}_i, \sum_{r \in \mathcal{R}, a \in r} \lambda_r < b(a)$$

on peut strictement augmenter la bande passante allouée à $\mathcal{R}_i$.

Donc, il existe un chemin r tel que $\lambda_r = \lambda_{\min}$ et tel qu'il existe au moins une arête $a \in r$ qui est saturée.

Équité *max-min* : définition et propriétés

- Définition : la plus petite bande passante allouée est maximale.
- Si aucune arête utilisé par le chemin $\mathcal{R}_i$ n'est saturée

$$\forall a \in \mathcal{R}_i, \sum_{r \in \mathcal{R}, a \in r} \lambda_r < b(a)$$

  on peut strictement augmenter la bande passante allouée à $\mathcal{R}_i$.

  Donc, il existe un chemin r tel que $\lambda_r = \lambda_{\min}$ et tel qu'il existe au moins une arête $a \in r$ qui est saturée.
- Soit une arête e de r saturée. S'il existe un chemin r', $e \in r'$, de bande passante $\lambda_{r'} > \lambda_r$ alors on peut donner un peu de la bande passante de r' à r tout en ayant $\lambda'_{r'} > \lambda'_r > \lambda_r$.

Équité *max-min* : définition et propriétés

- Définition : la plus petite bande passante allouée est maximale.
- Si aucune arête utilisé par le chemin $\mathcal{R}_i$ n'est saturée

$$\forall a \in \mathcal{R}_i, \sum_{r \in \mathcal{R}, a \in r} \lambda_r < b(a)$$

  on peut strictement augmenter la bande passante allouée à $\mathcal{R}_i$.

  Donc, il existe un chemin r tel que $\lambda_r = \lambda_{\min}$ et tel qu'il existe au moins une arête $a \in r$ qui est saturée.
- Soit une arête e de r saturée. S'il existe un chemin r', $e \in r'$, de bande passante $\lambda_{r'} > \lambda_r$ alors on peut donner un peu de la bande passante de r' à r tout en ayant $\lambda'_{r'} > \lambda'_r > \lambda_r$.

  Donc, il existe un chemin r tel que $\lambda_r = \lambda_{\min}$ et tel qu'il existe une arête $a \in r$ telle que : $\forall r', a \in r' \Rightarrow \lambda_{r'} = \lambda_{\min}$.

Équité *max-min* : allocation

- Bande passante minimale allouée :

$$\lambda_{\min} = \min_{a \in A} \frac{b(a)}{|\{r \in \mathcal{R} \mid a \in r\}|}.$$

Équité *max-min* : allocation

- Bande passante minimale allouée :

$$\lambda_{\min} = \min_{a \in A} \frac{b(a)}{|\{r \in \mathcal{R} \mid a \in r\}|}.$$

- La solution est-elle unique ?

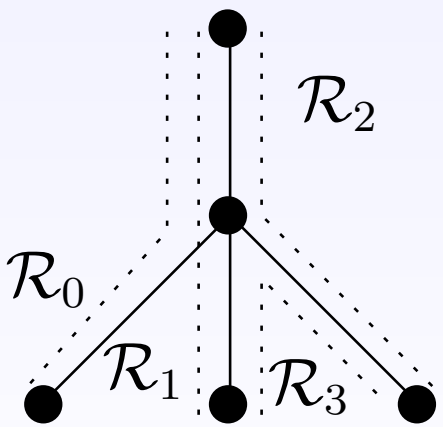
Équité *max-min* : allocation

- Bande passante minimale allouée :

$$\lambda_{\min} = \min_{a \in A} \frac{b(a)}{|\{r \in \mathcal{R} \mid a \in r\}|}.$$

- La solution est-elle unique ?

