TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ALGORITHMES

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

API	Application Programming Interface
ATM	Asynchronous Transfer Mode
CABO	Concurrent Architectures are Better than One
CPU	Central Processing Unit
Emulab	Emulation Laboratory
FAI	Fournisseur d'Accès à Internet
FIP	Fournisseur d'Infrastructure Physique
FIV	Fournisseur d'Infrastructure virtuelle
FS	Fournisseur de Services
GENI	Global Environment for Network Innovations
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISP	Internet Service Provider
LAN	Local Area Network
MAC	Media Access Control
MPLS	Multi-Protocol Label Switching
MVNO	Mobile Virtual Network Operator
OTN	Optical Transport Network
PIP	Physical Infrastructure Provider
PlanetLab	Planet Network Laboratoty
QoS	Quality of Service
SLA	Service Level Agreement
SP	Service Provider
TDM	Time-division multiplexing
TINA	Telecommunications Networking Information Architecture
VIOLIN	Virtual internetworking on overlay infrastructure,"
VIP	Virtual Infrastructure Provider
VLAN	Virtual Local Area Network
VPN	Virtual Private Network

INTRODUCTION

En trente ans d'existence, l'Internet a connu un succès flamboyant, il est devenu un outil presque indispensable dans la vie de tous les jours. Cependant, à cause de son environnement multifournisseur, l'introduction de nouvelles technologies au réseau internet ou la modification de ses technologies existantes nécessite un consensus entre ses différents intervenants. Seules des modifications mineures ou des mises à jour simples sont alors tolérées. Ce problème du réseau internet appelé par (Turner et Taylor, 2005) ossification est le facteur principal du ralentissement de son évolution. Par exemple, ce problème d'ossification est l'un des facteurs majeurs qui retardent le déploiement de la technologie IPv6.

Une des solutions proposées par les chercheurs à ce problème d'ossification est la virtualisation des réseaux. C'est un concept permettant de créer plusieurs réseaux virtuels sur une infrastructure physique partagée. Il consiste à ajouter une couche d'abstraction entre les utilisateurs et les ressources physiques tout en donnant l'illusion d'une interaction directe avec ces dernières. Plusieurs formes de virtualisation existent déjà telles que : la virtualisation des systèmes d'exploitation (Ex. : machine virtuelle), grille informatique (Ex. : infonuagique), et virtualisation des liens (Ex. : la technologie MPLS). La conception de routeurs virtuels par les manufacturiers d'équipements de réseau témoigne de l'importance de ce concept pour assurer l'évolution et la viabilité des réseaux de communications. En effet, la possibilité de créer plusieurs instances logiques sur un même routeur physique permet aux entreprises d'éliminer les coûts importants résultants de l'achat de routeurs physiques. Aussi, cette technologie offre aux entreprises la possibilité de répliquer facilement les routeurs pour des fins de redondances ou pour ajouter de la capacité.

La virtualisation des réseaux est un concept qui tire ses origines des recherches récentes sur l'Internet du futur. C'est la création de plusieurs réseaux virtuels logiques qui peuvent être construits selon les besoins des usagers. Par exemple, une start-up qui opère dans le secteur de la vidéo à la demande et qui n'a pas assez de moyens pour déployer sa propre

infrastructure réseau peut opter pour la solution de virtualisation de réseau. En effet, elle peut louer les ressources d'un fournisseur qui possède une installation physique. L'ensemble des ressources louées par la start-up constitue alors son réseau virtuel. La différence entre ce nouveau modèle et celui du réseau internet traditionnel est la présence de trois entités indépendantes : les fournisseurs d'infrastructure physique (FIP), qui gèrent le réseau physique, les fournisseurs de services (FS) qui offrent des services aux clients et les fournisseurs d'infrastructure virtuelle (FIV) qui construisent des réseaux virtuels en utilisant les ressources offertes par les différents FIP. En effet, ces entités ont remplacé le rôle des fournisseurs d'accès à internet (FAI) dans le modèle traditionnel.

Les motivations derrière ce concept sont nombreuses : partage rentable des ressources, solutions de réseautage personnalisé et convergence des infrastructures réseau existantes. D'une part, la virtualisation de réseau peut être utilisée pour partager les ressources d'une infrastructure existante en sous-réseaux et de les associer à différents réseaux virtuels gérés par divers fournisseurs de services. Ce partage permet de réduire le coût d'entrée dans des marchés pour les fournisseurs de services n'ayant pas les ressources suffisantes pour déployer des infrastructures physiques coûteuses. Par exemple, dans le marché du mobile, des opérateurs mobiles virtuels (MVNO) peuvent louer des services réseau aux opérateurs qui détiennent les infrastructures physiques. D'autre part, la couche d'abstraction introduite par la virtualisation représente un moyen pour introduire de nouvelles technologies (Ex. : nouveaux modèles de QoS) aux infrastructures existantes facilitant ainsi l'évolution des réseaux et personnalisant les protocoles de ces infrastructures afin d'adapter le réseau à des services spécifiques. De plus, elle représente une couche unificatrice au-dessus des différentes infrastructures réseau facilitant l'interconnectivité et la convergence des réseaux en réduisant la complexité des passerelles d'interconnectivité et des accords entre les différents opérateurs.

Problématique

La virtualisation des réseaux doit relever plusieurs défis. Elle doit permettre une isolation complète, logique et physique, entre les réseaux virtuels coexistants. Elle doit aussi permettre l'interopérabilité entre ces différents réseaux virtuels. Dans un environnement virtualisé, l'utilisation des ressources du réseau physique doit être optimisée en appliquant des algorithmes d'allocation de ressources efficaces (Bo et al., 2010). Ces algorithmes visent à utiliser les ressources virtuelles disponibles pour satisfaire la demande des utilisateurs. Plusieurs approches de sélection et d'allocation de ressources ont été proposées dans des recherches antérieures. Cependant, certaines approches (Minlan Yu, 2008) utilisant une entité centrale pour la création des réseaux virtuels s'avèrent très lentes en termes de temps d'exécution. En effet, l'unité centrale doit maintenir à jour les informations concernant tout le réseau physique, mais quand ce dernier devient très dynamique (Ex. : Ajout et suppression de nœuds) cela peut retarder la prise de décisions reliées à l'utilisation des ressources. D'autres approches ont proposé de distribuer la tâche de création des réseaux virtuels d'une manière coopérative sur l'ensemble des nœuds du réseau physique. Ces approches s'avèrent très gourmandes en ressources vu le nombre important de messages échangés entre les différents nœuds afin qu'ils puissent concevoir convenablement les réseaux virtuels (Houidi, Louati et Zeghlache, 2008b). De plus, aucune de ces approches ne tient compte de l'impact écologique des ressources sélectionnées. Notre objectif principal sera donc de proposer et de tester une nouvelle approche de sélection et d'allocation de ressources qui pourra pallier les lacunes des différentes approches présentées dans les recherches antérieures. Autrement dit, la solution proposée visera à diminuer les ressources en termes de bande passante et de CPU utilisées lors des processus de sélection et d'allocation de ressources. Par ailleurs, cette solution tient compte du coût d'utilisation des ressources en termes de consommation de CO_2.

Objectifs de la recherche

L'objectif principal de ce projet est de spécifier et de valider une architecture nouvelle et innovatrice pour la gestion efficace des ressources dans les infrastructures de virtualisation de

réseaux. Cette architecture inclut les composants suivants : un modèle d'affaires, des entités fonctionnelles (FIP, FIV et FS) et des algorithmes permettant la sélection et l'allocation efficace des différentes ressources fournies par le réseau physique. Cet objectif principal peut être divisé en plusieurs sous-objectifs qui sont les suivants :

1) analyser l'état de l'art des projets de recherche actuels, notamment les travaux en cours reliés aux réseaux virtuels, aux réseaux locaux et aux réseaux actifs et programmables. De plus, nous analyserons les architectures ainsi que les protocoles et concepts de virtualisation de réseau de communication proposés dans différents travaux de recherche. Par ailleurs, nous étudierons les stratégies d'allocation de réseaux virtuels dans les réseaux de communication ;

2) spécifier et valider les composants architecturaux qui interviennent dans le processus de sélection et d'allocation des ressources ;

3) spécifier et valider les protocoles qui régissent la communication entre ces composants architecturaux ;

4) spécifier et valider des algorithmes d'allocation et de gestion de ressource dans les architectures de virtualisation de réseaux ;

5) comparer la solution proposée à celles proposées par (Minlan Yu, 2008) et (Houidi, Louati et Zeghlache, 2008b).

Méthodologie de travail

Pour atteindre chacun de ces objectifs, nous avons suivi une approche itérative dont les principales étapes sont les suivantes :

- faire une analyse exhaustive des travaux de recherche actuels sur la virtualisation de réseaux ;
- spécifier et valider des modèles d'affaires décrivant les entités impliquées dans la virtualisation des réseaux ainsi que leurs relations/interactions ;
- définir et valider des algorithmes et des protocoles de sélection et d'allocation efficace des ressources fournies par l'infrastructure physique ;
- implémenter les algorithmes proposés par (Minlan Yu, 2008) et (Houidi, Louati et Zeghlache, 2008b) ;
- analyser les performances des algorithmes et protocoles proposés et les comparer avec celles obtenues par (Minlan Yu, 2008) et (Houidi, Louati et Zeghlache, 2008b).

Organisation du rapport

Le reste du rapport est organisé comme suit. Le chapitre 1 présente l'état de l'art de la virtualisation de réseaux. Nous décrierons en premier l'évolution du concept de la virtualisation dans les systèmes informatiques ainsi que dans les réseaux. Ensuite, nous analyserons l'architecture actuelle de la virtualisation du réseau. Nous évoquerons les différentes approches proposées dans la littérature pour la sélection et l'allocation des ressources offertes par l'infrastructure physique. Le chapitre 2 décrit les motivations, les principes et les défis d'une telle architecture. Aussi, nous analyserons le modèle d'affaires adopté pour élaborer notre approche. Puis, nous expliquerons les étapes qui mènent à l'élaboration d'une infrastructure réseau virtuelle. Dans le chapitre 3, nous décrirons en détail les architectures et les algorithmes de l'approche proposée. Dans la première partie du chapitre 4 nous présenterons les différents outils utilisés pour implémenter et simuler ces algorithmes ainsi que les différents résultats obtenus. La deuxième partie de ce chapitre est réservée pour la présentation et l'analyse des résultats obtenus. La conclusion et nos recommandations sont présentées à la fin du rapport.

CHAPITRE 1

REVUE DE LITTÉRATURE

La virtualisation (Aun Haider, 2009) est un procédé informatique qui consiste à ajouter une couche d'abstraction entre les ressources physiques et la représentation logique d'un système ou d'un réseau informatique (Kyriakos Zarifis, 2009). Afin de mieux comprendre cette technique, nous analyserons dans un premier temps les principales formes de virtualisation dans les systèmes informatiques. Ensuite, nous décrirons la virtualisation des réseaux à travers quelques technologies existantes et nous présenterons les plus importants bancs de tests de virtualisation de systèmes implémentés afin de permettre aux chercheurs d'y tester des nouveaux services et des architectures réseau. Enfin, nous présenterons les différents algorithmes d'allocation de ressources proposés dans la littérature.

1.1 Virtualisation dans les systèmes informatiques

Tout d'abord, il est essentiel de rappeler les différents types de techniques de virtualisation utilisés dans les systèmes informatiques. Nous en distinguons trois familles : virtualisation du stockage de données, virtualisation des serveurs et virtualisation des applications multimédias.

1.1.1 Virtualisation du stockage de données

La virtualisation du stockage de données, comme l'a définie (Bonnet, 2002), est une technique permettant de masquer la disparité des ressources de stockage (Ex. : équipements, protocole d'accès) et à les présenter comme une unité de stockage virtuelle homogène. En d'autres termes, cette technique crée une couche d'abstraction entre les applications et les ressources de stockage physique (*Voir* Figure 1.1). Pour être stocké, le trafic sortant des applications est intercepté par le périphérique de stockage virtuel qui le redirige vers les différents périphériques de stockage physiques (Ex. : disque dur externe, serveur de stockage). Ainsi, le périphérique de stockage virtuel permet d'avoir une vision homogène des

différents périphériques de stockage et donc de simplifier la gestion du stockage des données dans une entreprise. Il apporte aussi des fonctionnalités de réplication et de reproduction de données indépendamment des périphériques et de leur constructeur (Schmitt, 2008).

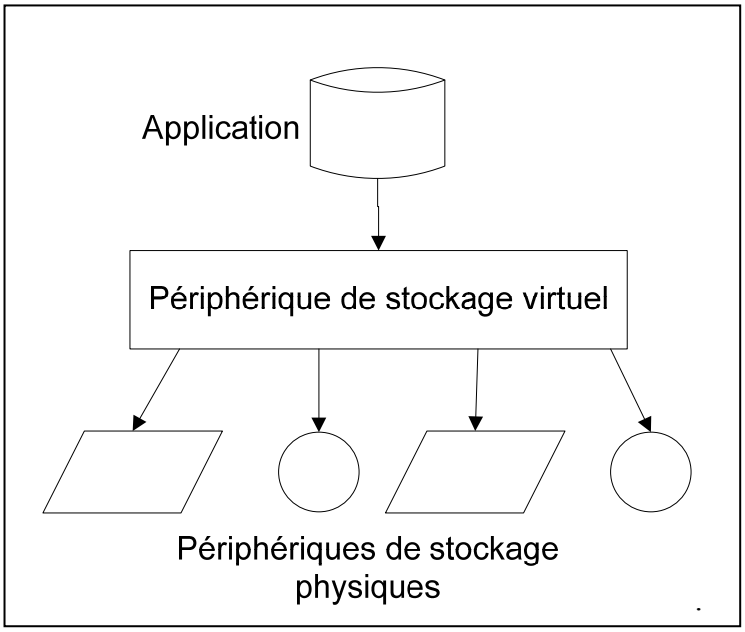

Figure 1.1 Virtualisation du stockage
Tirée de Monteiro (2002)

SAN Volume Controller (SVC) est un exemple de système de virtualisation de stockage, développé par IBM, permettant d'identifier toutes les ressources de stockage disponible au sein d'une entreprise et de veiller à ce que leur utilisation soit optimale (IBM, 2011). Ce système permet de combiner virtuellement la capacité de différents systèmes de stockage et permet de les gérer à partir d'une seule interface.

1.1.2 Virtualisation des serveurs

De nos jours, la majorité des entreprises utilisent des serveurs pour stocker leur base de données et pour partager leurs applications. Les entreprises optent de plus en plus à réduire le nombre de leurs serveurs à cause du coût important qu'engendre la maintenance, la mise à jour et le fonctionnement de ces équipements (Pradeep Padala, 2007). Ce besoin de réduire le

nombre de serveurs a incité les chercheurs à mettre en place une technique permettant de virtualiser les ressources d'un serveur. Cette technique consiste à partitionner un serveur physique en plusieurs entités virtuelles séparées (*Voir* Figure 1.2). Chaque entité virtuelle peut avoir son propre système d'exploitation et ses propres applications.

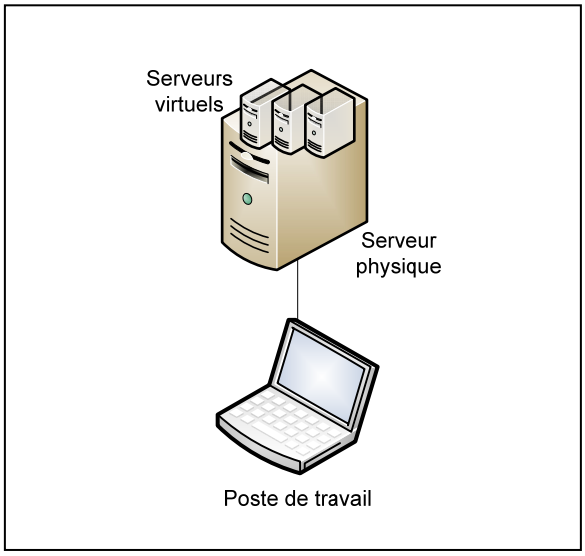

Figure 1.2 Virtualisation des serveurs

Un des avantages majeurs de cette technique est d'offrir la possibilité d'exécuter sur le même serveur plusieurs applications qui ne sont pas faites pour coexister. Par exemple, des applications qui requièrent des systèmes d'exploitation différents. Cette caractéristique permet de réduire considérablement les coûts en diminuant le nombre de serveurs physiques utilisés.

VMware ESX est un exemple d'application qui permet d'exécuter plusieurs machines virtuelles sur le même serveur. Chaque entité virtuelle présente un système entier indépendant avec son propre processeur, mémoire, espace disque et système d'exploitation. Ce partage de ressources du serveur entre plusieurs machines virtuelles permet de diminuer considérablement les coûts pour les entreprises. En effet, cette technique permet de réduire le nombre de serveurs physiques dont l'entreprise a besoin (VMware, 2011).

1.1.3 Virtualisation des applications

La virtualisation des applications, comme l'a définie (Gurr, 2008), est un procédé permettant d'ajouter une couche d'abstraction entre le système d'exploitation et les applications. Cette technique permet d'exécuter des applications sur un poste client sans les installer sur ce poste. En effet, ces applications sont réellement installées sur un serveur virtuel distant (*Voir* Figure 1.3).

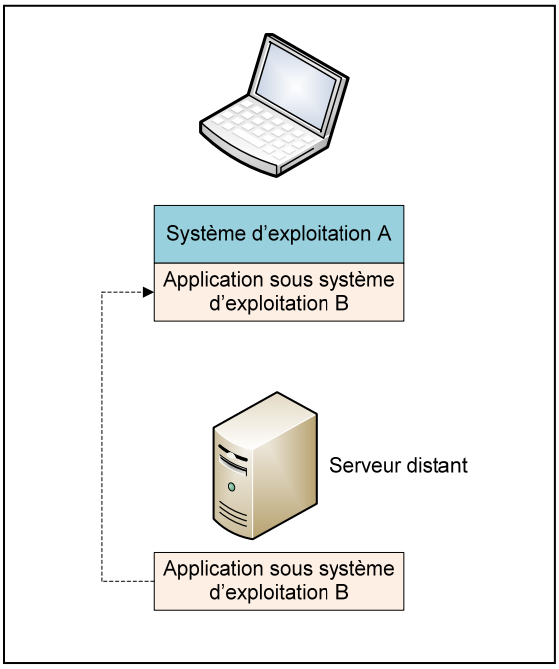

Figure 1.3 Virtualisation des applications

La virtualisation des applications permet d'exécuter des applications depuis un poste client indépendamment du système d'exploitation. Ce procédé permet d'éliminer le problème d'incompatibilité des applications.

