République **A**lgérienne **D**émocratique et **P**opulaire
Ministère de l'**E**nseignement **S**upérieur et de la **R**echerche **S**cientifique

UNIVERSITE **M**OULOUD **M**AMMERI DE **T**IZI-**O**UZOU

FACULTE DU**G**ENIE **E**LECTRIQUE ET D'INFORMATIQUE
DEPARTEMENT D'INFORMATIQUE

Mémoire de Fin d'Etudes
De MASTER ACADEMIQUE

Domaine : Mathématiques et Informatique
Filière : Informatique

Spécialité : **Réseaux, Mobilité et Systèmes Embarqués**

Présenté par
HADJEM Abdelmounaim Boussad

Encadré par: IBOUHMADOUCHE Imene

Co-Encadré par: DAOUI Mehammed

Thème

Conception d'une architecture Openstack pour infrastructure NFV-Service FunctionChaining au sein d'ATM Mobilis

Mémoire soutenu publiquement le 24/09/2018 devant le jury composé de:

Présidente : Mme AOUDJIT Rachida

Co-Encadreur : Mr DAOUI Mehammed

Examinateur : Mr RAMDANE Mohamed

Dédicaces

Je dédie ce modeste travail à :

Ma Mère en premier, la personne qui m'a le plus encouragé à étudier, à m'instruire et à trouver ma voie par-dessus tout. Ainsi que pour tout le reste je ne saurais jamais t'être assez reconnaissant.

Mon Père, Merci pour la rigueur et la discipline que tu m'as appris.

A vous deux Merci pour le courage que vous m'avez appris à avoir et pour tout votre soutien. Que le tout Puissant vous Accorde une longue vie pleine de bonne santé et de prospérité.

A mon jeune frère Massi, ta responsabilité m'a aidé et m'aide toujours à prendre les bons choix, tu es un frère et un ami proche, je suis déjà très fière de toi.

A Lynda, Ma meilleure amie, Merci d'avoir été présente, dans les meilleurs comme dans les pires moments, depuis si longtemps déjà, et je te souhaite bon courage pour ton propre PFE.

A tous mes Amis et A tous ceux qui me sont chers sur lesquels j'ai toujours pu compter, Merci infiniment.

A l'UMMTO Debate Club, j'ai fait tellement de chemin avec vous, nous ne le réalisons peut-être pas encore mais ce que nous avons vécu et accompli ensemble, prendra sa vraie signification plus tard.

Remerciements

Je voudrais d'abord remercier le bon Dieu, pour sa grâce par laquelle je vis chaque jour, qui m'a permis d'accomplir ce modeste travail.

Je remercie énormément Mr Daoui Mehammed non seulement pour son travail d'encadreur mais surtout pour ses efforts dévoués pour aider tous ses étudiants durant ces dernières années. Ainsi que d'avoir toujours été à mon écoute.

Je remercie Mr Neddaf Mehdi Directeur HSIC, de m'avoir permis de faire ce stage tellement enrichissant au sein de Mobilis. Et de m'avoir encouragé dès notre première rencontre qui était fortuite.

Je remercie Mme Ibouhmadouche Imene, notre ingénieur encadreur au sein de Mobilis, pour tous ses efforts à subvenir à nos besoins, ainsi quepour ses conseils précieux et ce malgré une charge de travail énorme.

Je remercie également Mr Jean Michel Meulien Pour son accompagnement constant durant notre projet de fin de cycle, son aide fut extrêmement précieuse.

Je remercie Mme Aoudjit Rachida d'avoir accepté de présider ma soutenance, ainsi que d'avoir été un agréable professeur pour moi.

Je remercie Mr Ramdane Mohamed Chef du Département, pour avoir accepté de prendre part à ma soutenance, ainsi que pour votre aide durant notre projet de fin cycle, mais aussi pour la relation de travail agréable durant ces années en faisant face aux différents soucis quotidien au département.

Merci à mes camarades avec lesquelles j'ai collaboré sur ce travail, dans des conditions qui n'ont pas été toujours simples. Et Merci mes camarades de promotion pour les moments mémorables passés ensemble ces dernières années.

Pour finir, une pensée chaleureuse à tous ceux qui m'ont aidé de près ou de loin, à tous ceux qui m'ont offert une leçon de vie marquante ou un geste de gentillesse isolé, ainsi qu'aux personnes appréciées qui ne sont plus.

A

ACL Access Control List

B

BSS Business Support System

C

CDN Content Delivery Network

CPE Customer Premises Equipment

CPU Central Processing Unit

D

DAS Direct Attached Storage

DHCP Dynamic Host Configuration Protocol

DNS Domain Name System

DPI Deep Packet Inspection

E

EC2 Elastic Compute Cloud

EFS Encrypting File System

ETSI European Telecommunications Standards Institute

F

FTP File Transfer Protocol

H

HTTP HyperText Transfer Protocol

IaaS Infrastructure as a Service

ICMP Internet Control Message Protocol

IP Internet Protocol

IT Information Technology

L

LDAP Lightweight Directory Access Protocol

LUN Logical Unit Numbers

M

MAC Media Access Control

MP3 Mpeg Audio Layer 3

N

NAS Network Attached Storag

NF Network Function

NFS Network File System

NFV Network Function Vitualisation

NFVI Network Function Vitualisation Infrastructructure

NIB Network Information Base

NFV-MANO Network Function Vitualisation Managent and Orchestration

NFVO Network Function Vitualisation Orchestration

NTP Network Time Protocol

O

OSI Open Systems Interconnection

OSS Operations Support System

P

PaaS Plateform as a Service

PDF Portable Document Format

PoP Point OF Presence

Q

QoS Quality of Service

R

RAM Random Access Memory

S

SaaS Software as a Service

SDN Software Sefined Network

SECaaS Security as a Service

SIEM Security Information Management System

SMB Server Message Block

SMTP Simple Mail Transfer Protocol

SNMP Simple Network Management Protocol

SPOF Single Point Of Failure

SSH Secure Shell

STORaaS Storage as a Service

T

TCP Transmission Control Protocol

TFTP Trivial File Transfer Protocol

TIC Technologies de l'information et de communication

TLS Transport Layer Security

V

VDU Virtual Deployment Unit

VIM Virtualized Infrastructure Manager

VLAN Virtual Local Area Network

VM Virtual Machine

VNF Virtualized Network Function

VNFD VNF Desciptor

VNFM Network Function VitualisationManagent

VPN Virtual Private Network

W

WAN Wide Area Network

Chapitre I : NFV : Origine, Concepts et Etat de l'art

Chapitre II :Openstack et projet Tacker

Chapitre III : Service de Chainage NFV

Chapitre IV : Implémentation et Réalisation

"Ne faites pas en mieux, mais différemment !" - Steve Jobs. Pourquoi cette citation ? Eh bien le progrès ne s'arrête jamais ! A défaut de ce que pourrait penser la plupart des gens, le progrès ne s'agit pas toujours d'améliorer, d'optimiser et de pousser à la limite ce qui existe déjà, mais le progrès est aussi et surtout l'innovation, l'apport de nouvelles idées, de nouveaux concepts différents pour entrouvrir une nouvelle voie.

A l'heure où les technologies de l'information et de la communication sont indissociables de pratiquement tous les aspects de la vie humaine, que ça soit l'aspect social, culturel ou économique, ces derniers font face à des challenges majeurs, notamment dans l'industrie. En effet, pour pouvoir être compétitives les entreprises doivent encore à ce jour impérativement posséder leur propres infrastructures, ces dernières sont souvent très encombrantes et couteuses. Le Cloud Computing a permis de répondre à ce problème en faisant en sorte de fournir les besoins en infrastructure comme si il s'agissait d'un service de commodité.

Cependant, le problème que pose un matériel dédié et spécialisé pour fournir ces ressources est un frein majeur. Et cette fois-ci l'innovation réside dans l'idée de dissocier le matériel du programme s'exécutant dessus, dans le but de standardiser le matériel pour ainsi rendre flexible le déploiement et la prestation de services à grande échelle. C'est ce à quoi aspire ce nouveau paradigme qu'est le NFV qui est encore loin du concret.

Notre projet de fin d'étude dans le cadre de l'obtention du Master en Réseaux mobilité et systèmes embarqués, a pour but la conception d'une stratégie de chainage de fonctions virtuelles pour le déploiement de service NFV au sein d'ATM Mobilis.

La structure de notre projet comporte quatre grands axes. Le premier traite des limites des infrastructures traditionnelles, le développement des concepts du cloud computing, du NFV et du SDN. Puis dans le second, met en avant la plateforme Openstack et le service Tacker, des outils incontournables dans la concrétisation futur du NFV. Par la suite, le cœur du projet qu'est le chainage de fonctions virtuelles. Pour finir, le quatrième est consacré à l'implémentation et la réalisation de la solution, avant de terminer par une conclusion générale sommant le travail accompli.

Chapitre I:NFV : Origine, Concepts et Etat de l'art

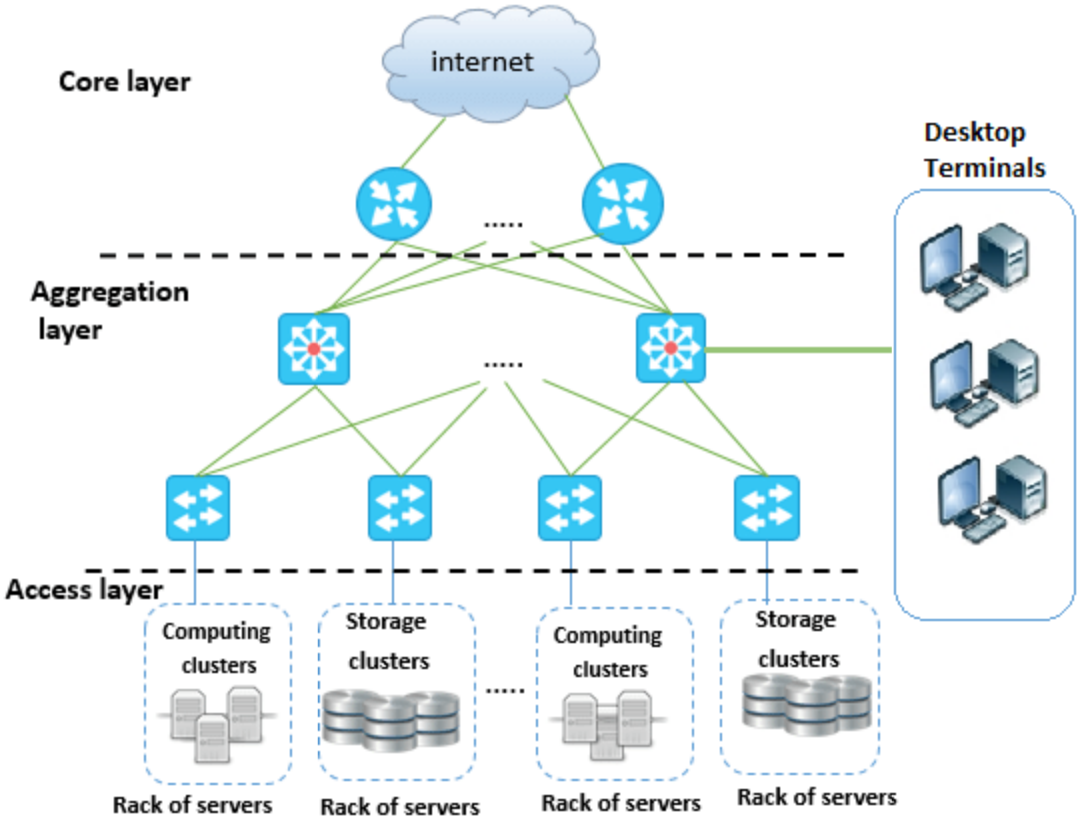
Intorduction

Ce chapitre est consacré à l'émergence d'une technologie révolutionnaire qu'est le NFV, et les challenges et problématiques auxquels elle répond ou tente de répondre. Ainsi que les points communs qu'elle possède avec d'autres domaines parallèles comme le Cloud Computing et le SDN, et se conclue par le stade actuel d'avancement du NFV.

1. Les infrastructures traditionnelles et leurs limites

Depuis leur apparition les réseaux traditionnels ont été synonymes d'équipements spécialisés imposants et propriétaires, nous insistons ici sur les deux termes "spécialisés et propriétaires" car ces derniers et par rapport aux concepts développés par la suite nous expriment tout le sens de la thématique.

Bien qu'avec le progrès considérable qu'a connu l'électronique intégrée ayant permis la réduction des tailles imposantes des équipements, n'en demeure cependant que ces installations avaient toujours et ont jusqu'à ce jour besoin d'espace pour leur aménagement, ces équipements sont le plus souvent des serveurs occupant plusieurs fonctions tels que du networking, le stockage, les applications de travail ...etc. Leur aménagement seul n'étant bien évidement pas suffisant, l'interconnexion de ces derniers avec les postes de travail étant la face cachée de l'iceberg, en effet pour une installation moyenne quelques fois plusieurs kilomètres de fils pour interconnecter les machines sont nécessaires en plus de l'ajout de switchs et de routeurs, d'où l'encombrement que cela crée, en plus des travaux possibles pour réaménager ces câbles, ce qui nous amène aux couts assez conséquents que comporte l'aménagement d'un data center sur ou hors site. A cela s'ajoute un contrôle strict de sécurité de l'installation en termes de climatisation, système de prévention contre l'incendie, une alimentation d'urgence et redondante.



Complexité des Data centers standards.

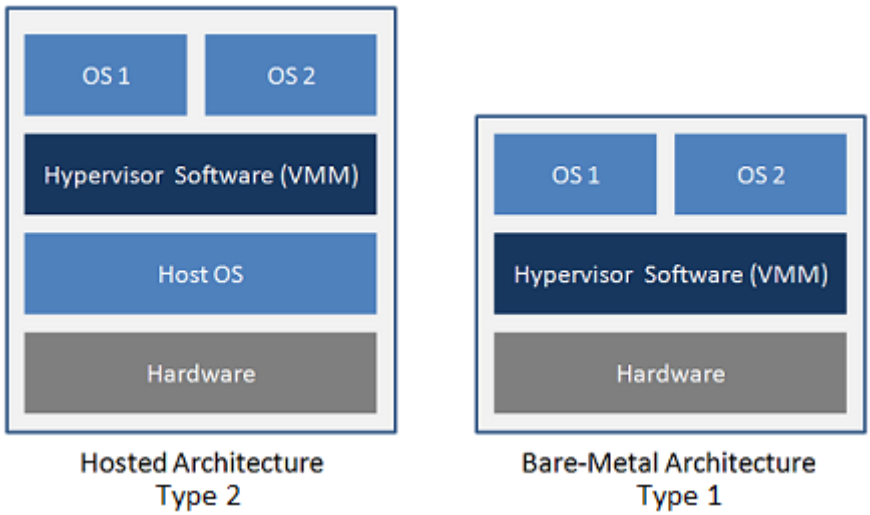
Et pendant longtemps les entreprises ne pouvaient contourner cette situation. Et bien souvent c'était un investissement couteux, très risqué que ne pouvaient se permettre que ceux possédant un capital important. En plus les coûts et l'espace ne sont pas les seules contraintes. En effet, si l'aménagement d'un Datacenter permettait de pouvoir poser une infrastructure pour les dix prochaines années, on pouvait se dire que l'investissement serait hautement bénéfique et sûr dans le temps, or avec les technologies de l'information qui évoluaient et continuent d'évoluer de jour en jour, il ne fallait pas longtemps avant que le besoin d'amélioration, et donc de réaménagement ne se fasse ressentir. La contrainte de flexibilité étant la plus problématique, de nouvelles technologies voient le jour et ne s'accorde pas forcément avec les anciennes, les standards sont constamment mis à jour, et les marchés sont en constante évolution, de nouvelles demandes apparaissent. Pour faire face à ces challenges, une adaptabilité rapide est indispensable et ne peut être fournie par les équipements traditionnels. En revenant au terme de "spécialisé", cité ci-dessus, nous comprenons que chaque fonction s'exécute sur son propre équipement dédié, et donc pour chaque fonction ou service qu'on veut déployer il nous faut acquérir son hardware, si jamais un appareil tombait en panne, une partie de l'activité interne ou externe ou même les deux de l'entreprise peuvent être affectées, pouvant occasionner des pertes et des frais de réparation. Toutes ces raisons ont poussé l'industrie de la télécommunication et des technologies de l'information à repenser son fonctionnement et à rechercher de nouveaux

paradigmes, afin de s'abstraire de ces contraintes de cout et de flexibilité ou à défaut, de pouvoir les réduire considérablement. C'est ainsi qu'est apparue le concept de la virtualisation qui sera vu dans le point suivant.

2. La Virtualisation

La virtualisation est un mécanisme informatique qui consiste à faire fonctionner plusieurs systèmes, serveurs ou applications, sur un même serveur physique. La virtualisation est un composant technique clé dans le Cloud Computing et dans la NFV [3].

Le principe général est qu'un logiciel de virtualisation (appelé « hyperviseur ») est installé sur le système d'exploitation principal ou bien hôte (hosted). On dit alors qu'il est de type 2. S'il est installé à même la machine (bare metal) on dit alors qu'il est de type 1 ou natif. Il permet la création d'environnements clos et indépendants sur lesquels seront installés d'autres systèmes d'exploitation (« systèmes invités »). Ces environnements sont des « machines virtuelles ». La figure ci-dessous illustre très bien ce concept.



Principe de la virtualisation.

Avantages de la virtualisation

La virtualisation offre les avantages suivants :

- consolidation et rationalisation d'un parc de serveurs en entreprise : les entreprises ne sont plus obligées d'acheter un serveur physique pour chaque application,
- rationalisation des coûts de matériels informatiques,

- possibilité d'installer plusieurs systèmes (Windows, Linux) sur une même machine,
- portabilité des serveurs : une machine virtuelle peut être déplacée d'un serveur physique vers un autre (lorsque celle-ci a, par exemple, besoin davantage de ressources),
- accélération des déploiements de systèmes et d'applications en entreprise,
- mise en place d'un Plan de retour d'activité rapide en cas d'incident, ou de panne,
- administration simplifiée de l'ensemble des serveurs,
- réduction de la facture d'électricité, en diminuant le nombre de serveurs physiques.

3. La NFV (Network Funtion Virtualization)

3.1. Description

L'idée principale derrière le NFV est la séparation des équipements du réseau physique des fonctions qui s'exécutent sur ces derniers. NFV promet par exemple des TSPs (Telecom Service Providers) avec plus de flexibilité pour permettre encore plus l'ouverture de leurs performances réseau et de leurs services à leurs utilisateurs ou à d'autres clients potentiels, avec la possibilité de déployer ou de supporter de nouveaux services réseaux plus et moins chers afin d'atteindre une meilleure agilité dans les services. Pour réaliser ces prouesses, le NFV trace le chemin pour une multitude de différences dans la façon dont l'approvisionnement des services réseaux est réalisé en comparaison à la pratique actuelle [3]. En résumé, ces différences sont les suivantes :

a. Découplage du software et du hardware

Comme le réseau n'est plus une composition d'appareils physiques (hardware) intégrés et d'entités logicielles, l'évolution des deux est indépendante l'une de l'autre. Cela permet des chronologies de développement et de maintenance séparées pour le matériel physique et le logiciel.

b. Déploiement flexible de fonctions réseaux

Le détachement du software du hardware contribue au ré assignement et partage des ressources de l'infrastructure, ainsi, tous les deux, hardware et software, peuvent exécuter différentes fonctions à différents moments. Cela permet à l'operateur réseau de déployer de nouveaux services réseaux plus rapidement par-dessus la même plateforme physique. Par conséquent, des composants peuvent être instanciés dans n'importe quel appareil permettant le NFV dans le réseau et leur connections peuvent être configurées d'une manière flexible.

