TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ACL	Access Control List
ARP	Address Resolution Protocol
CPU	Central Processing Unit
CTPD	Cloud Tracing, Profiling and Debugging
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
IPTV	IP Television
KVM	Kernel-based Virtual Machine
LTTng	Linux Tracing Toolkit next generation
MIB	Management Information Base
MRTG	Multi Router Traffic Grapher
MTU	Maximum Transfer Unit
NMS	Network Management System
ODBC	Open DataBase Connectivity
OID	Object Identifier
POP	Post Office Protocol
QoS	Quality of Service
RRD	Removable Rigid Disk
SGBD	Système de Gestion de Base de Données
SNMP	Simple Network Management Protocol
SSH	Secure Shell

SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TTL	Time To Live
VEB	Virtual Ethernet Bridge
VL2	Virtual Layer 2
VLAN	Virtual Local Area Network
VLB	Valiant Load Balancing
VM	Virtual Machine
VoIP	Voice over IP
VPN	Virtual Private Network
V-WAN	Virtual Wide Area Network

CHAPITRE 1

INTRODUCTION GÉNÉRALE

1.1 Contexte

À leurs naissances, les ordinateurs étaient des gros mainframes, stockés dans des chambres. Ces chambres seront nommées plus tard des centres de données. Dans les années 1980s, l'industrie des ordinateurs a connu le boom des micro-ordinateurs. Désormais, les ordinateurs peuvent être utilisés dans les bureaux. Ainsi les centres de données sont morts.

Au début des années 2000s, Salesforce.com est venu avec une idée révolutionnaire. Soit la proposition des applications d'entreprise à partir d'un simple site web standard, accessible à travers un navigateur web. Autrement dit, Salesforce.com offrait des services informatiques distants, qui sont les applications d'entreprises, accessibles à travers l'internet. C'est le début de la délocalisation des centres de données hors des locaux des entreprises.

Depuis, d'autres opérateurs de centres de données ont offert l'hébergement de toutes sortes de services informatiques pour les entreprises, épargnants à ces dernières les coûts d'installations et de maintenances. Et vu ces gains économiques, le passage des compagnies et de leurs clients aux centres de données distants est en hausse. Ainsi, les données en ligne sont en explosion.

Pour accompagner cette hausse des données en ligne, les propriétaires des réseaux de fibres optiques fournissent plus de bande passante aux cœurs de réseaux. Ainsi pour les équipementiers, ils fournissent des commutateurs qui correspondent aux vitesses et capacités des réseaux. Et c'est de même que des évolutions ont été apportées aux technologies des centres de données.

Les centres de données ne peuvent plus être des simples unités d'hébergement de données. Il s'agit d'un modèle économique pour les entreprises, alors les centres de données doivent

offrir une efficacité des coûts. Le coût financier est largement entrainé par la consommation d'énergie.

La virtualisation est une technologie réponse à l'efficacité énergétique. Elle offre un moyen avec lequel le même traitement peut être fait sur moins d'équipements, en augmentant l'utilisation de ces équipements. Par conséquent, cela permet une utilisation plus efficace de l'énergie à moindre coût (Hammadi et Mhamdi, 2014). Alors, la virtualisation a été introduite aux réseaux d'interconnexion des centres de données, formés par des commutateurs.

La virtualisation des commutateurs, dans les réseaux d'interconnexion des centres de données, est la réutilisation d'une seule ressource physique pour plusieurs autres commutateurs, dits virtuels ; ou bien, la consolidation de plusieurs de ces ressources physiques, pour obtenir un commutateur virtuel plus puissant, avec plus de fonctionnalités (Clayman, Galis et Mamatas, 2010). Les ressources physiques sont, parmi autres, des groupes de CPU, de mémoires, d'interfaces réseaux, etc …

La virtualisation offre des avantages lié à la rapidité du déploiement et d'opération, l'extensibilité et la migration des commutateurs virtuels. En effet, un commutateur virtuel n'est qu'un fichier image de commutateur virtuel et n'est pas lié à un équipement physique. Alors, les commutateurs démarrent et arrêtent plus rapidement que les commutateurs physiques. Pour la migration, il suffit de copier le fichier image d'une machine à une autre. Et en cas de besoin de plus de capacité physique, il suffit de copier des fichiers à une machine plus puissante, sans aucun besoin de reconfiguration.

L'émergence de la virtualisation est accompagnée d'un suivi par les travaux de recherche, dont deux ont collaboré avec ce mémoire. Le projet V-WAN (Virtual Wide Access Network) de l'équipementier de télécommunications Ciena (Ciena, 2014) développe une solution intégrale, offrant entre autres, la supervision des équipements réseautiques virtuels. Et le projet CTPD (Cloud Tracing, Profiling and Debugging) de l'équipe de recherche sur le

LTTng (DORSAL, 2013), adapte son outil de traçage du système d'exploitation Linux, pour tracer les différentes couches de la virtualisation.

1.2 Problématique

Les grands avantages de la virtualisation des commutateurs, dans les réseaux d'interconnexion des centres de données, ne doivent pas laisser oublier les défis que le commutateur doit toujours porter : traiter les entêtes de paquets reçus, et commuter ces paquets vers les interfaces appropriées; et cela sans pertes de données, et dans les moindres délais possibles. D'où l'importance de la supervision de la performance des commutateurs.

La supervision de la performance des commutateurs permet de détecter les problèmes au niveau des nœuds des réseaux. Une connaissance préalable de la performance des commutateurs est nécessaire pour administrer et optimiser la performance du réseau en entier lors des moments critiques, où le réseau fait face à de grandes charges. Et, avoir un service robuste et une qualité de service constante.

Plusieurs travaux de recherche ont traité les commutateurs physiques (Yu et autres, 2011). De nos jours, des bilans de performance peuvent être tracés pour les commutateurs physiques. Mais la nature de l'environnement virtuel des commutateurs dans les centres de données rend la détection de la prévenance des chutes de performance particulièrement difficile, et parfois sollicite des manières différentes de supervision de la performance.

Dans ce mémoire, nous repondons aux cinq questions suivantes :

1- la définition de métriques décrivant la performance des commutateurs virtuels : Pour toutes les opérations exécutées par le réseau d'interconnexion des centres de données, il est nécessaire de savoir les points forts et les limites de chaque commutateur virtuel (Chen et Hu, 2002). Alors, quelles sont les métriques de performance nécessaires pour décrire un commutateur virtuel ?

2- la collecte de données au moindre coût : Afin d'avoir des données précises, les méthodes de collecte de données depuis les commutateurs, ne doivent pas avoir une influence sur les valeurs des métriques de la performance à extraire. Alors, quels sont les méthodes de collecte des données de la performance des commutateurs à utiliser, sans avoir à se soucier de l'exactitude des données retournées ?

3- la collecte automatique des données : Comme les variations en performance sont souvent imprévues, il faut une supervision permanente de la performance. Aussi, vu la complexité du réseau d'interconnexion des centres de données, et le grand nombre de commutateurs dans ce réseau, il est impossible de superviser la performance manuellement. Alors, comment collecter les données, des commutateurs virtuels, d'une manière dynamique et automatique ?

4- l'impact de la couche physique sur la performance des commutateurs virtuels : Sachant qu'il y a une machine physique, qui opère comme hôte au commutateur virtuel; et que la machine physique peut être l'hôte de plusieurs commutateurs virtuels, souvent, circulant des flux hétérogènes; comment évaluer l'influence de la couche physique, sur la performance des commutateurs virtuels ?

5- la performance des commutateurs virtuels : Alors que la virtualisation offre un modèle économique très rentable en termes coût, est-ce que les commutateurs virtuels peuvent être aussi performants que les commutateurs physiques ?

D'autres questions sont à traiter dans la problématique de la supervision de la performance des réseaux virtuels des centres de données. Notamment, la supervision des flux de données et de la performance des routes dans les réseaux virtuels des centres de données. Mais vu la limite du temps de la maîtrise, le mémoire se concentre juste sur les cinq questions précédemment décrites.

1.3 Objectifs du mémoire

L'objectif principal du mémoire est de développer un outil pour automatiser la collecte, le stockage, la gestion et l'analyse des données de performance; afin de construire un profil de la performance du commutateur virtuel, d'une part, et afin de permettre la comparaison des performances des différents commutateurs, d'autre part.

En effet, un profil de performance permettra la supervision de la performance des commutateurs virtuels, ce qui aidera à déterminer les nœuds défaillants dans le réseau d'interconnexion des centres de données, et peut être aidé au diagnostic des anomalies dans ces réseaux.

Une méthodologie sera établie pour répondre aux questions de la problématique. L'outil résultant sera testé dans un cadre spécifique, sur des commutateurs virtuels. Un réseau d'une architecture de centres données, représentera un cas d'utilisation. Le réseau sera ainsi mis en place pour tester l'outil de la supervision de la performance dans un environnement quasiment réel.

Deux bilans seront dressés pour avoir une description de la performance des commutateurs virtuels. Un premier bilan illustrera l'influence du CPU et de la mémoire, du serveur physique, sur la performance des commutateurs virtuels.

Le deuxième bilan aura pour objectif de comparaison de la performance des commutateurs virtuels avec les commutateurs physiques. Le bilan déterminera si les commutateurs virtuels sont assez performants face aux commutateurs conventionnels physiques.

1.4 Plan du mémoire

En plus de l'introduction générale, quatre autres chapitres constituent ce mémoire, organisés comme la suite :

Le chapitre 2 est une revue de littérature des réseaux d'interconnexion des centres de données; de leurs architectures et de la technologie de virtualisation qui caractérise les commutateurs qui composent ces réseaux. La revue de littérature est faite aussi pour les métriques de la performance des réseaux, les outils de l'analyse de cette performance et la supervision de la performance dans le cas des centres de données.

Le chapitre 3, dans une première partie, illustre la construction d'un profil de métriques de la performance, la sélection de l'outil de la collecte des métriques de la performance et le développement d'un modèle pour l'automatisation de la collecte des données. Dans une deuxième partie, ce chapitre présente comment est faite l'analyse de la performance des commutateurs virtuels et des réseaux d'interconnexions de centres de données.

Le chapitre 4 expose l'implémentation de l'environnement d'expérimentations, et les résultats relevés pour la comparaison de la performance des commutateurs virtuels à celle des commutateurs physiques. Aussi, à base des résultats de l'analyse de la performance des commutateurs composants, un bilan de la performance du réseau d'interconnexion du centre de donnée est dressé.

Le chapitre 5 est une synthèse du mémoire, et une perspective d'un possible travail future.

CHAPITRE 2

REVUE DE LITTÉRATURE

2.1 Introduction

Ce chapitre présente la revue de littérature des réseaux d'interconnexion des centres de données et de l'analyse de la performance des réseaux. Dans un premier temps, la virtualisation, la principale technologie caractérisant des centres de données, et les architectures spécifiques des centres de données sont exposées. Dans la seconde section, les notions théoriques nécessaires à la compréhension de la problématique de l'analyse de la performance des réseaux sont introduites. La dernière partie présente ce qui a été déjà fait pour la supervision de la performance des réseaux d'interconnexion de centres de données.

2.2 Les réseaux d'interconnexion des centres de données

2.2.1 La virtualisation

La virtualisation est moyen avec lequel le même traitement peut être fait sur moins de machines physiques, par l'augmentation de leurs utilisation. La virtualisation, initialement développée pour les serveurs, est aujourd'hui présente pour l'ensemble des équipements informatiques. Elle est un composant fondamental dans les réseaux d'interconnexion des centres de données.

2.2.1.1 Définition de concept

Traditionnellement, un système d'exploitation est installé et couplé à une machine physique. Une migration de l'unité de stockage, contenant le système d'exploitation, à une machine physique différente générera un mal fonctionnement dans cette machine. En effet, les pilotes des périphériques sur ce système d'exploitation ne seront pas compatible avec la deuxième machine physique (Eli, 2012).

La virtualisation est la création d'une couche intermédiaire au-dessus de la machine physique, dite hyperviseur. Le système d'exploitation sera installé au-dessus de ce hyperviseur, et ce système d'exploitation sera dit instance de système d'exploitation, ou machine virtuelle. L'instance de système d'exploitation est installée et couplée à l'hyperviseur et non pas à la machine physique. Toutefois, les applications exécutées sur ces instances ont l'illusion d'avoir des machines physiques indépendantes (Eli, 2012). Figure 2.1 illustre la différence entre machine virtuelle et machine traditionnelle.

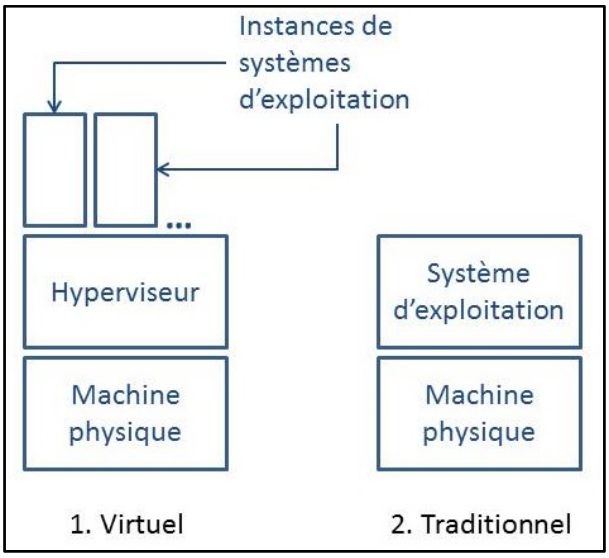


Figure 2.1 Machine virtuelle
et machine traditionnelle

L'importance de la virtualisation se voit dans la migration de système d'exploitation. À la différence des machines traditionnelles, les instances de systèmes d'exploitation n'ont aucune vue de la machine physique (Eli, 2012). Alors, il suffit d'avoir le même type d'hyperviseur sur l'autre machine physique pour migrer l'instance de système d'exploitation, et ce sans avoir des maux fonctionnements (Eli, 2012). En effet, la migration n'est plus que copier un fichier d'une machine à l'autre.

L'autre point fort de la virtualisation est que sur une seule machine physique, il est possible d'installer plusieurs instances de systèmes d'exploitation. Cela se traduit par la consolidation

de plusieurs machines à performances basses, à une seule machine puissante (Eli, 2012). Encore plus, une machine sous-utilisée peut devenir pleinement utilisée en y installant plusieurs instances au-dessus. D'où l'importance économique de la virtualisation.

2.2.1.2 L'hyperviseur

Il y a deux types d'hyperviseurs, nommés type 1 et type 2 (Eli, 2012).

Hyperviseur type 1

Le type 1 est un hyperviseur qui s'installe directement sur la machine physique. Il est dit aussi hyperviseur natif ou hyperviseur métal nu (Eli, 2012). L'hyperviseur type 1 est un système d'exploitation basique, qui n'offre aucune fonctionnalité. Il faut ainsi un agent de gestion, installé sur une machine distante pour gérer et configurer l'hyperviseur, et y installé des instances de systèmes d'exploitation (Eli, 2012). Figure 2.2 illustre le principe.

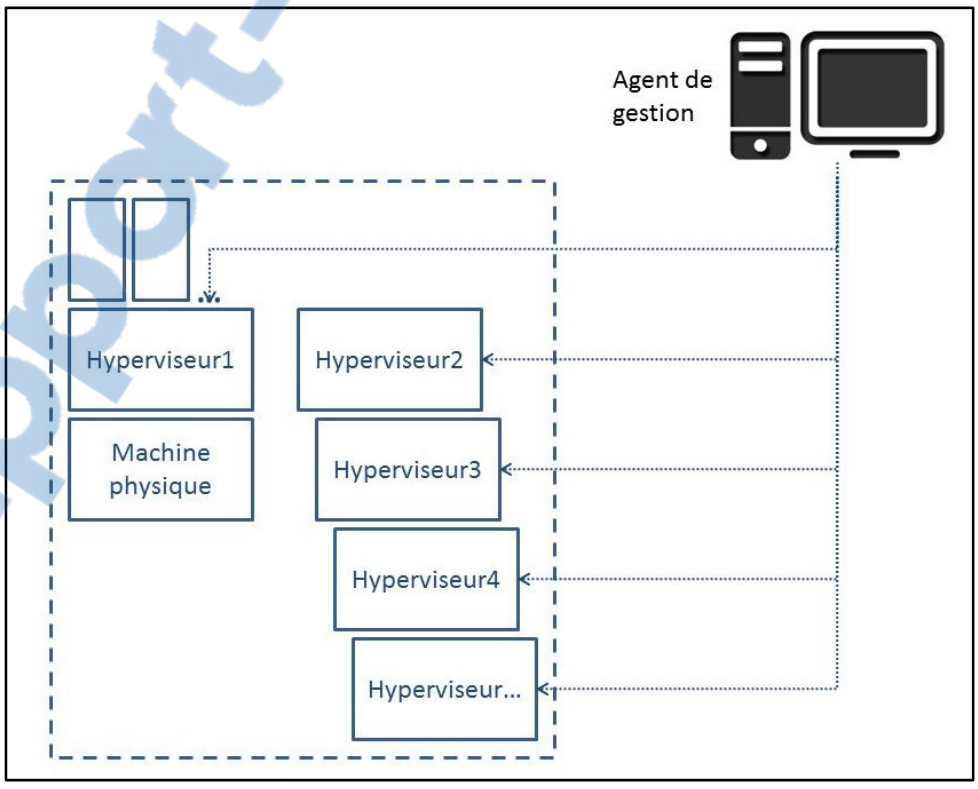


Figure 2.2 Agent de gestion de l'hyperviseur type 1

L'agent de gestion peut gérer les hyperviseurs de plusieurs machines. Ainsi, il est possible de faire migrer une instance entre les différentes machines. L'application pertinente de cette migration est que l'agent peut migrer automatiquement des instances d'une machine à l'autre, en fonction de leurs besoins en ressources. Ainsi, des machines peuvent ne contenir aucune instance, et être éteintes, ce qui représente une grande économie d'énergie (Eli, 2012).

L'agent de gestion peut avoir une plus grande intelligence, en offrant la fonction de l'allocation dynamique de ressources. Le principe de l'allocation dynamique de ressources est d'allouer aux instances des ressources totales supérieures aux ressources disponibles de la machine physique. Les ressources réellement allouées à une instance seront ainsi variantes, selon le besoin de celle-ci (Eli, 2012).
Xen (Xen Project, 2013) est un exemple d'hyperviseurs type 1.

Hyperviseur type 2

Le type 2, ou l'hyperviseur hébergé, est hyperviseur qui s'installe au-dessus d'un système d'exploitation (Eli, 2012). Figure 2.3 illustre les couches dans un système avec un hyperviseur type 2.

À la différence de l'hyperviseur type1, le type 2 n'a pas besoin d'un agent de gestion. En effet l'instance peut s'ouvrir dans une fenêtre, comme une simple application. L'avantage de ce type d'hyperviseur, c'est qu'il est simple à manipuler, mais cela est au prix des grandes fonctionnalités du type 1. Le type 2 est plus adapté à l'usage limité, avec quelques instances.

L'hyperviseur type 2 n'offre pas d'allocation dynamique de ressources. Dès la création de l'instance, l'ensemble des ressources lui seront allouées automatiquement, même s'elles ne sont pas utilisées au complet (Eli, 2012).

VMware (VMware, 2013) et KVM (KVM, 2013) sont des exemples d'hyperviseurs type 2.

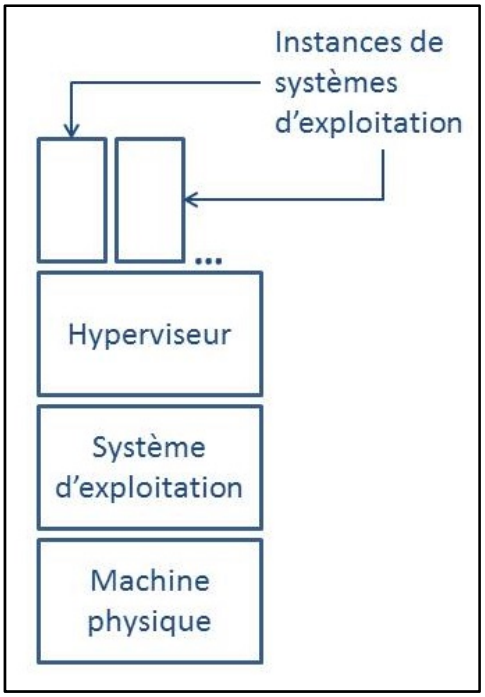


Figure 2.3 Système
avec l'hyperviseur type 2

2.2.1.3 Réseaux virtuels

Le concept de la virtualisation des réseaux est que plusieurs réseaux virtuels partagent un même réseau physique. Le principe est illustré par l'exemple de Figure 2.4, où deux réseaux virtuels, RV1 et RV2, sont au-dessus de l'hyperviseur, qui fournit le mécanisme de partage d'un seul réseau physique.

La virtualisation a été appliquée aux réseaux depuis plus d'une décennie pour virtualiser, à titre d'exemple, les liens, avec des technologies comme le VLAN et le VPN. Cependant, la fonctionnalité d'un réseau repose sur ses nœuds. Ainsi, une approche plus récente est de virtualiser les routeurs et les commutateurs (Fabienne, 2013).

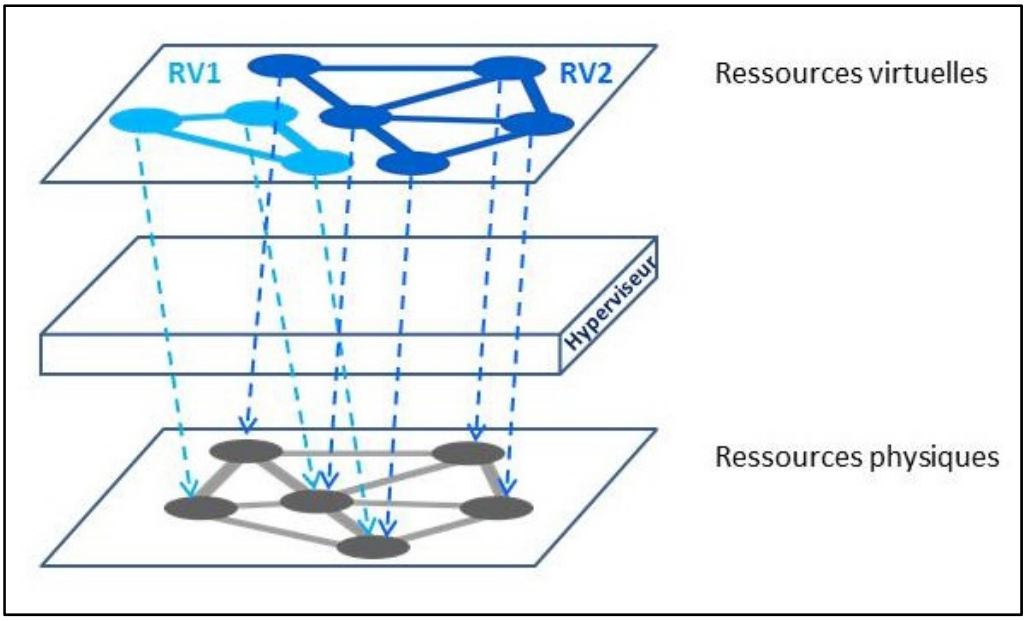


Figure 2.4 Exemple de réseau virtuel

La virtualisation des commutateurs permet le partage les ressources d'un hôte physique entre plusieurs commutateurs virtuels. Alors, pour fournir un accès réseau, la pile réseau de la machine hôte est virtualisée (Fabienne, 2013). Un mécanisme de multiplexage assigne les paquets des interfaces d'un commutateur virtuel à l'interface physique de la machine hôte, et vice versa (Fabienne, 2013). Et un émulateur abstrait les interfaces depuis le matériel, pour que le commutateur virtuel les voir comme des interfaces physiques.

2.2.1.4 Commutateurs virtuels

Un format plus courant des commutateurs virtuels est les modules de pont implémentés dans les distributions Linux, dits Virtual Ethernet Bridge (VEB) (Fabienne, 2013). Ces VEB, qui ne sont que des ponts, ont évolué pour donner des commutateurs virtuels plus complets, implémentant les VLANs et supportant les configurations de paramètres ACL et QoS. Ces commutateurs virtuels évolués font toujours usage des VEB pour créer des connexions.

Les commutateurs virtuels, ainsi que toutes les machines virtuelles, sont souvent connectées à des interfaces réseaux virtuelles, qui partagent la même interface physique. Figure 2.5,

illustre une implémentation typique simple des commutateurs virtuels; où les machines virtuelles sont connectées au commutateur virtuel, et c'est ce dernier qui les interconnecte vers le réseau extérieur.

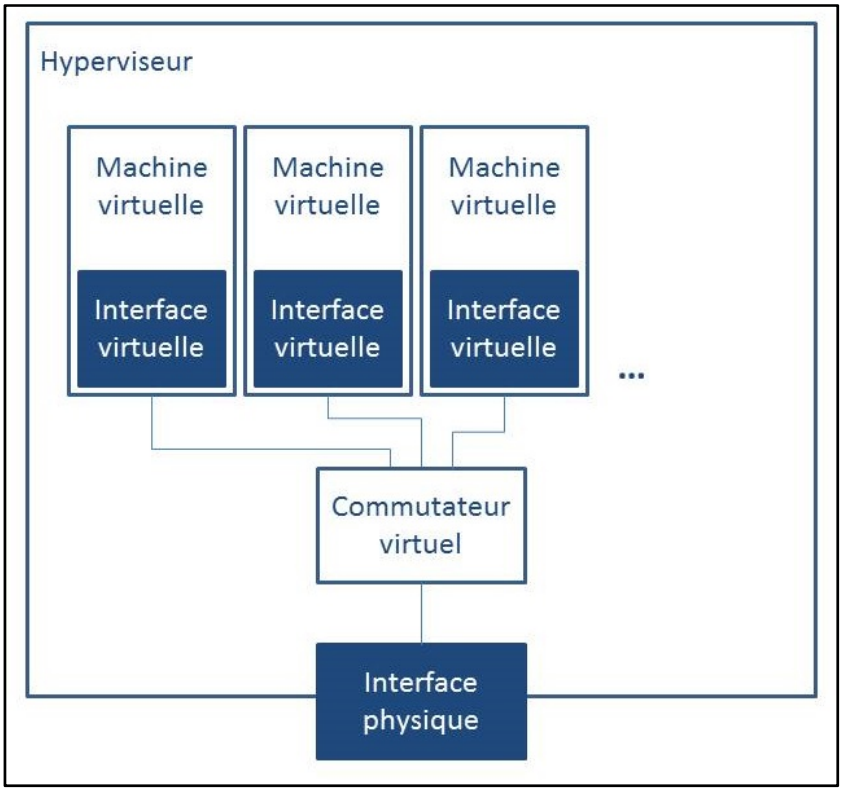


Figure 2.5 Exemple d'architecture d'un serveur
avec un commutateur virtuel

Les commutateurs virtuels peuvent être utilisés pour monter tous les réseaux d'interconnexion des centres de données, interconnectant plusieurs machines virtuelles et physiques.

