Virtualisation réseau et interconnexion de machines virtuelles en utilisant **Open vSwitch** Module Neutron d'Openstack

Joël Berthelin

Plan

- Constats
- Brigde-utils : commutateurs/ponts virtuels.
- LAN virtuels : VLANs et VXLANs.
- Optimisation du chemin des données : Data Plane Développement Kit (DPDK)
- OpenVSwitch (OVS) : commutateur de réseau multicouche virtuel
  - OVS : architecture / déploiement / commandes
  - Types d'interconnexion d'un commutateur OVS
  - OVS avec DPDK , Openflow, Faucet
  - Native OVS firewall driver
- 3 Exemples de déploiements : Linux, Gns3, Openstack.
- Module Neutron d'Openstack
- Open Virtual Network (OVN) : une mise en réseau virtuelle.
- Architectures OVN ~ Architecture SDN

Constat 1 : Forte progression de l'usage de la virtualisation.

- Un serveur physique héberge plusieurs machines virtuelles (VM) isolées entre elles grâce a un hyperviseur mais qui peuvent échanger des données grâce au réseau.
- Changement de paradigme : tous les services ne sont plus hébergés sur le même serveur (BDs, services WEB, stockages des zones, les authentifications) mais distribuer sur des machines virtuelles.
- Avantages :
  - optimiser la gestion des installations, des sauvegardes et des plan de reprise d'activité.
  - sécuriser les données : compartimenter les tâches pour les protéger des unes des autres.
  - accroitre la disponibilité en augmentant les instances ou les ressources et en mettant en place de fonction de bascule.
  - faciliter la mise en place d'environnement de développement, de test, de changement de version ou la migration sur des clusters externes
  - accroitre la montée en charge sur plusieurs machines
  - « baisser » le nombre de serveurs physiques
- Inconvénient :
  - forte augmentation de l'usage du réseau entre les machine virtuelles

Constat 2 : Le cluster est partout.

- La mise en cluster (ou en grappe ou ferme) de serveurs est aujourd'hui courante. Elle consiste à regrouper plusieurs ordinateurs indépendants appelés nœuds (nodes), afin de permettre une gestion globale et de dépasser les limitations d'un ordinateur pour :
  - augmenter la disponibilité (mise en place de fonctions de bascule ('fail-over') et de rétablissement ('recovery'));
  - faciliter la montée en charge ;
  - permettre une répartition de la charge ;
  - faciliter la gestion des ressources (processeur, mémoire vive, disques durs, bande passante réseau).
- Cette association de nœuds peut prendre des formes différentes, ils peuvent être tous actifs ou bien certains peuvent être en attente et démarrer en cas de panne ou de forte charge ; ils peuvent partager des ressources comme du stockage (souvent regroupées sur des NAS : zones OS, zones users, zones de données, …) , des accès réseau (routages , proxy, authentification, …) , sous le contrôle de l'un d'entre eux ou de différentes manières.

Constat 3 : Forte progression de la commutation et des cartes réseaux au sein des baies.

- Utilisation de commutateurs avec des ports 10Gb/s, 25 Gb/s, 40 Gb/s et 100 Gb/s
  - Exemple: Gamme DELL EMC Networking
- Utilisation de connectique en fibre optique dans les armoires.
- Un serveur est livré avec 4 à 8 cartes réseaux 1Gb/s et 10Gb/s.
- On utilise l'agrégation de port pour augmenter les débits (LACP)
- Exemple : un armoire 42U => 1 commutateur 48 ports 10Gb/s + 18 serveurs (connectés par 2 ports) + 1 NAS (connecté par 8 ports) + 4 ports d'interconnection hors armoire.

Bridge-utils : Création de ponts virtuels simples

- Linux : Packages bridge-utils et iproute2
- Gestion possible du protocole Spanning Tree Protocol (STP) pour gérer les liaisons multiples et rédondantes et des priorités/coûts associés
- Creation de bridge : $ brctl addbr br0
- Gestion du STP : $ brctl showstp br0

  bridge id 8000.00004c9f0bd2

  designated root 0000.000480295a00

  root port 1 path cost 104 max age 20.00 bridge

  max age 200.00 hello time 2.00 bridge hello time 20.00 forward delay 150.00 bridge forward delay 15.00 ageing time 300.00 gc interval 0.00 hello timer 0.00 tcn timer 0.00 topology change timer 0.00 gc timer 0.33 flags

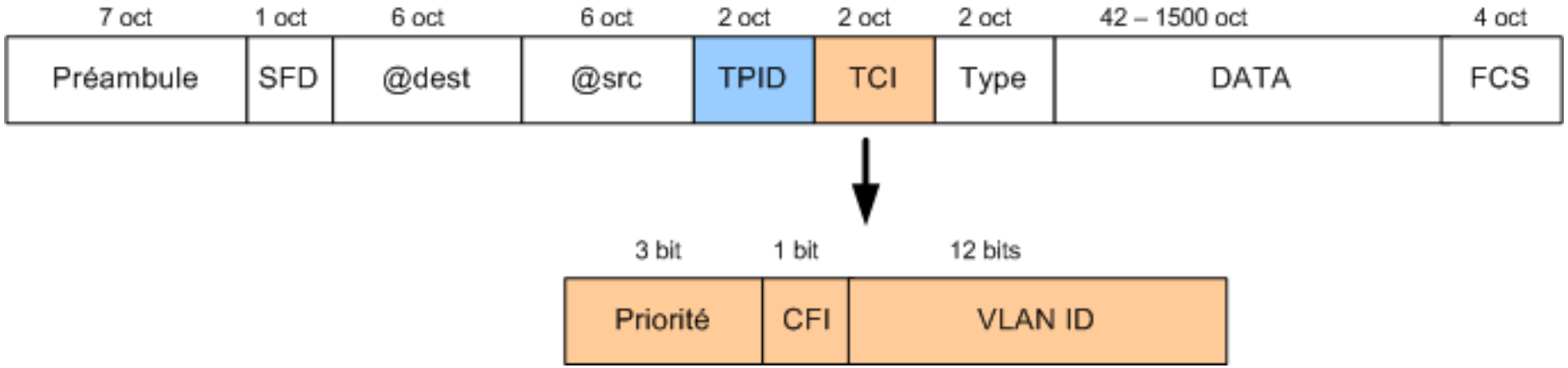
VLAN : Principes

- Niveau 2 du modèle OSI.
- Segmentation d'un support physique en segments logiques.
- Avantages:
  - Limite la diffusion des broadcasts.
  - Plus grande flexibilité de la segmentation du réseau. (indépendance géographique)
  - Amélioration de la sécurité.
  - Priorisation des flux.

VLAN : Types

- Par port
- Par adresse MAC
- Par adresse IP
- Par Protocole de niveau 3

VLAN : Trame Ethernet 802.1Q



- TPID: type du tag, 0x8100 pour 802.1Q
- Priorité: 8 niveaux de priorité définis par l'IEEE 802.1P
- CFI: Ethernet ou token-ring
- VID: VLAN identifier, jusqu'à 4096 vlans

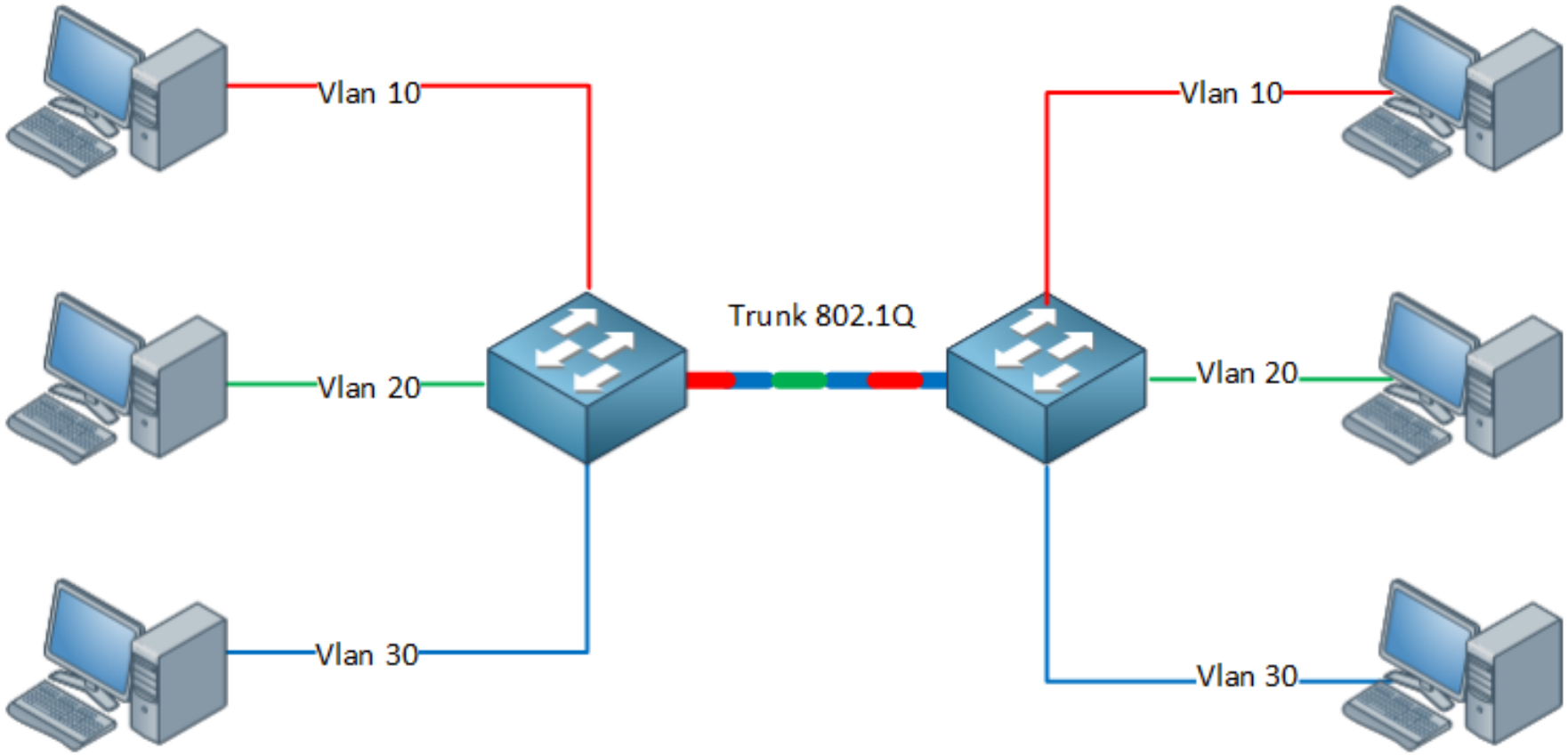
VLAN : Terminologie

- VID : Vlan ID
- PVID : Port Vlan ID
- Mode d’accès
  - Port Access ou untagged(non étiqueté)
  - Port trunk ou tagged(étiquetté)
  - Matériel Aware (Vlan-informé)
- VLANs de base
  - VLAN par défaut généralement VLAN 1
  - VLAN utilisateur
  - VLAN de management
  - VLAN natif

VXLAN : Interconnexion inter-VLAN

- Se fait au niveau 3 du modèle OSI,
- Donc routage, entre interfaces virtuelles
- Seul un routeur peut réaliser l’interconnexion entre deux VLANs et modifier ainsi le VID associé à une trame.