1.2 Virtualisation dans les environnements réseaux

La virtualisation dans les environnements réseaux est un concept qui permet à plusieurs instances virtuelles (ou réseaux virtuels) de coexister et de partager les ressources d'une ou de plusieurs infrastructures physiques de réseaux (El Barachi, Kara et Dssouli, 2010). Comme l'a affirmé (Chowdhury et Boutaba, 2010), ce concept de coexistence de plusieurs réseaux n'est pas récent. En effet, plusieurs technologies existantes de réseau se basent sur ce concept.Dans cette section, nous allons décrire certaines de ces techniques qui permettent de déployer des réseaux virtuels à travers une même infrastructure réseau physique. Parmi ces techniques, nous retrouvons : les réseaux locaux virtuels (VLAN-Virtual Local Area Network), les réseaux privés virtuels (VPN-Virtual Privat Network), les réseaux de recouvrement et les réseaux actifs et programmables.

1.2.1 Réseaux locaux virtuels

La technologie VLAN permet de diviser les réseaux locaux (LAN) en plusieurs réseaux locaux logiques. Les ordinateurs appartenant au même réseau logique peuvent communiquer directement entre eux comme dans un LAN. Par contre, deux ordinateurs appartenant à des VLAN différents ne peuvent communiquer que par le biais d'un routeur. Chaque VLAN constitue alors un domaine de diffusion (Broadcast domain) indépendant. Cette technologie permet de confiner le trafic de diffusion à l'intérieur de chaque VLAN ce qui réduit l'utilisation de la bande passante et améliore ainsi les performances du réseau. Par ailleurs, la sécurité dans le réseau local est améliorée, car les utilisateurs de VLAN différents ne peuvent communiquer ensemble directement (Zeng et Cheng, 2009). Les VLAN peuvent être classés selon leur critère de commutation et la couche au niveau de laquelle cette commutation s'effectue :

VLAN de niveau 1 (VLAN par port) :

Un nom est assigné à un groupe d'un ou plusieurs ports du commutateur. Pour que deux ordinateurs connectés sur des ports de nom différents puissent communiquer, le trafic doit être acheminé par un routeur (Zeng et Cheng, 2009).

VLAN de niveau 2 (VLAN MAC) :

Les trames sont classées par le commutateur selon leur adresse MAC source. En effet, les adresses MAC des stations sont classées par l'administrateur du réseau dans des listes appelées MAC-to-VLAN. Chaque liste correspond alors à un réseau logique indépendant.

VLAN de niveau 3 (VLAN par sous-réseau) :

Les machines appartenant au même sous-réseau sont classées dans le même VLAN.

1.2.2 Virtual Private Network (VPN)

Un VPN est un réseau dédié permettant de connecter plusieurs sites à l'aide du principe de tunnellisation (Diab, Tohme et Bassil, 2008). Ce dernier permet de transmettre des données, via un tunnel virtuel, d'une manière plus sécurisée en utilisant des algorithmes de cryptographie. Cette technologie est utilisée généralement pour interconnecter plusieurs sites, d'une même entreprise, séparés géographiquement. Selon le protocole utilisé au niveau du plan de données, la technologie VPN peut être classée en trois catégories : VPN couche 1, VPN couche 2 et VPN couche 3.

VPN couche 3 (L3VPN) :

Il permet de connecter des sites géographiquement dispersés à travers le réseau commuté par paquets d'un ou de plusieurs fournisseurs de services. En effet, chaque site client détient un nombre d'équipements client de bord (Customer Edge-CE) connectés aux équipements fournisseur de bord (Provider Edge-PE) dans le réseau cœur. Ce dernier permet au trafic reçu de transiter entre les PE en utilisant ses routeurs internes. Le L3VPN garantit aussi pour chaque utilisateur un trafic privé et séparé des autres utilisateurs connectés au réseau cœur (Mohapatra, Metz et Yong, 2007).

VPN couche 2 (L2VPN) :

Il permet de transporter des trames de la couche 2 (Ex. : relais de trames, ATM) entre les différents sites participants (Chowdhury et Boutaba, 2010).

VPN couche 1 (L1VPN) :

Il utilise et renforce le contrôle et la gestion de L2VPN et de L3VPN. En effet, les utilisateurs ont la possibilité de réguler leur trafic en appliquant des techniques d'ingénierie de trafic (Dhaini, Pin-Han et Xiaohong, 2010). De plus, plusieurs technologies de transport de la couche 1 peuvent être utilisées (Ex. : TDM, OTN).

1.2.3 Réseaux de recouvrement

Un réseau de recouvrement (*overlay*) est un réseau virtuel implémenté sur un ou plusieurs réseaux physiques existants (Kawahara et al., 2011). La mise en place d'une telle architecture n'affecte pas les fonctionnalités du réseau sous-jacent. Cette technique a été alors utilisée pour implémenter plusieurs nouvelles fonctionnalités au réseau internet. Par exemple, nous pouvons considérer la qualité de service comme une couche de recouvrement au-dessus du réseau internet et qui utilise ses propres mécanismes pour router l'information(Kawahara et al., 2009). Plusieurs autres couches de recouvrement ont été proposées dans la littérature pour une meilleure gestion du routage (Elaoud et al., 2005), garantir une protection contre les attaques par saturation (Beitollahi et Deconinck, 2008) et la distribution de données (Byers et al., 2004) ainsi que supporter la multidiffusion (Jannotti et al., 2000). L'inconvénient des réseaux de recouvrement est l'effort considérable nécessaire pour les créer et les maintenir. C'est pour cette raison que seulement quelques-unes des nombreuses nouvelles architectures ont été testées sur des réseaux de recouvrement (Anderson et al., 2005).

1.2.4 Réseaux actifs et programmables

Pour répondre aux besoins des nouvelles applications multimédias, il est parfois nécessaire d'ajouter de nouveaux services ou de modifier les services réseau existants (Ex. : Intégration des services différenciés au réseau internet afin d'améliorer la qualité de service offerte). Cependant, ces changements sont généralement des processus longs et coûteux. L'objectif des réseaux actifs et programmables est donc de proposer des interfaces programmables ouvertes (API) et standardisées (Campbell et al., 1999) permettant la conception, l'implémentation et l'introduction de nouveaux services (Olivier FESTOR, 2000). Par exemple, le projet PIN 1650 (Biswas et al., 1998) de IEEE initié en 1997 propose un ensemble d'interfaces de programmation permettant, entres autres, de programmer facilement des services pour la gestion de la qualité de service sur IP(Olivier FESTOR, 2000).

1.3 Projets de virtualisation de réseaux

Afin de valider une nouvelle architecture réseau, les chercheurs ont besoin de la tester sur le réseau internet « réel » afin de déceler la moindre anomalie qui a pu échapper aux simulateurs ou émulateurs dans les premiers tests réalisés. Par contre, les fournisseurs d'accès Internet (FAI) ne veulent en aucun cas prendre le risque de perturber le bon fonctionnement de leur réseau pour tester de nouveaux protocoles, car la moindre interruption de service ou panne peut leur causer une perte considérable de revenu. Afin de résoudre ce problème, des plateformes de test ont été implémentées dans le but de permettre aux chercheurs de tester leurs nouvelles architectures sur une topologie et avec un trafic internet « réels » sans affecter les performances de ce réseau (Spyropoulos, Fdida et Kirkpatrick, 2007). Bien que leur but soit le même, les différentes plateformes développées ont des architectures et des caractéristiques différentes. Elles peuvent être classées selon les caractéristiques suivantes (Chowdhury et Boutaba, 2010):

Technologie du réseau :

Certaines architectures de virtualisation de réseaux ont été développées pour des technologies de réseau spécifiques. Par exemple, X-Bone pour la technologie IP, Tempest pour les réseaux ATM et GENI pour les réseaux multi technologiques (Chowdhury et Boutaba, 2010).

Couche de virtualisation :

Les chercheurs ont été influencés par le modèle en couche du réseau Internet existant. En effet, ils ont proposé des projets qui visent à virtualiser ces couches allant de la couche physique (Ex. : UCLP) jusqu'à la couche application (Ex. : VIOLIN) (Chowdhury et Boutaba, 2010).

Domaine architectural :

La plupart des projets ont mis l'accent sur un domaine architectural bien défini. Ce domaine influence les choix de conception des architectures de ces projets ainsi que les services qu'ils offrent. Par exemple, le projet VNRMS vise le domaine de gestion des réseaux virtuels (Chowdhury et Boutaba, 2010).

Niveau de virtualisation :

Le niveau de virtualisation veut dire le type de ressources virtualisées dans un réseau (Ex. : nœuds, liens). Par exemple, PlanetLab se base sur la virtualisation des nœuds tandis que UCLP se base sur la virtualisation des liens (Chowdhury et Boutaba, 2010).

Dans cette section, nous présentons un exemple de projet réalisé pour chacune de ces classes.

1.3.1 Technologie du réseau (X-Bone)

Plusieurs projets de virtualisation ont été conçus pour tester des technologies réseau bien définies. Le X-Bone, par exemple, est la première infrastructure de réseaux virtuels développée afin de permettre aux chercheurs de tester leurs protocoles et service dans des conditions réelles sur des réseaux IP (Joe Touch 1998). Cette infrastructure permet de créer

des liens virtuels entre les routeurs et les hôtes virtuels en utilisant le principe de tunnellisation. Elle coordonne la configuration et la gestion des différents réseaux virtuels et permet la découverte et le déploiement des ressources physiques disponibles au niveau des différents nœuds. X-Bone offre une abstraction entre les différents réseaux de recouvrement ce qui permet de protéger le trafic et d'isoler ainsi les tests des différents protocoles. Cette architecture de virtualisation supporte la concurrence, la récursion et la revisitation. La concurrence permet la coexistence de plusieurs réseaux de recouvrement. La récursion offre la possibilité de déployer un réseau de recouvrement à l'intérieur d'un autre. La revisitation permet de réutiliser le même nœud par un réseau de recouvrement. L'accès aux différentes fonctionnalités du réseau X-Bone se fait à partir d'une interface web ou une interface XML (Chun et al., 2003) , (Carbone et Rizzo, 2009).

1.3.2 Couche de virtualisation (Emulab)

Les projets de virtualisation peuvent être classés selon les quatre couches du réseau IP défini par IETF et dépendamment de leurs caractéristiques. Par exemple, Emulab (Emulation Laboratory) est une plateforme de test au niveau liaison basée sur la technologie de commutation VLAN. Cette plateforme utilise VLAN pour émuler des réseaux à l'intérieur d'un réseau local (Chun et al., 2003). En effet, cette plateforme offre aux chercheurs la possibilité de développer, de débuguer et d'évaluer leurs systèmes. Emulab a été mis en place par le groupe de recherche Flux de l'Université Utah. Actuellement, plusieurs centaines de nœuds ont été implémentés sur vingt-quatre sites à travers le monde. Les utilisateurs peuvent utiliser les ressources des nœuds et des liens Emulab en utilisant l'interface web Emulab. À travers cette interface, les utilisateurs peuvent spécifier le type de nœuds, leur système d'exploitation ainsi que la nature des liens reliant ces nœuds(Yuen, 2006).

1.3.3 Domaine architectural (CABO)

La gestion du réseau (Leon-Garcia et Mason, 2003), la conception, la création et le déploiement d'architecture réseaux(Rooney et al., 1998), la création et le déploiement des protocoles des réseaux actifs (da Silva, Yemini et Florissi, 2001) sont des domaines

architecturaux visés par plusieurs projets de virtualisation. CABO (Concurrent Architectures are Better than One) est une architecture visant à faciliter le déploiement de nouveaux services en séparant les fournisseurs d'infrastructure physiques (FIP) des fournisseurs d'accès Internet (FAI). En effet, actuellement les FAI gèrent leurs infrastructures physiques et fournissent en même temps des services aux utilisateurs. Dans l'architecture d'internet actuel, un FAI doit obtenir l'accord des autres FAI avant d'introduire le moindre changement au réseau. Prenons par exemple le cas d'un FAI qui veut offrir de la qualité de service (QoS) à ses clients. Étant donné que le trafic passera certainement par d'autres réseaux, le FAI, même s'il a modifié son propre réseau, doit aussi convaincre les autres FAI à apporter les mêmes modifications sur leurs réseaux respectifs. Ce problème bloque l'évolution du réseau internet. Pour pallier ce problème, CABO propose de partager le rôle du FAI en deux parties : d'une part la gestion de l'infrastructure physique et la virtualisation de ressources réalisées par le FIF et d'autre part la fourniture de services et l'allocation de ressources virtuelles réalisées par le fournisseur de service (FS)(Spyropoulos, Fdida et Kirkpatrick, 2007).

1.3.4 Niveau de virtualisation (PlanetLab)

Planetlab (Planet NETwork Laboratory) est un projet qui se base sur la virtualisation des nœuds (Peterson et al., 2003). En effet, PlanetLab est un réseau de recouvrement implémenté en 2002 sur le réseau internet et composé actuellement d'environ mille nœuds répartis partout dans le monde (*Voir* Figure 1.4). L'accès aux nœuds est réservé aux personnes et aux organismes affiliés. Chaque personne ou organisme se fait attribuer des parties des ressources d'un nœud pour créer un environnement distribué virtualisé. Ces parties sont des machines virtuelles instanciées au niveau des nœuds physiques de PlanetLab. Les ressources physiques telles que le CPU, l'espace disque et la bande passante doivent être partagées convenablement entre les machines virtuelles afin d'optimiser l'utilisation de ces ressources et d'augmenter ainsi le nombre de parties assignées. Le réseau PlanetLab comprend aussi un nœud central qui représente le cœur du système. En effet, ce nœud comporte le logiciel de

gestion de toute la plateforme ainsi que la base de données des différents utilisateurs et nœuds du système (Chun et al., 2003) , (Carbone et Rizzo, 2009).

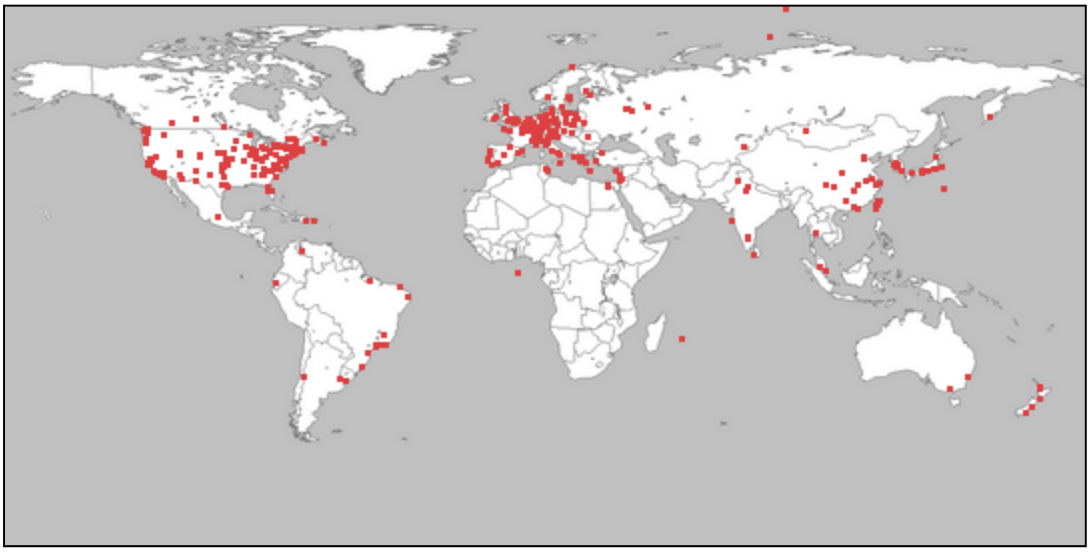

Figure 1.4 Répartion des nœuds PlanetLab dans le monde
Tirée de The Trustees of Princeton University (2007)

Parmi les nouvelles architectures proposées certaines exigent des changements non seulement au niveau matériel et logiciel mais elles exigent aussi d'établir des accords entre les fournisseurs d'infrastructure IP pour implémenter de nouvelles fonctionnalités au niveau architecture. CABO et le logiciel intégré de virtualisation de réseau 4Ward (AB, 2009) représentent deux exemples de telles architectures. Alors que l'architecture de CABO établit une séparation entre les fournisseurs d'infrastructure et les fournisseurs de service, celle de 4Ward définit trois rôles distincts : 1- fournisseurs d'infrastructures physiques, 2- fournisseurs de réseaux virtuels, 3- opérateurs de réseaux virtuels. Les fournisseurs d'infrastructures physiques gèrent les ressources physiques. Les opérateurs de réseaux virtuels permettent de créer les réseaux virtuels. Les opérateurs de réseaux virtuels offrent des services aux usagers à travers les réseaux virtuels. Un des défis majeurs pour la conception de telles architectures est de mettre en place une stratégie de sélection et d'allocation de ressources qui permet d'utiliser efficacement les ressources disponibles. Dans

la section qui suit, nous présentons les différentes stratégies proposées dans des travaux de recherches antérieures.

1.4 Stratégies d'allocation de ressources

La majorité des algorithmes proposés , par exemple ceux proposés par ((Houidi, Louati et Zeghlache, 2008b), (Bo et al., 2010), (Houidi, Louati et Zeghlache, 2008a), (Zhu et Ammar, 2006), (Turner, 2006), (Yu et al., 2008)) s'exécutent en deux phases : Dans la première phase, on commence par assigner les différents nœuds virtuels qui nécessitent une plus grande puissance de traitement aux nœuds physiques offrant le plus de CPU. Dans la seconde partie, on assigne les différents liens virtuels aux liens de l'infrastructure physique en utilisant des algorithmes du plus court chemin. Cette séparation permet de réduire la complexité du processus de réservation de ressources. En d'autres termes, l'implémentation d'un algorithme qui assigne à la fois les nœuds et les liens est très complexe et c'est pour cette raison que la majorité des approches proposées jusqu'à ce jour séparent ces deux étapes. Les auteurs (Chowdhury, Rahman et Boutaba, 2009)proposent des algorithmes permettant une meilleure corrélation entre ces deux phases en utilisant des techniques d'optimisation à l'aide de modèles mathématiques de relaxation. Les tests de performance réalisés dans cet article ont prouvé que cette méthode permet une meilleure utilisation des ressources. Plusieurs autres hypothèses ont été considérées par les chercheurs afin de simplifier leurs algorithmes d'allocation de ressources (Aun Haider, 2009). Parmi ces dernières, on peut citer :

1) l'utilisation des topologies telles que le Hub and Spoke dans les réseaux physiques : Généralement les algorithmes proposés divisent le réseau substrat en plusieurs grappes. Chaque grappe est composée d'un nœud central qui a la plus grande valeur de CPU (c'est le hub) et de l'ensemble de nœuds qui lui sont directement connectés. Cette topologie est appelée Hub and Spoke (Houidi, Louati et Zeghlache, 2008a) ;

2) la négligence des contrôles d'admission : Ces algorithmes considèrent les ressources du réseau physique infinies. En d'autres termes, toutes les requêtes envoyées par le SP sont automatiquement acceptées par le FIV sans contrôler les ressources disponibles dans l'infrastructure physique (Zhu et Ammar, 2006) ;

3) l'utilisation des versions de requêtes offline : Toutes les requêtes sont connues d'avance donc les algorithmes utilisés sont des algorithmes statiques ((Zhu et Ammar, 2006), (Turner, 2006)).

