L'art de reconfigurer un noeud de routage

François Clad, Pascal Mérindol, Jean-Jacques Pansiot

L'art de reconfigurer un noeud de routage

Francois Clad, Pascal Merindol et Jean-Jacques Pansiot

ICube - Université de Strasbourg - {fclad, merindol, pansiot}@unistra.fr

Dans cet article nous proposons un algorithme efficace permettant de prévenir l'occurrence de boucles de routage transitoire, dans le cas d'opérations de modifications topologiques planifiées à l'échelle d'un routeur. Notre approche consiste à décomposer une modification topologique sous la forme d'une séquence de mises à jour, garantissant l'absence de boucle lors de chaque transition intermédiaire. L'objectif de notre algorithme est de minimiser le nombre d'opérations nécessaires pour reconfigurer un routeur.

1 Introduction

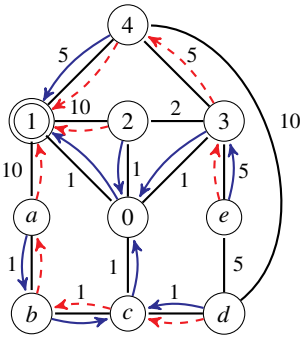
Avec le développement des applications temps-réel, les fournisseurs d'accès à Internet doivent faire face à des contraintes de plus en plus fortes. Ces contraintes se traduisent sous la forme de conventions de services définissant les garanties attendues d'un fournisseur, en termes de qualité et d'interruption de service. Actuellement, ces dernières sont principalement causées par des changements topologiques, dont une part non négligeable résulte d'opérations de maintenance [MIB+08]. Une autre source de changement topologique peut être la modification intentionnelle du poids de certains liens, afin d'optimiser le routage en fonction des fluctuations du trafic. Enfin, les changements imprévus, tels la panne d'un lien ou d'un routeur, sont également une source importante de problèmes de convergence, bien que leur impact puisse être limité grâce à des techniques de re-routage rapide. Chacun de ces changements force les routeurs à recalculer leurs tables de routage, faisant ainsi entrer le réseau dans un état transitoire pendant lequel des boucles de commutation peuvent apparaître. En effet, avec des protocoles à état des liens tels OSPF et IS-IS, l'ordre de mise à jour des routeurs n'est pas contrôlé et peut conduire à un plan de commutation transitoirement inconsistant. Ce phénomène est néfaste en terme de performance : perte de paquets, augmentation des délais et saturation de la capacité du lien. Plusieurs méthodes ont déjà été proposées pour résoudre ce problème. Néanmoins, celles-ci requièrent la modification des protocoles.

Dans cet article, nous présentons un mécanisme efficace permettant de prévenir l'apparition de boucles de commutation lors d'une opération de maintenance sur un routeur ou un sous-ensemble de ses interfaces. Notre solution est basée sur une mise à jour progressive du poids des liens dont l'état doit être modifié, et repose uniquement sur des fonctionnalités conformes aux standards. Ces travaux représentent une généralisation de l'approche que nous proposions dans [CMP+13] pour le cas d'une opération de maintenance affectant un unique lien. En effet, bien que théoriquement applicables, nos précédents algorithmes ne profitent pas de la possibilité de modifier simultanément le poids de plusieurs liens sortants d'un même routeur, produisant ainsi des séquences de reconfigurations trop longues pour être utilisables en pratique. Nous proposons ici un algorithme plus efficace capable de calculer des séquences minimales en temps polynomial.

2 Des boucles de routage aux contraintes vectorielles

Dans les réseaux utilisant des protocoles de routage à état des liens, les informations de commutation sont propagées via des messages de signalisation de type *Link-State Advertisement (LSA)*. Sur la base de ces informations, chaque routeur calcule localement un ensemble de plus courts chemins vers chacune des destinations annoncées à travers le réseau. L'extension *Equal Cost Multi-Path (ECMP)* permet à plusieurs chemins de coûts égaux d'être utilisés simultanément pour une même destination.