VLAN : Exemple déploiement 802.1Q



VXLAN : Virtual eXtensible Local Area Network

- Le protocole VxLAN permet d'encapsuler des trames Ethernet de couche 2 OSI dans des datagrammes UDP de couche 4. Le numéro de port UDP de destination par défaut attribué par l'IANA[1] pour le VXLAN est le 4789
- Les hébergeurs exploitent le protocole VxLAN, chaque VM sur une machine physique est identifiée par un identifiant réseau VxLAN sur 24 bits, nommé VNI. L'hyperviseur de chaque serveur gère l'encapsulation et la décapsulation des trames contenues dans les VxLANs (VTEP : VxLAN Tunnel EndPoint). (Overhead estimé/Ethernet = 6,17%)

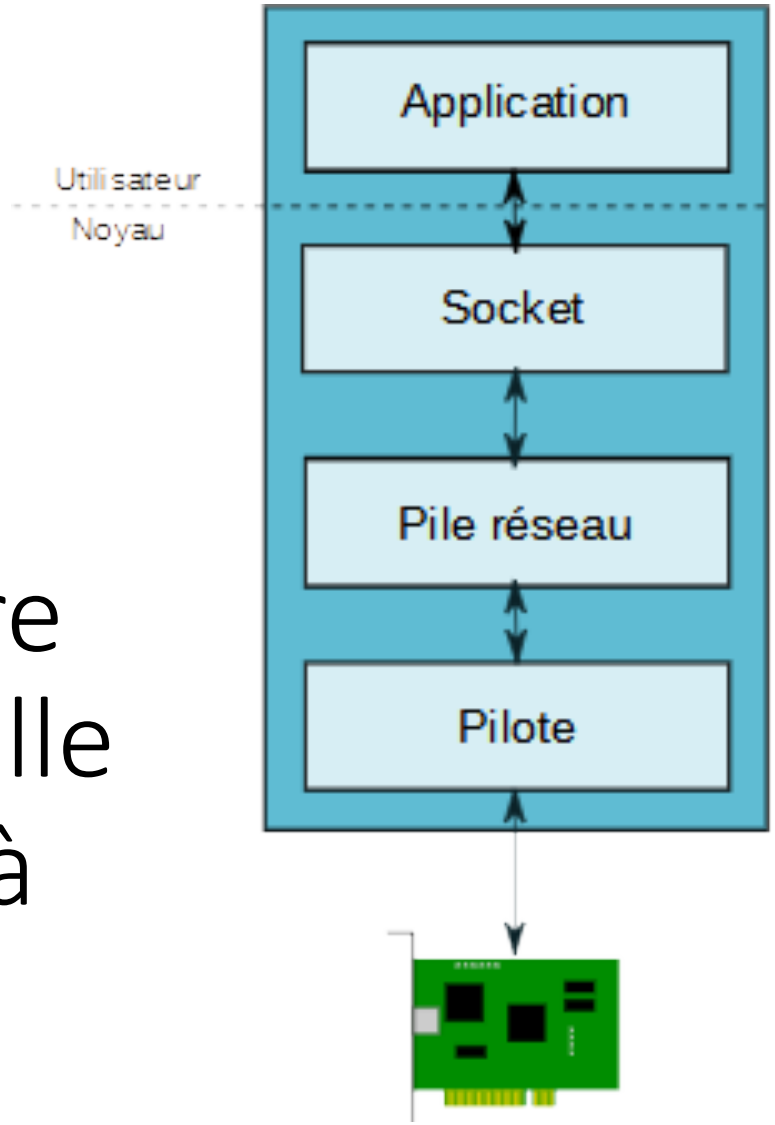
Trame Ethernet utilisant 802.1q standard

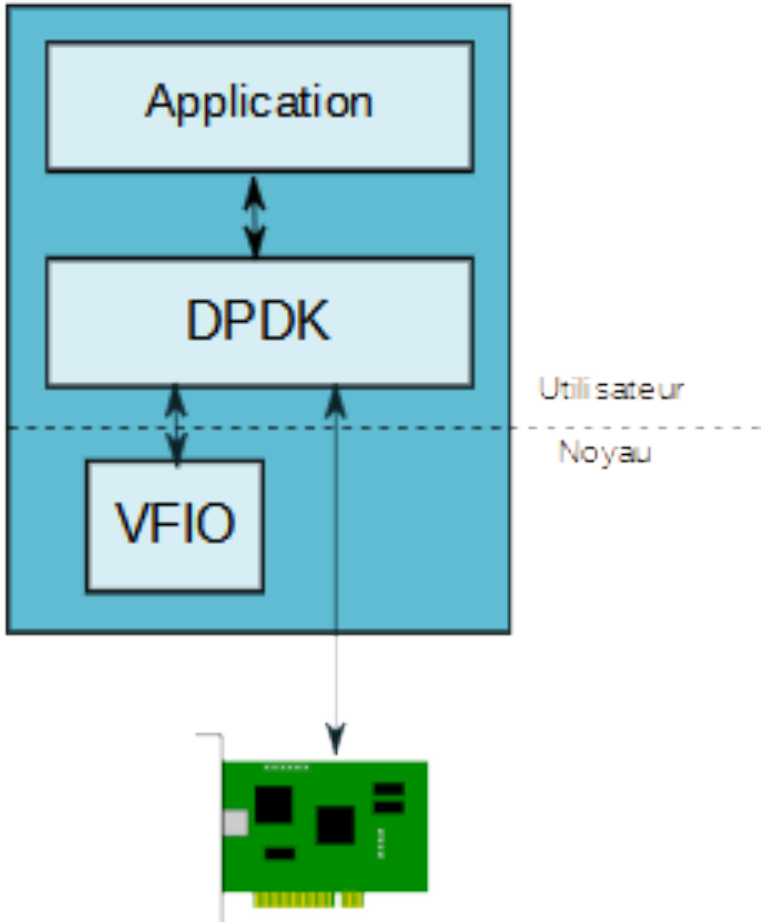
Trame Ethernet utilisant VXLAN standard

Optimisation du chemin des données : Data Plane Développement Kit (DPDK) -1

- DPDK est un projet qui a démarré chez Intel en 2010, en coopération avec d'autres constructeurs tels que la société 6Wind, le projet est devenu un logiciel libre sous licence BSD en 2013. Depuis début 2017, la gouvernance du projet a été transmise à la **Linux Foundation**.
- Le principe de DPDK est de s'abstraire au maximum des services du noyau pour prendre en charge directement en mode utilisateur les différents mécanismes nécessaires le plus efficacement possible. En particulier DPDK s'appuie sur un ensemble de Polled Mode Drivers (PMD), pilotes en mode utilisateur, sans interruption (c'est-à-dire avec attente active) qui supportent les cartes réseau les plus courantes dans le monde des serveurs. Ces pilotes sont rendus possibles par les interfaces UIO ou VFIO du noyau qui rendent accessible les registres de commande d'un périphérique depuis une bibliothèque utilisateur.

Architecture traditionnelle comparée à DPDK





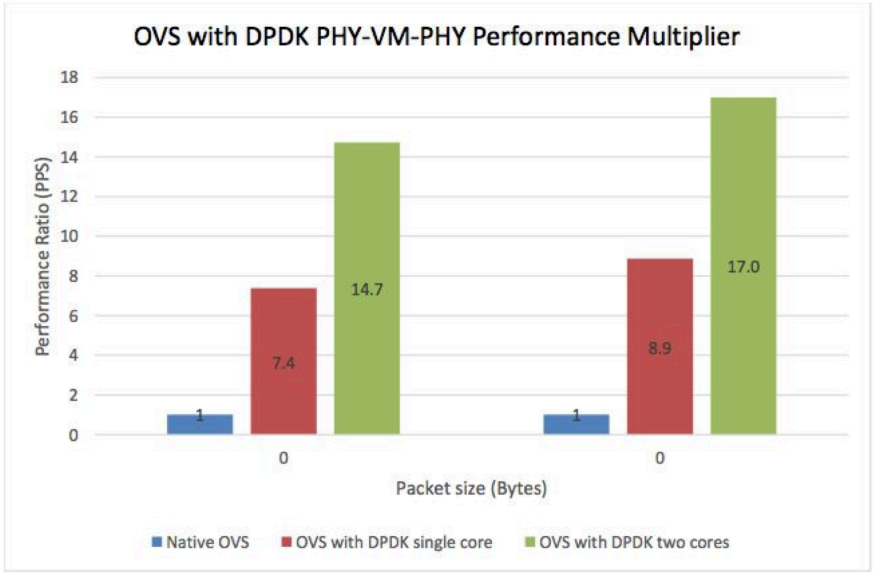
Architecture d’une application DPDK

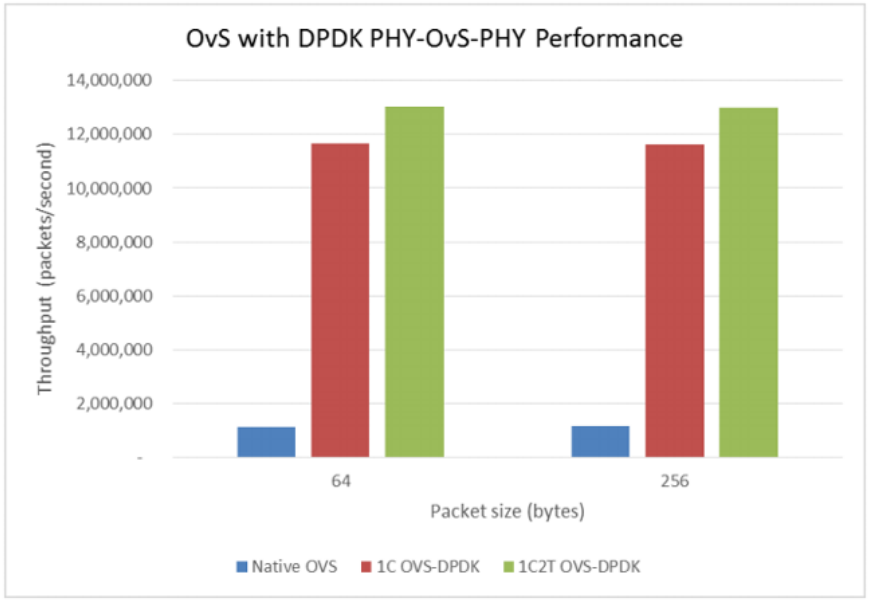
- Une application DPDK est construite autour de trois composants principaux :
- EAL Environment Abstraction Layer la couche de portabilité entre les différents systèmes (supporte FreeBSD et Linux)
- PMD Polled Mode Drivers les pilotes de carte réseau en mode polling. DPDK supporte un certain nombre de cartes réseau 1 Gb/s, 10 Gb/s et 40 Gb/s de plusieurs constructeurs (Intel, Mellanox...), ainsi que des interfaces virtuelles pour des architectures de type Xen, KVM ou VMWare.
- Bibliothèques de support : l’environnement DPDK fourni des bibliothèques hautement optimisées pour réaliser des opérations de base dans une application réseau : classification des paquets, gestion de files d’attente, gestion de la mémoire

OVS avec DPDK (OVS-DPDK)

- Assure une commutation optimisée dans le « user space » grâce à des CPU dédiés, et vient se substituer aux opérations réseau d'ordinaire réalisées dans le kernel.
- Permet le transfert direct des paquets entre l'espace utilisateur et l’interface physique, en contournant la pile du réseau du noyau.
- Offre une amélioration significative des performances par rapport à la transmission du noyau, grâce à l'élimination de la gestion des interruptions et de la traversée de la pile du réseau du noyau.
- Nécessite de dédier des CPUs à DPDK.

