Les dessous de la virtualisation des fonctions réseau

Jérôme Durand
Cisco Systems
11 rue Camille Desmoulins
92130 Issy-les-Moulineaux

Résumé

Les infrastructures réseau nécessitent la mise en œuvre de toujours plus de fonctions différentes : routage, firewall, IPS/IDS, proxy, optimisation, mesure de performance... Si les constructeurs proposent des logiciels toujours plus complets, on constate que les boitiers s'empilent dans les datacenters et les sites distants, ce qui engendre de la complexité, des coûts et une rigidité incompatible avec la flexibilité et la vitesse d'exécution attendue.

Une option qui devient de plus en plus considérée est de virtualiser ces fonctions réseau. Au lieu de déployer de multiples boitiers, un serveur x86 sera installé (ou plusieurs si besoin de redondance) et les fonctions attendues seront virtualisées sur cette infrastructure. On parle alors de NFV - Network Function Virtualization.

Quels sont les challenges liés à cette virtualisation ? Comment garantir les performances en environnement virtualisé ? Comment créer et déployer des fonctions réseau virtuelles ? Toutes ces questions sont adressées dans cet article.

Mots clé

Cloud, Datacenter, DPDK, KVM, NFV, Performance, Orchestration, OVS, SR-IOV, UCPE, Virtualisation, VNF, WAN

Introduction

Si l'on ouvre la baie réseau d'un site distant, on trouvera souvent bien plus qu'un simple routeur d'accès. Des *firewalls*, IPS, solutions d'optimisation WAN, des solutions DDI (DNS / DHCP / IPAM – *IP Address Management*) ou même de simples services locaux sont souvent installés. Parmi ces derniers on trouve des serveurs de fichiers, serveurs d'impression, solutions monétiques, solutions de vidéosurveillance ou encore de simples serveurs *Linux* ou *Windows* utilisés pour résoudre des incidents ou d'autres services. La multiplication des "boitiers" sur les sites distants entraîne des coûts opérationnels importants et des délais (livraison, installation physique) et aussi une augmentation de la complexité et des designs non fonctionnels (par exemple un IPS aurait du mal à détecter des menaces si celui-ci est déployé en aval d'une solution d'optimisation WAN qui compresse les données).

Pour simplifier le réseau, une option qui devient de plus en plus considérée est la virtualisation des fonctions réseau (NFV). Au lieu de déployer de multiples boitiers, un serveur *x86* sera installé (ou plusieurs si besoin de redondance) et les fonctions attendues seront virtualisées sur cette infrastructure. Les intérêts sont les suivants :

– Le déploiement d'une nouvelle fonction ne nécessite aucune opération logistique, d'opération d'installation coûteuse, ni de contrainte douanière ! Il suffit de charger l'*appliance* virtuelle sur le site distant et une nouvelle fonction peut être disponible immédiatement.

– Le design est simplifié puisqu'il n'y a plus de contrainte physique de connexion des équipements. S'il faut modifier ou adapter une infrastructure réseau (par exemple pour replacer l'IPS en amont de l'optimisation WAN) cela peut se faire en quelques clics sans re-câblage.

Globalement le coût global du WAN peut être largement diminué tout en le rendant plus dynamique. L'intérêt est aussi de pouvoir faire face ponctuellement à des problématiques sur tout ou partie des sites. Si un évènement devait nécessiter sur un site une transmission vidéo et la mise en place d'une solution ad-hoc, il devient possible de s'appuyer sur l'infrastructure de virtualisation en place. La capacité de retirer cette solution une fois la diffusion terminée permet aussi d'envisager d'autres modes d'acquisition (abonnement à l'heure d'utilisation par exemple !)

Dans les datacenters, la virtualisation est maîtrisée depuis des années pour délivrer des applications et la containérisation gagne du terrain. Il est tentant d'utiliser les mêmes principes pour les fonctions réseau. Des acteurs majeurs du cloud (comme *AWS, Azure, Google...*) l'ont compris en fournissant des plateformes IAAS (*Infrastructure-as-a-Service*). Ici l'objectif est de gagner en agilité et de simplifier l'architecture du datacenter en ne reposant que sur une plateforme de "*compute*" homogène (des serveurs *x86* et l'architecture de réseau datacenter associée). Ici aussi l'avantage est de pouvoir passer d'un modèle rigide d'appliance à un modèle "*pay as you grow*" dans lequel l'organisation n'est pas obligée de surinvestir au démarrage. Elle peut rajouter des VNF (*Virtual Network Functions*), augmenter la performance, au fur et à mesure de l'augmentation de ses besoins.

Cet article a pour objectif de donner quelques clés afin de mieux appréhender les projets de virtualisation de fonctions réseau.

1 La structure d'une VNF

Il est indispensable de comprendre la structure d'une VNF pour pouvoir l'opérer de manière optimale.

1.1 L'image disque

L'image disque est l'élément de base qui constitue une machine virtuelle. Il s'agit simplement d'un fichier qui comprend toute la structure d'un disque réseau complet (fichier et arborescence).

De très nombreux formats existent dont les principaux listés ci-dessous :

– .qcow2 et son ancêtre .qcow (format standard de l'hyperviseur KVM) ;
– .vmdk (format *VMWare*) ;
– .vhd et .vhdx (Format utilisé par *Microsoft Hyper-V*. VHD - *Virtual Hard Drive*) ;
– .iso (format traditionnel pour les disques d'installation) suivant la norme ISO-9660 ;
– .img (initialement introduit par *Apple*) ;
– .raw (copie brute bit à bit du disque sans ajout de *metadata*).