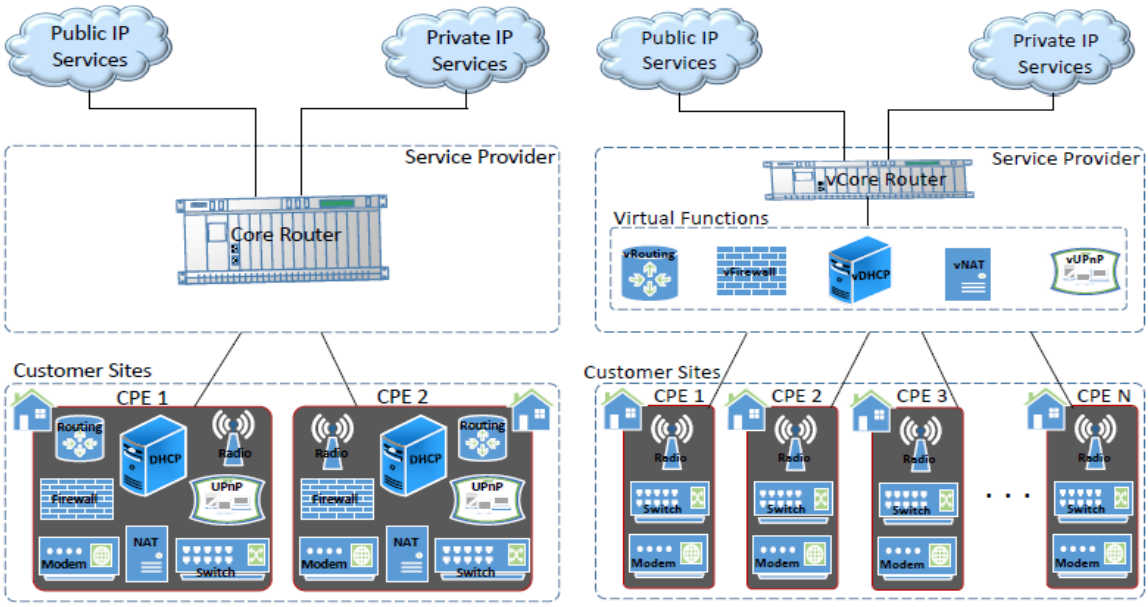
c. Le scaling dynamique

Le découplage de la fonctionnalité des fonctions réseau en composants logiciels instanciables fournit une plus grande flexibilité d'adaptation de la performance réelle (scaling) de la VNF d'une manière plus dynamique avec sa granularité plus affinée. Par exemple, selon le trafic courant pour lequel l'opérateur réseau a besoin d'approvisionnement de capacité. Cela vaut le coup de faire remarquer que le concept général du découplage des NFs (network functions) du hardware dédié ne demande nécessairement pas la virtualisation des ressources. Cela signifie que les TSPs pourraient toujours faire l'achat ou développer des NFs et les exécuter sur des machines physiques. La différence est que ces NFs devraient être capable de s'exécuter sur des serveurs basiques. Toutefois, les gains (telle que la flexibilité, scalabilité dynamique des ressources, la gestion des réserves d'énergie) de l'exécution de ces fonctions sur des ressources virtuelles sont des arguments de vente très convainquant de la NFV. Sans besoin de mentionner qu'il est également possible d'avoir des scénarios hybrides ou des fonctions s'exécutant sur des ressources virtuelles co-existantes avec celles s'exécutant sur des ressources physiques. De tels scénarios hybrides sont d'une importance cruciale durant la transition vers le NFV.

3.2. Cas d'utilisation de la NFV

l'ETSI a proposé un nombre de cas d'utilisation pour la NFV. Nous expliquerons dans ce qui suit comment la NFV pourrait être appliquée à l'équipement clients (Customer Premises Equipment) (CPE), mais aussi à un réseau Evolved Packet Core (EPC) [3] :

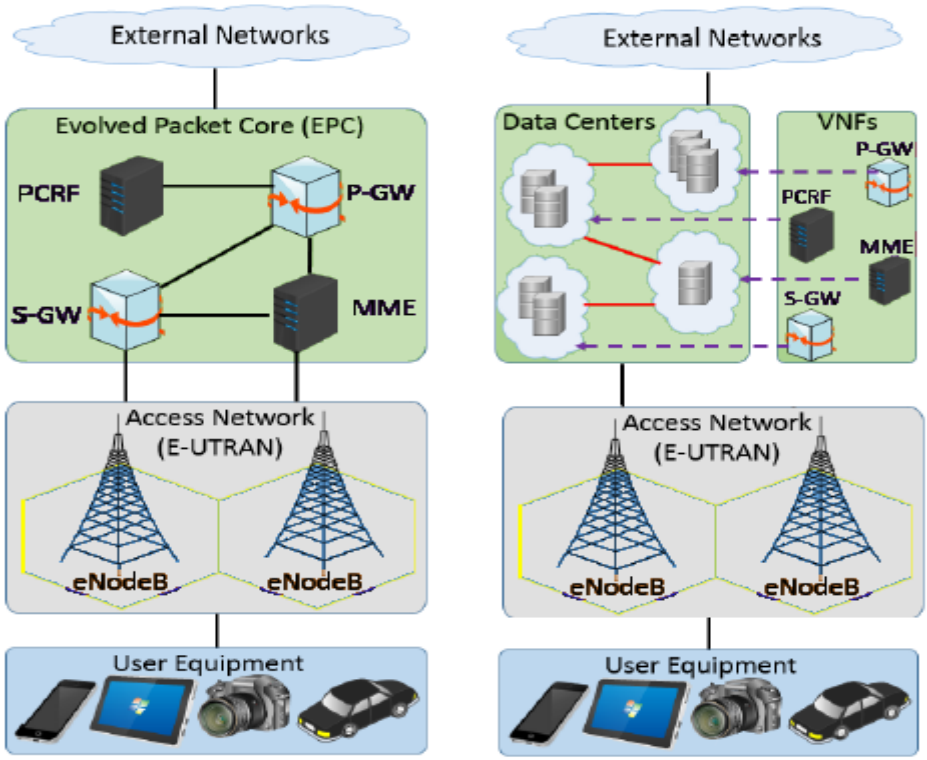
a. Equipments clients - Customer Premises Equipment (CPE)



Virtualisation du service CPE.

Une implémentation typique (actuelle) d'un CPE est faite des fonctions suivantes: Dynamic Host Configuration Protocol (DHCP), Network Address Translation (NAT), routing, Universal Plug and Play (UPnP), Firewall, Modem, radio et le switching. Dans cet exemple, un seul service (CPE) est constitué de huit fonctions. Ces fonctions peuvent avoir des conditions de priorité. Par exemple, si les fonctions font partie de chaine de service, il se peut que qu'il faille exécuter le firewall avant le NAT. Actuellement, il est nécessaire d'avoir ces fonctions sur des appareils physiques situés dans les locaux des clients. Avec une telle implémentation, s'il y avait le besoin d'appliquer des changements dans le CPE, en ajoutant, supprimant ou mettant à jour une fonction, il se peut qu'il soit nécessaire qu'un technicien de l'ISP (Internet Service Provider) aille chez chaque client individuellement. Il est même possible d'avoir besoin d'un changement complet d'un appareil en cas d'ajouts. Non seulement ce serait cher pour les ISPs, mais aussi pour les clients. Une possible implémentation basée sur la NFV dans laquelle certaines fonctions du CPE sont transférées vers une infrastructure partagée au sein de l'ISP, qui pourrait aussi être un Datacenter. Ceci rend les changements cités précédemment plus simples puisque, par exemple, la mise à jour du DHCP pour tous les clients ne concernerait que des changements au niveau de l'ISP. De la même manière, l'ajout d'une nouvelle fonction comme le contrôle parental pour tout ou une catégorie de clients pourrait être fait en une seule fois. En addition à la réduction des coûts opérationnels pour les ISPs, cela mènera potentiellement à des CPEs moins chers à grande échelle.

b. Evolved Packet Core

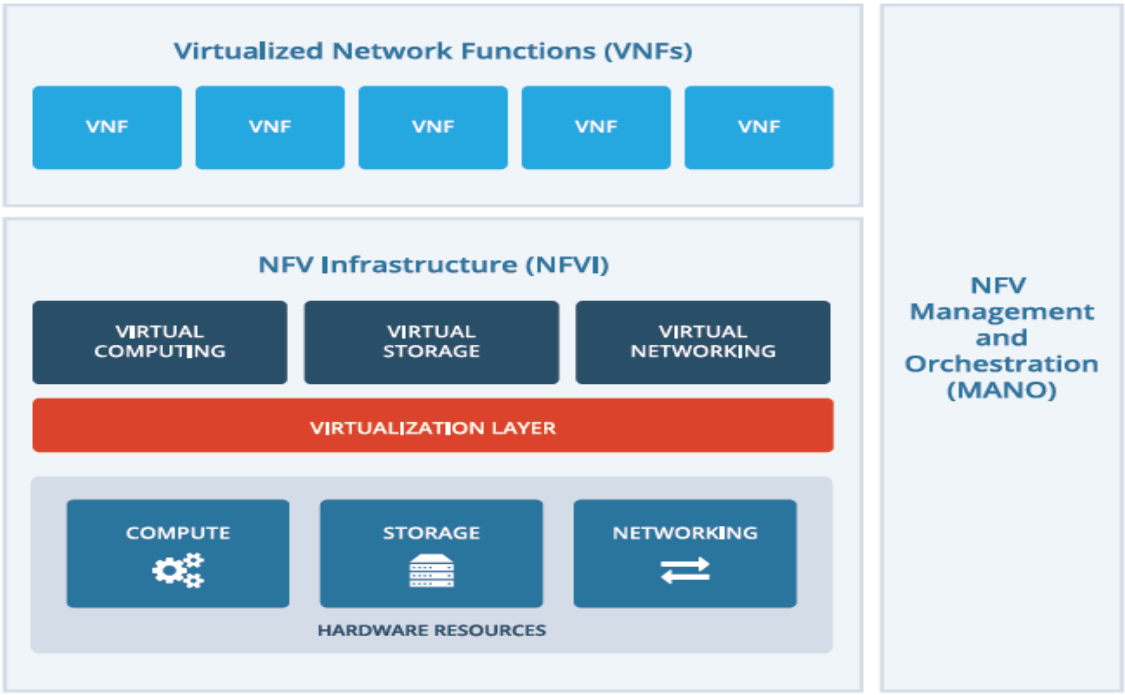


Virtualisation de l'EPC.

La virtualisation de l'EPC est un autre exemple de la NFV qu'a attiré beaucoup d'attention de la part de l'industrie. L'EPC est le réseau noyau pour la norme Long Term Evolution (LTE) comme il est spécifié par le consortium 3GPP. Du côté gauche de la figure ci-dessus, nous voyons une architecture basique du LTE sans la NFV. L'appareil utilisateur (UE) se connecte à l'EPC via l'accès réseau LTE (E-UTRAN). Le (eNodeB) est la station de base de la radio LTE. L'EPC exécute des fonctions essentielles incluant le pistage des abonnées, gestion de la mobilité et de la session. Il se constitue de quatre NFs: Serving Gateway (S-GW) - passerelle de desserrement, Packet Data Network (PDN) Gateway (PGW) - passerelle du réseau de paquets de données, Mobility Management Entity (MME) - entité de gestion de mobilité, et du Policy and Charging Rules Function (PCRF) - règles et politique de tarification. Il est aussi connecté à des réseaux externes, qui peuvent inclure le sous-système réseau de l'IP multimediacore -IP MultimediaCore Network Subsystem (IMS). Dans l'EPC actuel, toutes ses fonctions sont basées sur des équipements propriétaires. En conséquence, même des changements mineurs sur une fonction donnée pourraient requérir le remplacement de l'équipement. La même chose s'applique quand la capacité des équipements doit être changée. Du côté droit de la figure, la même architecture est mise en avant dans laquelle cette fois ci l'EPC est virtualisé. Dans ce cas, toutes ou une partie les fonctions sont transférées dans une infrastructure partagée (cloud). La virtualisation de l'EPC pourrait potentiellement mener vers une meilleure flexibilité et un scaling dynamique, et ainsi permettre aux TSPs de répondre facilement, de façon moins onéreuse, aux conditions du marché. Par exemple, il pourrait y avoir un besoin d'augmentation des user plane ressources - plan d'utilisateur des ressources sans affecter le plan de control. Dans ce cas, les VNFs tel qu'un MME virtuel pourrait scaler (adapter ses performances aux ressources) indépendamment selon leurs conditions de ressources spécifiques. De la même façon, les VNFs qui traitent avec le modèle de données pourrait requérir un nombre différent de ressources que ceux qui traitent avec le signal seulement. Cette flexibilité mènerait vers une utilisation plus efficace des ressources. Pour conclure, cela permet aussi une meilleur mise à jour des logiciels vers des versions améliorées dans le réseau de fonctions EPC, et ainsi permettre des lancements plus rapides des services innovants.

3.3. L'architecture NFV

Selon l'ETSI(Institut Européen de la Standardisation des Télécom), l'architecture NFV est composée de trois éléments clés : Network Function Virtualization Infrastructure (NFVI), les VNFs et NFV MANO [3].



Architecture NFV.

a. NFV Infrastructure (NFVI)

La NFVI est la combinaison à la fois des ressources du hardware et du software qui constitue l'environnement dans lequel les VNFs sont déployés. Les ressources physiques incluent le hardware, le stockage et le réseau (fait de nœuds et de liens) qui fournit le traitement, le stockage et la connectivité aux VNFs. Les ressources virtuelles sont l'abstraction des ressources de calcul, de stockage et du réseau. L'abstraction est réalisée en utilisant une couche de virtualisation (basée sur un hypervisor), qui découple les ressources virtuelles des ressources physiques de la couche inférieure. Dans un environnement de data center, les ressources de computing et de stockage peut être représentés par une ou plusieurs VMs, alors que les reseaux virtuels sont fait de virtual links et de nœuds. Un nœud virtuel est un composant logiciel doté soit la fonctionnalité du hosting ou du routing. Par exemple, un système d'exploitation encapsulé dans une VM. Un lien virtuel est une interconnexion logique de deux nœuds virtuels, apparaissant à ces derniers comme un lien physique direct avec des propriétés dynamiquement paramétrables.

b. Les Fonctions Réseau Virtuelles et les Services VNFs

Une fonction réseau(NF) est un bloc fonctionnel dans une infrastructure réseau qui possède des interfaces externes très bien définies et dont le comportement fonctionnel très bien défini également. Des exemples de NFs sont des éléments d'un réseau domestique,e.g. Une passerelle résidentielle - Residential Gateway (RGW) ; et les fonctions réseaux conventionnelles, e.g. serveurs DHCP, des pare-feu, etc. Par conséquent, un VNF est une implémentation d'une NF déployée sur une ressource virtuelle telle qu'une VM. Une seule VNF peut être composée d'une multitude de composants internes, et par conséquent peut être déployée sur plusieurs VMs, dans lequel chaque VM est l'hôte d'un seul composant de la VNF. Un service est une offre fournie par un TSP, cette dernière est composée d'une ou plusieurs NFs. Dans le cas du NFV, les NFs qui constituent le service sont virtualisées et déployées sur des ressources virtuelles telles qu'une VM. Cependant, dans la perspective de l'utilisateur, les services ; qu'ils soient basés sur des fonctions s'exécutant sur des équipements dédiés ou sur des VMs doivent avoir la même performance. Le nombre, type et ordre des VNFs sont déterminés par la spécification fonctionnelle et comportementale du service. Et donc, le comportement d'un service est dépendant de celui des VNFs le constituant.

c. NFV Management and Orchestration (NFV MANO) - gestion et orchestration

Selon le framework MANO de l'ETSI, le NFV MANO fournit la fonctionnalité requise pour l'approvisionnement des VNFs, et les opérations en relation avec ce dernier, telle que la configuration des VNFs et de l'infrastructure sur laquelle elles s'exécutent. Cela inclue l'orchestration et la gestion du cycle de vie des ressources physiques ou logiciels qui supportent l'infrastructure de virtualisation, et la gestion du cycle de vie des VNFs. Cela inclue aussi les bases de données qui sont utilisées pour le stockage des informations et des modèles de données qui définissent à la fois le déploiement et les propriétés du cycle de vie des fonctions, des services et des ressources. NFV MANO se concentre sur toutes les tâches de gestion spécifiques à la virtualisation nécessaire dans le NFV framework. De plus le framework définie des interfaces qui peuvent être utilisées pour les communications entres des composants du NFV MANO, aussi bien que la coordination avec les systèmes de gestion des réseaux traditionnels tels que Operations Support System (OSS) et Business Support Systems (BSS) dans le but de permettre le management des VNFs aussi bien que des fonctions s'exécutant sur des équipements existants.

Il existe toujours des questions en suspens, précisément en ce qui concerne le support de l'hétérogénéité. Il est observable à travers les solutions NFV actuelles que les fournisseurs divergent sur ce qui constitue une NFVI et des VNFs: (1) quelles NFs devraient être déployées dans des nœuds de data center, et lesquelles dans des nœuds d'opérateurs;

(2) quelles fonctions devraient être déployées sur des VMs dédiées et lesquelles sur des conteneurs; (3) quelle est le type et la quantité de ressources NFVI que requière l'exécution de fonctions spécifiques; et (4) les exigences opérationnelles des environnements qui incluent et les VNFs et les fonctions s'exécutant sur des équipements existants.