Benchmarcks OVS avec DPDK

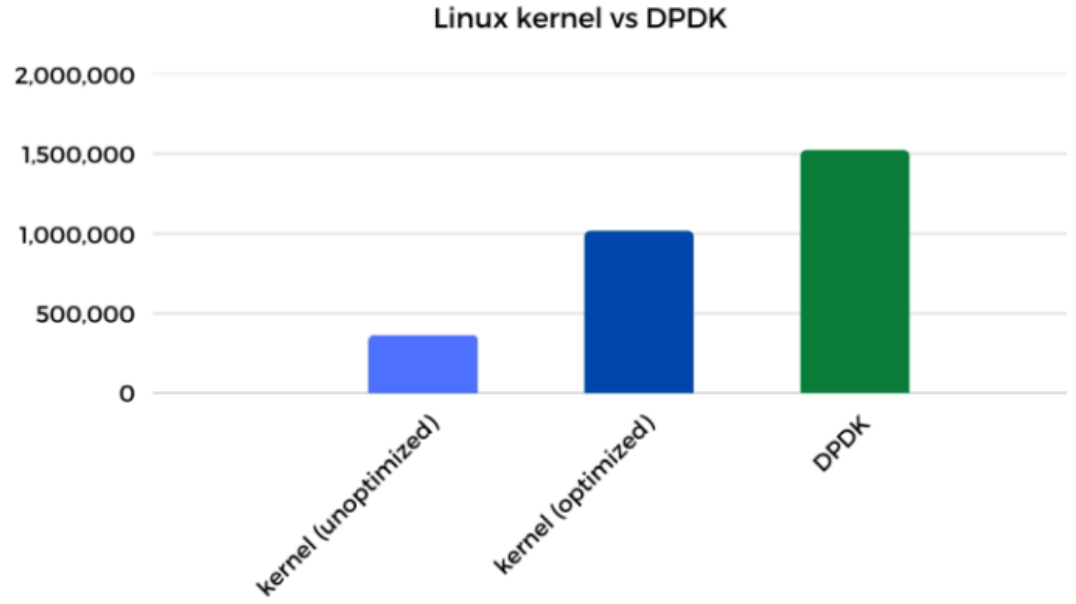




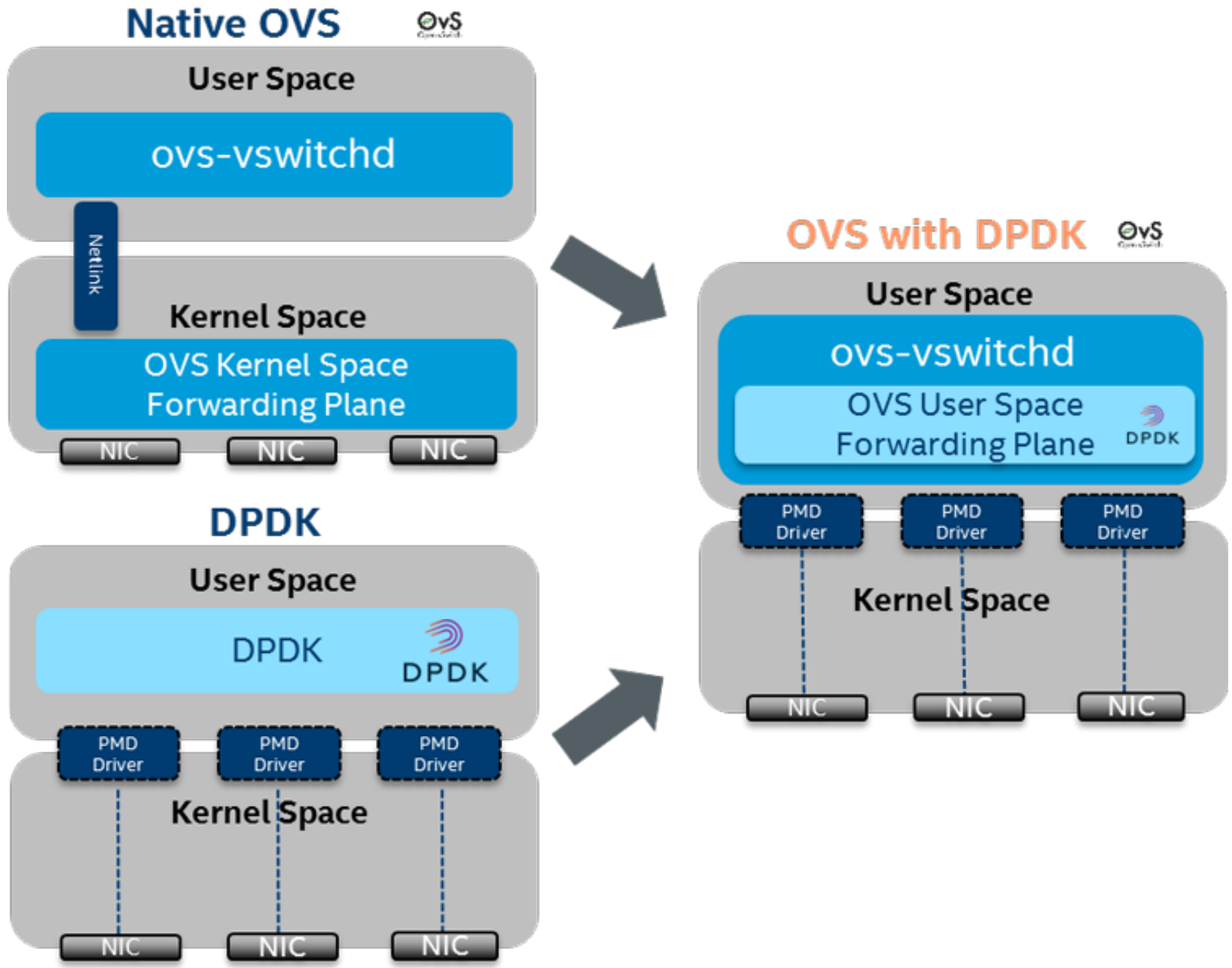
DPDK accélère le vitesse du trafic sous OVS

- pour les petites trames,
- en utilisant des machines multi-core,
- nécessite de réserver des CPU pour les échanges réseau.

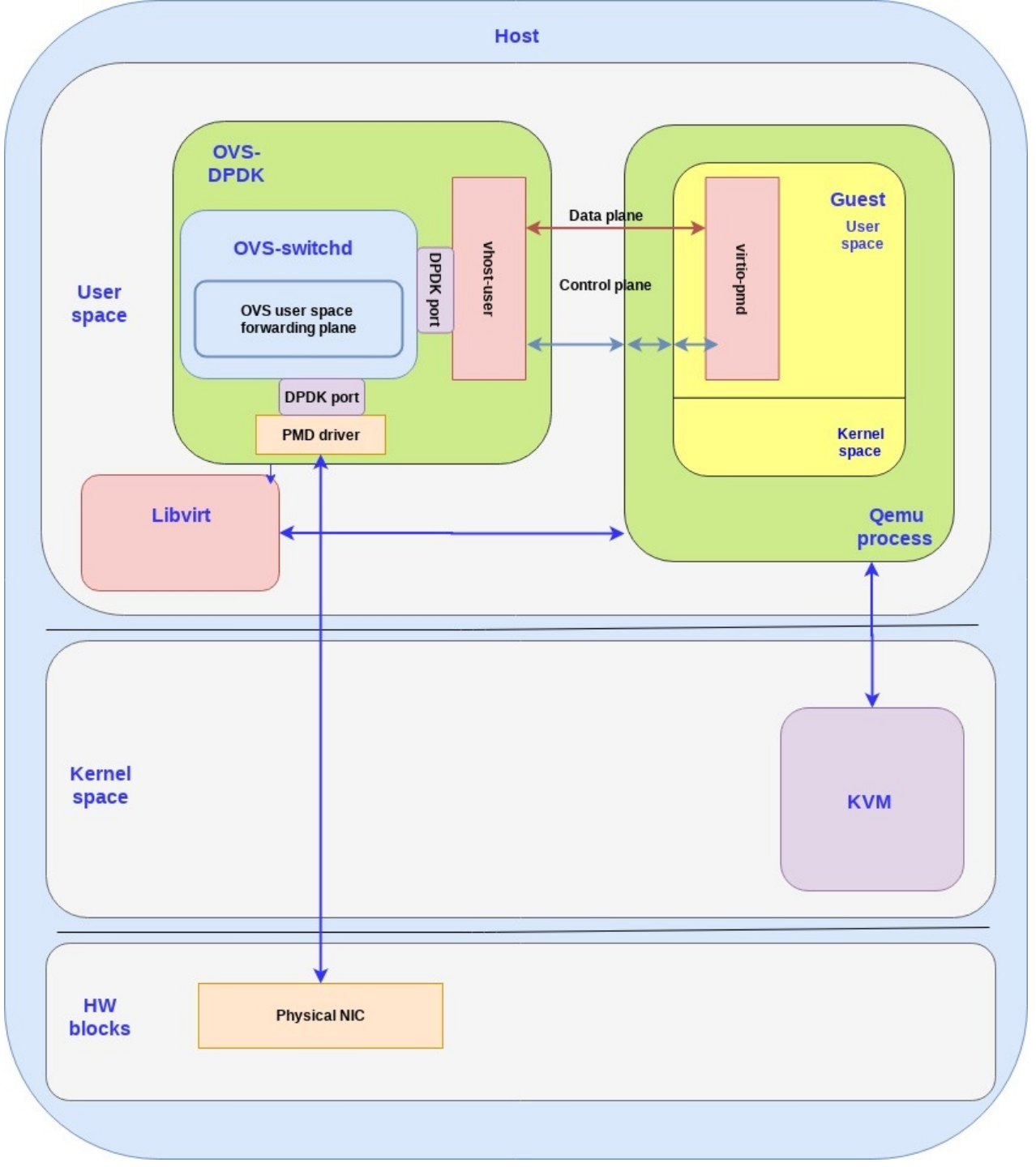
Le gain moyen est de 150 %



Architecture OVS-DPDK- 1



Architecture OVS-DPDK-2



Objectifs d'Open vSwitch (OVS) - 1

- Initialement **Open vSwitch,** appelé plus communément **OVS,** est une implémentation logicielle d'un commutateur de réseau **multicouche** (Couche 2 et 3) virtuel comme alternative à l'implémentation Linux Bridge (initialement Couche 2)
- C'est une implémentation **open-source** d'un **commutateur multicouche virtuel distribué** qui fournit une **pile de commutation** pour **les environnements de virtualisation matérielle** , tout en prenant en charge plusieurs protocoles et normes utilisés dans les réseaux informatiques.
- Cette fonction est **similaire aux solutions de commutation virtuelle propriétaires** telles que le VMware vSphere Distributed Switch (vDS) et Cisco Nexus 1000V
- Il a été porté dans presque tous les **hyperviseurs sous Linux** (Xen, KVM, Proxmox VE, Qemu, VirtualBox, Vmware ESX, Lxc, Docker, libvirt) et l'hyperviseur Microsoft HyperV

