Vérification de tables de routage par utilisation d'un ensemble représentatif d'en-têtes

Yacine Boufkhad, Ricardo De La Paz, Leonardo Linguaglossa, Fabien Mathieu, Diego Perino, Laurent Viennot

Vérification de tables de routage par utilisation d'un ensemble représentatif d'en-têtes

Yacine Boufkhad[1], Ricardo de la Paz[2], Leonardo Linguaglossa[2], Fabien Mathieu[3], Diego Perino[3], and Laurent Viennot[2]

[1] *Université Paris Diderot - Paris 7,* [2] *INRIA,* [3] *Alcatel-Lucent Bell Labs France*

Vérifier des tables de routage consiste à tester la validité des tables de l'ensemble des routeurs d'un réseau donné. Par exemple, il peut s'agir de tester l'absence de *boucle* ou de *trou noir*. Dans cet article, nous proposons une approche en deux étapes : construire un ensemble représentatif d'en-têtes de paquets, puis tester les propriétés désirées sur ces en-têtes. Toute la difficulté est de construire un ensemble relativement petit tout en garantissant qu'un paquet avec un en-tête arbitraire va se comporter exactement comme au moins l'un des en-têtes de l'ensemble représentatif.

À partir d'un modèle très général inspiré du paradigme *Software Defined Networking* (SDN), nous montrons que le problème de la détection de boucles peut se résoudre en temps polynomial en la taille d'un ensemble représentatif. Nous montrons aussi qu'une condition naturelle sur les règles de routage, inspirée des travaux de Boutier et Chroboczek sur le routage par source et destination, permet de construire un ensemble représentatif dont la taille est au plus le nombre de règles de routage plus un. Par comparaison, sans cette condition naturelle, la taille de l'ensemble représentatif peut dans le pire des cas être exponentielle en la taille des en-têtes.

Keywords: SDN, techniques de vérification et test, tables de routage

1 Introduction

Un des enjeux de l'administration des réseaux est le diagnostic de problèmes tels que l'existence de boucles ou de trous noirs. Cette opération peut être réalisée par l'analyse des configurations du plan de contrôle ou des tables de routage du plan de commutation. Nous choisissons dans cet article la seconde approche car elle permet de détecter un plus grand nombre de problèmes et simplifie l'analyse dans le cas de protocoles et d'implantations hétérogènes [4].

La vérification de tables est un problème complexe, surtout lorsque la multiplication des protocoles fait que les règles peuvent s'appliquer à de multiple champs de l'en-tête des paquets. C'est le cas des systèmes *Software Defined Networking* (SDN), où une règle peut spécifier une valeur (ou un intervalle) pour chaque champ, les métacaractères ($*$) étant autorisés.

Dans ce modèle très général, la détection d'anomalies dans les tables est un problème NP-difficile. Notre objectif est de trouver des hypothèses plus restrictives permettant de résoudre ces problèmes en temps polynomial. Nous proposons une approche en deux étapes : construction d'un ensemble représentatif d'en-têtes de paquets ; test des propriétés désirées sur ces en-têtes. Nous montrons que le problème de la détection de boucles peut se résoudre en temps polynomial étant donné un ensemble représentatif H, et nous indiquons brièvement comment le même genre de technique s'applique à la détection d'autres anomalies du plan de commutation. Nous montrons ensuite qu'une condition naturelle sur les règles de routage, inspirée des travaux de Boutier et Chroboczek sur le routage par source et destination, permet de construire un ensemble représentatif dont la taille est au plus le nombre de règles de routage plus un. Par comparaison, sans cette condition naturelle, la taille de l'ensemble représentatif peut dans le pire des cas être exponentielle en la taille des en-têtes. Le calcul d'un ensemble représentatif est NP-difficile dans le cas général. Néanmoins, il devient polynomial lorsque les règles sont données sous forme de préfixes ce qui est le cas du routage IP.