2.2.2 Les réseaux de centres de données

Les centres de données sont des ensembles de serveurs interconnectés par des commutateurs, des routeurs et des liens à très grande vitesse. Donc, il fallait des réseaux d'interconnexion

des centres de données qui maximisent l'efficacité des traitements des grands flux, tout en minimisant le coût.

Plusieurs architectures ont été conçues spécifiquement pour les centres de données, afin de répondre à leurs besoins spécifiques. Ces architectures spécifiques ont pour but d'améliorer les performances pour l'extensibilité, la migration de machine virtuelle, l'agilité, la tolérance aux pannes et la consommation d'énergie. Le choix d'une solution pour les réseaux d'interconnexion des centres de données pour dresser un défi influence ou limite la résolution des autres défis. D'où l'importance du choix de l'architecture de réseaux des centres de données (Hammadi et Mhamdi, 2014).

2.2.2.1 Architectures conventionnelles

La topologie conventionnelle des architectures des réseaux d'interconnexion de centres de données est la topologie arborescente (Hammadi et Mhamdi, 2014), en Spine/Leaf, comme dans Cisco systems (2007b), Al-Fares, Loukissas et Vahdat (2008) et Greenberg, Hamilton et Jain (2009).

Architecture Multi-tiers

L'architecture repose sur deux ou trois couches hiérarchiques de commutateurs et routeurs, comme le montre Figure 2.6.

La couche cœur sert à une passerelle vers l'extérieur. Il est connecté d'une manière redondante avec le réseau extérieur et le niveau agrégation du centre de données. Pour créer une architecture complètement redondante et éliminer le cas du point de défaillance unique.

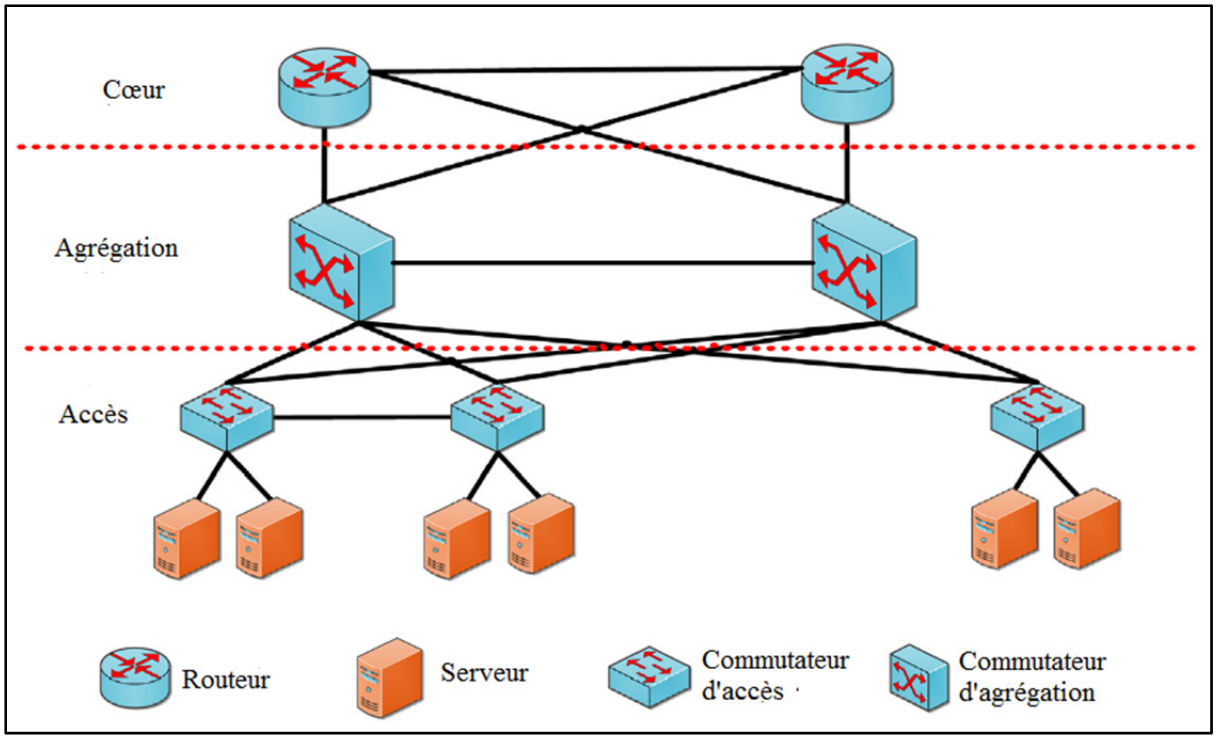


Figure 2.6 Architecture multi-tiers
Tirée de Hammadi et Mhamdi (2014, p. 2)

La couche agrégation, avec plusieurs liaisons uplinks de la couche d'accès relié à lui, a la responsabilité d'agréger les flux entrants et sortant du centre de données. La couche agrégation offre aussi des services à valeur ajoutée, comme le load balancing, le pare-feu et le déchargement SSL.

La couche accès fournit un attachement physique aux ressources serveurs.

L'architecture multi-tiers est principalement utilisée dans les centres de données pour les plateformes web, afin d'assurer la sécurité aux accès malfaisants, en séparant strictement les trois niveaux : serveurs web, application et base de données. C'est la couche agrégation qui transporte du trafic entre serveurs, à travers la couche accès (web à application ou application à base de données) (Cisco systems, 2007b).

Architecture Fat-tree

Malheureusement, même si les commutateurs les plus performants sont utilisés, la topologie résultante ne peut supporter que 50% de la bande passante agrégée, depuis la couche accès. L'architecture Fat-tree permet l'agrégation totale de la bande passante des clusters constitués de milliers de serveurs (Al-Fares, Loukissas et Vahdat, 2008).

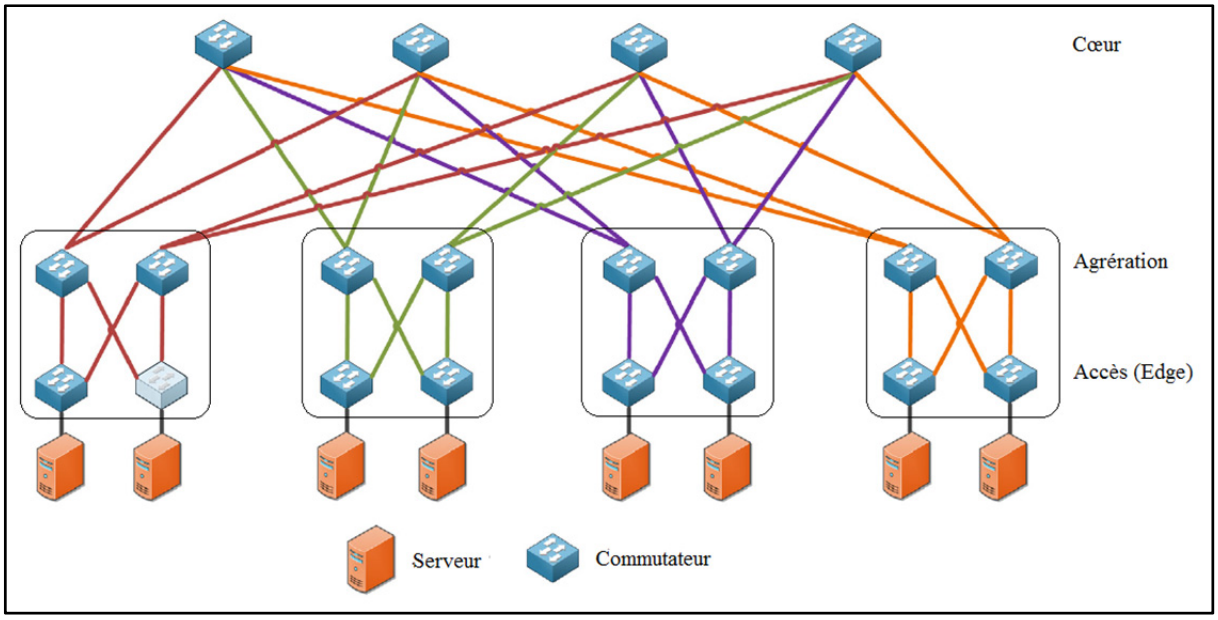


Figure 2.7 Architecture Fat-tree
Tirée de Hammadi et Mhamdi (2014, p. 3)

Fat-tree, illustré dans Figure 2.7, regroupe les commutateurs des couches accès et agrégation dans des pods. Chaque pod est interconnecté à l'ensemble des commutateurs cœur (Hammadi et Mhamdi, 2014).

Architecture VL2

L'architecture VL2 s'attaque à l'agilité et la tolérance aux pannes, supporte la migration des machines virtuelles sans coupure dans la connexion TCP, sans changement de son adresse IP. C'est une architecture similaire à Multi-tiers et Fat-tree, sauf qu'elle offre plusieurs chemins et une connectivité plus riche entre les couches agrégation et cœur (Hammadi et Mhamdi, 2014).

De plus, VL2 implémente le protocole VLB (Valiant Load Balancing) pour distribuer le trafic sur l'ensemble des chemins, sans avoir besoin d'une coordination centralisée. En effet, tout serveur sélectionne arbitrairement un chemin pour chaque flux il envoie à un autre serveur dans le centre de données (Greenberg, Hamilton et Jain, 2009). L'architecture est illustrée dans Figure 2.8.

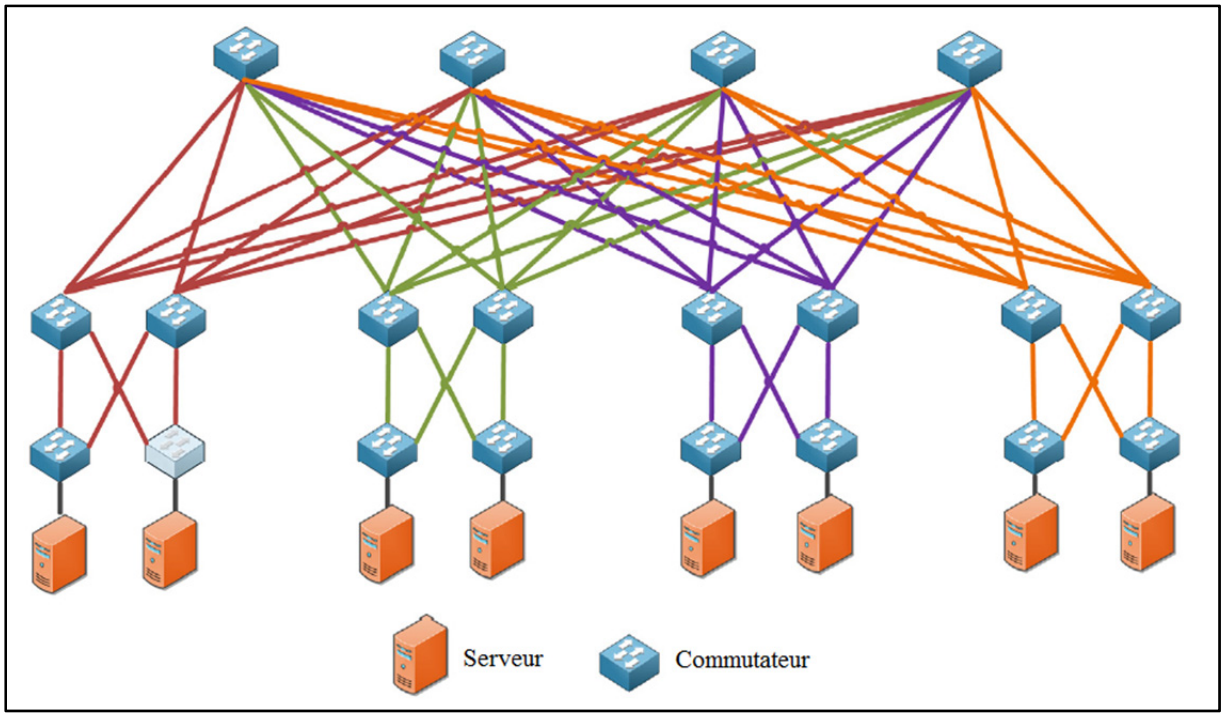


Figure 2.8 Architecture VL2
Tirée de Hammadi et Mhamdi (2014, p. 4)

2.2.2.2 Tendances futures

Les trois topologies exposées sont les architectures arborescentes, en spine/leaf. Ils sont des topologies centrées autour des commutateurs. Des topologies futuristes sont en test, et ils utilisent les serveurs comme relais pour le trafic. Cela réduit le diamètre du centre de données, tout en gardant la même grande capacité et le support des différents types de trafics (Juniper networks, 2013) (Guo, 2008) (Guo, 2009).

2.3 L'analyse de la performance des réseaux

2.3.1 Les métriques de la performance

Les métriques de la performance des réseaux sont des informations décrivant les conditions de transmission de paquets. Le délai (Angrisani et autres, 2006), la gigue (Matta et Takeshita, 2002), le taux de perte de paquets (Boulos, 2009), le débit effectif (Matta et Takeshita, 2002), le taux d'utilisation de lien (Papagiannaki et autres, 2003b) et le taux de consommation des ressources physiques (Papagiannaki, Cruz et Diot, 2003a), sont les métriques de la performance des réseaux les plus utilisées.

Le délai de paquet est le composant le plus significatif du délai de bout-en-bout du paquet, qui est le plus important paramètre de la qualité de service. Le délai de paquet est le temps pour les paquets à traverser un des éléments de transfert dans leurs chemins. Il consiste en trois composants : le délai de transmission, le délai de traitement et le délai de la file d'attente.

Le délai de transmission est le temps nécessaire, pour le commutateur, pour placer le paquet sur le lien de sortie. Le délai de traitement est le temps nécessaire pour déterminer le port de sortie, et lui transférer le paquet. Le délai de la file d'attente est le temps passé par le paquet à l'entrée et/ou la sortie du commutateur.

Alors que le délai de transmission est relié à la capacité des liens et la taille des paquets, le délai de traitement et le délai de la file d'attente décrivent quantitativement la performance du commutateur (Angrisani et autres, 2006).

Le délai de la file d'attente est un paramètre dynamique qui est proportionnel au nombre de paquets qui sont déjà en attente à la file d'attente de sortie à un instant donné (Salehin et Rojas-Cessa, 2014). Autrement dit, la mesure de délai de la file d'attente du commutateur est clé pour déterminer le niveau de congestion dans le réseau.

Un calcul précis du délai de la file d'attente permet à un fournisseur de service de terminer et maintenir, respectivement, la qualité de service et le Service Level Agreement, pour les applications sensibles au délai (Salehin et Rojas-Cessa, 2014). La variation de la qualité de service est directement liée la variation du délai de la file d'attente.

Le délai de traitement de paquets est le temps entre l'arrivée du paquet à la file d'attente d'entrée de l'interface réseau (niveau liaison de données), et le temps de traitement de paquet au niveau application (Salehin et Rojas-Cessa, 2013). C'est le seul délai, lié à la performance du commutateur, qui est toujours présent, puisque le délai de la file d'attente n'existe qu'au cas de congestion.

La gigue est la variation en délai de transmission de bout en bout des paquets d'un même flux. Cela se traduit par le décalage temporel entre les paquets à la réception. La gigue est une métrique importante pour les applications sensibles à la latence, comme la voix sur IP.

Le taux de perte de paquets est la métrique décrivant la fiabilité du réseau. Il est souvent utilisé pour superviser la qualité de service. C'est une métrique clé pour les services de temps réel. Si un faible taux de perte de paquets est tolérable pour les services de la voix sur IP, la perte 1.1% de paquets en IPTV génère un taux de mal satisfaction de 80% des clients du service (Boulos, 2009).

Le débit effectif est le nombre de bits par unité de temps qu'une source peut émettre sur le réseau. Cela varie en fonction du temps passé par le paquet dans la file d'attente. Ce qui fait de la bande passante disponible et le délai de la file d'attente deux faces d'une même métrique.

Le taux d'utilisation de lien est la capacité du lien consommée par le flux émis par la source. C'est commun de collecter le taux d'utilisation de lien pour en déduire le délai de paquet. En effet, un seuil d'utilisation de lien est établi, souvent de 50%. Si un lien est utilisé à moins de 50%, alors, il est capable de transporter n'importe quel flux de la même capacité de lien, dans

les autres liens voisins, en cas de panne (Papagiannaki, Cruz et Diot, 2003a). Aussi, il avait été prouvé dans une étude, que les liens utilisés à moins de 50%, introduisent un délai de la file d'attente minimum (Papagiannaki, Cruz et Diot, 2003a).

La consommation du CPU représente la limitation physique du réseau. Elle est nécessaire pour le dimensionnement et la conception de réseaux. Une consommation élevée du CPU peut produire des lenteurs dans la performance et échec de réponse de certains services.

2.3.2 Les outils de l'analyse de la performance

Les logiciels de supervision de réseaux sont les outils les plus répondus pour la supervision de la performance des réseaux et de leurs nœuds composants. Cela a pour cause de la simplicité de l'utilisation de ces logiciels, qui représente leurs données sur des interfaces web, des graphes ou des bases de données (Mongkolluksamee, Pongpaibool et Issariyapat, 2010).

Il existe une vaste gamme de logiciels de supervision de la performance. Mais, seuls cinq logiciels sont clairement distingués en termes d'utilisation, selon Sourceforge et Google Trends (Mongkolluksamee, Pongpaibool et Issariyapat, 2010). Ils sont Nagios (Nagios, 2014), Zenoss (Zenoss, 2015), OpenNMS (OpenNMS, 2015a), GroundWork (GroundWork, 2015) et Hyperic (Hyperic, 2015). Les trois premiers sont destinés à toutes sortes de réseaux. Tant qu'aux deux derniers sont destinés, de plus des réseaux traditionnels, aux machines virtuelles.

Nagios

Nagios (Nagios, 2014) est le plus difficile à configurer et le plus désagréable dans la présentation de données. Cependant, un intérêt considérable est donné à Nagios par rapport aux autres logiciels. En effet, les téléchargements de Nagios représentent 50% des

téléchargements des cinq logiciels présentés dans ce chapitre (Mongkolluksamee, Pongpaibool et Issariyapat, 2010).

Nagios supervise l'état de fonctionnement de l'infrastructure informatique et les services incluant, entre autres, HTTP, POP, SSH et SNMP. Il offre une vue centralisée de tout le réseau, grâce à son interface web.

Nagios est extensible pour des réseaux aussi larges de 100,000 hôtes et 1,000,000 services (Mongkolluksamee, Pongpaibool et Issariyapat, 2010). De plus, il est très flexible et configurable. En effet, le cœur de Nagios, Nagios Core, effectue des sondages auprès des équipements du réseau et de leurs services. Et ce sont des plugins extérieurs qui récupèrent les données de ces sondages pour vérifier et calculer les métriques supervisées.

Figure 2.9 illustre le fonctionnement de Nagios. Tout d'abord, Nagios Core est l'ordonnanceur d'événement dans ce système. Il ordonne périodiquement l'exécution des plugins extérieurs, pour connaître l'état des nœuds supervisés. Une fois il a le résultat, Nagios Core l'affiche sur l'interface web (Nagios, 2013).

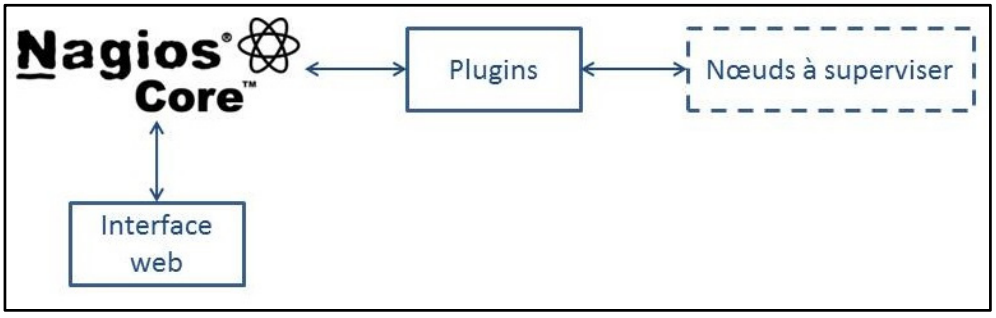


Figure 2.9 Fonctionnement de Nagios

Tout utilisateur peut développer son propre plugin pour supporter tout équipement ou service particulier. Comme exemples de modules développés par la communauté de Nagios, il est à noter :

- NDOUtils (Nagios, 2015d), pour l'enregistrement des données Nagios;
- NagiosQL (NagiosQL, 2015), offrant des Gabarits simplifiant la configuration;
- Nmap2Nagios (Nagios, 2015e), pour la découverte automatique d'équipement dans le réseau; etc…

Zenoss

Zenoss (Zenoss, 2015) offre la visibilité du réseau, depuis l'infrastructure physique, jusqu'au niveau application. Il offre une interface web avec les disponibilités dans le réseau et des graphes de la performance. À la différence de Nagios, Zenoss offre la découverte automatique. Et c'est une fonctionnalité qui met Zenoss à l'écart des autres outils.

Figure 2.10 illustre le fonctionnement de Zenoss. Zenoss est composé de quatre couches. La couche collection collecte les données depuis les nœuds supervisés et les transmette à la couche données. La couche traitement gère les communications entres les couches collection et données, traite les commandes de l'utilisateur et exécute la collecte de données périodiquement. La couche données stocke les données collectées, anisi que la configuration. La couche utilisateur offre un portail web (Zenoss, 2015).

Mais, la fiabilité de Zenoss est mise en question, puisqu'il affiche des données erronées durant la période de stabilisation (Modern Course, 2015). Toutefois, l'inconvénient majeur de Zenoss est qu'il disperse les données de supervision sur une base de données MySQL, une base de données interne Zope et des fichiers RRD sur le disque.

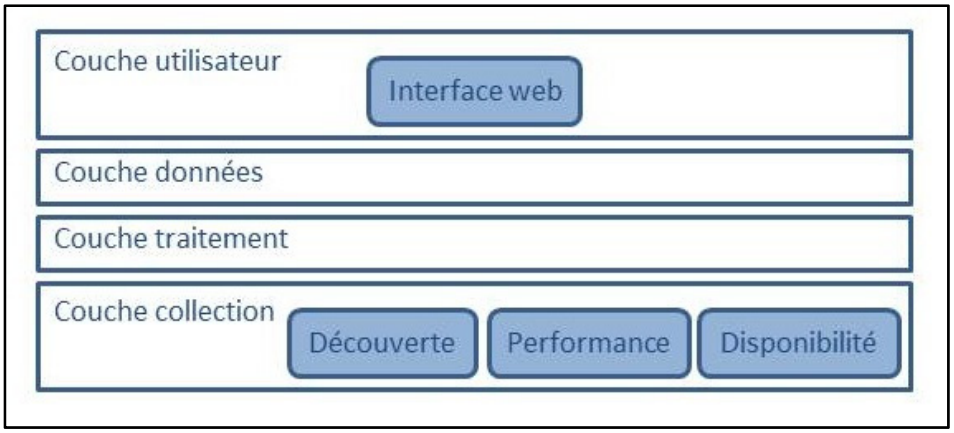


Figure 2.10 Fonctionnement de Zenoss

OpenNMS

Comme Nagios et Zenoss, OpenNMS (OpenNMS, 2015a) collecte les données sur la disponibilité et la performance. Mais à la différence de ces deux concurrents, OpenNMS sépare cette collecte en deux démons : un pour la disponibilité et l'autre pour la performance. Aussi, OpenNMS ne collecte pas les données seulement avec SNMP, SSH et ICMP, mais aussi avec HTTP(s), JMX, WMI et autres protocoles (OpenNMS, 2015b).

Figure 2.11 illustre le fonctionnement de OpenNMS. OpenNMS fait la découverte des équipements dans le réseau, vérifie leurs disponibilités et collecte les données de la performance. Les résultats sont enregistrés dans une base de données, représentés graphiquement et affichés sur une interface web. Cette interface web sert à administrer et superviser le réseau.