Dans les approches proposées, les chercheurs ont utilisés trois méthodes pour concevoir leurs algorithmes : La méthode centralisée, la méthode distribuée et la méthode hybride. Nous allons décrire le fonctionnement de chacune de ces approches ainsi que leurs avantages et inconvénients.

1.4.1 Méthode centralisée

La méthode centralisée comme celle proposée par ((Zhu et Ammar, 2006), (Turner, 2006), (Yu et al., 2008)) consiste à déléguer la tâche de gestion et du déploiement du réseau virtuel à une seule entité centrale. Cette dernière doit maintenir à jour les informations concernant toute la topologie de l'infrastructure physique afin de déployer d'une manière optimale les réseaux virtuels. En revanche, si l'infrastructure physique est très dynamique, c'est-à-dire s'il y a fréquemment d'ajout et/ou de suppression de nœuds, le temps de traitement au niveau de l'entité centrale peut être considérable(Houidi, Louati et Zeghlache, 2008a).

1.4.2 Méthode distribuée

Afin de corriger ces retards causés par cet aspect centralisé, d'autres chercheurs tels que (Houidi, Louati et Zeghlache, 2008a)ont proposé des algorithmes distribués qui partagent la tâche d'allocation des ressources entre plusieurs nœuds physiques. Ces nœuds doivent coopérer en échangeant des messages afin de garder à jour leurs informations concernant l'état de la disponibilité des ressources dans le réseau physique(Michael Till Beck, 2012). Il

s'est avéré que cette méthode est complexe à implémenter et très gourmande en ressources à cause du nombre important de messages échangés entre les différents nœuds physiques.

1.4.3 Méthode hybride

La troisième solution, proposée par (Bo et al., 2010), est une architecture hybride qui utilise un nœud central dont le rôle est de subdiviser la requête reçue du FS en plusieurs sous requêtes et de déléguer la tâche d'allocation de ressources à d'autres nœuds physiques. Jusqu'à ce jour, aucun algorithme n'a été implémenté et testé pour cette dernière architecture. Nous proposons alors dans le chapitre 3 une stratégie d'allocation de ressources utilisant une approche hybride. Mais, avant de décrire cette stratégie, nous allons présenter dans le chapitre qui suit l'architecture de virtualisation de réseau choisi pour implémenter une telle stratégie.

CHAPITRE 2

ARCHITECTURE DE VIRTUALISATION DE RÉSEAUX ORIENTÉE SERVICES

Dans une architecture de virtualisation de réseaux, plusieurs acteurs entrent en jeu pour établir et gérer les réseaux virtuels. Dans ce chapitre nous allons présenter les motivations qui ont incité les chercheurs à s'intéresser à une telle architecture. Ensuite, nous décrirons le modèle d'affaires adopté ainsi que les rôles de ces différents acteurs. Puis, nous expliquerons les plus importants principes et défis de cette architecture ainsi que les étapes qui mènent à l'élaboration d'une infrastructure réseau virtuelle.

2.1 Motivations

La Figure 2.1 illustre les trois principaux scénarios de motivations pour le développement du concept de virtualisation de réseaux. Ces motivations sont : le coût important du déploiement des infrastructures physiques des réseaux, la personnalisation des services fournis selon les besoins des utilisateurs et la convergence des différentes infrastructures existantes.

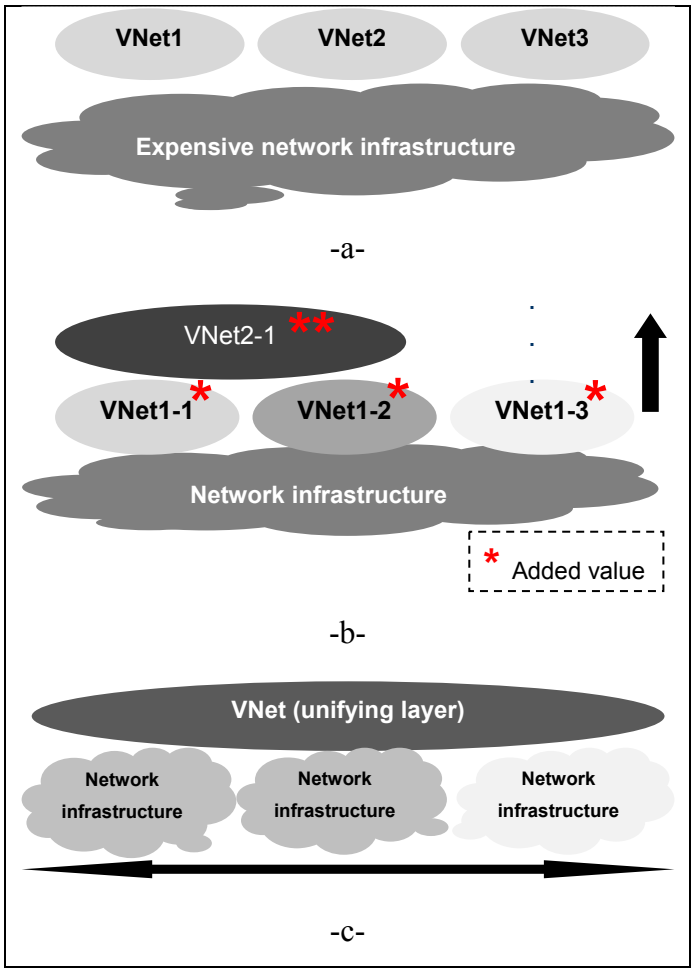

Figure 2.1 Scénarios de motivation pour la virtualisation des réseaux
Tirée de El Barachi, Kara et Dssouli (2010)

2.1.1 Coût du déploiement des infrastructures physiques

Le coût important du déploiement des infrastructures physiques des réseaux présente un obstacle devant les FS qui ont des capacités financières limitées (*Voir*Figure 2.1-a). En effet, ces derniers ne peuvent pas concevoir et déployer leurs propres infrastructures et par conséquent ils restent dépendants aux fournisseurs d'infrastructures physiques. La virtualisation des réseaux s'avère une solution à ce problème en permettant la division des ressources d'une infrastructure physique en plusieurs parties et les affecter aux différents réseaux virtuels (El Barachi, Kara et Dssouli, 2010).

2.1.2 Personnalisation des solutions réseau

La virtualisation des réseaux offre aux FS la possibilité de mettre en place de nouvelles technologies (par exemple de nouveaux protocoles de routage) sur les infrastructures existantes (*Voir* Figure 2.1-b). Aussi, ce concept permet de personnaliser les protocoles existants selon les besoins des clients (El Barachi, Kara et Dssouli, 2010).

2.1.3 Convergence des infrastructures existantes

La coopération entre les différents types de réseaux est une tâche délicate qui nécessite le recours à des passerelles complexes et de différents accords entre les opérateurs. La virtualisation des réseaux permet de simplifier ce processus en créant une couche d'abstraction au-dessus des différentes infrastructures physiques de réseaux facilitant ainsi leur interfonctionnement (*Voir* Figure 2.1-c)(El Barachi, Kara et Dssouli, 2010).

2.2 Modèle d'affaires

Un modèle d'affaires décrit les différentes entités qui interagissent entre elles pour fournir un service donné (El Barachi, Kara et Dssouli, 2010). Ce modèle est généralement utilisé comme base pour la définition de différentes architectures de communication. Avant de présenter le modèle d'affaires utilisé dans notre travail, rappelons ceux utilisés traditionnellement dans les environnements réseaux à savoir les modèles TINA, Parlay et Service Web.

2.2.1 Modèle d'affaire TINA

Le modèle TINA (Telecommunication Information Networking Architecture) est l'un des modèles d'affaires les plus utilisés dans le domaine des télécommunications. Ce modèle définit cinq différents rôles : le consommateur (Consumer), le revendeur (Retailer), le tiers fournisseur de service (Third Party Service Provider), le courtier (Broker) et le fournisseur de réseau (Network Provider). Le consommateur est l'entité qui utilise les services (Ex. :

personne privée, ménage ou entreprise). Le revendeur propose les différents services aux consommateurs et il leur offre des garanties sur la qualité des services proposés. Le tiers fournisseur de service offre des services à des parties prenantes autres que les consommateurs. Le courtier offre des informations afin de trouver les services et les parties prenantes. Le fournisseur de réseau est l'entité qui fournit les services de transport et gère les équipements de communication tels que (Ex. : routeurs et commutateurs) (L.Kristiansen, 1998).

2.2.2 Modèle d'affaire Parlay

Parlay offre un ensemble d'interfaces de programmation (API) ouvertes qui visent à faciliter le développement des applications de télécommunication en offrant aux développeurs un accès contrôlé aux capacités des réseaux de télécommunications. Ce modèle définit trois rôles principaux : L'application client (Application Client) qui utilise les services Parlay ; l'opérateur de l'entreprise (Entreprise Operator) qui s'abonne aux services ; l'opérateur de Framework (Framework Operator) qui gère les abonnements (El Barachi, Kara et Dssouli, 2010).

2.2.3 Modèle d'affaires Service Web

Le Service Web (WS) est un modèle développé afin de permettre la communication de données entre des applications hétérogènes (Ex. : applications utilisant des systèmes d'exploitation différents). Ce modèle définit trois principaux rôles: Le fournisseur WS (WS Provider) qui détient un ou plusieurs services web (s); le demandeur WS (WS Requestor) qui souhaite utiliser un service web, et le registre WS (WS Registry) qui met en contact les fournisseurs et les demandeurs (El Barachi, Kara et Dssouli, 2010).

2.2.4 Modèle d'affaire pour les environnements de virtualisation de réseaux

Pour ce travail, nous adoptons le modèle d'affaires proposé par (El Barachi, Kara et Dssouli, 2010). Ce modèle est composé de cinq entités différentes : les fournisseurs d'infrastructure

physique (FIP), les fournisseurs de service (FS), les fournisseurs d'infrastructure virtuelle (FIV). Ces derniers partagent la tâche de gestion des réseaux réservée exclusivement aux fournisseurs d'accès à Internet (FAI) dans le réseau Internet traditionnel. Les autres entités intervenant dans cette architecture sont le consommateur et le registre de services et de ressources (*Voir* Figure 2.2).

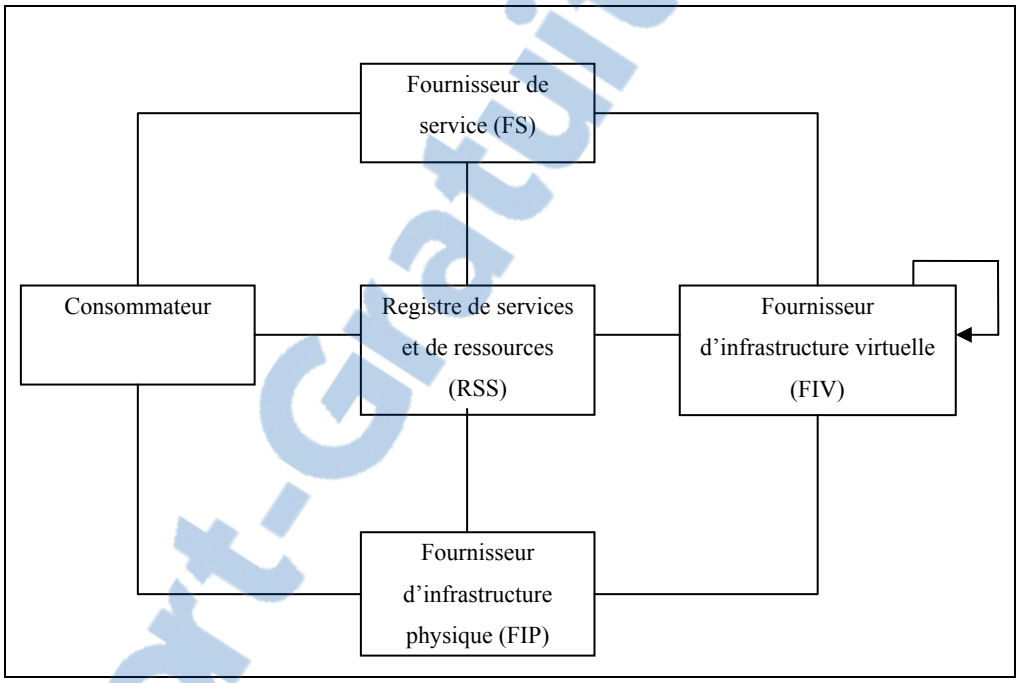

Figure 2.2 Modèle d'affaires proposé pour la virtualisation des réseaux
Reproduite et adaptée avec l'autorisation de El Barachi, Kara et Dssouli (2010)

Fournisseurs d'infrastructure physique (FIP)

Ils sont responsables du déploiement de l'infrastructure physique et de la gestion de ses ressources. Ils sont aussi chargés de la maintenance et la mise à jour du réseau physique. Les différents FIP communiquent et collaborent selon des ententes prédéfinies (El Barachi, Kara et Dssouli, 2010).

Fournisseurs de service (FS)

Ils offrent des services spécifiques aux utilisateurs. Ces services sont spécifiés dans un accord de niveau de service (SLA) négocié entre les deux parties. Par exemple, si le FS est un hébergeur de sites Internet, il peut assurer à son client un délai minimum de connexion continue sans perte de paquets. Le contrat peut contenir les contreparties que le prestataire doit fournir à son client en cas du non-respect du contrat (Ex. : indemnité financière).

Fournisseurs d'infrastructure virtuelle (FIV)

Ils jouent le rôle le plus important dans un environnement de virtualisation de réseaux. En effet, ils louent les ressources physiques offertes par les FIP afin de déployer des réseaux virtuels. Ils sont responsables aussi de la gestion et de la maintenance de toute l'infrastructure virtuelle.

Consommateurs (Utilisateurs)

Les consommateurs d'un environnement virtuel sont similaires aux consommateurs du réseau internet actuel sauf qu'ils auront un choix plus vaste de services offerts vu le nombre important de réseaux virtuels déployés par chaque fournisseur. Chaque consommateur peut être connecté simultanément à plusieurs fournisseurs pour des services différents.

Registres de services et de ressources (RSS)

Ils ont un rôle similaire à celui du courtier dans le modèle TINA. En effet, cette entité est responsable de mettre en contact toutes les parties en fournissant des informations sur les services et les ressources qui sont enregistrées.
La Figure 2.3 illustre la localisation des champs d'action des principaux acteurs dans les modèles de références OSI et TCP/IP.

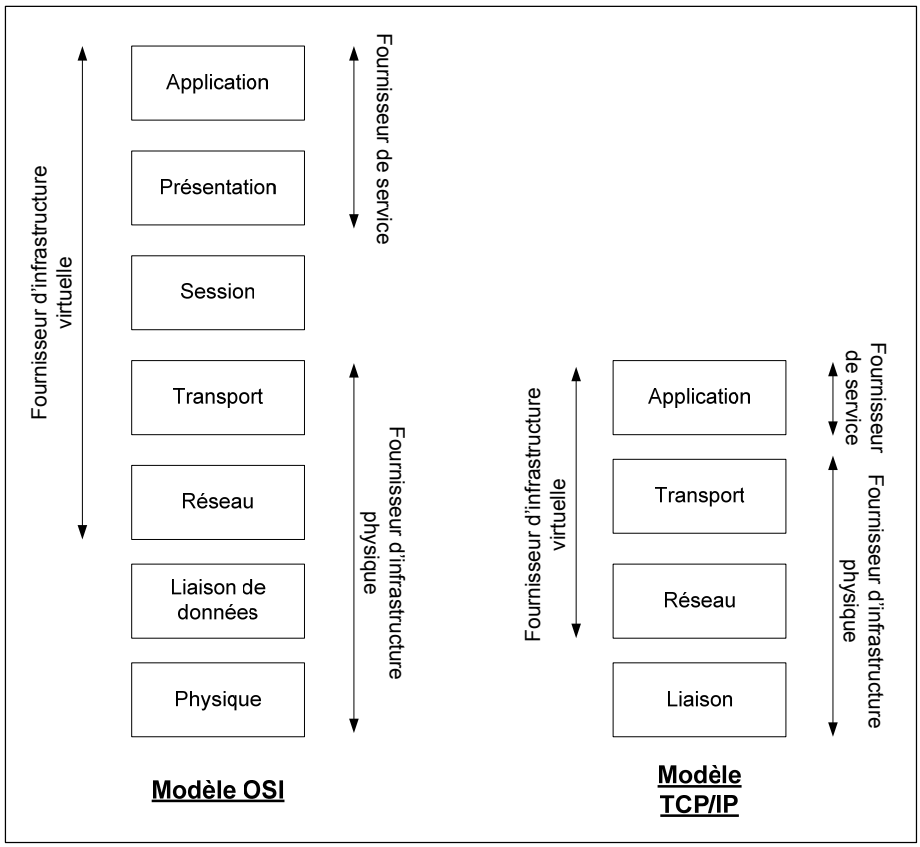

Figure 2.3 Champs d'action des différents acteurs dans les modèles OSI et TCP/IP
Reproduite et adaptée avec l'autorisation de El Barachi, Kara et Dssouli (2010, p. 5)

2.3 Architecture de virtualisation d'un réseau

Un réseau virtuel est un ensemble de nœuds virtuels, dont chacun est hébergé dans un nœud physique, connecté par des liens virtuels. Ces derniers s'étendent sur un ou plusieurs chemins physiques et incluent une partie des ressources fournies par le réseau physique. La Figure 2.4présente un environnement qui comprend deux réseaux virtuels : 1 et 2, déployés et gérés respectivement par le FIV_1 et le FIV_2. Chaque réseau virtuel loue les ressources physiques fournies par FIP_1 et FIP_2. Le client U_1 utilise les services offerts par FS_2, le client U_2 quant à lui s'offre les services de FS_1 et le dernier client U_3 est connecté simultanément à ces deux prestataires pour des services différents (Chowdhury et Boutaba, 2010).