L'exemple proposé sur la figure 1a illustre le phénomène des boucles de commutation transitoires dans le cas du retrait d'un routeur. La topologie du réseau y est représentée comme un graphe, dans lequel les nœuds représentent les routeurs et les arêtes représentent les connections directes, ou adjacences IGP,



(a) Chemins ECMP vers 1

prochains sauts ***avant*** *prochains sauts* ***après***

Dest. 1 $l_1 = \{a,b,a\}$ $c_1 = \big((6 \quad -4 \quad -4 \quad -6), (8 \quad -2 \quad -2 \quad -4)\big)$
$l_5 = \{b,c,c\}$ $c_5 = \big((8 \quad -2 \quad -2 \quad -4), (10 \quad 0 \quad 0 \quad -2)\big)$
Dest. 2 $l_2 = \{c,d,c\}$ $c_2 = \big((-1 \quad 9 \quad 7 \quad -3), (1 \quad 11 \quad 9 \quad -1)\big)$
Dest. 3 $l_3 = \{c,d,c\}$ $c_3 = \big((-3 \quad 5 \quad 7 \quad -4), (-1 \quad 7 \quad 9 \quad -2)\big)$
Dest. 4 $l_4 = \{c,d,c\}$ $c_4 = \big((2 \quad 0 \quad 2 \quad -4), (4 \quad 2 \quad 4 \quad -2)\big)$

GBA : $c_1, c_4 \rightarrow c_2, c_3, c_5$ $S_{GBA} = \big\{(7 \quad 1 \quad 3 \quad 0), (9 \quad 10 \quad 8 \quad 0)\big\}$

(b) Contraintes de prévention de cycles et séquence minimale calculée avec GBA

FIGURE 1: Illustration du phénomène des boucles transitoires

utilisées par les routeurs pour s'échanger les informations de routage. Les arêtes sont également pondérées en fonction du coût attribué à chaque adjacence IGP. Pour des raisons de simplicité, nous avons choisi dans cet exemple d'utiliser des poids symétriques, bien que cette propriété ne fasse pas partie de nos hypothèses. Le réseau est modélisé comme un graphe dirigé et valué $G = (N, E, w)$, où N est l'ensemble des routeurs, E l'ensemble des adjacences entres ces routeurs, et w est une fonction associant un poids à chaque lien. Enfin, les flèches bleues (traits pleins) et rouges (traits pointillés) représentent les plus courts chemins utilisés par chaque routeur pour joindre la destination 1, respectivement avant et après le retrait de 0. Un tel ensemble de chemins vers une destination d donnée est appelé *graphe des plus courts chemins inversé*, et noté $RSPDAG(d)$. Cet exemple permet d'observer comment la convergence induite par le retrait du routeur 0 peut conduire à l'apparition de boucles de commutation. Si le routeur c met à jour sa *Forwarding Information Base (FIB)* avant le routeur b, alors c commencera à envoyer son trafic à destination de 1 suivant la nouvelle topologie tandis que b s'en tiendra à la topologie initiale. Une boucle de commutation apparaitra donc entre b et c. Un phénomène similaire peut se produire entre les routeurs a et b, si le second est mis à jour avant le premier, mais également pour d'autres destinations. Même si ces boucles disparaissent lorsque les routeurs impliqués ont convergé, des paquets peuvent être perdus durant la période d'instabilité.