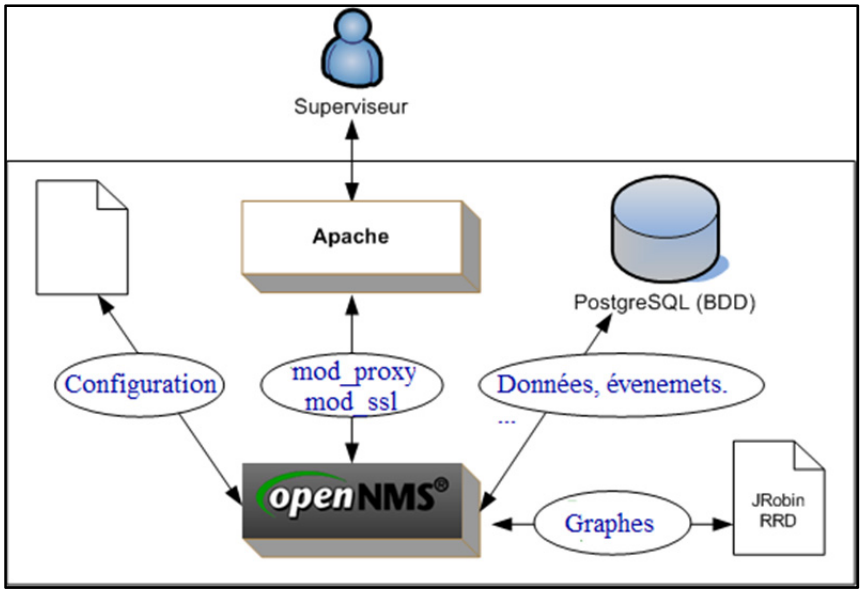


Figure 2.11 Fonctionnement OpenNMS
Tirée de Wiki monitoring (2013)

Le seul inconvénient de OpenNMS est qu'il est compliqué et intuitif à configurer. Plusieurs fichiers de configurations doivent être modifiés, à la différence de Nagios qui en demande la modification d'un couple de fichiers seulement. La clé de la réussite avec une supervision à base de OpenNMS est de s'investir en temps.

GroundWork

Groundwork (GroundWork, 2015) est plus qu'un simple outil de supervision de réseaux. Il ne s'agit pas d'un outil indépendant, mais plutôt, d'une base de fondement, alimentée par des outils déjà existants. Figure 2.12 illustre l'architecture de GroundWork.

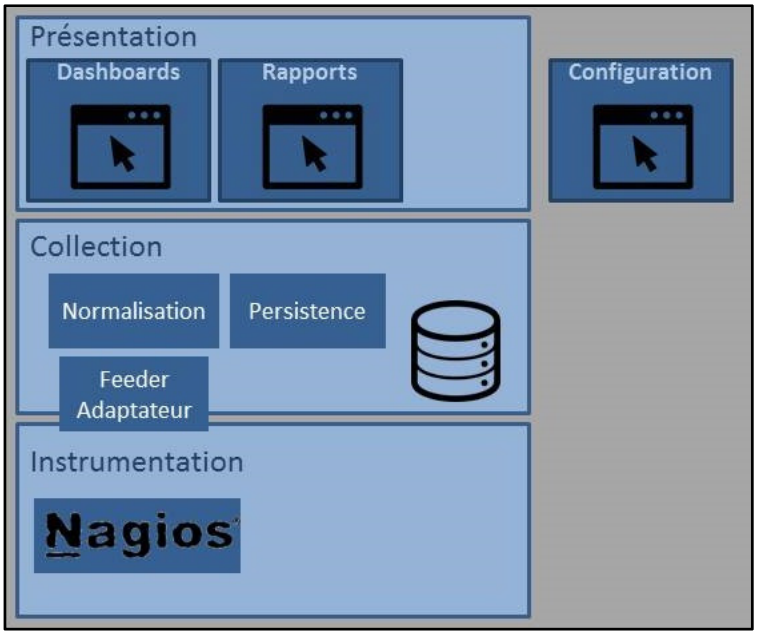


Figure 2.12 Architecture GroundWork
Adaptée de GroundWork (2007)

GroundWork est constitué de trois modules :

- instrumentation, qui cherche les données depuis les équipements supervisés;
- collection, qui collecte les données, les normalise et les enregistre dans une base de données;
- présentation, qui affiche les données de la supervision;
- la configuration se fait par un module externe.

Chacun des modules peut contenir des outils existants. À titre d'exemple, GroundWork profite de l'extensibilité de Nagios, pour s'en servir et collecter les données grâce à lui.

Ce qui rend GroundWork spécial, est qu'il offre la découverte automatique des machines virtuelles, des plateformes KVM et VMware vSphere. GroundWork supervise, ainsi, la disponibilité des machines virtuelles, leurs systèmes d'exploitation et les applications hébergées dans les machines virtuelles (GroundWork, 2015).

2.3.3 Analyse de la performance dans les réseaux de centres de données

Alors que les applications et les services sont en massive migration vers les centres de données, les administrateurs réseaux sont mis au défi de superviser leurs réseaux de nœuds virtuels avec les outils existants. Cela a relevé une éventuelle nécessité aux nouveaux outils.

Les solutions développées pour la supervision des réseaux d'interconnexion des centres de données, s'intéressent plutôt sur la disponibilité des ressources physiques, en relevant les utilisations de CPU, du stockage, de la mémoire et la bande passante des interfaces réseaux (GroundWork, 2015) (Hyperic, 2015). Une vue limitée du trafic est offerte par les outils d'aujourd'hui (Clayman, Galis et Mamatas, 2010).

Pourtant, plusieurs études ont relevé l'importance de la supervision de la performance des réseaux virtuels. Ciuffoletti (2010) fait le point sur l'hétérogénéité de l'infrastructure dans les centres de données, et argumente qu'une hétérogénéité de la performance du réseau rend le load balancing ineffective, le streaming avec des variations de délais et le calcul parallèle impossible.

Le travail de Fabienne (2013) traite l'influence de l'hyperviseur sur le débit effectif des commutateurs virtuels. Les expérimentations ont été faites sur des commutateurs virtuels à base de Linux. L'hyperviseur de la couche au-dessous changeait durant l'expérimentation. Figure 2.13 montre la couche hyperviseur, sous expérimentation. Le débit effectif du commutateur virtuel, dans chaque cas d'hyperviseurs expérimentés, a été comparé avec celui d'un commutateur Linux natif.

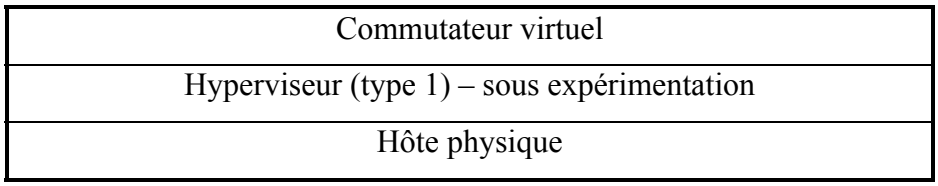


Figure 2.13 Architecture de l'expérimentation sur l'impact de l'hyperviseur sur la performance

Les résultats montrent que l'hyperviseur a une influence sur la performance des commutateurs virtuels. Le débit effectif ne peut pas dépasser les 25% du débit effectif d'un commutateur natif. Cette efficacité de 25% peut même ne pas être atteinte dans le cas de trafique à petits paquets (Fabienne, 2013).

Dans Martinovic, Balen et Rimac-Drlje (2013), l'impact du système d'exploitation hébergeur est étudié. Un serveur, d'un pool de serveurs identiques, a été déployé pour chaque système d'exploitation hébergeur différent. Sur ces serveurs, des machines virtuelles identiques, ainsi que le même hyperviseur ont été utilisées. Figure 2.14 montre la couche système d'exploitation hébergeur, sous expérimentation.

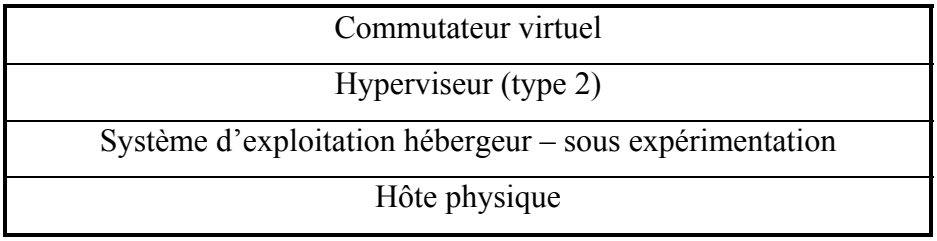


Figure 2.14 Architecture de l'expérimentation sur l'impact du système d'exploitation hébergeur sur la performance

Les résultats montrent que le choix de système d'exploitation hébergeur affecte la performance du commutateur virtuel, en termes de performance graphique, l'utilisation de la mémoire physique et de l'espace disque de stockage. Cependant, le choix de système d'exploitation hébergeur n'a pas d'impact sur l'utilisation de CPU (Martinovic, Balen et Rimac-Drlje, 2013).

Alors que Fabienne (2013) traite l'impact de l'hyperviseur, et Martinovic, Balen et Rimac-Drlje (2013) traite l'impact du système d'exploitation hébergeur; Sundling (2010) traite l'impact de la couche physique sur la performance des machines virtuelles. Dans cette expérimentation, les ressources physiques des machines virtuelles ont été variées et la réponse des applications, sur ces machines virtuelles, a été observée.

L'expérimentation montre qu'au changement de nombre de CPU alloués à une machine virtuelle, le comportement des applications, l'exécution parallèle et la charge de CPU peuvent changer. Cela relève un impact du CPU sur la performance des applications exécutées sur les machines virtuelles (Sundling, 2010).

En termes de mémoire, l'expérimentation à relever une dégradation de performance dans le cas d'affectation d'un espace mémoire trop petit, par exemple de 4G, pour une machine virtuelle hébergeant une application demandant un espace mémoire plus grand, par exemple 16G. Cette dégradation au swapping intensif pour cause de manque de mémoire (Sundling, 2010).

Cette expérimentation met le point sur l'option de limite de mémoire. Cette option, s'elle est activée, limite l'allocation de mémoire, par l'hyperviseur, à une valeur autre que celle réellement assignée (Sundling, 2010). Comme explication, si une mémoire de 2G est allouée à une machine virtuelle, et il y a une limite de 512M, le système d'exploitation verra une mémoire physique de 2G, mais l'hyperviseur ne va permettre l'accès qu'à 512M (Sundling, 2010). Si une application a besoin de plus de 512M, il y aura une dégradation de la performance, comme expliquée précédemment.

2.4 Conclusion

La technologie de virtualisation est la technologie clé, déployée dans les réseaux d'interconnexion de centres de données. Les nœuds, les liens et les interfaces des réseaux ont adopté cette technologie. Les architectures des réseaux d'interconnexion des centres de données sont adaptées aux besoins des centres de données, en extensibilité, migration des machines virtuelles, la tolérance aux pannes et la consommation d'énergie.

Pour superviser la performance des réseaux, plusieurs métriques sont utilisées, et une multitude d'outils logiciels offre cette supervision. Mais quand il s'agit de la supervision des

réseaux virtuels, utilisés pour l'interconnexion dans les centres de données, les outils logiciels n'offre rien plus que la vérification de la disponibilité.

La difficulté de la supervision des réseaux virtuels n'a pas empêché l'avancement des travaux sur la performance. Parmi ces travaux, certains ont atteint des conclusions que la performance des commutateurs virtuels est déterminée par l'hyperviseur et le système d'exploitation hébergeur. Aussi, la performance des applications sur les machines virtuelles est déterminée par le matériel physique au-dessous.

CHAPITRE 3

MÉTHODOLOGIE DE RECHERCHE

3.1 Introduction

À la lumière des problématiques décrites dans les sections 1.2, dans la suite, les métriques décrivant la performance des commutateurs seront sélectionnées. Puis un outil pour automatiser la collecte de données sera développé, à base d'un des outils déjà existants. Cette solution permettra l'analyse de la performance des commutateurs virtuels, et de comparer les performances des différents commutateurs.

3.2 Collecte de données

3.2.1 Les métriques à superviser

Des travaux de recherche sur la performance des réseaux ont atteint des conclusions. Parmi les métriques de la performance présentées dans la section 2.3.1, la métrique de la performance des réseaux la plus importante est le débit effectif, principalement affecté par le taux de perte de paquets. Pour les applications à temps réel, le délai de bout-en-bout et la variation de délai s'ajoute comme métriques pertinentes (Chen et Hu, 2002).

D'une manière plus pointue, parmi toutes ces métriques, seul le délai de paquet, un constituant du délai de bout-en-bout, décrit quantitativement la performance des commutateurs (Angrisani et autres, 2006).

Le délai de paquet est le temps passé par un commutateur pour la vérification des erreurs, la lecture de l'entête, la recherche d'interface de sortie sur la table de transmission, la décrémentation du TTL, la fragmentation de paquets, le placement du paquet dans la file d'attente et la transmission vers le lien externe. Figure 3.1 illustre le détail du délai de paquet.

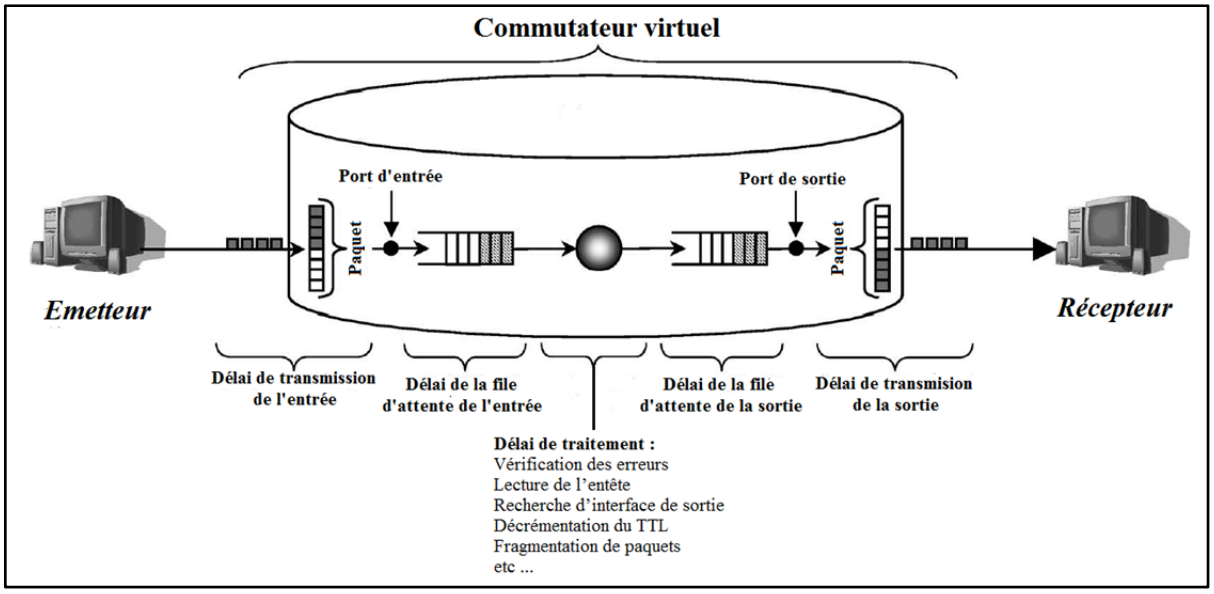


Figure 3.1 Délai de paquet détaillé
Adaptée de Angrisani et als (2006, p. 2)

Un tel traitement de paquets engendre une série de processus exécutés par le CPU. Pour un commutateur physique, un circuit intégré pour des fins de traitement de paquets exécute les processus dans le CPU. Pour un commutateur virtuel, le traitement de paquets est fait par un programme logiciel, exécuté comme tous logiciels dans un ordinateur (Zhu, Deng et Chen, 2011).

Dans les deux cas, le circuit intégré pour le commutateur physique et le programme logiciel pour le commutateur virtuel, des threads pour l'exécution des processus sont déployés. Entre autres, l'organisation des threads, les périodes d'exécution des threads et l'exécution parallèle des threads déterminent le délai de paquet.

De plus des caractéristiques d'exécution de processus, le délai de paquet décrit aussi le temps passé par un paquet dans la file d'attente du commutateur, en attendant la transmission d'un autre paquet.

Le temps d'attente dans une file d'attente est le plus influençant sur le délai de paquet, et il dépend du type de la file d'attente. Cette dernière est une des caractéristiques du commutateur. La file d'attente unique, la file d'attente multiple et la file d'attente nulle sont les trois types de files d'attente les plus communs (Gu, 2008).

La file d'attente unique correspond à un système d'une seule file d'attente avec un ou plusieurs liens de sortie. Dans ce système, un paquet sortant doit attendre l'ensemble des paquets sortants du commutateur, n'importe quelle soit leurs interfaces de sortie.

La file d'attente multiple correspond à un système de plusieurs files d'attente, dont chacune des files d'attente est associée un des liens de sortie. Dans ce système, un paquet sortant attend seulement les paquets sortant vers le même lien.

La taille de la file d'attente unique est toujours supérieure à celle de la file d'attente dans le système à file d'attente multiple, dans le même environnement. Ainsi, les commutateurs à file d'attente multiple sont dits des commutateurs à grande vitesse.
La file d'attente nulle correspond à un système sans file d'attente. Dans ce système, si un paquet arrive au commutateur, et tous les liens sont occupés, alors ce paquet sera écarté. Les commutateurs à ce type de file d'attente sont dédiés aux services à temps réel.

Le délai de paquet décrit aussi le débit des interfaces du commutateur. Ce débit détermine le temps nécessaire pour le commutateur pour lire un paquet depuis un lien et pour émettre un paquet dans un lien.

Ainsi, vu que toutes les caractéristiques d'un commutateur, dont l'architecture du CPU, la conception des files d'attente et le débit des interfaces, sont décrites par le délai de paquets; nous concluant que le délai de paquet est la métrique la plus appropriée pour faire l'analyse de la performance des commutateurs.

3.2.2 L'outil de la collecte de données

L'ensemble des logiciels de supervision de la performance présentés dans la section 2.3.2 supporte le SNMP et offre des calculs de métriques de performance. Le tableau 3.1 compare les différents outils présentés précédemment.

Alors que Zenoss, OpenNMS et GroundWork ont des inconvénients en relation avec des aspects divers, Nagios surmonte ces inconvénients grâce à son extensibilité. Cette dernière permet d'ajouter des fonctionnalités grâce à des plugins. Ainsi, la découverte automatique du réseau, non incluse initialement dans Nagios, est offerte par l'intermédiaire d'un plugin.

L'autre point fort de Nagios est relativement très clair à configurer, contrairement à OpenNMS dont la configuration est compliqué, avec la multitude de ficher de configuration, et Hyperic dont la configuration est plus longue. Une comparaison de ces outils est présentée dans Tableau 3.1.

Tableau 3.1 Comparaison des outils de la supervision de la performance

Logiciel	**SNMP**	**Calcul de performance**	**Auto-découverte**	**Virtualisation**	**Remarques**
Nagios	Oui	Oui	Via plugin	Non	Extensible grâce aux plugins
Zenoss	Oui	Oui	Oui	Non	Pas d'enregistrement dans une base de données centralisée
OpenNMS	Oui	Oui	Oui	Non	Plusieurs fichiers de configurations
GroundWork	Oui	Oui	Oui	Oui	Dépendants d'autres outils

Vu ces avantages, et le fait qu'il offre tout le nécessaire pour la supervision de disponibilité et la performance des nœuds du réseau; Nagios s'est vu approprier la tête des outils de la supervision de la performance, en nombre d'utilisateurs. Ainsi, une communauté très active supporte Nagios Aujourd'hui.

Alors que Nagios offre tout le nécessaire pour ce projet, il n'y a pas de besoins à chercher un autre outil. De plus, sa configuration relativement très claire à faire, et sa communauté très active ont influencé le choix de Nagios comme outil de collecte de donnée dans ce projet.

Dans ce projet, on aura besoin de collecter des données comme la bande passante, le MTU, l'état de lien, etc. des données qui sont collectées avec SNMP, et qui sont tous supporter par Nagios. De plus, la communauté Nagios a développé des plugins pour faire des calculs de métriques de la performance, à base des données de SNMP, comme, le plugin MRTG pour le calcul du débit effectif du lien.

SNMP (Nagios, 2014) est le standard le plus utilisé pour passivement superviser le trafic, afin de déduire la performance de réseau. Il consiste en quatre éléments :

- des agents de management ou des probes installés sur les différents éléments du réseau;
- les MIB contenants les données collectées par les agents;
- une station de management ou console pour rassembler l'information depuis les probes;
- un protocole pour échanger l'information entre les stations et les probes (Matta et Takeshita, 2002).

Figure 3.2 résume les éléments du standard SNMP.

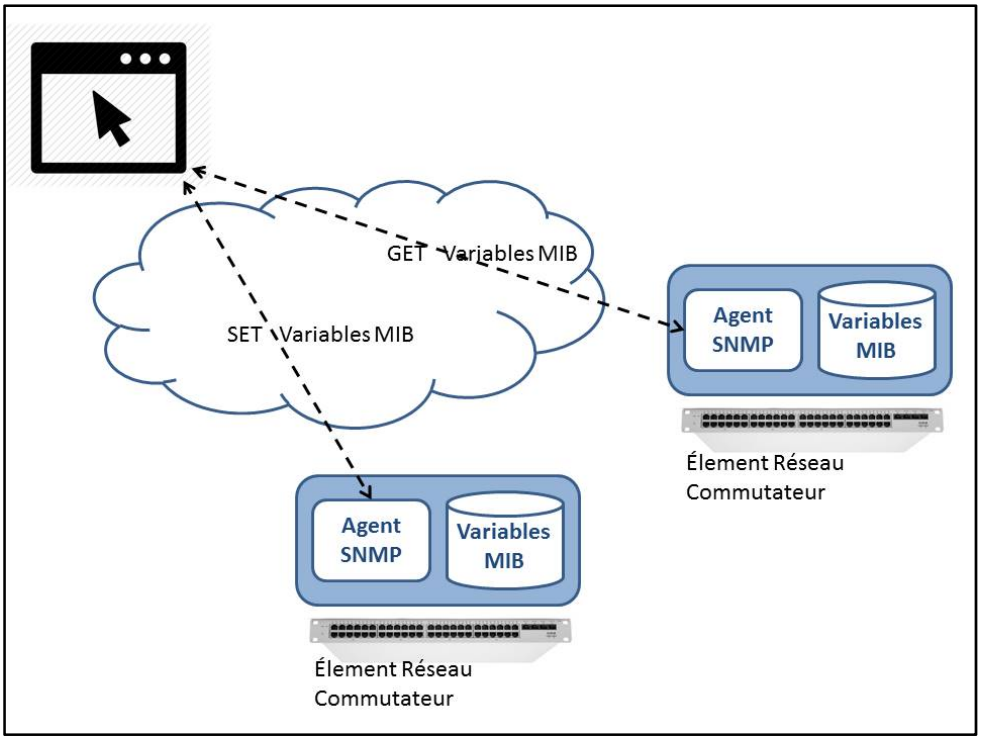


Figure 3.2 Standard SNMP

Actuellement, l'outil SNMP déjà existant se concentre sur le relevé de statistiques sur l'utilisation des ressources réseau. À base de ces dernières, c'est possible d'avoir une information sur la durée et l'amplitude des épisodes de congestion.

Les statistiques de SNMP sont souvent collectées périodiquement dans des intervalles de cinq minutes, et correspondent à l'activité moyenne des liens et des nœuds réseau durant cet intervalle. La raison d'établir un assez large intervalle est la taille du réseau. L'intervalle doit être aussi large pour permettre la collecte des données de tout le réseau.

Pour limiter le trafic SNMP dans le réseau, les opérateurs souvent sélectionnent un nombre limité de variables MIB à superviser (Papagiannaki, Cruz et Diot, 2003a). Dans ce projet trois variables MIB sont supervisées :

- l'état du lien;
- la taille maximale de paquets en octets, MTU.

Afin de configurer la collecte de ces deux objets SNMP sur Nagios, on aura besoin des Object ID (OID). Tableau 3.2 donne les OID des trois variables MIB.

Tableau 3.2 Les Objects ID des variables MIB

Variable MIB	**OID**	**Description**
ifLinkStatus	.1.3.6.1.2.1.2.2.1.10	l'état du lien, up ou down
ifMTU	.1.3.6.1.2.1.2.2.1.4	Taille du plus grand paquet, transitant par l'interface en octets

La configuration de la collecte de ces données dans Nagios est détaillée dans l'annexe IV.

À travers le plugin MRTG, le débit effectif du lien en octets/secondes, T, est relevé.

À base de deux des données relevées, il est possible de déduire le temps de latence. En effet, le temps de latence est calculé grâce à l'équation suivante :

$$\tau = \frac{\mathrm{MTU}}{\mathrm{T}} \tag{3.1}$$

Où τ est le temps de latence, T le débit effectif du lien et MTU la taille maximale de paquet.

À partir du temps de latence, il est possible de calculer le délai de paquet. Le délai de paquet est le temps passé par un paquet dans un commutateur. En effet, le délai de paquet est la différence entre le délai de bout en bout et le temps de latence.

Soit t_0 le temps de latence entre deux hôtes, VM1 et VM2, comme illustré dans Figure 3.3.

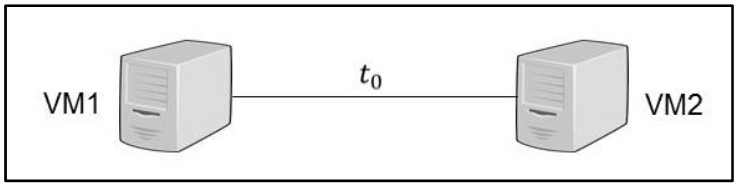


Figure 3.3 Temps de latence

Plaçant un commutateur SW entre les deux hôtes, comme illustré dans Figure 3.4.

