Routage vert et compression de règles SDN

Frédéric Havet[1] and Nicolas Huin[1] and Joanna Moulierac[1] and Khoa Phan[2]

[1] *University Nice Sophia Antipolis, Laboratoire I3S, UMR 7172, CNRS, INRIA, COATI, 06900 Sophia Antipolis, France.*
[2]*Department of Electronic and Electrical Engineering, University College London, United Kingdom.*

La technologie SDN permet de séparer le plan de contrôle et le plan de données qui cohabitent actuellement sur les routeurs dans les architectures réseaux classiques et de réaliser le routage par un ou plusieurs contrôleur(s) centralisé(s). Nos travaux portent sur l'utilisation de cette technologie pour minimiser la consommation d'énergie dans les réseaux, notamment en permettant au contrôleur d'éteindre à distance des liens non utilisés. Une des problématiques est que les tables de routage SDN ne peuvent contenir qu'un nombre très limité de règles. Ceci est dû au type particulier de mémoire utilisé pour permettre l'ajout à distance de règles de routage par le contrôleur SDN. Dans ce papier, nous étudions le problème de compression de tables de routage † bidimensionnelles avec priorité, en particulier la complexité algorithmique et proposons des algorithmes d'approximation. Nous proposons ensuite des algorithmes de routage vert qui effectuent en même temps le choix des routes, la compression des tables de routages et la mise en veille des liens non utilisés. Ces algorithmes sont testés sur les réseaux de la librairie SNDLib.

1 Introduction

Dans les réseaux classiques, les routeurs calculent à l'aide de protocoles de routage distribués sur quelle interface de sortie diriger les paquets pour une destination donnée. Dans la technologie "Software Defined Networks" (SDN), un ou plusieurs contrôleurs prennent en charge le calcul des routes et les routeurs sont réduits à de simples dispositifs de transmission. Quand un paquet arrive avec une nouvelle destination à laquelle ne correspond aucune règle de routage, le routeur contacte le contrôleur qui se charge d'établir une route et d'envoyer les règles de routage au routeur. Cette gestion centralisée et dynamique du réseau est une opportunité pour mettre en place des politiques vertes d'économie d'énergie qui s'adaptent à la charge de trafic des réseaux.

Dans ce papier, nous utilisons ce paradigme pour effectuer un "routage SDN vert" consistant à minimiser la consommation d'énergie du routage dans les réseaux de télécommunications. En effet, la différence de consommation entre un lien utilisé à pleine capacité et celle d'un lien non sollicité étant faible, il est intéressant d'essayer de regrouper sur certains liens le trafic et d'éteindre les autres qui ne sont pas utilisés. Le paradigme SDN nous permet d'avoir une gestion dynamique du réseau et de sa topologie. Ce problème a conduit à plusieurs travaux d'extinction de liens, notamment sur un réseau d'un data center [6] ou un réseau de télécommunications [1] .

Cependant, la majorité des dispositifs pouvant implémenter des règles SDN utilisent de la mémoire TCAM. Cette mémoire coûte cher, et consomme beaucoup d'énergie. De plus, les règles SDN installées sont plus complexes que des règles de routage classique. La taille maximum des tables de routage est donc très limitée et a été abordée par exemple dans [2] qui propose des algorithmes de routage (sans compression) qui limitent la taille maximale des tables de routage. Nous nous plaçons dans un cadre plus général où la compression des tables est permise et où en outre nous minimisons le nombre de liens réseaux employés. De plus, ce problème de placement de règles est exacerbé par l'extinction de liens sur le réseau. En réduisant le nombre de liens actifs, la taille moyenne des chemins augmente et donc le nombre total de règles nécessaires pour assurer un routage. Pour résoudre cette difficulté, les tables de routage peuvent être compressées en utilisant des règles par défaut ou d'agrégation que ce soit pour un élément de l'en-tête du paquet (ici source ou destination) ou pour tous les éléments.

†. Nous utilisons ici le terme "table de routage" dans un sens général. Il peut désigner une *routing table* ou une *forwarding table*.

Nos présentons ici des résultats théoriques de [5, 4], ainsi que les résultats des premières expérimentations sur les algorithmes tirés de ces travaux.