Dans cet article, nous étudions comment prévenir l'apparition de telles boucles lors d'opérations de maintenance sur un unique routeur. Bien que nous nous focalisions ici sur le cas du retrait d'un routeur, une méthode similaire peut être appliquée pour un ajout ou des changements de valuation. Intuitivement, afin d'empêcher les boucles de commutation de se former, il faudrait pouvoir forcer l'ordre de mise à jour des routeurs. S'il existe déjà une solution basée sur un ordonnancement explicite des mises à jour [FB07], celle-ci requiert l'intégration d'extensions dans les standards définissant les protocoles de routages à état des liens. Au contraire, nous souhaitons proposer une solution reposant uniquement sur les mécanismes de base de ces protocoles, et qui pourrait ainsi être déployée de manière incrémentale sur les routeurs. Pour cela, nous allons progressivement réduire l'*attractivité* du routeur qui doit être retiré en incrémentant les poids associés à ses liens sortants. Concrètement, ce routeur sera chargé d'envoyer une séquence de LSAs calculés par notre algorithme. Les autres nœuds recalculeront alors leur routage en fonction des LSAs reçus. L'envoi de deux LSAs successifs devra être espacé d'un délai permettant aux routeurs de converger vers un nouvel état de routage. Ainsi, l'envoi d'un grand nombre de LSAs se traduira par une augmentation de la durée de convergence totale et une plus forte sollicitation du processeur des routeurs. Ces deux aspects étant des problématiques majeures pour les protocoles de routage, notre objectif algorithmique sera de minimiser la longueur de la séquence à injecter pour garantir l'absence de boucle (minimiser le nombre de LSAs intermédiaires). Dans ce but, il est possible de profiter de l'opportunité offerte par les protocoles à état de liens de modifier simultanément, dans le même LSA, le poids sur plusieurs liens sortant d'un même routeur. Ce nouveau degré de liberté permet de réduire le nombre de LSAs requis. Il s'agit d'une généralisation à plusieurs dimensions du problème de minimisation des séquences de mise à jour introduit dans [CMP+13].

Sur l'exemple de la figure 1a, l'apparition d'une boucle entre les routeurs a et b est conditionnée par l'ordre dans lequel les LSA indiquant le retrait du nœud 0 sont pris en compte par ces deux routeurs. En effet, cette boucle peut survenir car a envoie initialement son trafic à destination de 1 vers b, alors que l'inverse a lieu après le retrait de 0. Ainsi, la boucle ne peut pas se produire si a converge vers 1 avant que b ne modifie ses états de routage. Cela se produit, par exemple, si un poids de 8 est configuré sur le lien $(0,1)$.

Le plus court chemin de a vers 1 utilisera alors le prochain saut 1 alors que celui de b passera toujours par 0. Cette reconfiguration de poids, préalable au retrait de 0, permettra donc d'éviter une boucle entre a et b pour la destination 1. Si le poids sur le lien $(0,1)$ est configuré à 7 ou à 9, il est *trop tôt* ou *trop tard*. Dans le premier cas, l'incrément de poids n'est pas suffisant pour forcer le routeur a à abandonner sa route initiale : le routeur reste dans un état intermédiaire utilisant deux routes de coûts égaux. Dans le second cas, l'incrément est trop important et déclenchera la convergence des deux routeurs. On peut observer une contrainte stricte sur le poids qui devra être configuré sur le lien $(0,1)$. En revanche, les liens $(0,2)$ et $(0,3)$ sont déjà moins attractifs pour ces deux routeurs que le lien $(a,1)$, de sorte qu'aucune contrainte ne leur est associée pour cette boucle. Leurs poids n'a aucune incidence sur l'apparition de la boucle. Il est possible de combiner dans un seul vecteur les contraintes associées à plusieurs boucles si elles sont *compatibles*.

