Découpage rigide des réseaux 5G avec FlexE

Nicolas Huin, Jeremie Leguay, Sébastien Martin, Paolo Medagliani

Découpage rigide des réseaux 5G avec FlexE

Nicolas Huin, Jérémie Leguay, Sébastien Martin et Paolo Medagliani

Huawei Technologies, French Research Center

La cinquième génération des réseaux mobiles (5G) promet d'améliorer l'utilisation des resources réseau grâce au *network slicing*, qui permet aux opérateurs de créer des réseaux virtuels pour chacun de ses clients avec une qualité de service (QoS) spécifique. Le découpage du réseau peut être fait de façon "souple" ou "rigide". La première manière est plus facile à mettre en place mais ne permet qu'une séparation logique des tranches et ne donne pas de garanties de QoS lors de la surcharge de l'une des tranches. Découper le réseau de façon "rigide", grâce à des technologies telles que Flex Ethernet (FlexE), permet de garantir une isolation physique des tranches. Cependant cette technologie nécessite d'allouer la bande passante par canaux. Dans cet article, nous introduisons le problème de routage et d'allocation de canaux FlexE dans un réseau 5G et proposons une décomposition par génération de colonnes pour le résoudre. Nous comparons ensuite cette approche sur des scénarios IPRAN de plusieurs tailles et montrons que notre décomposition permet d'obtenir des bornes inférieures et des solutions proches de l'optimal.

Mots-clefs : network slicing, column generation, 5G

1 Introduction

Les réseaux mobiles de cinquième génération promettent d'offrir à un grand nombre d'utilisateurs des services réseau personnalisés en terme de bande passante et de latence. La cohabitation d'utilisateurs ayant des besoins très différents force les opérateurs à repenser la gestion de leur réseau. Pour répondre à ce besoin, le *network slicing* simplifie la gestion des ressources en permettant de découper le réseau en *tranches*. Il existe deux types de *découpage* : le découpage "souple" (*soft slicing* [DPP+18]) et le découpage "rigide" (*hard slicing* [har18]). Alors qu'une surcharge de trafic dans l'une des tranches peut avoir un impact sur les autres dans le cas du *soft slicing*, le *hard slicing* permet une isolation parfaite.

Flex Ethernet [fle18] (FlexE) est une des technologies permettant d'effectuer du *hard slicing* grâce à un calendrier de transmission qui assure aux tranches un découpage temporel de la bande passante sur les liens. Cette garantie de performance nécessite cependant d'allouer la bande passante aux tranches par canaux de 1 Go (5 premiers canaux) ou 5 Go (ensuite).

Dans cet article, nous présentons le problème de routage et d'allocation de canaux FlexE dans les réseaux 5G. Il se rapproche du problème de *conception de réseau avec liens à coûts discrets* [GKM99]. Cependant, plusieurs différences sont à noter : les demandes ne peuvent pas être routées sur plusieurs chemins, elle ont des contraintes de QoS (e.g., latence de bout en bout) et un *ratio de convergence* peut être appliqué pour réaliser un multiplexage statistique sur un sous-ensemble de demandes (voir ci-dessous). Nous proposons une formulation étendue résolue par génération de colonnes ainsi que deux heuristiques puis nous les comparons sur des instances réalistes de réseaux 5G.

2 Problème de routage et d'allocation de canaux

Soit un réseau représenté par un graphe $G = (V,E)$ où V représente l'ensemble des nœuds du réseau et E l'ensemble des liens. Chaque lien possède un coût d'utilisation c_e et la capacité du lien dépend de l'ensemble des canaux qui peuvent y être activés. Les activations de canaux peuvent être exprimés sous forme de configurations, dénotées $s \in S^e$, où S^e est l'ensemble des configurations possible pour un lien e. La bande passante d'une configuration s est donnée par ξ_{es}, la somme des bande passantes des canaux activés. Par exemple, un lien Flex Ethernet de 10 Go peut être configuré à 1 Go, 2 Go, 3 Go, 4 Go, 5 Go ou 10 Go. Les délais de traitement et transmission sur les nœuds et les liens sont donnés par λ_v et λ_e,

respectivement. Une tranche est définie par un ensemble de demandes K à provisionner. Chaque demande $k \in K$ doit être routée du nœud s_k vers le nœud t_k avec un délai maximal de bout en bout Λ_k et requiert D_k unités de bande passante.

