# Passage à l'échelle de l'infrastructure réseau d'une solution de virtualisation avec VXLAN

**Benjamin Collet**
Université de Strasbourg — Direction informatique
14, rue René Descartes
67 084 Strasbourg

**Christophe Palanché**
Université de Strasbourg — Direction informatique
14, rue René Descartes
67 084 Strasbourg

## Résumé

*La Direction informatique de l'Université de Strasbourg héberge plus de 600 systèmes répartis sur une centaine d'hyperviseurs* KVM. *L'historique et les contraintes imposées par les applicatifs hébergés sur les machines virtuelles ont entraîné un éclatement sur une centaine de réseaux, dont la plupart sont sécurisés par des pare-feux fonctionnant sur le système OpenBSD.*

*Pour faciliter la migration des machines virtuelles d'un hyperviseur à l'autre, ces réseaux doivent être provisionnés sur l'ensemble des équipements des salles machines et des hyperviseurs. Nous sommes arrivés aux limites de cette solution qui, bien que fonctionnelle, génère une charge considérable sur les différents équipements et une complexité de configuration des hyperviseurs.*

*Dans un premier temps, nous présentons les différentes solutions étudiées, ainsi que les raisons qui nous ont amenés à retenir* VXLAN *pour résoudre les problématiques auxquelles nous étions confrontés. Dans un deuxième temps, nous expliquons les principes de fonctionnement du protocole et la démarche de mise en œuvre dans notre environnement actuel de virtualisation, ainsi que dans la plateforme de* cloud computing *en cours de déploiement. Par ailleurs, nous détaillons l'impact sur les infrastructures réseaux.*

*Pour finir, nous évoquons les évolutions possibles de cette architecture dans le cadre de l'arrivée de notre datacenter et l'émergence de nouvelles technologies telles qu'*Ethernet VPN.

## Mots clefs

*VXLAN, réseau, virtualisation, Linux, OpenBSD*

## 1 Introduction

La Direction informatique de l'Université de Strasbourg héberge plus de 600 systèmes répartis sur une centaine d'hyperviseurs KVM. L'historique et les contraintes imposées par les applicatifs hébergés sur les machines virtuelles ont entraîné un éclatement sur une centaine de réseaux, dont la plupart sont sécurisés par des pare-feux fonctionnant sur le système OpenBSD. Bien que la rationalisation des plateformes matérielles ait été effectuée, celles des architectures réseaux pour les applicatifs n'était pas suffisante pour limiter cet éclatement et permettre sa mise en œuvre difficile dans de courts délais.

L'infrastructure réseau au sein des salles machines gérées par la Direction informatique est composée de 32 commutateurs placés en *Top of Rack* (placés en haut de l'armoire), regroupés en cinq piles, ou stacks. Les réseaux transportés par ces commutateurs sont remontés en niveau deux exclusivement jusqu'aux différents pare-feux. Afin de réduire et faciliter le paramétrage et l'exploitation, et permettre simplement la migration à chaud des machines virtuelles, tous les *VLAN* utilisés sont configurés sur les ports connectés à des hyperviseurs, ainsi que sur ces derniers.

Ces choix techniques nous avaient permis de limiter la charge d'exploitation ; cependant avec l'explosion de l'usage de la virtualisation cette solution a commencé à atteindre ses limites. Tout d'abord, sur le plan technique, nous avons constaté une explosion des tables de commutation, principalement liée à une multiplication des *VLAN* instanciés sur les équipements et les hyperviseurs. Ensuite, sur le plan humain, ce mode opératoire implique une dépendance forte sur l'équipe réseau dès qu'il s'agit de rajouter une machine virtuelle sur un nouveau réseau, car cela implique des modifications de configuration sur l'ensemble des équipements sur lesquels sont connectés des hyperviseurs. Enfin, le nombre limité de *VLAN* (4096) et la charge actuelle des équipements ne nous laisse aucune marge de manœuvre pour l'ouverture future d'un service de cloud privé à l'échelle des composantes et laboratoires de l'Université de Strasbourg.

# 2 Solutions étudiées

## 2.1 Configuration et nettoyage manuels à la demande

La première solution étudiée pour pallier les problèmes précités consiste à nettoyer les configurations actuelles pour ne laisser sur chaque port des commutateurs que les *VLAN* effectivement utilisés par l'hyperviseur qui y est connecté. Cette solution implique qu'à chaque opération de création, de déplacement ou de suppression de machine virtuelle, une modification de la configuration des équipements réseau et des hyperviseurs ait lieu. Cette modification de configuration comprend également la remontée des *VLAN* sur l'ensemble des équipements traversés depuis le port sur lequel est connecté l'hyperviseur jusqu'au pare-feu.

