Rapport projet
Recherche et Développement
Instanciation de topologies réseau complexes

Arthur Garnier
Maître d'apprentissage : Lucas Nussbaum
Tuteur Académique : Thibault Cholez

11 Janvier 2016

Table des matières

1 Introduction

L'équipe proposant le sujet travaille sur le développement d'une plate-forme, Grid'5000, pour les chercheurs travaillant dans le domaine des sciences informatiques. L'objectif de la plate-forme est de mettre à disposition une infrastructure informatique permettant de faire des tests à échelle réelle. L'accent est mis sur les calculs distribués incluant le Cloud, le HPC [1] et les masses de données. La force de Grid'5000 est sa possibilité de personnalisation, il est possible pour chaque utilisateur de choisir son OS, le matériel dont il a besoin (Ethernet 10, Infiniband, GPU dédiés, ...) grâce aux 1000 noeuds de calculs mis à disposition.

Les utilisateurs ont également la possibilité de configurer des VLANs pour leurs expériences et donc de gérer entièrement l'isolation réseau des noeuds. Ce dernier point est important dans le cadre de la recherche sur les systèmes distribués : ils nécessitent de travailler sur des topologies réseau complexes afin d'être proche des réseaux réels tels qu'Internet. Plusieurs solutions ont déjà été développées à cet effet, une première partie sera donc dédiée à une étude bibliographique de ces solutions afin de dégager une vision globale de ce qui existe. L'objectif étant d'assembler un maximum des avantages de ces solutions autour de notre outil de gestion de VLAN, KaVLAN, dans l'optique d'offrir aux utilisateurs un outil d'instanciation de topologie complexe simple d'utilisation dans la plate-forme Grid'5000.

1. High performance computing

2 Solutions existantes

2.1 Présentation

Les systèmes et les réseaux sont de plus en plus complexes, cette croissance de complexité implique pour les scientifiques de se baser de plus en plus sur des expérimentations tant les modèles théoriques deviennent compliquer à modéliser. Il existe sur les systèmes informatiques trois moyens de réaliser des expérimentations[**?**] : échelle réelle, émulation et simulation. Les expérimentations à échelle réelle consistent à exécuter son expérience sur des machines physiques. L'émulation permet de réaliser un environnement sous forme de modèle, souvent cela consiste à lancer sur une ou plusieurs machines physiques des machines virtualisées pour reproduire l'environnement souhaité et d'y exécuter une application réelle. Enfin, la simulation consiste à mettre en place un environnement logiciel entièrement dédié à l'expérimentation en se basant sur un modèle théorique.

Grid'5000 est une plate-forme d'expérimentation à échelle réelle, mais il est important de connaître les avantages et les inconvénients de ce système par rapport aux autres, mais également de savoir si parmi les solutions existantes d'échelle réelle il existe des différences. Afin de rendre la comparaison la plus pertinente possible, les solutions existantes les plus connues dans le monde scientifique ont été visées, c'est à dire en se basant sur une base de connaissances du sujet ainsi qu'un recensement des papiers concernant les "SDNs" [2] les plus répandus.

Concernant les critères de comparaison, certaines propriétés sont communes et citées dans la majorité des papiers de chaque solution ou du moins peuvent être conclues, soit via la documentation soit par une lecture plus fine.

Pour réaliser cette étude, quatre émulateurs, deux infrastructures à échelle réelle et un simulateur ont été étudiés (cf. Ligne « Type expérimentation » du tableau **??**). On peut penser qu'un seul simulateur est assez faible comme point de comparaison et non représentatif, mais comme précisé précédemment, les simulateurs deviennent de moins en moins utilisables avec l'évolution de la complexité des réseaux.

Afin de donner une vision globale des critères de comparaison et des solutions comparées, un tableau **??** a été préparé.