L'ensemble des contraintes à résoudre pour satisfaire le problème des boucles transitoires peut être modélisé sous la forme de *vecteurs d'incréments*. Ceux-ci représentent un ensemble d'incréments de poids à appliquer sur les liens sortants du nœud modifié. Soit un vecteur d'incrément v, on note $v[i]$ son i-ème composant et $|v|$ son cardinal. Deux vecteurs v_1 et v_2 de même cardinal peuvent être comparés et une relation d'ordre partiel existe entre eux. Les opérations de comparaison usuelles entre vecteurs sont définies de manière standard. Soit un routeur $x \neq 0$, on définit $\Delta_d^0(x)$ comme le vecteur d'incréments minimum tel que le routeur x utilise au moins un *nouveau chemin*, n'incluant pas 0, pour atteindre la destination d. Les valeurs dans ce vecteur sont obtenues à partir des propriétés de plus courts chemins dans les graphes G et $G' = G \setminus \{0\}$. Soient l_i le i-ème lien sortant du routeur 0, $C(x,l_i,d)$ le coût du plus court chemin élémentaire de x à d passant par l_i dans G et $C'(x,d)$ le coût du meilleur chemin de x à d dans G', on a : $\forall\ i \in [\![1,k]\!],\ \Delta_d^0(x)[i] = C'(x,d) - C(x,l_i,d)$. Ce différentiel représente un ensemble d'incréments de distances. Par définition de $\Delta_d^0(x)$, on sait donc que le routeur x a cessé d'utiliser une route vers d passant par 0 après l'application du vecteur d'incréments v si et seulement si $v > \Delta_d^0(x)$. En effet, cela signifie pour que, pour le routeur x, le chemin vers d passant par 0 est maintenant plus coûteux que son chemin final. Ces vecteurs nous permettent de modéliser des contraintes sur les poids des liens entourant le nœud 0 afin de prévenir l'apparition de boucles. A chaque boucle transitoire L, pouvant apparaitre lors du retrait du nœud 0, on associe une contrainte c telle que : $c := (\underline{c} := \min_{\forall x \in L}(\Delta_d^0(x)), \bar{c} := \max_{\forall x \in L}(\Delta_d^0(x)))$. Une telle contrainte $c = (\underline{c}, \bar{c})$ est *satisfaite* si et seulement si il existe un vecteur intermédiaire v tel que $v > \underline{c}$ et $v \not\geq \bar{c}$. L'absence de la boucle associée est alors garantie, car au moins l'un des routeurs impliqués aura convergé vers son état final avant que la boucle ne soit formée. Une séquence d'incréments intermédiaires permettra donc d'éviter toutes les boucles potentielles à la condition que chacune des contraintes associées à ces boucles soit satisfaite par au moins l'un des vecteurs qui composent la séquence.

3 Calcul d'une séquence de mise à jour minimale

Cette partie décrit de manière informelle le fonctionnement de notre algorithme, *Greedy Backward Algorithm (GBA)*. Les considérations sur son efficacité sont décrites en détail dans [CMV+13]. Au niveau macroscopique, GBA se décompose en deux étapes :

i) extraire les contraintes associées aux boucles potentielles ;

ii) calculer de manière itérative la séquence d'incréments :

  (a) calculer le vecteur glouton courant et l'ajouter en fin de séquence ;

  (b) mettre à jour l'ensemble des contraintes en supprimant celles déjà satisfaites.

En pratique, et pour chaque destination d, les contraintes associées à chaque boucle potentielle peuvent être extraites efficacement via des algorithmes de détection de boucles ou de calcul de chemins dans le graphe résultant de l'union de $RSPDAG(d)$ et $RSPDAG'(d)$. Il s'agit alors de calculer une séquence minimale satisfaisant l'ensemble de ces contraintes. GBA fonctionne à "l'envers", c'est-à-dire qu'il calcule les vecteurs composant la séquence dans l'ordre inverse de leur application. Le premier vecteur calculé sera celui contenant les valeurs les plus élevées, et donc le dernier à être appliqué. En effet, selon les définitions fournies précédemment, une contrainte est satisfaite par un vecteur v, même si certains éléments de v sont supérieurs à la *borne supérieure* de la contrainte, tant qu'au moins l'un des éléments $v[i]$ composant le vecteur intermédiaire est inférieur à $\bar{c}[i]$. De cette simple propriété résulte un grand nombre de

combinaisons possibles pour satisfaire un sous-ensemble de contraintes. Dans l'exemple proposé sur la figure 1b, on voit notamment que la contrainte c_4 peut-être combinée avec n'importe quelle autre contrainte, mais pas toutes à la fois. Cependant, cette possibilité n'existe plus s'agissant des bornes inférieures des contraintes. Ainsi, il est possible de calculer le vecteur glouton minimal respectant les bornes inférieures de toutes les contraintes, sans se préoccuper des bornes supérieures. Formellement, soit C l'ensemble des contraintes, ce vecteur est défini comme suit : $\forall\, i \in [\![1,k]\!], v[i] = max_{c\in C}(\underline{c}[i]) + 1$. Si $v[i] < 0$, on considèrera $v[i] = 0$. Un tel vecteur respecte également, a minima, la borne supérieure des contraintes c_m vérifiant $\exists\, i \in [\![1,k]\!], \underline{c}_m[i] = max_{c\in C}(\underline{c}[i])$. GBA se base sur cette propriété afin de calculer de manière déterministe une séquence de vecteurs intermédiaires satisfaisant l'ensemble des contraintes. À chaque étape, le vecteur glouton courant est calculé et les contraintes satisfaites par ce vecteur sont supprimées. Cette suppression pourra être faite de manière explicite si les bornes supérieures des contraintes sont connues, mais également de manière *implicite*. Dans le second cas, on choisira de mettre à jour l'ensemble des contraintes en remplaçant l'état final par celui induit par le dernier vecteur calculé. Le même processus est ensuite répété jusqu'à ce que toutes les contraintes aient été satisfaites. Nous avons prouvé dans [CMV$^+$13] qu'une séquence ainsi calculée est minimale.

