

Mécanismes de coordination et routage dans les chemins, les arbres et les anneaux

Eric Angel, Evripidis Bampis, Fanny Pascual

IBISC, 523 place des terrasses de l'Agora, 91000 Evry, France
email : {angel, bampis, fpascual}@lami.univ-evry.fr

On s'intéresse à un problème de routage de paquets dans des réseaux de type chemin, arbre et anneau. Plus spécifiquement, on s'intéresse à la situation où n paquets de différentes tailles (ou longueurs) partent d'une même source et ont chacun une destination dans le réseau. On étudie la perte de performance due au fait d'avoir des paquets "individualistes" (chaque paquet veut arriver le plus tôt possible à sa destination) et un réseau complètement décentralisé (chaque lien ne connaît que les paquets qu'il doit router). Cette perte de performance est étudiée dans le cas de différentes politiques d'ordonnancement des liens (par exemple, parmi les paquets qui demandent à être routés sur un lien, le lien ordonnance en premier le plus petit paquet) pour deux problèmes : celui qui consiste à minimiser la date d'arrivée du dernier paquet, et celui qui consiste à minimiser la date d'arrivée moyenne des paquets.

Keywords: Prix de l'anarchie, mécanisme de coordination, routage, arbre, anneau

1 Introduction

On s'intéresse au routage de paquets (aussi appelés tâches) dans un réseau (modélisé par un graphe), dans lequel chaque paquet a une longueur quelconque et est représenté par un agent dont le but est de minimiser la date d'arrivée du paquet à sa destination. On suppose que n tâches partent au même moment d'une source (un sommet du graphe), et que chaque tâche a une destination quelconque. On considère que le réseau est de type "store and forward", c'est à dire qu'une seule tâche peut être routée en même temps sur un lien, et une tâche ne peut pas être routée sur plusieurs liens en même temps. On se place dans un contexte distribué : à un instant donné, chaque lien (i.e. chaque arc de notre graphe) connaît uniquement les tâches qui ont demandé à être routées sur ce lien. Chaque lien a une politique d'ordonnancement qui lui permet de donner un ordre aux différentes tâches qui veulent être routées au même moment sur ce lien : quand un lien n'est pas déjà en train de router une tâche, il en choisit une à router parmi les tâches en attente. La politique des liens est décidée à l'avance et ne doit pas dépendre des tâches à ordonnancer.

On étudie la performance de différentes politiques pour les deux problèmes suivants : le problème de la Date d'Arrivée Maximum, qui consiste à minimiser la date à laquelle toutes les tâches ont été routées (i.e. on veut minimiser la date d'arrivée de la dernière tâche), et le problème de la Date d'Arrivée Moyenne, qui consiste à minimiser la date d'arrivée moyenne des tâches (celle-ci étant égale à la somme des dates d'arrivée des tâches, divisée par le nombre de tâches).

Les politiques étudiées sont les suivantes :

- SPT : *Shortest Processing Time*. La tâche qui a la plus petite longueur (i.e. temps d'exécution) est ordonnancée en premier.
- LPT : *Largest Processing Time*. La tâche qui a la plus grande longueur est ordonnancée en premier.
- LRT : *Largest Remaining Time*. La tâche dont le temps de parcours restant est le plus grand est ordonnancée en premier. Le temps de parcours restant est égal à la longueur de la tâche multipliée par le nombre d'arcs que la tâche doit encore traverser pour arriver à sa destination.
- LRD : *Largest Remaining Distance*. La tâche qui a la plus grande distance restante est ordonnancée en premier. La distance restante d'une tâche est le nombre d'arcs qu'elle doit encore traverser pour arriver à sa destination.

Pour modéliser le fait que chaque paquet est autonome et cherche à minimiser sa date d'arrivée, sans se soucier des conséquences que cela pourrait avoir sur les dates d'arrivée des autres paquets, on représente chaque paquet comme un agent, qui connaît les caractéristiques des autres agents et la politique d'ordonnement des liens, et a plusieurs stratégies possibles : un paquet peut soit attendre (il ne demande pas à être routé), soit demander à être routé sur un lien donné (qu'il choisit). Avec les politiques que l'on étudie, une tâche ne diminuera jamais sa date d'arrivée en attendant : elle aura donc toujours intérêt à demander à être immédiatement routée. Ainsi, dans le cas du chemin et de l'arbre, comme il existe un seul chemin entre une source et une destination, chaque tâche demandera à être routée vers le prochain lien qui la mènera à sa destination; dans le cas de l'anneau une tâche devra choisir entre deux liens sur lequel elle veut être routée. On se trouve dans une situation stable (appelée équilibre de Nash) quand, connaissant les stratégies choisies par les agents, chaque agent n'a pas intérêt à changer de stratégie.