2. Software defined network

	ModelNet	Emulab	CloudLab	MiniNet	Maxinet	SSF	Distem
Noeuds virtuels	Oui	Oui/Non	Non	Oui	Oui	Non	Oui
Émulation réseau	Oui	Oui	Oui	Oui	Oui	Oui	Oui
Emulateur utilisé	DummyNet	DummyNet	DummyNet	tc [3] & HTB [4]	tc & HTB	**X**	tc
Précision [5]	M-H	H	H	F-M	M-H	F	M-H
Application native	Oui	Oui	Oui	Oui	Oui	Non	Oui
Scaling	Élevé	Limité à élevé	Limité	Moyen	Élevé	Très élevé	Élevé
Limite machine physique	Non	**X**	**X**	1	Non	1	Non
Type expérimentation	Émulé	Réel ou émulé	Réel	Émulé	Émulé	Simulé	Emulé
Description topologie	XML	NS	Rspec	Python+ CL [7]	Python+ CL	DML [6]	CL+REST +Ruby

Table 1 – Tableau comparatif des solutions de SDN

2.2 Comparaison

Dans l'objectif de comparer efficacement l'ensemble des solutions, les points de comparaison présents dans le tableau seront utilisés, couplés à des informations complémentaires plus précises, si elles sont pertinentes.

2.2.1 Utilisation de noeuds virtuels

La virtualisation est une méthode largement utilisée dans la plupart des infrastructures informatiques pour l'isolation de service afin d'améliorer la sécurité et/ou la facilité d'administration, pour allouer plus efficacement la puissance de calcul, etc, ... La virtualisation consiste à démarrer un ou plusieurs systèmes d'exploitation ou services sur un ordinateur, on pourra par exemple lancer plusieurs fois le même système d'exploitation dans des environnements virtuels différents.

3. Traffic Control : tc-netem
4. The hierarchical token bucket (HTB) is a faster replacement for the class-based queueing (CBQ) queuing discipline in Linux.
5. Faible, Moyenne, Haute
6. Domain Modeling Language
7. Command line

On remarque sur **??** que ModelNet, Distem, Mininet et Maxinet, soit les émulateurs, utilisent cette méthode (cf. Ligne « Noeuds virtuels »). En effet, la virtualisation va permettre à un émulateur de créer plusieurs noeuds à partir d'un seul et de créer un lien réseau entre ceux-ci. Évidemment le nombre de machine virtuel sur un noeud physique est limité par les capacités de calculs et de mémoire de cette dernière. Emulab apparaît dans le tableau comme permettant de faire de l'expérimentation sur noeud physique ou émulé, il permet en effet de créer un environnement virtualisé[**?**], permettant donc d'effectuer soit des test sur un environnement réel, soit virtualisé, au choix de l'utilisateur.

La limitation de mémoire et puissance de calcul présente sur un noeud physique a été contournée par certains émulateurs.

2.2.2 Limite du nombre de machines physiques

Émuler sur une seule machine peut rapidement être limitant pour de grosses émulations, de ce fait certains émulateurs ont mis en place la possibilité d'étendre l'émulation à plusieurs noeuds physiques (cf. Ligne « Limite machine physique »). Ici, le meilleur point de comparaison est entre Mininet et Maxinet, en effet, ce dernier est une version étendue[**?**] de Mininet. En répartissant la charge sur plusieurs noeuds, d'après le tableau, deux facteurs entrent en compte : la précision des résultats de l'expérience par rapport à la réalité ainsi que la scalabilité [8].

On remarque néanmoins que, bien que SSF ne soit limité qu'à une seule machine, sa scalabilité est très élevée. Celle-ci peut-elle être augmentée au détriment d'un autre facteur ?

2.2.3 Scalabilité et Précision

La scalabilité et la précision sont liées, mais ces facteurs ont le mérite d'être traités séparément, car une haute scalabilité n'implique pas une autre précision, et inversement.

L'exemple le plus probant ici est le simulateur, SSF, il permet une forte scalabilité, allant à plusieurs dizaines de milliers de noeuds[**?**], en revanche sa précision est faible. Cette caractéristique est principalement due au fait que notre réseau va être basé sur un modèle et non sur une application réelle. La scalabilité a également été étendue par la

8. Désigne la capacité d'un produit à s'adapter à un changement d'ordre de grandeur de la demande (montée en charge), en particulier sa capacité à maintenir ses fonctionnalités et ses performances en cas de forte demande. *Source : Wikipedia*

mise en place d'un système de dilatation temporelle, qui permet à l'application de faire ce qui devrait être fait en X secondes, dépendant du facteur de dilatation choisi. Il est à noter que la mise en place d'un système de dilatation temporelle est une idée évoquée pour Mininet [9], ce système permettra d'augmenter la scalabilité de Mininet tout en ayant une précision correcte.