Bien que corrigeant en partie les problèmes de charge sur le matériel, étant donné l'ampleur du parc et les objectifs initiaux, cette solution semble peu réaliste tant au niveau de la pression induite sur les équipes d'exploitation système et réseau, même lors de modifications triviales sur une machine virtuelle, qu'au niveau des risques d'erreurs et d'oublis accrus. Ces deux points ont pour conséquence prévisible l'allongement des temps d'intervention.

## 2.2 *Multiple VLAN Registration Protocol* et automatisation

La deuxième solution étudiée reprend les principes de la solution évoquée précédemment, mais vise à éliminer le problème de la configuration manuelle de la remontée des *VLAN* sur l'ensemble des équipements traversés grâce à l'utilisation de *MVRP* (*Multiple VLAN Registration Protocol*), une application de la norme IEEE 802.1ak (*Multiple Registration Protocol*)[1]. *MVRP* permet d'éviter de configurer l'ensemble des équipements traversés ; la configuration des ports connectés à l'hyperviseur et au pare-feu est suffisante, les *VLAN* nécessaires étant instanciés de proche en proche par les équipements intermédiaires. De plus, nous avons étudié l'automatisation de la configuration de ces ports et du paramétrage réseau des hyperviseurs. Pour ces derniers, nous avons déployé il y a plusieurs années le gestionnaire de configuration *Chef*[2], nous permettant ainsi d'automatiser cette partie. Les équipements en production au sein de notre réseau implémentent différentes méthodes de configuration à distance : *Chef*, *Puppet*[3] et *NETCONF*[4].

Cependant, cette solution présente quelques inconvénients ; tout d'abord la question de la pérennité en cas de changement d'équipements réseau. En effet, tous les constructeurs n'implémentent pas *MVRP* et lui préfèrent parfois sur certaines gammes des protocoles propriétaires. Il en va de même pour les méthodes de

configuration à distance. De plus, ici aussi, cette solution ne corrige que partiellement le problème de charge sur les équipements : le nombre de *VLAN* instanciés reste important, bien que ceux inutiles soient automatiquement retirés par *MVRP*. Enfin, cela implique que les processus du cycle de vie des machines virtuelles entraînent automatiquement des modifications de configuration sur les équipements réseau, changeant les limites de responsabilité entre les équipes systèmes et réseaux.

### 2.3 *Virtual Extensible LAN*

Enfin, la dernière solution étudiée est *VXLAN* (*Virtual Extensible LAN*), récemment documenté par l'IETF dans la RFC 7348[5]. *VXLAN* est une technologie de réseau qui superpose un ou plusieurs réseaux sur un réseau existant (*Overlay Network*), masquant la complexité et les contraintes du réseau sous-jacent au réseau transporté. Cela permet d'éliminer l'usage des *VLAN* pour la séparation des flux des machines virtuelles et ne conserver qu'un réseau transportant *VXLAN*. Les équipements réseau n'ont donc plus connaissance que des points d'entrée-sortie et non plus de l'intégralité des adresses des systèmes communiquant au sein des différents *VLAN*. La configuration réseau des équipements n'est donc faite qu'une fois et n'est plus vouée à évoluer lors de l'ajout de réseau de virtualisation ou du déplacement d'une machine virtuelle.

Cette solution implique cependant l'existence d'une implémentation *VXLAN* sur les hyperviseurs et sur les équipements servant de passerelle pour les réseaux transportés. De plus, afin d'éviter de surcharger les différents hyperviseurs, l'instanciation d'un réseau *VXLAN* depuis un hyperviseur doit se faire uniquement lorsqu'une machine virtuelle appartenant à ce réseau est en fonctionnement.