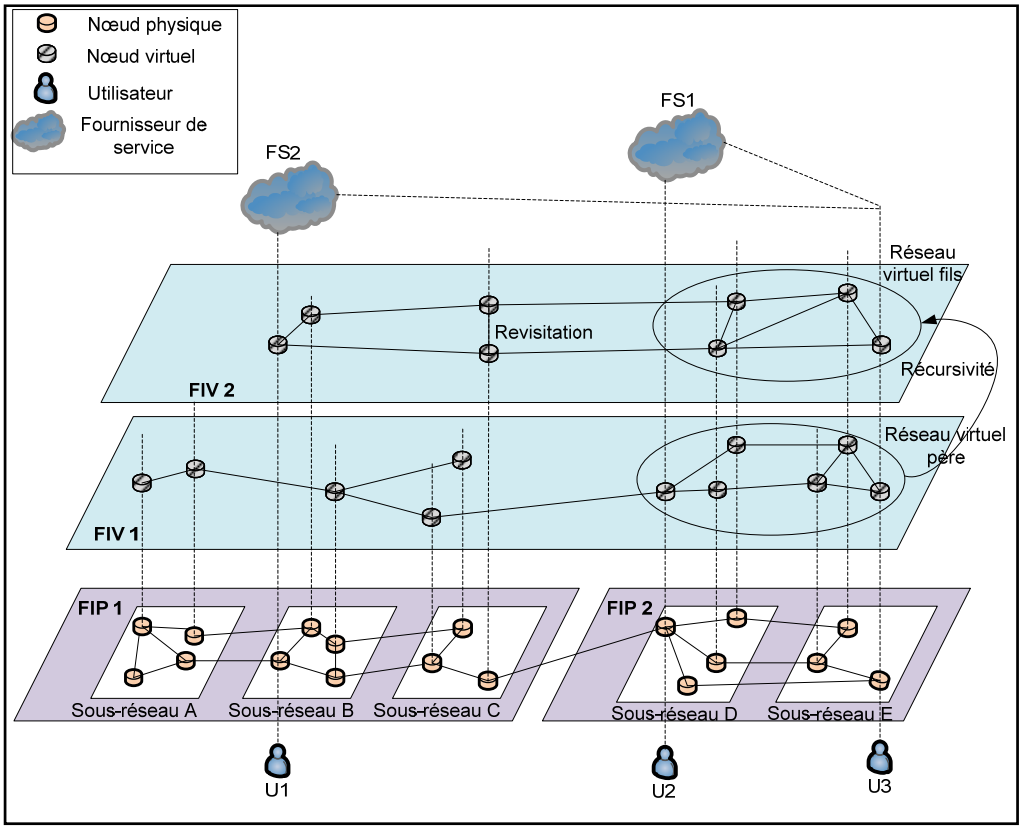

Figure 2.4 Architecture de la virtualisation des réseaux
Reproduite et adaptée avec l'autorisation de Chowdhury et Boutaba (2010)

2.3.1 Principes de l'architecture

Coexistence

C'est le principe le plus important de cette nouvelle architecture. Plusieurs réseaux virtuels déployés par différents FIV et implémentant des technologies différentes peuvent coexister dans le même environnement de virtualisation, interagir et partager la même infrastructure physique. Cette dernière peut être composée de plusieurs réseaux physiques gérés par différents FIP. Dans la Figure 2.4, les réseaux virtuels 1 et 2 coexistent et partagent la même infrastructure physique (Chowdhury, 2008).

Récursivité

Un ou plusieurs réseaux virtuels fils peuvent être créés à partir d'un réseau virtuel père. Dans la Figure 2.4, une section de l'infrastructure virtuelle (un réseau virtuel) gérée par FIV_2 est créée à partir d'une autre section de l'infrastructure virtuelle gérée par FIV_1 selon le principe de la récursivité (Chowdhury, 2008).

Héritage

Les réseaux virtuels enfants héritent les caractéristiques de leur réseau père. Dans la Figure 2.4, les contraintes imposées par FIP_2 au réseau virtuel père seront automatiquement transférées au réseau virtuel fils selon le principe d'héritage (Chowdhury, 2008).

Revisitation

Un nœud physique est composé de plusieurs instances logiques donc il peut héberger plusieurs nœuds virtuels de différents fournisseurs (Chowdhury, 2008). Par exemple, si un nœud virtuel utilise une partie seulement des ressources disponibles dans un nœud physique, le reste de ces ressources peuvent être utilisées par un ou plusieurs d'autres nœuds virtuels.

2.3.2 Défis de l'architecture

Flexibilité

Chaque FIV doit être capable d'utiliser n'importe quelle technologie, protocole de routage ou même des fonctions de contrôle personnalisé indépendamment du réseau physique qu'il utilise et des autres réseaux virtuels existants(Chowdhury, 2008). Par exemple, dans le réseau internet actuel, le déploiement du *source routing*, qui consiste à définir la route du paquet par l'expéditeur et non pas par le routeur, est très difficile à cause du manque de consensus entre

les différents FAI. Dans un environnent virtualisé, le FIV devra être capable d'offrir des technologies telles que le *source routing*.

Maniabilité

La séparation entre les FIP et les FIV permet à ces deux entités de se consacrer totalement et efficacement à la gestion de leurs réseaux respectifs. En effet, les FIP auront le contrôle total du réseau physique et fourniront l'accès à ses ressources disponibles aux FIV. Chaque FIV utilisera alors les ressources d'un ou de plusieurs FIP afin de satisfaire ses besoins en bande passante, en capacité de traitement ou en délai (Chowdhury, 2008).

Mise à l'échelle

La coexistence de plusieurs réseaux virtuels est l'une des principales motivations derrière la virtualisation des réseaux. Les FIP doivent pouvoir répondre aux besoins des FIV en termes de ressources sans affecter leurs performances (Chowdhury, 2008).

Isolation

La virtualisation des réseaux doit permettre une isolation complète, logique et physique, entre les réseaux virtuels coexistants. Ceci permettra d'améliorer la tolérance aux pannes ainsi que la sécurité du réseau (Chowdhury, 2008). En d'autres termes, si un réseau virtuel est victime d'une panne ou d'une attaque malicieuse ceci n'affectera que le réseau en question et non pas l'ensemble de l'architecture.

2.3.3 Étapes pour la création d'un réseau virtuel

Le FIP est responsable de la virtualisation des ressources physiques disponibles et de la publication de ces données dans un système de découverte de ressources afin qu'elles soient accessibles à tous les FIVs. Dès la réception d'une requête de la part d'un utilisateur, le

prestataire collabore avec les FIVs dans le but de concevoir un réseau virtuel dont les caractéristiques répondent à la demande du client (Houidi et al., 2009). Cette phase de conception peut être composée de trois étapes:

1) découverte des candidats : Les candidats représentent l'ensemble des nœuds et liens physiques qui répondent à la demande du FIV ;

2) sélection du candidat : Choisir le meilleur nœud en fonction de paramètres tels que son CPU et sa bande passante en utilisant des algorithmes d'optimisation ;

3) affectation du candidat : Affecter les ressources du candidat choisi au nœud virtuel correspondant.

Les trois étapes sont générées par des algorithmes de gestion et d'affectation de ressources qui sont fournis par des environnements de virtualisation de réseaux. Le chapitre qui suit décrit l'approche hybride que nous proposons et qui intègre de tels algorithmes.

CHAPITRE 3

APPROCHE HYBRIDE POUR LA GESTION DES RESSOURCES

Ce chapitre décrit l'approche que nous proposons pour la gestion des ressources dans un environnement de virtualisation de réseaux. Aussi, il explique en détail les principaux algorithmes proposés pour la gestion et l'affectation de ressources dans de tels environnements.

3.1 Modélisation du réseau et description des objectifs de l'algorithme

Dans cette section, nous rappelons le principe de la virtualisation des réseaux. Ensuite, nous décrivons la modélisation des réseaux physiques et virtuels ainsi que les valeurs résiduelles des ressources physiques. Enfin, nous définissons les paramètres importants d'évaluation de performance à savoir le coût physique des ressources, le coût virtuel des ressources, le revenu et la consommation en CO_2.

3.1.1 Rappel de l'architecture de la virtualisation des réseaux

Nous rappelons, comme le soulignent (Houidi, Louati et Zeghlache, 2008b), qu'un réseau virtuel est un ensemble de nœuds virtuels (Ex. : routeurs virtuels) interconnectés à l'aide de liens virtuels. Ces nœuds et liens virtuels sont assignés à un ensemble de nœuds et liens d'un réseau physique (*Voir* Figure 3.1). Un chemin physique est le chemin entre deux nœuds physiques. Ce chemin peut inclure un ou plusieurs liens physiques.

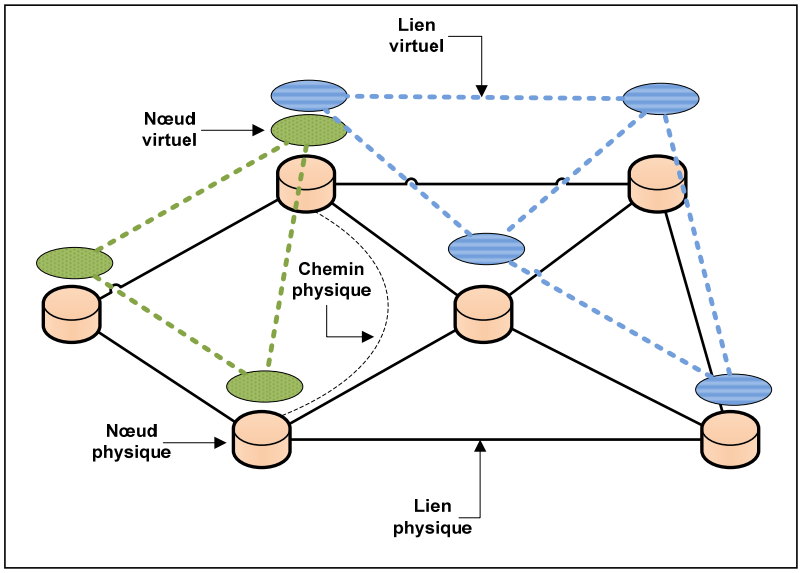

Figure 3.1 Intégration de deux réseaux virtuels dans un réseau physique partagé
Reproduite et adaptée avec l'autorisation de Houidi, Louati et Zeghlache (2008)

3.1.2 Modélisation du réseau physique

Le réseau physique peut être modélisé à l'aide d'un graphe pondéré non orienté : $G_s = (N_s, L_s, C_s^n, C_s^l)$, où N_s est l'ensemble des nœuds physiques et L_s est l'ensemble des liens physiques(Chowdhury et Boutaba, 2010). Chaque lien physique $l_s(i, j) \in L_s$ connecte les deux nœuds physiques i et j. C_s^n et C_s^l représentent respectivement l'ensemble des contraintes des nœuds et des liens physiques. Chaque nœud physique $n_s \in N_s$ est associé à une valeur de CPU $c(n_s)$ qui correspond à sa capacité disponible en CPU. Chaque lien physique $l_s \in L_s$ est associé à une valeur de bande passante $b(l_s)$ qui correspond à sa quantité de bande passante disponible. Nous considèrerons aussi pour chaque lien physique deux attributs additionnels $d(l_s)$ et $pp(l_s)$ qui représentent respectivement la tolérance de ce lien en terme de délai et de perte de paquets. Un chemin $P_s(n_s, m_s)$ est un ensemble de liens qui lient les deux nœuds physiques n_s et m_s.

Nous définissons aussi pour chaque nœud physique n_s un facteur d'émission de CO_2 $E_{mf}(loc(n_s))$. Selon (Corp., 2010), cet attribut dépend de la localisation géographique $loc(n_s)$ du nœud physique ns et quantifie l'émission en CO_2 de ce nœud qui résulte de sa

consommation d'électricité. Cette consommation est en fonction du type d'électricité (Ex. : hydraulique, éolienne, etc.) utilisé pour alimenter ce nœud physique. Le Tableau 3.1 présente la valeur de ce facteur dans les différentes provinces canadiennes(Corp., 2010).

Tableau 3.1 Facteur d'émission de CO_2 par province canadienne
Tiré de Corp (2010)

Province	t/Kwh
AB	0.000930
BC	0.000020
MB	0.000010
NB	0.000366
NL	0.000015
NS	0.000549
NT	0.000249
NU	0.000249
ON	0.000180
PEI	0.000192
QC	0.000006
SK	0.000810
YK	0.000249

3.1.3 Modélisation du réseau virtuel

Le réseau virtuel est aussi représenté par un graphe pondéré non orienté : $G_v = (N_v, L_v, C_v^n, C_v^l)$ (Chowdhury et Boutaba, 2010). Les nœuds et liens virtuels ont les mêmes contraintes que les nœuds et liens physiques. Le Tableau 3.2 liste l'ensemble de contraintes associées aux réseaux physique et virtuel. En plus, nous associons à chaque lien virtuel un indicateur $s(l_v)$ qui correspond au type de contrainte : 0 quand il s'agit d'une contrainte de délai (Ex. : lien utilisé pour le transfert de la voix et de la vidéo en temps réel), 1 quand il s'agit d'une contrainte de perte de paquets (Ex. : lien utilisé pour le transfert de données) et 3 si aucune

contrainte ne s'applique (ex. lien utilisé pour le transfert du trafic service au mieux « best effort »).

Tableau 3.2 Contraintes du réseau physique et du réseau virtuel

G_s	Réseau physique
N_s	Nœuds physiques
L_s	Liens physiques
C_s^n	Contraintes des nœuds physiques
$Loc(n_s)$	Localisation d'un nœud physique
$Emf(Loc(n_s))$	Émission en CO_2 pour un nœud n_s
C_s^l	Contraintes des liens physiques
P_s	Chemins physiques
G_v	Réseau virtuel
N_v	Nœuds virtuels
L_v	Liens virtuels
C_v^n	Contraintes des nœuds virtuels
C_v^l	Contraintes des liens virtuels
$s(l_v)$	Indicateur d'un lien virtuel l_v

3.1.4 Valeurs résiduelles des ressources physiques

L'intégration d'un réseau virtuel dans un réseau physique consiste à affecter des nœuds et des liens du réseau physique qui satisfont les exigences du réseau virtuel non seulement en termes de ressources requises (c.-à-d., CPU au niveau du nœud et bande passante au niveau du lien) mais aussi en termes de contraintes telles que les pertes de paquets et les délais de bout en bout à travers les liens physiques. Après chaque intégration, il est nécessaire de calculer les ressources restantes (résiduelles) pour les nœuds physiques ainsi que pour les liens physiques. Ces ressources sont modélisées comme suit :

Supposons que $R_N(n_s)$ correspond à la quantité résiduelle (disponible) de CPU d'un nœud physique $n_s \in N_s$ et que $R_L(l_s)$ correspond à la quantité résiduelle de bande passante d'un lien physique $l_s \in L_s$.

$$R_N(n_s) = c(n_s) - \sum_{\forall n_v \to n_s} c(n_v) \quad (3.1)$$

$$R_L(l_s) = b(l_s) - \sum_{\forall l_v \to l_s} b(l_v) \quad (3.2)$$

La notation $n_v \to n_s$ veut dire que le nœud virtuel n_v est une instance virtuelle du nœud physique n_s. Respectivement, la notation $l_v \to l_s$ veut dire que le lien virtuel l_v est une instance virtuelle du lien physique l_s. $c(n_v)$ et $b(l_v)$ représentent respectivement la valeur de CPU du nœud virtuel n_v et la valeur de la bande pasante du lien virtuel l_v.

Nous pouvons aussi quantifier les ressources résiduelles R(P) d'un chemin physique $P \in P_s$. En effet, R(P)est la bande passante résiduelle minimale des liens composant le chemin P.

$$R(P) = \min_{l_s \in P}(R_L(l_s)) \quad (3.3)$$

Ainsi, quand un réseau physique reçoit une requête de création d'un réseau virtuel, il doit s'assurer que ses ressources disponibles peuvent satisfaire les contraintes exigées par cette dernière. Le cas échéant, cette requête est rejetée. Si la requête est acceptée, ses nœuds virtuels et ses liens virtuels sont affectés à un ensemble de nœuds et chemins physiques selon le modèle suivant :

$$M: G_v(N_v, L_v) \to G_s(N_s, P_s)$$

tels que :

$$c(n_v) \leq R_N(n_s) \quad (3.4)$$

$$b(l_v) \leq R_L(P) \quad (3.5)$$

avec

$$d(l_v) \geq d(P) sis(l_v) = 0 \quad (3.6)$$

$$pp(l_v) \geq pp(P) sis(l_v) = 1 \quad (3.7)$$

Les inégalités (3.6) et (3.7) permettent de s'assurer que les contraintes de délai et de perte de paquets au niveau d'un lien virtuel sont garanties lorsqu'on affecte ce dernier au chemin physique *P*.

3.1.5 Approche proposée

L'objectif principal de notre approche est de proposer un algorithme qui permet d'affecter les nœuds et les liens virtuels des requêtes aux nœuds et aux liens physiques du réseau substrat en respectant les contraintes spécifiées dans chaque requête. Aussi, nous visons à maximiser le revenu, à réduire le coût pour le FIV de chaque affectation de requête et à minimiser l'émission du CO_2 résultant de la consommation d'énergie des différents nœuds physiques utilisés.

Nous définissons le coût physique d'une requête virtuelle comme étant la somme des ressources physiques utilisées par les nœuds et liens de cette requête.

$$Co\hat{u}tP(G_v) = \sum_{l_v \in L_v} \sum_{l_s \in L_s} b(l_v, l_s) + \sum_{n_v \in N_v} c(n_v) \tag{3.8}$$

Le paramètre $b(l_v, l_s)$ indique la quantité de bande passante allouée par le lien physique l_s pour le lien virtuel l_v.

Chaque requête a aussi un coût virtuel qui est la somme de ses exigences en CPU et bande passante. Ce coût est le prix que le FIV demandera au client pour accepter sa requête.

$$Co\hat{u}tV(G_v) = \sum_{l_v \in L_v} b(l_v) + \sum_{n_v \in N_v} c(n_v) \tag{3.9}$$

Nous estimons que le revenu est le gain du client lorsqu'il utilise la virtualisation de réseaux pour héberger sa requête. Cette valeur représente la différence entre le coût physique et le coût virtuel d'une requête.

$$R(G_v) = CostP(G_v) - CostV(G_v) \tag{3.10}$$

Notre approche vise aussi à réduire l'émission du CO_2 résultant de la consommation d'énergie des différents nœuds physiques utilisés pour héberger les nœuds virtuels d'une requête donnée. Pour un nœud virtuel $n_v \in N_v$, sa consommation en CO_2 est quantifiée par le produit de son CPU et du facteur d'émission du nœud physique $n_s \in N_s$ utilisé pour l'héberger. La consommation en CO_2 pour une requête est donc donnée par :

$$G_{Em}(G_v) = \sum_{\forall n_v \to n_s} Emf(Loc(n_s)) \times c(n_v) \tag{3.11}$$

3.2 Approche hybride pour la gestion des ressources

3.2.1 Relation entre les principaux acteurs

Dans toute architecture de virtualisation de réseaux, les trois principaux acteurs sont : le FS, le FIV et le FIP (*Voir* Figure 3.2). Par contre, la relation entre ces différentes entités peut varier dépendamment de l'approche utilisée. Cette section décrit le rôle de ces acteurs dans le cas de l'approche hybride.