Il est possible de démarrer directement l'image disque sur un hyperviseur, l'administrateur devra alors spécifier toutes les caractéristiques de la machine virtuelle au moment du démarrage (CPU, mémoire...). Cette étape peut être fastidieuse et source d'erreur car elle requiert que l'administrateur connaisse les paramètres adéquats et compatibles avec l'image disque démarrée.

De même, dans une approche NFV, on va généralement avoir une approche "industrielle" où l'on voudra répliquer les actions de nombreuses fois. On va donc chercher à simplifier les opérations et les automatiser au maximum.

1.2 Les packages

Pour les raisons décrites ci-dessus, on ne va généralement pas travailler sur des images disques seules, mais on va plutôt créer des "*packages*" qui vont contenir l'image disque évidemment, mais aussi tous les descriptifs techniques permettant de démarrer la VNF avec tous les paramètres requis. Ces "*packages*" sont un fichier compressé dont l'extension est généralement :

– .ova (*Open Virtual Appliance*), assimilé souvent par erreur à un "*package*" *VMWare ESXi* uniquement. Cette extension indique qu'il s'agit d'un fichier "*package*" au format OVF (*Open Virtualisation Format*). C'est un fichier tar.gz dont l'extension a été modifiée pour indiquer le type de contenu.
– .tar.gz (archive compressée), utilisée parfois sur KVM pour éviter une confusion avec les .ova et *VMWare ESXi*.

Il convient de noter qu'hormis le changement d'extension, ces deux fichiers sont des *packages* qui vont contenir toutes les informations nécessaires pour démarrer une VNF (image disque et informations complémentaires détaillées ci-après).

D'autres formats existent selon les hyperviseurs. Les principes expliqués ci-dessous restent généralement applicables.

1.3 Propriétés de l'image

Le fichier descripteur des propriétés de l'image permet de décrire comment démarrer l'image disque. C'est dans ce fichier qu'on pourra par exemple indiquer :

– le nombre de vCPU à allouer à la machine virtuelle ;
– la mémoire nécessaire ;

– la taille du/des disque(s). En effet même si une image disque est fournie avec une certaine taille, il est tout à fait possible de la déployer sur un volume plus grand. L'espace disque rajouté pourra être utilisé par la VNF dans son opération courante (copie de fichiers, génération de logs…) ;
– les interfaces réseau virtuelles à déployer sur la VNF (nombre et type) ;
– les drivers réseau supportés (pour SR-IOV par exemple…) ;
– le nom sous lequel monter des fichiers de démarrage (ou *bootstrap*) ;
– éventuellement de nombreux autres paramètres.

Il est possible de définir plusieurs profils différents au sein de ce fichier. Par exemple, on pourrait imaginer les profils suivants pour un routeur virtuel :

– *Small* : 2 vCPU, 4GB de RAM ;
– *Medium* : 4 vCPU, 4GB de RAM ;
– *Large* : 4 vCPU, 8GB de RAM.

L'administrateur peut simplement choisir le profil à utiliser au moment du déploiement du routeur virtuel.

Pour un environnement KVM, ce fichier est généralement nommé *image_properties.xml* et est lu par l'hyperviseur au moment du démarrage de la VNF.

1.4 Fichiers de démarrage

Des **fichiers de démarrage** peuvent aussi être fournis. Ils seront lus par la VNF au démarrage et permettront de rajouter d'éventuelles configurations nécessaires, spécifiques pour la VNF déployée. Les informations qui doivent être généralement fournies à une VNF au moment de son déploiement sont les suivantes :

– configuration réseau (adresses IP, passerelle par défaut, nom d'hôte…) ;
– configuration des fonctions assurées de la VNF (routage, sécurité, autre…) ;
– ajout de licences logicielles ;
– etc.

Pour pouvoir changer ces configurations de démarrage lors de chaque déploiement de VNF depuis un *package*, il est nécessaire que les fichiers de configuration (fournis dans le *package*) intègrent des variables. Au démarrage de la VNF, l'hyperviseur remplace les variables par les valeurs ad-hoc spécifiées par l'administrateur.

Il est important de comprendre que la possibilité d'avoir ces fichiers de démarrage, de même que leur nom et leur syntaxe, est complètement dépendante de la VNF. L'hyperviseur va simplement monter les fichiers spécifiés dans l'arborescence disque de la VNF au moment de son démarrage, avec les noms de fichiers cibles désirés. La lecture et l'interprétation de ces derniers dépend complètement de la VNF. Aussi, pour pouvoir créer un *package* avec ces fichiers de *bootstrap*, il faut connaître ce que la VNF supporte. Le degré de configuration possible sur une VNF lors de son démarrage peut être déterminant dans le choix de cette dernière.

Sur les systèmes basés sur une souche *Linux*, qui représentent la majeure partie des VNF installées, la solution de *bootstrap* est appelée **cloud-init**. Ce nom vient du fait que les images qcow2 des distributions *Linux*, principalement utilisées sur des infrastructures cloud *Openstack*, sont appelées "*Cloud Images*".

