Générateur de topologies pour les réseaux hétérogènes et multicouches

Noureddine Mouhoub, Maria Moloney, Damien Magoni

Générateur de topologies pour les réseaux hétérogènes et multicouches[†]

Noureddine Mouhoub[1] , Maria Moloney[2], Damien Magoni[1]

[1] *LaBRI-CNRS, Université de Bordeaux, 33405, Talence, France*
[2] *PEL, University College Dublin, Belfield, Dublin 4, Ireland*

L'utilisation croissante des réseaux virtuels et superposés a complexifié les architectures réseaux avec des couches de protocoles supplémentaires, nécessitant de nouveaux protocoles de routage comme le routage multicouche avec établissement automatique de tunnels. Initialement conçus par simulation, de tels protocoles nécessitent des topologies réseaux adaptées, ce qui pose problème car la plupart des générateurs de topologies existants ne peuvent produire que des topologies plates (monocouche). Nous proposons dans cet article un nouveau générateur de topologies réseaux multicouches nommé Multilayer Network Topology Generator (MNTG). Les simulations de protocoles de routage multicouche sur différentes topologies générées par MNTG montrent que ces topologies ont un impact significatif sur les résultats des simulations.

Mots-clefs : réseaux multicouches, encapsulation de protocoles, génération de topologies, graphes aléatoires.

1 Introduction

La simulation dans le domaine des réseaux est un outil crucial pour concevoir et analyser des protocoles. Une configuration appropriée de la topologie du réseau est essentielle pour obtenir des simulations réalistes. Depuis les années 90, différents générateurs ont été développés pour reproduire les topologies observées dans les réseaux réels. Néanmoins, ces générateurs ne peuvent produire qu'une topologie monocouche ayant un seul protocole. Avec l'avènement des réseaux virtuels et de la coexistence protocolaire (*ex.,* IPv4/IPv6, VPN, IoT, etc.), les protocoles se sont multipliés et les piles de protocoles sont devenues plus complexes. Des protocoles de routage dynamique capables de calculer des routes à travers plusieurs couches ont été proposés et leur évaluation nécessite l'utilisation de topologies de réseaux multicouches [LLKC19]. Dans ce contexte, un seul générateur de ce type, nommé MulNeG [PKK15], existe actuellement. Cependant, MulNeG s'applique aux réseaux de la couche application, tels que les réseaux sociaux et n'est pas capable de générer des topologies réseaux de couches inférieures comportant des conversions et encapsulations. Nous proposons dans cet article un générateur de topologies réseaux multicouche qui répond à ce besoin [‡].

2 Modèle du réseau multicouche

2.1 Modèle

Nous reprenons le même modèle de réseau et les mêmes notations que dans [LLKC19]. Un réseau multicouche est un quadruplet $N = (G, A, F, w)$, où : $G = (V, E)$ est un graphe orienté représentant la topologie du réseau avec n nœuds et m liens. $A = \{x, y, ...\}$ est l'ensemble fini des λ protocoles disponibles dans le réseau. L'ensemble des protocoles qu'un nœud v peut recevoir (resp. émettre) est noté par $In(v)$ (resp. $Out(v)$). L'ensemble $F(v)$ représente les *fonctions d'adaptation* que le nœud v peut effectuer. Une fonction d'adaptation qui encapsule un protocole x dans un autre y est notée $(x \rightarrow xy)$. L'opération inverse (désencapsulation ou extraction de x) est notée $(xy \rightarrow x)$. Enfin, une conversion de x vers y est notée $(x \rightarrow y)$. Le cas particulier $(x \rightarrow x)$ correspond à une retransmission sans changement de protocole. Enfin, $w : V \times F \times V \rightarrow \mathbb{R}^+$ est une

[†]Financé par le projet HÉRA ANR-18-CE25-0002.

‡. La version complète de cet article a été publiée dans les actes de la conférence IEEE VCC 2023.

fonction de poids où $w(v_i, f, v_j)$ est le coût d'utilisation de la fonction d'adaptation f par v_i et de l'envoi du paquet résultant sur le lien (v_i, v_j). Un exemple d'un tel réseau, où les protocoles IPv4 et IPv6 coexistent, est illustré dans la Figure 1.