  Évidemment non :

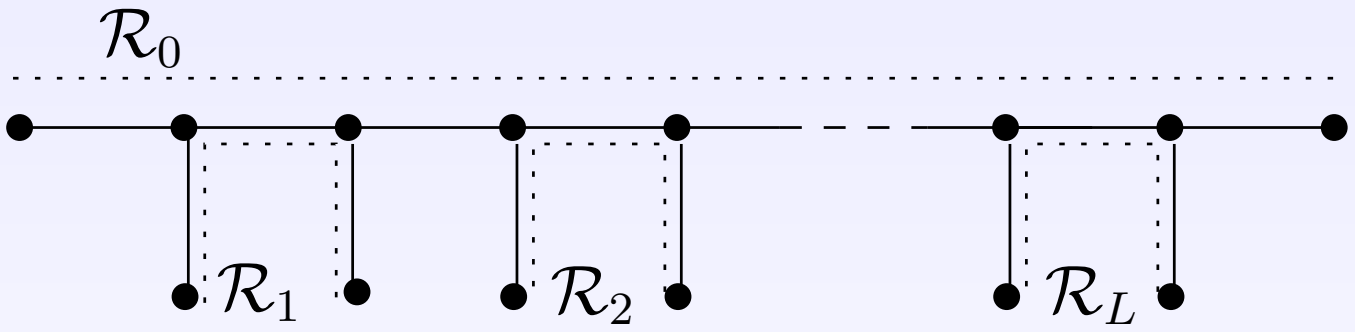


  Si toutes les arêtes sont supposées avoir une bande passante de 1, $\lambda_0 = \lambda_1 = \lambda_2 = \frac{1}{3}$ mais λ_3 peut prendre n'importe quelle valeur dans l'intervalle $[\frac{1}{3}; \frac{2}{3}]$.

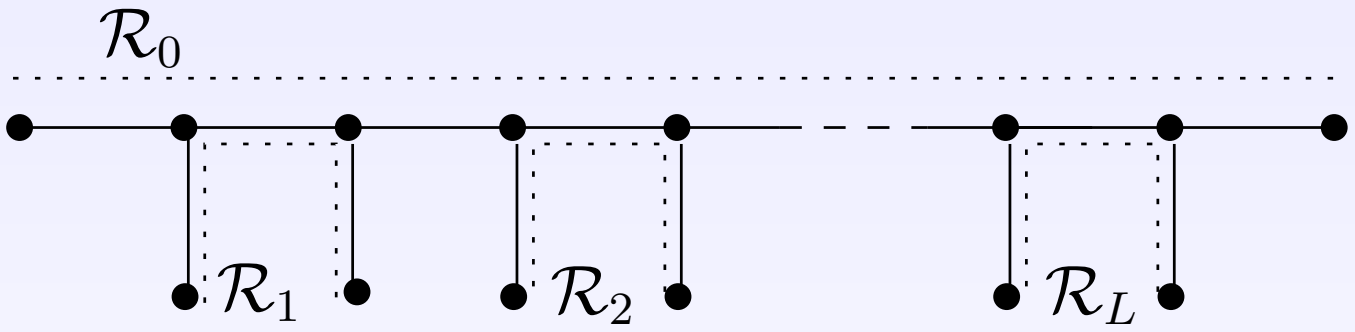
Équité max-min : algorithme

Algorithme : on applique récursivement le principe de l'équité max-min : on détermine les arêtes qui définissent la bande passante minimale allouée, on alloue la bande passante correspondante aux routes qui utilisent ces arêtes et on appelle récursivement l'algorithme sur les routes restantes avec les bandes passantes maximales disponibles mises à jour.