À titre d'exemple nous pouvons regarder le fonctionnement d'une distribution *Ubuntu* standard. Le fichier user_data, s'il est inclus dans le *package*, sera exécuté au premier démarrage. Ce script, un *bash shell* par exemple, pourra contenir toutes les opérations souhaitées par la VNF (configuration d'un mot de passe *root*, configuration réseau, exécution de logiciels…)

Voici un exemple de fichier user_data pouvant être utilisé comme *cloud-init* d'une VM *Ubuntu*. Deux interfaces ens3 et ens4 sont configurées avec des adresses IP qui sont des variables qui seront renseignées par l'administrateur au moment de l'instanciation de la VM. Un mot de passe *root* est également configuré et *sshd* est démarré, et autorise la connexion directe de l'utilisateur *root*.

```
#!/bin/bash
passwd root << EOF
my_password
my_password
Y
EOF
echo "Cloud-init running user-data off config drive (/dev/sr0)"
echo Setting up interfaces and addresses
ifconfig ens3 down
ifconfig ens3 $NICID_0_IP_ADDRESS netmask $NICID_0_NETMASK
ifconfig ens4 down
ifconfig ens4 $IP_ADDRESS netmask $NETMASK
route add default gw $GATEWAY ens3
netstat -rn
adduser lab sudo
ifconfig ens3 up
ifconfig ens4 up
sed -i 's/PermitRootLogin no/PermitRootLogin yes/'
/etc/ssh/sshd_config
sed -i 's/PasswordAuthentication no/PasswordAuthentication
yes/'
16
/etc/ssh/sshd_config
service ssh restart
```

Figure 1 - Cloud-init pour Linux Ubuntu standard

Les éditeurs logiciels prennent soin généralement de packager davantage leurs fichiers de *bootstrap*.

1.5 Fichier manifeste

Un fichier "manifeste" liste simplement les fichiers du *package* avec un *hash* (SHA1 par exemple) pour permettre de valider l'intégrité de chaque fichier. Comme les *packages* sont régulièrement copiés (parfois sur de multiples sites distants, à travers des WAN bas débit) et que leur contenu est critique, on doit s'assurer que chaque fichier est bien valide. C'est le rôle du fichier manifeste. La création du fichier manifeste est généralement automatiquement faite par un outil ou script.

1.6 Points de vigilance pour un projet NFV

Il est important de prendre en compte dans un projet NFV la nécessité de pouvoir instancier des VNF en personnalisant leurs attributs (à la fois leur environnement comme le processeur, la mémoire...) mais aussi leur configuration initiale. Du temps doit être prévu pour créer des *packages* parfaitement adaptés aux besoins initiaux.

La capacité de pouvoir fournir un fichier de configuration de *bootstrap* à une VNF doit rentrer en compte dans le choix de cette dernière. Il convient de privilégier les VNF pour lesquelles des configurations de *bootstrap* sont déjà utilisables clé en main, ou bien qui permettent de créer simplement ces fichiers de *bootstrap*, documentation à l'appui.

L'utilisation d'images disque qcow2, sans aucune explication quant à son fonctionnement, peut se faire dans un lab mais ne permettra pas une utilisation massive en production car cela nécessiterait trop d'opérations manuelles à chaque déploiement (définition des propriétés de la VNF, connexion en console et application des configurations de *bootstrap* à la main)

2 La gestion du réseau et l'optimisation des performances

2.1 Principes de base

L'hyperviseur a pour mission également de virtualiser le réseau. La VNF se connecte à des réseaux virtuels, qui sont éventuellement attachés à des interfaces physiques du serveur pour offrir une connexion avec l'extérieur.

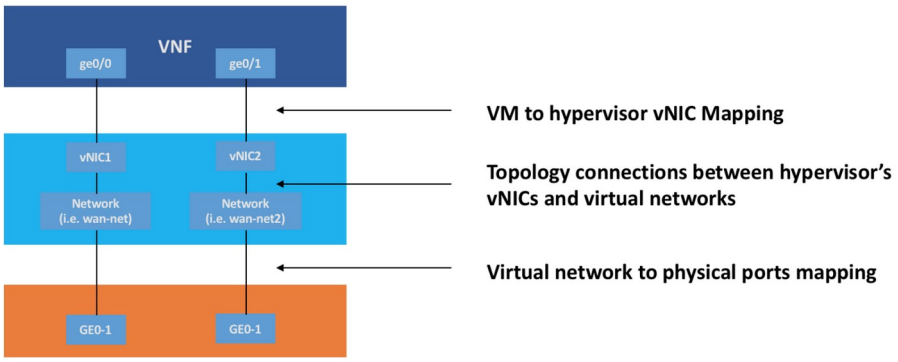

Figure 2 - Correspondance entre interfaces virtuelles et interfaces physiques

Des *mappings* sont faits entre les interfaces de la VNF et des vNICs (virtual NIC - interface réseau virtuelle). Une difficulté récurrente est de connaître cette correspondance pour être certain que les connexions faites au niveau de l'hyperviseur (qui ne connaît par défaut que les vNIC1, vNIC2, etc.) correspondent bien aux ports adéquats de la VNF (par exemple eth0, eth1...)