On remarque que malgré l'émulation, la précision de Mininet est faible à moyenne, c'est assez relatif, et cette « notation » a été faite, car en cas de forte charge CPU, la précision de Mininet chute rapidement[**?**], ce qui peut rapidement arriver sur un système avec un seul noeud physique. De ce fait la précision de Maxinet est plus élevée, car, bien évidemment en relativisant au nombre de noeuds physiques utilisés, la charge maximale est plus difficile à atteindre. L'émulation de façon plus générale est plutôt précise, à condition que les instructions soient exécutées au bon moment, un manque de puissance de calcul peut rapidement fausser les résultats.

Concernant CloudLab , la précision est maximale puisque les applications sont directement exécutées sur des noeuds physiques, de ce fait les facteurs pouvant influencer sur l'expérimentation seront les mêmes que dans la réalité, c'est à dire saturation du lien réseau, saturation CPU, etc. Pour Emulab tout dépend de l'utilisation qui est faite, en environnement réel le comportement est le même que sur Cloudlab. En revanche on remarque que la scalabilité est limitée, c'est dû au fait que le nombre de machines physiques disponibles dans l'infrastructure est limité, par exemple sur CloudLab le nombre de noeuds est d'environ 500. Il est important de noter que sur un noeud physique il est possible de lancer des noeuds virtuels, Emulab par exemple permet de créer plusieurs noeuds virtuels en multiplexant sur un noeud physique. Dans ce cas on revient à des caractéristiques plus proches d'un émulateur.

Une précision élevée ne correspond pas pour autant à une situation réaliste. La figure **??** illustre bien une des raisons. En effet la latence entre un réseau local et un réseau distant est souvent bien plus élevée dans le second cas, ce qui implique que les résultats d'une expérience réalisée dans un cluster peuvent être biaisés en raison de ces très faibles latences.

9. `https://github.com/mininet/mininet/wiki/Ideas#virtual-time-via-time-dilation`

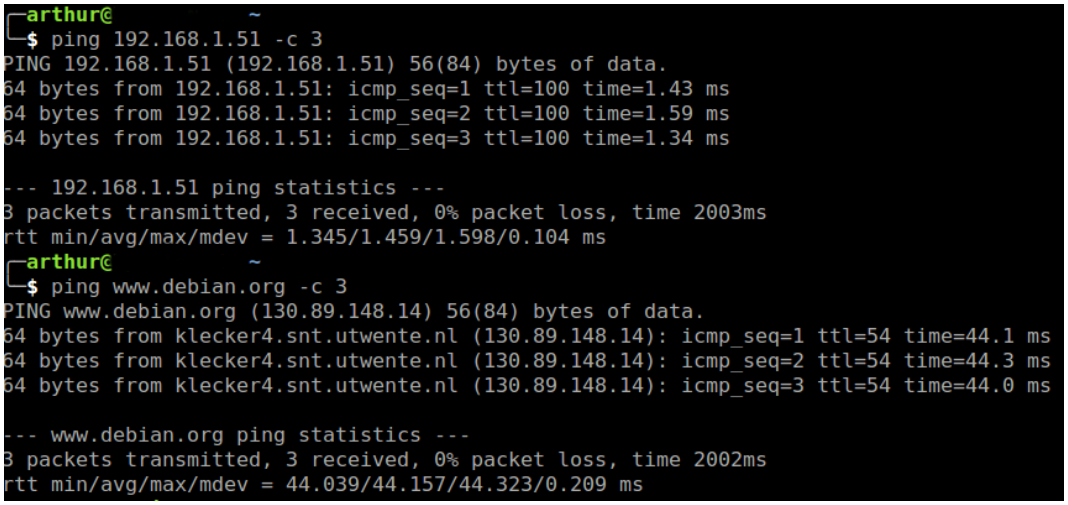


FIGURE 1 – Différence de latence entre un réseau local et distant

2.2.4 Créer une expérience plus réaliste grâce à l'émulation réseau

Que ce soit sur les émulateurs ou sur des infrastructures du type CloudLab, les expériences sont effectuées dans des lieux spécifiques, sur du matériel spécifique qui est loin de l'hétérogénéité d'Internet. Lorsqu'un paquet est envoyé à travers Internet, il est probable qu'il rencontre de la congestion, qu'il soit perdu ou que la latence soit très grande. Dans un environnement clos comme une émulation ou un cluster dans une salle machine, ce genre d'évènement est rare. Pour pallier à cette imprécision, toutes les solutions comparées embarquent un système d'émulation réseau.