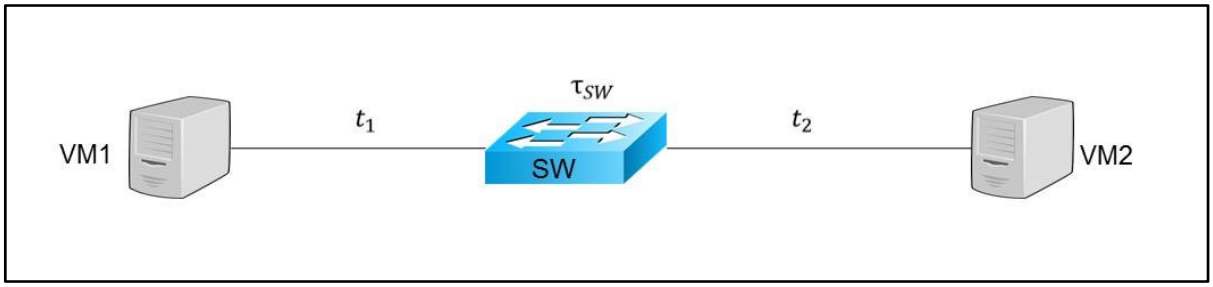


Figure 3.4 Délai de paquet

Soient t_1 le temps de latence entre VM1 et SW, t_2 le temps de latence entre VM2 et SW, et d_{Packet} le délai de paquet dans le commutateur SW. Le délai de paquet est calculé avec la formule suivante :

$$d_{Packet} = t_0 - (t_1 + t_2) \tag{3.2}$$

3.2.3 Automatisation de la collecte des données

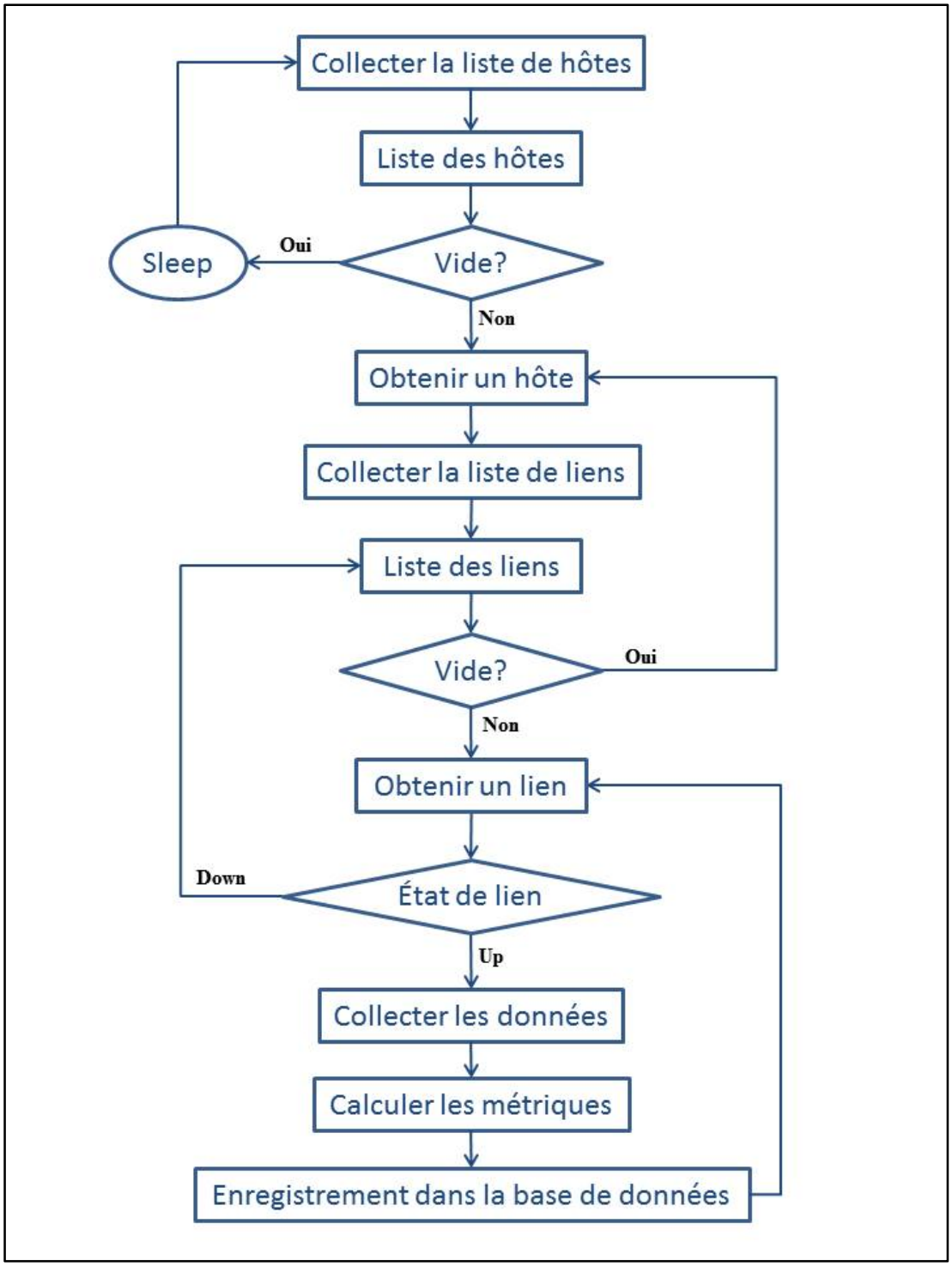
3.2.3.1 Algorithme de l'automatisation de la collecte des données

Nagios enregistre, d'une manière permanente et périodique, les données qu'il collecte, depuis les MIB du SNMP, dans une base de données. La base de données de Nagios contient, entre autres, les identifiants des hôtes supervisés, les variables MIB en fonction de lien ou de numéro de port, avec leurs valeurs mesurées. L'annexe VI détaille le processus d'enregistrement des données de Nagios dans une base de données.

Ces données de Nagios sont les données brutes à base desquelles les métriques de performance, présentées dans les sections 3.2.1 et 3.2.2, seront calculées. Le calcul de ces métriques est présenté dans la section 3.2.2. Le calcul des métriques de la performance doit être fait d'une manière automatique et périodique, corrélée avec la collecte des données brute.

Alors, des modules logiciels sont nécessaires pour automatiser la collecte des données et la supervision de la performance. La mission de ce module est de collecter les données depuis la base de données de Nagios, et faire les traitements nécessaires sur ces données afin d'en tirer les métriques de la performance, et ce d'une manière automatique. Les métriques calculées seront par la suite enregistrées dans une base de données.

Figure 3.5 détail l'algorithme de l'automatisation de la collecte des données et de supervision de la performance.



Algorithme 3.1 Automatisation de la collecte de données

1- collecter les identifiants des hôtes du réseau, depuis la base de données de Nagios;
2- mettre les identifiants d'hôtes dans une liste locale;
3- parcourir la liste des identifiants des hôtes tant qu'elle n'est pas vide. S'elle est vide, le processus est terminé, et le programme va en veille à l'attente de la prochaine période;
4- sinon, obtenir l'hôte suivant dans la liste;
5- un deuxième contact avec la base de données de Nagios est fait, pour obtenir la liste des liens de cet hôte;
6- mettre les identifiants de liens de cet hôte dans une liste locale;
7- parcourir la liste des identifiants de liens tant qu'elle n'est pas vide. S'elle est vide, le processus repend le parcours de la liste des hôtes, pour obtenir l'hôte suivant;
8- sinon, obtenir le lien suivant de la liste;
9- vérifier l'état du lien à partir des données précédemment collectées, s'il est up ou down. L'état de lien est obtenu avec la variable MIB UpLinkStatus, présentée dans la section 3.2.2;
10- si le lien est down, reprendre le parcours de la liste des liens;
11- sinon, faire la collecte des données de la supervision de Nagios. Cette dernière est présentée dans l'annexe VI;
12- à base des données de Nagios, calculer les métriques de la performance des hôtes, soit les délais de paquet, comme présenté dans les sections 3.2.1 et 3.2.2;
13- enregistrer les métriques calculées dans une base de données locale, comme présenté dans la section 3.2.3.2;
14- retourner au parcours de la liste de liens pour obtenir le lien suivant.

3.2.3.2 Enregistrement des données dans une base de données

La base de données définie pour héberger les métriques calculées est une base de données Mysql. MySQL (MySQL, 2013) est un système de gestion de base de données open source. C'est l'un des puissants SGBD qui existe sur le marché et l'un des plus populaires. Parmi les points forts de Mysql :

- support des sous-requêtes et des transactions;
- support de la réplication et de clés étrangères grâces au moteur InnoDB;
- support de ODBC (Open DataBase Coneectivity) qui permet l'indépendance des systèmes de base de données des systèmes d'exploitation.

Dans le cadre de ce travail, la base de données a été définie avec une table unique, afin de faciliter les recherches et les relevés de données. Elle est décrite comme dans Figure 3.6 :

```
mysql> describe performance;

+---------------+----------------+-------+-------+---------------+----------------------+
| Field         | Type           | Null  | Key   | Default       | Extra                |
+---------------+----------------+-------+-------+---------------+----------------------+
| id            | int(15)        | NO    | PRI   | NULL          |      auto_increment  |
| host          | varchar(30)    | YES   |       | NULL          |                      |
| metric        | varchar(30)    | YES   |       | NULL          |                      |
| timestamp     | datetime       | YES   |       | NULL          |                      |
| value         | double         | YES   |       | NULL          |                      |
+---------------+----------------+-------+-------+---------------+----------------------+
5 rows in set (0.00 sec)
```

Figure 3.5 Base de données de la solution développée

Dans la description de table performance, "id" est la clé primaire, "host" contient l'identifiant du hôte supervisé, "metric" la métrique de la performance, "timestamp" l'horodate de calcul de la métrique et "value" la valeur calculée de la métrique.

À chaque calcul de délai de paquet, une entrée est ajoutée à la base de données avec la valeur de ce délai de paquet, l'hôte concerné et l'horodate courante. Cet ajout se fait grâce à la requête SQL suivante :

INSERT INTO performance (host, metric, timestamp, value) VALUES (x, y, z, t);

Où “performance” est le nom de la table, “host, metric, timestamp, value” sont les noms des colonnes de la table et “x, y, z et t” sont les valeurs qu’il faut insérées dans la base de données.

Pour lire les données depuis la base de données, il est possible de lister l’ensemble du contenu de la base de données, ou bien filtrer le contenu à afficher. Cette gestion de l’affichage se fait avec la requête SQL suivante :

SELECT *colonnes* FROM performance WHERE *condition* LIMIT *nombre*;

Où “performance” est le nom de la table, *colonnes* est les noms de colonnes à afficher – le caractère ***** signifie toutes les colonnes – et *condition* est la condition sur les lignes à afficher. L’option LIMIT a été utilisée pour afficher le nombre limité de ligne, dont le besoin avait été. *nombre* est le nombre de ligne à afficher, depuis le bas de la table.

En cas de besoin, l’ensemble des données ou une partie des données peut être supprimé de la base de données. Cela se fait avec la requête SQL suivante :

DELETE FROM performance WHERE *condition*;

Où “performance” est le nom de la table, et *condition* est la condition sur les lignes à supprimer.

3.3 Analyse de la performance

3.3.1 Effet de la couche physique sur la performance

L’outil logiciel développé dans ce projet, présenté dans la section 3.2.3, a pour but de calculer le délai de paquet des commutateurs virtuels. Le délai de paquet décrit la performance de ces derniers, comme il est argumenté dans la section 3.2.1. Mais les commutateurs virtuels sont des instances logiciels, exécutées sur des machines physiques qui peuvent influencées la performance des commutateurs virtuels.

Pour relever l'influence de la couche physique, un commutateur virtuel sera déployé, et des ressources physiques lui seront allouées. Ces derniers sont des CPU et de l'espace mémoire mémoires. Figure 3.7 décrit le commutateur en expérimentation. Un flux *F* sera généré à travers le commutateur virtuel, et les ressources physiques allouées, CPU et mémoire, au commutateur virtuel seront variées. Des relevés de la performance seront effectués à chaque variation d'une ressource physique.

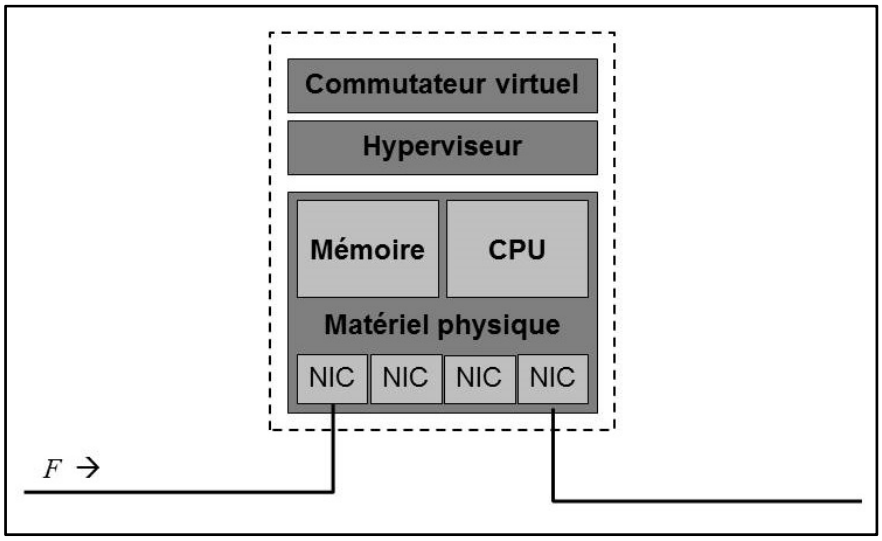


Figure 3.6 Influence de la couche physique sur la performance du commutateur virtuel

À la suite de ces expérimentations, des conclusions seront faites sur l'effet de la couche physique sur la performance de commutateur virtuel. Ainsi les expérimentations détermineront si le nombre de CPU ou l'espace mémoire alloué améliore ou détériore la performance des commutateurs virtuels. Les conclusions permettront de faire une corrélation entre la performance et les ressources physiques, ce qui aidera au dimensionnement des réseaux d'interconnexion des centres de données.

Les expérimentations seront réalisées sur un banc d'essai d'un réseau similaire aux réseaux d'interconnexion de centres de données. Les résultats et les conclusions seront présentés dans le chapitre 4 de ce mémoire.

3.3.2 Comparaison de la performance

Alors que la virtualisation ouvre les portes à un modèle économique très rentable, il reste à savoir si les commutateurs virtuels peuvent être aussi performants que les commutateurs physiques. La performance des commutateurs est décrite par le délai de paquet, comme il est argumenté dans la section 3.2.1.

Pour faire la comparaison, deux commutateurs, un virtuel et un physique, seront déployés. Les ressources physiques allouées aux commutateurs virtuels seront les mêmes que ceux du commutateur physique, comme le montre Figure 3.8. Les deux commutateurs auront le même espace mémoire "a" et la même fréquence de CPU "b". Un flux sera généré à travers les deux commutateurs pour relever le délai de paquet.

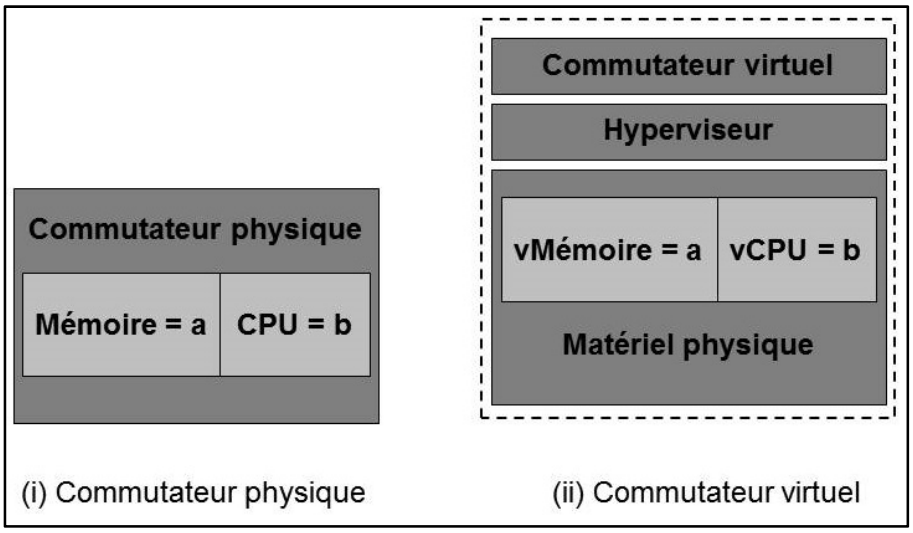


Figure 3.7 Comparaison des commutateurs virtuel et physique

Les conclusions de cette expérimentation détermineront s'il est possible d'avoir la même performance, pour les commutateurs physiques et virtuels.

L'expérimentation sera réalisée sur le même banc d'essai de la section 3.3.1. Les résultats et les conclusions seront présentés dans le chapitre 4 de ce mémoire.

3.4 Conclusion

Dans ce chapitre, nous avons déterminé la métrique décrivant la performance des commutateurs, soit le délai de paquet. Puis, le choix de l'outil pour collecter les données nécessaire pour le calcul du délai de paquet a été arrêté pour Nagios. Un outil pour automatiser la collecte des données de Nagios et calculer le délai de paquets a été présenté par la suite. Enfin, les manières pour déterminer l'effet de la couche physique sur la performance des commutateurs virtuels, et la comparaison de cette performance à celle des commutateurs physiques, ont été exposées.

CHAPITRE 4

EXPÉRIMENTATIONS ET RÉSULTATS

4.1 Introduction

Ce chapitre présente les expérimentations réalisées pour évaluer les modules logiciels développés dans le chapitre précédent. Ces modules logiciels ont pour mission le calcul du délai de paquet dans le commutateur. Les expérimentations ont été réalisées dans un banc d'essai de réseau d'interconnexion des centres de données.

La première partie présente l'environnement expérimental, et les différents scénarios exécutés sur ce banc d'essai. La suite du chapitre sera des expositions des résultats des scénarios présentés dans la première partie. Des synthèses sur la performance des réseaux d'interconnexion des centres de données, et de leurs composants, seront retenues à la fin.

4.2 Environnement expérimental

4.2.1 Métrique supervisée

La métrique supervisée est le délai de paquet dans les commutateurs, présentée dans la section 3.2.1. L'outil pour calculer le délai de paquet est présenté dans la section 3.2.3.

4.2.2 Commutateurs du banc d'essai

Arista 2.0.8

Arista 2.0.8 (Arista, 2015) est un commutateur de haute densité avec des interfaces Ethernet 100/1000 Mb et 10/40/100 Gb. Arista à une interface de management pour l'accès au commutateur pour des fins de configuration. Figure 4.1 illustre une vue du panneau arrière du commutateur virtuel Arista utilisé dans les expérimentations.

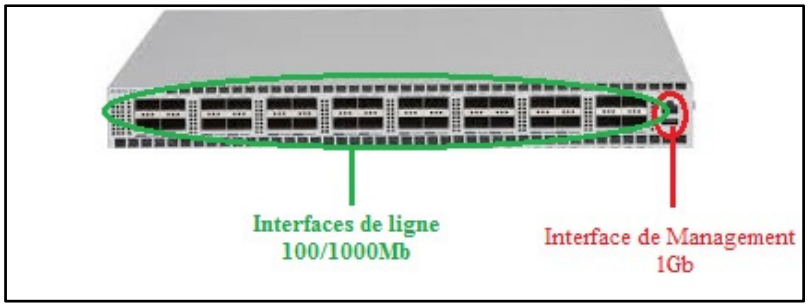


Figure 4.1 Panneau arrière du commutateur Arista 2.0.8

Extreme X450e-48p

Extreme X450e-48p (Extreme Network, 2013) est un commutateur qui répond aux besoins de transport et d'agréation de grosses quantités de données, à des vitesses allons de 1 Gb à 10 Gb. Figure 4.2 illustre une vue du panneau arrière du commutateur, utilisé dans les expérimentations.

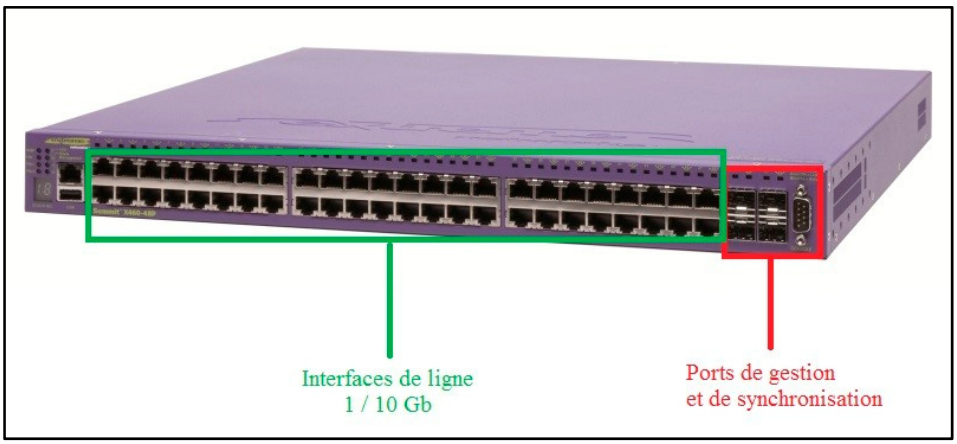


Figure 4.2 Panneau arrière du commutateur Extreme X450e-48p

Les caractéristiques physiques du Extreme X450e-48p sont présentées dans Tableau 4.1 :

Tableau 4.1 Caractéristiques physiques
de Extreme X450e-48p

CPU	Mémoire
1CPU 2GHz	2G

4.2.3 Déploiement du banc d'essai

Pour valider notre solution de la collecte automatique des métriques de la performance des commutateurs virtuels, un banc d'essai a été déployé. Ce dernier est un réseau d'architecture Spine-Leaf, similaire aux réseaux d'interconnexion des centres de données, présentés dans la section 2.2.2. Figure 4.3 illustre le banc d'essai déployé, constitué d'un Spine et de deux Leafs.

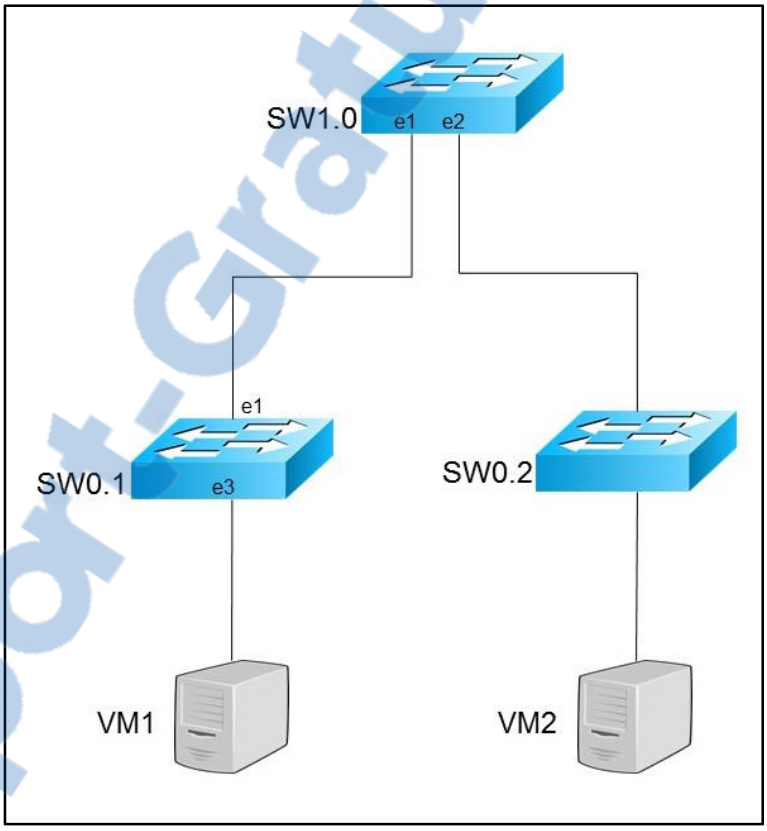


Figure 4.3 Banc d'essai

Le commutateur du niveau Spine, SW1.0, a deux interfaces, qui sont connectées aux deux commutateurs du niveau Leaf, SW0.1 et SW0.2. Les commutateurs du niveau Leaf ont chacun une interface qui les interconnecte au commutateur du niveau Spine, de plus des interfaces d'interconnexion des hôtes serveurs. Dans ce banc d'essai, un seul hôte a été connecté à chacun des commutateurs Leaf.

Le banc d'essai a été réalisé sur deux serveurs, dont les caractéristiques techniques sont illustrées dans Tableau 4.2. La technologie de virtualisation utilisée est KVM. Les interfaces de ligne des commutateurs, utilisés dans ces expérimentations, sont d'un débit de 1Gb.

L'annexe I présente le déploiement de l'hyperviseur KVM, et l'annexe II détaille les étapes de déploiement du banc d'essai.

Tableau 4.2 Caractéristiques des serveurs hébergeurs du banc d'essai

Serveur	**Système d'exploitation**	**CPU**	**Mémoire**	**Contenu**
Serveur 1	Ubuntu Server 12.04 x64 TLS	12CPU x 2GHz	24G	SW1.x
Serveur 2	Ubuntu Server 12.04 x64 TLS	12CPU x 2GHz	24G	SW0.x et VMx

Trois scénarios ont été réalisés sur le banc d'essai au-dessus. Le premier calcule la performance des commutateurs virtuels, dans un environnement d'opération ordinaire. Le deuxième dresse l'influence de la couche physique sur la performance des commutateurs virtuels. Le troisième scénario est pour comparer la performance des commutateurs virtuels, à celle des commutateurs physiques. Tableau 4.3 présente la configuration des nœuds du réseau.

Tableau 4.3 Composants du banc d'essai

Scénarios	**Commutateur**	**Composant**
Scénario 1	SWx.y	Arista 2.0.8
Scénario 2	SWx.y	Arista 2.0.8
Scénario 3	SW1.0 et SW0.1	Arista 2.0.8
	SW0.2	Ciena 5160

4.2.4 Scénarios des expérimentations

4.2.4.1 Scénario 1 : mesure de la performance d'un commutateur virtuel

Objectif

Dans ce scénario, on calcule la performance des commutateurs virtuels, dans un environnement d'opération ordinaire, pour avoir une base pour notre étude de la performance.

Environnement

Les trois commutateurs du banc d'essai sont des commutateurs virtuels, hébergés par des machines virtuelles. Dans ce scénario, les ressources allouées aux commutateurs, soient le CPU et l'espace mémoire, sont fixes, et sont représentées dans Tableau 4.4.