Ce *mapping* dépend directement de la structure de la VNF. C'est à ce niveau qu'est défini dans quel ordre les ports sont présentés à l'hyperviseur. Par exemple, la vNIC1 peut très bien être associée à un port de management "eth0" de la VNF, puis la vNIC2 associée à une interface "gi0", la vNIC3 à "gi1"... Aussi, une VNF peut très bien avoir un nombre d'interface minimal ou maximal à respecter. L'hyperviseur ne connaît pas ces informations.

Il convient donc de bien comprendre les VNF que l'on utilise pour les manipuler correctement au niveau du réseau et ne pas passer son temps à rechercher les *mappings*. Une technique largement utilisée pour savoir à coup sûr le *mapping* réalisé est de regarder les adresses MAC des vNICs au niveau de l'hyperviseur et de regarder sur quelles interfaces des VNF ces dernières sont associées. Dans la création du *package*, il est également possible de spécifier dans le fichier de propriété de l'image les noms des interfaces des VNFs et les correspondances avec les vNICs afin de pouvoir les visualiser dans l'hyperviseur.

2.2 Optimisation des performances réseau

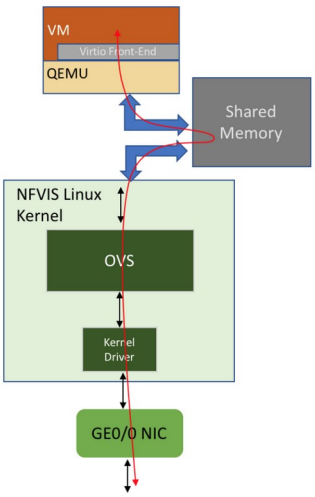

Figure 3 - OVS

Un *switch* virtuel, OVS (*Open Virtual Switch*) sur KVM, va commuter les paquets au niveau de l'hyperviseur (*kernel space*) afin de les distribuer vers la VNF appropriée.

Ensuite une deuxième analyse des paquets est faite au sein de la VNF destinatrice dans le "*user space*". Le driver "virtio" (*virtual I/O*) est présent au niveau de la VNF pour gérer ces interfaces virtuelles génériques.

Aussi pour chaque paquet, plusieurs analyses sont réalisées consécutivement au niveau *Kernel* et *User*, nécessitant à chaque fois des interruptions CPU, des copies dans la mémoire, causant une dégradation non maîtrisée des performances pour les VNF déployées.

Ce mode de fonctionnement peut vite s'avérer être inefficace car le CPU est utilisée pour de la commutation réseau et non pour assurer les fonctions de base de la VNF. Ceci est d'autant plus problématique quand un paquet doit transiter au travers de plusieurs VNF sur un même serveur puisque le CPU sera sollicité également pour la commutation

inter-VNF. On comprend que sans optimisation, l'impact peut vite être catastrophique et peser sur les performances globales du système.

Pour pallier ce problème, il existe plusieurs solutions :

- **Dédier la NIC (interface réseau physique) à une vNIC (interface virtuelle) d'une VNF particulière**. Dans ce cas le paquet ne requiert qu'une seule analyse. En revanche, cela est rarement souhaité car on perd tout l'intérêt de la virtualisation, avec la possibilité d'attacher plusieurs VNF à une même interface réseau.

- **SR-IOV** (*Single Root I/O Virtualization*) permet de réaliser cette connexion directe tout en permettant à plusieurs VNF d'être associées à un même port physique. Un *virtual forwarder* au niveau de la NIC transmet directement les paquets à la VNF ad-hoc après analyse du paquet assurée directement par le matériel. En revanche cela nécessite que la VNF dispose des drivers ad-hoc requis par la NIC qui doit elle aussi être développée dans les règles de l'art. SR-IOV apporte aussi généralement des restrictions quant à l'utilisation de certaines fonctionnalités. Aussi il convient de vérifier que l'utilisation de SR-IOV n'aille pas à l'encontre des objectifs recherchés.

  Les performances d'une VNF peuvent être considérablement améliorées avec SR-IOV, aussi il est important de prendre ce support en considération dans le choix de la VNF et bien évidemment s'assurer que la NIC du serveur supporte SR-IOV. Cela peut nécessiter aussi des modifications au niveau du BIOS : tous les étages sont concernés, le monde rêvé d'une virtualisation indépendante de l'infrastructure sous-jacente n'existe pas !

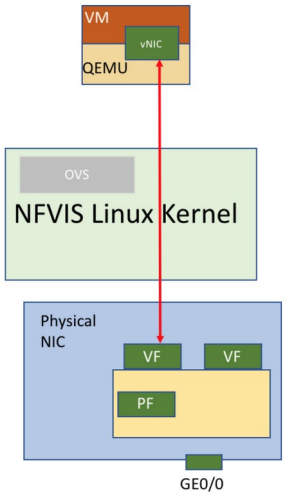

Figure 4 - SR-IOV

– **OVS-DPDK** [1] est une autre méthode d'optimisation qui n'a pas de pré-requis sur la VNF. **DPDK** (*Data Plane Development Kit*) [2] [3] assure une commutation optimisée dans le *user space* grâce à des CPU dédiés, et vient se substituer aux opérations réseau d'ordinaire réalisées dans le *kernel*.

  OVS-DPDK permet l'utilisation de DPDK dans un contexte de virtualisation avec OVS.