Il existe deux types d'émulateurs réseau[?] :

— Les émulateurs réseau virtuels :

Ils ont pour objectif de créer une topologie réseau à partir d'une topologie donnée en entrée et d'émuler le comportement réseau et des systèmes entièrement sur une ou plusieurs machines.

— Les émulateurs de liens réseau :

Ils sont plus simples et s'installent comme un programme. Il suffit ensuite de leur donner des règles pour modifier leur comportement, comme modifier des paquets venant d'une interface, augmenter la latence ...

Dans le cas de ModelNet[?] par exemple, l'ensemble des noeuds virtuels sont déployés sur une ou plusieurs machines et une partie interne à ModelNet se charge du routage,

de la latence, des pertes de paquets, etc. En revanche, dans le cas d'une infrastructure telle que CloudLab ou Grid'5000 il sera compliqué de reproduire ce réseau en dehors d'un émulateur. De ce fait, une autre solution a dû être envisagé, il s'agit de la deuxième citée ci-dessus : Les émulateurs de liens réseaux.

Distem est basé sur la même idée que ModelNet, mais il va plus loin dans l'émulation puisqu'il propose à partir d'un cluster, c'est-à-dire un ensemble de noeuds homogènes, d'émuler plusieurs noeuds hétérogènes[**?**]. C'est-à-dire qu'à partir d'un noeud physique avec une puissance de calcul donnée, il peut créer un ou plusieurs noeuds virtuels avec une puissance de calcul inférieure ou égale à celle du noeud physique. Ce qui se rapprochera plus des serveurs rencontrés sur Internet.

L'ensemble des solutions proposées permettent de réaliser des topologies réseau, la question de la portabilité entre celles-ci se pose donc.

2.2.5 Description des topologies

Chaque système se base sur une entrée pour savoir comment instancier la topologie souhaitée par l'utilisateur (cf. Ligne « Description topologie »). L'idée d'un système de description unique semble être souhaitable pour les utilisateurs envisageant de changer de système d'expérimentation, malheureusement comme le tableau **??** le montre, il semble que chaque solution adopte son propre système.

Sans explication vraiment justifiable, l'on peut imaginer que chaque solution à voulu tendre vers un langage plus complet, d'autres vers quelque chose de plus simple, d'autre plus évolutif, etc. De ce fait, hormis MaxiNet et MiniNet, dont le premier dépend de l'autre, qui ont le même système d'interaction avec l'utilisateur (Une API python, et des commandes), toutes les autres solutions sont différentes. On remarque tout de même que deux solutions sont présentes, l'une permet d'instancier la topologie à partir d'un fichier statique (XML, NS, Rspec, DML) et l'autre solution consiste à interagir directement avec le système au moyen de ligne de commande, ou d'une API (Python, REST et Ruby).

Devant cette constatation, il est préférable pour les utilisateurs d'avoir pour notre solution une syntaxe proche, voire identique à l'une des solutions présentées.

3 Solution proposée pour Grid'5000

L'ensemble des solutions proposées donne une vue globale de ce qui peut être fait, soit en émulant, soit en simulant soit en utilisant une infrastructure réelle. En fonction des objectifs que chaque solution cherche à atteindre, l'on peut déduire ce qu'attendent les utilisateurs pour se rapprocher d'une solution « idéale ».

L'objectif présenté dans ce sujet de recherche est double, le premier est de valider que notre outil permettant de gérer des VLANs, KaVLAN, permet d'instancier des topologies réseau sur des noeuds à plusieurs interfaces. Le second est qu'un outil permettant d'instancier des topologies réseau sur Grid'5000 de façon simple soit proposé aux utilisateurs de Grid'5000.

« Simple » est assez réducteur et subjectif, pour Grid'5000 c'est d'abord rendre l'apprentissage de l'outil rapide pour les nouveaux utilisateurs, mais également pour les utilisateurs venant d'autres plates-formes. De ce fait le choix de la façon d'instancier une topologie doit être le plus pertinent possible.

3.1 Choix du format de description de la topologie

Afin de décrire la topologie pour notre outil, nous avions le choix de proposer un nouveau langage ou de nous inspirer d'un déjà existant utilisé dans le même contexte. La première solution ne s'accorde pas avec la vision proposée ci-dessus, qui consiste à faciliter l'adaptation de l'outil par les utilisateurs déjà existants de SDN. L'utilisation d'une syntaxe déjà existante étant de mise, il faut par conséquent choisir entre toutes.