Tableau 4.4 Ressources allouées dans l'état initial

Machine virtuelle	CPU	Mémoire
SWx.y	1CPU x 2GHz	2G
VMx	1CPU x 2GHz	2G

4.2.4.2 Scénario 2 : influence de la couche physique sur la performance du commutateur virtuel

Objectif

Dans ce scénario, on cherche à avoir une compréhension de l'influence de la couche physique sur la performance du commutateur virtuel.

Environnent

La même architecture de l'état initial est reprise, avec des interfaces de ligne de 1Gb. Les ressources physiques allouées aux commutateurs seront variées, afin d'avoir l'effet du CPU et de la mémoire sur les commutateurs virtuels. Deux cas seront mis en expérimentation : l'augmentation de nombre de CPU et l'augmentation de l'espace mémoire.

Les flux seront générés entre les deux machines terminales, VM1 et VM2. Les augmentations de ressources et les mesures de la performance seront relevées sur un des commutateurs, puisque les trois commutateurs virtuels sont symétriques.

4.2.4.3 Scénario 3 : comparaison d'un commutateur virtuel et un commutateur physique

Objectif

Dans ce scénario, on compare la performance d'un commutateur virtuel, à celle d'un commutateur physique. L'objectif est d'avoir une idée sur la performance des nouvelles technologies de virtualisation face aux technologies existantes.

Environnement

Un des commutateurs virtuels de l'état initial est remplacé par un commutateur physique. Cette configuration physique du commutateur Extreme correspond à celle du commutateur Arista dans scénario 1, avec 1 CPU 2GHz et une mémoire 2G.

Les flux seront générés entre deux machines hôtes, VM1 et VM2. Les mesures de la performance seront relevées sur le commutateur physique. Le délai de paquet des commutateurs virtuels Arista est celui calculé dans le scénario 1. Le délai de paquet du commutateur physique Extreme y sera déduit.

4.3 Relevé des données

4.3.1 Données brutes

4.3.1.1 Mesures par SNMP

Depuis Nagios, on collecte à base de SNMP les mesures de deux variables MIB :
ifLinkStaus : l'état du lien, up ou down;
ifMTU : la taille du plus grand paquet, transitant par l'interface.

L'installation et la configuration de Nagios sont montrées dans l'annexe III. L'enregistrement dans la base de données de Nagios est montré dans l'annexe VI.

Afin de configurer la collecte de ces deux objets SNMP sur Nagios, on aura besoin des Object ID (OID). Tableau 4.5 donne les OID des trois variables MIB.

Tableau 4.5 Les Objects ID des variables MIB

Variable MIB	OID	Description
ifLinkStatus	.1.3.6.1.2.1.2.2.1.10	l'état du lien, up ou down
ifMTU	.1.3.6.1.2.1.2.2.1.4	Taille du plus grand paquet, transitant par l'interface en octets

La configuration de la collecte de ces données dans Nagios est détaillée dans l'annexe IV.

Dans Figure 4.4, une capture de la base de données de Nagios montrant un exemple des données collectées.

```
mysql> select * from data_collection;
+--------------+---------------+----------------------+----------------------+-----------+
 | id           | host          | metric               | timestamp            | value     |
+--------------+---------------+----------------------+----------------------+-----------+
 | 276          | sw1-0         | Port+1+ifLinkStatus| 2015-06-19 10:11:43|           1 |
 | 278          | sw1-0         | Port+1+ifMTU       | 2015-06-19 10:11:43|      1500 |
+--------------+---------------+----------------------+----------------------+-----------+
305 rows in set (0.00 sec)
```

Figure 4.4 Vu de la base de données de Nagios

4.3.1.2 Mesures par le plugin MRTG de Nagios

De plus des mesures de ces deux variables MIB, on collecte, à base du plugin MRTG, le débit de l'interface. En effet, MRTG fait appel au SNMP, pour relever des mesures, à base desquelles il calcule le débit du flux entrant et le débit du flux sortant, comme montré dans Figure 4.5.

```
mysql> select * from data_collection;
+--------+--------------------+-------------------------+-----------------------------------------------+
 | host   | metric             | timestamp               | value                                         |
+--------+--------------------+-------------------------+-----------------------------------------------+
 | sw1-0 | Port+1+ifTraffic| 2015-06-19 10:11:43 | AVG IN = 680.0 B/s AVG OUT = 11.2 KB/s |
+--------+--------------------+-------------------------+-----------------------------------------------+
305 rows in set (0.00 sec)
```

Figure 4.5 Vu de la base de données de Nagios

L'annexe V donne plus d'amples sur le plugin MRTG.

4.3.2 Données calculées

4.3.2.1 Base de données

Les données calculées sont enregistrées dans une base de données dédiée. Cette base de données est à table unique, appelée performance. La description de la base de données est présentée dans la section 3.2.3. Le déploiement de la base de données est présenté dans l'annexe VI.

4.3.2.2 Débit effectif de l'interface

On identifie le débit effectif d'une interface par ifThroughput. Il est calculé en octets/seconde. Et grâce à la formule suivante :

$$\text{Débit effectif} = \max(\text{Débit du flux entrant}, \text{Débit du flux sortant}) \quad (4.1)$$

Autrement exprimé avec la formule :

$$\text{ifThroughput} = \max(\text{ifTraffic AVG IN}, \text{ifTraffic AVG OUT}) \quad (4.2)$$

Dans ce qui suit, une capture de la table des métriques calculées, montrant un exemple d'un résultat enregistré de calcul de ifThroughput, en octets/seconde.

```
mysql> select * from performance;
+---------------+---------------+-----------------------+-----------------------+---------------+
| id            | host          | metric                | timestamp             | value         |
+---------------+---------------+-----------------------+-----------------------+---------------+
| 17            | sw1-1         | Port+1+ifTroughput | 2015-06-19 10:27:44 |  11200000 |
+---------------+---------------+-----------------------+-----------------------+---------------+
29 rows in set (0.00 sec)
```

Figure 4.6 Vu de la base de données des résultats

4.3.2.3 Latence d'un lien

On identifie la métrique du délai du lien par ifLatency. Il est calculé en seconde, et grâce à la formule :

$$\text{Délai du lien} = \frac{\text{Taille de paquets}}{\text{Débit effectif}} \tag{4.3}$$

Où la taille de paquets est le MTU minimum du chemin de ces derniers.
Autrement exprimé avec la formule :

$$\text{ifLatency} = \frac{\min(\text{ifMTU})}{\text{ifThroughput}} \tag{4.4}$$

Dans ce qui suit, une capture de la table des métriques calculées, montrant un exemple d'un résultat enregistré de calcul de ifLatency, en seconde.

```
mysql> select * from performance;
+--------------+-------+-----------------------+-----------------------+--------------------------+
| id           | host  | metric                | timestamp             | value                    |
+--------------+-------+-----------------------+-----------------------+--------------------------+
| 18           | sw1-1 | Port+1+ifLatency      | 2015-06-19 10:27:46 | 0.4900281869914375 |
+--------------+-------+-----------------------+-----------------------+--------------------------+
29 rows in set (0.00 sec)
```

Figure 4.7 Vu de la base de données des résultats

4.3.2.4 Vérification de l'exactitude des mesures

Pour vérifier l'exactitude des mesures relevées par Nagios, un calcule de temps de latence d'un lien a été comparé avec l'outil Ping. Les mesures ont été relevées d'un cas simplifié de deux hôtes connectés par une liaison directe, comme l'illustre Figure 4.8.

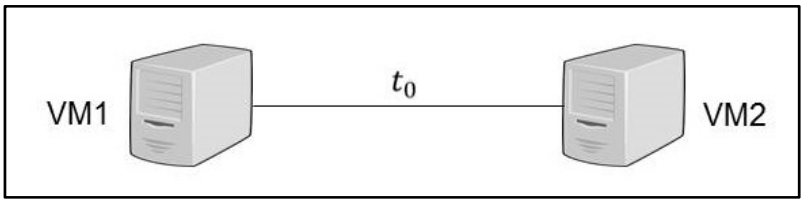


Figure 4.8 Temps de latence

Où t_0 est le temps de latence entre deux hôtes, Host1 et Host2.
Pour Nagios, t_0 est calculé avec la formule présentée dans la section 3.2.2. Pour Ping, t_0 est calculé avec l'outil présenté dans l'annexe VII.

Une centaine de mesures ont été relevées, sur des périodes de 5 minutes, pour un flux de paquets de taille de 1500 octets. Les valeurs du délai de paquet sont supposées de décrémenter légèrement après la première mesure, car celle-ci comporte aussi le temps d'apprentissage : résolution ARP et résolution de la table de transfère. Figure 4.9 illustre les résultats des mesures pour (a) Nagios et (b) Ping; Tableau 4.6 illustre la moyenne et les valeurs maximale et minimale des mesures, en ms.

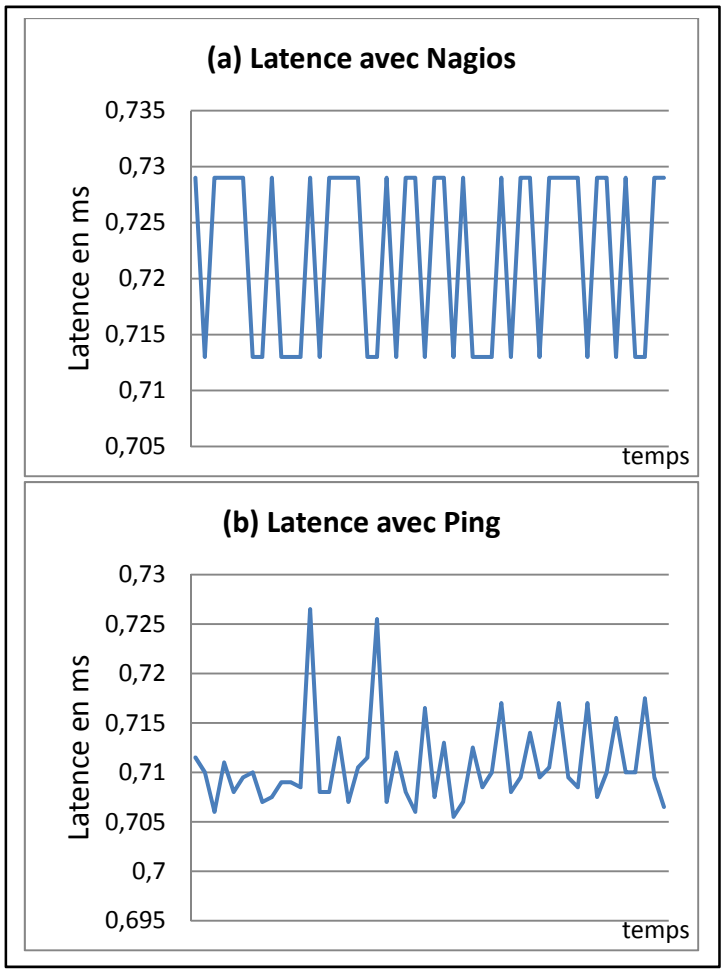


Figure 4.9 Comparaison de la latence
mesurée par Nagios et Ping

Tableau 4.6 Comparaison de la latence mesurée par Nagios et Ping
pour les paquets de 1500 octets

Outil	Maximum	Minimum	Moyenne
Nagios	0.729	0.713	0.722
Ping	0.726	0.705	0.710

Tableau 4.7 Différence entre la latence mesurée par Nagios et Ping pour les paquets de 1500 octets

	$\Delta_{Maximum}$	$\Delta_{Minimum}$	$\Delta_{Moyenne}$
Différence	0.342 %	1.051 %	1.592 %

Vu que les mesures prises par Nagios et Ping sont en moyenne différentes de 1,592%, on conclut que les valeurs de Nagios sont exactes.

4.4 Analyse de la performance des commutateurs virtuels

4.4.1 Mesure de la performance d'un commutateur virtuel

Une centaine de mesures ont été faites, sur des périodes de 5 min. La figure 4.10 illustre les résultats de ces mesures. Tableau 4.9 illustre la moyenne et les valeurs maximale et minimale des mesures, en secondes.

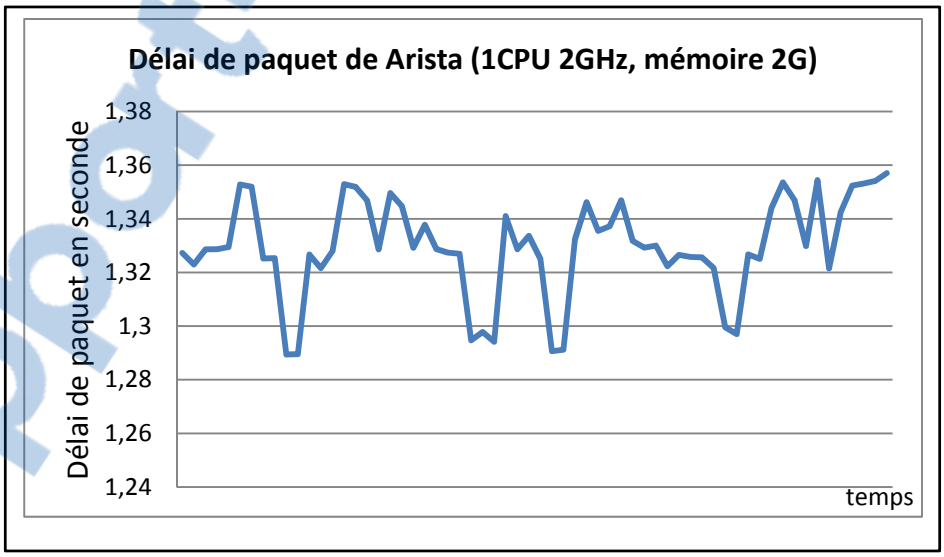


Figure 4.10 Variation dans le temps du délai de paquet de Arista (1CPU 2GHz, mémoire 2G)

Tableau 4.8 Statistiques du délai de paquet de Arista
(1CPU 2GHz, mémoire 2G)

Maximum	Minimum	Moyenne
1.357	1.289	1.330

En conclusion, le délai de paquet dans le commutateur virtuel Arista, avec la configuration physique d'un CPU 2GHz et une mémoire 2G est :

$$d_{Packet}\left(Arista_{2GHz,2G}\right) = 1.323 \pm 0.034 \text{ s} \tag{4.5}$$

4.4.2 Influence de la couche physique sur la performance du commutateur virtuel

4.4.2.1 Influence du CPU

Dans ce cas, un deuxième CPU de 2GHz est alloué au commutateur virtuel Arista. Une centaine de mesures ont été faites, sur des périodes de 5 min. La figure 4.11 illustre les résultats de ces mesures. Tableau 4.10 illustre la moyenne et les valeurs maximale et minimale des mesures, en secondes.

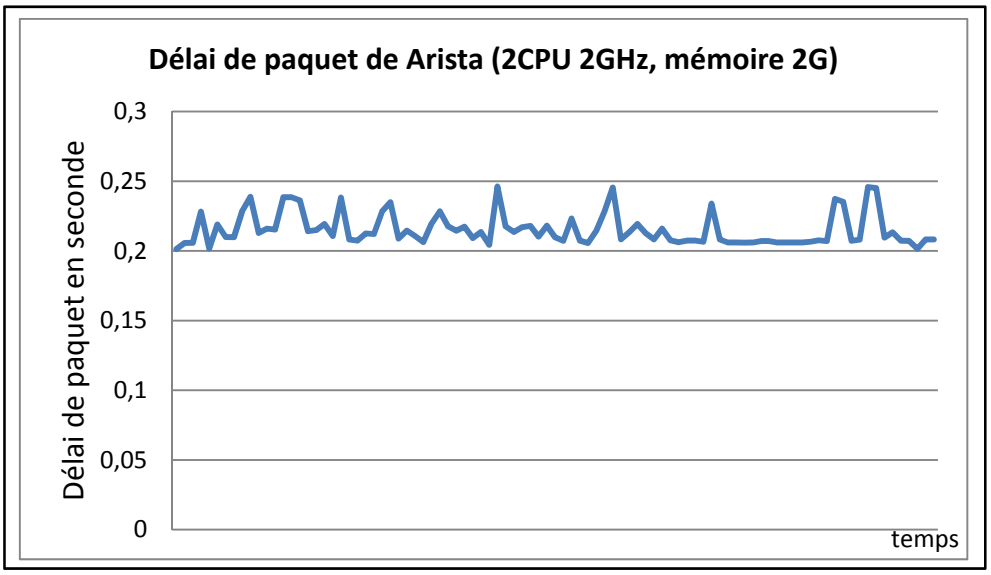


Figure 4.11 Variation dans le temps du délai de paquet de Arista
(2CPU 2GHz, mémoire 2G)

Tableau 4.9 Statistiques du délai de paquet de Arista
(2CPU 2GHz, mémoire 2G)

Maximum	Minimum	Moyenne
0.246	0.201	0.215

En conclusion, le délai de paquet dans le commutateur virtuel Arista, avec la configuration physique d'un CPU 4GHz et une mémoire 2G est :

$$d_{Packet}\left(Arista_{4GHz,2G}\right) = 0.223 \pm 0.023\ s \tag{4.6}$$

En comparant ce résultat avec celui obtenu dans le scénario 1, le délai de paquet d'un commutateur virtuel Arista à un CPU de 4GHz est de l'ordre de quelques centaines de millisecondes, alors que celui d'un commutateur virtuel Arista à un CPU de 2GHz est de secondes.

Le nombre de CPU a été levé, par la suite, graduellement jusqu'à 7 CPU. Figure 4.12 et Tableau 4.11 résument les résultats.

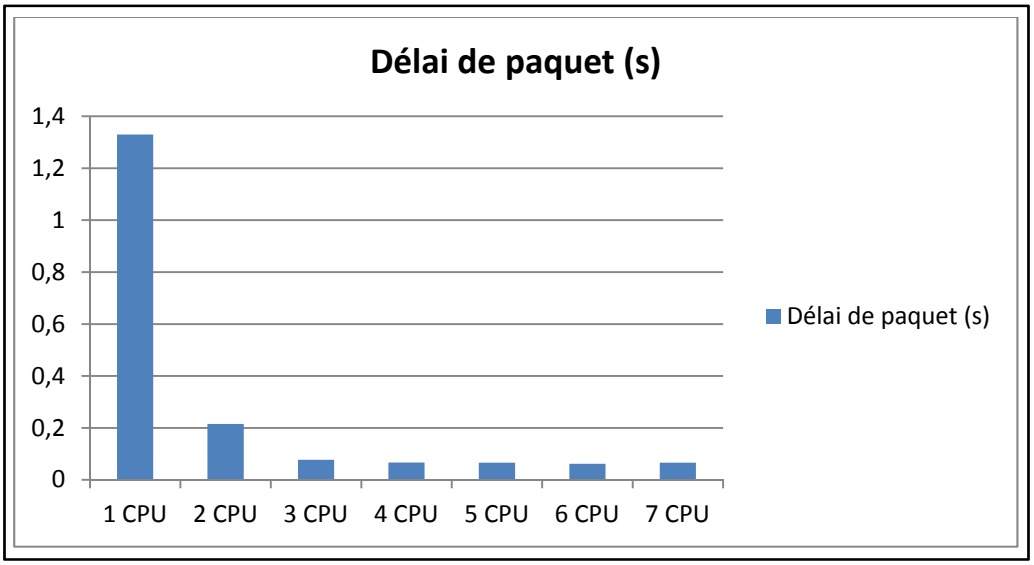


Figure 4.12 Délai de paquet en fonction de nombre de CPU

Tableau 4.10 Délai de paquet
en fonction de nombre de CPU

Nombre de CPU	Délai de paquet en seconde
1	1.330
2	0.215
3	0.077
4	0.067
5	0.066
6	0.066
7	0.066

Le délai de paquet décrémente rapidement avec le changement de nombre de CPU de 1 à 4, en allant de 1.284 s à 0.067 s, pour se stabilisé à 0.066 s. Cela signifie que Arista opère dans sa meilleure performance à partir de 4 CPU. La décrémentation de la valeur du délai de paquet lorsque le nombre de CPU est varié de 1 à 4, laisse faire la conclusion :
La fréquence du CPU peut influencer la performance des commutateurs virtuels.

4.4.2.2 Influence de la mémoire

Dans ce cas, l'espace mémoire alloué au commutateur virtuel Arista a été levé, par rapport scénario 1, de 2G à 3G. La taille de paquet est de 1500 octets. Une centaine de mesures ont été faites, sur des périodes de 5 min. Figure 4.13 illustre les résultats de ces mesures. Tableau 4.12 illustre la moyenne et les valeurs maximale et minimale des mesures, en secondes.

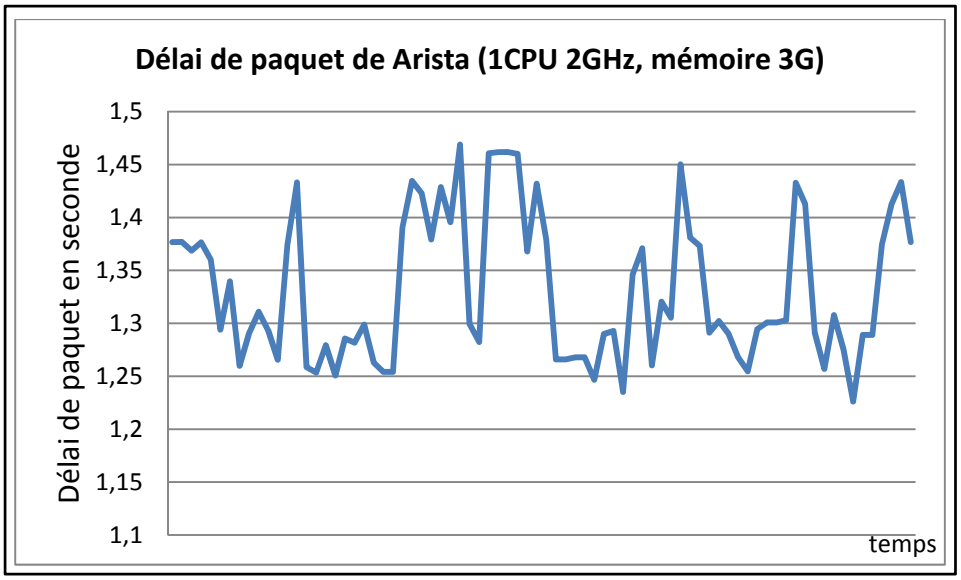


Figure 4.13 Variation dans le temps du délai de paquet de Arista (1CPU 2GHz, mémoire 3G)

Tableau 4.11 Statistiques (en s) du délai de paquet de Arista (1CPU 2GHz, mémoire 3G)

Maximum	Minimum	Moyenne
1.468	1.226	1.331

En conclusion, le délai de paquet dans le commutateur virtuel Arista, avec la configuration physique d'un CPU 2GHz et une mémoire G est :

$$d_{Packet}\left(Arista_{2GHz,3G}\right) = 1.341 \pm 0.127 \text{ s} \tag{4.7}$$

On remarque que la mémoire n'a pas d'influence significative sur la performance du commutateur virtuel. Pour valider ce résultat, pour trois tailles de paquet différentes, 500 octets, 1500 octets et 3000 octets, l'espace mémoire a été levé graduellement de 2 G à 8 G.

Paquets de 500 octets

Figure 4.15 et Tableau 4.14 résume les résultats de la variation du délai de paquet en fonction de l'espace mémoire alloué dans le cas d'un paquet de 500 octets.

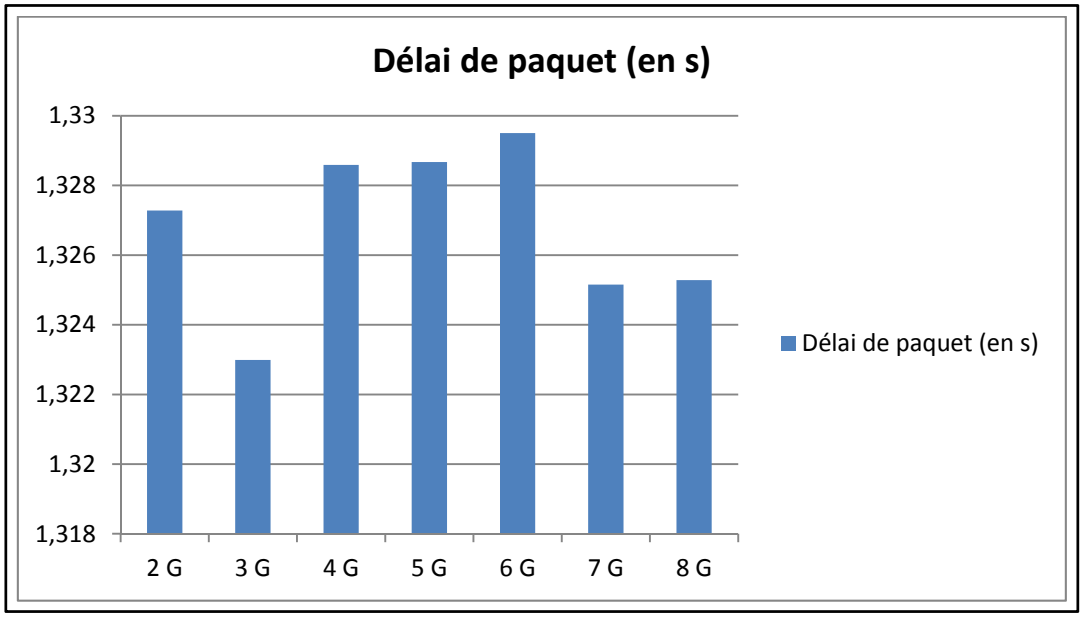


Figure 4.14 Délai de paquet en fonction de l'espace mémoire
cas de paquet de 500 octets

Tableau 4.12 Délai de paquet en fonction
de l'espace mémoire - cas de paquet de 500 octets

Espace mémoire en G	**Délai de paquet en seconde**
2	1.327
3	1.322
4	1.328
5	1.328
6	1.329
7	1.325
8	1.325

Les résultats montrent que le délai de paquet du commutateur Arista est pratiquement stable pour un paquet de 500 octets et une variation de l'espace mémoire de 2 G à 8 G, fluctuant entre 1.322 s et 1.329 s.