Nous avons tout d'abord modélisé avec un programme linéaire en nombre entiers le problème de routage SDN vert n'utilisant que des compressions à l'aide de la règle du port par défaut. La description de ce programme linéaire pour la version non dirigée peut être trouvée dans [5] et on peut facilement en dériver une version dirigée. Ce programme linéraire ne permet malheureusement que de traiter des réseaux ayant au plus quelques dizaines de routeurs. D'autre part, il ne considère qu'un type très restreint de compression.

Nous avons donc étudié le problème de compression de tables bidimensionnelles avec priorité. Ce problème est NP-complet [4]. Nous donnons une heuristique simple qui est une 3-approximation. Elle est décrite dans la sous-partie 3.1.

Nous étudions alors la résolution simultanée du choix des routes et de ce dernier type de compression. Puisque le routage vert (sans contraintes) [3] et la compression [4] sont NP-complets, nous proposons une heuristique de routage et d'extinction de liens qui prend en compte la capacité des liens et des tables de routage du réseau (voir sous-partie 3.1).

Finalement, nous avons testé les heuristiques proposées sur des réseaux SNDlib [7]. Les résultats obtenus sont présentés dans la sous-partie 3.2.

2 Définition du problème

Nous représentons un réseau par un graphe dirigé $G = (V,A)$. Un sommet est un routeur et un arc représente un lien entre deux routeurs. Chaque lien a une capacité maximum et le nombre de règles d'un routeur est limité par la taille de sa table de routage. Pour un ensemble de demandes $\mathcal{D}$, une solution de routage consiste à assigner à chaque demande un chemin tel que les contraintes de capacité et de taille de table soient respectées. Le problème du routage vert consiste à trouver une solution qui maximise le nombre de liens inactifs (sur lesquels aucun trafic ne passe) afin d'éteindre ceux-ci.

Les décisions de routage d'un routeur SDN s'effectuent grâce à des tables de flots implémentées via de la mémoire TCAM. Une entrée est définie par une règle de matching et une action à effectuer. Lors de la réception d'un paquet, le dispositif de routage sélectionne la règle correspondante avec la plus haute priorité et effectue l'action correspondante. Un paquet n'ayant pas d'action correspondante est traité par la règle par défaut, qui a la plus basse priorité. Dans ce travail, pour éviter des délais de transmission entre les routeurs et le contrôleur, nous définissons la règle par défaut qui va transmettre les paquets sur un port par défaut (sans contacter un contrôleur). Chaque dispositif possède un port par défaut.

Ces tables peuvent aussi contenir aussi des règles dites *d'agrégation*. Elles permettent de généraliser une action à effectuer sur l'ensemble des paquets correspondant à un motif particulier. Nous nous concentrons sur les motifs liés aux sources et destinations. Ces règles permettent de réduire la taille des tables en compressant les règles de même type pour les différents flots.

La compression d'une table consiste à réduire le nombre de règles de routage à l'aide de la règle par défaut ainsi que les règles d'agrégation. La figure 1 montre un exemple de compression optimale avec le port par défaut seul et avec l'utilisation de règle d'agrégation.

3 Résultats

3.1 Heuristique de compression (WC)

Puisque le problème de compression est NP-difficile [4], nous proposons une heuristique de compression simple des tables de routages ainsi qu'une heuristique de routage. Cette heuristique calcule en fait trois tables compressées et garde la plus petite des trois.

La première table, appelée *compression par défaut*, est la table obtenue par l'heuristique DP : On prend un des ports p_m qui apparait le plus dans les règles, et on remplace toutes les règles de port p_m par la règle par défaut $(*,*,p_m)$.