Dans l'exemple de la figure 1, la contrainte c_5 présente avec 8 la valeur la plus élevée sur le premier élément, correspondant au lien $(0,1)$, parmi les bornes inférieures des cinq contraintes. De même, c_2 dispose de la plus grande valeur pour le lien $(0,2)$, et est à égalité avec c_3 sur lien $(0,3)$. Les trois premières valeurs du vecteur calculé par GBA seront donc, dans l'ordre, 9, 10 et 8, tandis que, en l'absence de contrainte sur le lien $(0,c)$, la dernière valeur restera nulle. Ce vecteur satisfait les contraintes c_2, c_3 et c_5, qui seront alors supprimées. Lors de sa deuxième itération, on ne considèrera plus que les contraintes c_1 et c_4, lesquelles seront satisfaites par le vecteur $(7\ \ 1\ \ 3\ \ 0)$. La séquence $\{(7\ \ 1\ \ 3\ \ 0),(9\ \ 10\ \ 8\ \ 0)\}$ ainsi calculée est de taille minimale. En effet, dans la mesure où $\underline{c}_5 \geq \bar{c}_1$, ces deux contraintes ne peuvent pas être satisfaites par un unique vecteur. On remarque également que les routeurs a à e ne sont pas considérés comme des destinations dans le calcul de la séquence. En effet, une phase préliminaire à GBA consiste à éliminer les destinations pour lesquelles aucune boucle ne peut apparaitre lors d'un changement topologique donné. Ce mécanisme permet en pratique d'éliminer une grande partie des destinations, réduisant ainsi de manière significative la complexité de GBA.

4 Conclusion

Afin de permettre des modifications de l'état d'un routeur dans les réseaux utilisant des protocoles de routage à état des liens, nous avons conçu un algorithme efficace, appelé *GBA*, reposant sur des mises à jour successives du poids des liens sortants de ce routeur. Dans le cas de l'ajout ou du retrait d'un routeur, notre algorithme permettra de progressivement dévier le trafic vers, ou hors, de ce routeur tout en garantissant l'absence de boucles de routage transitoires. Dans la pratique les séquences de mises à jour peuvent être calculées par n'importe quelle machine ayant une connaissance complète du réseau, et sont ensuite appliquées par le routeur dont l'état doit être modifié.

Références

[CMP$^+$13] F. Clad, P. Merindol, J.-J. Pansiot, P. Francois, and O. Bonaventure. Graceful Convergence in Link-State IP Networks : A Lightweight Algorithm Ensuring Minimal Operational Impact. *IEEE/ACM Transactions on Networking*, 2013.

[CMV$^+$13] F. Clad, P. Merindol, S. Vissicchio, J.-J. Pansiot, and P. Francois. Graceful Router Updates for Link-State Protocols. In *Proceedings of IEEE ICNP*, Göttingen, Germany, October 2013.

[FB07] P. Francois and O. Bonaventure. Avoiding transient loops during the convergence of link-state routing protocols. *IEEE/ACM Transactions on Networking*, 15(6) :1280–1292, December 2007.

[MIB$^+$08] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, Y. Ganjali, and C. Diot. Characterization of Failures in an Operational IP Backbone Network. *IEEE/ACM Transactions on Networking*, 16 :749–762, August 2008.