3.4. Les points à prendre en considération dans la conception NFV

La NFV ne sera une solution acceptable pour les TSPs que si elle respecte certains points essentiels identifiés ci-dessous:

a. Architecture réseau et Performance

Dans le but d'être acceptable, les architectures NFV devraient être capables d'atteindre des performances au moins similaire à celles obtenues des fonctions exécutées sur hardware dédié.

b. Sécurité et Résilience

La nature dynamique de la NFV exige que les technologies, politiques et processus de sécurité soient incorporés. En particulier, il y a deux risques de sécurité importants qui doivent être considérés dans la conception NFVI: (1) les fonctions ou services de différents abonnés doivent être protégées/isolées les unes des autres. Cela aide à garantir que les fonctions soient résistantes aux erreurs et aux attaques puisque un échec ou une brèche sécuritaire dans une fonction ou service n'affecterais pas les autres. (2) la NFVI devrait être protégée des services d'abonnement offerts. Une façon de sécuriser la NFVI est le déploiement interne de pare-feux au sein même de l'environnement virtuel. Tout cela permettrait au NFV MANO l'accès aux VNFs sans laisser passer le trafic malicieux provenant des réseaux clients dans la NFVI.

c. Fiabilité et disponibilité

Tandis que dans le domaine de l'IT des pannes ne durant que quelques secondes sont tolérables et un utilisateur typiquement n'a besoin que de réinitialiser, les télécommunications sont des exigences de services qui permettent que les pannes soient en dessous du reconnaissable (de l'ordre des millisecondes).Le rétablissement du service doit être automatique. De plus, des services impactant des pannes doivent être limités à un certain nombre d'utilisateurs (certaines géographie).Des pannes sur un réseau à grande échelle ne sont pas du tout acceptable. Ces conditions extrêmes de fiabilité et de disponibilité ne sont pas que des exigences clients, mais souvent aussi celles des organisations régulatrices car les TSPs sont considérés comme des infrastructures nationales critiques.

d. Support pour l'hétérogénéité

L'argument de vente principale de la NFV est basé sur le fait de casser les barrières créées par les services sur hardware propriétaire. Il en va sans dire que l'ouverture et l'hétérogénéité sera au cœur du succès de la NFV. Une plateforme NFV se doit d'être un environnement ouvert, partagé et capable d'exécuter des applications de différents fournisseurs. Les fournisseurs de service doivent être libres de choisir le hardware qui les convient, changer de fournisseurs, et traiter avec l'hétérogénéité du hardware sans restrictions. Une telle plateforme doit protéger les VNFs des spécifications technologiques des couches réseau physiques inférieures. Ces plateformes doivent permettre la possibilité de construite des end-to-end services par-dessus plusieurs infrastructures sans problèmes, sans le besoin de solutions technologiques spécifiques.

e. Prise en charge héritée

La prise en charge pour à la fois les NFs physiques et virtuelles est importante pour les opérateurs faisant la transitions vers la NFV tant ils peuvent avoir besoin de gérer des atouts physiques existants aux cotés de fonctions virtuelles pour une durée indéterminée. Cela peut nécessiter une stratégie d'orchestration qui réduit l'écart entre les services hérités et la NFV.

f. Scalabilité du réseau et l'automatisation

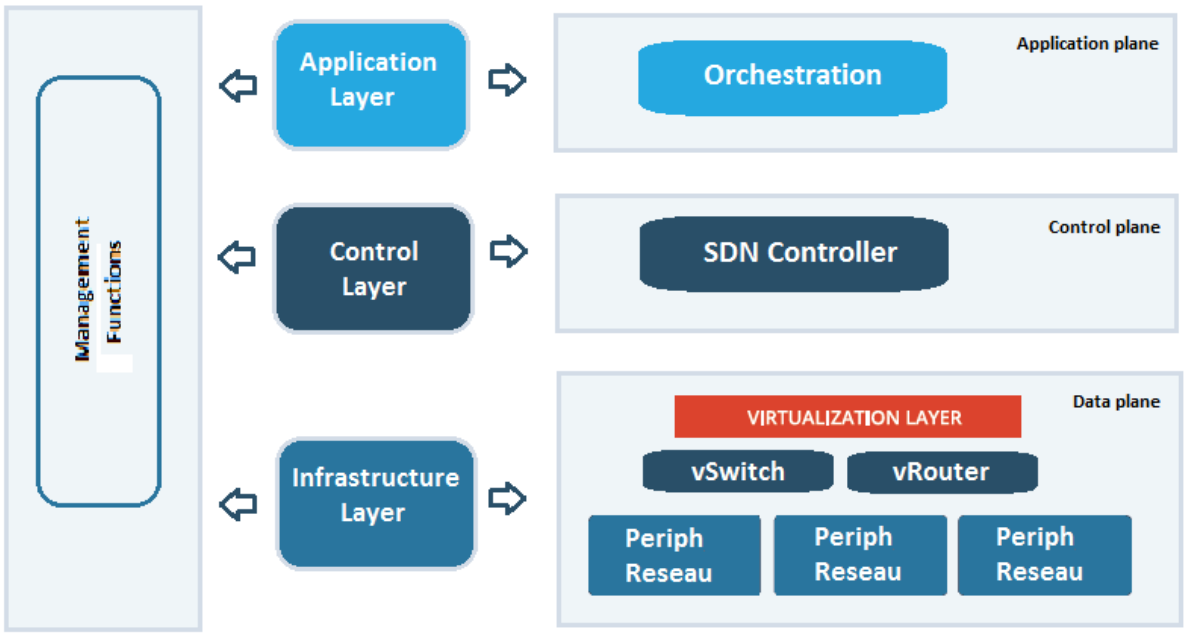
Dans le but d'atteindre tous les objectifs bénéfiques de la NFV, une solution réseau scalable et responsive est nécessaire. Pendant que les conditions de conception ci-dessus sont respectées, la NFV doit être acceptablement scalable pour supporter des millions d'abonnés. Pour donner un exemple, la plupart des concepts NFV actuels ayant fait leurs preuves sont basées sur le déploiement d'une VM pour héberger une VNF. Tout comme une seule VM peut ne pas être capable de prendre en charge certaines fonctions, il n'est pas non plus économique de déployer une VM par VNF, car le résultat serais un pool de VM trop large et mènerais a des problèmes de scalabilité au niveau de la couche de virtualisation. Cependant, la NFV ne peut être scalée que si toutes les fonctions peuvent être automatisées. Ainsi, l'automatisation des procédures est d'une importance capitale dans le succès de la NFV. De plus, le besoin d'environnements dynamiques exige que les VNFs soient déployées et retirées à la demande, et scalées pour répondre au trafic changeant.

4. Concepts communs : Cloud computing et SDN

4.1. Software Defined Networking (SDN)

Le réseau basée ou définit par logicielle (ou SDN) désigne un ensemble de technologies innovantes et évolutives visant à permettre un contrôle centralisé des ressources réseau, une meilleure programmabilité et une orchestration de ces ressources, idéalement adaptée aux bandes passantes élevées associées aux applications modernes [4].

Les SDN ont pour but pratique de rendre programmables les réseaux par le biais d'un contrôleur centralisé. Avec ces derniers, on établit une séparation claire entre le plan de contrôle (qui définit comment un équipement forwarde le trafic) et le plan de données (la partie des commutateurs et routeurs qui assure effectivement le mouvement des données) ce qui rend cette infrastructure sous-jacentes une simple voie véhiculant les paquets réseau et programmable à volonté. Dans les SDN, le plan de contrôle est placé dans un contrôleur centralisé qui a une visibilité sur l'ensemble du réseau, y compris les hôtes qui s'y connectent. Il dispose d'une vision complète de la topologie du réseau [9].



Architecture SDN.

La NFV et le SDN ont beaucoup en commun et sont hautement complémentaires puisque les deux militent en faveur du passage à l'open software et un hardware réseau standard. La NFV et le SDN cherchent à tirer parti de l'automatisation et de la virtualisation afin de réaliser leurs objectifs respectifs. Mais ces derniers résolvent différents problèmes dans différents environnements à travers différents domaines. NFV a pour but d'accélérer l'innovation et le déploiement et la fourniture rapide et flexible des services en utilisant des technologies de virtualisation standard. NFV ne dépend pas du SDN et vice-versa, pourtant les deux concepts et solutions peuvent être combinés pour un plus grand potentiel [3].

4.2. Cloud Computing

Le cloud computing ou informatique en nuage est une infrastructure permettant l'accès à des services hébergés qui sont gérés par des serveurs distants auxquels les usagers se connectent via une liaison Internet sécurisée.

Le Cloud computing permet aux compagnies ou aux particuliers l'accession à des ressources de calcul, telles que les machines virtuelles VMs, le stockage ou des applications, de manière utilitaire comme l'électricité au lieu d'avoir à aménager et entretenir les infrastructures sur sites. Ces services peuvent être rapidement fournis et déployés avec des efforts de gestion ou d'interaction avec le fournisseur minimes [9].

Le modèle du cloud ou nuage est composé de cinq caractéristiques essentielles et de trois modèles pour les services principaux. Ces derniers sont introduits brièvement dans sous-sections suivantes :

a.Caracteristiques essentielles du Cloud Computing

- Self-service ou service libre à la demande.
- Large accès au réseau.
- Mise en commun des ressources.
- Elasticité rapide.
- Service mesuré.

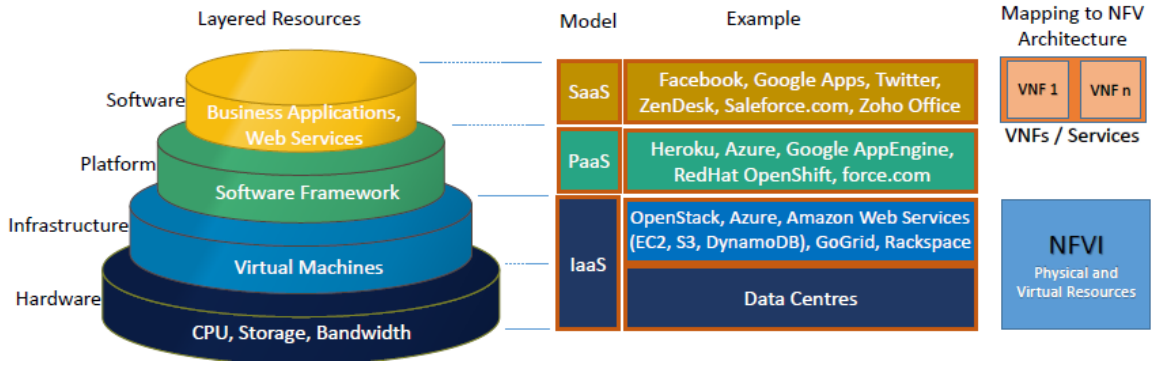
b.Modèles principaux de services du Cloud Computing

- Software as a Service (SaaS) - Logiciel en tant que service.
- Platform as a Service (PaaS) - Système d'exploitation en tant que service.
- Infrastructure as a Service (IaaS) - Infrastructure physique en tant que service.

Toutefois, il existe plusieurs autres modèles déjà en phase d'exploitation comme le SECaaS qui la sécurité en tant que service, ou de développement comme BPaaS qui est pour le Business Process autrement dit les affaires mais n'en n'est qu'au stade embryon néanmoins.

c. Relation entre le Cloud Computing et la NFV

En général, la NFV ne se restreint pas qu'aux fonctions pour des services dans les télécommunications. En fait, plusieurs applications IT s'exécutent déjà sur des serveurs de base dans un cloud. Cependant, puisque la plupart des cas d'utilisation les plus prometteurs de la NFV trouvent leur origine dans l'industrie des télécommunications, et parce que les performances et les exigences en fiabilité des fonctions dites carrier grade (extrêmement fiable) sont plus hautes que celles des applications IT, il est donc considéré qu'une performance NFV acceptable [3].



Comparaison entre l'architecture du cloud et de la NFV.

Dans la figure ci-dessus, les modèles de services du cloud sont mappés de façon à être comparés avec l'architecture NFV. On peut observer que le modèle IaaS correspond aux ressources physiques et virtuelles dans la NFVI, tandis que les services et les VNFs dans la NFV sont similaires au modelé de service SaaS dans le cloud computing.

Etant le choix le moins onéreux pour les tests et l'implémentation, la plupart des concepts NFV ayant fait leurs preuves ainsi que les premières implémentations ont été basés sur le déploiement de fonctions sur des VMs dédiés dans le cloud. La flexibilité du cloud computing, incluant un déploiement rapide de nouveaux services, facilement scalables, et une duplication réduite, font de lui le meilleur candidat pour offrir une chance d'atteindre l'efficacité et la réduction de coûts qui motivent les TSPs en faveur de la NFV.

d. Recherches surla NFV basée cloud:

Pour que la NFV fonctionne acceptablement dans un environnement de cloud computing, l'infrastructure sous-jacente se doit de fournir un certain nombre de fonctionnalités allant du scheduling (planification) jusqu'au networking, et de l'orchestration aux capacités de surveillance. Tandis qu'OpenStack a été identifié comme un des composants principaux de la NFV basée cloud, actuellement il ne remplit pas encore certaines exigences. Par exemple, à travers l'analyse d'une faille, il a été noté que parmi d'autres failles, OpenStack ne fournit pas une description détaillée des ressources réseau incluant les conditions du Quality-of-Service (QoS), et ne prend pas en charge un service de réservation de ressource et donc ne fournit aucune interface pour la réservation de ressource.

En plus, à travers des mesures effectuées, certaines dégradations de performances ont été rapportées. Quelques efforts ont déjà été dédiés pour l'étude des conditions nécessaires pour rendre les performances du cloud "carrier-grade". En particulier, OpenANFV propose un framework basée sur OpenStack qui utilise un accélérateur hardware pour améliorer les performances des VNFs. Motivé par l'observation que certaines fonctions (e.g., DPI Deep Packet Inspection, et NAT), les serveurs de standards industriels pourraient ne pas atteindre les niveaux exigés de performance. En conséquence, OpenANFV vise à fournir un approvisionnement élastique et automatique pour l'accélération hardware des VNFs dans OpenStack. A cette fin, les VNFs testées (DPI et NAT) ont eu accès à un ensemble prédéfini de fonctionnements accélérés, et ont pu communiquer à travers une interface indépendante du hardware avec l'hyperviseur pour configurer l'accélérateur. Des performances ont été rapportées 20 et 10 fois plus rapides pour DPI et NAT respectivement [3].

5. Etat de l'art

5.1. Le projet Open Platform pour NFV (OPNFV)

OPNFV est un projet open source fondé et hébergé par la fondation Linux, et composé de TSPs et de fournisseurs. Il a pour but d'établir une plateforme de référence carrier-grade, intégrée et open source afin de faire progresser l'évolution de la NFV et de garantir la régularité, la performance et l'interopérabilité parmi de multiple composants open source. Le premier aboutissement du projet est référé comme OPNFV Arno, et a été sorti en juin 2015. La version fournit une construction initiale des composants de l'architecture ETSI: NFVI et manager de l'infrastructure virtuelle (VIM). Il est fait pour les développeurs, et donc peut être utilisé pour explorer les déploiements NFV, le développement des applications VNF, ou l'évaluation des performances NFV et pour les tests basés sur les cas d'utilisation. En particulier, Arno possède des capacités d'intégration, de déploiement et de test de composants d'autres projets tels Ceph, KVM, OpenDaylight, OpenStack and Open vSwitch. En addition, les end-users et développeurs peuvent déployer leurs propres tierce partie de VNFs sur Arno pour tester ses fonctionnalité et ses performances dans des scenarios de trafic et de cas d'utilisation variés [3].

5.2. Implémentations de la NFV:

a. HP OpenNFV:

Le HP OpenNFV est une plateforme basée sur l'architecture NFV référence de HP, sur laquelle des services et des réseaux peuvent être construits dynamiquement. Cette architecture NFV de HP est alignée vers la fourniture de solutions pour chacun des blocs fonctionnels définis par l'architecture ETSI, comme point de départ. La NFVI ainsi que les VNFs composants de l'architecture incluent principalement des serveurs HP et composants de virtualisation, tandis que MANO est basé sur 3 solutions ; NFV Director, NFV Manager, et Helion OpenStack. Le NFVDirector est un orchestrateur qui gère automatiquement les services end-to-end, en manageant, et allouant les ressources d'un pool approprié basé sur des politiques de management de ressources globales. Les managers de VNFs sont responsables des actions pendant le cycle de vie de ces dernières, par exemple, en décidant du scale-in ou scale-out (baisse ou élévation des performances). Il est aussi inclue une plateforme cloud Helion OpenStack pour l'exécution des VNFs [3].

b. CloudNFV:

CloudNFV ont proposé leur propre architecture NFV qui est constituée de 3 éléments principaux : la virtualisation active, un orchestrateur NFV, un Manager NFV. La virtualisation active est un modèle de données représentant tous les aspects des services, fonctions et ressources. L'orchestrateur VNF possède une politique de réglementation qui, combinée avec l'ordre des services et le statu des ressources disponibles, détermine la localisation des fonctions qui constituent le service ainsi que les connections entre elles. Le Manager VNF utilise un modèle donnée/ressource structuré selon les règles du TMF (forum de standardisation) et le concept des "opérations dérivées" est utilisé pour manager les VNFs. Les opérations dérivées sont utilisées pour l'intégration du statu des ressources disponibles avec les ressources consenties dans des fonctions d'un service NFV donné. La différence principale entre ETSI NFV MANO et CloudNFV est qu'au contraire du premier, ce dernier considère les deux concepts du management et de l'orchestration comme des applications qui peuvent exécuter un modèle de données unifié [3].

c. Openstack Tacker:

Tacker est un projet openstack officiel mettant en place un Manager VNF (VNFM) et un Orchestrateur NFV (NFVO) pour déployer et gérer des services réseaux et des fonctions de réseaux virtuels (VNF) sur une plateforme d'infrastructure NFV telle qu'openstack. Il est basé sur l'architecture du framework MANO de l'ETSI. Ce projet est à la base de notre réalisation dans le cadre de ce mémoire de Master. Il sera étudié de façon plus détaillée dans le chapitre suivant.

Conclusion

Dans ce chapitre nous avons passé en revue le développement des infrastructures de télécommunication depuis les technologies traditionnelles vers de nouveaux concepts révolutionnaires tels que le Cloud Computing, Le NFV ainsi que le SDN. Le chapitre suivant traitera de la plateforme Openstack et le service Tacker.

Chapitre II: Openstack et projet Tacker

Introduction

Ayant vu dans le chapitre précédent, les nouveaux concepts que sont le Cloud Computing et le NFV, ce chapitre traite de la plateforme Openstack ainsi que du service Tacker qui permettent d'implémenter ces concepts, leur composition ainsi leur mode de fonctionnement sont méticuleusement passés en revue.

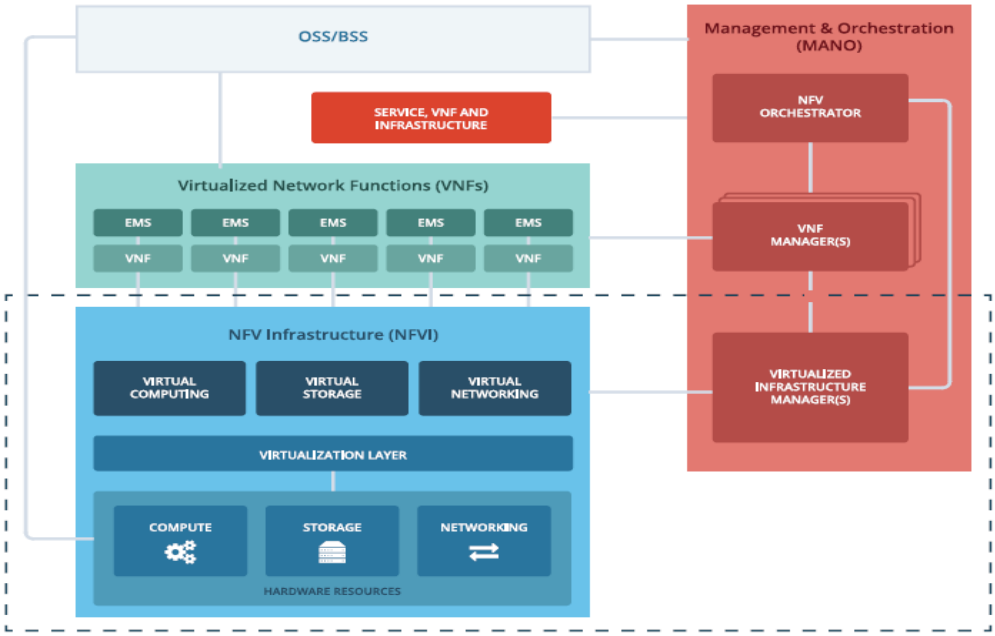
1. La plateforme Openstack:

1.1. Description

OpenStack est un système opératoire libre pour cloud privé ou public de type IaaS qui permet le contrôle de large pools de ressources de calcul (compute), de stockage et de networking à travers un Datacenter géré à partir d'un Dashboard (panneau de bord ou de contrôle). Il permet aux administrateurs de contrôler le cloud tout en donnant libre court aux utilisateurs pour gérer leurs ressources à travers une interface web. Il s'installe sur un système d'exploitation libre comme Ubuntu ou Debian ou propriétaire tel que Windows.

OpenStack joue le rôle d'une couche de management assurant la communication entre la couche physique où se trouvent des serveurs physiques occupés par des hyperviseurs différents (VMware ESX, Citrix Xen, KVM, qemu,...) et la couche applicative (Applications, utilisateurs, administrateurs,…).