Paquets de 1500 octets

Figure 4.14 et Tableau 4.13 résume les résultats de la variation du délai de paquet en fonction de l'espace mémoire alloué dans le cas d'un paquet de 1500 octets. Les résultats montrent que le délai de paquet du commutateur Arista est pratiquement stable pour un paquet de 1500 octets et une variation de l'espace mémoire de 2 G à 8 G, fluctuant entre 1.315 s et 1.331 s.

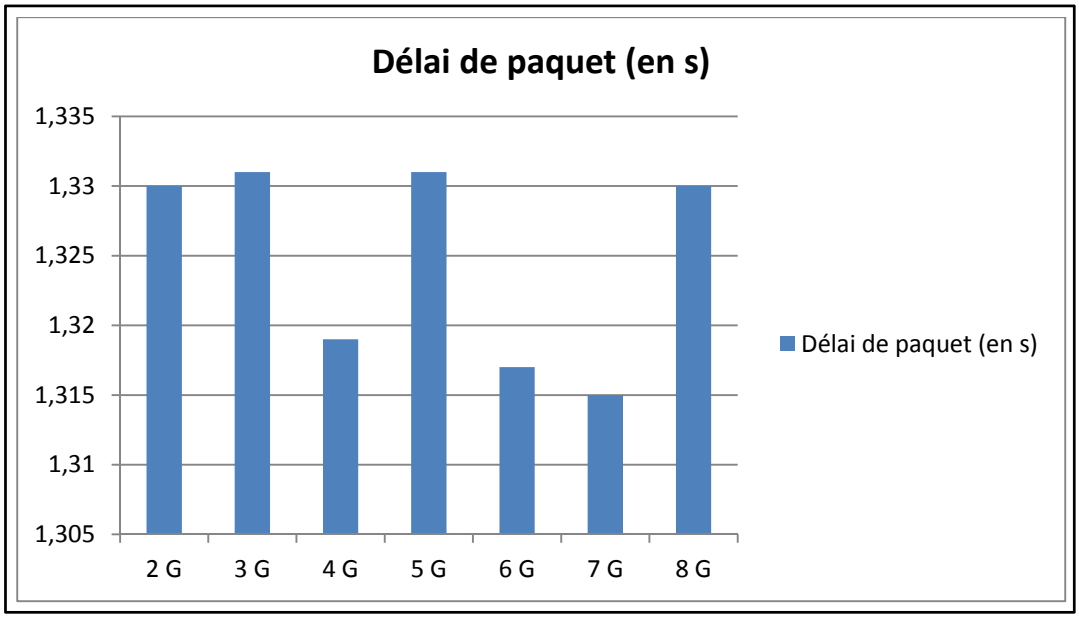


Figure 4.15 Délai de paquet en fonction de l'espace mémoire
cas de paquet de 1500 octets

Tableau 4.13 Délai de paquet en fonction
de l'espace mémoire - cas de paquet de 1500 octets

Espace mémoire en G	Délai de paquet en seconde
2	1.330
3	1.331
4	1.319
5	1.331
6	1.317
7	1.315
8	1.330

Paquets de 3000 octets

Figure 4.16 et Tableau 4.15 résume les résultats de la variation du délai de paquet en fonction de l'espace mémoire alloué dans le cas d'un paquet de 3000 octets.

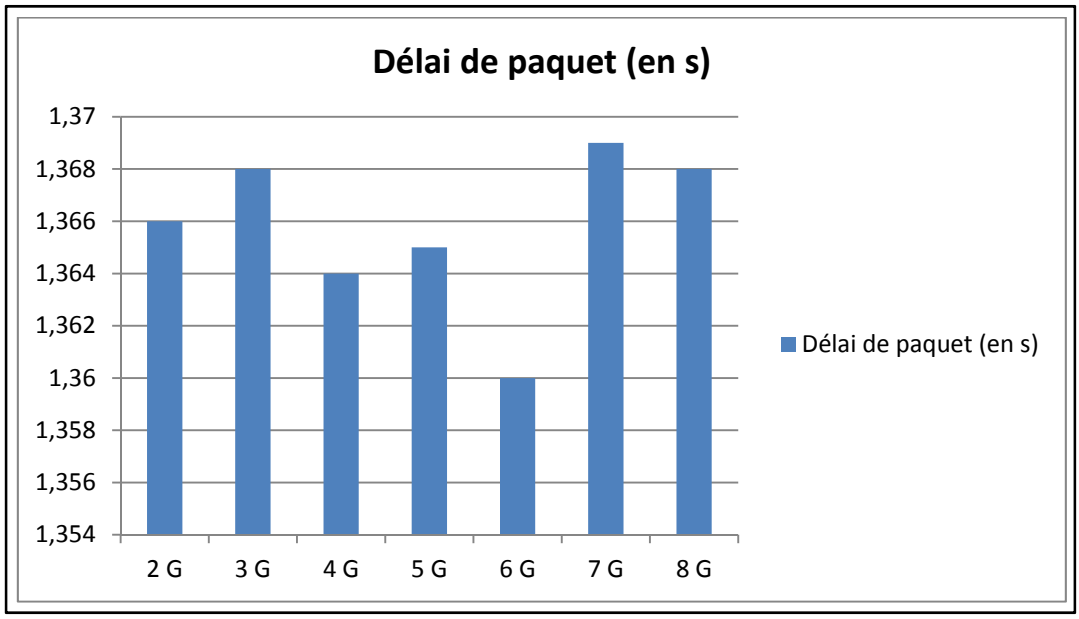


Figure 4.16 Délai de paquet en fonction de l'espace mémoire
cas de paquet de 3000 octets

Tableau 4.14 Délai de paquet en fonction
de l'espace mémoire - cas de paquet de 3000 octets

Espace mémoire en G	**Délai de paquet en seconde**
2	1.366
3	1.368
4	1.364
5	1.365
6	1.360
7	1.369
8	1.368

Les résultats montrent que le délai de paquet du commutateur Arista est pratiquement stable pour un paquet de 3000 octets et une variation de l'espace mémoire de 2 G à 8 G, fluctuant entre 1.360 s et 1.369 s.

Conclusion

Trois tailles de paquets, 500 octets, 1500 octets et 3000 octets, faisaient l'objet d'une expérimentation sur le commutateur virtuel Arista. Pour chacune des tailles de paquet, l'espace de mémoire alloué au commutateur virtuel Arista a été varié de 2 G à 8 G. Dans Figure 4.16 un résumé des résultats.

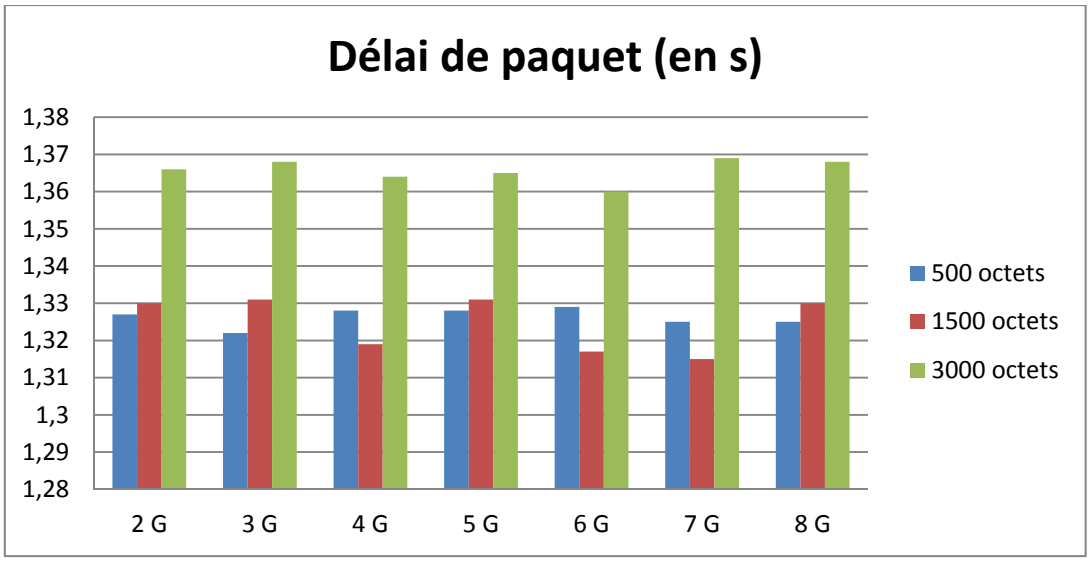


Figure 4.17 Résumé de l'impact de l'espace mémoire sur la performance

Dans les trois cas, la performance ne varie pas significativement en fonction de la variation de l'espace mémoire. Cela confirme la première observation faite plus haut, et puis on conclut :

L'espace mémoire alloué à un commutateur virtuel, n'a pas une influence significative sur sa performance.

4.4.3 Comparaison d'un commutateur virtuel et un commutateur physique

Une centaine de mesures ont été faites, sur des périodes de 5 min, sur chacun des commutateurs physiques Extreme et virtuel Arista 2.0.8. Les ressources physiques alloué au commutateur virtuel Arista sont les même que ceux du commutateur physique Extreme. Cette configuration corresponde à celle du scénario 1. Figure 4.15 illustre les résultats des mesures sur le commutateur physique Extreme. Tableau 4.14 illustre la moyenne et les valeurs maximale et minimale des mesures, en secondes.

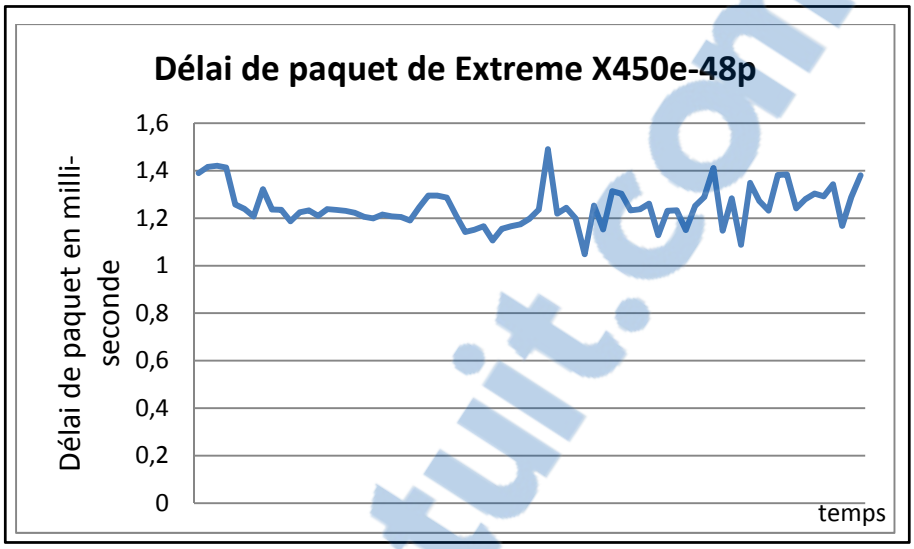


Figure 4.18 Variation dans le temps du délai de paquet de Extreme X450e-48p

Tableau 4.15 Statistiques (en ms) du délai de paquet de Extreme X450e-48p

Maximum	Minimum	Moyenne
1.490	1.048	1.247

En conclusion, le délai de paquet dans le commutateur physique Extreme est :

$$d_{Packet}(Extreme) = 1.269 \pm 0.221\text{ms} \quad (4.8)$$

En comparant les résultats de Extreme avec ceux obtenus pour Arista dans la section 4.4.1, il est à noter que le délai de paquets du commutateur physique Extreme est de l'ordre de millisecondes, alors que le délai de paquets du commutateur virtuel Arista est de l'ordre de secondes. On conclut ainsi que le commutateur physique Extreme est largement plus performant que le commutateur virtuel Arista.

4.5 Conclusion

Un banc d'essai a été déployé pour expérimenter l'outil logiciel développé dans la section 3.2.3. Par la suite, deux expérimentations ont été exécutées sur ce banc d'essai. La première pour relever l'impact de la couche physique sur la performance des commutateurs virtuels, et la deuxième pour comparer les performances d'un commutateur virtuel et un commutateur physique.

Les résultats ont montré que le CPU est un déterminant de la performance du commutateur virtuel, alors que l'espace mémoire alloué n'a pas d'impact significatif sur la performance. En comparant la performance du commutateur virtuel à celle du commutateur physique, une excellence de ce dernier est enregistrée.

CHAPITRE 5

CONCLUSION GÉNÉRALE

Alors que la migration vers les centres de données prend de la popularité, la performance des réseaux d'interconnexion de ces derniers, reste en question. En effet, les centres de données sont largement constitués d'équipements virtuels, et leurs réseaux d'interconnexion ne feront pas l'exception, et ils sont en migration vers les commutateurs virtuels. Cette migration, motivée par les gains économiques, met en question la supervision de la performance des commutateurs virtuels.

La supervision de la performance des commutateurs virtuels étant difficile, ce mémoire répond à une partie de la problématique. En effet, ce mémoire a pour objectif le développement d'un outil pour l'automatisation la supervision de la performance des commutateurs virtuels, afin de réaliser deux objectifs. Le premier est de relever l'impact de la couche physique sur la performance des commutateurs virtuels; et le deuxième est de comparer la performance des commutateurs virtuels, à celle des commutateurs physiques. Le chapitre 1 de ce mémoire présente la problématique et les objectifs de ce mémoire.

Le chapitre 2 expose l'état de l'art en matière de technologies de réseaux d'interconnexion de centres de données, et de la supervision de la performance. La virtualisation est la technologie clé des centres de données, dont les réseaux bénéficient et des architectures spécifiques de réseaux pour les besoins des centres de données sont déployées. Plusieurs métriques de performance des réseaux traditionnels sont utilisées selon les besoins de la supervision et une variété des outils pour superviser ces métriques sont en existence. Des travaux de recherche ont usage des outils de la supervision de la performance des réseaux traditionnels, pour relever l'impact de l'hyperviseur et le système d'exploitation hébergeur sur la performance des commutateurs virtuels, et l'impact du matériel physique sur performance des applications sur les machines virtuelles.

Le chapitre 3 présente la méthodologie de recherche pour répondre à la problématique. Il était argumenté que le délai de paquet dans le commutateur est la métrique qui décrit le plus la performance des commutateurs. Après une comparaison des outils les plus utilisés, Nagios a été sélectionné comme outil à base duquel les mesures seront faites. Ensuite, un outil logiciel a été présenté pour automatiser la collecte des données depuis Nagios et calculer le délai de paquet. Puis, les scénarios des expérimentations pour valider l'outil logiciel développé ont été détaillés.

Le chapitre 4 présente les résultats des expérimentations. Le CPU du commutateur détermine largement la performance d'un commutateur. En passant d'un CPU de 2 GHz à 7 CPU de 2 GHz, le délai de paquet à passer de l'ordre de secondes, à quelques dizaines de millisecondes. L'espace mémoire alloué au commutateur virtuel n'a pas d'impact significatif sur la performance du commutateur virtuel. En effet, en changeant l'espace mémoire alloué de 2 G à 8 G, le délai de paquet fluctuait légèrement autour d'une valeur fixe. En comparant la performance du commutateur virtuel à celle d'un commutateur physique, une large différence est observée : le délai de paquet du commutateur virtuel est de l'ordre de secondes, alors que celle du commutateur physique est de l'ordre de milliseconde.

En conclusion, pour les commutateurs déployés dans les réseaux d'interconnexion des centres de données, la virtualisation attrayante d'un point de vue économique, peut avoir des défauts majeurs d'un point de vue performance, en comparaison aux technologies traditionnelles. Toutefois, la performance peut être améliorée en allouant plus de fréquence CPU à la couche virtualisation.

Alors que nous sommes capables d'avoir une description de la performance d'un commutateur virtuel, et nous sommes capables de comparer la performance de plusieurs commutateurs; la perspective future est de comparer la performance de plusieurs routes dans un même réseau de centre de données, en se basant sur la performance des commutateurs.

ANNEXE I

INSTALLATION DE L'HYPERVISEUR KVM

KVM est un hyperviseur de type 2.
Libvirt est un gestionnaire de machines virtuelles.
Virt-manager est une interface grafique pour la gestion des machines virtuelles.

Installation

sudo apt-get install qemu-kvm libvirt-bin virt-manager

Affichage de la liste des machines virtuelle

virsh list

Définition d'une machine virtuelle

virsh define *fichier_xml*

Démarrage d'une machine virtuelle

virsh start *Nom_de_domaine_de_la_machine_virtuelle*

Destruction d'une machine virtuelle

virsh destroy *Nom_de_domaine_de_la_machine_virtuelle*

Suppression d'une machine virtuelle

virsh undefine *Nom_de_domaine_de_la_machine_virtuelle*

ANNEXE II

DÉPLOIEMENT D'UN RÉSEAU D'INTERCONNEXION DE CENTRES DE DONNÉES

A II.1. Introduction

Architecture de centre de données utilisée pour l'enivrement d'expérimentation est l'architecture Spine Leaf. C'est la base de deux architectures très répondues pour les centres de données, soient FoConn (22) et Fat-Tree (23).

Deux serveurs physiques ont été alloués pour le déploiement du réseau des commutateurs virtuels :

Serveur 1 : 172.16.0.69
Serveur 2 : 172.16.0.22

L'architecture Spine Leaf utilisée est à 2 spines SW0.x et 4 leafs SW1.x. Les commutateurs dans cette architecture sont des commutateurs virtuels Arista. Les hôtes utilisateurs VMx sont des machines Ubuntu.

Des tunels GRE-Tap, netxy, ont simulé les liaisons entre les commutateurs. Des bridges, brx, ont simulé les liens entre les hôtes et les commutateurs d'accès.

Tous les composants du réseau ont été reliés à une interface de gestion, br0, pour les connecter à la station de monitoring.

La station de monitoring est une machine sur laquelle l'outil de supervision de la performance, Nagios, a été déployé.

La figure suivante illustre le détail de la topologie déployée.

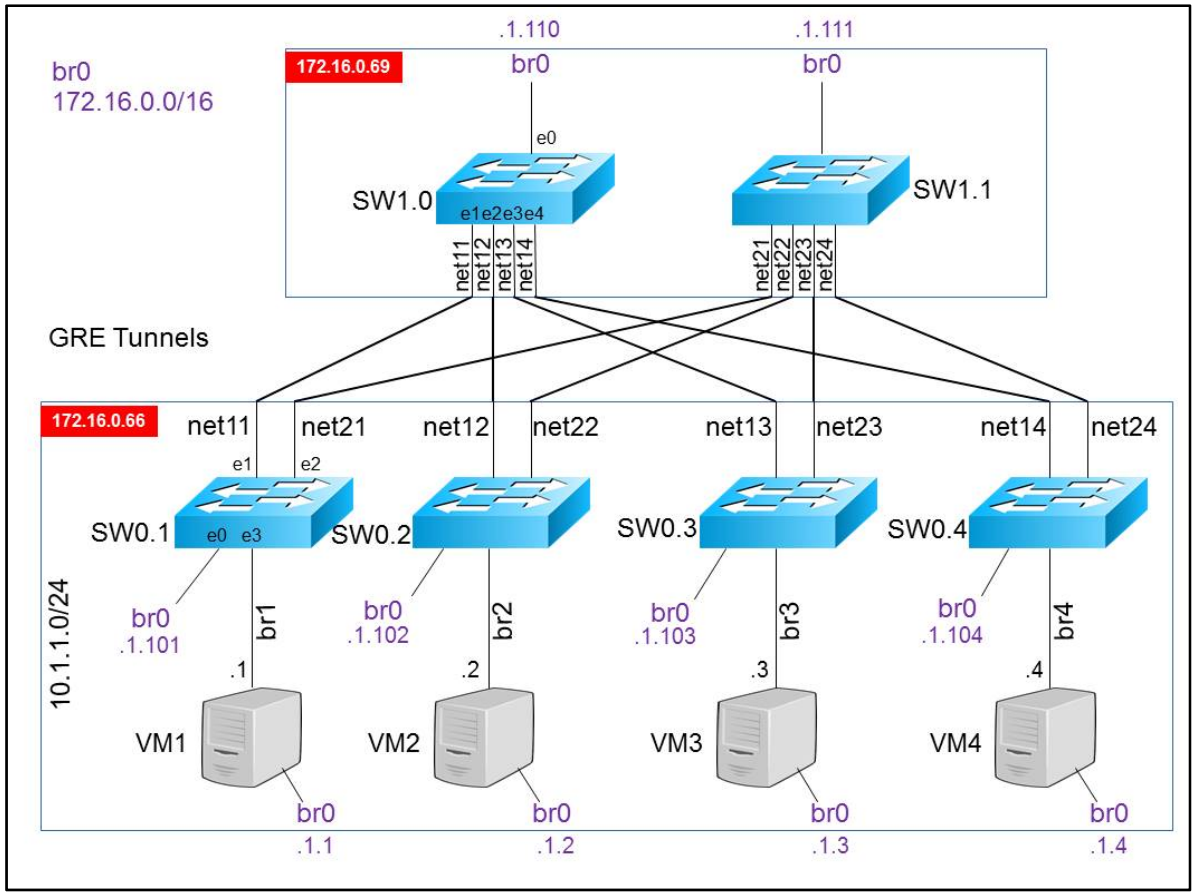


Figure-A II-1 Topologie de centre de donnée déployée

A II.2. Création des tunnels GRE-Tap (layer 2)

Sur le serveur 172.16.0.69, créer le tunnel net11

ip link add net11 mode gretap remote 172.16.0.66 local 172.16.0.69 ttl 255 key 11
ip link set net11 up

De la même manière, créer les tunnels, net12, net13, net14, net21, net22, net23 et net24

ip link add netxy mode gretap remote 172.16.0.66 local 172.16.0.69 ttl 255 key xy
ip link set netxy up

Créer l'autre bout du tunnel sur le serveur 172.16.0.66

ip link add netxy mode gretap remote 172.16.0.69 local 172.16.0.66 ttl 255 key xy
ip link set netxy up

C'est le numéro clé, key xy, qui fait le lien entre les deux bouts d'un tunnel.

Les tunnels GRE-Tap étaient connectés aux commutateurs par le média des bridges

A II.2.1. Création des bridges

Créer les bridges sur les deux serveurs.

Pour le tunnel net11

```
sudo brctl addbr br11
sudo brctl addif br11 net11
sudo ifconfig br11 up
```

De la même manière, créer les bridges pour le reste des tunnels

```
sudo brctl addbr brxy
sudo brctl addif brxy netxy
sudo ifconfig brxy up
```

A II.3. Création des commutateurs virtuels

Télécharger Aboot-veos-serial-2.08.iso et vEOS-lab-4.14.2F.vmdk depuis la page de téléchargement de Arista.

Configurer le fichier XML de la machine virtuelle

```
<domain type='kvm'>
 <name>Nom_de_domaine_Commutateur</name>
 <memory>Espace_Mémoire</memory>
 <vcpu>Nombre_CPU</vcpu>
 <os>
  <type arch='x86_64' machine='pc-1.0'>hvm</type>
  <boot dev='cdrom'/>
 </os>
 <features>
  <acpi/>
  <apic/>
  <pae/>
 </features>
 <clock offset='utc'/>
 <on_poweroff>destroy</on_poweroff>
 <on_reboot>restart</on_reboot>
 <on_crash>restart</on_crash>
 <devices>
  <emulator>/usr/bin/kvm</emulator>
  <disk type='file' device='disk'>
   <driver name='qemu' type='vmdk'/>
```

```
    <source file='Chemin_vers_le_fichier_Arista-vEOS/test-vEOS-lab-4.14.2F.vmdk'/>
    <target dev='hda' bus='ide'/>
    <alias name='ide0-0-0'/>
    <address type='drive' controller='0' bus='0' unit='0'/>
   </disk>
<disk type='file' device='cdrom'>
   <source file='Chemin_vers_le_fichier_Arista-vEOS/Aboot-veos-serial-2.0.8.iso'/>
   <target dev='hdc'/>
   <readonly/>
 </disk>
   <controller type='ide' index='0'>
    <alias name='ide0'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x1'/>
   </controller>
   <interface type='bridge'>
    <mac address='Adresse_MAC'/>
    <source bridge='Nom_du_Bridge'/>
    <model type='virtio'/>
   </interface>
   <serial type='pty'>
    <source path='/dev/pts/3'/>
    <target port='0'/>
    <alias name='serial0'/>
   </serial>
   <input type='mouse' bus='ps2'/>
   <graphics type='vnc' port='5900' autoport='yes' keymap='en-us'/>

   <video>
    <model type='cirrus' vram='9216' heads='1'/>
    <alias name='video0'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/>
   </video>
   <memballoon model='virtio'>
    <alias name='balloon0'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x06' function='0x0'/>
   </memballoon>
  </devices>
  <seclabel type='dynamic' model='apparmor' relabel='yes'>
  </seclabel>
</domain>
```

Dans le cas du SW0-1

Nom_de_domaine_Commutateur = SW0-1

SW0-1 a 1 interface de management, et 4 interfaces de commutation.
L'interface de management br0 est configurée comme la suite :

```
<interface type='bridge'>
  <mac address='00:00:01:02:03:89'/>
  <source bridge='br0'/>
  <model type='virtio'/>
</interface>
```

L'interface de commutation br11 est configurée comme la suite :

```
<interface type='bridge'>
  <mac address='00:11:01:02:03:89'/>
  <source bridge='br11'/>
  <model type='virtio'/>
</interface>
```

Pour le reste des interfaces de commutation sont configurées comme la suite

```
<interface type='bridge'>
  <mac address='00:xy:01:02:03:89'/>
  <source bridge='brxy'/>
  <model type='virtio'/>
</interface>
```

A II.3.1. Lancer le commutateur

Définir le commutateur

sudo virsh define *chemin_vers_le_fichier_xml*

Démarrer le commutateur

sudo virsh start *Nom_de_domaine_du_commutateur*

Se connecter au commutateur

virsh console *Nom_de_domaine_du_commutateur*
login : admin
localhost> enable
localhost# bash sudo reboot
Créer plusieurs copies de vEOS-lab-4.14.2F.vmdk, pour les différents commutateurs Arista.