Objectifs d'Open vSwitch (OVS) - 2

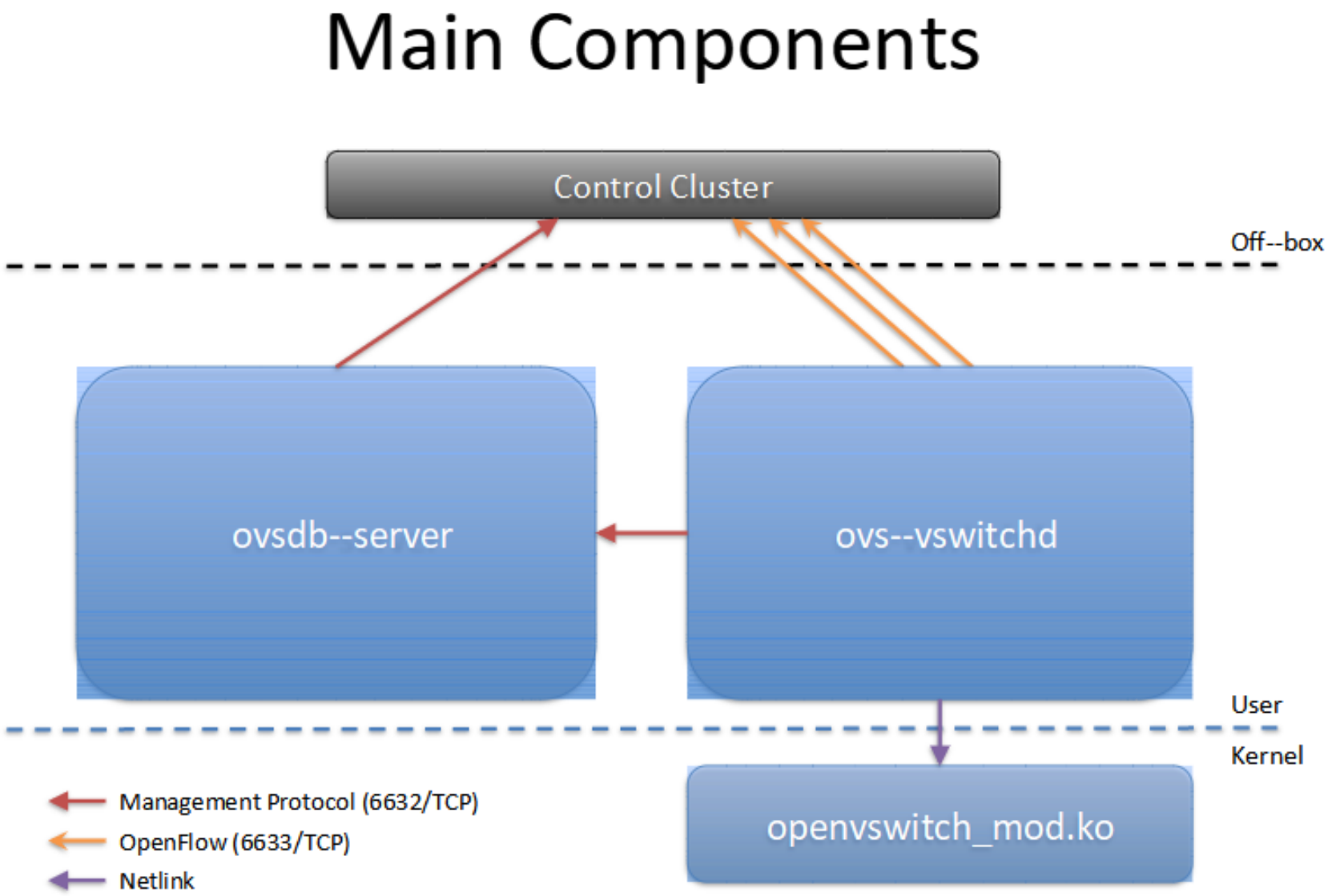
Il prend en charge des interfaces de gestion de

- de la sécurité : isolation vLan (802.1Q) , traffic filtering, spanning tree (STP et RSTP)
- du Monitoring : NetFlow, sFlow, SPAN, RSPAN (mirroring)
- de Liens : LACP, Multicast, Channel bounding
- des Interconnections avec des interfaces physiques : 802.1ag, GRE, VXLAN, IPsec, …
- de la QOS : Traffic queuing, traffic shaping

Il dispose d’outils de gestion de

- D’administration : OpenFlow, OpenDaylight, OpenStack
- L’automatisation : OpenFlow, OVSDB management protocol

Open vSwitch Architecture 1



Open vSwitch Architecture - 2

- **ovsdb-server** : est un serveur base de donnée ou sont stockés les différentes configurations du switch, celui-ci permet de garder une continuité dans le comportement du switch au-delà du redémarrage de celui-ci.
- **ovs-vswitchd** : est le processus principal dans le fonctionnement du switch, il permet entre autres la communication avec le contrôleur via OpenFlow ou encore la récupération des données se trouvant dans la base de données.
- **openvswitch.ko** (Module Kernel) : Le Kernel est le noyau du système d'exploitation il consiste en la partie qui s'occupe réellement de la commutation et du tunneling.

Open vSwitch : commandes

- 3 outils sont a disposition de l'administrateur pour configurer et surveiller l'état des switchs OVS : **ovs-vsctl, ovs-ofctl et ovs-dpctl.**
- **ovs-vsctl :** utilitaire pour interroger et configurer ovs-vswitchd, permet la gestion des **Bridges / Ports / Interfaces (internes/externes) / Controllers / Liens (ex vlan) / Agrégration de liens / Tunnels (GRE, IPSec)**
- **ovs-ofctl :** utilitaire pour interroger et gérer les tables et les fluxs SDN **OpenFlow**
- **ovs-dpctl** utilitaire pour interroger et gérer les chemins des données (**datapath**)

Open vSwitch : Exemples commandes

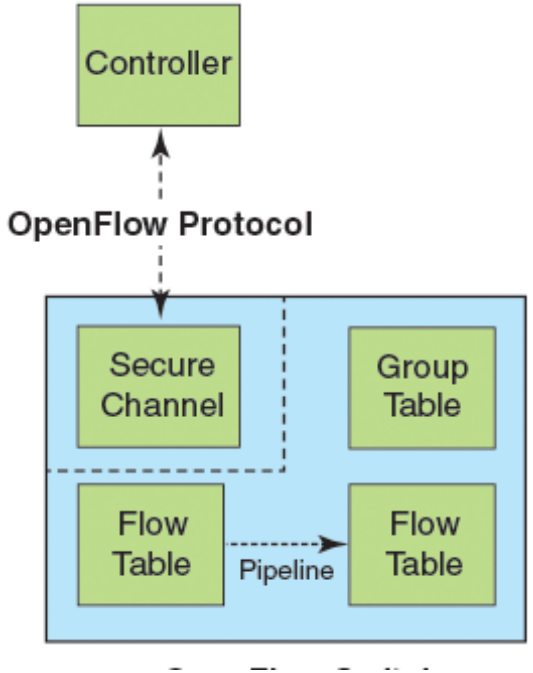
Création du commutateur virtuel br0
ovs-vsctl add-br br0
#up du bridge br0
ifconfig br0 up
Connecte votre l'interface réelle eth0 au bridge br0
ovs-vsctl add-port br0 eth0
Création d un port trunk1 des 4 vlans 1,10,11,100
ovs-vsctl add-port br0 trunk1 trunks=1,10,11,100
#Création d un port vlan10.1 associé au VLAN 10
ovs-vsctl add-port br0 vlan10.1 tag=10
affecte @IP a l'interface vlan10.1
ifconfig vlan10.1 192.168.10.2

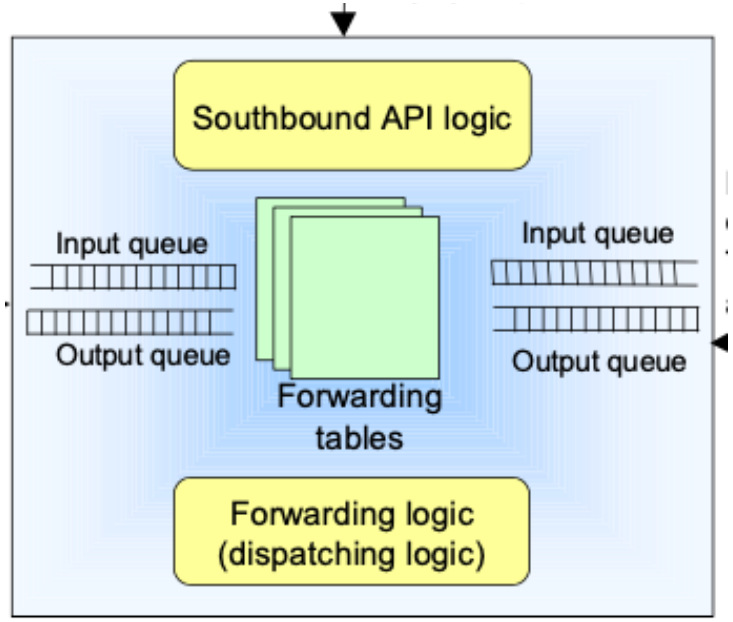
Types d’interconnexion d’un commutateur OVS