  Ici les performances sont améliorées car une seule commutation est faite dans le *user space* sur des ressources dédiées (ie. des cœurs dédiés) avec des mécanismes de traitement optimisés. Une VNF utilisant OVS-DPDK offre généralement des performances moins forte que SR-IOV mais il n'y a pas de question de dépendance *hardware* forte.

  FD.IO VPP (Vector Packet Processing) [4] est similaire à DPDK pour booster les performances en reposant sur l'exécution dans le *user space* d'algorithmes optimisés pour la commutation de paquets.

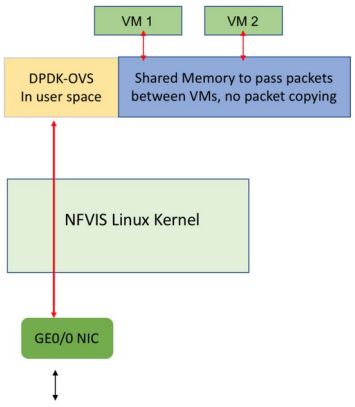

Figure 5 - OVS-DPDK

– Des innovations sont en cours dans le domaine des **SmartNICs** [5] [6] qui visent à solutionner ces problématiques de performance. L'idée est de reposer sur des NICs plus intelligentes capables de réaliser "en *hardware*" les actions réseau désirées par les différentes VMs (par exemple imposer un label MPLS, router un paquet, ...) Le but est de reproduire ce que des ASICs ou *Network Processors* font pour optimiser les performances sur des équipements réseau physiques et surtout économiser des cycles CPU pour pouvoir faire tourner un maximum de VNF sur un même *host*. Les enjeux sont avant tout économiques. Si des réalisations ont été menées avec succès chez des géants du cloud (*Microsoft, Google*...) qui maîtrisent l'intégralité du *stack* (SmartNIC, hyperviseur, VNF…), il n'y a pas encore de solution utilisable "sur étagère" suffisamment ouverte pour permettre aisément à chacun de tirer profit de ces innovations dans des environnements hétérogènes. Un des challenges réside dans la capacité de programmer la SmartNIC depuis le *host*. Plusieurs techniques sont utilisées comme la programmation directe d'un FPGA en VHDL, l'utilisation de *P4* ou même *OpenFlow*. Il est même possible de descendre un code *C* dans la *SmartNIC* pour y exécuter des fonctions *control-plane* (par exemple un routeur BGP). On ne peut pas vraiment à ce jour déployer ce type de technologies sans réelle expertise. De nombreuses start-ups se sont lancées à l'assaut de la conquête des *SmartNICs* et il est probable que des avancées considérables soient faites dans les années à venir.

2.3 Points de vigilance pour un projet NFV

La performance reste la clé dans un environnement virtualisé comme c'est le cas sur des équipements physiques dédiés. Elle doit évidemment être prise en compte dans le choix de tous les éléments : Serveur, NIC, hyperviseur, VM. Des tests doivent donc être menés pour valider les performances complètes d'un système (VNFs avec hyperviseur et NIC). Le support de SR-IOV sur le trio VNF, hyperviseur, NIC est un réel avantage qui boostera les performances. Investiguer sur le support de SR-IOV et les impacts associés comme la perte de fonctionnalités est donc important. Il faut vérifier également la possibilité d'utiliser SR-IOV pour la communication entre les VNF, en s'assurant que les *drivers* et *firmwares* de la NIC soient compatibles et fonctionnels.

La présence d'OVS-DPDK ou de toute autre optimisation du *vSwitch* offre également un réel avantage, à défaut de pouvoir déployer SR-IOV. Il convient cependant de vérifier si cela est réaliste, DPDK monopolisant des ressources. Par exemple sur un UCPE (*Universal CPE* - serveur optimisé pour remplacer le routeur d'accès d'un site distant avec des services virtualisés) de petite taille à bas coût avec quatre cœurs, il n'est pas vraiment réaliste d'en monopoliser un uniquement pour OVS-DPDK (en plus de celui utilisé par l'hyperviseur). Cela ne laisse pas beaucoup de place pour les fonctions réseau !

Il convient aussi de ne pas céder trop facilement aux sirènes des *SmartNICs* avant d'avoir vérifié qu'il est possible d'en exploiter la capacité. Le sujet est extrêmement prometteur mais ne peut pas encore être mis entre toutes les mains.

3 Orchestration et management NFV

Une solution de réseau virtualisée se compose généralement de plusieurs systèmes distincts (sinon, la virtualisation perd un peu de son sens !) Chaque VNF est généralement managée par un outil adapté propre. Par exemple toutes les VNF *firewall* d'un même réseau vont ainsi être intégrées dans la console de management fournie par le constructeur. Cette console fera toutes les actions de supervision, configuration... Dans un système à n VNF, on peut arriver aisément à n+2 consoles de management :

- une console par VNF (n au total) ;
- une console pour le management des hyperviseurs ;
- une console pour le management des *hardwares*. Ce dernier ne doit pas être omis : la bonne santé des serveurs est la base pour garantir des services fonctionnels !

La création de services de bout en bout ne peut se faire efficacement que si tous ces éléments fonctionnent ensemble. Faire chaque installation / configuration séparément sera inefficace dans des grands réseaux. Le rajout d'une orchestration globale, assurant le déploiement global d'un nouveau serveur, avec les VNF adéquates, leur configuration initiale et leur intégration dans chaque console de manager doit être considéré.