Afin de mesurer la perte d'efficacité due à l'absence de contrôle centralisé (c'est à dire à l'application de politiques locales sur les liens et non pas d'un algorithme centralisé), et au fait d'avoir des agents autonomes cherchant à minimiser leur date d'arrivée, on utilise le *prix de l'anarchie* [KP99] : le prix de l'anarchie correspond au rapport maximal entre la valeur de la fonction objectif globale dans le *pire* équilibre de Nash, et sa valeur dans une solution optimale. Ainsi, alors que le rapport d'approximation mesure l'impact sur les performances d'avoir des capacités de calcul limitées, le rapport de compétitivité l'impact de ne pas connaître le futur à l'avance, le prix de l'anarchie (aussi appelé rapport de coordination) mesure l'impact d'avoir des utilisateurs cherchant chacun à maximiser leur propre fonction objectif et non la fonction objectif du système.

1.1 Autres travaux

Le prix de l'anarchie a été introduit dans [KP99] et a depuis été étudié dans de nombreux travaux. Les deux problèmes les plus étudiés sont le problème d'ordonnement (qui représente un réseau de liens parallèles entre deux noeuds : une source et une destination), introduit dans [KP99] et le problème introduit dans [RT02] et qui consiste à router dans un graphe un très grand nombre de paquets de taille infinitésimale (ce problème devient alors un problème de flot). Dans ces deux problèmes, le but de chaque tâche est de minimiser sa date d'arrivée, et l'objectif global est de minimiser la dernière date d'arrivée.

Afin de réduire le prix de l'anarchie, les auteurs étudient dans [CKN04] un problème d'ordonnement sur des liens parallèles et proposent de réduire le prix de l'anarchie en introduisant la notion de mécanisme de coordination c'est-à-dire en donnant à chaque lien une politique adaptée (voir aussi [ILMS05]). Notre objectif est ici de continuer cette étude sur des réseaux dont la topologie n'est plus simplement un ensemble de liens parallèles. Notre étude commence par les réseaux les plus simples : les chemins, les arbres, et se poursuit dans le cas des anneaux.

Le routage de paquets dans des graphes, où chaque paquet a une source et une destination, et où l'on souhaite minimiser la date d'arrivée maximum est un problème NP-difficile [SSW91] qui a donné lieu à de nombreuses publications (voir par exemple [LMR99, OR97]). Cependant, dans ces travaux, les auteurs considèrent que les tâches ont toutes la même longueur, et donnent principalement des algorithmes centralisés ou distribués mais dans lesquels chaque lien a plus d'information qu'uniquement les tâches qu'il route.

1.2 Résultats

Les résultats que nous avons obtenus sont résumés dans le tableau suivant. Les colonnes Max (resp. Moyenne) contiennent les prix de l'anarchie obtenus pour le problème de la Date d'Arrivée Maximum (resp. Date d'arrivée Moyenne). Chaque ligne contient les valeurs obtenues pour une politique donnée (les résultats obtenus sont les mêmes pour LPT et LRT). Par exemple, si on a un anneau dont les liens ont une politique SPT, alors le prix de l'anarchie pour le problème de la Date d'Arrivée Maximum est compris entre 2 et 3.

Ces résultats sont valables quelque soit la politique appliquée par les liens pour départager des tâches de même priorité (par exemple si un lien a comme politique LRD et si plusieurs tâches à la même distance de leur destination demandent à être routées par ce lien, alors ce lien va choisir la tâche à ordonner suivant une autre politique : SPT, numéro d'identification, etc.)

Politique	Chemin		Arbre		Anneau	
	Moyenne	Max	Moyenne	Max	Moyenne	Max
SPT	1	2*	1	2*	$1.34 < x \leq 2$	$2 \leq x < 3$
LPT, LRT	non borné	non borné*	non borné	non borné*	non borné	non borné*
LRD	non borné	1	non borné	2	non borné	$1.5 \leq x < 3$

TAB. 1 – Prix de l’anarchie pour les problèmes Date d’arrivée Maximum et Date d’arrivée Moyenne, suivant les politiques des liens. Par manque de place, cet article contient uniquement les preuves des résultats signalés par une étoile.