- Interfaces internes (virtuelles) ou externes (réelles)
  - $ ovs-vsctl add-port br0 eth0
- VLAN-tagged avec ajout d’une Quality of Service (QoS)
  - $ ovs-vsctl add-port br0 tap0 tag=100
  - $ ovs-vsctl set interface tap0 ingress_policing_rate=10000
- Tunnels VXLAN
  - $ ovs-vsctl add-port br0 vxlan0 -- …
- Tunnels IPsec + GRE
  - $ ovs-vsctl add-port br0 ipsec_gre0 -- …
- Chemin de données (datapath)
  - $ ovs-vsctl add-br br0 -- set bridge br0 datapath_type=netdev
  - $ ovs-vsctl add-port br0 phy0 -- set Interface phy0 type=dpdk options:dpdk-devargs=0000:01:00.0 ofport_request=1 …

Le protocole **OpenFlow**

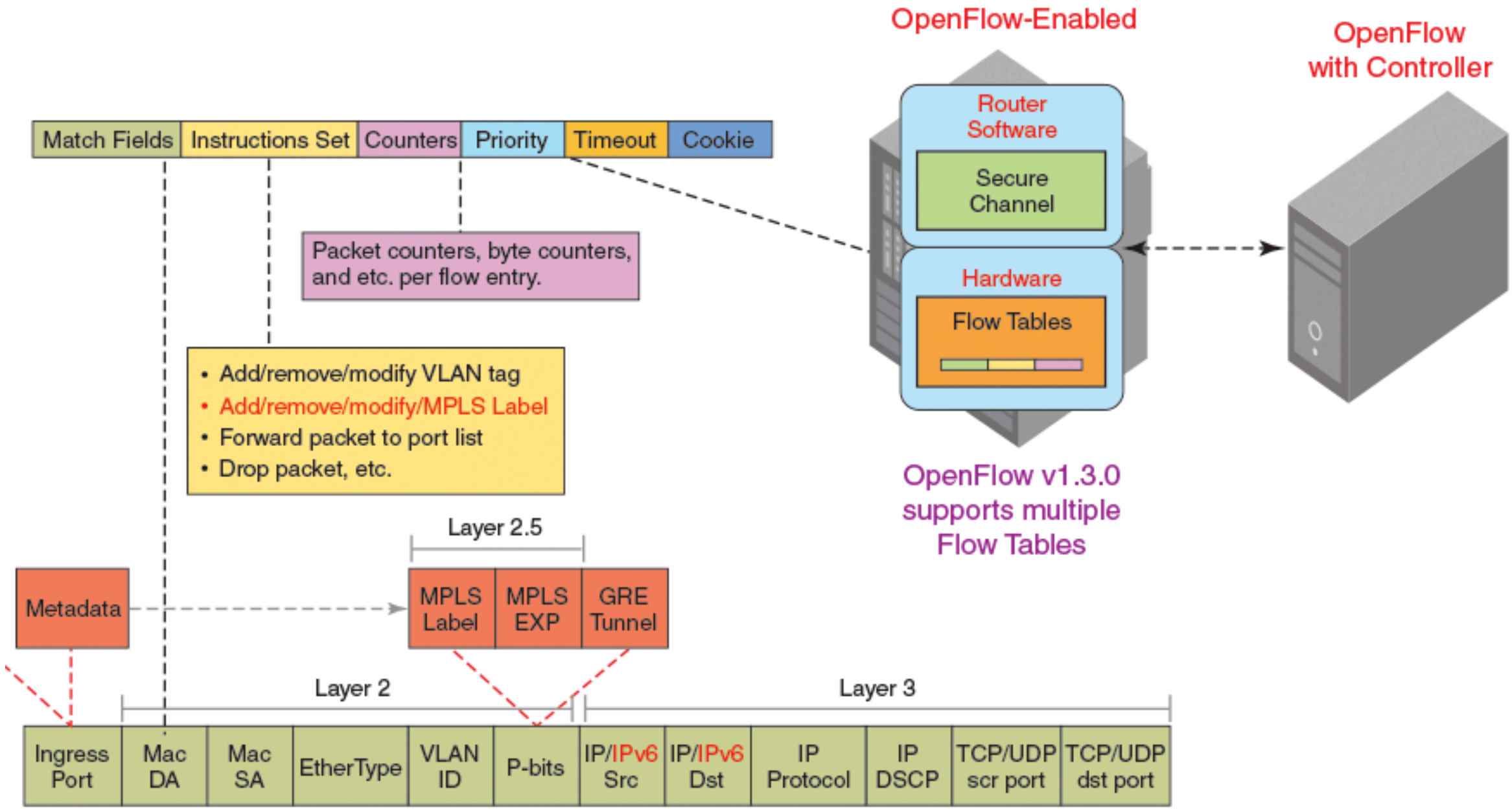
- Les commutateurs Ethernet et les routeurs les plus modernes contiennent **des tables de flux qui sont utilisées pour effectuer des fonctions de transfert selon les couches 2,3 et 4**, indiquées dans les en-têtes de paquets.
- Bien que chaque fournisseur ait des tables de flux différentes, il existe **un ensemble commun de fonctions** pour une large gamme de commutateurs et de routeurs
- **Le protocole OpenFlow** permet aux administrateurs de réseau de programmer les tables de flux (flow tables) dans leurs différents routeurs IP et commutateurs Ethernet afin de séparer les trafics, chacun avec son ensemble de fonctionnalités et caractéristiques de flux.
- Ainsi ce protocole entre un **contrôleur central OpenFlow et un commutateur OpenFlow est utilisé pour programmer la logique de transfert ou d'acheminement du commutateur.**



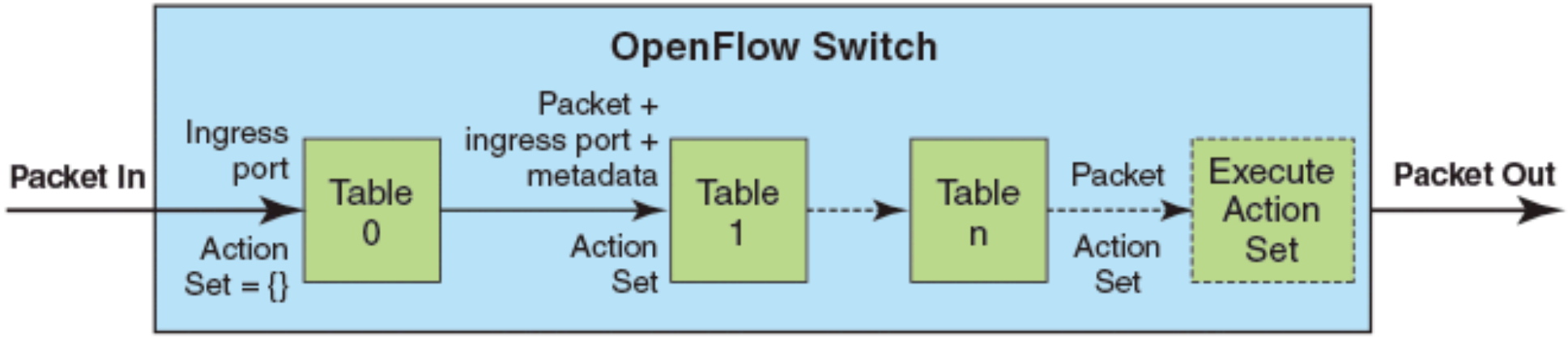


Le protocole OpenFlow 1.3.x :Table de flux

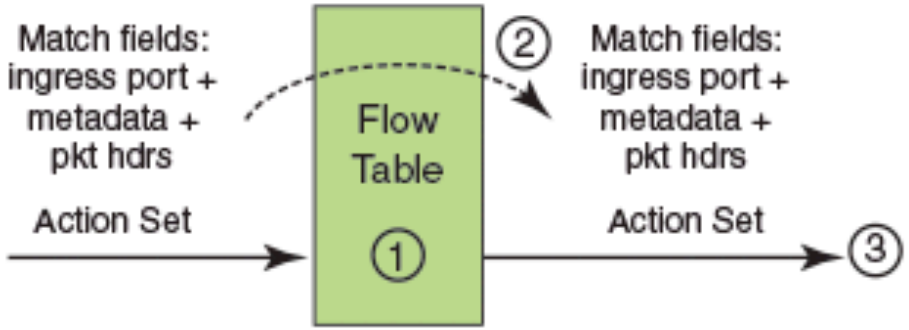
- Chaque table de flux du commutateur contient un ensemble d'entrées de flux qui présentent les règles d'acheminement des paquets.
- Chaque entrée de flux est associée à zéro ou plusieurs actions qui dictent la façon dont le commutateur gère les paquets correspondants. Si aucune action n'est présente, le paquet est supprimé. Les listes d'actions présentes dans les entrées de flux doivent être traitées dans l'ordre spécifié (grâce sa priorité).
- Une entrée de flux est composée de : **Match_fields:Counters:priorité:Actions**
  - **Match fields** : champs de correspondance qui définissent le modèle du flux de paquets à travers l'instanciation des champs d'en tête allant de la couche Ethernet à la couche Transport : ex => **Port, VLAN, @Ethernet destination et source, @IP destination et source, TCP/UDP destination et source**
  - **Counters** : des compteurs sur les paquets : ex => **par Flow table, par Flow entry, par Port, par Queue**
  - **Actions** : Actions à appliquer aux paquets qui correspondent à l'entrée de flux : ex => **Forward, Drop, Enqueue, Modify Field, Re-submit, Learn**


OpenFlow-Enabled
OpenFlow with Controller
Router Software
Secure Channel
Hardware
Flow Tables
Match Fields
Instructions Set
Counters
Priority
Timeout
Cookie
Packet counters, byte counters, and etc. per flow entry.
• Add/remove/modify VLAN tag
• Add/remove/modify/MPLS Label
• Forward packet to port list
• Drop packet, etc.
OpenFlow v1.3.0 supports multiple Flow Tables
Layer 2.5
Metadata
MPLS Label
MPLS EXP
GRE Tunnel
Layer 2
Layer 3
Ingress Port
Mac DA
Mac SA
EtherType
VLAN ID
P-bits
IP/IPv6 Src
IP/IPv6 Dst
IP Protocol
IP DSCP
TCP/UDP scr port
TCP/UDP dst port

Enchaînement des étapes du pipeline des actions



{a} Packets are matched against multiple tables in the pipeline



1. **Find highest - priority matching flow entry**
2. **Apply instructions:**
   i. Modify packet & update match fields (apply actions instruction)
   ii. Update action set (clear actions and/or write actions instructions)
   iii. Update metadata
3. **Send match data and action set to next table**

{b} Per-table packet processing

Le protocole OpenFlow : exemple de flux

- $ ovs-ofctl add-flow br0 \ "table=0, dl_src=01:00:00:00:00:00/01:00:00:00:00:00, actions=drop »
  - **Les paquets avec une adresse source multidiffusion ne sont pas valides.**
- $ ovs-ofctl add-flow br0 \ "table=2 actions=learn(table=10, NXM_OF_VLAN_TCI[0..11], \ NXM_OF_ETH_DST[]=NXM_OF_ETH_SRC[], \ load:NXM_OF_IN_PORT[]->NXM_NX_REG0[0..15]), \ resubmit(,3) »
  - **Apprentissage MAC+VLAN pour le port d'entrée**