Fournisseur de service (FS)

Le fournisseur de service est une entité qui offre des services aux utilisateurs selon un contrat négocié entre les deux parties (FS et utilisateur). Ces services sont déployés à travers une infrastructure réseau fournie par le FIV. Ainsi, FS présente une requête de création d'une infrastructure réseau virtuel au FIV en spécifiant ses besoins en ressources réseau (ex. topologie du réseau virtuel, capacité des nœuds, bande passante au niveau des liens) et en

qualité de service (Ex. : délai bout à bout et la perte de paquets à travers le réseau virtuel). La gestion des ressources et de la qualité de service du réseau virtuel est déléguée au FIV.

Fournisseur d'infrastructure virtuelle (FIV)

Le fournisseur d'infrastructure virtuelle est composé de quatre entités : un nœud central (NC) et un ensemble de nœuds de gestion (NG). Le NC est une entité centrale qui a une vue d'ensemble sur toute l'infrastructure physique sous-jacente. Ses deux fonctions principales sont la décomposition de la requête reçue du FS en plusieurs sous-requêtes et l'envoi de ces dernières aux différents NGs (*Voir* Figure 3.2). L'approche de décomposition de requête de réseau virtuel proposée dans la littérature utilise la méthode de décomposition en étoile ((Zhu et Ammar, 2006), (Houidi, Louati et Zeghlache, 2008b)). En d'autres termes, le nœud de la requête avec la valeur de CPU disponible la plus élevée est considéré comme le nœud central de la sous-requête et les autres nœuds qui lui sont directement connectés sont ses nœuds secondaires. Dans le cadre de ce projet, nous nous basons sur une nouvelle approche basée sur les contraintes qui s'appliquent ou non au niveau des liens virtuels. Ainsi, nous supposons que les requêtes sont décomposées en trois types de sous-requêtes : avec contrainte de délai, avec contrainte de perte de paquet et sans contraintes. Pour simplifier notre analyse, nous supposons que chaque NG est responsable d'assigner un seul type de sous-requêtes au réseau physique. Nous admettons que le NG_1 est responsable des sous-requêtes avec contrainte de délai, le NG_2 est responsable des sous-requêtes avec contrainte de perte de paquets et le NG_3 est responsable des sous-requêtes sans contraintes (*Voir* Figure 3.2).

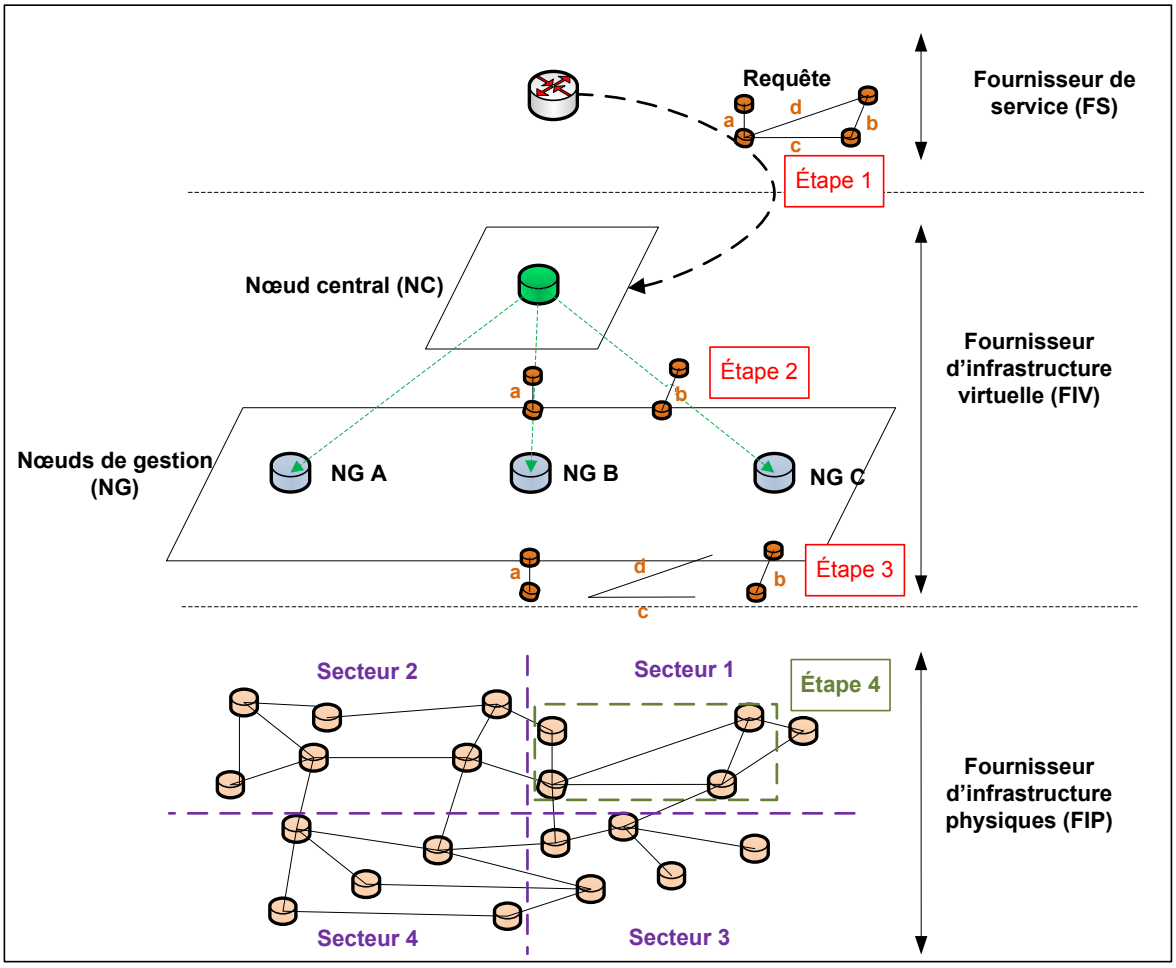

Figure 3.2 Approche hybride de base

La sélection de ces entités se fait ainsi : Le NC est hébergé par le nœud physique qui a le plus de ressources disponibles en termes de CPU et de bande passante. En effet, tel que (Minlan Yu, 2008) l'a proposé, ces ressources se calcule en multipliant la quantité de CPU disponible du nœud et la quantité de bande passante offerte par les liens physiques qui lui sont directement connectés. Cette relation est définie dans l'équation (3.12). Les nœuds NG_1, NG_2 et NG_3 sont hébergées par les nœuds physiques dont les ressources physiques (CPU et bande passante) sont les plus élevés après ceux du nœud NC. Ils arrivent respectivement en deuxième, troisième et/ou quatrième position dans le classement des nœuds en termes de ressources disponibles.

$$R_{N,L}(n_s, l_s) = c(n_s) \times \sum_{l_s \in L_s} b(l_s) \quad (3.12)$$

Fournisseur d'infrastructure physique (FIP)

Le FIP est une entité responsable du déploiement de l'infrastructure physique et de la gestion de ses ressources physiques. L'infrastructure physique est divisée en quatre secteurs (*Voir* Figure 3.2). Chaque secteur est associé à un facteur d'émission $E_{mf}(\mathrm{Loc(n_s)})$. Ce facteur quantifie l'émission en CO_2 qui résulte de la consommation d'électricité des différents nœuds de ce secteur. Nous admettons que le secteur 1 est le secteur le plus propre ensuite arrivent dans l'ordre en terme d'énergie la moins propre les secteurs 2, 3 et 4. L'affectation des nœuds et des liens virtuels aux nœuds et liens physiques est décrite dans les trois approches hybrides présentées dans les sous-sections suivantes

3.2.2 Approche hybride de base

Dans cette section, nous allons décrire le protocole de communication permettant aux différentes entités intervenantes dans notre architecture d'échanger des messages afin de mieux gérer le processus d'allocation de ressources. Ensuite, nous allons présenter les différentes étapes de ce processus dans une architecture de virtualisation de réseaux utilisant l'approche hybride de base. Ces étapes sont les suivants: la réception et mise en file d'attente des requêtes, la décomposition des requêtes, l'affectation des nœuds et l'affectation des liens.

Description générale

Tel que décrit ci-dessus, le FS envoie au FIV une ou plusieurs requêtes de création de réseaux virtuels. Pour simplifier notre analyse, nous modélisons chaque requête par un ensemble de nœuds et de liens virtuels. Les contraintes d'une requête correspondent aux contraintes des nœuds et des liens constituant le réseau virtuel.

La Figure 3.2 illustre les différentes étapes de traitement d'une requête aux niveaux FS, FIV et FIP. La première étape correspond à la transmission des requêtes du FS au FIV. Chaque requête est reçue par le NC du FIV (*Voir* Figure 3.2: étape 1). Afin d'alléger le traitement de ces requêtes au niveau FIV et FIP, le NC divise la requête en plusieurs sous-requête selon le type de contrainte identifiées au niveau des liens virtuels. Chaque sous-requête représente un sous-réseau ayant une topologie réseau bien défini et un nombre de nœuds et de liens fixes. Ainsi, le NC groupe les nœuds dont les liens ont le même indicateur $s(l_v)$.

Ces sous-réseaux sont ensuite envoyés aux NGs correspondants (*Voir* Figure 3.2: étape 2). Dès la réception d'une requête du NC, le NG analyse les contraintes spécifiées dans cette requête et consulte sa base de données afin de vérifier si le secteur 1 (secteur utilisant l'énergie électrique la plus propre) a assez de ressources disponibles pour satisfaire les contraintes de cette sous-requête.

Deux cas sont possibles :

1) s'il n'y a pas assez de ressources dans ce secteur, le NG consulte la base de données à nouveau jusqu'à ce qu'il identifie le secteur ayant les ressources nécessaires pour héberger la requête. Le cas échéant, cette requête est rejetée ;

2) s'il y a assez de ressources dans le secteur1, le NG utilise alors des algorithmes d'optimisation de ressources afin d'affecter efficacement les différents nœuds et liens virtuels. Par exemple, l'algorithme du k plus court chemin est utilisé pour trouver le meilleur chemin pour héberger un lien virtuel (Voir Figure 3.2: étape 3).

Une fois tous les sous-réseaux virtuels, d'une requête, affectés à un ou plusieurs secteurs, le NC se charge de les lier ensemble (*Voir* Figure 3.2: Étape 4) pour constituer le réseau virtuel demandé par le FS. Dans l'exemple illustré par la figure 3.2, à l'étape 4, les deux sous-requêtes affectées ont été liées ensemble avec les deux liens manquants (lien c et lien d).

Protocole de communication

Tout comme l'approche proposée par (Houidi, Louati et Zeghlache, 2008b), le fonctionnement de notre architecture se base sur un échange asynchrone de six types de messages entre le FS, le NC et les NGs. Ces messages sont utilisés par ces entités pour s'échanger les informations concernant le processus d'affectation des différentes requêtes au réseau physique. Le rôle de chaque message est décrit dans ce qui suit:

REQ(G_v) : Ce message est envoyé du FS au NC comme requête pour la création d'un réseau virtuel G_v. Il décrit le réseau virtuel selon les attributs listés dans le Tableau 3.2.

SUBREQ($G_v(i)$) : Après avoir reçu une requête G_v d'un FS, le NC la décompose en plusieurs sous-requêtes. Ensuite, il envoie un message SUBREQ(G_v(i)) au NG correspondant pour la création d'un sous-réseau $G_v(i)$.

SUCCESS($G_v(i)$) : Ce message est envoyé par le NG au NC pour l'informer que le sous-réseau $G_v(i)$est créé et intégré avec succès dans l'infrastructure physique ou que le lien entre les sous-réseaux est crée et intégré avec succès. Ce message est aussi envoyé du NC au FS pour l'informer du succès de la création du réseau virtuel G_v.

FAILURE($G_v(i)$) :Le NG envoie ce message au NC pour l'informer qu'il n'a pas trouvé assez de ressources disponibles dans le réseau physique pour satisfaire les contraintes spécifiées dans la requête SUBREQ $G_v(i)$. Le NC place alors la sous-requête dans sa file d'attente afin qu'il puisse la renvoyer au NG pour une deuxième tentative après l'écoulement d'un délai t.

UPDATE (G_s) : Suite à l'affectation d'un réseau virtuel $G_v(i)$ au réseau physique, les ressources disponibles des liens et des nœuds physiques utilisés diminueront. Ce message est envoyé par le NG qui a été responsable de cette affectation aux deux autres NGs pour les

informer des nouvelles valeurs de ressources disponibles afin qu'ils mettent à jour leurs bases de données respectives.

LINKSUBREQ($G_v(i)$) :Ce message est envoyé par le NC aux NGs pour reconstituer le réseau virtuel demandé par le FS à partir des sous-réseaux créés par les NGs. Ce message inclut ces sous-réseaux ainsi que l'information nécessaire pour établir les liens entre les différents sous-réseaux

Scénarios possibles

Afin de mieux comprendre le rôle des messages décrits auparavant, nous considérons l'exemple suivant. Supposons que le FS envoie au NC une requête $G_v$dont les liens sont sensibles au délai et à la perte de paquets. Dans ce cas, seulement le NG_1 et le NG_2 seront sollicités pour héberger les nœuds et liens de cette requête. Pour simplifier l'exemple, nous considérons que cette requête sera décomposée en seulement deux sous-requêtes. Dans un premier scénario, les NGs sollicités réussissent à trouver les ressources nécessaires pour héberger les deux sous-requêtes. Dans un deuxième scénario, seulement le NG_1 réussira à affecter la sous-requête qu'il a reçue. Dans la section qui suit, nous expliquerons en détail les messages échangés entre le FS, le NC et les NGs pour chaque scénario.

- **Affectation de la requête réussie**

  La Figure 3.3 illustre les messages échangés pour une affectation de la requête réussie. Ainsi, le FS envoie un message REQ(G_v) au NC afin de lui acheminer la requête G_v. Après la décomposition de cette requête en deux sous-requêtes $G_v(1)$ et $G_v(2)$, le NC envoie les messages SUBREQ($G_v(1)$) et SUBREQ($G_v(2)$) respectivement à NG_1 et à NG_2. Ces messages servent à acheminer les sous-requêtes selon le type de contraintes identifiées aux NGs correspondants. Ainsi, le NG_1 reçoit la sous-requête $G_v(1)$ et affecte les nœuds et les liens virtuels de cette dernière à un ensemble de nœuds et liens

physiques. Il envoie ensuite un message SUCCESS au NC afin de l'informer du succès de l'intégration du sous-réseau $G_v(1)$. Puis, le NG1 envoie un message UPDATE à chacun des nœuds NG2 et NG3 pour qu'ils mettent à jour leurs bases de données. Le NG2 effectue la même procédure que NG1 lorsqu'il reçoit la sous-requête $G_v(2)$. Sauf qu'il envoie à la fin un message UPDATE à NG1 et à NG3 (Voir Figure 3.3).

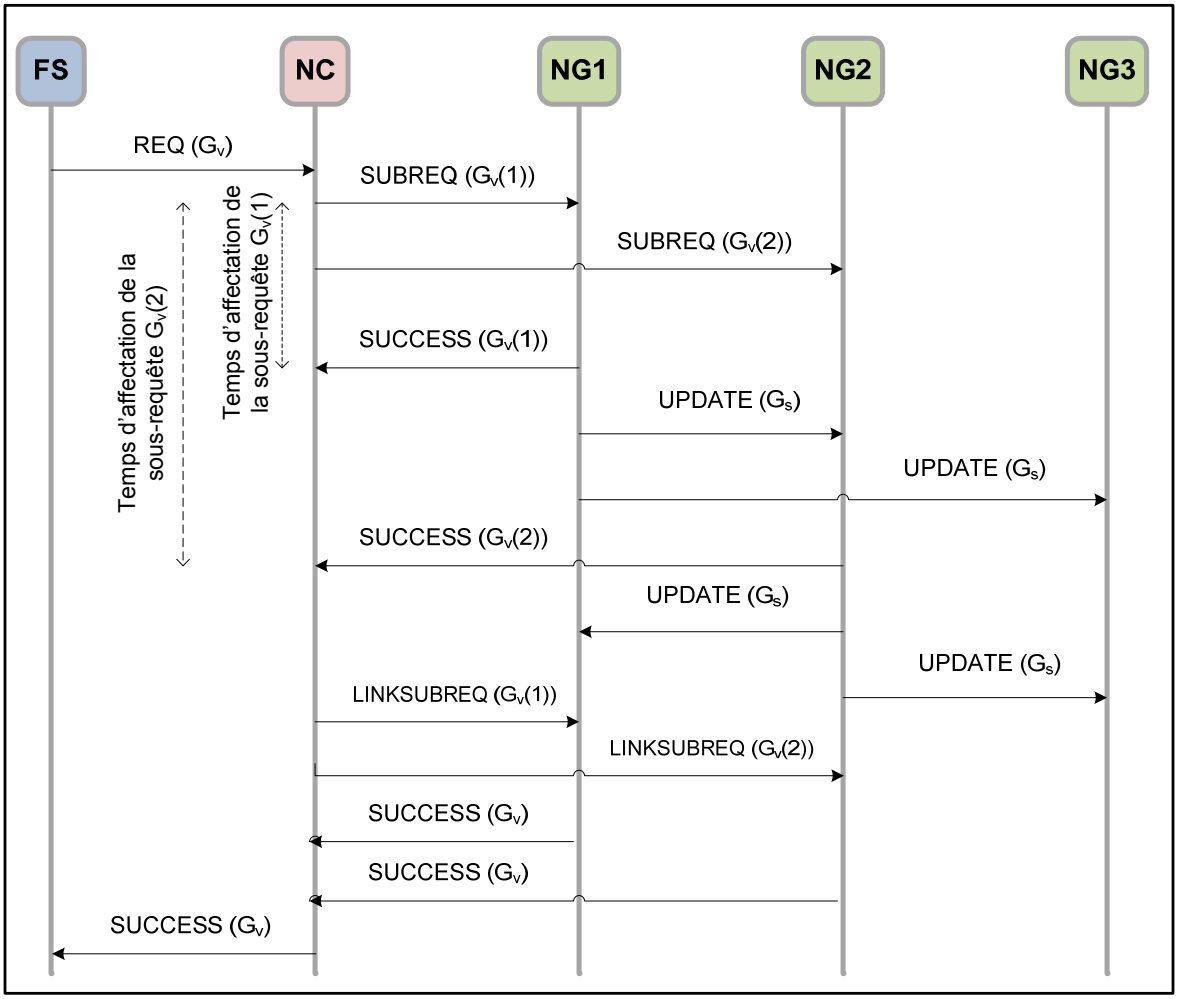

Figure 3.3 Messages échangés lors d'un succès d'affectation

- **Échec de l'affectation de la requête**

La Figure 3.4 illustre les messages échangés lors d'un échec de l'affectation d'une requête. Dans ce scénario, le NG2 ne trouve pas assez de ressources disponibles pour héberger la sous-requête $G_V(2)$. Il envoie donc un message FAILURE au NC pour l'informer de cet échec. Ce dernier, met alors cette sous-requête dans sa file d'attente et après un délai t, il réachemine la sous-requête au NG2 pour une deuxième tentative (Voir Figure 3.4). Il faut noter que chaque sous-requête peut être gardée seulement durant un temps t dans la file d'attente. En effet, après l'écoulement du temps t, si la sous-requête n'a pas été hébergée, elle sera supprimée.