Un multiplexage statistique peut être appliqué à un sous-ensemble $K_C \subseteq K$ des demandes. Ce multiplexage repose sur le fait que ces demandes ont une faible probabilité d'être actives en même temps et permet donc de réduire l'utilisation de la bande passante. En pratique, chaque lien physique possède un ratio de convergence μ_e qui est calculé sur le ratio des capacités entre les différentes parties du réseau (accès, aggrégation, cœur). La bande passante utilisée par une demande $k \in K_C$ est donc de $\mu_e D_k$. Cependant, il faut aussi s'assurer que l'allocation de bande passante d'un lien soit suffisante pour transmettre la plus grande de demande de K_C . La bande passante sur un lien nécessaire pour un ensemble de demandes $\bar{K} \subseteq K_C$ est donc donnée par $\max(\sum_{k\in\bar{K}} \mu_e D_k, \max_{k\in\bar{K}}(D_k))$. Par exemple, si deux demandes de 4 Go que l'on peut multiplexer passent sur un lien avec un ratio de 4, 4 Go doivent être alloués (et non 2 Go).

Le problème de routage et d'allocation des canaux nécessite de *router les demandes de la tranche et de sélectionner les configurations de canaux sur les liens en respectant les contraintes de QoS et en minimisant le coût d'utilisation de la bande passante.*

3 Modèle et algorithme

Nous proposons le programme linéaire en nombre entier suivant, nommé *FlexE-CG* par la suite :

$$\min \sum_{e\in E} c_e \sum_{s\in S^e} \xi_{es} y_{es} \tag{1a}$$

$$\text{s.t} \sum_{k\in K\setminus K_C} \sum_{p\in P^k : e\in p} D_k x_{kp} + \sum_{k\in K_C} \sum_{p\in P^k : e\in p} \mu_e D_k x_{pk} \leq \sum_{s\in S^e} \xi_{es} y_{es} \quad \forall e \in E \tag{1b}$$

$$\sum_{k'\in K\setminus K_C} \sum_{p\in P^{k'} : e\in p} D_{k'} x_{k'p} + \sum_{p\in P^k : e\in p} D_k x_{pk} \leq \sum_{s\in S^e} \xi_{es} y_{es} \quad \forall e \in E, k \in K_C \tag{1c}$$

$$\sum_{p\in P^k} x_{pk} \geq 1 \quad \forall k \in K \tag{1d}$$

$$\sum_{s\in S^e} y_{es} \leq 1 \quad \forall e \in E \tag{1e}$$

$$x_{pk} \in \{0,1\} \quad \forall k \in K, p \in P^k \tag{1f}$$

$$y_{es} \in \{0,1\} \quad \forall e \in E, s \in S^e \tag{1g}$$

Cette formulation étendue est une formulation par chemin : les variables x_{pk} indiquent que la demande k est routée sur le chemin p et les variables y_{es} indiquent la configuration de canaux actifs sur un lien e (une configuration correspond à un ensemble de canaux actifs sur le lien). La fonction objectif vise à minimiser le coût d'utilisation du la bande passante. Chaque lien à un coût c_e par unité de bande passante. Les contraintes (1b) assurent que la réservation des canaux est suffisante pour tous les chemins utilisant le lien (en appliquant le convergence ratio) et les contraintes (1c) assurent que chaque demande appartenant à K_C puissent être transmise séparemment sans considérer le convergence ratio. Les contraintes (1d) forcent l'utilisation d'un seul chemin par demande et les contraintes (1e) limitent l'utilisation d'une configuration de canaux par lien.

Cette formulation contient un nombre exponentiel de variables et il est donc nécessaire d'utiliser un algorithme de génération de colonnes afin de la résoudre (voir Figure 1). La génération de colonnes repose sur une résolution d'un problème maître *réduit* (c-à-d avec un sous-ensemble de colonnes) suivi d'un problème de pricing qui cherche des colonnes à ajouter au problème maître pour améliorer la solution. Les deux problèmes sont résolus successivement jusqu'à ce qu'aucune colonne intéressante ne puisse être trouvée. Dans notre cas, l'algorithme de pricing correspond à trouver un chemin p pour la demande k qui viole l'inégalité

$$\sum_{e\in p} \left(\mu_e D_k \pi_e^{(1b)} + D_k \pi_{ek}^{(1c)} \right) - \pi_k^{(1d)} \leq 0 \tag{2}$$