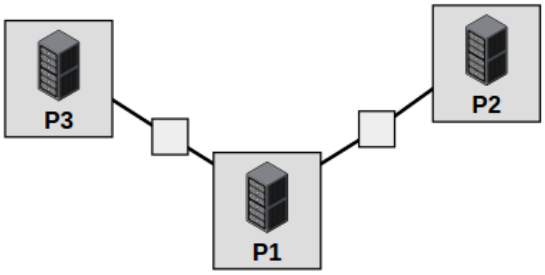
Tout d'abord, l'utilisation de DML ne semble pas appropriée, en effet destiné à un simulateur, il recense beaucoup de paramètres non applicables dans le cadre d'une expérimentation sur des noeuds physiques, ce qui le rend peu intuitif. Vient ensuite l'utilisation d'un API ou d'un fichier statique. Le problème de l'API Python ou Ruby est qu'elle impose l'installation de l'interpréteur pour l'utilisateur, et de plus demande des bases dans l'un ou l'autre langage ce qui peut rebuter certains utilisateurs. Finalement, le choix se portera vers des fichiers de descriptions statiques.

Parmi ceux-ci, il reste XML, NS et Rspec. NS a été le langage de prédilection pendant plusieurs années sur Emulab, mais il s'est révélé assez difficile à aborder et est de plus en plus remplacé par Rspec. CloudLab étant implanté sur le modèle d'Emulab (ils utilisent

la même infrastructure), il utilise Rspec également, qui se veut plus simple d'utilisation. C'est en réalité une syntaxe basée sur XML définissant une représentation structurée des ressources. Ainsi, l'utilisateur écrit en XML un fichier décrivant les ressources dont il a besoin. L'avantage du XML est qu'il reste facile à lire, la structuration est claire au premier coup d'oeil, comme le montre le code ci-dessous :

```
  <?xml version="1.0" encoding="UTF-8"?>
<rspec xmlns="http://www.geni.net/resources/rspec/3">
  <node client_id="P1">
    <interface client_id="dhcp_1"/>
    <interface client_id="interface-P1-1">
      <ip address="192.168.1.1" type="ipv4" netmask="255.255.255.0" />
    </interface>
  </node>
  <node client_id="P2">
    <interface client_id="interface-P2-1">
      <ip address="192.168.1.2" type="ipv4" netmask="255.255.255.0" />
    </interface>
  </node>
  <node client_id="P3">
      <interface client_id="interface-P3-1"/>
  </node>
  <link client_id="link-0">
  <interface_ref client_id="dhcp_1" />
  <interface_ref client_id="interface-P3-1" />
  </link>
  <link client_id="link-1">
    <interface_ref client_id="interface-P1-1" />
    <interface_ref client_id="interface-P2-2" />
  </link>
</rspec>
```

Inspiré de cette topologie :



Un argument supplémentaire en faveur de Rspec est qu'Emulab et Cloudlab sont des infrastructures répandues dans le monde scientifique, en particulier HPC, ce qui fait que ce langage est susceptible d'être utilisé par une vaste communauté de chercheurs. De ce

fait nous avons choisi Rspec comme langage de définition de nos topologies. Notre objectif est de rester compatibles avec CloudLab tout en se réservant la possibilité d'étendre le langage pour ajouter des fonctionnalités potentiellement demandées par les utilisateurs de Grid'5000.

3.2 Choix concernant la scalabilité

Grid'5000 est une plate-forme mettant à disposition environ mille noeuds physiques, chaque noeud pouvant être réservé par un utilisateur avec un accès « bare metal », c'est-à-dire un accès total au noeud de calcul, du choix de l'OS au hardware et au réseau. De ce fait, si l'utilisateur n'a pas accès à assez de noeuds, il lui est possible de déployer Distem sur plusieurs de ces noeuds et donc déployer un environnement émulé avec des milliers de noeuds. En effet, Distem est développé en utilisant l'infrastructure de Grid'5000, de ce fait son déploiement sur l'infrastructure est documenté [10].