## 3 Solution retenue : *VXLAN*

### 3.1 Principes de fonctionnement

Comme nous l'avons vu précédemment, *VXLAN* est une technologie d'*Overlay Network*. Il permet de transporter et de séparer différents réseaux en encapsulant les flux de couche 2 d'une machine virtuelle, ou d'un serveur, dans des paquets UDP [1] (couche 4), en rajoutant un en-tête permettant de discriminer les différents réseaux transportés grâce à un identifiant *VXLAN* (*VXLAN Network Identifier*, *VNI*) sur 24 bits :

```
VXLAN Header:

 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|R|R|R|R|I|R|R|R|            Reserved                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                VXLAN Network Identifier (VNI) |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Cette encapsulation est faite à partir de l'hyperviseur jusqu'à la passerelle *VXLAN*. À chaque extrémité, ou *Virtual Tunnel End-Point* (*VTEP*) est maintenue une table de correspondance comprenant l'adresse MAC des machines utilisant *VXLAN* pour communiquer, l'adresse IP du *VTEP* correspondant [2], et le *VNI*.

Les machines virtuelles n'ont aucunement conscience que leur trafic est transporté sur un *Overlay Network*, l'encapsulation et la désencapsulation se faisant au niveau des hyperviseurs ou de la passerelle *VXLAN* [3] :

1. Le port 4789 a été assigné par l'IANA, cependant certaines implémentations historiques utilisent le port non standard 8472.
2. Dans notre cas, un hyperviseur hébergeant une machine virtuelle ou l'adresse de la passerelle VXLAN vers le reste du réseau.
3. Il est important de noter que pour que les machines virtuelles puissent utiliser une taille maximale de paquets (*Maximum Transmission Unit*, *MTU*) standard de 1500 octets, il faut augmenter la *MTU* de l'interface de transport (`eth0` dans la figure 1) pour

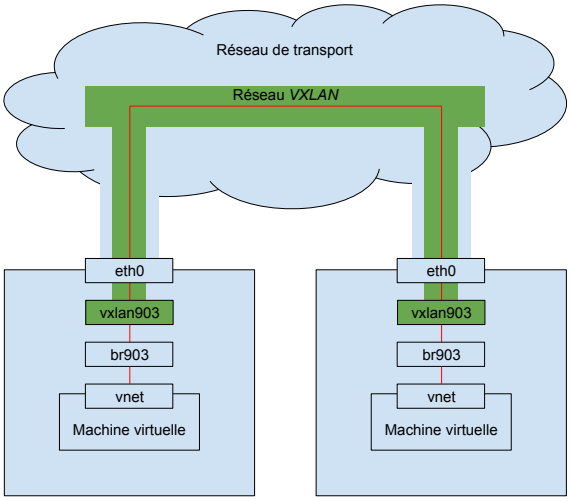


*Figure 1 - Encapsulation* VXLAN *entre deux machines virtuelles*

Afin de permettre la transmission des flux *broadcast*, *unknown unicast* et *multicast* (*BUM*), notamment pour permettre le remplissage de la table de correspondance, à chaque *VNI* est associé un groupe *multicast*[4], qui peut être commun à plusieurs *VNI*.

Le peuplement de la table de correspondance s'effectue lorsqu'un *VTEP* reçoit un paquet en provenance d'une source qui ne lui est pas encore connue.

## 3.2 Intégration des hyperviseurs

### 3.2.1 Avec l'infrastructure actuelle (*libvirt* sous Linux)

La configuration de *VXLAN* sous Linux nécessite une version du noyau supérieure à 3.7-rc3, avec l'option `CONFIG_VXLAN` activée à la compilation. Les informations suivantes sont nécessaires à la création de l'interface *VXLAN* :

— l'identifiant du *VNI* ;

— le groupe multicast utilisé pour le trafic *BUM* ;

— l'interface physique de sortie vers le réseau *VXLAN*.

Par exemple :

```
# ip link add vxlan903 type vxlan id 903 group 239.0.3.135 dev eth0
# ip link set up dev vxlan903
# ip link show vxlan903
11: vxlan903: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc noqueue
state UNKNOWN mode DEFAULT group default
    link/ether 4e:9d:48:79:82:84 brd ff:ff:ff:ff:ff:ff
```

prendre en compte l'ajout de l'en-tête *VXLAN*, qui a une taille d'au moins 50 octets (20 octets d'en-tête IP, 8 octets d'en-tête UDP, 8 octets d'en-tête *VXLAN* et 14 octets d'en-tête Ethernet de la trame originale).

4. Il est possible de se passer entièrement de *multicast*, mais cela nécessite de provisionner à l'avance l'ensemble des correspondances. Dans ce cas les *VTEP* dupliquent le trafic de *broadcast* et l'envoient en *unicast* vers chacune des destinations.

Il possible de consulter l'état de la table de correspondance mentionnée précédemment grâce à la commande suivante :