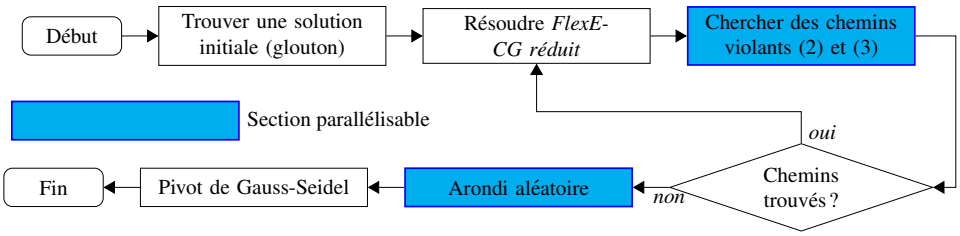

FIGURE 1: Algorithme de génération de colonnes pour *FlexE-CG*

Instance	Bornes inférieures		Temps	
	Compacte	*FlexE-CG*	Compacte	*FlexE-CG*
$\texttt{VLAN}_{\lvert V\rvert=50,\lvert K\rvert=60}$	**1.3965E+09**	1.3807E+09	26.10	2.03
$\texttt{FlexE}_{\lvert V\rvert=50,\lvert K\rvert=60}$	**1.2851E+09**	1.0661E+09	14.86	2.35
$\texttt{VLAN}_{\lvert V\rvert=1250,\lvert K\rvert=300}$	0.1323E+09	1.0421E+09	Limite atteinte	86.09
$\texttt{FlexE}_{\lvert V\rvert=1250,\lvert K\rvert=300}$	0.1273E+09	1.0802E+09	Limite atteinte	488.61

TABLE 1: Comparaison des bornes inférieures obtenues par la formulation compacte et *FlexE-CG*. Les valeurs en **gras** sont des bornes inférieures optimales.

si $k \in K_{\mathrm{C}}$, ou

$$\sum_{e \in p} \left(D_k \pi_e^{(1b)} + \sum_{k' \in K_{\mathrm{C}}} D_k \pi_{ek'}^{(1c)} \right) - \pi_k^{(1d)} \leq 0 \tag{3}$$

sinon, où le vecteur $\pi \in \mathbb{R}_+^{|E|+|E\times K_{\mathrm{C}}|+|K|+|E|}$ représente les valeurs duales des contraintes de *FlexE-CG*. Ces inégalités correspondent aux contraintes du dual associées aux variables de chemins du primal. Pour trouver ces chemins, nous cherchons un plus court chemin contraint par délai où le poids d'un lien est donné par $\mu_e D_k \pi_e^{(1b)} + D_k \pi_{ek}^{(1c)}$ (resp. $D_k \pi_e^{(1b)} + \sum_{k' \in K_{\mathrm{C}}} D_k \pi_{ek'}^{(1c)}$) pour une demande $k \in K_{\mathrm{C}}$ (resp. $k \in K \setminus K_{\mathrm{C}}$).

Pour initialiser *FlexE-CG*, nous utilisons un algorithme *glouton* de plus courts chemins contraints successifs qui cherche un chemin de coût minimum pour chaque demande. Il favorise les liens qui possèdent de la bande passante libre déjà allouée. Comme la génération de colonnes nécessite de relâcher les contraintes d'intégralité, nous utilisons un algorithme d'arrondi aléatoire afin de dériver une solution entière.

Pivot de Gauss-Seidel. Enfin, nous utilisons un algorithme permettant d'améliorer une solution existante, similaire à celui du *link-rerouting* présenté dans [GKM03], qui tente de réduire les allocations de canaux de chaque lien de façon gloutonne et de rerouter les demandes impactées sur ce réseau réduit. Plus précisément, nous sélectionnons le lien ayant le plus de bande passante libre allouée et nous retirons les demandes utilisant ce lien du réseau et libérant les ressources associés. L'allocation des canaux du lien est alors réduite aux canaux précédents et les demandes sont reroutées dans le réseau grâce à l'algorithme glouton. Si la nouvelle solution présente un coût plus petit, cette solution est gardée, sinon nous rétablissons l'ancienne solution. L'algorithme s'arrête lorsque tous les liens ont été considérés.

4 Résultats

Réseau IPRAN. Le réseau considéré est un réseau IPRAN avec plusieurs domaines reliés à un réseau cœur maillé. Les domaines sont composés de nœuds d'accès (réseau d'accès) connectés en *single* ou *dual homing* à un anneau avec des raccourcis (réseau d'agrégation). Les demandes peuvent exister entre les nœuds d'accès ou entre les nœuds d'accès et le cœur. Nous considérons un scénario où FlexE est déployé sur tous les liens, et un scénario où l'allocation de bande passante est faite grâce à VLAN (scénario de *soft slicing*). Dans le cas du *soft slicing*, la bande passante est allouée à une granularité de 1 Mo avec un minimum de 100 Mo.