1.3 Notations

On note l_i la longueur de la tâche i , et le nombre d’arcs entre la source et la destination de i est noté x_i . C_i représente la date d’arrivée de la tâche i , et C_{max} représente la date d’arrivée de la tâche terminée en dernier. On appelle A_i l’arc sortant de la source emprunté par la tâche i , et \mathcal{B}_i est l’ensemble des tâches qui empruntent l’arc A_i avant la tâche i .

2 Cas du chemin et de l’arbre

2.1 Étude de la politique SPT

Théorème 1 Dans un arbre, le prix de l’anarchie de la politique SPT pour le problème de la Date d’Arrivée Maximum est inférieur à 2.

Preuve : Soit C_{max}^* la date d’arrivée de la dernière tâche arrivée à destination dans une solution optimale de ce problème. Considérons une tâche i , et montrons que sa date d’arrivée, C_i , est inférieure à $2C_{max}^*$. Puisque nous sommes dans un arbre, pour chaque couple (source, destination), il y a un seul chemin possible. De plus, plus une tâche est petite, plus elle part tôt de la source. Ainsi, une fois qu’une tâche est partie, elle ne rattrapera aucune tâche (partie précédemment), et ne sera rattrapée par aucune tâche (partie ensuite). La date d’arrivée de la tâche i est donc égale au temps que i attend avant de partir de la source, plus le temps de parcours de i dans l’arbre. Soit W_i le temps d’attente de i avant son départ de la source. On a donc : $C_i = W_i + (x_i l_i)$. De plus, W_i est inférieur à la somme de toutes les tâches (y compris i) qui empruntent le même arc que i à partir de la source, et C_{max}^* est nécessairement supérieure ou égale à cette somme. Ainsi $W_i < C_{max}^*$. De même, C_{max}^* doit être supérieure ou égale au temps minimum nécessaire pour que i atteigne sa destination : $x_i l_i \leq C_{max}^*$. Ainsi $C_i < 2C_{max}^*$, et donc $C_{max} < 2C_{max}^*$. \square

Théorème 2 Soit $\varepsilon > 0$. Dans un chemin, le prix de l’anarchie de la politique SPT pour le problème de la Date d’Arrivée Maximum est supérieur à $2 - \varepsilon$.

Preuve : Soit $0 < \varepsilon < 1$. Montrons que le prix de l’anarchie est supérieur à $2 - \varepsilon$ en considérant l’instance suivante : on a un chemin de $n = \lceil \frac{2}{\varepsilon} \rceil$ arcs. La source est le noeud étiqueté 0 et chaque noeud $i < n$ a un arc sortant vers le noeud $i + 1$. On a $(n - 1)$ tâches de longueur $1 - 1/n$ et une tâche t de longueur 1. La destination des tâches de longueur $1 - 1/n$ est le noeud 1, et la destination de t est le noeud n . Dans une solution optimale, t est ordonnancée en première position et la date d’arrivée maximum est $C_{max} = n$. Si les politiques de chaque arc sont SPT, alors t sera ordonnancée en dernière position et sa date d’arrivée sera son temps d’attente avant son départ $((n - 1)(1 - 1/n))$ plus le temps nécessaire pour atteindre sa destination (n), c’est à dire $(n - 1)(1 - 1/n) + n$. Ainsi le prix de l’anarchie est égal à $\frac{(n-1)(1-1/n)+n}{n} = 2 - \frac{2}{n} + \frac{1}{n^2} > 2 - \varepsilon$. \square

On déduit des théorèmes 1 et 2 le corollaire suivant :

Corollaire 1 Dans un arbre (resp. un chemin), le prix de l’anarchie de la politique SPT pour le problème de Date d’Arrivée Maximum tend vers 2.

2.2 Étude de la politique LPT

Théorème 3 Dans un chemin, le prix de l'anarchie de la politique LPT pour le problème de Date d'Arrivée Maximum n'est pas borné par une constante.