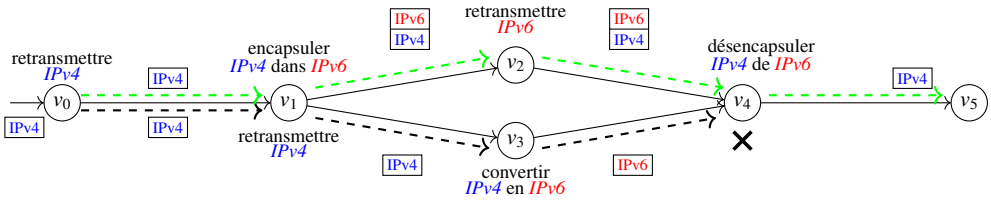


FIGURE 1 – Exemple de réseau englobant les conversions et encapsulations des protocoles IPv4 et IPv6. Le chemin du haut en vert est faisable, et contient un tunnel de v_1 à v_4. Le chemin du bas en noir n'est pas faisable. La pile de protocoles spécifique à chaque chemin est affichée à côté de chaque lien sortant.

2.2 Piles de protocoles et chemins faisables

Une séquence de fonctions d'adaptation $f_i f_{i+1} \dots f_{j-1}$ appliquée à une pile de protocoles de départ h_i induit une pile de protocoles d'arrivée h_j, où chaque pile $h_k = f_{k-1}(h_{k-1})$ pour tout $i+1 \leq k \leq j$. On notera $f(h)$ l'application de la fonction d'adaptation f à la pile h dont le sommet est $Top(h)$. Parfois, l'application n'est pas possible. Dans cette situation, nous notons $f(h) = \phi$ (pile interdite), ainsi on ne pourra pas aller plus loin, *i.e.,* $f(\phi) = \phi$ quelle que soit la fonction d'adaptation f.

Un chemin multicouche p d'un nœud v_i à v_j est une séquence mixte de nœuds et de fonctions d'adaptation avec une pile de départ et une pile d'arrivée $h_i v_i f_i v_{i+1} f_{i+1} \dots v_{j-1} f_{j-1} h_j v_j$. On dit que p est *faisable* ssi : la séquence $v_i v_{i+1} \dots v_{j-1} v_j$ est un chemin classique dans G et chaque $f_k \in F(v_k)$; la séquence $f_i f_{i+1} \dots f_{j-1}$ appliquée à la pile h_i induit la pile h_j ; les piles de départ et d'arrivée ne sont pas interdites *i.e.,* $h_i \neq \phi$ et $h_j \neq \phi$ et leurs sommets appartiennent aux nœuds correspondants *i.e.,* $Top(h_i) \in In(v_i)$ et $Top(h_j) \in In(v_j)$. Enfin, le poids d'un chemin faisable p est noté par $w(p) = \sum_{k=i}^{j-1} w(v_k, f_i, v_{k+1})$.

3 Générateur multicouche

Dans la section précédente, nous avons défini le réseau multicouche comme un graphe orienté avec des protocoles et des fonctions d'adaptation. Pour créer un générateur de topologies de réseau multicouche, nous proposons une approche en deux étapes. Tout d'abord, nous générons le graphe sous-jacent, soit aléatoirement, soit en fusionnant des cartes de réseaux réels. Ensuite, nous répartissons les fonctions d'adaptation sur les nœuds, soit de manière aléatoire, soit en fonction des capacités des nœuds (monoprotocoles ou multiprotocoles). Cette approche offre une modélisation réaliste en tenant compte des capacités spécifiques de chaque nœud.

3.1 Algorithmes

La méthode *mono-aléatoire* consiste à générer un seul graphe aléatoire avec des fonctions d'adaptation distribuées de manière probabiliste sur les nœuds, déterminant ainsi les nœuds multiprotocoles. L'algorithme associé distribue les fonctions d'adaptation sur chaque nœud en utilisant un ensemble de protocoles communiqué. Pour garantir l'utilisation de toutes les fonctions de la distribution, l'ensemble est initialisé avec tous les protocoles disponibles. L'Algorithme 1 distribue aléatoirement les fonctions d'adaptation une par une aux nœuds, vérifiant si chaque fonction peut gérer les protocoles des nœuds correspondants. Si oui, la fonction est attachée au nœud avec une certaine probabilité p.