Les besoins en ressources IT pour les applications web souvent fluctuent avec les demandes des utilisateurs, de manière prédictible et tout aussi imprédictible, échouer à répondre à la demande dans les deux cas peut impacter la satisfaction des clients et donc des revenus. La capacité d'allocation dynamique des ressources est l'un des bénéfices primaire dans l'utilisation d'un cloud Openstack [1].



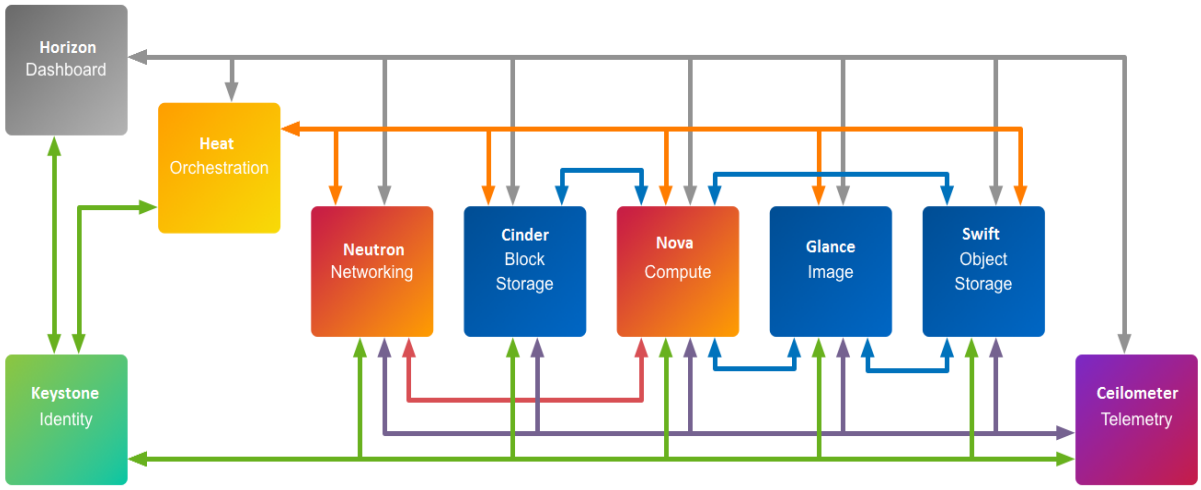
Openstack-TACKER VIM.

Openstack fait office de VIM (virtualized Infrastructure Manager) pour le service NFV Tacker en revenant sur l'architecture NFV présenté illustré dans la figure 5. du chapitre précédent, comme il est expliqué dans la figure ci-dessus. Le VIM est responsable du contrôle et de la gestion de la Network Function Virtualization Infrastructure (NFVI), c'est lui qui fournit les VNFs avec les ressources dont elles ont besoin [6].

1.2. Principaux Services Openstack

OpenStack est composé d'une série de projets au code source libre auquel contribue une large communauté. Chaque projet possède une interface pour se connecter avec d'autres composants OpenStack ou avec des utilisateurs externes, mais aussi offre une ***application programming interface (API)*** qui facilite l'intégration de ces services dans la solution fournit par openstack [1].

Les différents services d'openstack interagissent entre eux de différentes façons dans une architecture et une organisation bien définie, il y a des services principaux comme les services d'identité et de l'orchestration qui gèrent l'ensemble des autres services qui ne sont pas moins secondaires, mais cependant s'occupent de taches importantes mais qui n'interviennent pas directement en premier dans l'ordre chronologique des interactions avec l'administrateur ou encore l'utilisateur. Tout cela regroupé dans une interface complète faisant office de panneau de contrôle qu'est Horizon. La figure suivante schématise l'organisation et les interactions de ces services.



Architecture d'Openstack.

Les composants openstack les plus importants sont [1] [4]:

a. Nova compute

Gère le cycle de vie des instances de compute (alias les serveurs virtuels) dans un environnement OpenStack. Les responsabilités incluent la génération dynamique, la planification et la mise hors service de machines virtuelles à la demande, et possède un support de base pour le système de containers. Il s'exécute comme un ensemble de daemons par-dessus un serveur linux existant pour fournir ce service.

b. Stockage

b.1. Swift

Stocke et récupère arbitrairement des objets data non structurés via une API *RESTful* basée sur HTTP (une sorte d'API service web qui utilise REST (Representational State Transfer) REST est le style d'architecture pour les systèmes hypermédias utilisée pour le World Wide Web). Sert à la création d'espace de stockage redondant et évolutif pour le stockage de plusieurs Péta-Bytes de données destinées au stockage à long terme. Le service est hautement tolérant aux pannes avec sa réplication de données et son architecture de type scale-out distribuée. Son implémentation diffère des serveurs de fichiers avec répertoires montables. Dans ce cas, le service écrit les objets et les fichiers vers plusieurs disques, en s'assurant que les données soient répliquées à travers le cluster de serveurs, offrant plusieurs points d'accès pour éviter les SPOF « *Single Point Of Failure*». Le projet Swift est conçu pour le stockage, à long terme, de gros volumes, il est basé sur le stockage en mode objet. Le système de stockage d'objets OpenStack organise les données dans une hiérarchie de la forme suivante:

- **Compte :**
  Représente le niveau supérieur de la hiérarchie, il possède toutes les ressources. Le compte étant un synonyme d'un locataire définit les noms pour les conteneurs qui sont réutilisable si ces conteneurs se trouvent dans deux comptes différents.

- **Conteneur :**
  Il peut y avoir n'importe quel nombre de conteneurs dans un compte. Chaque conteneur définit un nom pour les objets qui peuvent être identiques s'ils sont dans des conteneurs différents. En plus de contenir des objets, le conteneur peut également être utilisé pour contrôler l'accès aux objets en utilisant une liste de contrôle d'accès (ACL).

- **Objet :**
  Stocke le contenu des données, tels que les documents, les images, etc. Il peut également stocker des métadonnées personnalisées.

- **Swift Proxy :**
  Le serveur Proxy est le composant central qui relie l'architecture Swift. Pour chaque demande, il recherche l'emplacement du compte, du conteneur ou de l'objet dans le 'ring' et l'envoie en conséquence. Le serveur proxy est également utilisé pour diffuser des fichiers depuis ou vers l'utilisateur.

b.2. Cinder

Fournit un stockage de blocs persistants aux instances en cours d'exécution. En effet, Il permet à l'utilisateur la création et la suppression d'un volume et l'attache à une VM ou le détache de celle-ci. Ce service était inclus dans Nova à l'origine sous le nom nova-volume. Son architecture de pilote enfichable (pouvant être couplé avec d'autres services) facilite la création et la gestion des périphériques de stockage en blocs.

c. Neutron networking service

Permet le Network-Connectivity-as-a-Service pour d'autres services d'OpenStack, comme Nova Compute d'Openstack. Fournit une API pour que les utilisateurs puissent définir les réseaux virtuels et physiques auxquels les instances de compute se connectent au démarrage. Possède une architecture enfichable compatible pour la plupart des fournisseurs connus de réseaux et de technologies. Il fait office de contrôleur SDN. Il offre des fonctionnalités réseaux avancées (tunneling, QoS, Réseaux virtuels et équilibrage de charge, etc...). Pour ce faire, Neutron utilise différentes plugins tels qu'OpenVSwitch et ML2 « *Modular Layer* 2 ».

d. Dashboard horizon

Fournit un portail de self-service basé web qui sert aux interactions avec les services sous-jacents d'OpenStack, comme le lancement d'une instance, la distribution d'adresses IP ou la configuration des contrôles d'accès. Horizon est l'implémentation canonique du **Dashboard OpenStack** (implémentation originelle).

e. les services partagés

Les services partagés d'openstack sont les services de bases communs à tous les autres services openstack et indispensables à leur fonctionnement. Ils se résument dans les points suivants:

e.1. Keystone Identity service

Fournit un service d'authentification et d'autorisation pour les autres services d'OpenStack. Donne un catalogue de points de terminaison (endpoints) pour tous ces services. Un endpoint est une URL d'authentification qui permet d'accéder à un service openstack. C'est un point de contact entre services ou bien entre utilisateurs et services.

Lors de l'authentification, Keystone fournit un Token (jeton) d'autorisation donnant accès à l'utilisation des ressources et services tels que le stockage ou le calcul.

e.2. Glance image service

Stocke et récupère des images de disques de machines virtuelles. Toutes les instances Compute sont lancées à partir d'images glance.

e.3. Barbican

Barbican est une API REST conçue pour le stockage sécurisé, la gestion de mots de passe, de clés de cryptage et de certificats X.509. Il a été conçu de manière à être utilisable dans tous les environnements, incluant les larges clouds éphémères. Barbican est souvent utilisé comme un système de gestion de clés pour permettre des cas d'utilisation tels que la vérification de signature d'image, et de cryptage de volume.

f. Les services d'orchestration

f.1. Heat

Il gère l'infrastructure et Orchestre de nombreuses applications de cloud composées en utilisant soit le format de template natif d'openstack *HOT* (Heat Orchestration Template) qui est un Template de topologies d'infrastructure pour des applications cloud, sous la forme de fichiers texte qui peuvent être traités comme du code, ou le format CloudFormation d'AWS, à travers une API de Requête compatible avec ce format. Il permet, par exemple, de demander à Nova de démarrer une machine virtuelle supplémentaire en cas de charge importante de façon automatique.

f.2. Mistral

Mistral est le service workflow d'openStack. Il a pour but de fournir un mécanisme pour définir des tâches et des workflows sans écriture de code, mais aussi de les gérer et de les exécuter dans un environnement cloud.

Le **workflow** est un processus d'automatisation des tâches permettant un enchaînement automatisé des différentes opérations et étapes de validation d'une tâche plus ou moins complexe.

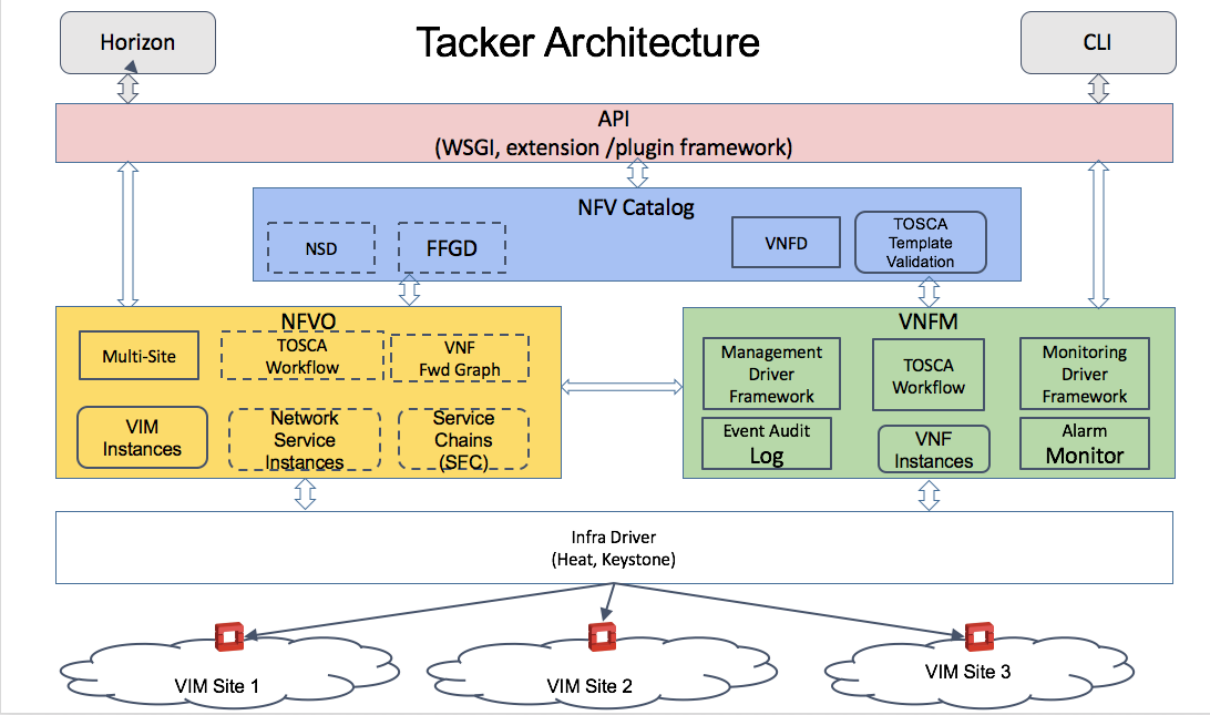
f.3. Ceilometer

C'est un service de contrôle et de mesure du cloud openstack pour la facturation, la scalabilité, le benchmarking et à des fins statistiques. Il permet également le suivi de ressources avec le composant (Gnocchi) et de l'envoi d'alertes avec le composant (Aodh).

2. OpenStackTacker - NFV Orchestration

2.1. Description

Tacker est un projet openstack officiel mettant en place un Manager VNF (VNFM) et un Orchestrateur NFV (NFVO) pour déployer et gérer des services réseaux et des fonctions de réseaux virtuels (VNF) sur une plateforme d'infrastructure NFV telle qu'openstack. Il est basé sur l'architecture du framework MANO de l'ETSI et fournit une pile fonctionnelle pour orchestrer des services réseaux end-to-end utilisant les VNFs [2].



Tackerhigh-level architecture.

2.2. Composants de Tacker

Les principaux composants de tacker sont le VNF Manager, le NFV Orchestrator et le NFV Catalog. Leurs rôles sont décrits et expliqués dans ce qui suit.

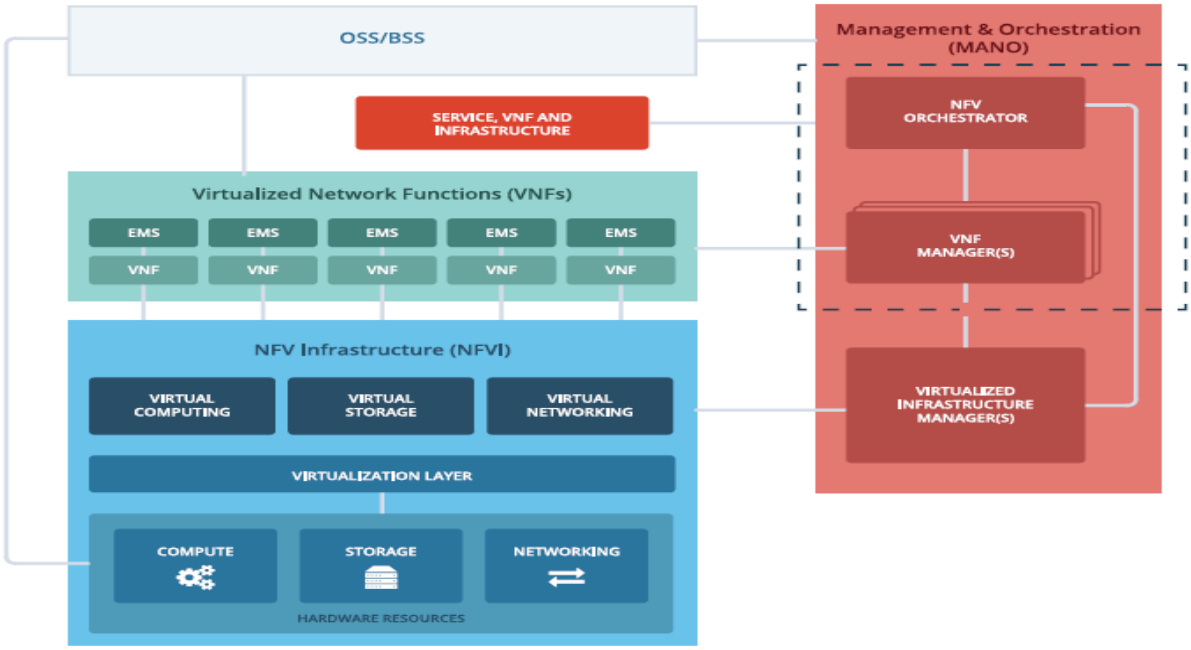
a. VNF Manager

Le VNF manager gère le cycle de vie basique des VNFs (création, mise à jour, suppression) et facilite la configuration initiale de ces dernières. Il possède des caractéristiques telles que Enhanced Platform-aware (EPA) et ainsi lui permet le placement de workloads NFV haute performance (correspondance des VMs aux capacités du hardware). De plus, il permet le contrôle et la surveillance des VNFs déployées, et l'auto reconstruction (healing) / auto scalablité (scaling) des VNFs basées sur une politique définie.

L'opération du scaling consiste en l'augmentation ou la libération de ressources selon des alarmes définies suivant l'exécution en temps réel des VNFs. La capacité de traitement est augmentée ou baissée en créant et approvisionnant automatiquement un VNFC. L'opération de healing redémarre un VNFC (vnf component). Le healing automatique est déclenché quand une alarme est levée contre un VNFC.

b. NFV Orchestrator

Le NFV Orchestrator est responsable du déploiement d'un service réseau end-to-end modélisé en utilisant des VNFs décomposée. Il permet d'édicter la stratégie de placement efficace des VNFs (monitoring et chainage optimale pour un cout réduit). Il offre une grande capacité de connecter le VNFs au moyen d'un SFC (service de chainage de fonctions) - décris dans un VNF Forwarding graph descriptor. Il s'occupe également de la vérification et de l'allocation des ressources du VIM. Il possède aussi la capacité d'orchestration de VNFs au travers de plusieurs VIM et de sites multiples (POP: network point of presence).



VNFM et NFVO dans l'architecture NFV.

c. NFV Catalog

Le catalogue NFV comme son nom l'indique est un annuaire depuis lequel se servent le VNFM et NFVO pour accomplir leurs différents rôles, Il est aussi indispensable que ces deux derniers. Il contient les éléments suivants:

c.1. VNF Descriptors

Le VNFD est un outil qui décrit des informations sur le déploiement et le comportement des VNFs sous un model TOSCA. La création d'une VNFs se fait selon un modelé VFND. Ce dernier exprime le type de la VNF.

c.2. Network Services Decriptors

Le NSD est un template (modelé) TOSCA qui décrit le déploiement et le comportement d'un Network Service (NS) utilisant une collection de VNFs. Il définit les types d'instanciations des NSs (network services).

c.3. VNF Forwarding Graph Descriptors

Le VNFFGD est la template TOSCA principale qui fournit des informations sur le déploiement et le comportement du VNF Forwarding Graphs (VNFFGs), (graphe de transit à travers les VNFs), en d'autres termes le chemin emprunté par le trafic à travers les VNFC (VNF Component). Il définit comment certains trafics correspondant à certains critères décrits dans VNFFGD est conditionné à transiter par une ou plusieurs fonctions réseau dans une topologie réseau connective. Il décrit également les règles du SFC (Service Function Chaining).Là encore, tout comme le NFVD, le VNFFGD définit des types de VNFFG, ce dernier englobe les NFPs.

- **Network ForwardingPaths (NFPs)**

Le NFP est un composant du VNFFG qui décrit une liste ordonnée ou successive de points de connexion appartenant aux VNFs qui constituent une chaine de VNFs. Il contient un Service Function Chain (SFC) et un classifier.

- **Service Function Chaining (SFCs)**
  Le SFC est un composant d'un NFP et forme un chemin que les trafics empruntent. Une chaine peut contenir de multiple VNFs à travers un ou plusieurs hôtes physique ou virtuel. Une VNF peut faire partie de plusieurs chaines.

- **Classifiers**
  Le classifier est un composant d'un NFP et définit des règles pour distinguer quel trafic doit entrer le SFC auquel il est relié.

c.4. TOSCA Template Validation

C'est un service de validation de template TOSCA, ces dernières sont écrites dans le YAML. Il s'agit vérifier que les Template ne comportes pas d'erreurs et sont aptes à être transmise au service Heat pour qu'il puisse les traduire et ainsi les exécuter pour répondre à la requête demandée.