A II.3.2. Configurer le commutateur

Configurer l'interface de management

(config)# interface management 1
(config-if)# ip address *Adresse_du_hôte/masque*

Dans le cas de SW0-1 Adresse_du_hôte/masque = 172.16.1.101/16

(config)# copy start run

A II.4. Création des machines virtuelles

Télécharger une image d'une machine Ubuntu.

A II.4.1. Créer le fichier xml

```
<domain type='kvm'>
  <name>Nom_de_domaine_de_ la_machine </name>
  <memory>Taille_Espace_Mémoire</memory>
  <vcpu>Nombre_de_CPU</vcpu>
  <os>
    <type arch='x86_64' machine='pc-0.12'>hvm</type>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <pae/>
  </features>
  <clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <devices>
    <emulator>/usr/bin/kvm</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='qcow2' cache='none'/>
      <source file='Chemin_vers_l'image_de_machine_virtuelle'/>
      <target dev='hdc' bus='ide'/>
      <alias name='ide0-1-0'/>
      <address type='drive' controller='0' bus='1' unit='0'/>
    </disk>
    <controller type='ide' index='0'>
      <alias name='ide0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x1' />
 </controller>
 <controller type='usb' index='0'>
      <alias name='usb0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x2' />
```

```
    </controller>

    <interface type='bridge'>
      <mac address='Adresse_MAC'/>
      <source bridge='Nom_du_Bridge'/>
      <target dev='vnet1'/>
      <model type='virtio'/>
      <alias name='net0'/>
    </interface>

<input type='mouse' bus='ps2'/>
    <graphics type='vnc' port='5901' autoport='yes' keymap='en-us'/>
    <video>
      <model type='cirrus' vram='9216' heads='1'/>
      <alias name='video0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'$
    </video>
    <memballoon model='virtio'>
      <alias name='balloon0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'$
    </memballoon>
  </devices>
  <seclabel type='dynamic' model='apparmor' relabel='yes'>
  </seclabel>
</domain>
```

A II.4.2. Lancer la machine virtuelle

Définir le commutateur

sudo virsh define *chemin_vers_le_fichier_xml*
Démarrer le commutateur

sudo virsh start *Nom_de_domaine_de_ la_machine*

Se connecter au commutateur

ssh *nom_de_session@adresse_IP*

ANNEXE III

DÉPLOIEMENT DE NAGIOS

A III.1. Introduction

Nagios s'identifie comme un outil de supervision de réseaux, permettant d'assurer la visibilité des réseaux et réduire les temps d'arrêt. C'est un des outils de supervision des réseaux les plus populaires dans le monde, avec plus de 1 millions d'utilisateurs. Nagios offre la supervision de toute l'infrastructure réseau, incluant les serveurs, les commutateurs, les routeurs, les applications, etc ... Il donne l'état des éléments du réseau en temps réel, présenté sur une interface personnalisée. Nagios peut s'étendre au fur et à mesure avec l'extension du réseau, grâce à sa capacité de supervision de milliers d'équipements sur des réseaux locaux différents. Les ressources de cette annexe sont Ubuntu Geek (2013), Nagios (2015a), JSquared Consulting Syslog (2014) et Nagios (2015b).

A III.2. Environement

Hôte

Ubuntu Server 12.04 x64 LTS

Nagios et son plugin

Nagios 3.5.1
Plugin 2.03

Téléchargement des fichiers

nagios-3.5.1.tar.gz
nagios-plugins-2.0.3.tar.gz

A III.3. Installation des dépendances

```
sudo apt-get update
sudo apt-get install build-essential
sudo apt-get install libapache2-mod-php5
sudo apt-get install apache2
sudo apt-get install libgd2-xpm-dev
```

Démarrage du serveur apache

```
service apache2 start
```

Lancement du serveur apache d'un navigateur web

http://*adresse_IP_du_serveur*

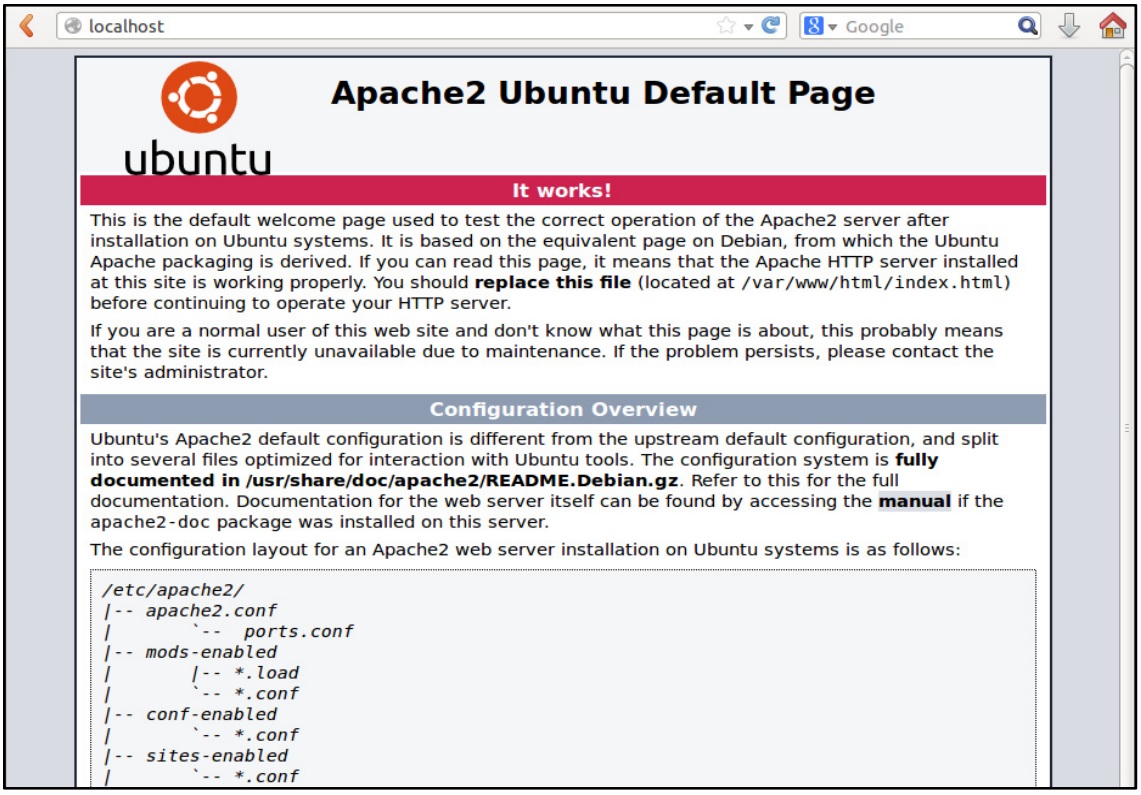


Figure-A III-1 Page d'accueil Apache

A III.4. Création d'un compte utilisateur Nagios

sudo useradd nagios
sudo groupadd nagios

A III.5. Compilation et installation de Nagios

tar -zxvf nagios-3.5.1.tar.gz
cd nagios/
./configure
make all
sudo make install
sudo make install-init
sudo make install-commandmode
sudo make install-config

A III.6. Configuration de l'interface web

sudo mkdir /etc/httpd
sudo mkdir /etc/httpd/conf.d

```
sudo mkdir /etc/httpd/conf.d/nagios.conf
sudo make install-webconf
```

Création d'un login et d'un mot de passe pour l'accès a l'interface web de Nagios

```
sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
```

Entrée et confirmation le mot de passe

A III.7. Compilation et installation du plugin Nagios

```
tar -zxvf nagios-plugins-2.0.3.tar.gz
cd nagios-plugins-2.0.3/
./configure --with-nagios-user=nagios --with-nagios-group=nagios
make
sudo make install
```

A III.7.1. Lancement de Nagios

Ajout du chemin du fichier de configuration de Nagios dans celui de apache

Fichier de configuration de apache : /etc/apache2/sites-enabled/000-default.conf
Fichier de configuration de nagios : /etc/httpd/conf.d/nagios.conf

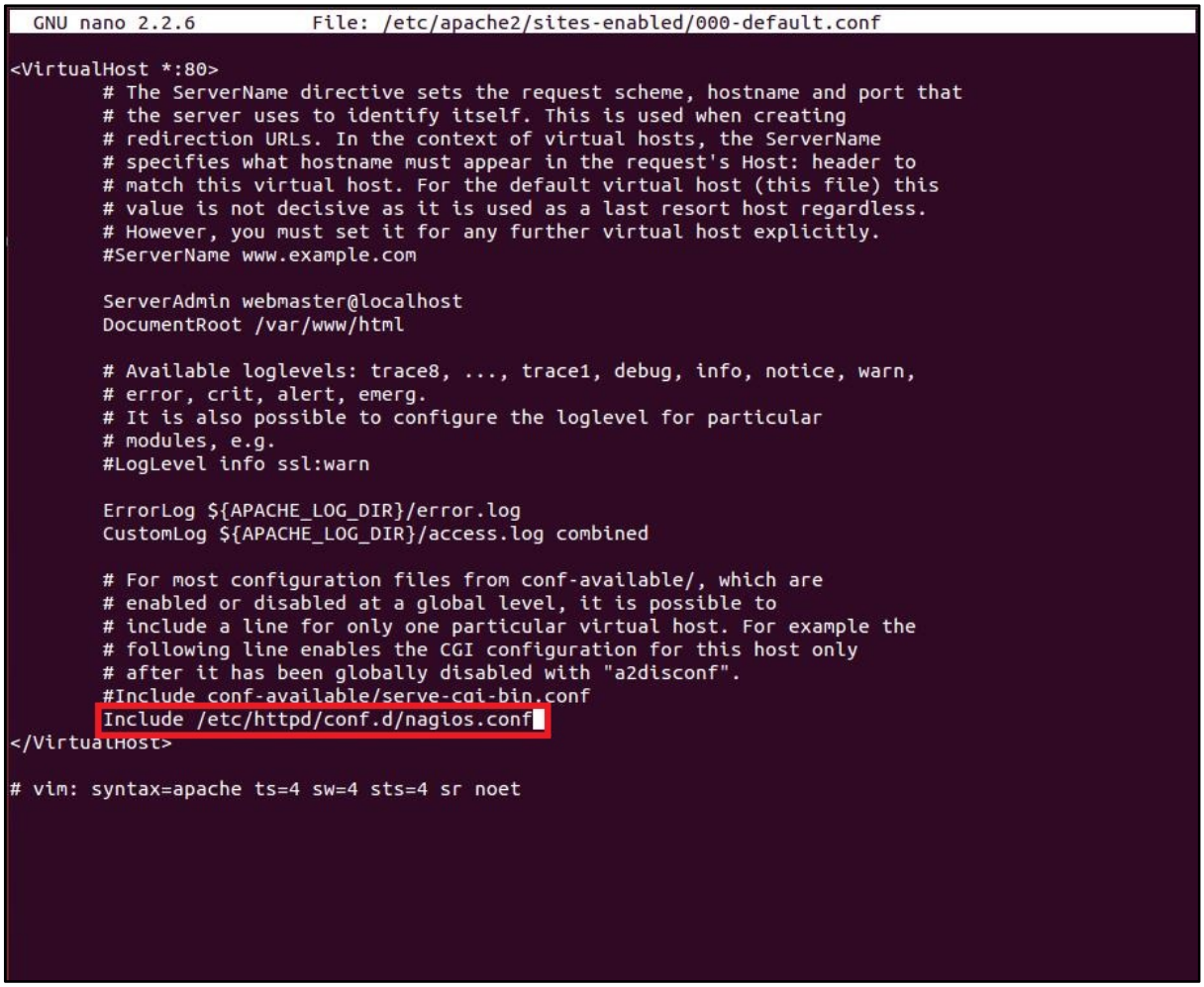


Figure-A III-2 Fichier de configuration d'Apache

A III.7.2. Vérification que Nagios est bien installé, et sans erreurs

sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

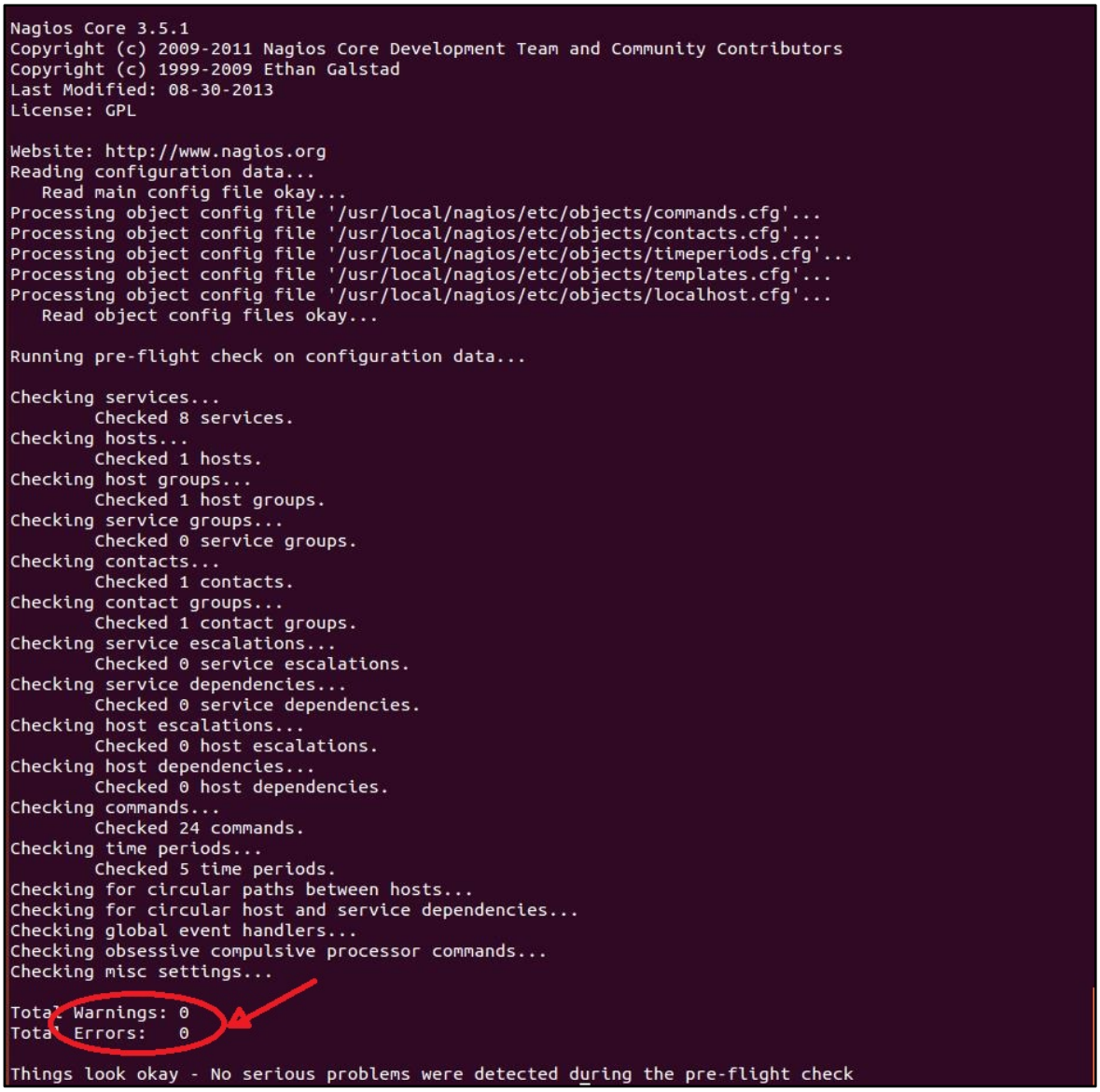


Figure-A III-3 Option v de Nagios

A III.8. Lancement des services

Redémarrage de apache

sudo service apache2 restart

Démarrage de Nagios

sudo service nagios start

Lancement de Nagios à partir du navigateur web

http://*adresse_IP_du_serveur*/nagios/

Connexion à Nagios

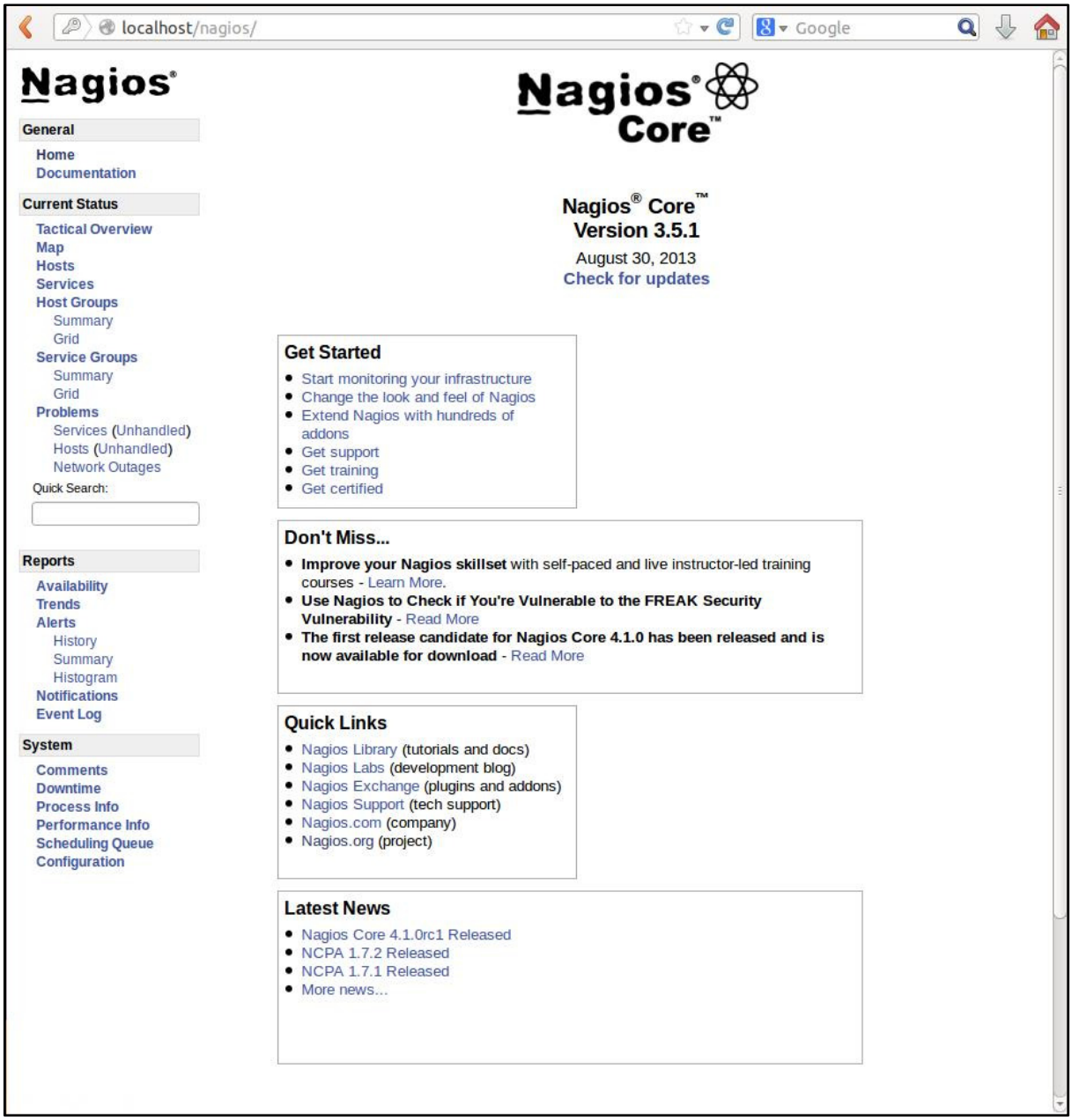


Figure-A III-4 Page d'acceuil de Nagios

ANNEXE IV

CONFIGURATION DE L'ENVIRONEMENT POUR LA SUPERVISION DE LA PERFORMANCE

Les ressources de cette annexe sont Nagios (2015c), System Administrator Recipes (2013), WISC (2015) et IEIF (2015).

A IV.1. SNMP sur la station de monitoring

A IV.1.1. Installation du package SNMP

```
sudo apt-get install snmp snmpd
sudo service snmpd start
```

A IV.1.2. Vérification du plugin SNMP de Nagios

Les plugins nagios doivent être installés après l'installation du SNMP. Sinon, il faut suivre les étapes suivantes :

Réinstallation des plugins Nagios. Ils seront installés dans "/usr/lib/nagios/plugins"

Les premiers plugins sont installés dans "/usr/local/nagios/libexec"

Copie du plugin SNMP de "/usr/lib/nagios/plugins" à "/usr/local/nagios/libexec"

cp /usr/lib/nagios/plugins/check_snmp /usr/local/nagios/libexec/check_snmp

sudo chown nagios:nagios /usr/local/nagios/libexec/check_snmp

A IV.2. SNMP sur les commutateurs

SNMP est déjà onstallé sur les commutateurs Arista. Pour activer SNMPv2 sur un commutateur Arista, il faut créer une premiére community :

(config)#snmp-server community public

Le status de SNMP peut être afficher avec

(config)#show snmp

A IV.3. Configuration de Nagios

A IV.3.1. Fichier de configuration de Nagios

Modification du fichier de configuration de Nagios pour ajouter le chemin vers le fichier de configuration des commutateurs : /usr/local/nagios/etc/nagios.cfg

En enlevant le signe de commentaire (#) de la ligne suivante:

#cfg_file=/usr/local/nagios/etc/objects/switch.cfg

A IV.3.2. Fichier de configuration des commutateurs

Le ficher de configuration des commutateurs est le fichier : /usr/local/nagios/etc/objects/switch.cfg

A IV.3.2.1. Définition du commutateur à superviser

```
define host{
        use             generic-switch
        ; Hérite des valeurs par défaut depuis un template
        host_name       sw0-2               ; Nom du commutateur
        alias           Leaf Level Switch   ; Nom plus complet du commutateur
        address         172.16.1.101        ; Adresse IP du commutateur
        hostgroups      switches            ; Groupe auquel le commutateur est associé
        }
```

A IV.3.2.2. Définition de services à superviser par ICMP

La superviser du taux de perte de paquets et du délai de latence se fait avec ICMP.

```
define service{
        use                             generic-service
        host_name                       sw0-2
        service_description             PING
        check_command                   check_ping!200.0,20%!600.0,60%
        ; La commande utilisée pour superviser le service
        normal_check_interval           5
        ; Vérification toutes les 5 minutes dans des conditions normales
        retry_check_interval            1
        ; Revérifier le service toutes les minutes jusqu'à la stabilisation
        }
```

A IV.3.2.3. Définition de services à superviser par SNMP

Exemple : temps d'opération

```
define service{
        use                     generic-service
        host_name               sw0-2
        service_description     Uptime
        check_command           check_snmp!-C public -o sysUpTime.0
        }
```

Sur la ligne check_command, il faut préciser le nom de la variable du MIB, dans ce cas sysUpTime.0. Ou bien son Identifiant d'objet OID; dans ce cas sysUpTime.0 peut être remplacé par le OID 1.3.6.1.2.1.1.3.0.

L'option -C public precise le nom de la community.