Grid'5000 dispose de nombreux noeuds de calculs et très diversifiés [11], que ce soit en puissance de calcul, en RAM ou en stockage, certains proposant même des unités de calculs graphiques dédiés, ce qui peut sur certaines applications changer considérablement le temps de calcul par rapport à un CPU et donc rendre très hétérogène les temps de réponse de chaque noeud si besoin.

L'infrastructure est également basée sur un réseau haute performance composé d'Ethernet 10G et d'un réseau Infiniband, ce qui, si l'on veut se rapprocher d'un réseau Internet est un peu trop idéal étant donné les faibles latences que ces réseaux impliquent.

3.3 Choix de la solution d'émulation réseau

Comme vu précédemment, l'émulation réseau permet de générer des perturbations sur le réseau. L'utilisation d'un système non virtualisé impose l'utilisation d'un émulateur de lien réseau, de ce fait nous utiliserons l'étude qui a été faite sur trois d'entre eux[**?**]. Il est d'autant plus indispensable de proposer cette fonctionnalité que toutes les autres solutions proposent.

Le principe de ces émulateurs est de capturer des paquets entrants ou sortants et d'y appliquer une règle. Ces règles ressemblent aux règles de configurations d'un pare-feu

10. `http://distem.gforge.inria.fr/tutorial.html`
11. `https://www.grid5000.fr/mediawiki/index.php/Special:G5KHardware`

(exemple de Dummynet) :

```
ipfw pipe 1 config bw 100Kbit/s
```

Cette règle dit simplement que la bande passante (Bandwidth abrégée bw) du tuyau 1 (pipe 1) est au maximum de 100Kbit/s. Un tuyau est défini par un flux de données entre deux adresses IPs, on peut dire qu'il existe un tuyau entre 192.168.1.1 et 192.168.1.2 et lui appliquer la règle ci-dessus, et elle ne sera appliquée uniquement si un paquet vient de l'une de ces deux adresses vers l'autre.

Concernant le choix entre TC, NISTNet et Dummynet, nous nous sommes orientés vers Dummynet bien qu'il soit moins fiable sur les liaisons haute vitesse. En effet, sur un lien à 1Gbps, le débit tombe à 900Mbps, ceci peut avoir des conséquences sur certaines expérimentations, mais DummyNet est le seul à permettre la capture de paquets entrants **et** sortants : dans de nombreux cas l'utilisation, deux points d'interception peuvent être utiles (comme l'émulation sur un système où l'application est elle-même lancée).

Il est également à noter que si les utilisateurs ont besoin de performances réseau, ils peuvent toujours mettre en place eux même une des deux autres solutions.

4 Implémentation de la solution d'instanciation

4.1 Fonctionnement de KaVLAN

Une bonne compréhension du fonctionnement de KaVLAN est nécessaire au bon développement de l'outil d'instanciation.

Le fonctionnement de Grid'5000 repose sur l'allocation des ressources qui se déroule via des réservations faites par les utilisateurs. L'utilisateur a à sa disposition, un ensemble de ressources qui sont par exemple des noeuds, ou des VLANs. Une fois réservés il est libre d'en faire ce qu'il veut le temps de la réservation. KaVLAN intervient directement sur les switchs via des commandes SNMP [12] ou en se connectant en SSH sur ceux-ci, afin de modifier le VLAN d'une interface d'un noeud. Avant d'effectuer cette action, il vérifie que l'interface à modifier correspond à un noeud qui est réservé par l'utilisateur, et que le VLAN de destination est également réservé par l'utilisateur.

Il existe actuellement 3 types de VLANs sur Grid'5000 :

12. Simple Network Management Protocol

— Locaux : Il y en a 3 par site, et possèdent une passerelle entre le VLAN de production (Vlan par défaut) et le VLAN local.
— Routés : Il y en a 6 par site et permettent de créer un routage vers les noeuds dans ceux-ci afin d'y accéder depuis n'importe quel endroit dans Grid'5000 sans passerelle.
— Globaux : Il y en a 10 au total, ils permettent de créer des VLANs d'interconnexion entre les sites de Grid'5000 en utilisant QnQ.