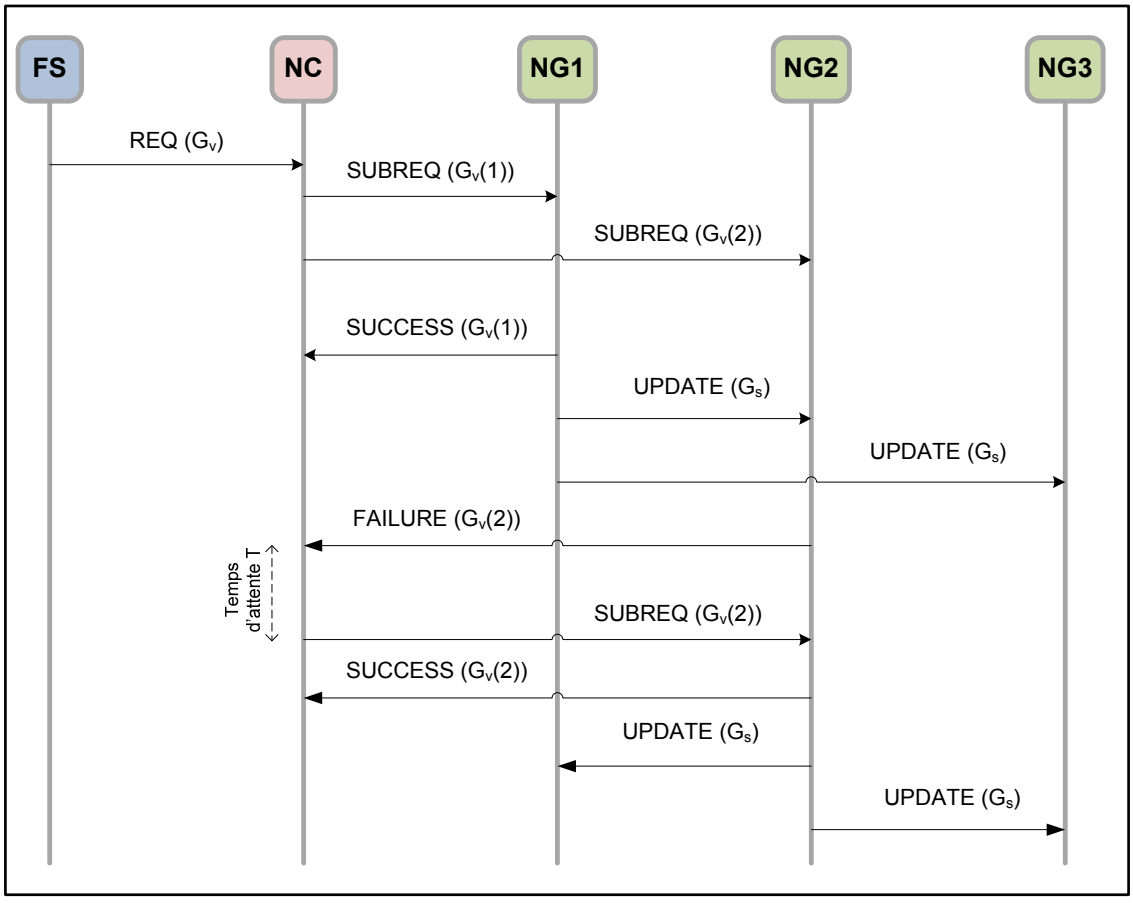

Figure 3.4 Messages échangés lors d'un échec d'affectation

La description détaillée de ces messages n'est pas abordée dans le cadre de ce mémoire. Nous nous intéressons, dans ce mémoire, à la modélisation de cette approche en définissant et en validant les quatre phases qui la composent. Ces phases sont :

1) réception et mise en file d'attente des requêtes ;
2) décomposition des requêtes ;
3) affectation des nœuds ;
4) affectation des liens.

De plus, nous proposons d'analyser l'effet de la classification et la priorité ainsi que la localisation sur la gestion de ressources dans ce type d'environnement.

Afin de mieux comprendre le fonctionnement de notre approche, nous analyserons ces phases en détail dans les prochaines sous-sections de ce chapitre.

Réception et mise en file d'attente des requêtes

La première phase de notre approche est la phase de réception et la mise en attente des requêtes. Pour modéliser l'arrivée des requêtes au NC, nous supposons que toutes les requêtes sont stockées dans une base de données. Les requêtes sont sélectionnées d'une manière aléatoire et arrivent au NC avec un taux λ. Les requêtes sélectionnées sont ensuite placées dans une file d'attente de longueur k au niveau du NC. Ces requêtes sont traitées en utilisant la méthode FIFO avec un taux de service μ ((Jordan, 2010) et (Meempat et Sundareshan, 1993)) . Rappelons que pour qu'un système soit stable, il faut satisfaire la condition définie par l'inégalité (3.14).

$$\rho = \frac{\lambda}{\mu} \tag{3.13}$$

$$\rho < 1 \tag{3.14}$$

Le paramètre ρ est le taux d'utilisation du système.

Dans l'exemple illustré à la Figure 3.5, la base de données est composée de 100 requêtes. Les requêtes arrivent au niveau NC avec un taux λ=2 requêtes/s et sont traitées avec un taux de service μ=4 requêtes/s. Dans ce cas, le taux d'utilisation du système est de 0.5. La longueur de la file d'attente du NC est de 8 requêtes.

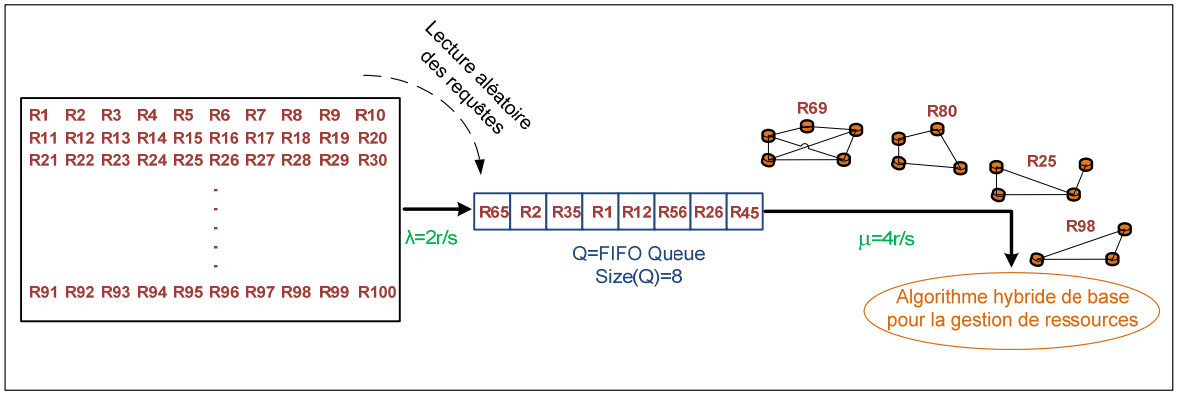

Figure 3.5 Réception et mise en file d'attente des requêtes dans l'algorithme de base

Décomposition des requêtes

Chaque requête reçue par le NC est décomposée en plusieurs sous-requêtes chacune d'elles représente un sous-réseau du réseau virtuel de cette requête afin d'alléger leurs traitements. Cette décomposition est basée sur les types de contraintes identifiées au niveau de chaque requête. En effet, dans notre travail, un sous-réseau est une topologie dont les liens ont le même indicateur $s(l_v)$.

La Figure 3.6 illustre un exemple de décomposition de requête. Initialement, le réseau virtuel est composé de six nœuds (a, b, c, d, e et f) et six liens ($l_v(a, b)$, $l_v(b, c)$, $l_v(c, d)$, $l_v(c, e)$, $l_v(d, e)$ et $l_v(d, f)$). Chaque lien est associé à un indicateur $s(l_v)$qui spécifie le type de contrainte qui caractérise ce lien. L'algorithme de décomposition regroupe dans un même sous-réseau tous les liens qui sont directement connectés et ayant la même valeur de

l'indicateur $s(l_v)$. Dans cet exemple, le premier sous-réseau généré est composé de trois nœuds (a, b et c) et 2 liens (l_v(a, b), l_v(b, c)). La contrainte caractérisant ces liens est la perte de paquets (Par exemple, ce sous-réseau est utilisé pour offrir un service de sauvegarde de données). La figure 4.6 illustre trois types d'indicateurs : contrainte perte de paquet avec l_v(a, b)=1, l_v(b, c)=1 et l_v(e, d)=1, contrainte de délai avec l_v(c, d)=0, l_v(c, e)=0et sans contraintes avec l_v(d, f)=2. Durant cette phase de décomposition, certains nœuds sont marqués en noir pour indiquer qu'ils ont été déjà affectés à un sous-réseau. Ces nœuds ne seront pas considérés par l'algorithme lors de la phase d'affectation des nœuds virtuels aux différents nœuds physiques. Par exemple, pour la sous-requête 2, le nœud c est marqué en noir puisque ce nœud fait partie de la sous-requête 1 et que l'algorithme d'affectation des nœuds l'affectera à un nœud physique durant le traitement de la première sous-requête (*Voir* Figure 3.6)

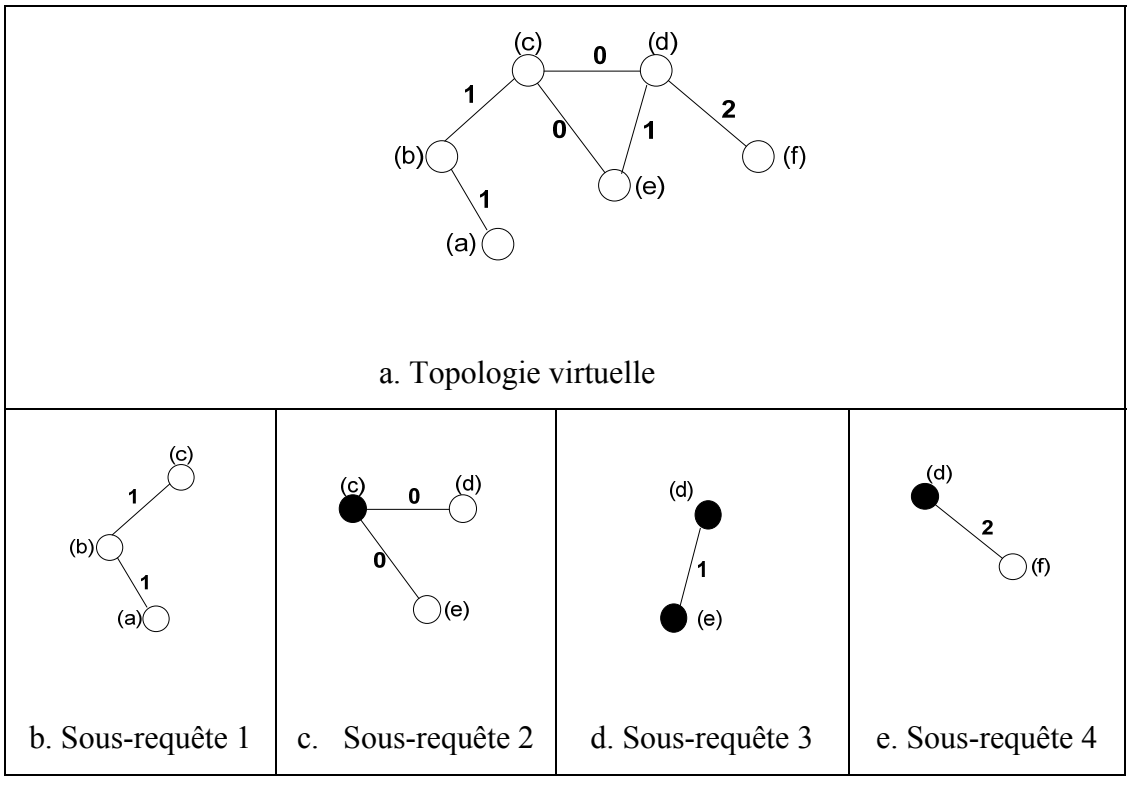

Figure 3.6 Exemple de décomposition d'une requête

Algorithme de décomposition des requêtes

Le paramètre d'entrée de cet algorithme est la requête envoyée par le FS. Cette requête est modélisée par une topologie qui comprend un ensemble de nœuds virtuels N_v interconnectés par des liens virtuels L_v. Pour chaque nœud et lien virtuel, le FS impose un ensemble de contraintes C_v^n et C_v^l. Le résultat de cet algorithme sera un ou plusieurs sous-réseaux virtuels. L'algorithme commence par une section d'initialisation (*Voir* Algorithme 3.1: lignes 1, 2 et 3). La requête initiale est affectée à la topologie résiduelle G_r. Cette dernière est une variable utilisée pour stocker l'ensemble de nœuds et de liens qui n'ont pas été affectés à une sous-requête. L'algorithme est répété tant qu'il existe des liens dans le réseau G_r. D'abord, le NC affecte le premier lien $l_v(x, y)$ de la liste ainsi que les nœuds x et y formant ses extrémités au premier sous-réseau $G_v(1)$ (*Voir* Algorithme 3.1: lignes 7 et 8). Puis, il marque ces deux nœuds à l'aide de la fonction Tag() afin de désigner qu'ils font déjà partie d'un sous-réseau (*Voir* Algorithme 3.1: ligne 9). Ensuite, il sélectionne un second lien du réseau (*Voir* Algorithme 3.1: ligne 10) et vérifie s'il est directement connecté à un des liens déjà sélectionnés et sison indicateur $s(l_v)$ est le même que celui du premier nœud sélectionné (*Voir* Algorithme 3.1: ligne 11). Si c'est le cas, il les regroupe dans un même sous-réseau (*Voir* Algorithme 3.1: lignes 11 et 12) et marque les nouveaux nœuds x'et y'.Une fois tous les liens du réseau G_r sont vérifiés, le premier sous-réseau $G_v(1)$ est formé, ses liens seront supprimés du réseau résiduel (*Voir* Algorithme 3.1: ligne 17). Le NC refait le même traitement jusqu'à l'affectation de tous les liens du réseau résiduel.

Algorithme 3.1 Décomposition des requêtes

Entrée: $G_v = (N_v, L_v, C_v^n, C_v^l)$ *// Le réseau virtuel*
Sortie: $G_v(i) = (N_v(i), L_v(i), C_v^n(i), C_v^l(i))$ *// Un sous-réseau virtuel*

1. $G_r = G_v$
2. $i = 1$
3. $G_V(i) = 0$
4. **while** $(L_r \neq \emptyset)$ **do**

```
5.        l_v(x,y) ∈ L_r
6.        x ∈ N_r , y ∈ N_r
          // Le premier lien avec ses extrémités est assigné au sous-réseau G_V(i)
7.        L_V(i) = L_V(i) ∪ l_v(x,y)
8.        N_V(i) = N_V(i) ∪ {x,y}
9.        Tag {x,y}
10.       for (l_v(x',y') ∈ L_r) do
          // Pour chaque lien du réseau résiduel, vérifier s'il a le même indicateur que
          // l_v(x,y) et s'il lui est directement connecté
11.         if ((s(x',y') = s(x,y) and ((x' ∈ N_V(i)) or (y' ∈ N_V(i))) ) do
            // Le lien et ses extrémités sont affectés à G_V(i)
12.             L_V(i) = L_V(i) ∪ l_v(x',y')
13.             N_V(i) = N_V(i) ∪ {x',y'}
14.             tag {x',y'}
15.         end if
16.       end for
17.       L_r = L_r − L_v(i)
18.       i = i + 1
19.   end while
```

Une fois la requête subdivisée en sous-requête, le NC envoie ces dernières aux NGs correspondants. Chaque NG qui reçoit un sous-réseau virtuel se charge de lui affecter les ressources physiques nécessaires : nœuds et liens physiques. Les deux sous-sections suivantes décrivent respectivement l'affectation des nœuds et l'affectation des liens.

Affectation des nœuds

L'affectation d'un réseau virtuel à l'infrastructure physique se fait en deux étapes : l'affectation des nœuds puis l'affectation des liens. La première étape consiste alors à affecter les nœuds d'un réseau virtuel aux nœuds du réseau physique.

L'infrastructure physique est divisée en plusieurs secteurs. Le NG tente dans un premier temps d'héberger les nœuds dans le secteur avec la plus petite valeur de coefficient

d'émission en CO_2. S'il n'y pas assez de ressources disponibles en CPU dans cette zone, le NG consulte un second secteur et ainsi de suite jusqu'à trouver les ressources nécessaires dans l'un des quatre secteurs de l'infrastructure physique. Une fois le secteur trouvé, le NG affectera chaque nœud virtuel au nœud physique dont la valeur de CPU disponible est la plus proche de la valeur de CPU requis.

Afin d'expliquer ce dernier point, nous illustrons dans la Figure 3.7 un scénario d'affectation de nœuds d'une sous-requête dans un des secteurs du réseau physique. La sous-requête en question est composée de trois nœuds a, b et c qui requièrent respectivement huit, six et dix unités de CPU. Ces valeurs sont encadrées au niveau de chaque nœud de la requête (*Voir* Figure 3.7). L'algorithme commence par affecter le nœud qui demande le plus de ressources des trois nœuds de la requête, à savoir le nœud b. Le choix du nœud physique est basé sur ses ressources en CPU. En effet, l'algorithme choisit le nœud dont la capacité de CPU disponible est suffisante et minimale (la plus petite valeur de CPU). Dans notre exemple, b est affecté au nœud E de l'infrastructure physique parce que 20 est la plus petite et la plus proche valeur qui satisfait le besoin en CPU à savoir la valeur de 10. La même méthode est utilisée pour affecter les deux autres nœuds de la sous-requête (a et c). Les travaux de recherche dans le domaine optent pour l'affectation du nœud physique ayant la plus grande capacité de CPU. Cependant, cette stratégie ne permet pas d'utiliser efficacement les ressources disponibles de l'infrastructure physique. En effet, supposons que l'algorithme affecte le nœud b au nœud C. Une fois le nœud b est affecté, la capacité résiduelle au niveau du nœud C est de 40. Supposons qu'un nœud virtuel d'une deuxième sous-requête reçue par le NG exige une capacité de 50, l'infrastructure physique ne possède pas un nœud physique d'une telle capacité, alors que plusieurs ressources sont inutilisées.