Ces solutions d'orchestration sont fournies entre autres par les éditeurs de solutions NFV mais peuvent être disponibles en Open Source (éventuellement *Ansible*, ou encore *Heat* dans le cadre d'un projet *OpenStack*). Dans la plupart des cas, ces systèmes nécessitent un travail non négligeable de l'administrateur qui doit définir ses règles (liste des VNF,

chaînage, etc.) avec la capacité de s'interfacer avec de nombreux équipements hétérogènes. Bien souvent, cela s'apparente davantage à de la programmation qu'à de la configuration réseau, démontrant au final combien un projet NFV requiert des composantes dans tous les domaines : réseau, système, *hardware*, virtualisation et programmation.

4 Bâtir son infrastructure NFV

Les mécaniques de dimensionnement sont radicalement différentes pour la virtualisation en datacenter et pour les UCPE (*Universal CPE*).

Dans le premier cas, tout un ensemble de serveurs vont être installés et l'objectif est d'optimiser les ressources en faisant fonctionner un maximum de VNF sur chaque serveur tout en contrôlant que la performance dans son ensemble est bonne. Le coût du serveur en soit n'est pas le problème, la question à adresser est la rentabilité du serveur par rapport au nombre de VNF qu'il va héberger [7]. Cet article ne rentrera pas trop dans ces considérations qui sont davantage celles de fournisseurs de services.

Dans le second cas, la problématique est autrement différente puisqu'il n'est pas possible de faire une quelconque optimisation sur chaque site où l'on ne va déployer que un ou deux UCPE (pour une éventuelle redondance). L'objectif ici est de choisir un serveur économiquement adapté qui répond au plus juste au besoin et aux évolutions souhaitées. Aussi il est nécessaire en amont d'évaluer les pré-requis pour les machines virtuelles souhaitées en termes de CPU, mémoire, disque. Il faudra rajouter les ressources consommées par l'hyperviseur et éventuellement le *switch* virtuel si OVS-DPDK est utilisé. Une marge sera ajoutée pour éventuellement faire face au besoin d'une nouvelle VNF ou s'il faut plus de performance. Tous ces pré-requis vous donneront une base de départ (en CPU, mémoire, disque...) et il faudra trouver ensuite un serveur qui adresse ces besoins.

La création d'un site pilote pour valider les performances et le bon fonctionnement des services est indispensable avant d'investir ! Dans le cas général, on évitera de faire de l'*over-subscription* ou de l'*hyper-threading* sur les services réseau. Des opérations en pseudo temps-réel sont attendues, chaque service doit réellement pouvoir disposer des ressources qui lui sont données. Évidemment, si votre architecture contient également quelques serveurs simples (serveur de fichier, serveur d'impression...) alors il sera possible à ce niveau d'optimiser un peu les ressources.

Le dernier point à prendre en compte dans le choix du serveur est lié aux critères environnementaux :

– Quel format ? Rappelez-vous que ce serveur ne sera pas installé dans une salle blanche classique mais dans un espace généralement plus austère dans lesquels les serveurs peuvent souffrir. Le matériel doit être peu profond pour s'adapter aux baies Telecoms de 600 mm encore nombreuses et également être potentiellement durci pour faire face aux conditions spécifiques des sites distants. Il faudra éventuellement vérifier si un équipement sans ventilation mécanique est nécessaire. Souvent, sur les sites distants, le routeur est installé dans un bureau également occupé par des employés : un équipement silencieux est ainsi nécessaire.

– Besoin de POE ? Pour des petits sites, un serveur avec quelques ports POE et un point d'accès peuvent suffire. Il peut en effet s'avérer très pratique de disposer de ports POE.

– SR-IOV ? Est-ce que le hardware dispose des hardwares permettant l'utilisation de SR-IOV pour la communication avec l'extérieur, ou pour la communication entre VNF ?

– Connectivité WAN ? Est-ce que l'Ethernet est suffisant ? Faut-il prévoir une interface 4G ou DSL ?

– Le serveur est-il simplement manageable ? Comment pourrez-vous upgrader le BIOS à distance ? Est-il est possible de se connecter sur le serveur comme on se connecte à un routeur simplement ?

– Le serveur est-il sécurisé ? A-t-on l'assurance qu'il ne s'agit pas de matériel contrefait malveillant qui pourrait intercepter vos échanges ? Quelle garantie avez-vous du constructeur quant à l'authenticité du matériel ?

5 Le service chaining

Le développement de la virtualisation des fonctions réseau, en datacenter comme sur les sites distants, a mis en évidence le besoin de pouvoir chaîner aisément les fonctions réseau.

Prenons l'exemple d'un design assez classique qui nécessite qu'un paquet transite d'abord par le routeur SD-WAN, puis par un *firewall*, ensuite par une solution d'optimisation WAN avant de revenir par le routeur SD-WAN pour être chiffré avant d'être envoyé sur le WAN. Il convient tout d'abord de placer les différentes fonctions dans le bon ordre. Il serait en effet difficile de scruter un paquet au niveau du *firewall* si celui-ci était placé après un mécanisme d'optimisation WAN, qui compresse les paquets et modifie largement la gestion de TCP !

Dans un monde où chaque solution est déployée sous forme "d'*appliance*" physique, le séquencement des opérations est assez simple puisqu'il va suivre le câblage, via des opérations de routage ou commutation traditionnelles. Chaque boitier va traiter le paquet et l'acheminer au suivant après traitement.