TOSACA (Topology and orchestration standard for cloud application) est un langage standard de description de topologie de services web pour cloud, également de description de leurs composants, relations, et les processus qui les gèrent. Il est écrit en YAML un format expliqué juste dans ce qui va suivre.

Il décrit aussi ce qui est nécessaire de préserver à travers les déploiements de services dans différents environnement pour permettre l'interopérabilité des services cloud et leur gestion

quand les applications sont portées sur des environnements cloud alternatifs. TOSCA peut être utilisé pour la définition de VNFs, surveillance de nœuds et de politiques actives de healing et de scalabilité (scaling), chainage de VNFs ...etc.

YAML, acronyme de **YetAnotherMarkupLanguage** dans sa version 1.0[1], il devient l'acronyme récursif de **YAML Ain'tMarkupLanguage** (*« YAML n'est pas un langage de balisage »*) dans sa version 1.1[2], est un format de représentation de données par sérialisation Unicode. Il reprend des concepts d'autres langages comme XML, ou encore du format de message électronique tel que documenté par RFC2822.

Son objectif est de représenter des informations plus élaborées que le simple CSV (un format basique) en gardant cependant une lisibilité presque comparable, et bien plus grande en tout cas que celle du XML.

Depuis 2015, Symfony 2, Drupal 8 et phpMyAdmin, entre autres, l'utilisent pour leurs formats d'entrée et de sortie.

Conclusion

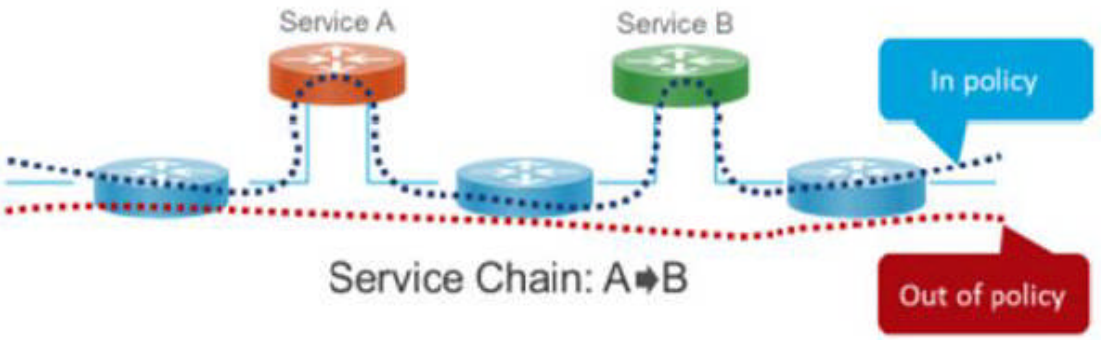
Dans ce chapitre nous nous sommes penchés en profondeur sur le principe de fonctionnement d'openstack, la plateforme cloud qui prendra en charge notre projet, ainsi que l'outil Tacker avec lequel nous allons le développer. Le prochain chapitre mettra l'accent sur le Service Function chaining et les différents composants qui entrent en jeux dans la configuration d'une stratégie de chainage des VNFS.

Chapitre III: Service de Chainage NFV

Introduction

Dans le contexte de NFV, certains services doivent être déployés en « coupure » (par exemple un pare-feu, accélérateur WAN, …), cela signifie que le déploiement se fait au travers de plusieurs phases, par exemple lors d'une mise à jour au lieu de le faire sur tous les serveurs simultanément, cela se fait sur un serveur ou groupe de serveurs d'un même sous réseau à la fois. Cette contrainte peut impliquer une localisation ou stratégie de placement comme vu précédemment, de la fonction réseau virtuelle ou non (typiquement le pare-feu en entrée de site).

Cette contrainte est pénalisante en termes de coût, de temps de déploiement et de maintenance. Si la fonction réseau est virtualisée, il est beaucoup plus facile de l'implémenter là où elle est optimale. Il faut donc que le réseau puisse rediriger le trafic à destination de l'utilisateur vers la fonction réseau en coupure et après le passage dans la fonction réseau renvoyer le trafic vers l'utilisateur et cela dans les deux sens de la communication [7].



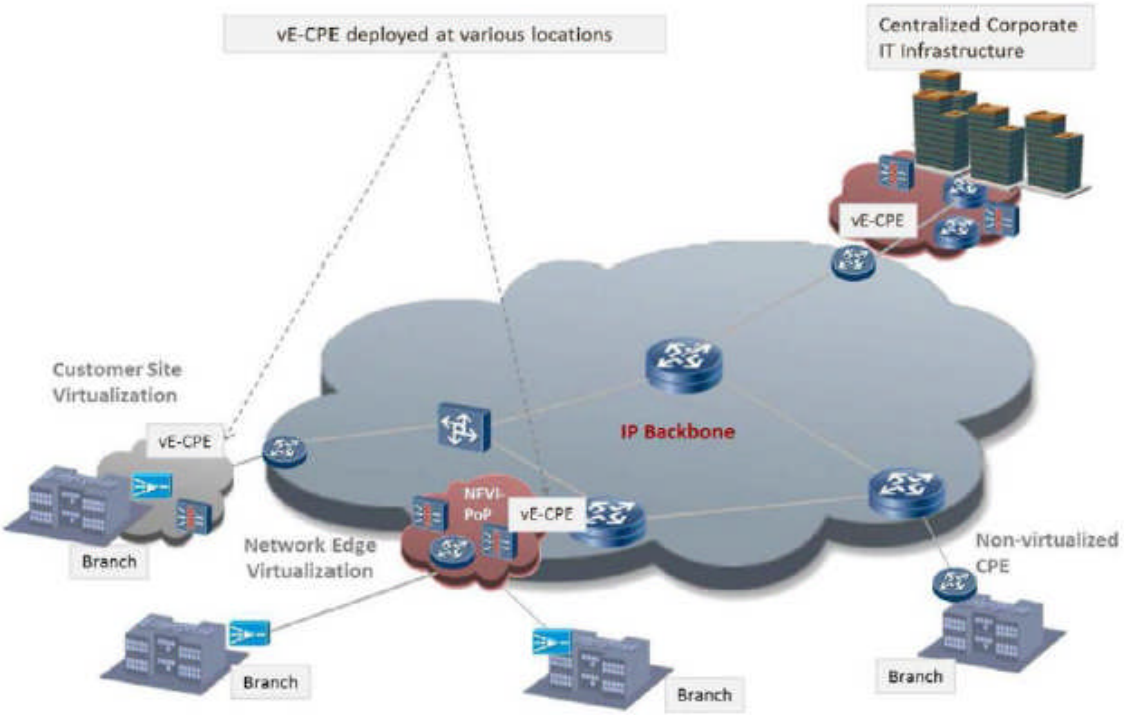
Détournement du trafic par SFC.

De plus, il existe une demande pour pouvoir disposer de plusieurs services réseau successifs (par exemple, un utilisateur peut avoir besoin d'un pare-feu et d'un load-balancer). Il est donc nécessaire de s'affranchir de la localisation de ces services et donc au niveau réseau de pouvoir rediriger le flux réseau à travers ces services. La technologie employée pour permettre d'utiliser plusieurs services successifs « en coupure » est le **« Service Fonction Chaining ».**

1. Implémentation Centralisé ou Décentralisé

Dans le cas du modèle centralisé, les NFV sont exécutées dans un « Datacenter », au cœur du réseau de l'opérateur, cela permet un meilleur taux de consolidations et de mutualiser les investissements de l'opérateur pour ces propres infrastructures. Les problèmes de déploiement sur site de matériel sont aussi évités.

Dans le modèle décentralisé, les VNFs sont exécutées dans les nœuds réseaux de l'opérateur ou dans les locaux de l'utilisateur, cela pour rapprocher l'utilisateur et sa NFV. Certaine NFV n'ont de sens que si elles sont près du site client (par exemple un accélérateur WAN). La politique de sécurité peut obliger à ce qu'un pare-feu soit hébergé dans le site [7].



Type de déploiement.

2. Développement de VNFs pour le système vCPE en utilisant OpenWRT

2.1. OpenWrt

Le projet OpenWrt est un système opératoire Linux pour les appareils embarqués. C'est est une distribution GNU/Linux minimaliste pour matériel embarqué (routeurs, tablettes, téléphones…). Au lieu d'essayer de créer un firmware unique et statique, OpenWrt fournit un file system complètement accessible en écriture (programmable) avec gestion de packages. Cela nous libère des applications et de leurs configuration fournis par les fournisseurs et nous permet de customiser l'appareil a travers des packages qui s'adapte à n'importe quelle application. Pour les développeurs, OpenWrt est le Framework pour construire une application sans avoir à construire un firmware complet autour de celle-ci. Pour les utilisateurs cela signifient la possibilité d'une personnalisation complète, pour l'utilisation du dispositif de manières jamais envisagées auparavant. OpenWrt supporte plusieurs fonctionnalités telles que : Firewall, Router, DHCP, DNS, QoS qui sont déployés en utilisant plusieurs autres logiciels open source s'exécutant sous une interface de configuration unique unifiée (UCI). De plus, son system de personnalisation peut être conçu pour s'exécuter dans des clouds ou des environnements virtuels. Par conséquent, OpenWrt peut être déployé dans le système vCPE comme des VNFs pour fournir une fonction réseau spécifique ou plusieurs pour les clients [5].

2.2 VNFs basées OpenWrt avancées

Malgré qu'OpenWrt possède toutes les fonctionnalités pouvant être utilisés dans un système vCPE, la version courante d'Openstack Tacker ne supporte que des services de bases comme le firewall ou le routing. En conséquence, pour implémenter un système vCPE compréhensif qui inclut plus de fonctions réseau telles que le DHCP, DNS, et QoS, le driver de management OpenWrt VNFM de Tacker et l'image par défaut OpenWrt doivent être mis à jour [5].

2.3. Mise à jour du driver OpenWrt VNFM management de Tacker pour supporter les VNFs DHCP, DNS, and QoS

Tacker utilise un driver de management spécial, VNFM, qui se trouve dans le répertoire suivant “tacker/vnfm/mgmt_drivers/openwrt/openwrt.py", Pour gérer, n'importe quel Virtual Network Function Descriptor (VNFD) qui utilise une image OpenWrt. Ce driver peut injecter des règles de configuration qui sont défini dans le VNFD vers les instances OpenWrt en utilisant le protocole SSH. Essentiellement, ces règles sont du même format que la syntaxe de l'interface de configuration unifiée (UCI) d’OpenWrt qui est le système

centralisé de configuration de ce dernier. Ainsi, nous avons seulement besoin d'ajouter "dnsmasq" qui contrôle les services DHCP and DNS de OpenWrt et "qos" à la liste de services connus de Tacker dans le fichier "openwrt.py" : KNOWN_SERVICES = ('firewall', 'network', '**dnsmasq**', '**qos**') .Après la modification les règles de configuration de l'UCI du DHCP, DNS, et du QoS dans le VNFD Template peuvent être utilisés pour la création de VNFs [5].

2.4. Personnalisation de l'image OpenWrt pour supporter le service QoS

Parce que l'image par défaut OpenWrt ne contient pas le package qui fournit le service QoS, une image personnalisé doit être construite. La procédure de construction et les commandes sont les suivants [1]:

```
$ cd ~

$ mkdir openwrt&& cd openwrt

$ wget https://archive.openwrt.org/chaos_calmer/15.05.1/x86/kvm_guest/openwrt15.05.1-x86-kvm_guest-combined-ext4.img.gz  -O openwrt-x86-kvm_guest-combined-ext4.img.gz

$ gunzip openwrt-x86-kvm_guest-combined-ext4.img.gz

$ mkdir -p imgroot

$ sudo kpartx -av openwrt-x86-kvm_guest-combined-ext4.img

# Replacer loopXp2 avec le résultat de la commande au-dessus, e.g., loop0p2

$ sudo mount -o loop /dev/mapper/loopXp2 imgroot

$ sudo chroot imgroot /bin/ash


# Configurer le mot de passe de cette image en un vide, suivre la commande suivante puis taper entrer deux fois pour le mot de passe

$ passwd


# Configurer le DHCP pour le reseau de OpenWRT pour permettre aux VNFs de pinger

$ uci set network.lan.proto=dhcp; uci commit

$ exit

$ sudo umount imgroot
```

```
$ sudo kpartx -dv openwrt-x86-kvm_guest-combined-ext4.img
```

Puis il faut crée l'image à partir de ce fichier téléchargé et configuré avec la commande suivante:

```
$ openstack image create "openWRT2" \
  --file openwrt-x86-kvm_guest-combined-ext4.img \
  --disk-format raw --container-format bare \
--public
```

Après le succès de la construction, une image personnalisée dans le dossier bin peut être utilisé pour le déploiement d'une VNF QoS sur Tacker.

Les templates VNFD et l'image personnalisée OpenWrt peuvent être trouvées sur le site https://anda.ssu.ac.kr/~openwrt/

3. Routage dans les systèmes NFV

3.1. Description

Le routage doit être repensé et optimisé dans les réseaux d'opérateurs dans un contexte où l'équipement réseau peut être virtualisé par l'utilisation de systèmes NFV. Les techniques classiques qui permettaient un routage robuste vis-à-vis des détériorations du réseau doivent être revues de façon à s'adapter au contexte de virtualisation. De ce fait, l'orchestration de nœuds VNFs, le chaînage de nœuds pour la création de routes deviennent des problèmes difficiles à la fois d'un point de vue théorique et d'un point de vue pratique. En effet, nous cherchons à déterminer dynamiquement des chaînes (ou des chemins) de nœud VNF dans lesquelles chaque nœud offre un service (routage, équilibrage de charge, pare-feu, inspection) alors que les nœuds VNFs peuvent être migrés, répliqués et redimensionnés au cours du temps. Nous considérons également des flux de demandes pour lesquels un ensemble de services doit être appliqué. L'objectif est de prendre des décisions concernant la construction de chemins dans une suite de nœuds VNFs, l'affectation en ligne de flux à des chaînes de VNFs, le partage des ressources et la connexion des clients aux nœuds. Ceci, de manière la plus efficace possible tout en répondant à des objectifs de haute disponibilité et de fiabilité du réseau [8].

3.2. Quelques approches effectuées

Les recherches académiques ainsi que l'industrie focalisent leur attention sur les questions d'optimisation qui permettront de trouver les méthodes et les algorithmes pour résoudre les problèmes de chainage optimal décrits ci-dessus. Les recherches menées sont basées sur les techniques classiques issues du domaine de recherche opérationnelle. Ainsi, des connaissances en théorie des graphes, en construction de flots et en ordonnancement de tâches sont requises. En outre les aspects re-optimisation seront couverts en utilisant des modèles probabilistes tels que l'optimisation stochastique ou la programmation dynamique stochastique. Enfin, le dernier axe d'approche est constitué de techniques d'optimisation en ligne expliqués dans les points suivants :

a. Approches en lignes stochastiques

La présence d'un modèle aléatoire décrivant les demandes futures nécessite de considérer des méthodes en ligne probabilistes et des méthodes d'optimisation stochastique pour construire les politiques en ligne. En effet les méthodes stochastiques nous permettent de prendre en compte les futurs changements potentiels dans le réseau. Cette approche est considérée en utilisant des objectifs robustes et des objectifs en moyennes traités avec un couplage de méthodes déterministes et de programmation dynamique Stochastique ou avec la programmation stochastique.

b. Approches de pire cas en ligne

Dans les approches de pire des cas aucune information sur les demandes futures n'est disponible, et le but est alors de faire face aux situations qui seront les pires possibles pour l'algorithme. Ce genre d'études correspond à la meilleure façon d'éviter les cas les plus néfastes au réseau. L'objectif est l'étude de ces cas par les techniques usuelles de pire cas pour des algorithmes en ligne.

4. Service de Chainage de Fonction (SFC) de Tacker

Cette spécification décris le plan pour l'introduction du service de chainage de fonction (SFC) dans Tacker. Dans son état actuel, Tacker permet la gestion des VNFs; le but étant d'inclure également la gestion du chainage de fonction [2].

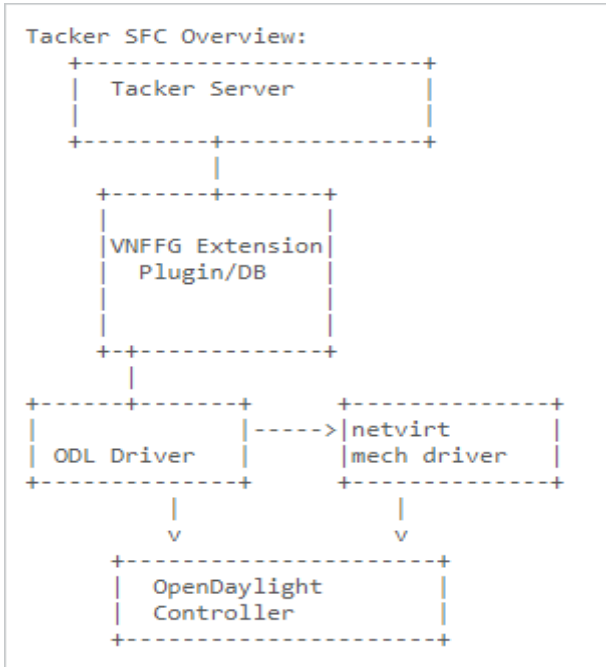
4.1. Description

Il y a un désir profond de la part de la communauté NFV d'être capable d'orchestrer et gérer le SFC. Un utilisateur du NFV voudrait non seulement pouvoir créer des VNFs, mais aussi définir les SFCs de façon à diriger le trafic entre les VNFs. Ceci concorde très bien avec le travail déjà fait dans Tacker afin de gérer et surveiller les VNFs, mais aussi pousse un pas plus loin pour inclure le SFC.

Le but est d'avoir la possibilité de définir une chaine dans l'orchestrateur via une construction logique et abstraite, tout en superposant cette chaine sur la couche réseau. L'étape suivante consiste en le fait d'avoir la possibilité de classifier le trafic des clients (tenants) qui devraient passer à travers ce SFC. La combinaison de VNFs, SFC, et la classification des trafics qui passent à travers ces derniers est décrite comme le Graphe de Transfert des VNFs (VNFFG). Cette spécification traite le changement nécessaire pour orchestrer un VNFFG.

4.2. Solution proposée

La solution que propose Tacker joint à la plateforme Openstack en prenant en compte ce qui a été défini ci-dessus de même que les interactions nécessaires avec les différents modules d'openstack suivent l'architecture suivante



Structure du service de chainage de fonctions (SFC) de Tacker.