Native **OVS firewall driver**

- Historiquement, Open vSwitch (OVS) ne pouvait pas interagir directement avec iptables pour mettre en place des groupes de sécurité.
- L'agent OVS et le service de calcul utilisent un bridge Linux entre chaque instance de VM et le bridge d'intégration OVS br-int pour mettre en œuvre les groupes de sécurité.
- Le dispositif de bridge Linux contient les règles iptables relatives à l'instance.
- Utilise les modules noyau **conntrack** et **nf_conntrack_proto_gre**
- Exemples protection spoofing arp :
  - table=71, priority=95,arp,reg5=0x1,in_port=1,dl_src=fa:16:3e:a4:22:10,arp_spa=192.168.0.1 actions=resubmit(,94)
  - table=71, priority=95,arp,reg5=0x1,in_port=1,dl_src=fa:16:3e:8c:84:13,arp_spa=10.0.0.1 actions=resubmit(,94)

Contrôleur FAUCET

- Faucet est un contrôleur OpenFlow compact et optimisé open source incluant de nouvelles fonctionnalités et extensions :
  - Supporte les matériels compatibles Openflow v1.3.x : OpenvSwitch, Lagopous, HPE Aruba, Allied Telesis, Noviflow, Netronome, Northbound Networks, Cisco Catalyst 9000.
  - DPDK
  - Routage Inter VLAN, statique et BGP.
  - Tableaux de bord basés sur Grafana pour la surveillance
  - Intégration de Prométhée pour la surveillance et l'instrumentation de FAUCET
  - Listes de contrôle d'accès flexibles (ACLs) basées sur les ports et le VLAN
  - Plan de données pour les Network Function Virtualisation (NFV) => Fonctions de déchargement telles que DHCP, NTP, Pare-feu et IDS

Open vSwitch : exemple_1 de déploiement sur une machine personnelle

- Sous Linux, FreeBSD, NetBSD, Raspberry ou Docker :
  - Installer des modules supplémentaires au noyau si nécessaire :
    - tun, iproutes2, openvswitch, nf_*, iptable_nat, xt_*
  - Installer les packages openvswitch nécessaires :
    - **openvswitch-common, openvswitch-switch**,
    - openvswitch-switch-dpdk, openvswitch-ipsec, openvswitch-pki, openvswitch-vtep
  - Lancer les services openvswitch nécessaires :
    - systemctl start openvswitch-switch.service
    - systemctl start ovs-vswitchd.service
    - systemctl start ovsdb-server.service
- Sous Windows, l'installation semble possible sous HyperV

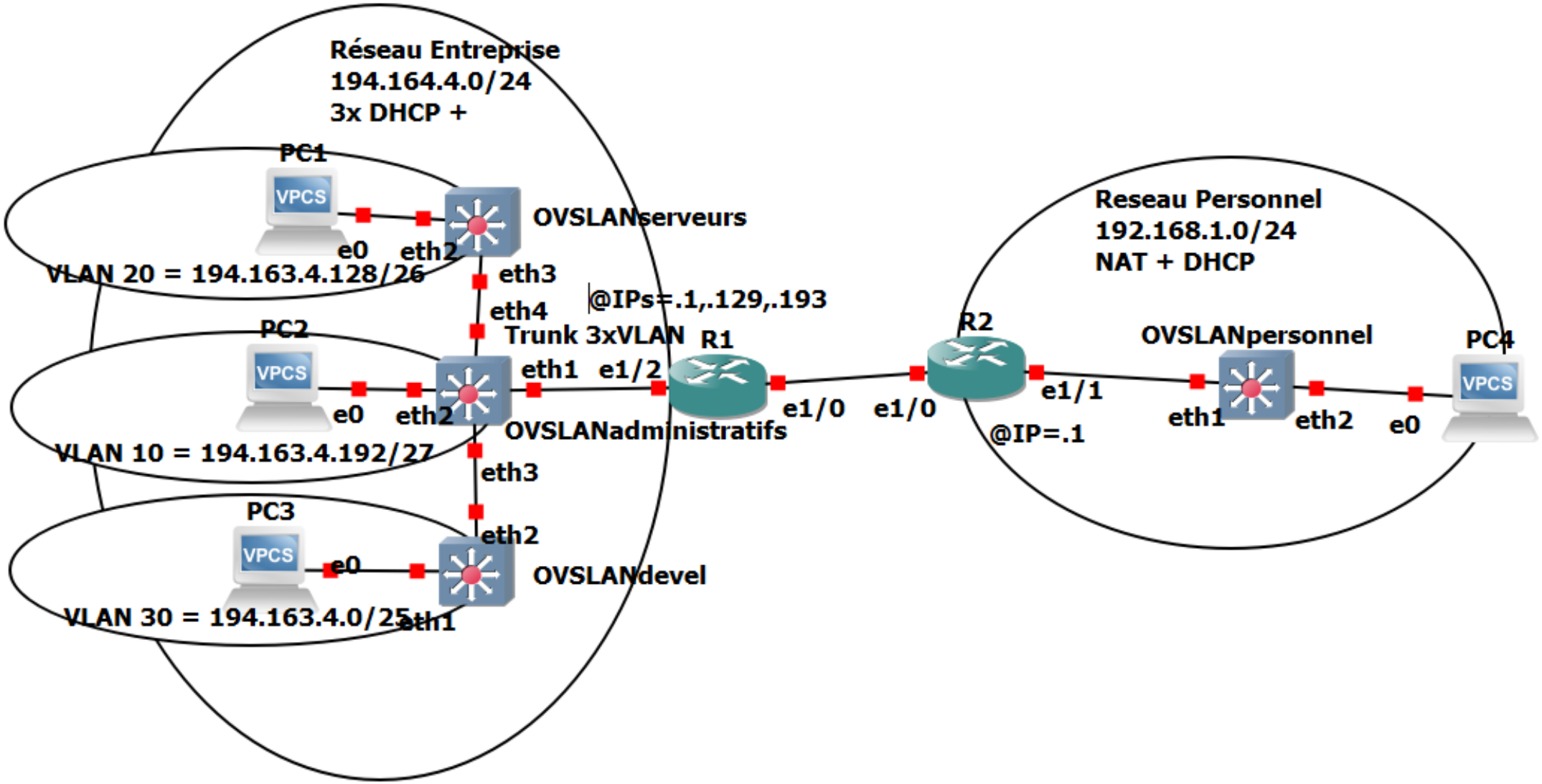
Exemple_2 de déploiement sous GNS3 : Réalisation de 4 VLAN/LAN utilisant des OVS interconnectés par 2 routeurs sous GNS3

- Installer **GNS3**, puis installer **Virtualbox**, puis installer **GNS3_VM**, puis installer **l'appliance openvswitch** sous GNS3
- Réalisation de 4 LANs dont 3 VLANs

  Les réseaux (LAN) d'entreprise => bloc d'adresses IP = 194.164.4.0/24
  - VLAN 10 : les admistratifs :14 hôtes
  - VLAN 20 : les serveurs : 28 hôtes
  - VLAN 30 : les développeurs : 62 hôtes

  Le réseau « personnel » => bloc d'adresses IP = 192.168.1.0/24
  - Pas de VLAN : un seul réseau privé personnel
- 2 routeurs R7200 interconnectés
- **Des bridges openvswich** remplacent ici les bridges GNS3 de base

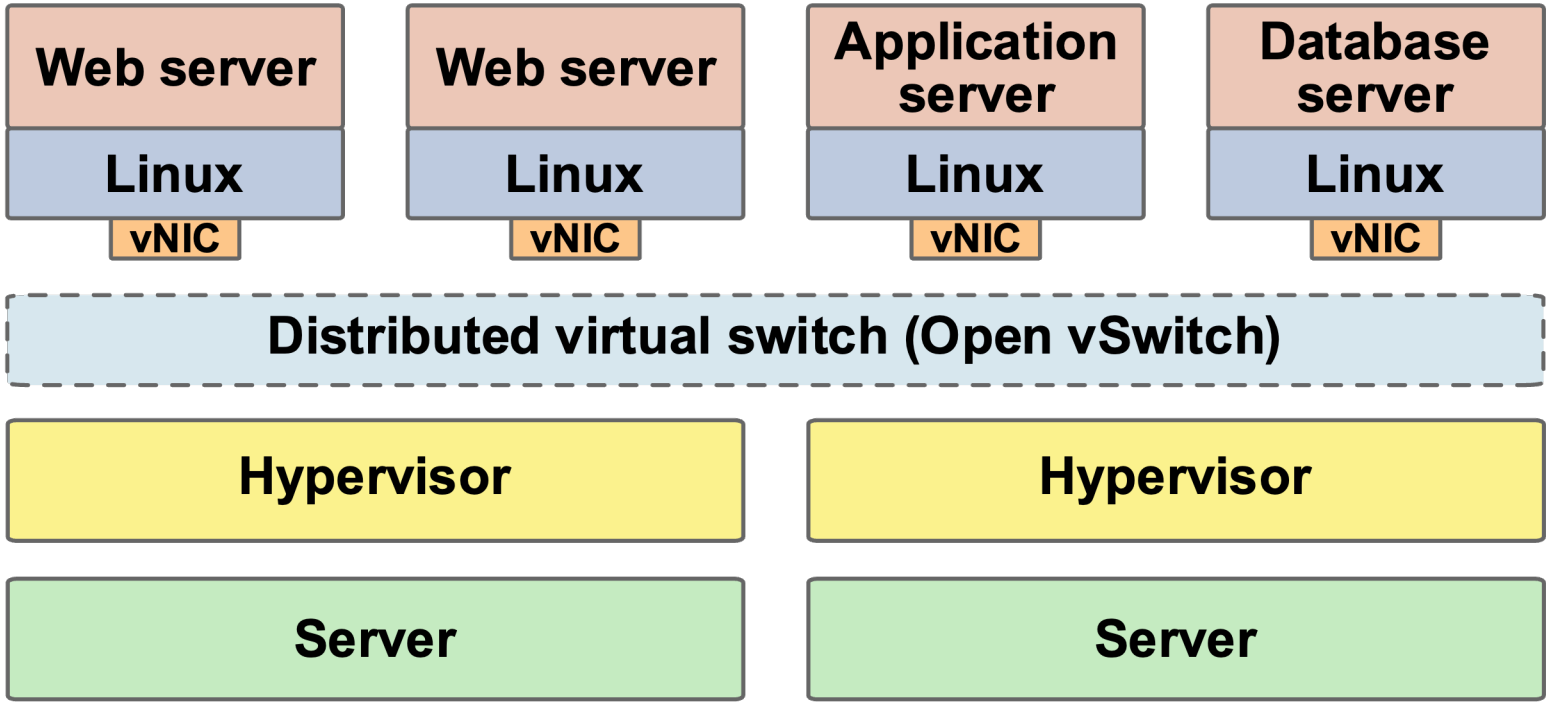
Exemple_2 de déploiement sous GNS3 : Réalisation de 4 VLAN/LAN utilisant des OVS interconnectés par 2 routeurs