5.1 Solutions actuelles

Les designs virtualisés aujourd'hui tendent à reproduire plus ou moins ces mêmes méthodes compte tenu de l'absence d'alternative. Des VLAN sont implémentés entre les différentes VNF et une chaîne est faite en aboutant statiquement les éléments entre eux.

Les VLANs pourront s'appuyer sur le *switch* virtuel, la NIC du serveur ou encore un commutateur externe. Le choix dépendra des fonctionnalités et performances désirées. Notons que paradoxalement il peut s'avérer plus performant de reposer sur un commutateur externe pour décharger le serveur d'un énième cycle d'opérations.

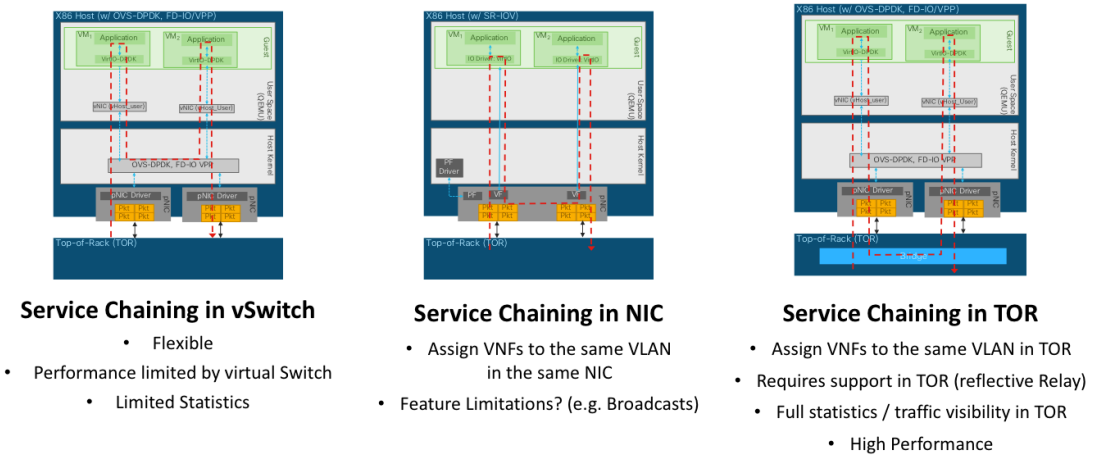

Figure 6 - Différentes solutions de service-chaining

Ces solutions, bien que fonctionnelles, ne répondent pas intégralement aux objectifs d'un réseau virtualisé :

– La virtualisation a pour objectif de simplifier l'ajout / la suppression de fonctions réseau. Il convient donc de s'assurer que des contraintes réseau ne viennent pas gâcher cette flexibilité souhaitée.

– Comme les solutions partagent des ressources physiques communes (processeurs, mémoires…), on va d'autant plus chercher à optimiser le traitement des paquets. Pour cela, on va s'assurer qu'une même opération ne soit pas faite sur plusieurs VNF. Dans l'exemple précédent, on peut concevoir qu'une fonction DPI soit implémentée sur le routeur SD-WAN, mais aussi sur le *firewall* et la solution d'optimisation. Des solutions doivent permettre d'éviter de répéter une opération si coûteuse en ressources.

Innovations

5.1.1 NSH - Network Service Header

Afin d'avancer sur le "*service chaining*", le groupe de travail IETF[1] sfc (*Service Function Chaining*) a été créé. La problématique complète ainsi qu'une architecture ont été définis dans les RFC 7498 et 7665.

1. Internet Engineering Task Force

Les flux sont tout d'abord pris en charge par un "*Service Classifier*" (SC) et sont ensuite acheminés entre "*Service Function Forwarders*" (SFF), qui vont à leur tour successivement appeler des "*Service Functions*" (SF).

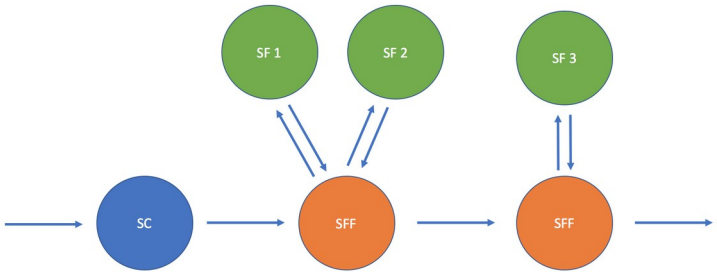

Figure 7 - Service Function Chaining - Architecture simplifiée

Bien évidemment, différentes fonctions peuvent être portées par une seule et même VNF. Par exemple, le routeur SD-WAN pourra être à la fois *Service Classifier*, puis le SFF qui appellera des *Service Functions* présentes sur d'autres VNF.

L'architecture laisse libre le mécanisme permettant de faire transiter les paquets entre SFF et SF. Par exemple, des tunnels VXLAN ou GRE pourront permettre une communication simple au-dessus d'une infrastructure IP : plus besoin d'ajouter des VLANs. L'ajout d'une nouvelle SF (par exemple un *proxy web*) se fera simplement en déclarant une nouvelle SF au niveau du SFF voulu.