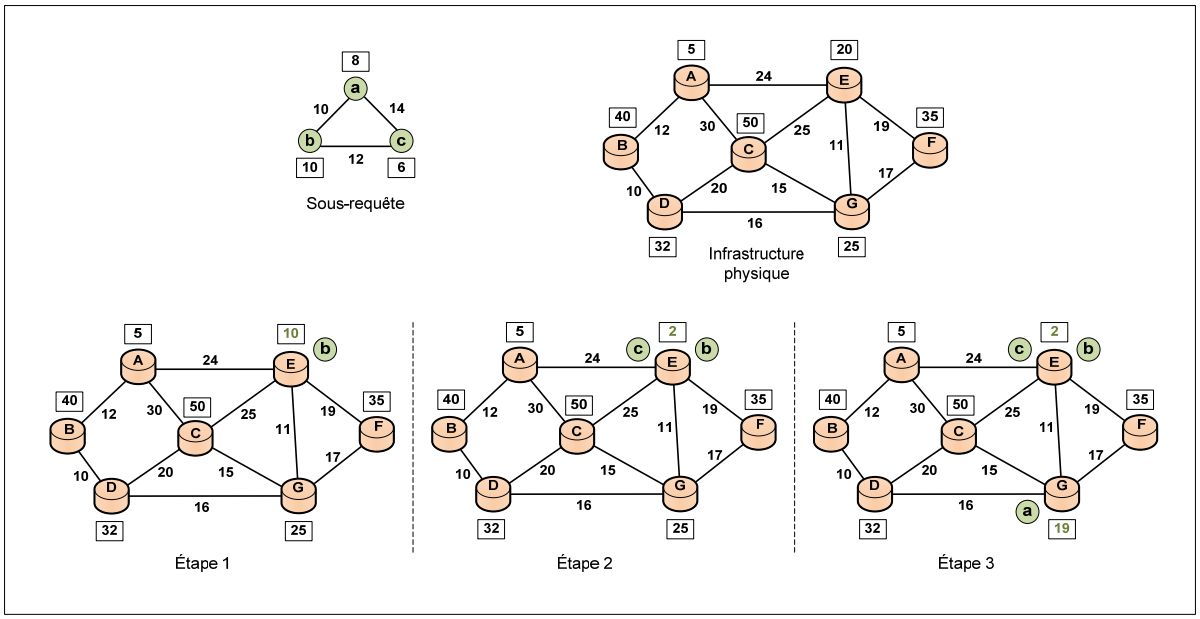

Figure 3.7 Exemple d'affectation de nœuds d'une requête

Algorithme d'affectation des nœuds

Le paramètre d'entrée de cet algorithme est un sous-réseau $G_v(i)$ ainsi que la topologie physique G_s. L'objectif de cet algorithme est d'affecter les nœuds virtuels du sous-réseau $G_v(i)$ aux nœuds physiques du réseau G_s. Cette affectation se base sur la disponibilité des ressources en CPU et la localisation des nœuds physiques. Chaque nœud physique utilisé aura une nouvelle valeur de CPU résiduelle disponible. Cette valeur est calculée selon l'équation (3.1).Le paramètre de sortie de cet algorithme est donc le réseau physique G_s avec les ressources résiduelles disponibles en CPU.

L'algorithme est exécuté par le NG dès la réception d'une sous-requête $G_v(i)$ du NC (*Voir* Algorithme 3.2: ligne 1). Le NG commence par sélectionner un nœud virtuel n_v de l'ensemble des nœuds $N_v(i)$ de la sous-requête (*Voir* Algorithme 3.2 : ligne 3). Il effectue ensuite les recherches des ressources nécessaires pour satisfaire les contraintes du nœud n_v dans le premier secteur. Tout comme dans l'algorithme proposé par (Houidi, Louati et Zeghlache, 2008b), nous utilisons deux fonctions prédéfinies Sort() et Head(). La fonction

Sort() permet de tirer un ensemble de nœuds selon leurs ressources disponibles en CPU. Cette fonction est utilisée dans notre algorithme afin de trier les nœuds physiques du premier secteur (*Voir* Algorithme 3.2: ligne 5). La fonction Head() est ensuite utilisée pour extraire le premier nœud de la liste. Si les ressources de ce nœud sont suffisantes pour satisfaire les besoins du nœud n_v, ce dernier sera alors hébergé dans le secteur 1 (*Voir* Algorithme 3.2: ligne 6). Le nœud physique choisi par notre approche est celui dont la valeur de CPU disponible est la plus proche de celle exigée par n_v (*Voir* Algorithme 3.2: lignes 7, 8, 9, 10, 11 et 12). Le NG sélectionne ensuite un autre nœud virtuel et réexécute les mêmes étapes (*Voir* Algorithme 3.2: de la ligne 2 à ligne 12) traitement et ainsi de suite jusqu'il n'y aura plus de nœuds virtuels à traiter. Notons que si ce secteur ne peut pas satisfaire les besoins du nœud n_v, les autres secteurs seront parcourus un par un jusqu'à trouver les ressources nécessaires dans l'un de leurs nœuds physiques. Le cas échéant, la requête est rejetée.

Algorithme 3.2 Affectation des nœuds dans l'algorithme de base

Entrées : $G_v(i) = (N_v(i), L_v(i), C_v^n(i), C_v^l(i))$ *// Un sous-réseau virtuel*
$G_s = (N_s, L_s, C_s^n, C_v^l)$ *// Le réseau physique*
Sortie: $G_s = (N_s, L_s, C_s^n, C_v^l)$ *// Le réseau physique*

1. **Upon** receiving $G_v(i)$ **do**
2. **while** $(N_v(i) \neq \emptyset)$ **do** *// Tant que il y a des nœuds virtuels non assignés*
3. **for** $\forall n_v \in N_v(i)$ **do**
4. $j = 1$ *// Indice pour le secteur 1*
5. **Sort** $(N_s(j))$ *// Trier les nœuds du secteur 1 selon la valeur de CPU*
// S'il y a assez de ressources dans le secteur 1
6. **if** $(c(\mathbf{Head}(N_s(j))) \geq c(n_v))$
// Trouver l'indice du nœud physique avec la plus proche valeur de CPU
7. $\forall n_s \in N_s(j)$
8. $k = 0$
9. **while** $(c(n_s)) \geq c(n_v))$
10. $k = k + 1$
11. **end while**
12. $n_v \rightarrow n_s$ *// n_v est hébergé dans le nœud n_s d'indice k*
// Mettre à jour la valeur du CPU de n_s
13. $c(n_s) = c(n_s) - c(n_v)$
// Supprimer le nœud n_v de la liste des nœuds à affecter

```
14.                         N_v(i) = N_v(i) − n_v
15.                     else
16.                         j = j + 1  // Passer au prochain secteur
17.                         if (j = 5)  // Les 4 secteurs ont été balayés
18.                             exit  // Pas de ressources disponibles pour  n_v
19.                         else
20.                             go to  5  // Aller à la 5ème ligne
21.                         end if
22.                     end if
23.                 end for
24.     end while
```

Affectation des liens

La seconde étape de l'algorithme d'affectation d'un sous-réseau virtuel est l'affectation de ses liens. Rappelons qu'un lien virtuel l_v possède comme attribut sa bande passante $b(l_v)$. Il peut être un lien avec contrainte de délai $d(l_v)$ ou de perte de paquets $pp(l_v)$ ou sans contraintes. L'objectif de cet algorithme est de trouver pour chaque lien virtuel de la sous-requête un chemin physique dont les ressources physiques disponibles pourront satisfaire les contraintes de ce lien. Afin de choisir les chemins les moins gourmands en termes de bande passante, nous avons utilisé l'algorithme du k plus court chemin.

Afin de mieux expliquer cette approche, nous illustrons dans la Figure 3.8 un nouveau scénario d'affectation de deux sous-requêtes dans un des secteurs du réseau physique. Il faut noter que l'infrastructure physique utilisée dans cette figure est différente de celle utilisée dans la figure 3.7. La notation w/x/y/z/ sur chaque lien virtuel veut dire que ce dernier exige une bande passante de valeur w, une tolérance de délai d'une valeur x et une tolérance à la perte de paquets d'une valeur y. Le paramètre z correspond à l'attributs(l_v) qui indique le type de lien avec ou sans contraintes.

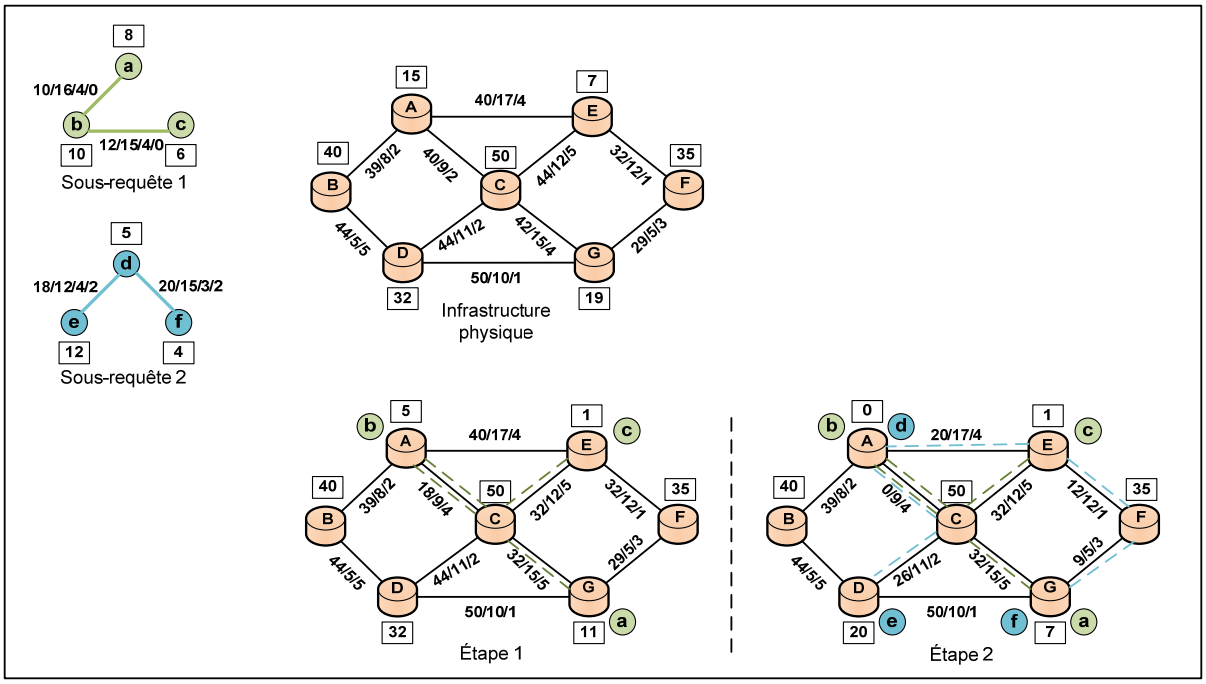

Figure 3.8 Exemple d'affectation de deux sous-requêtes

La première étape consiste à affecter la sous-requête 1. En effet, les nœuds b, a et c sont d'abord affectés respectivement aux nœuds physiques A, G et E selon l'algorithme d'affectation de nœuds expliqué précédemment. Ensuite les liens virtuels $l_v(a,b)$ et $l_v(b,c)$ sont affectés respectivement aux chemins physiques $P(G-C-A)$ et $P(A-C-E)$. Pour choisir ces chemins, nous avons utilisé l'approche suivante :

1) trouver le premier plus court chemin entre les nœuds G et A en utilisant la bande passante comme valeur de coût de chaque lien physique : Ce chemin est le $P(G-C-A)$ avec un coût de 40+42=82 ;
2) vérifier si chaque lien du chemin $P(G-C-A)$ dispose d'une valeur de bande passante supérieure ou égale à dix, valeur exigée par le lien $l_v(a,b)$: la condition est vérifiée ;
3) le lien virtuel $l_v(a,b)$ est sensible au délai car $s(l_v(a,b))=0$. Vérifier alors si chaque lien du chemin $P(G-C-A)$dispose d'une valeur de tolérance de délai inférieure ou égale à 16, valeur exigée par le lien$l_v(a,b)$: la condition est vérifiée donc le chemin $P(G-C-A)$ est validé ;

4) mettre à jour la disponibilité en bande passante des liens physiques utilisés selon l'équation (3.2) ;
5) trouver le premier plus court chemin entre les nœuds A et E en utilisant toujours la bande passante comme valeur de coût de chaque lien physique : Ce chemin est le $P(A-E)$ avec un coût de 40 ;
6) vérifier si chaque lien du chemin $P(A-E)$ dispose d'une valeur de bande passante supérieure ou égale à 12, valeur exigée par le lien $l_v(b,c)$: la condition est vérifiée ;
7) le lien virtuel $l_v(b,c)$ est sensible au délai car $s(l_v(b,c)) = 0$. Vérifier alors si chaque lien du chemin $P(A-E)$dispose d'une valeur de tolérance de délai inférieure ou égale à 15, valeur exigée par le lien$l_v(b,c)$: la condition n'est pas satisfaite donc le chemin est rejeté ;
8) trouver le deuxième plus court chemin entre les nœuds A et E: Ce chemin est le $P(A-C-E)$ avec un coût de 84 ;
9) vérifier si chaque lien du chemin $P(A-C-E)$dispose d'une valeur de bande passante supérieure ou égale à 12, valeur exigée par le lien $l_v(b,c)$: la condition est vérifiée ;
10) vérifier si chaque lien du chemin $P(A-C-E)$ dispose d'une valeur de tolérance de délai inférieure ou égale à 15, valeur exigée par le lien $l_v(b,c)$: la condition est vérifiée donc le chemin $P(A-C-E)$ est validé ;
11) mettre à jour la disponibilité en bande passante des liens physiques utilisés selon l'équation (3.2).

Les étapes de 1 à 11 sont répétées pour affecter le lien virtuel $l_v(b,c)$ au le chemin physique $P(A-C-E)$.

La même approche est ensuite utilisée pour affecter la sous-requête 2. Notons qu'il n'y a pas de contraintes au niveau des liens de cette sous-requête $(s(l_v(d,e)) = s(l_v(d,f)) = 2)$. Ainsi, seule la condition de la bande passante est vérifiée. Pour cette deuxième sous-requête les liens virtuels $l_v(e,d)$ et $l_v(d,f)$sont affectés aux chemins physiques $P(D-C-A)$ et $P(A-E-F-G)$.

Algorithme d'affectation de liens

Le paramètre d'entrée de cet algorithme est un sous-réseau $G_v(i)$ ainsi que la topologie physique G_s. L'objectif de cet algorithme est d'affecter les liens virtuels du sous-réseau $G_v(i)$ aux chemins physiques du réseau G_s. Chaque lien physique utilisé aura une nouvelle valeur résiduelle de bande passante disponible. Cette valeur est calculée selon l'équation (3.2). Le paramètre de sortie de cet algorithme est donc le réseau physique G_s avec moins de ressources disponibles en bande passante.

Dans notre approche, nous traitons les liens virtuels de la sous-requête un par un. Pour chacun de ces liens, nous appliquons l'algorithme du k plus court chemin afin de trouver le meilleur chemin pouvant satisfaire ses exigences (*Voir* Algorithme 3.3: ligne 5). Cet algorithme génère le chemin le plus court entre deux nœuds physiques. En effet, c'est le chemin le moins coûteux en termes de bande passante. Chaque lien physique appartenant à ce chemin doit avoir assez de bande passante disponible pour satisfaire les besoins du lien virtuel en question (*Voir* Algorithme 3.3: ligne 6). Si cette première condition est satisfaite, une deuxième sera vérifiée dépendamment du type de contrainte caractérisant les liens de cette sous-requête. En effet, pour une contrainte du type tolérance au délai, chaque lien du chemin physique sélectionné doit offrir une tolérance au délai inférieure ou égale à celle exigée par le lien virtuel (*Voir* Algorithme 3.3: ligne 12). De la même manière, si la contrainte est du type perte de paquets, alors chaque lien du chemin physique sélectionné doit assurer une tolérance à la perte de paquets inférieure à celle exigée par le lien virtuel (*Voir* Algorithme 3.3: ligne 22). Si l'une de ces conditions n'est pas satisfaite, le chemin est rejeté et un deuxième plus court chemin sera sélectionné. Nous rappelons que si le service demandé par la requête est du type service au mieux (« best effort »), seule la disponibilité de la bande passante sur les liens physiques est vérifiée.