URLs pour réaliser le projet

- GNS3 all in one :
- https://www.gns3.com/software/download
- Virtualbox Oracle :
- https://www.virtualbox.org/wiki/Downloads
- GNS3 VM :
- https://www.gns3.com/software/download-vm
- GNS3 Appliance openvswitch: https://gns3.com/marketplace/appliances/open-vswitch
- Cisco IOS 12.4.13 :
- https://dept25.cnam.fr/~jb

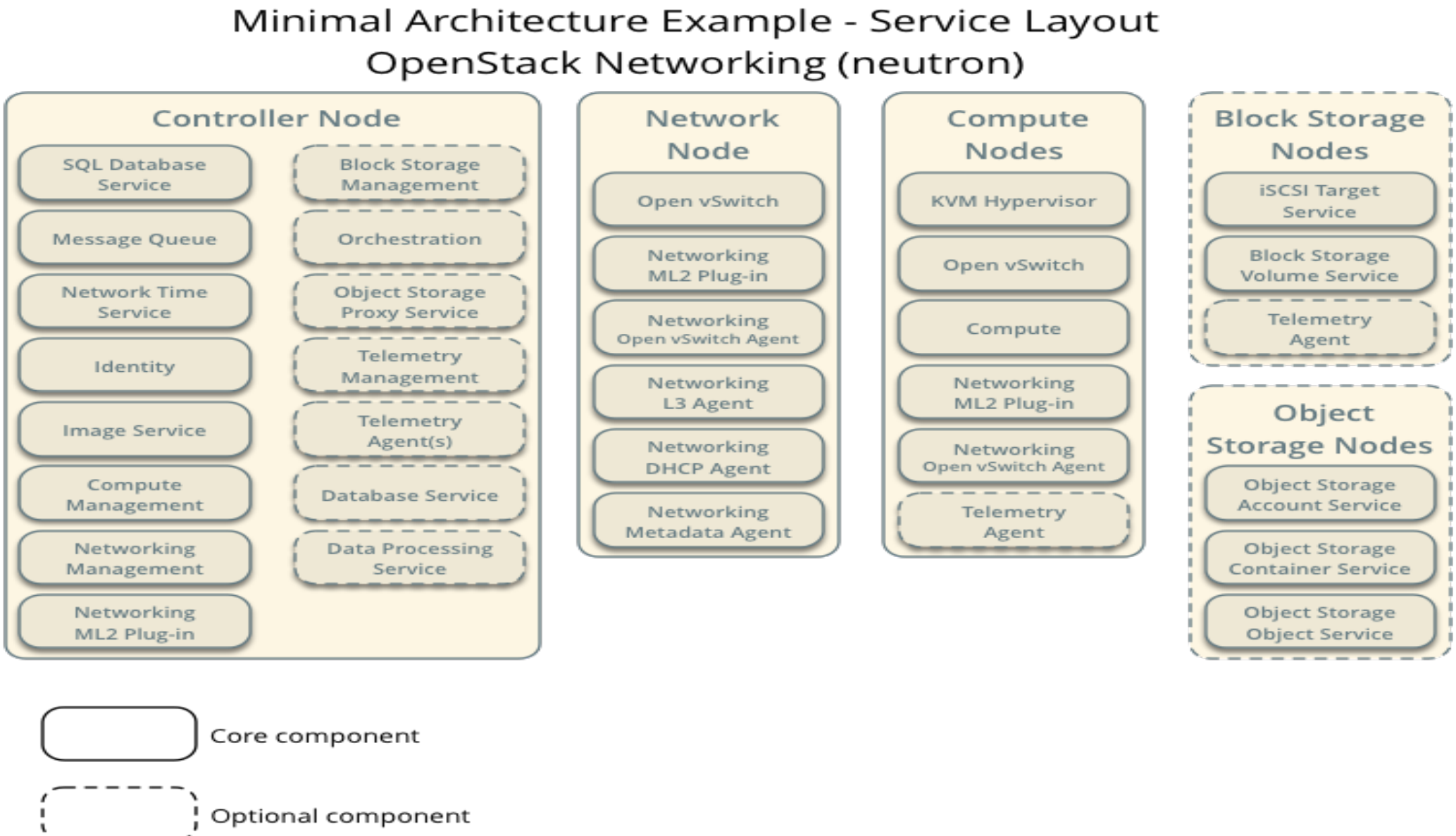
Exemple_3 de déploiement sur un Cluster de plusieurs réseaux logiques **isolés** sur un seul réseau physique



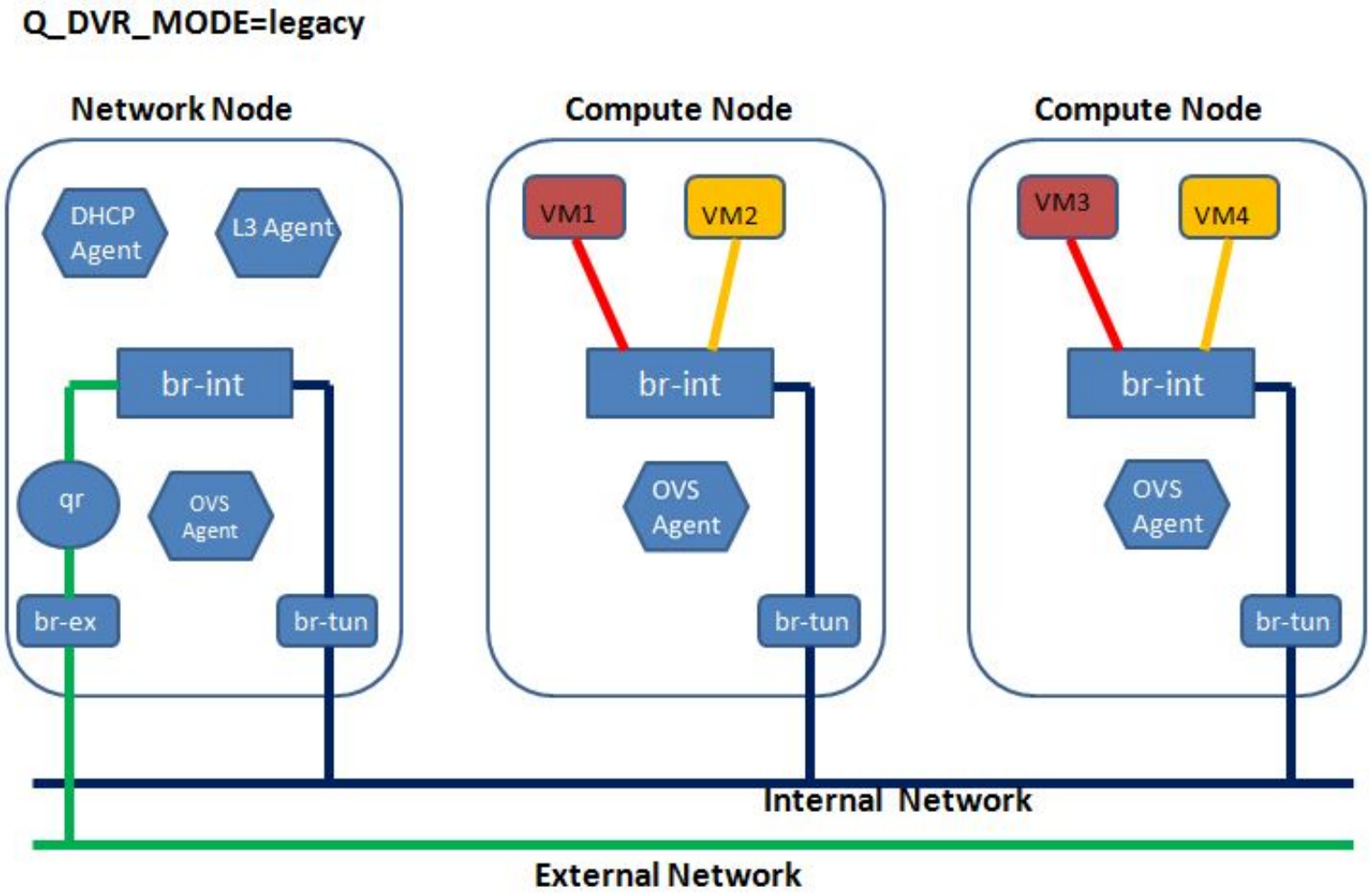
Exemple_3 : OpenStack (1)

- **OpenStack** est basé sur une architecture **distribuée modulaire** composée de plusieurs services :
  - **NOVA :** virtualisation des machines
  - **SWIFT/CINDER :** virtualisation du stockage respectivement stockage par objets et par blocs
  - **NEUTRON : virtualisation du réseau => gestion du réseau**
    - Si on effectue une installation « self-service » on peut utiliser OpenVSwitch
    - Les administrateurs de projet peuvent créer des réseaux virtuelles
    - Plusieurs types de déploiement
      - Avec tolérance aux pannes
      - Sans tolérance aux pannes