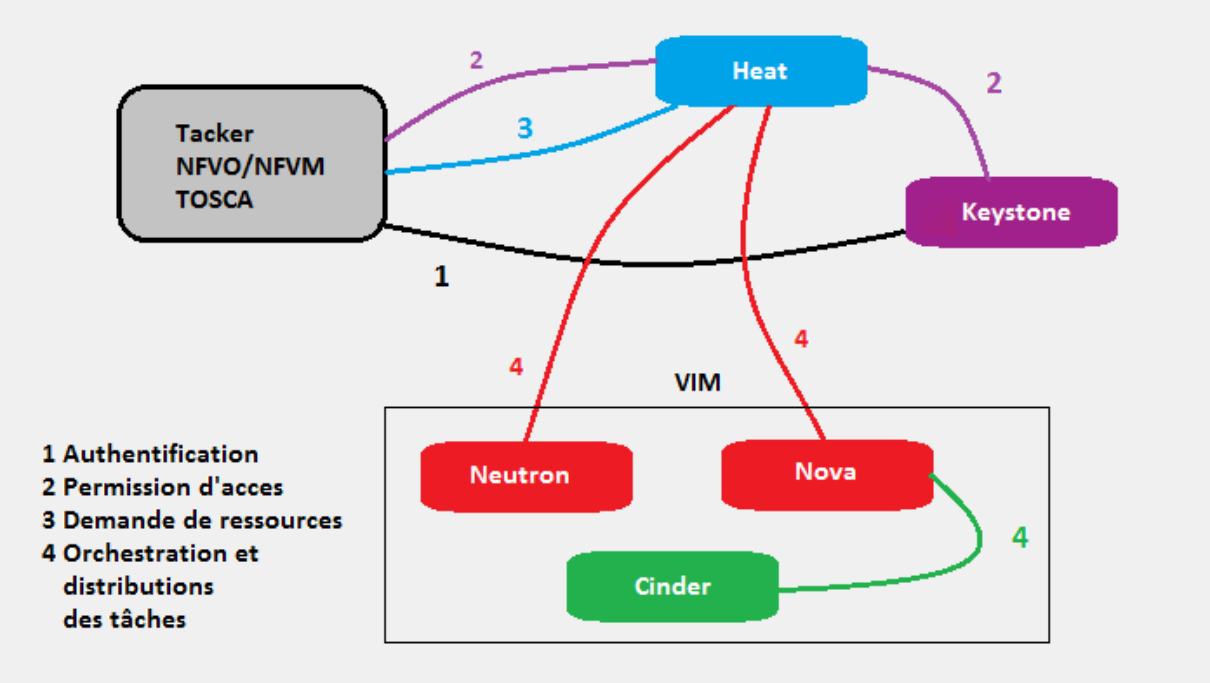
Tacker Server a besoin de l'extension d'un autre plugin,'vnffg' , similaire au 'vnfm'. decrit et expliqué précédemment.

Les pilotes connus pour la création de SFCs sont networking-sfc (un SFC basé sur Neutron) et OpenDaylight SFC. Cette spécification implémente OpenDaylight supporté comme un pilote fonctionnel, et networking-sfc driver (parallèlement avec plusieurs autres) est un effort séparé. Le pilote OpenDaylight est considéré comme expérimentale, et à des buts de testes jusqu'à l'arrivée du pilote Neutron en plus du support networking-odl qui sera ajouté pour pouvoir créer des SFCs pour ODL via Neutron. Quand ce support existera, le pilote OpenDaylight deviendra obsolète en faveur du pilote Neutron.

De façon similaire les pilotes de Classification seront écris. Ces drivers seront manipulés comme "mechanism drivers" qui se chargera de la classification dans le SFC principal du pilote OpenDaylight. Le pilote du mécanisme de classification par défaut sera 'netvirtsfc" qui correspond à un classifier OpenDaylight pour le projet OVSDB(Open vSwitch Database) NetVirt. Après migration vers le pilote Neutron, ce pilote deviendra aussi obsolète.

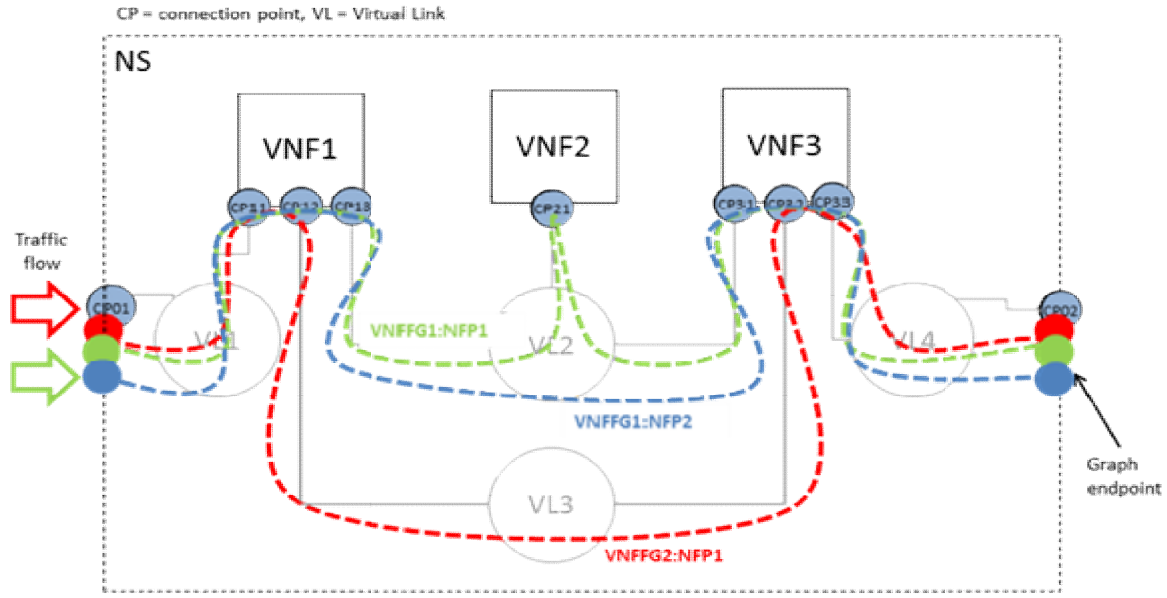
4.3. Interaction Tacker/Openstack

Les interactions entre Tacker et les différents services principaux d'Openstack sont bien structuré. Pour donner un exmple concret, pour la création d'une VNF, l'utilisateur doit d'abord la décrire dans un VNFD. Le VNF Manager (Tacker) va s'occuper de la procédure de création. Cependant, pour qu'il puisse accéder aux ressources du VIM (OpenStack), il doit d'abord s'authentifier chez Keystone. Si l'opération d'authentification est accomplie, un Token et un point d'extrémité depuis Heat seront accordés à Tacker. Ce dernier va exprimer ses besoins en matière de ressources à travers le VNFD. Le Heat traduit à son tour ce VNFD a un langage qu'il comprend et ce faisant appel au service de validation TOSCA pour vérifier que ce VNFD est correcte, et avant d'attribuer les ressources, il vérifie avec le Token les droits de Tacker. S'il a le droit d'accéder à ces ressources, Heat va distribuer les tâches de création sur les différents composants du VIM, par exemple pour le raccordement au réseau le service Neutron qui gère le networking va s'en charger, en ce qui concerne la création de l'instance avec les gabarits ou flavors nécessaires (besoins en ressources RAM et espace disque), cette tâche incombera au service compute Nova, de même pour le stockage un volume disque est alloué a cette instance, une tache d'en se charge le service Cinder [1] [2].



Interaction Tacker/Openstack.

4.4. Caractéristiques d'un SFC



Exemple de SFC.

Tout comme la montre la figure ci-dessus, ceci est un exemple de chainage de VFNs, avec 2 graphes ou VNFFG 1 et 2 respectivement en bleu et vert pour le premier et en rouge pour le second. La particularité ici est que le VNFFG1 comporte deux instances, la première est constituée du NFP1 (Network Function Path) et le seconde est constituée du NFP2. Le NFP1 est formé par la succession de points de connexion cp01, VL1, cp12, cp21, VL2, cp32, VL4, cp02. En effet, un VNFFG doit posséder au moins un Path (chemin), et donc peut en posséder plusieurs à configurer selon le comportement du réseau et de ses ressources.

Une VNF inclut une **VDU/s**, un **connection point/s** et un **virtual link/s**. Par conséquent un VNFD valide doit absolument comporter ces 3 composants. Chaque composant est référé comme un nœud et peut avoir certains types, capacités, propriétés, attributs et exigences. Ces composants décrits par les **node_templates** dans le VNFD template.

a. VDU (Virtual Deployment Unit)

Un VDU (Virtual Deployment Unit) est une partie basique de la VNF. Il s'agit de la VM hébergeant la fonction réseau. C'est ici qu'est spécifié le type de l'image utilisé, dans l'exemple en dessous il s'agit de "openWRT" (ceci n'est que le nom de l'image ce peut être n'importe qu'elle autre nom), ainsi que le flavor qu'utilisera Nova pour créer l'instance, à savoir 1 cpu, 512 MB de RAM et 1 GB d'espace disque. Et puis chose importante le driver utilisé pour lire cette template doit être spécifié, il dépend du type d'image utilisée, ici il s'agit du driver openwrt.

```
VDU1:
  type: tosca.nodes.nfv.VDU.Tacker
  capabilities:
    nfv_compute:
      properties:
        num_cpus: 1
        mem_size: 512 MB
        disk_size: 1 GB
  properties:
    image: OpenWRT
    config: |
      param0: key1
      param1: key2
    mgmt_driver: openwrt
    monitoring_policy:
      name: ping
      parameters:
        count: 3
        interval: 10
      actions:
        failure: respawn
```

Exemple de VDU.

b. Le couple CP-VL (Connection Point - Virtual Link)

Un Connection point est utilisé pour connecter le virtual link interne ou le virtual link externe. Il peut être un NIC virtuel ou un SR-IOV NIC (Single Root I/O Virtualization). Chaque point de connexion doit se rattacher à une VDU. Un CP a toujours besoin d'un virtual link et du rattachement virtuel associé avec ce dernier.

```
CP1:
 type: tosca.nodes.nfv.CP.Tacker
 properties:
   management: true
   order: 0
   anti_spoofing_protection: false
 requirements:
   - virtualLink:
       node: VL1
   - virtualBinding:
       node: VDU1
```

Exemple de CP.

Un Virtual link fournit la connectivité entre les VDUs. Cela représente l'entité logique du VL. C'est qu'est spécifié le réseau auquel sera raccordé la VNF, dans le cas ci-dessous c'est "net_mgmt" plusieurs couple CP-VL peuvent être spécifié dans un fichier, le minimum étant d'un couple CP-VL pour spécifier au moins un réseau.

```
VL1:
 type: tosca.nodes.nfv.VL
 properties:
   network_name: net_mgmt
   vendor: Tacker
```

Exemple de VL.

c. VNFFGD (Virtual Network Funtion Forwarding Graph Descriptor)

En revenant au VNFFG Descriptor, il décrit donc le VNFF Graph, il est constitué de deux parties principales, la partie qui déclare le ou les paths (chemins) du graphe, il s'agit de la chaine de VNFs que doit emprunter le trafic, voici un exemple de VNFFG basique ne comprenant qu'un seul path, ce dernier lui même fait que de deux VNFs:

```
description: Sample VNFFG template

topology_template:
  node_templates:

    Forwarding_path1:
      type: tosca.nodes.nfv.FP.TackerV2
      description: demo chain
      properties:
        id: 51
        path:
          - forwarder: IDS
            capability: CP2
          - forwarder: openwrt
            capability: CP4
```

Path dans le VNFFG.

La seconde partie du VNFFG est appelée "groups", il est presque semblable au Forwarding Path sauf qu'ici il est question de l'ensemble des caractéristique de la chaine, à savoir les VDUs, les CPs ainsi que les VLs y compris les Paths concernant la chaine déclaré précédemment. La figure ci-dessous illustre un exemple basique :

```
groups:
  VNFFG1:
    type: tosca.groups.nfv.VNFFG
    description: Traffic to server
    properties:
      vendor: tacker
      version: 1.0
      number_of_endpoints: 2
      dependent_virtual_link: [VL1,VL2]
      connection_point: [CP2,CP4]
      constituent_vnfs: [IDS,openwrt]
    members: [Forwarding_path1]
```

Description du graphe dans le VNFFGD.

Conclusion

Dans ce chapitre nous avons traité du cœur de notre thématique qu'est le chainage des VNFs. S'agissant d'un nouveau paradigme il n'existe pas encore d'algorithme spécifique mais des approches tout au plus. Nous allons effectuer une approche stochastique durant la partie réalisation, il s'agit comme expliquée précédemment d'une approche aléatoire selon les besoins et les obstacles rencontrés.

Chapitre IV: Implémentation et Réalisation

Introduction

Nous avons choisi la plateforme OpenStack comme support pour la réalisation de notre projet de fin d'études, car cette dernière en plus d'être un logiciel Open source auquel contribue une large communauté dans le monde, il est très en avances sur les autres solutions cloud proposées. Son utilisation est la plus répondu aussi bien chez les entreprises que les particuliers, d'où les raisons qui font que ATM Mobilis l'ai aussi adopté en plus de constituer une véritable alternative aux solutions propriétaires existantes, souvent très coûteuses.

Il est à la fois sécurisé et stable, et comme vu dans le premier chapitre le cloud se trouve être le meilleur moyen pour développer le NFV et lui permettre de faire une percé majeure dans l'industrie IT. Et Openstack s'adapte parfaitement à ce nouveau paradigme encore ambigu qu'est le NFV.

La suite de notre procédure consistera à l'aide de l'outil NFV Tacker, en la création de quelques VNFs (fonction réseau virtualisées) qui rassemblées formeront un unique service, il s'agira d'implémenter une fonction Pare-feu, une fonction DHCP et une fonction DNS. Notre Objectif sera ensuite de proposer une stratégie de chainage de ces VNFs qui consiste à déterminer l'ordre et le chemin spécifique dans lequel le trafic ou les paquets de manière plus précise, par lequel ils transiteront, à travers les VNFs.

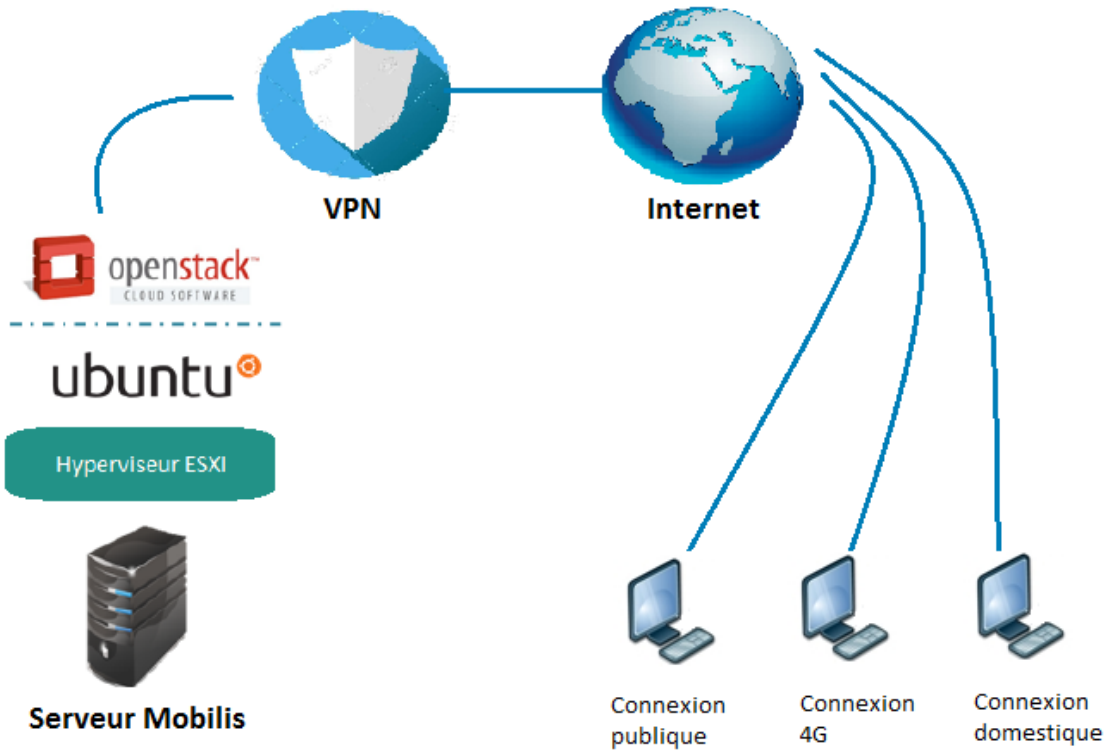
1. Préparation de l'environnement de travail

L'architecture de notre environnement de travail se résume à deux serveurs performants de 128 Go de RAM et de 1 Tera Octets d'espace mémoire chacun, sur lesquels est installé sur chacun un hyperviseur de type 1 **ESXI**, ce qui veut dire comme vu précédemment, ne possède pas de système opératoire hôte, et donc installé directement sur le hardware (bare metal). Sur un des serveurs 4 VMs ont été créées et sur l'autre 2 autre VMs ont été créées, chacune (les 6 VMs) à partir d'une image de l'OS ubuntu server 16.04. Pour chaque VM lui a été alloué 16 Go de RAM ainsi que 100 Go de mémoire. Spécialement ubuntu car openstack a d'abord été développé originellement sur Linux, car leur politique open source s'accordent, d'où même jusqu'à présent les distributions Linux restent les plus optimales pour Openstack malgré qu'il s'installe parfaitement sur Windows et Mac.

La raison pour laquelle les VMs ont été séparées n'est autre que la gestion optimisée des capacités des deux serveurs, et ne concerne en rien le projet en lui-même. Par ailleurs, Nous nous focaliserons sur une seule VM, les autres ayant servis beaucoup plus aux tests de différentes installations et configuration pour l'apprentissage et la familiarisation avec les différents outils et concepts.

Et donc après avoir installé ubuntu serveur 16.04. et toutes les reconfigurations nécessaires notamment concernant l'environnement d'exécution et de développement Python , Openstack a été installé à son tour, la version ocata a été favorisé du a sa stabilité mais aussi au fait qu'elle ne soit ni trop ancienne potentiellement problématique avec les nouveaux drivers et périphériques, ni trop récente et donc potentiellement instable avec des bugs pas encore résolus.

Cette infrastructure nous été fourni par Mobilis, et pour des raisons de flexibilité ainsi que l'opportunité de pouvoir y travailler à tout moment, Mobilis nous a fourni un accès distant à cette infrastructure sécurisé à travers un VPN (Virtual Private Network), accessible depuis n'importe quel point d'accès internet.



Infrastructure d'accès à l'environnement de travail.

Le logiciel utilisé pour la connexion au VPN est FortiClient un outil largement utilisé, FileZila est utilisé pour le transfert de fichiers vers les VMs. Pour finir MobaXterm est utilisé pour avoir un accès sécurisé au VMs via SSH. Tous ces outils doivent être installés sur la machine souhaitée pour accéder à l'environnement de travail.

La figure ci-dessus illustre l'environnement en ne prenant en compte que les parties les plus importantes, et met en scène la possibilité d'accès à distance offerte.

2. Installation d'Openstack

L'installation Openstack a été la plus difficile, car nous avons dû régler plusieurs bugs et erreurs qui ont pris beaucoup de temps à résoudre de par les recherches entreprises.

Il existe deux méthodes d'installation, une première module par module manuellement qui est longue et fastidieuse néanmoins nécessaire une première fois pour la maitrise du concept, mais pas du tout adapté à l'industrie. La deuxième est automatique via un script d'installation, une manière plus rapide, flexible et parfaitement adapté aux exigences industrielles.

C'est donc cette deuxième méthode sur laquelle notre choix s'est tourné pour l'implémentation finale ayant déjà expérimenté la première méthode préalablement. Voici les étapes principales suivies :

2.1. Création d'un compte utilisateur ubuntu

La création d'un nouvel utilisateur est nécessaire du fait que l'exécution de l'installation d'openstack plus précisément de son fichier exécutable stack.sh ne doit pas se faire sur root pour des raisons d'intégrité du système. La commande suivante permet création de l'utilisateur **stack**.

```
root@stage2:~# useradd -s /bin/bash -d /opt/stack -m stack
```

Commande de création d'utilisateur ubuntu.