2 Modélisation du réseau

Considérons le modèle réseau suivant, inspiré par SDN. Une *instance réseau* est la donnée des éléments suivants : un graphe G représentant des routeurs et leurs connexions ; des tables de routage $T(u)$ associées à chaque $u \in V(G)$. Une table de routage est une liste ordonnée de règles $r_1, \ldots, r_k$. Chaque règle est composée d'un masque et d'une action à appliquer aux paquets dont l'en-tête vérifie le masque. Un masque est une chaîne de ℓ lettres de l'alphabet $\{1, 0, *\}$, où ℓ est le nombre de bits de l'en-tête (supposé de taille fixe), qui peut aussi se voir comme une formule booléenne. Par exemple, le masque $1*0**$ sera vérifié par tout en-tête dont le premier bit vaut $h_1 = 1$ et le troisième $h_3 = 0$, et peut donc aussi se voir comme la formule $h_1 \wedge \overline{h_3}$. Par abus de notation, r désigne aussi l'ensemble des en-têtes qui vérifient le masque de r. Nous considérons ici trois actions possibles : transférer le paquet à un voisin v, jeter le paquet ou traiter le paquet (quand celui-ci est arrivé à destination). Quand un paquet arrive, les règles sont testées dans un certain ordre $(r_1, \ldots, r_k)$ et il est soumis à l'action de la première règle vérifiée. Autrement dit, r_i est appliquée à un paquet d'en-tête h ssi $h \in r_i \cap \overline{r_1} \cap \cdots \cap \overline{r_{i-1}}$. Si aucune règle n'est vérifiée ($h \in \overline{r_1} \cap \cdots \cap \overline{r_k}$), le paquet est jeté. Malgré sa simplicité, ce modèle capture la plupart des situations classiques de routage, comme le routage IP (basé sur le choix du plus long préfixe correspondant à l'adresse de la destination).

Pour une instance, nous considérons les vérifications relativement classiques suivantes [2, 3, 4].

Boucle: Un paquet peut-il tourner en boucle dans le réseau ? Formellement, est-ce qu'il existe un en-tête h et un cycle $u_1, \ldots, u_k, u_1$ dans G tels que pour chaque $1 \leq i < k$, l'action appliquée à h par u_i est le transfert au successeur dans le cycle ?

Trou noir: Est-ce que des paquets peuvent être jetés par certains routeurs alors qu'ils sont transférés ou traités par d'autres ?

Atteignabilité: Étant donnés deux sommets $s, t \in V(G)$, est-ce que des paquets peuvent aller de s à t ?

Cohérence: Est-ce que deux sommets $s, t \in V(G)$ jettent exactement les mêmes paquets ?

On peut également considérer des problèmes plus simples :

Non vérifié: Pour un routeur donné, existe-t-il un en-tête qui n'est vérifié par aucune règle ?

Inutilité: Pour un routeur donné, et-ce qu'il existe des règles qui ne seront jamais appliquées quel que soit l'en-tête du paquet entrant ?

On peut facilement voir que **Non vérifié** est équivalent à **SAT**. Plus généralement, la satisfiabilité d'une formule booléenne peut être exprimée dans n'importe lequel des problèmes ci-dessus, ce qui prouve qu'ils sont NP-difficiles (cela avait déjà été remarqué dans [4]). Nous proposons de capturer la difficulté de tous ces problèmes dans le calcul d'un ensemble représentatif d'en-têtes. Une fois cet ensemble calculé, tous ces problèmes deviennent polynomiaux.

3 Ensemble représentatif d'en-têtes

Pour une instance réseau G donnée, on appelle *collection* $\mathcal{R}$ de G la collection de tous les ensembles d'en-têtes définis par les différentes règles des routeurs : $\mathcal{R} = \cup_{u \in V(G)} T(u)$, où $T(u)$ désigne la collection des ensembles définis par les règles de u.

Une *combinaison* est un ensemble obtenu en combinant des règles de $\mathcal{R}$ par intersection, union et passage au complémentaire. L'ensemble de toutes les combinaisons possibles est noté $\mathcal{C}(\mathcal{R})$. $\mathcal{C}(\mathcal{R})$ peut aussi être vu comme la *tribu* engendrée par $\mathcal{R}$, i.e. la plus petite collection contenant $\mathcal{R}$ et stable par intersection, union et passage au complémentaire. Notons que pour ℓ donné, $\mathcal{C}(\mathcal{R})$ est de cardinalité au plus 2^ℓ (c'est une collection d'ensembles d'en-têtes possibles).