Bien que ces VLANs soient pratiques pour de petites expérimentations, il parait évident que pour une topologie complexe il n'y en aura pas assez. Nous avons estimé le besoin à 400 VLANs par site pour les plus grosses topologies, mais la mise en place d'autant de VLANs avec les types disponibles actuellement est soit non envisageable (manque d'IP par exemple pour les VLANs routés) soit trop contraignante (mise en place de 400 passerelles pour les VLANs locaux). De ce fait nous avons dû mettre en place un nouveau type dédié aux topologies, des VLANs simples, permettant d'être mis en place rapidement, sans serveur DHCP, ni passerelle. Avec cette configuration il faut faire attention à laisser un lien entre le VLAN de production et le VLAN de l'utilisateur, deux possibilités se présentent :

— L'utilisateur prévoit un noeud passerelle entre le VLAN de production et un de ses VLANs dans sa topologie.
— L'outil d'instanciation permet de créer une passerelle dynamique entre le VLAN de production et les autres. C'est-à-dire que l'utilisateur peut exécuter une commande pour changer à la volée le VLAN de la passerelle et accéder à l'un des noeuds dans celle-ci. En théorie c'est facilement faisable avec KaVLAN, mais l'IP de l'interface dans le VLAN de l'utilisateur doit correspondre au sous-réseau des noeuds à atteindre, ce qui demande un peu de configuration à chaque fois.

Une fois le fonctionnement de KaVLAN assimilé et les nouveaux VLANs mis en place, il est possible de commencer le développement.

4.2 Développement de topo_maker

La majorité des outils étant codés en Ruby sur Grid'5000, et les nombreux gems [13] à disposition ont orienté le développement vers ce langage.

L'idée proposée dans le sujet du projet était de s'orienter vers un code jetable, uni-

13. Librairies proposées pour Ruby

quement pour tester la mise en place de topologies sur des noeuds à 4 interfaces, les spécifications ayant évolué dans le temps le développement s'est plus orienté vers un code structuré permettant une évolution rapide si de nouvelles fonctionnalités étaient nécessaires. Le code se divise en trois parties majeures, le Parser permettant d'analyser le fichier Rspec donné en entrée par l'utilisateur et d'en retourner les parties nécessaires de façon structurée. La seconde partie est le modèle il représente sous forme de classe chaque partie nécessaire à la mise en place d'une topologie réseau : Les VLANs, les noeuds et leurs interfaces. La dernière partie étant un Contrôleur permettant de créer les objets du modèle avec les informations retournées par le Parser.

Il a été décidé que ce n'était pas le rôle de topo_maker de se charger des réservations des noeuds et des VLANs, de ce fait il vérifie que les ressources de la réservation donnée en paramètre (sous forme de numéro de réservation) que les ressources nécessaires dans la topologie réseau correspondent au moins aux ressources disponibles dans la réservation et si ce n'est pas le cas il avertit l'utilisateur des ressources nécessaires.

Une fois l'ensemble des données récupérées et structurées dans le modèle de l'application, celle-ci exécute dans l'ordre le déploiement des OS sur les noeuds, la configuration IP puis la mise en place dans le VLAN. Enfin il ressort un fichier YAML permettant d'avoir une vue globale du système (IP des noeuds, quels VLANs ont été utilisés pour quelles interfaces, ...).

4.3 Fonctionnalités manquantes à l'utilisation

Lors des premiers tests, l'absence de certaines fonctionnalités se ressentait et pouvait être gênante.

La première est que depuis la version Jessie de Debian, le serveur sshd n'autorise plus la connexion sur le compte root des noeuds à l'aide du mot de passe. L'outil de déploiement présent sur Grid'5000 ajoute la clé RSA publique de l'utilisateur dans les clés autorisées [14], mais lors de l'utilisation de topologie complexe il n'est pas rare de passer de noeud en noeud, de ce fait la clé privée n'est pas transférée et il est obligatoire d'utiliser le mot de passe, ce n'est pas possible.

Pour palier à ce premier problème topo_maker se charge de générer une paire de clés

14. Les clés RSA permettent de se connecter à un serveur SSH sans mot de passe en étant authentifié par la clé privée associée.

RSA et ajoute sur chaque noeud réservé la clé privée et autorise la clé publique associée. Il en résulte que la clé privée est disponible sur chaque noeud pour pouvoir passer à un autre noeud.

Le second problème est qu'une fois dans un VLAN, un noeud n'a plus d'accès à Internet, il est alors impossible d'installer de nouveaux paquets sans repasser au moins une interface dans le VLAN de production. Pour contourner ce problème, nous avons étendu le langage Rspec pour ajouter un champ aux utilisateurs, leur permettant de préciser les paquets qu'ils souhaitent installer sur chaque noeud.