Par contre, un nouveau protocole appelé NSH (*Network Service Header*) a été conçu pour permettre une communication au niveau service des éléments entre eux. Un en-tête NSH est ajouté aux paquets et va propager notamment le *Service Path Identifier* déterminé au niveau du SC, qui sera utilisé sur chaque SFF pour appeler successivement les différentes SF. Le protocole NSH est hautement flexible et permet de transporter tout type de metadata, caractérisant mieux le service et/ou le flux. Un *application-ID* peut par exemple être ajouté par la première SF qui fera du DPI, et pourra être ensuite consommé par les SF suivantes qui n'auront plus à déterminer l'application transportée.

Afin de paramétrer tous les éléments de la chaîne dynamiquement depuis un contrôleur central, un plan de contrôle NSH est également en cours de définition à l'IETF.

Il est peut-être un peu tôt pour espérer avoir des solutions offrant un *service chaining* intégralement dynamique et optimal compte tenu de la jeunesse relative de la standardisation de NSH à l'IETF. Cependant il semble indispensable de prendre en compte dès aujourd'hui les plans d'intégration de NSH dans chaque solution de virtualisation. Quand un réseau hybride physique/virtuel est en place, il est important de vérifier que les ASIC ou autres *Network Processors* utilisés sont assez flexibles pour intégrer la gestion de NSH, protocole complexe à traiter du fait de sa très haute dynamicité.

5.1.2 Segment Routing (SR et SRv6)

Segment Routing [8] permet simplement de faire transiter un paquet par des nœuds désirés. Ceci est réalisé en insérant des labels MPLS (pour SR) ou extensions IPv6

(pour SRv6). On comprend aisément que SRv6 peut jouer un rôle clé dans le *service-chaining* [9] en insérant les adresses IPv6 des différentes VNF à traverser dans chaque paquet. Un autre élément très intéressant de SRv6 est la capacité d'exploiter certains des nombreux bits des adresses IPv6 pour donner des directives ou consignes à chaque VNF. Si ces approches sont étudiées sur des infrastructures "*cloud*" spécialisées, on ne voit pas encore d'application simple clé-en-main pour des organisations standards. Ici aussi il faut rester à l'écoute des innovations à venir. Notons cependant que SR est disponible sur *Linux* et que cela présage d'un avenir radieux.

6 Conclusion

Le fantasme collectif dans lequel la virtualisation réseau permet de s'affranchir complètement du *hardware* n'existe pas. Cette apparente indépendance permettra certes au débutant de rapidement instancier quelques VNF (*Virtual Network Functions*) et de les opérer comme des équipements physiques, mais cela se fera au détriment de la performance et n'offrira pas non plus la flexibilité attendue : à quoi bon virtualiser si l'opération des VNFs reste unitaire et manuelle, avec une performance faible qui n'apporte au final aucun bénéfice pour l'utilisateur ?

Un obstacle principal au succès de ces projets est le manque global de connaissance à la fois sur les technologies réseau et sur les systèmes (serveurs, *Linux*, virtualisation, …). Les experts réseau sont souvent trop éloignés des problématiques systèmes adressées par des équipes spécialisées et l'inverse est malheureusement tout aussi vrai ! Redessiner les contours des organisations et brasser les équipes ne peut qu'aider au succès de ces projets à cheval entre ces technologies. Cette évolution doit se faire en douceur et s'inscrire dans la durée pour permettre une véritable dynamique sur la virtualisation des fonctions réseau.

Sur un plan purement technologique, il existe comme souvent plusieurs approches, qui sont plus ou moins intégrées ou packagées. L'avantage d'avoir une solution prescriptive est de simplifier la vie de l'administrateur en lui permettant de démarrer rapidement et en toute sécurité sur des solutions validées et testées pour garantir une performance optimale. II reste cependant indispensable de s'assurer de l'ouverture de la solution pour supporter des VNF tierces et des *packages* customisés. L'expert pourra se risquer à créer lui-même son "*stack*", mais il devra au préalable vérifier la compatibilité entre tous les éléments pour garantir la performance de l'ensemble mais également la bonne administration globale du système. Tout doit être fait pour éviter de se retrouver dans une *situation* inextricable dans laquelle chaque fournisseur (hyperviseur, VNF, NIC...) se rejette la balle au moindre incident.

Il reste certain que la démocratisation de la virtualisation des fonctions réseau touche de plus en plus de monde et des solutions existent pour démarrer simplement. Le sujet est cependant si innovant qu'il est indispensable de faire de la veille technologique régulièrement.

Bibliographie

[1] https://software.intel.com/en-us/articles/open-vswitch-with-dpdk-overview

[2] http://dpdk.org

[3] https://conf-ng.jres.org/2017/document_revision_2092.html?download

[4] https://fd.io/technology/

[5] https://www.microsoft.com/en-us/research/uploads/prod/2018/03/Azure_SmartNIC_NSDI_2018.pdf

[6] https://www.netronome.com/blog/ovs-offload-models-used-nics-and-smartnics-pros-and-cons/

[7] https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/vmware-tuning-telco-nfv-workloads-vsphere-white-paper.pdf

[8] J. Durand, Segment Routing, l'art de faire plus avec moins, 2017

[9] C. Filsfils, P. Camarillo, J. Leddy, D. Voyer, S. Matsushima, Z. Li, SRv6 Network Programming - draft-filsfils-spring-srv6-network-programming-07, 2019