Preuve : Considérons l'instance suivante : on a un chemin de $m = 2^{n-1} + 1$ arcs, où n est n'importe quel (grand) nombre entier. La source est le noeud 0 et chaque noeud $i < m$ a un arc sortant vers le noeud $i + 1$. On a n tâches $\{1, \dots, n\}$, et chaque tâche i a une longueur $1/2^{i-1}$, et sa destination est le noeud $(2^{i-1} + 1)$. Soit C_{max}^* la date d'arrivée maximum dans une solution optimale, et $C_{max}(SPT)$ (resp. $C_{max}(LPT)$) la date d'arrivée maximale si les politiques des arcs sont SPT (resp. si les politiques des arcs sont LPT). On a : $C_{max}^* \leq C_{max}(SPT)$. Montrons que $C_{max}(SPT) \leq 3$. On montrera ensuite que la date d'arrivée maximum quand les politiques des machines sont LPT est non bornée.

Si les politiques sont SPT, alors, le seul temps d'attente pour une tâche est son temps d'attente avant le départ de la source. Le temps d'attente maximum est celui de la tâche la plus grande (tâche 1), et est égal à $\sum_{i=2}^n (1/2^{i-1}) < 1$. On sait que $C_{max}(SPT)$ est inférieur ou égal au temps d'attente maximum, plus le temps maximum nécessaire pour qu'une tâche atteigne sa destination. Le temps de parcours maximum est 2 (c'est celui de la plus grande tâche). Donc la date d'arrivée maximale est ici $1 + 2 = 3$.

Dans le cas où les politiques des arcs sont LPT, une tâche peut rattraper, mais ne peut pas dépasser une plus grande qu'elle. Ainsi, quand la tâche i est arrivée à sa destination, la tâche $i + 1$ a encore $2^{i-1} + 1$ arcs à traverser (elle avait $2^i + 1$ arcs à traverser et a déjà traversé 2^{i-1} arcs). Ceci représente un temps de parcours de $(2^{i-1} + 1)l_{i+1} = (2^{i-1} + 1) \times (1/2^i) > 1/2$. Puisque la tâche 1 atteint sa destination au temps 2, on en déduit $C_{max}(LPT) > 2 + \frac{1}{2}(n - 1)$. Le prix de l'anarchie est $\frac{C_{max}(LPT)}{C_{max}^*} > \frac{2}{3} + \frac{n-1}{6}$, ce qui peut devenir aussi grand que l'on veut lorsque n devient grand. \square

Ce résultat est aussi valable pour la politique LRT, et la preuve est la même.

3 Cas de l'anneau

On considère un anneau dans lequel il y a entre chaque noeud voisins un arc dans chaque sens. Ainsi, contrairement à ce qui se passait dans l'arbre, plusieurs chemins sont possibles et chaque tâche doit choisir à la source un des deux liens sortant de la source. Chaque tâche va choisir le lien qui va lui permettre de minimiser sa date d'arrivée. On peut ici bénéficier des bornes inférieures sur le prix de l'anarchie obtenues dans le cas du chemin. En effet, dans le cas où l'on a un anneau de très grande taille et si les destinations des tâches sont par exemple du côté droit de la source, alors dans la solution optimale comme dans l'équilibre de Nash, toutes les tâches vont demander à partir du côté droit de la source. On peut alors assimiler l'anneau à un chemin (l'arc gauche de la source n'étant jamais emprunté).

Références

- [CKN04] G. Christodoulou, E. Koutsoupias, and A. Nanavati. Coordination mechanisms. In *ICALP, LNCS 3142*, pages 345–357, 2004.
- [ILMS05] N. Immorlica, L. Li, V.S. Mirrokni, and A. Schulz. Coordination mechanisms for selfish scheduling. In *WINE, LNCS 3828*, pages 55–69, 2005.
- [KP99] E. Koutsoupias and C. H. Papadimitriou. Worst-case equilibria. In *STACS, LNCS 1563*, pages 404–413, 1999.
- [LMR99] F.T. Leighton, B.M. Maggs, and A.W. Richa. Fast algorithms for finding O(congestion + dilatation) packet routing schedules. *Combinatorica*, pages 375–401, 1999.
- [OR97] R. Ostrovsky and Y. Rabani. Universal O(congestion + dilatation + $\log^{1+\epsilon} n$) local control packet switching algorithms. In *STOC*, pages 644–653, 1997.
- [RT02] T. Roughgarden and E. Tardos. How bad is selfish routing? *Journal of the ACM*, 49(2):236–259, 2002.
- [SSW91] D.B. Shmoys, C. Stein, and J. Wein. Improved approximation algorithms for shop scheduling problems. In *SODA*, pages 148–15, 1991.