4.4 Évolutions prévues

Dans l'état actuel, il est possible pour l'utilisateur de réaliser des topologies réseau avancées, mais plusieurs points évoqués précédemment ne sont pas encore implémentés et peuvent freiner l'adaptation de l'outil.

Une première évolution envisagée est la mise en place de la passerelle dynamique proposée dans la partie **??**. En effet la mise en place de cette fonctionnalité permettra aux utilisateurs de ne pas « perdre » une interface, jusqu'alors uniquement utilisée pour acceder à un VLAN particulier.

La seconde est l'intégration de l'émulation réseau, c'est un point important dans la réalisation de topologies complexes afin d'apporter des caractéristiques spécifiques à des liens réseau. FreeBSD n'étant pas un « Linux », mais un système « BSD », son fonctionnement est un peu différent des OS que l'on met à disposition sur Grid'5000 (Debian, Ubuntu, CentOS, ...), et il l'image doit être adaptée au matériel sur lequel il démarre. Une étape d'adaptation d'une image FreeBSD pour les nouveaux noeuds à quatre interfaces du site de Nancy est donc prévue pour ensuite pouvoir inclure ce processus dans Topo_maker et donc leur faire profiter de l'émulation réseau.

5 Conclusion et perspectives

	ModelNet	Emulab	CloudLab	MiniNet	Maxinet	Distem	topoMaker
Noeuds virtuels	Oui	Oui/Non	Non	Oui	Oui	Oui	Non
Émulation réseau	Oui	Oui	Oui	Oui	Oui	Oui	Bientôt
Emulateur utilisé	DummyNet	DummyNet	DummyNet	tc & HTB	tc & HTB	tc	DummyNet
Précision	M-H	H	H	F-M	M-H	M-H	H
Application native	Oui	Oui	Oui	Oui	Oui	Oui	H
Scaling	Élevé	Limité à élevé	Limité	Moyen	Élevé	Élevé	Limité 1000 nœuds
Limite machine physique	Non	**X**	**X**	1	Non	Non	**X**
Type expérimentation	Émulé	Réel ou émulé	Réel	Émulé	Émulé	Emulé	Réel
Description topologie	XML	NS	Rspec	Python+ CL	Python+ CL	CL+REST +Ruby	Rspec

Table 2 – Tableau comparatif des solutions de SDN comprenant notre solution

Dans ce travail nous avons premièrement étudié l'existant en se basant différents papiers afin d'identifier les forces et les faiblesses de chaque solution. Les systèmes comparés proposent tous des solutions afin d'avoir les meilleurs résultats dans un ou plusieurs des critères étudiés. Malheureusement il ne semble pas - encore - possible de rendre tous ces critères optimaux. L'émulation a permis d'améliorer grandement la précision des résultats face à la simulation, au détriment de la scalabilité. Cette dernière tend pourtant à largement augmenter avec la puissance de calcul et la mémoire vive des noeuds de calculs d'aujourd'hui, au point que certaines solutions comme Distem visent une émulation de 100,000 noeuds dans un futur très proche.

Nous avons ensuite réalisé une étude des choix technologiques à appliquer sur notre infrastructure dans l'objectif de rendre l'expérience de l'utilisateur la plus satisfaisante possible. La solution proposée sur Grid'5000 est proche de ce qui est proposé sur d'autres infrastructures comme CloudLab, permettant de faire des expérimentations sur des noeuds physiques. La scalabilité de notre solution bien que limitée reste assez élevée grâce au nombre de noeuds disponible sur l'infrastructure. L'ensemble de cette étude a permis de combler un manque de Grid'5000 par rapport aux solutions existantes : l'instanciation de topologies réseau complexes. Jusque là possible, mais de manière très limitée par le

nombre de VLANs disponibles et le temps de configuration de chaque noeud à la main afin de la crée, elle est désormais possible à l'aide d'une description au format XML très semblable à celle proposée sur CloudLab.

L'ensemble du travail réalisé nous a permis d'obtenir solution comparable aux autres présentées dans le tableau **??**.

Dans l'avenir nous prévoyons de rendre cette solution la plus satisfaisante possible. Dans un premier temps la priorité sera d'implémenter des noeuds DummyNet permettant d'effectuer de l'émulation réseau. Et dans un second temps, prendre en compte les retours utilisateurs afin d'améliorer l'expérience de l'utilisateur.