Définition 1 *Un* ensemble représentatif d'en-têtes *associé à une collection $\mathcal{R}$ est un ensemble d'en-têtes H tel que tout élément non vide de $\mathcal{C}(\mathcal{R})$ contient un en-tête de H.*

$$H \text{ est un ensemble représentatif de } \mathcal{R} \Leftrightarrow \forall c \in \mathcal{C}(\mathcal{R}), c \neq \emptyset, \exists h \in c \cap H$$

Tous les problèmes de vérification évoqués dans la section précédente peuvent se résoudre en temps polynomial si l'on possède un ensemble représentatif de taille polynomiale, comme nous allons le montrer pour le problème **Boucle**.

$H = \emptyset$
Pour chaque $r \in \mathcal{R}$ **faire**
  $s \leftarrow \cup_{r' \subsetneq r} r'$
  Si $r \setminus s \neq \emptyset$ **alors**
    ajouter $h \in r \setminus s$ à H
Si $s \leftarrow \overline{\bigcup_{r' \in \mathcal{R}} r'} \neq \emptyset$ **alors**
  ajouter $h \in s$ à H

Algorithme 1: Construction d'un ensemble représentatif H à partir d'une collection de règles $\mathcal{R}$.

Théorème 1 *Étant donnée une instance réseau à n sommets et un ensemble représentatif de taille m headers associé à la collection des règles de G,* ***Boucle*** *peut être résolu en temps* $O(nm)$.

Preuve. On peut vérifier en temps $O(nm)$ qu'aucun en-tête de H ne boucle. L'algorithme consiste à calculer pour chaque en-tête $h \in H$ son graphe des transferts G_h, i.e. le sous-graphe de G contenant les arêtes $E(G') = \{(u,v) \in E(G)$ telles que h est transféré de u vers $v\}$, et à vérifier si G_h contient une boucle (par exemple avec un parcours en profondeur). G_h se calcule par en injectant h dans chaque table $T(u)$ pour $u \in V(G)$. La complexité de l'algorithme est donc de $O(nm)$ interrogations de tables.

Reste à montrer que si aucun en-tête de H ne boucle, aucun en-tête ne peut boucler. Supposons qu'il existe un en-tête h quelconque qui boucle et montrons qu'il existe $h' \in H$ qui boucle également. Soit $u_1, ..., u_k$ avec $u_k = u_1$ une suite de sommets sur lesquels h boucle. Soit $r^a_1, ..., r^a_{k_a}$ la liste ordonnée des règles de u_a, et i_a l'indice de la règle appliquée par u_a sur h. Étant donné le fonctionnement des tables, $h \in C_a$, où C_a est la combinaison $C_a = r^a_{i_a} \cap \overline{r^a_1} \cap \overline{r^a_2} ... \cap \overline{r^a_{i_a-1}}$. On a donc $h \in C_1 \cap C_2 \cap ... \cap C_k$. Par définition de H, il existe forcément $h' \in H \cap C_1 \cap C_2 \cap ... \cap C_k$. En particulier, $h' \in r^a_{i_a} \cap \overline{r^a_1} \cap \overline{r^a_2} ... \cap \overline{r^a_{i_a-1}}$, et chaque routeur u_a applique à h' la même règle i_a qu'à h. h' suit donc la même boucle que h. □

Notons que le théorème 1 implique que chaque parcours de table soit fait en temps constant, comme c'est typiquement le cas si des algorithmes optimisés sont utilisés [5]. Il peut se généraliser à tous les problèmes évoqués dans la section précédente. Pour chaque problème, il faut calculer le graphe de transfert G_h de chaque en-tête $h \in H$ ($O(nm)$ interrogations de tables) et tester l'anomalie recherchée. Par exemple, pour **Trou noir**, on teste l'existence d'un G_h non vide qui contient un routeur sans arête sortante et ne traitant pas le paquet. Comme dans la preuve du théorème 1, on montre que si un en-tête génère une anomalie, il existe un en-tête de H qui génère la même anomalie.