La méthode *multi-aléatoire* utilise plusieurs graphes aléatoires, un par couche de protocole disponible. Les nœuds multiprotocoles sont générés par fusion aléatoire des nœuds des graphes sous-jacents. La répartition des fonctions d'adaptation sur chaque nœud dépend de ses protocoles et de la probabilité p de chaque fonction. L'Algorithme 2 permet de créer une topologie multi-aléatoire, générant des nœuds multiproto-

Algorithme 1 : Génération de nœuds mono-aléatoires

Pour tout $v \in V(G)$ **faire**
  $A \leftarrow$ Tous_Protocoles ()
  # Distribution_Aléatoire_Fonctions (A, v)
  Pour tout $f \in F$, $f = (x \rightarrow y)$ *ou* $(x \rightarrow xy)$ *ou* $(xy \rightarrow x)$ **faire**
    Si $(x,y) \in A$ *et Nombre_Aléatoire_Réel (0, 1)* $<$ *Probabilité (*f*)* **alors**
      $F(v) \leftarrow F(v) \cup \{f\}$

coles en sélectionnant aléatoirement k nœuds parmi k graphes distincts pour créer un nœud communicant k protocoles. Une fois créé, les fonctions d'adaptation sont distribuées à l'aide de l'algorithme précédent.

Algorithme 2 : Génération de nœuds multi-aléatoires

Pour tout $k = 1, 2, \ldots, |V_j|$, $j \in \{1, 2, \ldots, \lambda\}$ **faire**
  Pour tout $l = 1, 2, \ldots, j$ **faire**
    $(G_i, v_i) \leftarrow \big($ Sélectionner_Graphe $(G_1, G_2, \ldots, G_\lambda)$, Sélectionner_Nœud $(V(G_i))\big)$
    $(A, v) \leftarrow \big($ Ajouter_Protocole (i), Fusionner_Nœud $(v_i)\big)$
  Distribution_Aléatoire_Fonctions (A, v)

La méthode *multi-réelle* utilise des graphes prédéfinis et des fichiers d'entrée multi-stack pour générer des nœuds multiprotocoles par fusion entre différents graphes sous-jacents. Les fonctions d'adaptation sont distribuées de manière plus réaliste, avec des nœuds monoprotocoles limités à une retransmission classique et des nœuds multiprotocoles capables de convertir, encapsuler/désencapsuler ou les deux. L'Algorithme 3 permet de créer des nœuds multiprotocoles à partir des données de cartographie réseaux réelles fournies par l'utilisateur dans un fichier multi-stack, puis distribue les fonctions d'adaptation sur ces nœuds.

Algorithme 3 : Génération de nœuds multi-réels

Pour tout $(v_1, v_2, \ldots, v_\lambda) \in$ *Fichier Multi-Stack* **faire**
  Pour tout $v_i \in (v_1, v_2, \ldots, v_\lambda)$ **faire**
    Si *Non_Vide(*v_i*)* **alors**
      $(A, v) \leftarrow \big($ Ajouter_Protocole (i), Fusionner_Nœud $(v_i)\big)$
  # Distribution_Réelle_Fonctions (A, v)
  Si $|A| = 1, A = \{x\}$ **alors**
    $F(v) \leftarrow F(v) \cup \{x \rightarrow x\}$
  Sinon
    Pour tout $(x,y) \in A$ **faire**
      $F(v) \leftarrow F(v) \cup \big(\{x \rightarrow y, y \rightarrow x\}$ et/ou $\{x \rightarrow xy, xy \rightarrow x\}$ et/ou $\{y \rightarrow yx, yx \rightarrow y\}\big)$