Puis ce nouvel utilisateur doit recevoir des privilèges d'administrateurs pour donner carte blanche en ce qui concerne le téléchargement, l'accès, la configuration et l'installation des fichiers nécessaires. Pour ce faire nous allons ajouter la ligne suivante dans le fichier /etc/sudoers.d/stack.

```
 GNU nano 2.5.3                    File: /etc/sudoers.d/stack

stack ALL= (ALL) NOPASSWD: ALL
```

Octroi de privilèges root à l'utilisateur stack.

2.2. Configuration de pré-installation

Apres s'être assuré que Python est parfaitement à jour et l'installation de l'outil complémentaire nécessaire Pip qui est un gestionnaire de paquets utilisé pour la gestion et l'installation de paquets écrits en Python, nous passons à l'étape qui consiste en le clonage du repository devstack, c'est à dire le dossier par lequel se fera l'installation.

```
stack@stage2:~$ git clone https://git.openstack.org/openstack-dev/devstack
Cloning into 'devstack'...
remote: Counting objects: 42238, done.
remote: Compressing objects: 100% (21106/21106), done.
remote: Total 42238 (delta 29949), reused 32248 (delta 20468)
Receiving objects: 100% (42238/42238), 8.81 MiB | 3.88 MiB/s, done.
Resolving deltas: 100% (29949/29949), done.
Checking connectivity... done.
stack@stage2:~$
```

Clonage du repository devstack.

Une fois le clonage terminé, un dossier 'devstack' est sensé avoir été créé. Nous accédons a celui-ci pour modifier quelques fichiers, d'abord le fichier 'functions-common' ou il faut modifier la ligne concernant le timeout de la connexion en l'augmentant de la valeur 300 à 1000, ce qui nous permettra ainsi que l'installation ne s'arrête pas brutalement à cause d'une connexion lente mais patientera temps de rétablissement d'une liaison effective.

```
GNU nano 2.5.3                    File: functions-common
  local proxies="http_proxy=${http_proxy:-} https_proxy=${https_proxy:-} no_proxy=$
  local update_cmd="$sudo $proxies apt-get update"
  if ! timeout 300 sh -c "while ! $update_cmd; do sleep 30; done"; then
      die $LINENO "Failed to update apt repos, we're dead now"
  fi
```

Modifications du fichier functions-common.

Puis vient le tour du fichier 'stackrc' ou cette fois-ci nous devons changer le protocole réseau utilisé pour le téléchargement des fichiers d'installation. Le protocole par défaut est **git**, nous le changeons en **https**, car le premier étant quelques fois problématique.

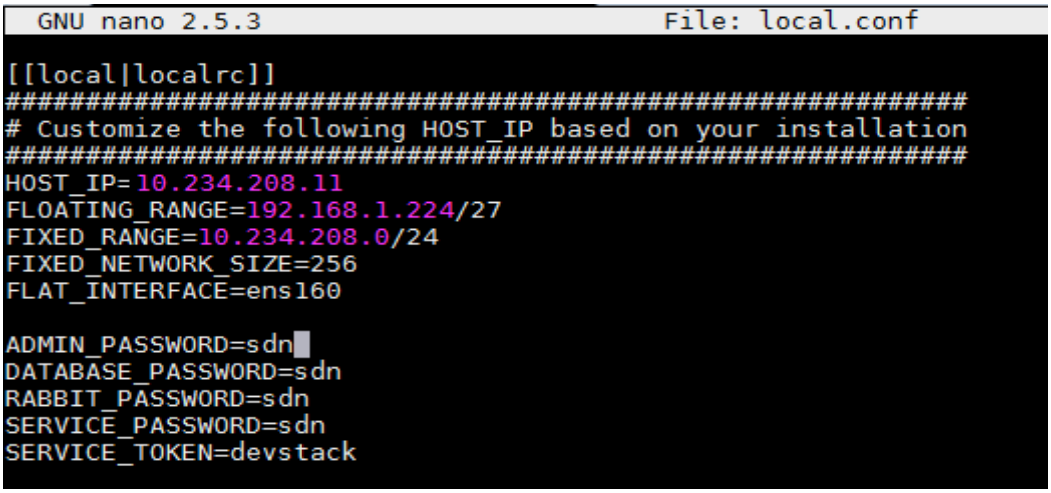
```
GNU nano 2.5.3                    File: stackrc

# Base GIT Repo URL
# Another option is https://git.openstack.org
GIT_BASE=${GIT_BASE:-git://git.openstack.org}
```

Modification du fichier stackrc.

Pour que cette action soit pleinement efficace, le Protocole spécifié dans le script d'installation **local.conf** doit également être https sous peine de rencontrer des problèmes de connexion.

Vient la partie de la configuration du script d'installation devant obligatoirement porter le nom de **local.conf**, ce dernier peut être téléchargé depuis le site officiel Openstack. Cependant certaines configurations dépendent de notre propre environnement doivent être faites par nous-mêmes.

Le script comporte deux sections, une première concernant les configurations selon notre système, une seconde traitant de la personnalisation des modules à installer.



Section 1.

Pour la première section , il s'agit de l'adresse IP de notre hôte qui est dans notre cas celle de notre VM 10.234.208.13, l'option Fixed network qui représente l'adresse réseau ainsi du masque de sous réseau dans notre cas 10.234.208.0/24, l'option Flat interface qui représente la carte réseau utilisée , dans notre cas notre VM possède deux carte réseau, cependant l'une lui a été configuré pour un accès internet, l'autre pour des raisons de gestion interne n'a pas d'accès internet, et donc il faut veiller à remplir l'option avec la carte ayant accès à internet ayant la même adresse IP avec celle de l'hôte sinon cette dernière sera obsolète.

Puis vient la spécification du mot de passe de l'administrateur et des mots de passes pour les services de base de données qui servira à créer les espaces de stockage propre à chaque module openstack, du service d'authentification ou d'identité Memcached qui utilise un mécanisme de jetons, du service Rabbit qui est un système de queue de message qu'utilise openstack pour la coordination des opérations entre les différents services, autant dire le backbone ou bien la colonne vertébrale d'openstack.

Pour la seconde section, nous devons ajouter le module Tacker ainsi que l'activation des drivers nécessaires pour ce dernier, notamment en ce qui concerne le plugin de chainage networking-SFC indispensable pour la suite de notre travail.

```
# Tacker
enable_plugin tacker https://git.openstack.org/openstack/tacker master
```

Ajout du module Tacker dans la section 2.

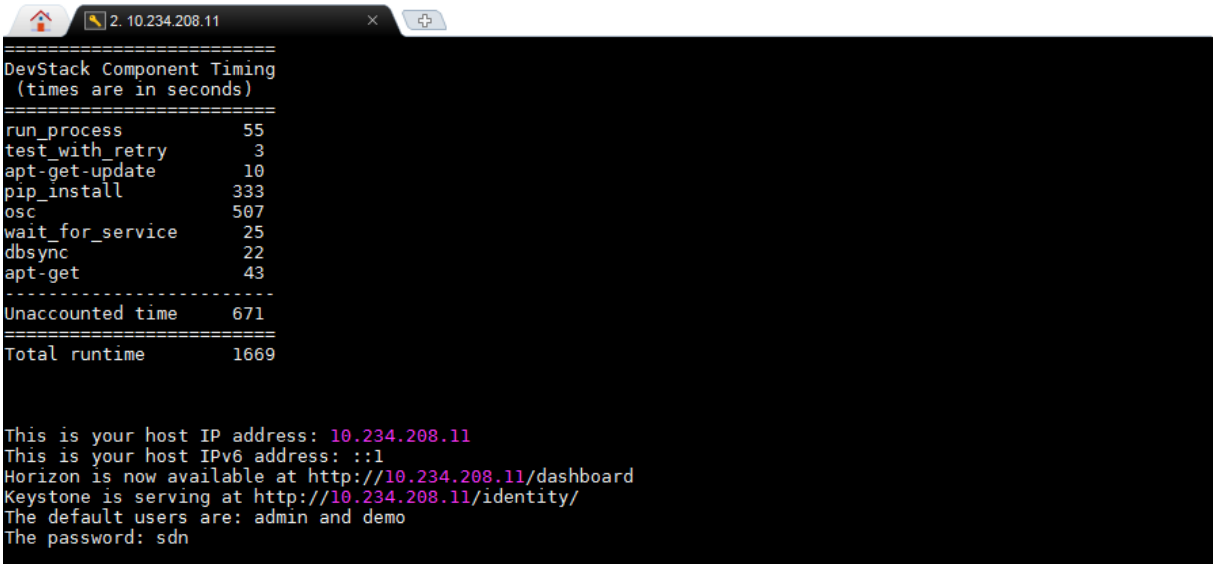
```
# Enable heat, networking-sfc, barbican and mistral
enable_plugin heat https://git.openstack.org/openstack/heat master
enable_plugin networking-sfc https://git.openstack.org/openstack/networking-sfc master
enable_plugin barbican https://git.openstack.org/openstack/barbican master
enable_plugin mistral https://git.openstack.org/openstack/mistral master
```

Driver de chainage networking-sfc.

Maintenant nous pouvons lancer l'exécution en tapant la commande ./stack.sh.

3. Interface Openstack

Si l'installation s'est correctement déroulée, la figure ci-dessous illustre ce qui devrait s'afficher comme résultat.



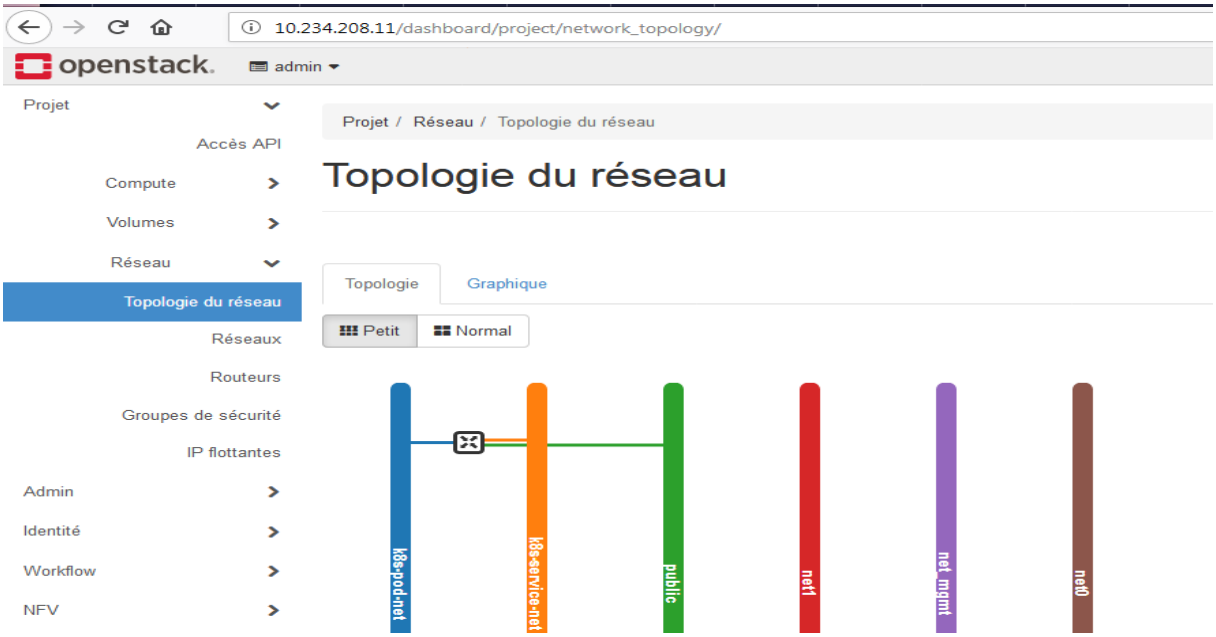
Fin de l'installation d'openstack.

Il y est indiqué quelques informations notamment le temps total qu'a pris l'exécution, et plus important encore l'adresse IP de notre interface de contrôle qui est logiquement l'adresse spécifié lors de la pré-configuration vu au-dessus, ainsi que les utilisateurs par défaut et le mot de passe ici 'sdn' spécifié précédemment.

En tapant l'adresse IP du Dashboard, une interface d'authentification s'affiche.

Interface d'authentification.

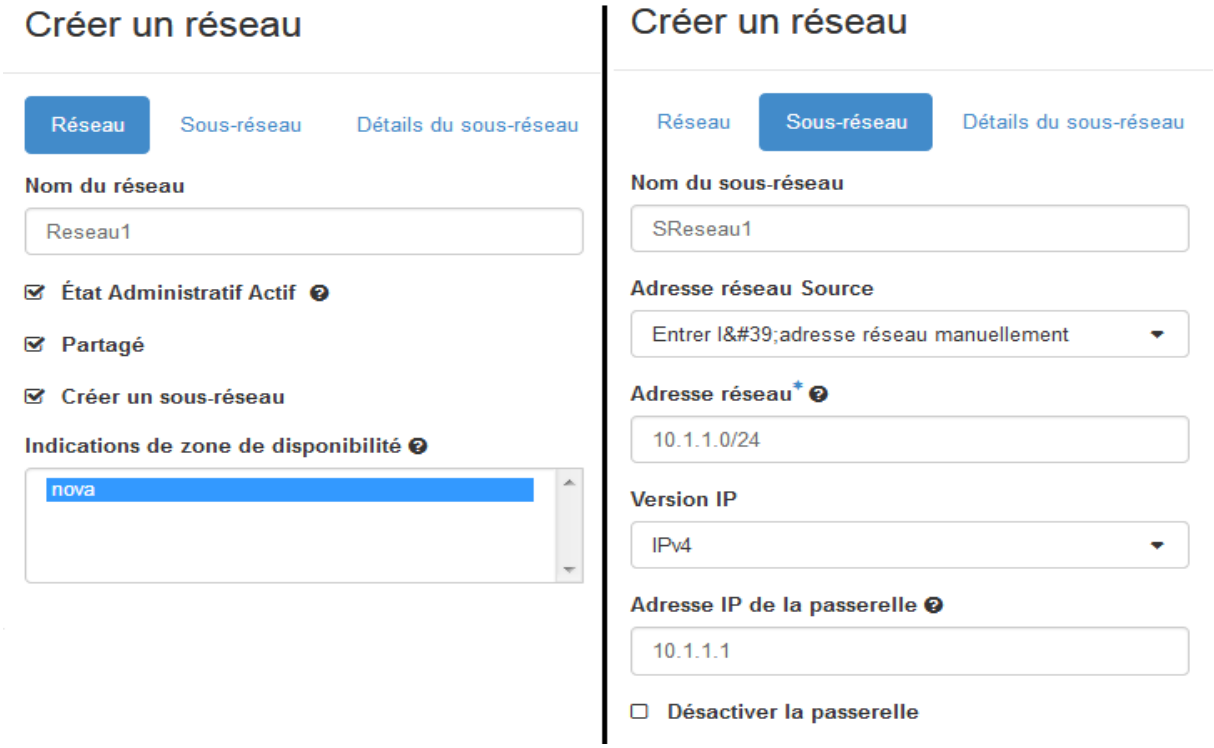
Une fois authentifié soit avec l'utilisateur par défaut admin , ou bien demo, nous accédons au Dashboard de contrôle d'Openstack, avec des onglets comportant tous les services précédemment spécifié dans le script d'installation.



Dashboard d'Openstack.

3.1. Configuration réseau

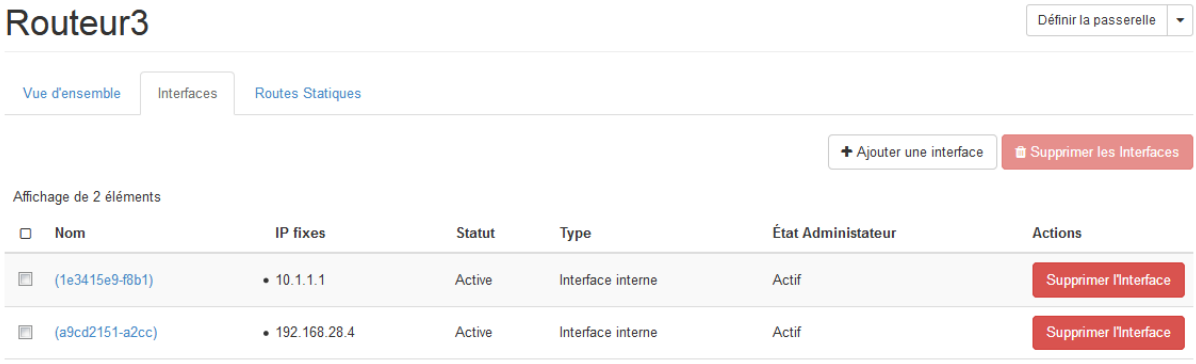
Nous avons créé un réseau privé "Reseau1" sur lequel seront déployées les VNFs que nous créerons plus tard. Les attributs de ce réseau sont décrits dans les figures suivantes :



Création de Reseau1.

Configuration DHCP ET DNS de "Reseau1".

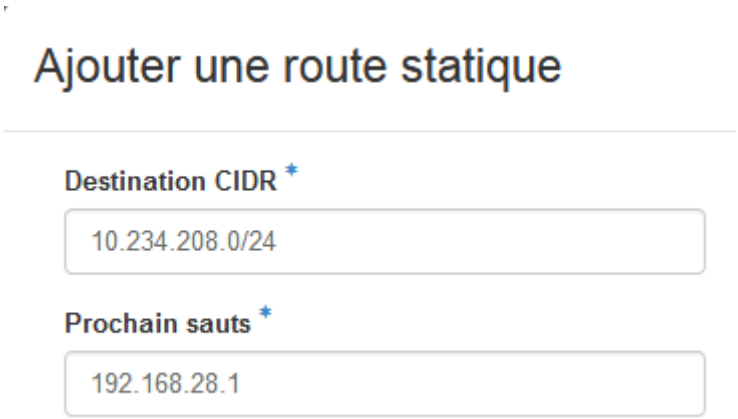
Puis ce réseau a été raccordé au réseau "public" possédant l'adresse 192.168.28.0/24 qui est directement relié à l'hôte avec l'adresse IP 10.234.208.11 MSR 255.255.255.0, cela ayant été fait automatiquement lors de l'installation de la partie Tacker dans Openstack. Ce raccordement a été fait via un routeur nommé "Routeur3", configuré de la manière suivante:



Routeur3.

Comme la figure ci-dessus le montre, le jonction des deux réseaux se fait par le biais de deux interfaces que nous avons configurées dans le routeur, 10.1.1.1 qui est la passerelle par défaut du réseau "Reseau1" comme l'illustre la figure "Création de Reseau1". La seconde interface est 192.168.28.4 qui est une adresse IP qu'on a alloué différente de la passerelle par défaut.

Nous ajoutons une route statique dans le routeur qui permettra a Reseau1 de pouvoir contacter le réseau de la machine hôte à savoir 10.234.208.0/24. en spécifiant le prochain saut qui est la passerelle par défaut du réseau "public" 192.168.28.1 .



Route statique Routeur3.

De même que pour "Reseau1", il faut configurer la connectivité depuis le réseau de la machine hôte qui est notre VM pour atteindre Reseau1 via le réseau "public". Et comme le "public" est déjà accessible à l'hôte par défaut, il suffit de spécifier le réseau de destination et la passerelle par laquelle accéder en spécifiant l'interface du côté "public". Cela se fait en tapant la commande suivante:

```
root@stage1:~# route add -net 10.1.1.0/24 gw 192.168.28.4
root@stage1:~# ping 10.1.1.1
PING 10.1.1.1 (10.1.1.1) 56(84) bytes of data.
64 bytes from 10.1.1.1: icmp_seq=1 ttl=64 time=0.492 ms
64 bytes from 10.1.1.1: icmp_seq=2 ttl=64 time=0.099 ms
64 bytes from 10.1.1.1: icmp_seq=3 ttl=64 time=0.084 ms
64 bytes from 10.1.1.1: icmp_seq=4 ttl=64 time=0.072 ms
```

Reseau1 atteint.