```
# bridge fdb show dev vxlan903
00:00:00:00:00:00 dst 239.0.3.135 via eth0 self permanent
```

On voit ici qu'aucune autre adresse n'est connue dans ce réseau *VXLAN*, toutes les communications seront donc diffusées au groupe *multicast* `239.0.3.135` depuis l'interface `eth0`.

Afin de pouvoir utiliser ce réseau *VXLAN*, il est maintenant nécessaire de créer un pont sur lequel seront attachées l'interface `vxlan903` et les interfaces des machines virtuelles :

```
# ip link add name br903 type bridge
# ip link set dev vxlan903 master br903
# ip link set up dev br903
```

L'interface de programmation (*Application Programmable Interface*, *API*) pour la virtualisation, *libvirt*[6], permet d'attacher automatiquement les interfaces d'une machine virtuelle dans des ponts définis par son fichier de configuration, cependant elle ne prend pas en compte la création et le paramétrage de ces ponts.

La *libvirt* met en revanche à disposition des administrateurs et développeurs plusieurs mécanismes de pré-traitement et de post-traitement pouvant faire appel à des programmes ou des scripts tiers[7], et ce à différents instants du cycle de vie d'une machine virtuelle :

| **Étape** | **Description** |
|---|---|
| *prepare* | Avant toute allocation de ressource à la machine virtuelle |
| *start* | Après l'allocation des ressources et avant le démarrage |
| *started* | Après le démarrage |
| *stopped* | Après l'arrêt de la machine virtuelle, avant la libération des ressources |
| *release* | Après la libération des ressources |

Les étapes qui nous intéressent particulièrement sont *prepare* et *release*, qui interviennent respectivement avant l'allocation des ressources et après leur libération. En effet, la *libvirt* a besoin que les toutes ressources, notamment les ponts, existent avant de pouvoir les allouer à la machine virtuelle.

Les scripts sont appelés par la *libvirt*, et l'intégralité de la configuration de la machine virtuelle est transmise par l'entrée standard, nous permettant ainsi d'effectuer les actions sur les ponts en fonction de la configuration de la machine :

1. lors de l'étape *prepare*, avant le démarrage, si le ou les ponts utilisés par la machine virtuelle n'existent pas, ceux-ci sont créés, ainsi que les interfaces *VXLAN* qui en font partie ;
2. lors de l'étape *release*, après la désallocation des ressources, si le ou les ponts ne sont plus utilisés par aucune autre machine virtuelle, ceux-ci ainsi que les interfaces *VXLAN* sont supprimés ;
3. lors d'une migration à chaud d'une machine virtuelle, la *libvirt* passe par les mêmes étapes et appelle donc les mêmes scripts.

### 3.2.2 Avec l'infrastructure future (*OpenNebula*)

Dans le cadre de la rationalisation de nos infrastructures de virtualisation, nous avons souhaité nous doter d'outils permettant de mieux gérer notre parc d'hyperviseurs et l'allocation des ressources aux machines virtuelles. La solution que nous avons retenue, *OpenNebula*[8], supporte nativement *VXLAN* ; cependant

une limitation imposait l'utilisation sur un hyperviseur d'une seule méthode de mise en réseau des machines virtuelles : *VLAN* ou *VXLAN*. Nous avons donc effectué une modification et l'avons remontée aux développeurs, qui l'intégreront dans une version future [5].

## 3.3 Intégration réseau

### 3.3.1 Réseaux filtrés avec OpenBSD

Les pare-feux qui hébergent nos réseaux filtrés sont basés sur OpenBSD[9] et *Packet Filter*[10]. Le support de VXLAN existant depuis la version 5.5 d'OpenBSD, l'intégration de ces réseaux se fait de façon native.

Pour configurer l'interface en mode dynamique, ce qui permet d'envoyer le trafic *unicast* directement à l'adresse du *VTEP* distant et d'utiliser le groupe *multicast* uniquement pour le trafic *broadcast* et *multicast*, elle doit être intégrée à un pont.

Afin d'éviter de diffuser les trafics *broadcast* et *multicast* de tous les réseaux sur tous les *VTEP* nous avons fait le choix d'utiliser un groupe *multicast* différent par *VXLAN*. L'adresse du groupe *multicast* est composée de `0xEF00` (`239.0`) pour les 16 bits de poids fort et du *VNI* représenté sur 16 bits pour les 16 bits de poids faible. Ainsi l'adresse du groupe *multicast* associé au *VNI* 307 sera `239.0.1.51` (`0xEF000133`). Les 4096 premiers *VNI*, et donc les groupes *multicast* appartenant à la plage `239.0.0.0/20` sont réservés pour la correspondance avec les *VLAN* existants.

Exemple de configuration d'une interface *VXLAN* :