3.2 Architecture et Implémentation

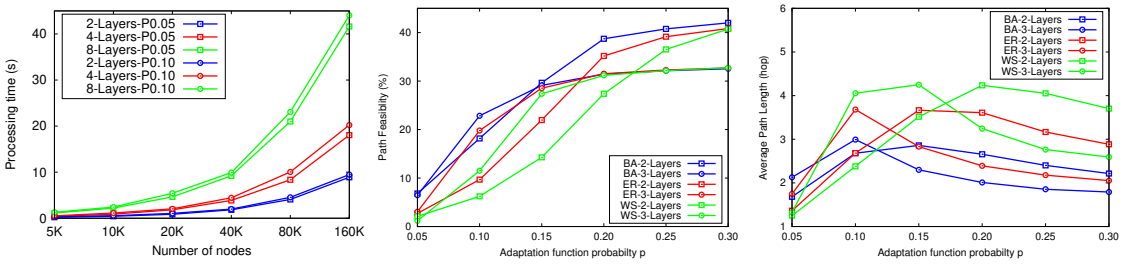
La génération de réseaux multicouches avec MNTG démarre par l'analyse des paramètres à partir d'un fichier d'entrée. Les graphes sont ensuite générés ou chargés en fonction des paramètres spécifiés. La création du réseau multicouche débute par la génération de toutes les fonctions d'adaptation possibles à partir d'un ensemble de protocoles. Les nœuds, incluant ceux multiprotocoles, sont générés en utilisant trois méthodes : mono-aléatoire, multi-aléatoires et multi-réelles. Notre générateur est implémenté en C++ pour s'assurer une exécution rapide. Il utilise la bibliothèque igraph [§] pour la génération de graphes aléatoires. Le code source est disponible en open source sur un dépôt Gitlab hébergé par l'Université de Bordeaux [¶].

§. `https://igraph.org`
¶. `https://gitub.u-bordeaux.fr/hera/mntg`

Trois types de fichiers d'entrée sont nécessaires pour générer une topologie de réseau multicouche : un fichier de paramètres, des fichiers de graphes sous-jacents et un fichier multi-stack. Les fichiers de graphes sous-jacents sont au format Pajek NET, tandis que le fichier multi-stack décrit les nœuds communs entre les graphes d'entrée est au format texte ASCII. Pour représenter les graphes multicouches, nous utilisons deux nouveaux formats. Le premier est un format texte ASCII décrivant les informations du réseau, des nœuds et des liens. Le deuxième format est basé sur le format DOT de Graphviz, offrant la possibilité d'ajouter des attributs et des clusters pour représenter la topologie produite par notre générateur.

4 Évaluation des performances

Des topologies ont été générées avec MNTG et utilisées par des simulations de l'algorithme de routage multicouche *Stack-Vector* [LLKC19]. Les topologies générées, ont été évaluées selon différents paramètres tels que la méthode de génération de graphes aléatoires (Barabàsi-Albert, Erdös-Rényi et Watts-Strogatz), la taille du réseau n, la probabilité p de disponibilité des fonctions d'adaptation sur les nœuds, et le nombre de couches (ou de protocoles) λ.



(a) Temps de traitement. (b) Pourcentage des chemins faisables. (c) Longueur moyenne des chemins.

FIGURE 2 – Consommation de ressources et impact des topologies générées sur l'algorithme Stack-Vector.

La figure 2a montre que le temps de génération d'une topologie est principalement impacté par le nombre de couches. Dans la figure 2b, nous pouvons voir que le pourcentage de chemins faisables varie fortement selon la probabilité des fonctions d'adaptation, du nombre de couches et du modèle de grpahe sous-jacent. De même, la figure 2c indique que la longueur moyenne de chemins varie de façon significative selon ces mêmes paramètres. Ces deux dernières figures montrent que les résultats sont fortement influencés par les paramètres du générateur. Par conséquent, il est important de simuler un protocole sur plusieurs types de topologies générées par divers jeux de paramètres pour capturer pleinement son comportement.

5 Conclusion et perspectives

Nous avons présenté l'architecture et les algorithmes d'un générateur de topologies de réseaux multicouches et évalué ses performances. Nous visons désormais à optimiser les méthodes de génération tout en améliorant la connectivité et le réalisme des fonctions d'adaptation. Nous utiliserons des probabilités spécifiques pour chaque fonction d'adaptation, calculées à l'aide de la combinatoire analytique. Nous validerons également les topologies générées en les comparant à des cartes de réseaux réels multicouches.

Références

[LLKC19] M. L. Lamali, S. Lassourreuille, S. Kunne, and J. Cohen. A stack-vector routing protocol for automatic tunneling. In *IEEE INFOCOM Conference*, pages 1675–1683, 2019.

[PKK15] Adrian Popiel, Przemysław Kazienko, and Tomasz Kajdanowicz. Muneg : The framework for multilayer network generator. In *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, page 1316–1323, 2015.