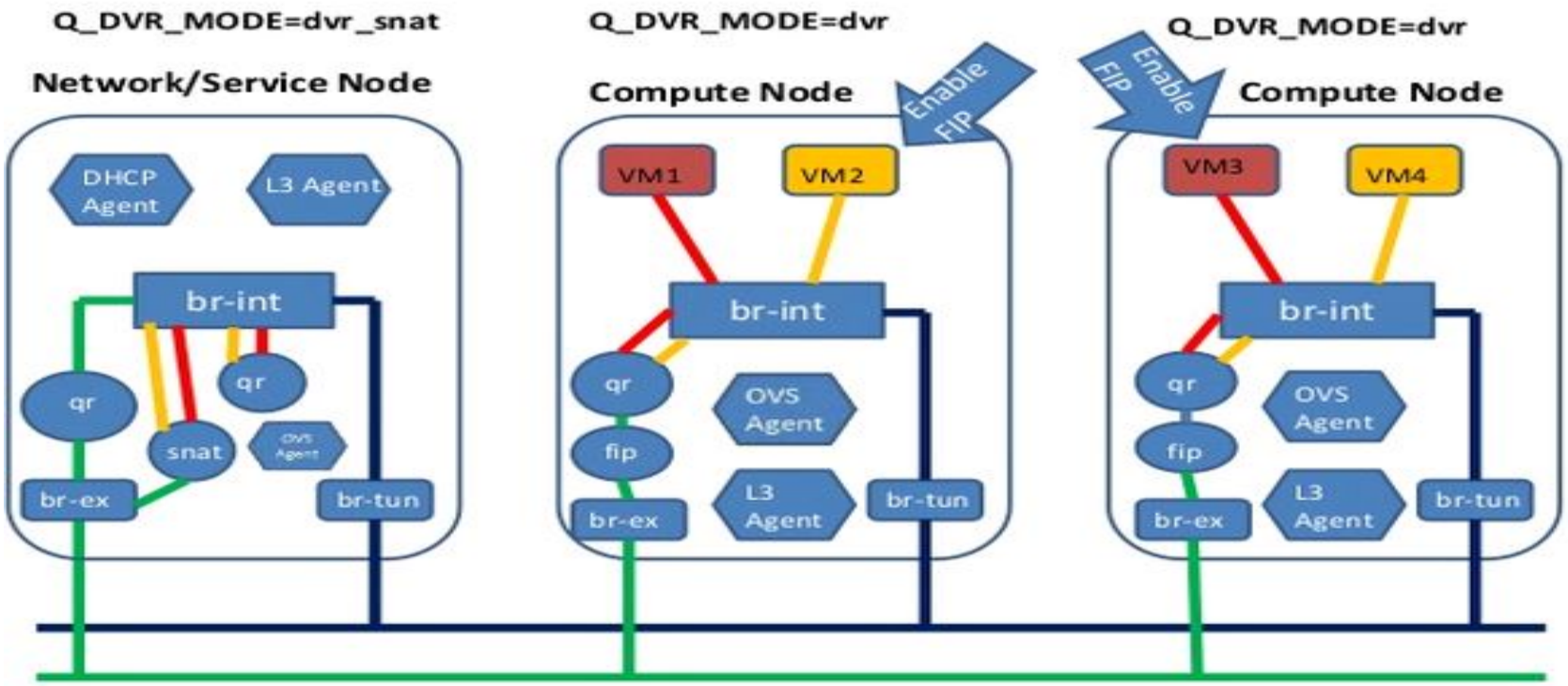
Exemple : OpenStack (2) répartition des services



Exemple_3 : Openstack (3) => Cluster de 2 machines (Compute Node) et d'un acces réseau unique (routeur)



Exemple_3 : Openstack (4) => Optimisation du réseau virtuel du Cluster de 2 machines (Compute Node) et d'un acces réseau distribué (3 routeurs)



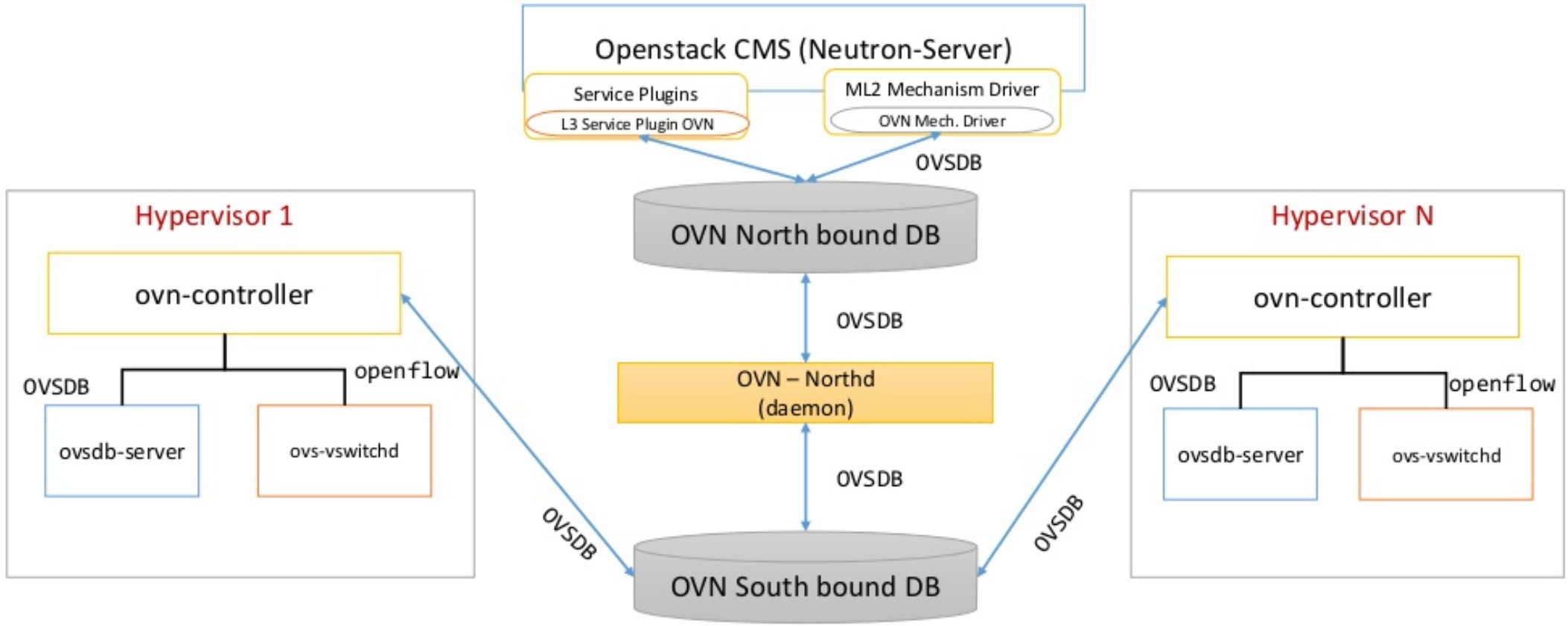
Module Neutron d'Openstack

- Donne une API pour créer des topologies de réseau riches et configurer des politiques de réseau avancées dans le cloud.
- Donne accès a des capacités réseau avancées: Ex => tunneling L2-in-L3 pour éviter les limites de VLAN, garanties de QoS de bout en bout, utilisation des protocoles de surveillance comme NetFlow, firewall, routage distribué, loadbalancer, …
- Interface Web Horizon GUI pour créer/supprimer des réseaux et des sous-réseaux Neutron L2 et L3
- Plugins pour s'interconnecter avec d'autres équipements réseau externes : Cisco, Juniper, …

Open Virtual Network (OVN)

- OVN complète les capacités existantes d'OVS en ajoutant un support natif pour les abstractions de réseau virtuel, comme les superpositions L2 et L3 virtuelles et les groupes de sécurité.
- OVN a été conçu pour fournir une mise en réseau virtuelle. Il fournit une couche d'abstraction plus élevée que Open vSwitch, en travaillant avec des routeurs et des commutateurs logiques, plutôt qu'avec des flux. OVN est destiné à être utilisé par les logiciels de gestion des nuages (CMS).
- Voici quelques fonctionnalités ajoutées par OVN :
  - Routeurs virtuels distribués
  - Commutateurs logiques distribués
  - Listes de contrôle d'accès
  - Serveurs DHCP et DNS

OVN – Architecture

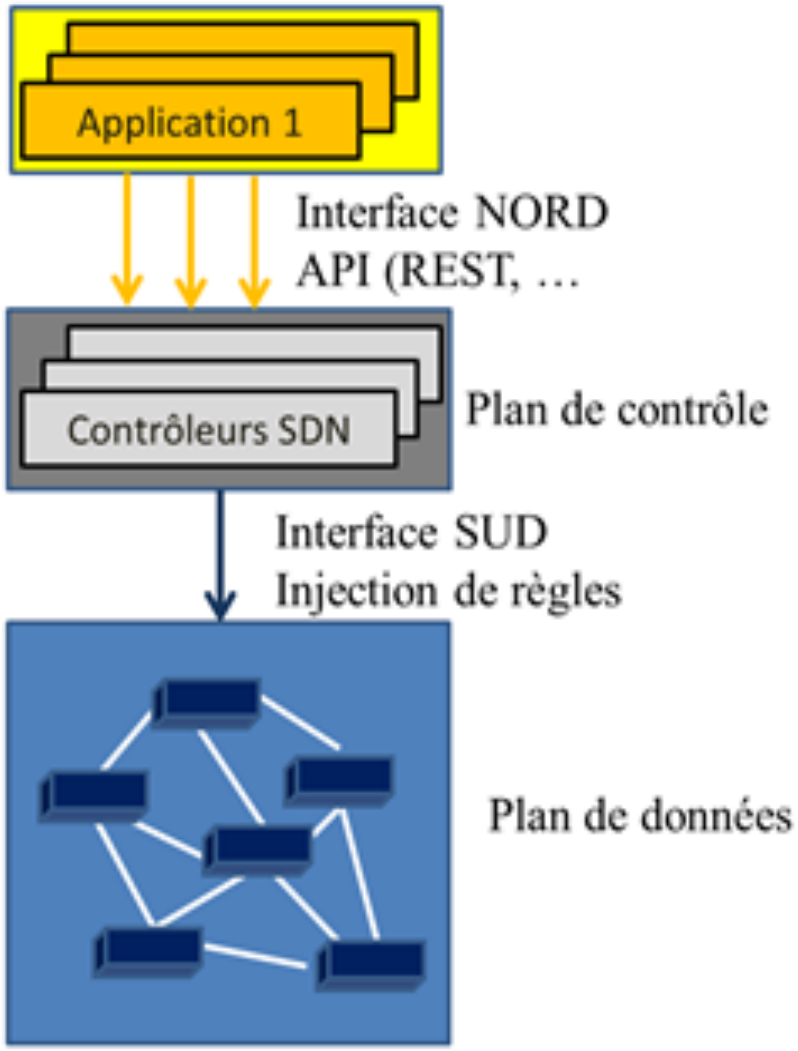


ovn-northd

Translate between the logical network elements configured by the CMS to the Northbound DB and model them to the Southbound DB tables, which holds the physical/infrastructure bindings and the logical flows which enable the logical connectivity.

Architectures OVN ~ SDN

- Objectifs : Piloter une infrastructure réseau par un logiciel.
- centraliser et mutualiser le plan de contrôle (tables de routages , protocoles de routages) entre tous les équipements
- réduire les équipements à leur fonction la plus élémentaire : transmettre (ou ne pas transmettre) des données sur le réseau
- les serveurs chargés de fournir le plan de contrôle de la solution SDN sont généralement appelés contrôleurs SDN



Webographie

- DPDK Intel NIC Performance Report Release 17.05. Mai 2017. http://fast.dpdk.org/doc/perf/ DPDK_17_05_Intel_NIC_performance_report.pdf.
- R. Rosen. Network acceleration with DPDK. Linux Weekly News, Juillet 2017. https://lwn.net/ Articles/725254/.
- Network Functions Virtualisation – Introductory White Paper. Dans SDN and OpenFlow World Congress, Darmstadt, Octobre 2012. https://portal.etsi.org/NFV/NFV_White_Paper.pdf.
- OpenStack Docs: Configuration du Réseau et https://docs.openstack.org/neutron/latest/admin/ovn/index.html