Retour sur l'exemple



Retour sur l'exemple



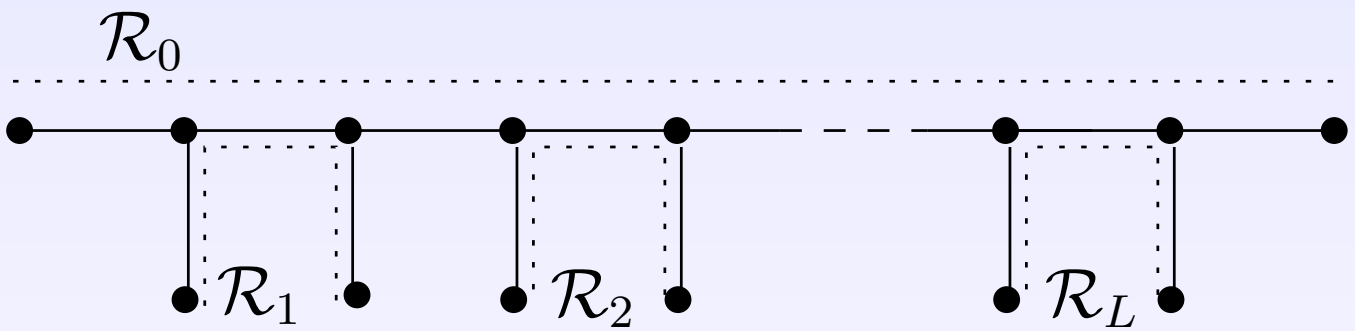
Dans une solution optimale : $\lambda_1 = \lambda_2 = ... = \lambda_L = \lambda_0 = \frac{1}{2}$

Le débit total est égal à : $\frac{L+1}{2}$ (ce qui est nettement moins que l'optimum L).

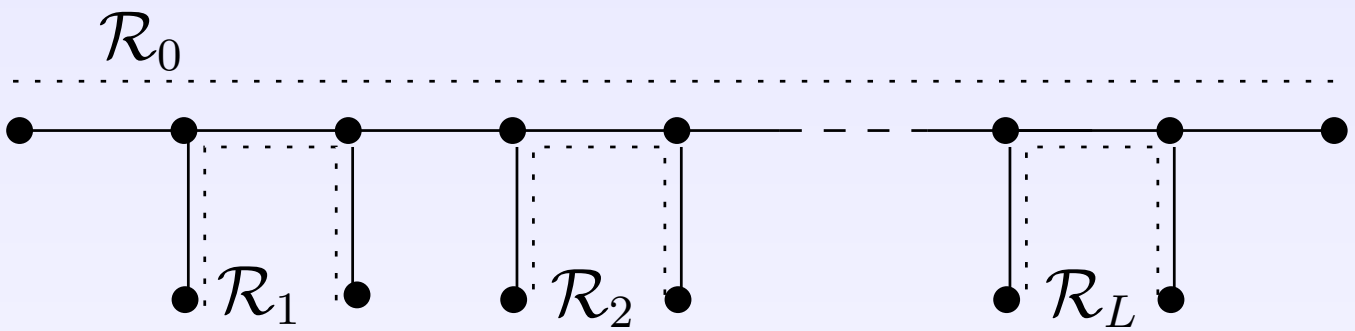
Équité proportionnelle

- Définition : on veut maximiser $\sum_{\mathcal{R}} \log \lambda_{\mathcal{R}}$.
- Comportement plus proche de TCP.

Retour sur l'exemple



Retour sur l'exemple



Dans une solution optimale : $\lambda_1 = \lambda_2 = ... = \lambda_L = 1 - \lambda_0$.
On veut maximiser : $L \log(1 - \lambda_0) + \log(\lambda_0)$.
On veut donc maximiser : $(1 - \lambda_0)^L \lambda_0$.
Le maximum est atteint pour $\lambda_0 = \frac{1}{L+1}$.

Le débit total est égal à : $L - \frac{L-1}{L+1}$ (ce qui est nettement plus proche de l'optimum L).
L'équité proportionnelle pénalise les routes longues au bénéfice du débit total.

Conclusion

- Une réalité très compliquée.
- Nécessité d'avoir une modélisation qui permette de prédire le comportement des applications.
- Nécessité de prendre en compte la topologie et plus généralement le comportement du réseau.
- Solution à adapter à l'environnement et à l'application visés.