4 Condition suffisante pour avoir un petit ensemble représentatif

Boutier et Chroboczek ont introduit une notion de complétude dans le cadre du routage par source et destination [1], où les règles sont des paires de masques préfixes appliqués aux adresses IP de départ et d'arrivée. Cette notion se généralise à notre modèle.

Définition 2 ([1]) *Une collection de règles* $\mathcal{R}$ *est* fortement complète *(resp.* faiblement complète*) si pour tous* $r, r' \in \mathcal{R}$*, on a* $r \cap r' \in \mathcal{R}$ *(resp.* $r \cap r' = \cup_{r'' \subseteq r \cap r'} r''$*).*

Théorème 2 *Toute collection de règles* $\mathcal{R}$ *faiblement complète admet un ensemble représentatif de taille au plus* $|\mathcal{R}| + 1$*, qui peut se construire en suivant l'algorithme 1.*

Preuve. L'algorithme 1 construit un ensemble H à partir d'une collection $\mathcal{R}$. Comme il n'ajoute au maximum qu'un en-tête par règle de $\mathcal{R}$, plus un en-tête qui ne vérifie aucune règle s'il en existe, la taille de H est au plus $|\mathcal{R}| + 1$ par construction. Reste à montrer que si $\mathcal{R}$ est faiblement complet, alors H est représentatif, c'est-à-dire que tout élément non vide $c \in \mathcal{C}(\mathcal{R})$ contient un élément de H.

Soit c un élément non vide de $\mathcal{C}(\mathcal{R})$. Soit $h \in c$ et $a(h)$ l'atome associé ($a(h) = \cap_{c' \in \mathcal{C}(\mathcal{R}), h \in c'} c'$). $a(h)$ est un élément de $\mathcal{C}(\mathcal{R})$ (car on est en cardinalité finie) minimal pour l'inclusion. Comme $a(h)$ est minimal, il peut s'écrire comme l'intersection de règles de $\mathcal{R}$ et de complémentaires de règles de $\mathcal{R}$: $a(h) = r_1 \cap r_2 \cap ... \cap r_{i-1} \cap \overline{r_i} \cap \overline{r_{i+1}} \cap ... \cap \overline{r_k}$. Comme $a(h) \subseteq c$, il suffit de montrer que $a(h)$ contient un en-tête de H.

Si $a(h)$ est obtenu uniquement par intersection de complémentaires de règles de $\mathcal{R}$ ($i = 1$), c'est forcément le complémentaire de l'union de toutes les règles de $\mathcal{R}$. La dernière ligne de l'algorithme 1 va alors sélectionner un entête adéquat.

Sinon ($i > 1$), on a $r_1 \cap \ldots \cap r_{i-1} \neq \emptyset$. Par la faible complétude, on peut écrire $r_1 \cap \ldots \cap r_{i-1} = (\bigcup_{r' \subseteq r_1 \cap r_2} r') \cap r_3 \cap \ldots \cap r_{i-1}$ qui se généralise en $r_1 \cap \ldots \cap r_{i-1} = \bigcup_{r' \subseteq (\bigcap r_{1\ldots i-1})} r'$. Soit r_m une règle incluse dans $\bigcap r_{1\ldots i-1}$, qui contient h, et de taille minimale. Étudions comment l'algorithme 1 a réagi à r_m pour $s = \cup_{r' \subsetneq r_m} r'$. On constate tout d'abord que $r_m \setminus s$ n'est pas vide car il contient au moins h (sinon, r_m ne serait pas de taille minimale). On voit également que tout élément de $r_m \setminus s$ appartient à $\overline{r_i} \cap \ldots \cap \overline{r_k}$. En effet, $r_m \cap \overline{r_i} \cap \ldots \cap \overline{r_k} = r_m \setminus \bigcup_{j=i\ldots m} (r_j \cap r_m)$. Chaque $r_j \cap r_m$ peut se décomposer en règles $r \subsetneq r_m$, d'où $\bigcup_{j=i\ldots m} (r_j \cap r_m) \subset s$, et donc $r_m \setminus s \subset r_m \cap \overline{r_i} \cap \ldots \cap \overline{r_k}$.