La deuxième table, appelée *compression par source*, est obtenue en agrégeant par source : On note p_s le port le plus présent pour les règles dont la source est s. Toutes les règles (s,t,p_s) sont remplacées par la règle d'agrégation $(s,*,p_s)$ et les règles (s,t,p) avec $p \neq p_s$ sont ajoutées avec une plus haute priorité que cette

Flot	Action
(0, 4)	Port-4
(0, 5)	Port-5
(0, 6)	Port-5
(1, 4)	Port-6
(1, 5)	Port-4
(1, 6)	Port-6
(2, 4)	Port-4
(2, 5)	Port-5
(2, 6)	Port-6

(a) Sans compression

Flot	Action
(0, 5)	Port-5
(0, 6)	Port-5
(1, 4)	Port-6
(1, 6)	Port-6
(2, 5)	Port-5
(2, 6)	Port-6
$(*,*)$	Port-4

(b) Port par défaut

Flot	Action
$(0,4)$	Port-4
$(1,5)$	Port-4
$(2,4)$	Port-4
$(2,5)$	Port-5
$(0,*)$	Port-5
$(*,*)$	Port-6

(c) Par source

Flot	Action
(1, 4)	Port-6
$(1,5)$	Port-4
$(0,6)$	Port-5
$(*,4)$	Port-4
$(*,5)$	Port-5
$(*,*)$	Port-6

(d) Par destination

Flot	Action
(1, 5)	Port-4
(2, 6)	Port-6
$(1,*)$	Port-6
$(*,4)$	Port-4
$(*,*)$	Port-5

(e) Optimale

FIGURE 1: Exemples de table de routage : (a) sans compression, (b) compression avec la règle par défaut seule, (c) compression par source, (d) compression par destination, et (e) la compression optimale

règle d'agrégation. Pour compresser encore un peu plus, on agrège certaines des règles d'agrégation en un règle par défaut comme suit : on détermine p_d un des ports les plus présents dans les règles d'agrégation et on remplace les règles $(s,*,p_d)$ par la règle par défaut $(*,*,p_d)$.

La troisième table, appelée *compression par destination*, est obtenue de manière similaire à la deuxième mais en considérant les destinations au lieu des sources. Nous montrons (voir [4]) que la plus petite de ces trois tables est au plus trois fois plus grande que l'optimale, ce qui montre que notre heuristique est une **3-approximation**.

Heuristique de routage

L'heuristique de routage se décompose en deux parties. La première consiste à trouver des chemins pour toutes les demandes en respectant les contraintes de capacités de lien et de taille de table de routage. La deuxième partie concerne l'extinction des liens.

Routage classique. La première partie de l'heuristique repose sur une recherche d'un plus court chemin pour chaque demande $(s,t) \in \mathcal{D}$. Elle s'effectue sur un graphe pondéré $G_{st} = (V,A',W)$ construit depuis G et dont le poids des arcs dépend des capacités des liens et de l'utilisation des tables de routage des routeurs.

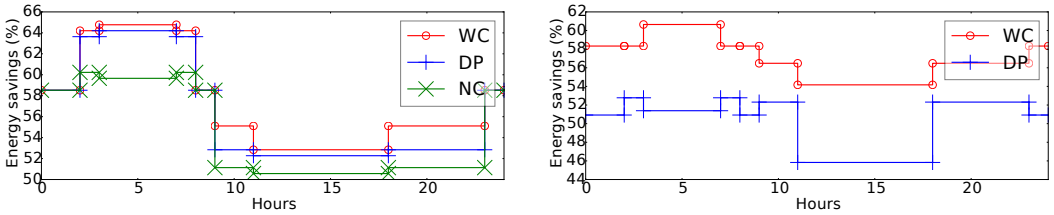
Lorsqu'un chemin p est trouvé pour la demande, pour chaque arc $(u_p,v_p) \in p$, la règle (s,t,v_p) est ajoutée au routeur u_p. Si jamais la table du routeur devient pleine, l'heuristique de compression décrite précédemment est appliquée.

Extinction des liens. Lorsque toutes les demandes sont satisfaites, les liens sont éteints un à un. L'algorithme de routage est relancé, sur le sous-graphe ne contenant pas le lien, pour chaque demande se trouvant sur le lien supprimé. Si une demande ne peut être satisfaite, le lien ne peut pas être éteint. L'heuristique s'achève lorsque tous les liens ont été évalués.

3.2 Résultats expérimentaux

Les heuristiques ont été appliquées sur les topologies et trafics réseaux tirés de SNDlib.