A IV.3.2.4. Table des OID

Tableau-A IV-1 Les OID en fonction des MIB

OID	Variable MIB	Description
.1.3.6.1.2.1.1.3.0	sysUpTime	Temps d'opération
.1.3.6.1.2.1.2.2.1.10.*numéro_de_port*	ifInOctets	Nombre d'octets reçu sur l'interface
.1.3.6.1.2.1.2.2.1.16.*numéro_de_port*	ifOutOctets	Nombre d'octets transmis de l'interface
.1.3.6.1.2.1.2.2.1.5.*numéro_de_port*	ifSpeed	Débit sur l'interface en bits/secondes

A IV.4. Vérifications

Après toutes modifications des fichiers de Nagios, il faut le redémarrer :

sudo service nagios restart

Avant de redémarrer Nagios, il faut vérifier s'il n'y a pas d'erreurs dans la configuration de nagios avec la commande :

sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

Pour vérifier manualement si snmp marche entre la station de monitoring et l'hôte distant :

snmpget -v2c -c public *adresse_IP_du_hôte_distant* 1.3.6.1.2.1.1.3.0

Les services à superviser seront affichés sur l'interface de Nagios comme suite :

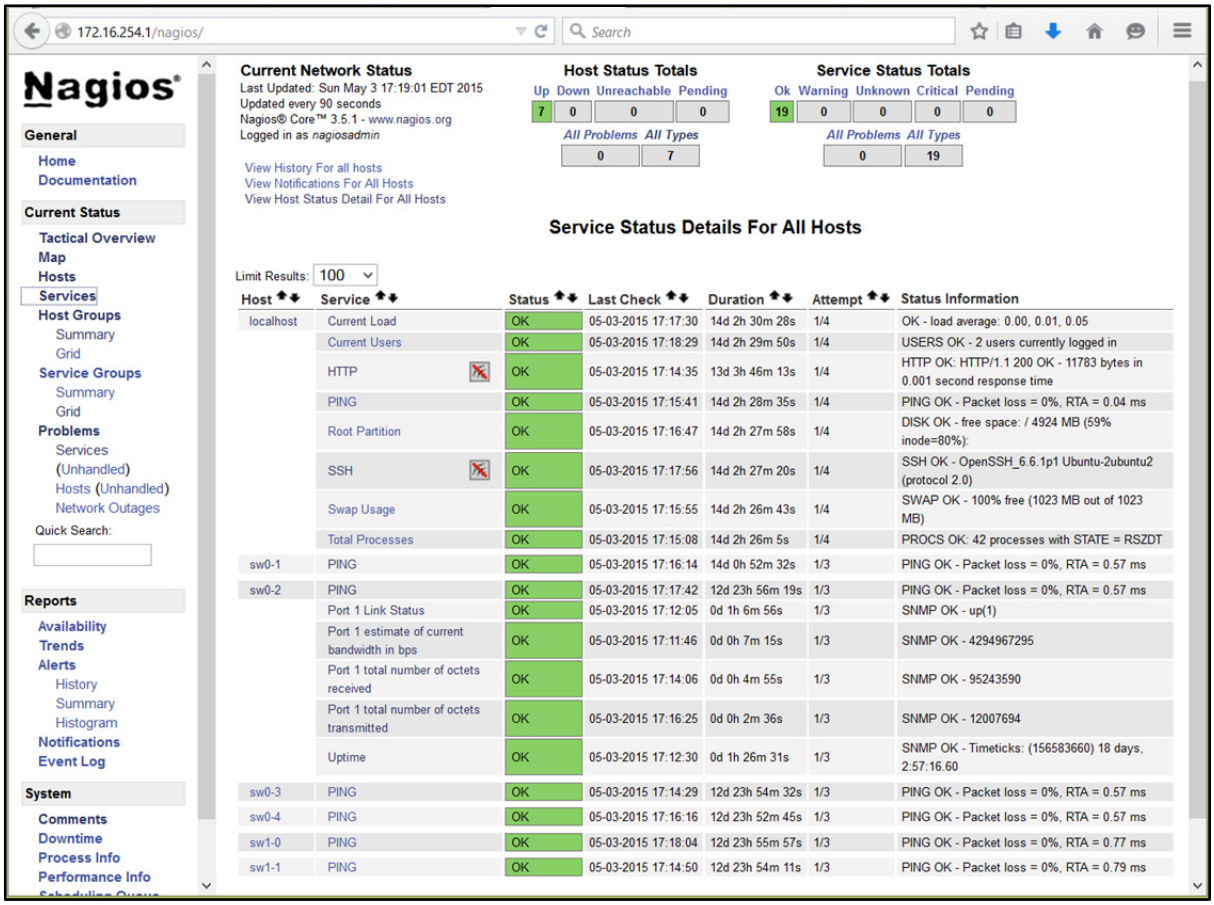


Figure-A IV-1 Les services configurés dans Nagios

ANNEXE V

PLUGIN MRTG POUR NAGIOS

A V.1. Introduction

MRTG pour Multi Router Traffic Grapher, est un plugin de Nagios, pour la supervision du trafic entrant et sortant des nœuds du réseau. MRTG opère à base de SNMP, qui dit être supporté par les nœuds du réseau, comme les commutateurs. La ressource de cette annexe est Iceflatline (2009).

A V.2. Déploiement

A V.2.1. Environnement

Ubuntu Server 12.04 x64 LTS

A V.2.2. Installation

```
sudo apt-get update
sudo apt-get install mrtg
sudo mkdir /etc/mrtg && sudo mv /etc/mrtg.cfg /etc/mrtg
```

A V.2.3. Configuration

MRTG inclue un script, dit cfgmaker, qui collecte les données des équipements désignés, afin de peopler le fichier de configuration */etc/mrtg/mrtg.cfg*

```
sudo cfgmaker --global 'WorkDir:/var/www/mrtg' --global 'Options[_]: bits,growright' --output /etc/mrtg/mrtg.cfg public@adresse_IP_du_premier_commutateur public@adresse_IP_du_deuxième_commutateur …
```

Pour l'activation de MRTG en mode daemon, et pour l'établissement de l'interval de mise à jour des données, il faut ajouter les deux lignes suivantes dans le fichier de configuration */etc/mrtg/mrtg.cfg*

```
RunAsDaemon: Yes
Interval: 5
```

A V.2.4. Création de l'interface web

A V.2.4.1. Création du fichier index

```
sudo mkdir /var/www/mrtg
sudo indexmaker --output /var/www/mrtg/index.html /etc/mrtg/mrtg.cfg
```

Sur le fichier de configuration de Apache, */etc/apache2/apache.conf,* ajouter la configuration de MRTG

```
Alias /mrtg "/var/www/mrtg/"
<Directory "/var/www/mrtg/">
        Options None
        AllowOverride None
        Require all granted
</Directory>
```

A V.2.4.2. Redémarrage de Apache

```
sudo service apache2 restart
```

A V.2.4.3. Démarrage de MRTG

```
sudo env LANG=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --logging /var/log/mrtg.log
```

A V.2.4.4. Lancement de l'interface web

http://*adresse_IP_du_serveur*/mrtg/

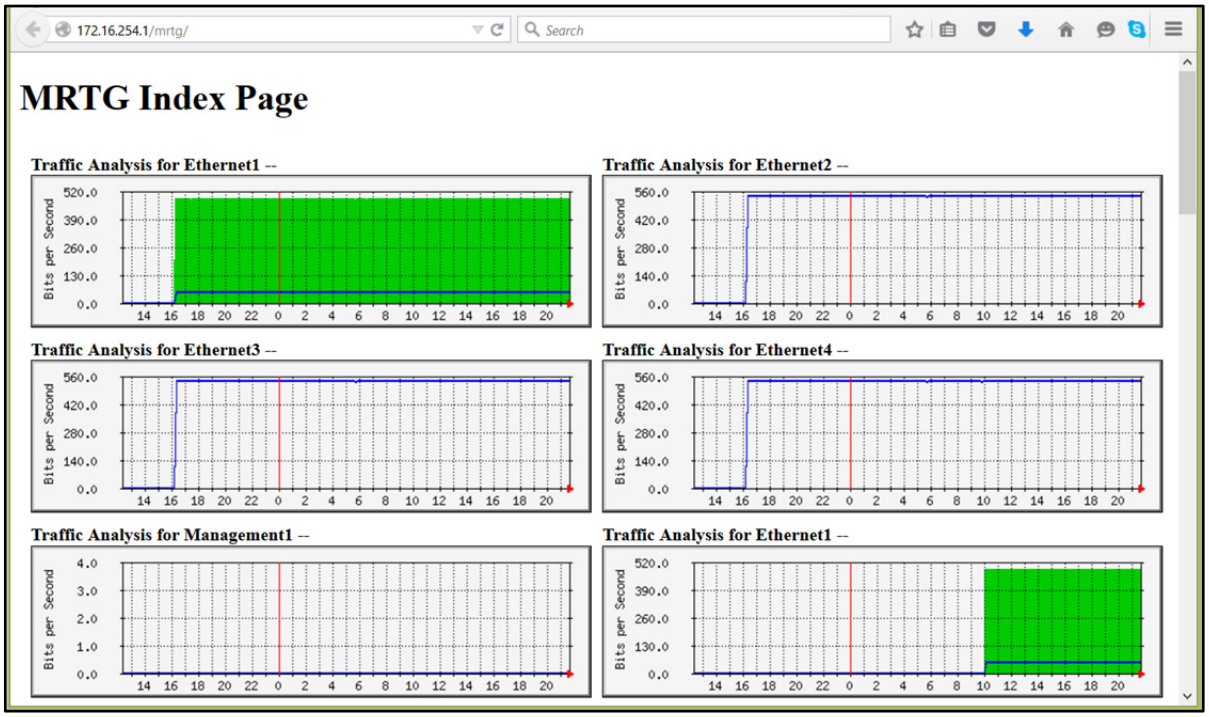


Figure-A V-1 Le plugin MRTG lancé

A V.3. Collecte des données MRTG avec Nagios

Dans le fichier de configuration des services ajouter la configuration MRTG, comme ce qui suit :

```
define service{
        use                     generic-service
        host_name               sw0-2
        service_description     Port 1 Traffic
        check_command
        check_local_mrtgtraf!/var/www/mrtg/172.16.1.102_ethernet1.log
                                !AVG!1000000,1000000!5000000,5000000!10
                                ; indiquer le chemin du fichier log
        }
```

Les fichiers log se trouvent dans le répertoire /var/www/mrtg/

ANNEXE VI

ENREGISTREMENT DES DONNÉES NAGIOS DANS UNE BASE DE DONNÉES

A VI.1. Introduction

L'outil NDOUtils est un modèle qui permet de lire les données depuis Nagios et les enregistrer dans une base de données MySQL. Mais, en déployant NDOUtils, un problème à fait face : NDOUtils demande de grandes ressources physiques, et influence la performance du service web de Nagios. Cette anomalie est traitée dans l'article (6).

A VI.2. Développement

Comme solution, un script a été développé afin de collecter les données depuis Nagios, et les enregistrer dans une base de données.

A VI.2.1. Création de base de données

La base de données est à une table unique dont la description est la suivante :

```
mysql> describe data_collection;
+-----------+-------------+------+-----+---------+----------------+
| Field     | Type        | Null | Key | Default | Extra          |
+-----------+-------------+------+-----+---------+----------------+
| id        | int(15)     | NO   | PRI | NULL    | auto_increment |
| host      | varchar(30) | YES  |     | NULL    |                |
| metric    | varchar(30) | YES  |     | NULL    |                |
| timestamp | datetime    | YES  |     | NULL    |                |
| value     | double      | YES  |     | NULL    |                |
+-----------+-------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

Figure-A VI-1 Table de la base de données de Nagios

"id" est la clé primaire de la table, "host" contient l'identifiant du hôte supervisé, "metric" la métrique de la performance, "timestamp" l'horodate de la prise de la mesure de la métrique et "value" la valeur mesurée de la métrique.

A VI.2.2. Peuplement de la base de données

Un script python a été développé pour peupler la base de données. Le script est divisé en deux parties. La première partie est d'intercepter le flux entre Nagios et son interface web d'affichage. Cette première partie est illustrée dans la figure suivante.

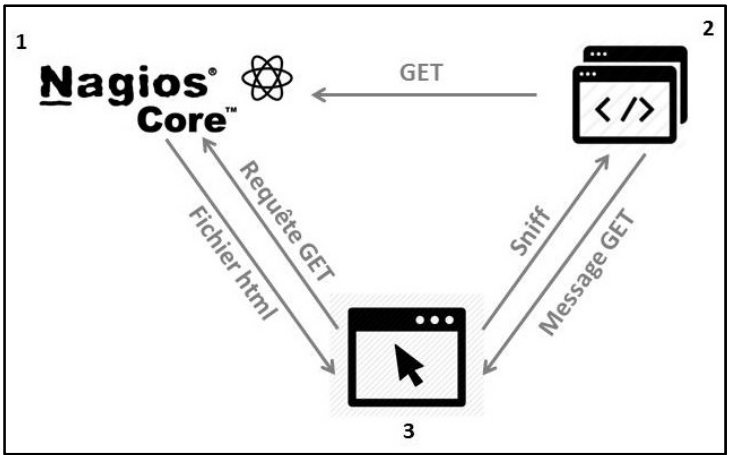


Figure-A VI-2 Interception des requettes HTTP de Nagios

1- lancer les deux requêtes GET pour télécharger les pages html d'affichage des hôtes et des services de Nagios.
2- sniffer les messages GET pour avoir les adresses des fichiers html.
3- en utilisant les adresse sniffées, télécharger les fichiers html dans l'environnement de travail.

Après avoir téléchargé les fichiers html, la deuxième partie du script parcourt le fichier html, collecte les données de la performance et l'enregistre dans la base données. La figure suivante illustre cette deuxième partie du script.

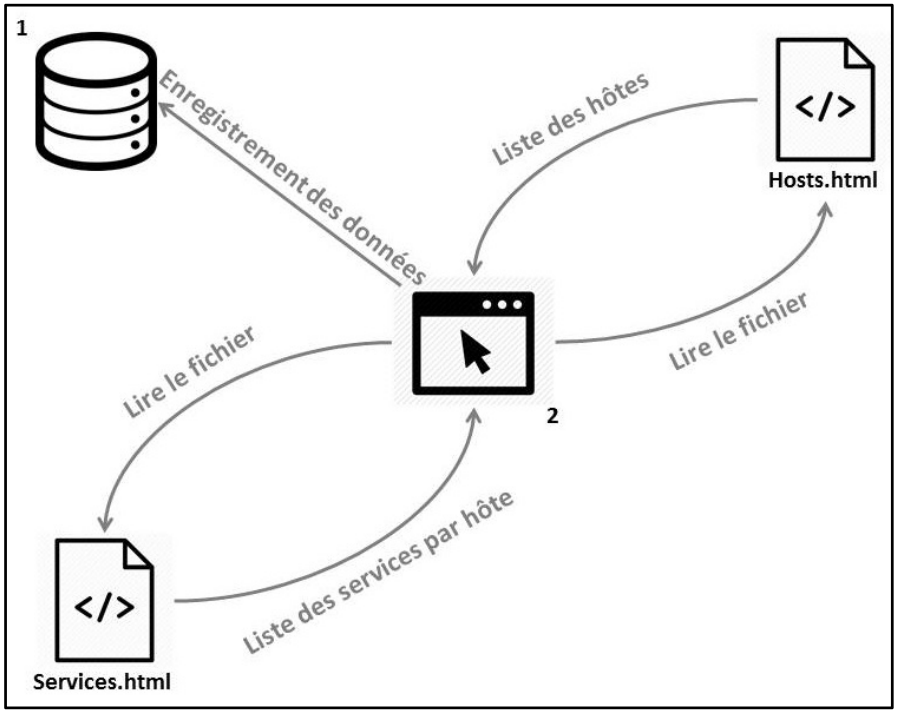


Figure-A VI-3 Collection des données depuis Nagios

1- lire le premier fichier contenant la liste des hôtes supervisés par Nagios;
2- pour chaque hôte obtenu, lire le fichier des services pour avoir la liste des services par hôte. Un service contient le nom de la métrique, son horodate et sa valeur;
3- enregistrer les données collectées dans la base de données.

Ce processus est automatisé, afin de s'exécuter toutes les minutes, et puis avoir un peuplement permanant de la base de données. La table de la base de données ressemble à ce qui suit.

id	host	metric	timestamp	value
6	sw0-1	Port+1+ifInOctets	2015-05-29 19:01:30	2734607
7	sw0-1	Port+1+ifOutOctets	2015-05-29 19:00:30	254741083
8	sw0-1	Port+1+ifHighSpeed	2015-05-29 19:01:30	1000
9	sw0-1	Port+2+ifInOctets	2015-05-29 19:00:30	15567
10	sw0-1	Port+2+ifOutOctets	2015-05-29 19:00:30	255508299
11	sw0-1	Port+2+ifHighSpeed	2015-05-29 19:06:30	1000
12	sw0-2	Port+1+ifInOctets	2015-05-29 19:03:30	232367803
13	sw0-2	Port+1+ifOutOctets	2015-05-29 19:00:30	25153344
14	sw0-2	Port+1+ifHighSpeed	2015-05-29 18:53:30	1000

Figure-A VI-4 Capture de la table des données de Nagios

ANNEXE VII

TEMPS DE LATENCE AVEC PING

Options de Ping

-s : taille de paquet en octets
-i : nombre de paquet à envoyer dans une période
-c : période d'envoi de paquets en secondes

La commande pour envoyer des Ping périodique d'une taille de 1500 octets est :

ping –s *taille_paquets* –i *nombre_paquets* –c *période adresse_IP_destination*

Le temps de latence égale à la moitié du temps ping.

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUE

Al-Fares, Mohammad, Loukissas, Alexander et Vahdat, Amin. 2008. « A scalable, commodity data center network architecture ». ACM SIGCOMM, p. 63 – 74.

Angrisani, leopoldo et autres. 2006. « Measurement of processing and queing delays introduced by open-source router in single-hop network ». IEEE Transactions on instrumentation and measurement, vol. 55, n° 4, p. 1065 – 1076.

Arista Networks. 2015. « Arista User Manual ». User guide.

Bilal, Kashif et autres. 2012. « A comparative study of data center network architectures ». ECMS.

Boulos, Fadi et autres. 2009. « Évaluation subjective des effets de pertes de paquets sur des vidéos codées en H.264/AVC ». Coresa, p. 66 – 69.

Chen, Thomas M. et Hu, Lucia. 2002. « Internet performance monitoring ». IEEE, vol. 90, n° 9, p. 1592 – 1603.

Ciena. 2014. « V-WAN solution application guide ». User guide.

Cisco systems. 2007a. « Cisco data center infrastructure 2.5 design guide; Chapitre 1 : Data center architecture overview ».

Cisco systems. 2007b. « Cisco data center infrastructure 2.5 design guide; Chapitre 2 : Data center multi-tier model design ».

Ciuffoletti, Augusto. 2010. « Monitoring a virtual network infrastructure ». ACM SIGCOMM, vol. 40, n° 5, p. 47 – 52.

Clayman, Stuart, Galis, Alex et Mamatas, Lefteris. 2010, « Monitoring virtual networks with Lattice ». IEEE/IFIP Network Operations and Management Symposium Workshops, p. 239 – 246.

Constantinescu, Doru et Popescu, Adrian. 2006. « Modeling of one-way transit time in IP routers ». IEEE AICT-ICIW, p. 16 – 25.

DORSAL. 2012. « CPTD ». En ligne. < www.lttng.org >. Consulté le 1er mai 2013.

Eli. 2012. « Virtualization ». En ligne. < www.elithecomputerguy.com >. Consulté le 1er mai 2013.

Extreme Networks. 2013. « ExtremeXOS Command Reference Guide for Release 15.4 ». User guide.

Fabienne, Anhalt. 2013. « Network virtualization : performance, sharing and applications ». Thèse de doctorat. École normale supérieure de lyon.

Greenberg, Albert, Hamilton, James R. et Jain, Navendu. 2009. « VL2 : ascalable and flexible data center network ». ACM SIGCOMM, p. 51 – 62.

GroundWork. 2007. « GroundWork Monitor Architecture Overview ». En ligne. < www.socallinuxexpo.org/scale5x/presentations/thomas.pdf >. Consulté le 27 avril 2015.

GroundWork. 2015. « GroundWork ». En ligne. < www.gwos.com >. Consulté en février 2015.

Gu, Qianping. 2008. « Delay analysis ». CMPT, p. 1 – 4.

Guo, Chuanxiong. 2008. « DCell : a scalabe and fault-tolerant network structure for data centers ». ACM SIGCOMM, p. 75 – 86.

Guo, Chuanxiong. 2009. « BCube : a high performance, server-centricnetwork architecture for modular data centers ». ACM SIGCOMM, p. 63 – 74.

Hammadi, Ali et Mhamdi, Lotfi. 2014. « A survey on architecture and energy efficiency in data center networks ». Elsevier Computer Communications, p. 1 – 21.

Hyperic. 2015. « Hyperic ». En ligne. < www.vmware.com/ca/fr/products/vrealize-hyperic >. Consulté en février 2015.

Hohn, N. et autres. 2004. « Bridging router performance and queuing theory ». ACM SIGMETRICS, p. 335 – 366.

Iceflatline. 2009. « How to install and configure MRTG ». En ligne. < www.iceflatline.com/ 2009/08/how-to-install-and-configure-mrtg-on-ubuntu-server >. Consulté en mars 2015.

IEIF. 2015. « ObjectID ifEntry ». En ligne. < www.alvestrand.no/objectid/1.3.6.1.2.1.2.2.1 .html >. Consulté en mars 2015.

Issariyapat, Chavee et autres. 2012. « Using Nagios as a Groundwork for developing a better network monitoring system ». IEEE Proceedings of PICMET, p. 2771 – 2777.

IST-Autoi. 2015. « autonomic internet (AutoI) ». En ligne. < ist-autoi.eu >. Consulté le 5 avril 2015.

JSquared Consulting Syslog. 2014. « installing Nagios on Ubuntu ». En ligne. < blog.jsquaredconsulting.com/2014/05/installing-nagios-4-on-ubuntu-1404.html >. Consulté en mars 2015.

Juniper networks. 2013. « The QFabric architecture : implementing a flat data center network ». Juniper networks white paper.

KVM. 2013. « KVM ». En ligne. < www.linux-kvm.org >. Consulté en 2013 – 2015.

Li, Dan et autres. 2009. « FiConn : using backup port server interconnection in data centers ». IEEE INFOCOM, p. 2276 – 2285.

Marik, Ondrej et Zitta, Stanislav. 2014. « Comparative analysis of monitoring system for data networks ». IEEE ICMCS, p. 563 – 568.

Martinovic, Goran, Balen, Josip et Rimac-Drlje, Snjezana. 2013. « Impact of the Host Operating Systems on Virtual Machine Performance ». IEEE MIPRO, p. 613 – 618.

Matta, Johnny M. et Takeshita, Atsushi. 2002. « End-to-end voice over IP quality of service estimation through router queuing delay monitoring ». IEEE GLOBECOM, vol. 3, p. 2458 – 2462.

Modern Course. 2015. « A comparison between zabbix, nagios, zenoss, cacti, and cricket ». En ligne. < www.moderncourse.com/use-zabbix-comparison-zabbix-nagios-zenoss-cacti-cricket >. Consulté le 27 avril 2015.

Mongkolluksamee, Sophon, Pongpaibool, Panita et Issariyapat, Chavee. 2010. « Strengths and limitations of Nagios as network monitoring solution ». IEEE JCSSE, Vol. 1, p. 96 – 101.

MySQL. 2013. « MySQL ». En ligne. < www.mysql.fr >. Consulté en 2013 – 2015.

Nagios. 2014. « Nagios ». En ligne. < www.nagios.org >. Consulté en 2014 – 2015.

Nagios. 2015a. « Nagios quick start ». En ligne. < nagios.sourceforge.net/docs/3_0/quick start-ubuntu.html >. Consulté en mars 2015.

Nagios. 2015b. « Nagios Support ». En ligne. < support.nagios.com/knowledgebase/faqs/ index.php?id=52&catid=35&faq_id=21 >. Consulté en mars 2015.

Nagios. 2015c. « Monitoring switches and routers ». En ligne. < nagios.sourceforge.net/docs/ 3_0/monitoring-routers.html >. Consulté en mars 2015.

Nagios. 2015d. « NDOUtils ». En ligne. < exchange.nagios.org/directory/Addons/Database-Backends/NDOUtils/details >. Consulté le 25 mars 2015.

Nagios. 2015e. « Nmap2Nagios ». En ligne. < exchange.nagios.org/directory/Addons/Configuration/Auto-2DDiscovery/nmap2nagios-2Dng/details >. Consulté le 4 avril 2015.

NagiosQL. 2015. « NagiosQL ». En ligne. < www.nagiosql.org >. Consulté le 4 avril 2015.

OpenNMS. 2015a. « OpenNMS ». En ligne. < www.opennms.org >. Consulté en février 2015.

OpenNMS. 2015b. « Comparison with other network management systems ». En ligne. < www.opennms.org/wiki/Comparison_with_other_network_management_systems >. Consulté le 27 avril 2015.

Papagiannaki, Konstantina, Cruz, Rene et Diot, Christophe. 2003a. « Network performance monitoring at small time scales ». ACM IMC, p. 295 – 300.

Papagiannaki, Konstantina, et autres. 2003b. « Measurement and analysis of single-hop delay on an IP backbone network ». IEEE Journal on selected areas in communications, vol. 21, n° 6, p. 908 – 921.

Parreira, Bruno et autres. 2012. « Network virtualization – a virtual router performance evaluation ». CRC Portugal. P. 1 – 7.

Salehin, Khodaker M. et Rojas-Cessa, Roberto. 2013. « Measurement of packet processing time of an internet host using asynchronous packet capture at the data link layer ». IEEE Communication QoS, Reliability, and Modeling Symposium, p. 2550 – 2554.

Salehin, Khodaker M. et Rojas-Cessa, Roberto. 2014. « Scheme for measuring queueing delay of a router using prob-gap model: the single-hop case ». IEEE Communications Letters, vol. 18, p. 696 – 699.

Suciu, George et autres. 2014. « Network management and monitoring for cloud systems». IEEE ECAI, p 1 – 4.

Sun, Yantao et autres. 2013. « Data center Network Architecture ». ZTE communications, vol. 11, n° 1, p. 54 – 60.

Sundling, Mattias. 2010. « Maximizmg Virtual Machine performance ». Quest Software White Paper.

System Administrator Recipes. 2013. « Nagios returns code 127 ». En ligne. < systemadministratorrecipes.blogspot.ca/2013/01/nagios-return-code-of-127-is-out-of.html >. Consulté en mars 2015.

Ubuntu Geek. 2013. « How to install Nagios in Ubuntu server ». En ligne. < www.ubuntugeek.com/how-to-install-nagios-4-0-1-monitoring-tool-in-ubuntu-13-10-server-saucy-salamander.html >. Consulté en mars 2015.

VMware. 2013. « VMware ». En ligne. < www.vmware.com >. Consulté le 16 octobre 2013.

Wiki monitoring. 2013. « OpenNMS ». En ligne. < wiki.monitoring-fr.org/opennms/start >. Consulté le 27 avril 2015.

WISC. 2015. « System Monitoring via Nagios and SNMP ». En ligne. < www.math.wisc.edu /~jheim/snmp/ >. Consulté en mars 2015.

Xen Project. 2013. « Xen Project ». En ligne. < www.xenproject.org >. Consulté le 16 octobre 2013.

Yu, Minlan et autres. 2011. « Profiling network performance for multi-tier data center applications ». Networked systems design and implementation, p. 57 – 70.

Zenoss. 2015. « Zenoss ». En ligne. < www.zenoss.com >. Consulté en février 2015.

Zhang, Yueping, Su, Ao-Jan et Jiang, Guofei. 2010. « Evaluating the impact of data center network architectures on application performance in virtualized envirements ». IEEE IWQoS, p. 1 – 5.

Zhu, Yuhao, Deng, Yangdong et Chen, Yubei. 2011. « Hermes : an integrated CPU/GPU 2microarchitecture for IP routing ». ACM DAC, p. 1044 – 1049.