4. Interface Tacker

4.1. Création du VIM

Avant de passer à la création des VNFs, Un VIM (Virtual Infrastructure Manager) doit être crée et enregistré, comme expliqué dans le chapitre précédent, le VIM est une partie de l'architecture NFV, cette fonction est rempli par Openstack, Tacker est la partie MANO de cette même architecture. C'est option qui permet à Tacker de supporter la gestion multi site car chaque VIM représentant une infrastructure donnée.

Nous avons créé fichier contenant les attributs nécessaire, qui servira de template TOSCA pour la création du VIM portant le nom de "vimpfe".

```
 GNU nano 2.5.3                          File: vim-config.yaml

auth_url: 'http://10.234.208.11/identity/v3'
username: 'admin'
password: 'sdn'
project_name: 'admin'
project_domain_name: 'Default'
user_domain_name: 'Default'
cert_verify: 'True'
```

VIM template tosca.

```
stack@stage1:~$ sudo nano vim-config.yaml
stack@stage1:~$ openstack vim register --config-file vim-config.yaml \
>         --description 'pfe' --is-default vimpfe
+----------------+-----------------------------------------------------------+
| Field          | Value                                                     |
+----------------+-----------------------------------------------------------+
| auth_cred      | {                                                         |
|                |     "username": "admin",                                  |
|                |     "project_id": null,                                   |
|                |     "project_name": "admin",                              |
|                |     "cert_verify": "True",                                |
|                |     "user_domain_name": "Default",                        |
|                |     "key_type": "barbican_key",                           |
|                |     "secret_uuid": "***",                                 |
|                |     "auth_url": "http://10.234.208.11/identity/v3",       |
|                |     "password": "***",                                    |
|                |     "project_domain_name": "Default"                      |
|                | }                                                         |
| auth_url       | http://10.234.208.11/identity/v3                          |
| created_at     | 2018-09-14 17:55:13.951084                                |
| description    | pfe                                                       |
```

Commande de création du VIM.

4.2 Création des VNFs

Apres avoir mis en place le réseau sur lequel seront mis en place les VNFs, ainsi l'établissement des tables de routage nécessaires, la création du VIM, vient à présent la création des VNFs.

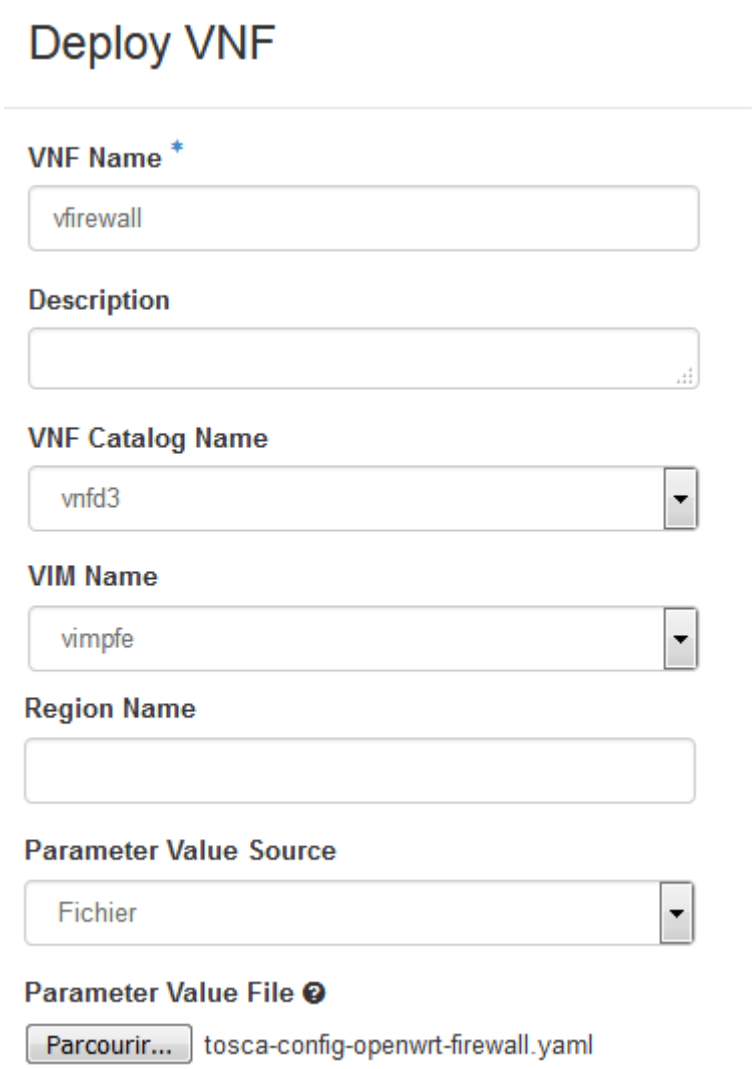
La création de ces dernière se fait en deux partie, tout d'abord un VNFD (VNF Descriptor) doit être crée, c'est une template qui comme vu précédemment contient les informations sur le VDU (Virtual Deployement Unit), c'est là ou est spécifié l'image, ici il s'agit du type openWRT, les besoins en espace disque et mémoire. Ainsi que les couples de CP (Connecting Point) et VL (Virtual Link) c'est dans cette partie que sont spécifiés le ou les réseaux auxquels la VNF sera rattachée, dans notre cas "Reseau1". La création du VNFD se fait de la manière suivante :

```
stack@stage1:~/tacker/samples/tosca-templates/vnfd$ openstack vnf descriptor create --vnfd-file tosca-vnfd-openwr
t.yaml vnfd3
```

VNFD du firewall.

```
stack@stage1:~$ openstack vnf descriptor create --vnfd-file tosca-vnfd-openwrtd.yaml vnfd4
```

VNFD du DHCP.



Déploiement du VNF firewall.

La seconde partie de la création des VNFs, consiste en leur déploiement. Ceci se fait en se basant sur le type de VNFD qui décrit les besoins de la fonction ainsi qu'un second fichier qui représente le comportement de cette fonction. C'est dans cette partie aussi que doit être spécifié le VIM. La VNF du DHCP est déployée de la même manière.

La commande **openstack vnf list** permet de vérifier le déploiement avec succès de nos VNFs, et qu'elles sont bien raccordé au réseau "Reseau1" selon la plage d'adresse allouée.

```
stack@stage1:~$ openstack vnf list
+--------------------------------------+-----------+----------------------+--------+--------------------------------------+--------------------------------------+
| ID                                   | Name      | Mgmt Url             | Status | VIM ID                               | VNFD ID                              |
+--------------------------------------+-----------+----------------------+--------+--------------------------------------+--------------------------------------+
| 2cde2ed2-a691-46db-af6e-59cdf34bfd7d | vdhcp     | {"VDU1": "10.1.1.56"} | ACTIVE | ca062884-9a14-4387-8f6c-f3517cf1d1c2 | 2417f4e7-f464-44f9-a05b-97280f86cb94 |
| 388029cc-dc70-460d-89a1-37b3af2f6363 | vfirewall | {"VDU1": "10.1.1.57"} | ACTIVE | ca062884-9a14-4387-8f6c-f3517cf1d1c2 | 01b4ac0c-9616-4bd7-922f-debcab7b3a3a |
+--------------------------------------+-----------+----------------------+--------+--------------------------------------+--------------------------------------+
stack@stage1:~$
```

Liste des VNFs.

Nous avons testé la connectivité vers les fonctions avec succès.

```
stack@stage1:~$ ping 10.1.1.57
PING 10.1.1.57 (10.1.1.57) 56(84) bytes of data.
64 bytes from 10.1.1.57: icmp_seq=1 ttl=63 time=2.20 ms
64 bytes from 10.1.1.57: icmp_seq=2 ttl=63 time=0.759 ms
64 bytes from 10.1.1.57: icmp_seq=3 ttl=63 time=0.835 ms
64 bytes from 10.1.1.57: icmp_seq=4 ttl=63 time=0.813 ms
^C
--- 10.1.1.57 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3001ms
rtt min/avg/max/mdev = 0.759/1.151/2.200/0.607 ms
stack@stage1:~$ ping 10.1.1.56
PING 10.1.1.56 (10.1.1.56) 56(84) bytes of data.
64 bytes from 10.1.1.56: icmp_seq=1 ttl=63 time=2.11 ms
64 bytes from 10.1.1.56: icmp_seq=2 ttl=63 time=0.956 ms
64 bytes from 10.1.1.56: icmp_seq=3 ttl=63 time=1.04 ms
64 bytes from 10.1.1.56: icmp_seq=4 ttl=63 time=0.810 ms
^C
--- 10.1.1.56 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 0.810/1.230/2.113/0.517 ms
stack@stage1:~$
```

VNF ping.

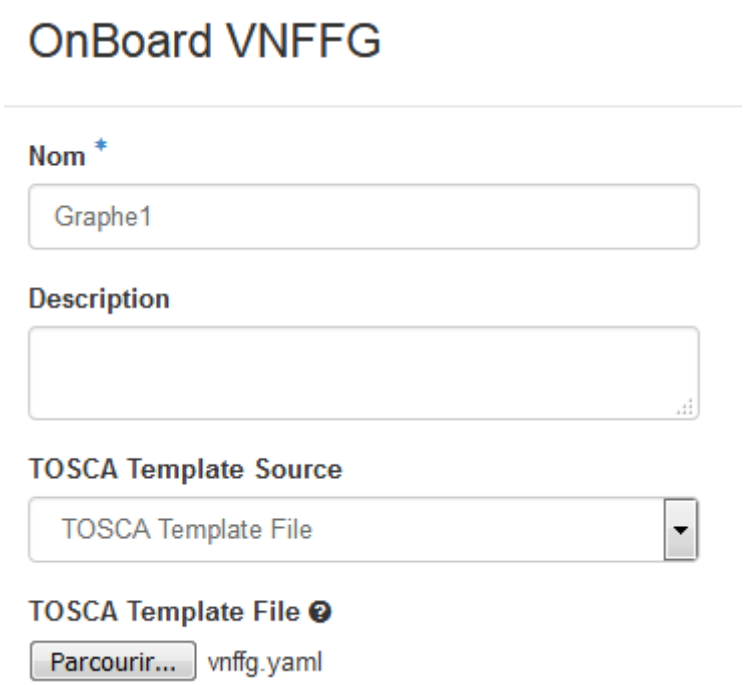
Ainsi que l'accessibilité via le SSH.

```
stack@stage1:~$ ssh root@10.1.1.57
The authenticity of host '10.1.1.57 (10.1.1.57)' can't be established.
RSA key fingerprint is SHA256:1P5WrppFTIWwvLWia8J/p3wlBomf0DSg24xuN8sfox0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.1.1.57' (RSA) to the list of known hosts.
root@10.1.1.57's password:


BusyBox v1.23.2 (2016-01-02 14:31:17 CET) built-in shell (ash)

  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M
 -----------------------------------------------------
 CHAOS CALMER (15.05.1, r48532)
 -----------------------------------------------------
  * 1 1/2 oz Gin            Shake with a glassful
  * 1/4 oz Triple Sec       of broken ice and pour
  * 3/4 oz Lime Juice       unstrained into a goblet.
  * 1 1/2 oz Orange Juice
  * 1 tsp. Grenadine Syrup
 -----------------------------------------------------
root@OpenWrt:~#
```

La fonction vfirewall via SSH.

4.3. Chainage des VNFs

Le chainage de VNFs se fait par le biais d'un VNF Forwarding Graph. Un VNFFG est créé d'une manière similaire à la création d'une VNF. Tout d'abord un VNFFG Descriptor est créé, c'est une template tosca décrivant le graphe emprunter par le trafic, le chaine de fonction y est décrit sous forme d'un Path (chemin), ainsi que les points de connexion qui relient la chaine. Nous avons créé un VNFFGD du nom de Graphe1 comme suit :



Création du VNFFG Descriptor.

Le contenu de notre template VNFF Desriptor contient les éléments suivants :

```
topology_template:
  node_templates:

    Forwarding_path1:
      type: tosca.nodes.nfv.FP.TackerV2
      description: demo chain
      properties:
        id: 51
        policy:
          type: ACL
          criteria:
            - name: block_icmp
              classifier:
                network_src_port_id: 654eaf96-3737-4b72-9dd7-445a26dfc2ee
                ip_proto: 1
        path:
          - forwarder: vnfd3
            capability: CP1
          - forwarder: vnfd4
            capability: CP1
```

Section Path et criteria du VNFFGD.

Ici les le vnfd3 et vnfd4 forme le Path ou chemin qu'empruntera le trafic relié tout deux par les connection points CP1 et CP2, quant à la section criteria, il s'agit tout simplement de la politique appliquée sur le trafic qui entrera dans la chaine, dans notre cas présent cas du blocage des paquets ICMP.

La création du VNFFG se fait de deux manières comme pour toute chose dans openstack, soit via l'interface graphique soit en ligne de commande. La figure ci-dessous illustre la méthode en ligne de commande :

```
stack@stage1:~$ openstack vnf graph create --vnffgd-name Graphe1 --vnf-mapping vnfd3:vfirewall,vnfd4:vdhcp --symm
etrical vnffg1
```

Création de la chaine vnffg1.

La chaine vnffg1 est créé à partir de son descripteur Graphe1, puis un mapping entre les VNFDs déclarés au sein de ce descripteur, et les fonctions de la chaine est fait, dans notre cas notre chaine est constituée des deux VNFs vfirewall et vdhcp. Le paramètre "symmetrical" indique que la chaine est appliquée au trafic dans les deux sens.

Conclusion

C'est ainsi que se conclut la partie implémentation et réalisation de notre projet ou nous avons créé une topologie réseau sur laquelle nous avons déployé deux fonctions virtuelles un firewall ainsi qu'une fonction DHCP, puis nous avons suggéré un chainage de ces deux fonctions avec une politique basique bloquant le protocole ICMP. Il ne s'agit là que d'un exemple basique des applications de routage et de sécurité du trafic encore plus poussées peuvent être configuré grâce à l'outil VNF Forwarding Graph que fournit le service Tacker.

Toute entreprise spécialisée dans la télécommunication a pour but de non seulement garder sa place dans de le marché mais de conquérir de nouveaux, et cela ne peut se faire sans que cette dernière ne suive constamment la tendance des nouvelles technologies. Et avec l'avènement du NFV, si ATM Mobilis réussi une percée dans ce domaine cela lui permettrait indéniablement de sortir des frontières Algériennes et de s'implanter dans des pays développés à travers ses partenaires étrangers. De plus, tout pays qui se veut être une puissance à terme, doit impérativement posséder des infrastructures de télécommunication à la pointe aussi bien dans le civile que militaire.

Durant ce projet de fin d'étude, nous avons pu nous pencher sur des technologies nouvelles faisant encore partie du domaine de la recherche mais qui commence déjà à être implémentée dans l'industrie de l'IT tant les exigences de cette dernière sont élevées.

En outre, le Could Computing qui a redéfinie irrémédiablement le mode de fonctionnement des entreprises en les libérant des contraintes d'une infrastructure sur site, en leur fournissant des services à la demande. Le brillant concept du NFV, qui bien que nouveau et pas encore prêt, possède un potentiel extrêmement prometteur sans tomber dans l'exagération, en aspirant à séparer le software du hardware, pour accélérer le déploiement ou redéploiement de services tout en facilitant considérablement l'évolutivité et la flexibilité de ces opérations complexes.

Ce stage nous aura permis d'avoir été confronté à des contraintes d'ordre technique et de temps sévères et ardus, de même que le manque de documentation et dans le cas des œuvres existantes, une ambigüité due à la nouveauté du concept. Mais cela nous a permis d'en ressortir avec une meilleure maturité en ce qui concerne l'analyse, la compréhension et la recherche de solution aux problèmes techniques, tant sur les aspects réseaux, administration, logiciel, configuration, installation et implémentation auxquels nous avons dû faire face en travaillant sur Ubuntu, Openstack et Tacker dans une mise en conditions de la vie professionnelle en entreprise. Mais aussi d'acquérir une certaine maitrise dans ces différents concepts et technologies que sont le Cloud Computing et le NFV, bien que ces derniers demeurent vastes et ne cessent d'évoluer de jour en jour selon les besoins de l'air compétitrice moderne.

Pour conclure, nous espérant que ce modeste travail puisse servir dans le futur aux étudiants, professeurs et professionnels.

[1] Documentation officielle Openstack: **https://docs.openstack.org/rocky/**

[2] Documentation officielle Tacker: **https://docs.openstack.org/tacker/latest/**

[3] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, NielsBouten, Filip De Turck, Raouf Boutaba, Network Function Virtualization: State-of-the-art and Research Challenges, IEEE COMMUNICATIONS SURVEYS & TUTORIALS 2015.

[4] TAHAR DJEBBAR Abdellah Seddik, ZENOUNE Safa, Mise en place d'une solution Cloud de type « *Security as a Service »* et « *Storage as a Service»* sur *OpenStack* au sein d'ATM Mobilis, INSTITUT NATIONAL DE LA POSTE ET DES TECHNOLOGIES DE L'INFORMATION ET DE LACOMMUNICATION 2017.

[5] Trinh Nguyen, Tri-Hai Nguyen, Myungsik Yoo, Developing the VNFs for the vCPE system using OpenWrt, Department of ICMC Convergence Technology, Soongsil University Seoul South Korea 2018.

[6] OpenStack Foundation Report, Accelerating NFV Delivery with OpenStack 2016.

[7] Antoine DESPLAN, Alice THOREL, Le chainage de fonctions réseau, une opportunité pour les réseaux de l'Enseignement/Recherche 2017.

[8] EmmauelHyon, Christoph Dürr, "Algorithmes pour le routage en ligne dans des systèmes NFV "Laboratoire d'informatique de l'Université Pierre et Marie Curie, Paris.

[9] https://searchcloudcomputing.techtarget.com

Résumé

La technologie fait des progrès de jour en jour, particulièrement dans le domaine de l'IT. Ainsi pour qu'une entreprise soit compétitrice, doit veiller en permanence à être à jour avec les nouveaux concepts tels que le Cloud Computing ou le NFV, et anticiper en se projetant vers l'avenir au risque avéré d'être laisser a la traine et disparaitre. Notre travail s'inscrit dans cette continuité en proposant un model basique de chainage de fonctions virtuelles par lequel un trafic peut transiter de telle sorte qu'à partir de cela, pouvoir mettre en place différents services performants dans un futur proche.

ملخص

إن التكنولوجيا في تقدم مستمر ، خاصة مجال الإعلام الآلي. فلمؤسسة كي تنافس على هذا المستوى يستوجب عليها متابعة التقنيات الحديثة باستمرار مثل الحوسبة السحابية و افتراضية وظائف شبكة الاتصالات و التّوقع بالتّخطيط للمستقبل و إلاّ مواجهة خطر التأخر و الزّوال.

يعد عملنا جزءًا من هذه الاستمرارية من خلال اقتراح نموذج أساسي لربط الوظائف الافتراضية الذي يمكن من خلاله للازدحام الشّبكي المرور، بحيث يمكن بناء على ذلك إنشاء خدمات متنوعة بجودة عالية في المستقبل القريب.

Summarize

Technology makes progress every day, particularly in the IT field. Thus for a corporation to compete, it has to be constantly updated with new concepts as Cloud Computing or NFV, and anticipate by projecting itself into the future or face a definite risk of being left behind or even disappear. This work goes within this framework by suggesting a basic virtualized function chaining model in which a network traffic can pass through, and building from that, being able to deploy different services with great performance in the close term.