```
# ifconfig vxlan307 tunnel 10.5.0.254 239.0.1.51 vnetid 307
# ifconfig vxlan307 10.4.3.254/24
# ifconfig bridge307 add vxlan307 up
```

Il est possible de consulter la table qui fait la correspondance entre les adresses MAC des machines et les adresses des *VTEP* grâce à la commande suivante :

```
# ifconfig bridge307
```

### 3.3.2 Réseaux non filtrés

Les réseaux de serveurs n'étant pas placés derrière un pare-feu sont minoritaires dans nos infrastructures, et ne concernent que deux catégories : les réseaux de maquettes et les réseaux historiques de serveurs, hérités de la fusion des services informatiques. Ils ne représentent qu'une dizaine de réseaux, voués à disparaître avec la migration des serveurs dans de nouveaux réseaux rationalisés.

Cependant, si le besoin se faisait sentir de déployer *VXLAN* pour ces réseaux avant la fin de leur migration, deux solutions sont envisagées :

— les migrer derrière des pare-feux simplement utilisés en passerelle *VXLAN*, sans filtrage additionnel ;

— acheter des passerelles *VXLAN* dédiées [6].

La deuxième option est peu probable compte-tenu du coût d'un tel équipement face au gain apporté pour une solution temporaire.

5. La modification est disponible dans la *pull request* suivante sur *Github* : https://github.com/OpenNebula/one/pull/64.
6. Ces fonctionnalités sont souvent incluses dans les gammes d'équipements de *datacenters* de la plupart des constructeurs.

# 4 Conclusion

La solution retenue pour la mise en place de *VXLAN*, avec l'utilisation de nos pare-feux existants en passerelle, nous permet de répondre dans de courts délais à la problématique exposée, sans achat de matériel supplémentaire. En effet, avec l'arrivée prévue d'un *datacenter* de 400 $m^2$ utiles à l'Université de Strasbourg en 2018, un investissement pour une refonte de l'infrastructure réseau des salles machines actuelles semble peu justifié. Ce travail préliminaire nous permettra en revanche de prendre en compte les aspects techniques d'une solution d'*Overlay Network* dès la conception de l'architecture destinée à être déployée au sein du *datacenter*. D'autres pistes sont également à l'étude, telle que *VXLAN-EVPN*[11], en cours de standardisation, qui permet d'utiliser *MP-BGP* dans le plan de contrôle, autorisant une plus grande souplesse notamment sur les aspects de mobilité d'hôtes.

# Bibliographie


[1] IEEE 802.1ak-2007 (Amendment to IEEE Std 802.1QTM-2005). IEEE Standards for local and metropolitan area networks - Virtual Bridged Local Area Networks - Amendment 7 : Multiple Registration Protocol, 2007.

[2] Alain Heinrich et Christophe Palanché. Automatisation de l'administration de 700 serveurs avec Chef. Dans *Actes du congrès JRES2013*, Montpellier, Décembre 2013. https://conf-ng.jres.org/2013/document_revision_1977.html?download.

[3] Puppet, as in 2015. https://puppetlabs.com/.

[4] R. Enns, M. Bjorklund, J. Schoenwaelder, et A. Bierman. RFC 6241 - Network Configuration Protocol (NETCONF), Juin 2011.

[5] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, et C. Wright. RFC 7348 - Virtual eXtensible Local Area Network (VXLAN) : A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks, Août 2014.

[6] libvirt, as in 2015. https://www.libvirt.org/.

[7] libvirt - hooks for specific system management, as in 2015. https://www.libvirt.org/hooks.html.

[8] Opennebula, as in 2015. http://opennebula.org/.

[9] Openbsd, as in 2015. http://www.openbsd.org/.

[10] Pf : The openbsd packet filter, as in 2015. http://www.openbsd.org/faq/pf/.

[11] A. Sajassi, J. Drake, Nabil Bitar, Aldrin Isaac, James Uttaro, et W. Henderickx. draft-ietf-bess-evpn-overlay-02 - A Network Virtualization Overlay Solution using EVPN, Octobre 2015. https://tools.ietf.org/id/draft-ietf-bess-evpn-overlay.txt.