Bornes inférieures. Nous comparons dans le tableau 1 les bornes inférieures obtenues par la relaxation de *FlexE-CG* à la fin de la génération de colonnes avec celles obtenues par la formulation compacte (par arc) du problème, avec un temps limite de 500s. Bien que la formulation compacte permette d'obtenir une borne optimale pour de petites instances (environ 50 nœuds) en moins de 30s, ses performances se dégradent pour de plus grosses instances. De plus, nous observons que *FlexE-CG* permet d'obtenir des bornes proches de la solution optimale et passent bien mieux à l'échelle. Alors que la formulation compacte ne trouve pas de solution optimale dans le temps imparti, la relaxation de *FlexE-CG* converge en moins de 500s (moins de 90s pour du *soft slicing*).

Instance	Écart avec la meilleure borne		Temps	
	Glouton	*FlexE-CG*	Glouton	*FlexE-CG*
$\texttt{VLAN}_{\lvert V\rvert=50,\lvert K\rvert=60}$	8.05%	4.23%	0.03	2.31
$\texttt{FlexE}_{\lvert V\rvert=50,\lvert K\rvert=60}$	9.89%	7.06%	0.02	2.64
$\texttt{VLAN}_{\lvert V\rvert=1250,\lvert K\rvert=300}$	3.67%	0.83%	0.16	96.01
$\texttt{FlexE}_{\lvert V\rvert=1250,\lvert K\rvert=300}$	9.40%	7.31%	0.13	765.46
$\texttt{VLAN}_{\lvert V\rvert=5000,\lvert K\rvert=600}$	3.92%	0.18%	0.87	313.82
$\texttt{FlexE}_{\lvert V\rvert=5000,\lvert K\rvert=600}$	9.47%	6.09%	0.68	1144.02

TABLE 2: Comparaison entre l'algorithme glouton et *FlexE-CG*.

Glouton vs *FlexE-CG* Le tableau 2 compare les solutions obtenues par l'algorithme glouton et la génération de colonnes. À la fin de chaque algorithme, le pivot de Gauss-Seidel est appliqué à la solution obtenue. L'algorithme glouton permet d'obtenir des valeurs proches de l'optimal en un temps très court (moins de une seconde même pour 5000 nœuds). *FlexE-CG* permet d'améliorer la solution obtenue par le glouton de 3 à 4% mais nécessite cependant plus de temps, jusqu'à environ 20 minutes. De plus, on peut observer une différence de temps et de qualité de la solution entre les deux types d'interfaces. Un réseau FlexE est beaucoup plus long à résoudre (20 minutes avec 5000 nœuds contre 5 minutes pour du VLAN) et les écarts avec la meilleure borne sont plus importants. Cela est dû à la plus grosse granularité de FlexE qui induit une différence plus importante avec la relaxation.

5 Conclusion

Nous avons présenté le problème de *routage et d'allocation de canaux* FlexE dans les réseaux 5G et proposons, pour le résoudre, une méthode de décomposition par génération de colonnes (*FlexE-CG*) ainsi qu'un algorithme glouton et une heuristique d'amélioration de solutions. Bien que plus lent que l'algorithme glouton, *FlexE-CG* permet d'obtenir des bornes inférieures proches de l'optimal ainsi que d'améliorer les solutions déjà bonnes de notre algorithme glouton. *FlexE-CG* pourrait être alors utilisé en complément de l'algorithme glouton dans des phases de reconfiguration du réseau, plus propices à des temps de calcul plus important.

Références

[DPP+18] A. Destounis, G. Paschos, S. Paris, J. Leguay, L. Gkatzikis, S. Vassilaras, M. Leconte, and P. Medagliani. Slice-based column generation for network slicing. In *IEEE INFOCOM 2018 (poster)*, April 2018.

[fle18] Flex Ethernet 2.0 Implementation Agreement, June 2018.

[GKM99] V. Gabrel, A. Knippel, and M. Minoux. Exact solution of multicommodity network optimization problems with general step cost functions. *Operations Research Letters*, 25(1) :15 – 23, 1999.

[GKM03] V. Gabrel, A. Knippel, and M. Minoux. A comparison of heuristics for the discrete cost multicommodity network optimization problem. *Journal of Heuristics*, 9(5) :429–445, Nov 2003.

[har18] Network Slicing Architecture, January 2018.