La figure 2 représente les économies d'énergie effectuées sur les réseaux germany50 ($|V| = 50, |A| = 176, |\mathcal{D}| = 2450$) et ta2 ($|V| = 65, |A| = 216, |\mathcal{D}| = 4160$) en utilisant aucune compression (NC), seulement le port par défaut (DP) et l'heuristique de compression compression (WC). La taille maximum des tables est fixée à 750, taille standard trouvée dans la littérature. Pour germany50, la méthode sans compression économise entre 50% d'énergie pour le trafic le plus haut (entre 10 et 20h) et 59% d'énergie pour le trafic le plus bas (au alentour de 5h). Les méthodes avec compression donnent de meilleurs résultats. Au trafic le plus bas, 64% des liens sont mise en veille et environ 52% pour le trafic le plus haut. La compression par ligne ou colonne effectue jusqu'à 3% d'économie en plus que la méthode utilisant seulement le port par défaut. Pour ta2, l'utilisation seule du port par défaut permet d'effecteur entre 50 et 52% d'économie d'énergie. Notre heuristique de compression permet quant à elle d'effectuer plus d'économie (entre 54 et 60%). Sans compression, il est impossible d'utiliser la technologie SDN avec une table avec 750 règles.



(a) germany50 (b) ta2

FIGURE 2: Économie d'énergie obtenue (en %) sur le réseau germany50 et ta2 en fonction de l'horaire pour l'heuristique de compression (WC), avec le port par défaut (DP), et sans aucune compression (NC)

Pour certains réseaux, cette limite de taille ne permet d'effectuer aucun routage. Pour germany50, un routage SDN vert est possible sans compression et effectue entre 50 et 60% d'économie d'énergie. Cependant, pour un réseau tel que ta2, le routage SDN vert n'est pas possible sans compression dû au grand nombre de demandes et donc de règles nécessaires dans les tables. Il devient possible en utilisant l'algorithme WC. Cette compression permet d'effectuer entre 52% et 64% d'économie d'énergie sur germany50 et entre 45% et 52% pour ta2. La compression WC permet, pour germany50, d'effectuer jusqu'à 3% d'économie en plus par rapport à l'utilisation seule du port par défaut. Cependant, pour des réseaux plus grands, la différence entre les deux types de compressions s'accentue. Pour ta2, la compression WC permet d'économiser entre 54% et 60% d'énergie, soit environ 10% de plus que le port par défaut.

4 Conclusion

Nous proposons des résultats sur la complexité de la compression de table de routage ainsi que des heuristiques de compression et de routage vert. L'ajout des règles d'agrégation permet d'effectuer une meilleure compression et donc une économie d'énergie plus importante que l'utilisation seule du port par défaut. La mise en veille de liens à pour effet de bord de réduire le degré sortant des nœuds (la plupart à 1) et de permettre une compression plus importante des règles.

Références

[1] Joseph Chabarek, Joel Sommers, Paul Barford, Cristian Estan, David Tsiang, and Steve Wright. Power awareness in network design and routing. In *INFOCOM'08 : the 27th IEEE Conference on Computer Communications.*, pages 457–465, april 2008.

[2] R. Cohen, L. Lewin-Eytan, J.S. Naor, and D. Raz. On the effect of forwarding table size on sdn network utilization. In *INFOCOM, 2014 Proceedings IEEE*, pages 1734–1742, April 2014.

[3] F. Giroire, D. Mazauric, J. Moulierac, and B. Onfroy. Minimizing routing energy consumption : From theoretical to practical results. In *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, pages 252–259, Dec 2010.

[4] Frédéric Giroire, Frédéric Havet, and Joanna Moulierac. Compressing Two-dimensional Routing Tables with Order. Research Report RR-8658, INRIA Sophia Antipolis, December 2014.

[5] Frédéric Giroire, Joanna Moulierac, and T Khoa Phan. Optimizing rule placement in software-defined networks for energy-aware routing. In *GLOBECOM*. IEEE, 2014.

[6] Brandon Heller, Srinivasan Seetharaman, Priya Mahadevan, Yiannis Yiakoumis, Puneet Sharma, Sujata Banerjee, and Nick McKeown. Elastictree : Saving energy in data center networks. In *NSDI*, volume 10, pages 249–264, 2010.

[7] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly. SNDlib 1.0–Survivable Network Design Library. *Networks*, 55(3) :276–286, 2010.