En résumé, l'algorithme 1 a choisi un en-tête h' pour r_m. Cet en-tête appartient à $r_1 \cap r_2 \cap \ldots \cap r_{i-1} \cap \overline{r_i} \cap \overline{r_{i+1}} \cap \ldots \cap \overline{r_k}$, donc il appartient à $a(h)$, donc à c. □

En pratique, la complexité du calcul de l'ensemble représentatif va dépendre de la nature des règles déployées dans l'instance réseau. Si dans le cas général ce calcul peut être NP-difficile (trouver un élément de $r \setminus s$ revient à résoudre une formule SAT), il existe des situations où c'est relativement simple.

Par exemple, si les règles de $\mathcal{R}$ sont uniquement composées de préfixes appliqués à un unique champ (comme dans le routage IP), la complexité est réduite. En effet, dans ce cas, chaque règle est constituée d'une suite de 1 et de 0 de longueur variable i suivie de $\ell - i$ lettres $*$, e.g. elle correspond à l'ensemble des headers dont les i premiers bits sont ceux donnés par la règle. On remarque que $\mathcal{R}$ est naturellement fortement complet (l'intersection correspond au préfixe le plus long). Le théorème 2 s'applique donc et assure l'existence d'un ensemble représentatif de petite taille (au plus $|\mathcal{R}| + 1$). On peut construire efficacement cet ensemble en disposant les règles $r \in \mathcal{R}$ dans un trie binaire. Les en-têtes représentatifs sont obtenus en parcourant ce trie : chaque nœud incomplet (ayant au plus un fils) du trie correspond à une règle telle que $r \setminus s \neq \emptyset$, pour laquelle on rajoutera un en-tête choisi en dehors du fils éventuel. La complexité est celle d'un parcours du trie, soit $O(\ell \cdot n)$, où n est le nombre de règles. Ce genre d'approche peut se généraliser pour des règles multi-champs avec des règles de type préfixe ou intervalle pour chaque champ.

5 Conclusion

Nous avons abordé le problème de la vérification des tables de routage dans le modèle général des systèmes SDN. Bien que la recherche d'anomalies dans les tables soit un problème NP-difficile en général, nous avons proposé une approche en deux étapes basée sur le concept d'ensemble représentatif d'en-têtes qui permet de vérifier la présence d'anomalies (comme des boucles) en temps polynomial. Cependant, le calcul de l'ensemble représentatif reste NP-difficile et sa taille peut être exponentielle en général. Nous avons également montré qu'une condition relativement naturelle de complétude sur les règles de routage permet de borner la taille d'un ensemble représentatif par le nombre de règles plus un. Dans des travaux futurs, nous comptons montrer que le calcul d'un ensemble représentatif est polynomial dans le cas assez générique de règles multi-champs de type préfixe par champ et obtenir ainsi un cadre efficace et suffisamment général pour la détection systématique des anomalies d'un réseau.

Références

[1] Matthieu Boutier and Juliusz Chroboczek. Source-sensitive routing. In *IFIP Networking*, 2015.

[2] Peyman Kazemian, Michael Chang, Hongyi Zeng, George Varghese, Nick Mckeown, Scott Whyte, and U C San Diego. Real Time Network Policy Checking using Header Space Analysis. In *NSDI*, 2013.

[3] Ahmed Khurshid, Xuan Zou, Wenxuan Zhou, Matthew Caesar, and P Brighten Godfrey. VeriFlow : Verifying Network-Wide Invariants in Real Time. In *NSDI*, 2013.

[4] Haohui Mai, Ahmed Khurshid, P Brighten Godfrey, and Samuel T King. Debugging the Data Plane with Anteater. In *SIGCOMM*, 2011.

[5] Matteo Varvello, Rafael Laufer, Feixiong Zhang, and T.V. Lakshman. Multi-layer packet classification with graphics processing units. CoNEXT, 2014.