Algorithme 3.3 Affectation des liens dans l'algorithme de base

$G_s = (N_s, L_s, C_s^n, C_v^l)$ *// Le réseau physique*

Sortie: $G_s = (N_s, L_s, C_s^n, C_v^l)$ *// Le réseau physique*

25. **while** $(L_v(i) \neq \emptyset)$ **do**
26. $k = 1$ *//Compteur pour l'algorithme du k-plus court chemin*
27. $l_v(a,b) \in L_v(i)$
28. **if** $((a \uparrow A)$ **and** $(b \uparrow B))$ avec $\{A, B\} \in N_s$
//Appliquer l'algorithme du k plus court chemin
29. $P_s(k) = k\_Shortest\_Path(G_s, A, B, k)$
30. **if** $(R(P_s(k)) < b(l_v(a,b)))$ avec $R(P_s(k)) = min_{l_s \in P_s(k)}(R_L(l_s))$
//Chemin rejeté
31. $k = k + 1$
32. **go to** 5
33. **else**
34. **switch** $(s(l_v(a,b)))$ **do**
35. **case** 0 *//Lien sensible au délai*
36. **if** $(d(P_s(k)) > d(l_v(a,b)))$ *//Chemin rejeté*
37. $k = k + 1$
38. **go to** 5
39. **else**
40. $l_v(a,b) \uparrow P_s(k)$ *//Affecter le lien*
41. $L_v(i) = L_v(i) - l_v(a,b)$ *//Supprimer le lien affecté*
//Mettre à jour les ressources des liens du chemin choisi
42. $\forall l_s \in P_s(k)$
43. $b(l_s) = b(l_s) - b(l_v(a,b))$
44. **end if**

45. **case** 1 *//Lien sensible à la perte de paquets*
46. **if** $(pp(P_s(k)) > pp(l_v(a,b)))$ *//Chemin rejeté*
47. $k = k + 1$
48. **go to** 5
49. **else**
50. $l_v(a,b) \rightarrow P_s(k)$ *//Affecter le lien*
51. $L_v(i) = L_v(i) - l_v(a,b)$ *//Supprimer le lien affecté*
//Mettre à jour les ressources des liens du chemin choisi
52. $\forall l_s \in P_s(k)$
53. $b(l_s) = b(l_s) - b(l_v(a,b))$
54. **end if**

55. **case** 0 *//Trafic best effort*
56. $l_v(a,b) \rightarrow P_s(k)$ *//Affecter le lien*

```
57.                    L_v(i) = L_v(i) − l_v(a, b) //Supprimer le lien affecté
58.                    //Mettre à jour les ressources des liens du chemin choisi
59.                    ∀l_s ∈ P_s(k)
60.                    b(l_s) = b(l_s) − b(l_v(a, b))
61.                end switch
62.            end if
63.        end if
64.    end while
```

3.2.3 Approche hybride avec indice de priorité

Dans l'approche hybride de base, nous avons proposé un modèle simple pour la réception et la mise en attente des requêtes. Ce dernier se base sur une seule file d'attente FIFO qui accueille toutes les requêtes. Afin de rendre ce modèle plus réaliste, nous avons opté pour le Weighted Fair Queuing (WFQ) comme politique d'ordonnancement des requêtes (You et Zhao, 2011). L'objectif de cette politique est de classer les requêtes selon leur indice de priorité. Cet indice est relié à un contrat de service qui permet au FS de payer pour accéder aux ressources les moins polluantes. Par conséquent, l'approche propose de leur donner la plus haute priorité, car plus cette dernière est élevée plus le FS a des chances de se voir allouer ce type de ressources. Pour ce faire, nous avons attribué un paramètre supplémentaire p à chacune des requêtes. Cet attribut et un nombre entier entre un et trois qui définit cet indice de priorité.

Cette approche est aussi divisée en quatre phases distinctes : 1) Réception et mise en file d'attente des requêtes; 2) Décomposition des requêtes; 3) Affectation des nœuds; 4) Affectation des liens. Il faut noter que les phases deux et quatre sont similaires à celles de l'approche hybride de base donc elles ne seront pas expliquées dans cette sous-section.

3.2.3.1 Politique d'ordonnancement WFQ

Le WFQ est une politique d'ordonnancement qui permet de partager la bande passante disponible entre des files d'attente FIFO ayant des poids différents. Chaque flux est associé à une file d'attente indépendante de manière à veiller à ce que le trafic le plus important obtienne une plus grande priorité par rapport au trafic le moins important (Dekeris, Adomkus et Budnikas, 2006).

3.2.3.2 Réception et mise en file d'attente des requêtes

Dans l'exemple illustré à la Figure 3.9, le modèle est composé de trois files d'attente de classe différentes : F_1 de classe 1, F_2 de classe 2 et F_3 de classe 3. Chaque requête a un indice de priorité de un à trois (Ex. : $R72^3$ est la requête numéro 72 d'indice de priorité trois). Les requêtes dont l'indice de priorité est égal à un sont envoyées à la file d'attente F_1de classe 1. Les requêtes dont l'indice de priorité est égal à deux sont envoyées à la file d'attente F_2de classe 2. Les requêtes dont l'indice de priorité est égal à trois sont envoyées à la file d'attente F_3de classe 3. La file d'attente de classe 1 à un poids de trois (w_1=3). En d'autres termes, à chaque itération trois requêtes de cette file seront traitées. Quant aux files d'attente 2 et 3, elles ont respectivement des poids de deux (w_2=2) et de un (w_3=1).

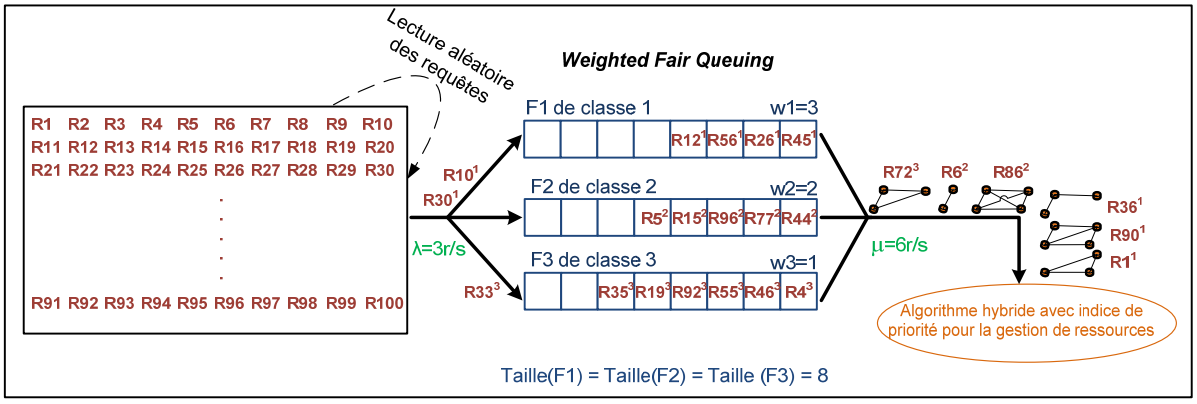

Figure 3.9 Réception et mise en file d'attente des requêtes dans l'algorithme de base

Dans notre approche, les requêtes munies d'un indice de priorité élevé sont non seulement traitées plus rapidement, mais aussi elles seront affectées dans les secteurs les plus verts (écologiques). En effet, les requêtes dont l'indice de priorité est égal à un sont affectées en priorité dans le secteur 1(secteur le plus vert). Si les ressources ne sont plus disponibles dans le secteur 1, alors ces requêtes sont affectées au secteur 2. Les requêtes dont l'indice de priorité est égal à deux sont affectées en priorité dans le secteur 2. Si les ressources ne sont pas disponibles dans ce secteur, alors ces requêtes sont affectées au secteur 3. Les requêtes dont l'indice de priorité est égal à trois sont affectées en priorité dans le secteur 3 sinon au secteur 4.

Algorithme de la mise en file d'attente WFQ

Cet algorithme prend comme entrées l'ensemble T_v des requêtes à classifier, le taux d'arrivée des requêtes λ, les trois files d'attente Q_1, Q_2et Q_3 ainsi que leur taille maximale t. Il génère comme résultat trois nouvelles files d'attente. En effet, après l'exécution de cet algorithme λ requêtes seront ajoutées aux trois files d'attente de départ. La première étape de cet algorithme consiste à sélectionner d'une manière aléatoire une requête $G_v(i)$ de l'ensemble T_v. Pour ce faire, nous utilisons la fonction prédéfinie Select() (*Voir* Algorithme 3.4: ligne 3). La deuxième étape consiste à affecter la requête sélectionnée à l'une des trois files d'attente du modèle. Comme nous l'avons expliqué auparavant, le choix de la file d'attente dépend de l'indice de priorité des requêtes $p(G_v)$. En effet, si l'indice de priorité d'une requête est égal à un alors l'algorithme l'ajoute à la fin de la file 1 (*Voir* Algorithme 3.4: ligne 7). De même, si cet indice est égal à deux alors la requête est ajoutée à la fin de la file 2 et s'il est égal à 3, elle est ajoutée à la fin de la troisième file (*Voir* Algorithme 3.4: lignes 13 et 19). Avant chaque ajout, l'algorithme vérifie, à l'aide de la fonction prédéfinie Size(), la taille de la file d'attente. Si elle a atteint sa taille limite t, la requête est rejetée (*Voir* Algorithme 3.4: lignes 9, 15 et 21).

Algorithme 3.4 Mise en file d'attente des requêtes avec le WFQ

Entrées : Q_1, Q_2, Q_3 *//Les 3 files d'attentes*
λ *//Taux d'arrivée des requêtes*
t *//Taille maximale des files d'attente*
$T_v = \sum_{i=1}^{m} G_v(i)$ *//L'ensemble des m requêtes à classifier*
Sorties: Q_1, Q_2, Q_3

```
1.  j = 1
2.  for j ≤ λ do
3.      Select (Gv (i)) // Sélectionner une requête aléatoirement
4.      switch (p(Gv (i))) do
5.      case 1 //Requête de priorité 1
6.          if (size(Q1) < t)
7.              Q1= Q1 + Gv (i) //Insérer la requête dans la file Q1
8.          else
            //File pleine: Interrompre l'exécution de l'itération et rejeter la requête
9.              break
10.         end if
11.     case 2 //Requête de priorité 2
12.         if (size(Q2) < t)
13.             Q2= Q2 + Gv (i) //Insérer la requête dans la file Q2
14.         else
            //File pleine: Interrompre l'exécution de l'itération et rejeter la requête
15.             break
16.         end if
17.     case 2 //Requête de priorité 2
18.         if (size(Q3) < t)
19.             Q3= Q3 + Gv (i) //Insérer la requête dans la file Q3
20.         else
            //File pleine: Interrompre l'exécution de l'itération et rejeter la requête
21.             break
22.         end if
23. end switch
24.     Tv = Tv − Gv (i) // Supprimer la requête sélectionnée
25.     j=j+1
26. end for
```

3.2.3.3 Affectation des nœuds

Dès la réception d'une sous-requête, le NG vérifie son indice de priorité. Si cet indice est égal à un, le NG tente dans un premier temps d'héberger les nœuds de cette sous-requête dans le secteur 1 dont les ressources sont les moins polluantes. S'il n'y pas assez de ressources disponibles en CPU dans ce secteur, le NG consulte alors les autres secteurs. Par contre, si l'indice de priorité de la sous-requête est égal à deux, c'est le secteur 2 qui sera balayé en premier par le NG pour trouver les ressources nécessaires pour les nœuds de cette sous-requête. En cas où les ressources disponibles dans ce secteur sont insuffisantes, le NG continue ses recherches dans le secteur 3 puis en cas de besoin dans le secteur 4. Enfin, lorsque l'indice de priorité d'une sous-requête est égal à trois, le NG limite ses recherches au secteur 4 dont le facteur d'émission de CO_2 est le plus élevé.

Algorithme d'affectation des nœuds

Le paramètre d'entrée de cet algorithme est un sous-réseau $G_v(i)$ ainsi que la topologie physique G_s. Son paramètre de sortie est le réseau physique G_s avec les ressources résiduelles disponibles en CPU. Cet algorithme diffère de celui de l'approche de base (*Voir* Algorithme 3.5) dans la phase de sélection des secteurs. En effet, comme nous l'avons expliqué dans le paragraphe précédent, cette sélection dépend de l'indice de priorité de la sous-requête $p(G_v(i))$. Une fois, le secteur où les recherches vont débuter est déterminé, la suite de l'algorithme est semblable à celle décrite dans l'approche de base (*Voir* Algorithme 3.5).

Algorithme 3.5 Affectation des nœuds avec indice de priorité

Entrées : $G_v(i) = (N_v(i), L_v(i), C_v^n(i), C_v^l(i))$ *// Un sous-réseau virtuel*
$G_s = (N_s, L_s, C_s^n, C_v^l)$ *// Le réseau physique*
Sortie: $G_s = (N_s, L_s, C_s^n, C_v^l)$ *// Le réseau physique*

1. **Upon** receiving $G_v(i)$ **do**
2. **while** $(N_v(i) \neq \emptyset)$ **do** *// Tant que il y a des nœuds virtuels non assignés*
3. **for** $\forall n_v \in N_v(i)$ **do**
4. **if (** $p(G_v(i)) = 1$) *// Indice de priorité = 1*
5. $j = 1$ *// Indice pour le secteur 1*
6. **else if** ($p(G_v(i)) = 2$) *// Indice de priorité = 2*
7. $j = 2$ *// Indice pour le secteur 2*
8. **else** *// Indice de priorité = 3*
9. $j = 4$ *// Indice pour le secteur 4*
10. **end if**
11. **Sort** $(N_s(j))$ *// Trier les nœuds du secteur j selon la valeur de CPU*
// S'il y a assez de ressources dans le secteur j
12. **if** $(c(\mathbf{Head}(N_s(j))) \geq c(n_v))$
// Trouver l'indice du nœud physique avec la plus proche valeur de CPU
13. $\forall n_s \in N_s(j)$
14. $k = 0$
15. **while** $(c(n_s)) \geq c(n_v))$
16. $k = k + 1$
17. **end while**
18. $n_v \uparrow n_s$ *// n_v est hébergé dans le nœud n_s d'indice k*
// Mettre à jour la valeur du CPU de n_s
19. $c(n_s) = c(n_s) - c(n_v)$
// Supprimer le nœud n_v de la liste des nœuds à affecter
20. $N_v(i) = N_v(i) - n_v$
21. **else**
22. $j = j + 1$ *// Passer au prochain secteur*
23. **if** $(j = 5)$ *// Les 4 secteurs ont été balayés*
24. **exit** *// Pas de ressources disponibles pour n_v*
25. **else**
26. **go to** 11 *// Aller à la 11ème ligne*
27. **end if**
28. **end if**
29. **end for**
end while

Ainsi, cette approche permet non seulement de gérer les ressources d'une manière efficace, mais aussi de prendre en considération le type d'énergie utilisée dans le but de minimiser l'émission de gaz à effet de serre. Par ailleurs, certains fournisseurs de service peuvent avoir des contraintes de localisation. Par conséquent, nous proposons d'étendre cette approche pour intégrer le critère de localisation tel que décrit dans la sous-section suivante.

3.2.4 Approche hybride avec indice de priorité et indice de localisation

Certains fournisseurs de services exigent de leurs FIV que leurs réseaux virtuels soient déployés à travers des infrastructures physiques situées dans des zones géographiques bien déterminées. Par exemple, si une société financière américaine prévoit déployer un réseau virtuel fourni par un FIV pour transférer des données qui doivent être stockées dans un centre de données, elle lui exigera certainement pour des raisons de sécurité que ses données soient transférées à travers des réseaux physiques localisés aux États-Unis. Par conséquent, nous avons décidé d'apporter des modifications à la solution précédente afin d'introduire cette contrainte de localisation. Pour ce faire, nous avons divisé le réseau physique géré par un FIV en trois régions. Chaque région est divisée en quatre secteurs (*Voir* Figure 3.10). Chaque secteur est associé à un facteur d'émission $\mathrm{Emf}(\mathrm{Loc}(n_s))$. Nous avons attribué à chaque requête un indice de localisation $r(G_v)$ en plus de l'indice de priorité$p(G_v)$. Dans cette architecture, nous définissons un NC et trois NGs par région. Ces nœuds sont déterminés de la manière que dans les deux précédentes approches : Le NC le nœud physique qui a le plus de ressources disponibles en termes de CPU et de bande passante dans les trois régions. Les nœuds NG_1, NG_2 et NG_3 sont les nœuds physiques dont les ressources physiques (CPU et bande passante) sont les plus élevées dans chaque région (*Voir* équation3.12).

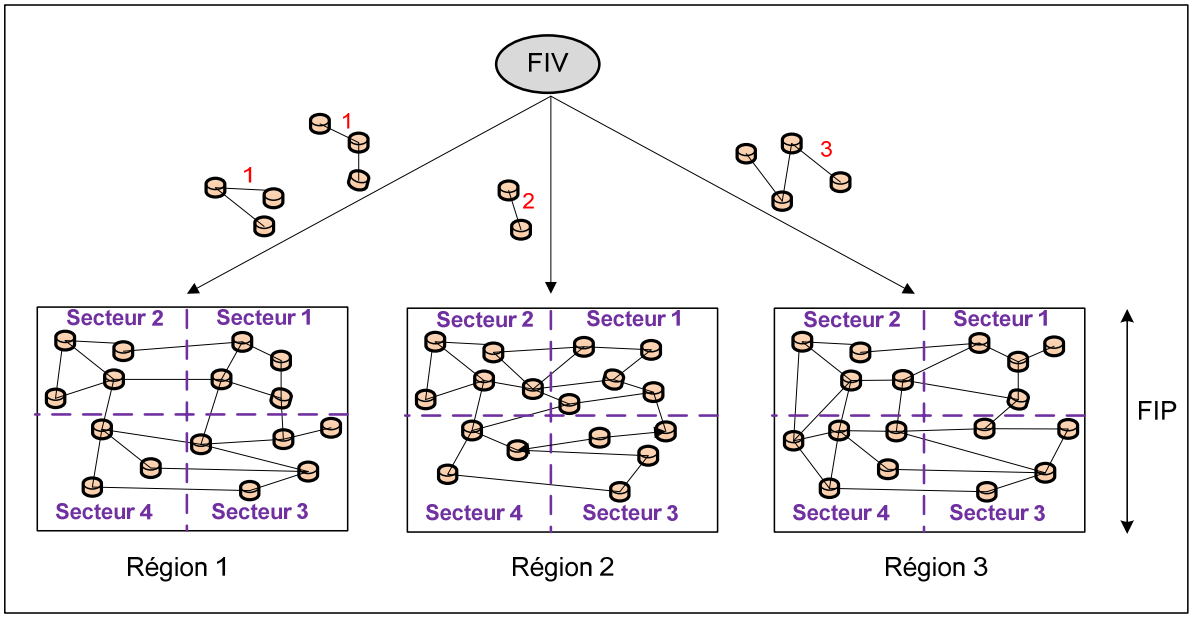

Figure 3.10 Approche hybride avec indice de priorité et indice de localisation

La phase de réception et de mise en file d'attente et la même que celle de la solution précédente (*Voir* Figure 3.9). Par contre, la phase d'affectation des nœuds est différente. En effet, chaque réseau virtuel est affecté à l'une des trois régions dépendamment de son indice de localisation. Si cet indice est égal à un, le FIV limite ces recherches de ressources disponibles dans la région 1. Si cet indice est égal à deux, le FIV limite ces recherches de ressources disponibles dans la région 2. Si cet indice est égal à trois, le FIV limite ces recherches de ressources disponibles dans la région 3. Dans le cas où les nœuds de cette région n'offrent pas assez de ressources en CPU aux nœuds d'une requête donnée, cette dernière est alors rejetée. L'indice de priorité est ensuite utilisé pour le choix des secteurs comme nous l'avons expliqué auparavant.

3.2.4.1 Algorithme d'affectation des nœuds

Pour chaque sous-requête $G_v(i)$, nous avons affecté son indice de localisation $r(G_v(i))$ à la variable reg (*Voir* Algorithme 3.6: ligne 2). Cette variable est ensuite utilisée comme deuxième attribut, en plus de l'indice de priorité, pour les nœuds physiques (Ex. : $N_s(3,4)$ est

l'ensemble des nœuds du secteur 4 de la région 3). Cet indice de localisation permet au NG de limiter ses recherches dans une des trois régions de l'infrastructure physique (Ex. : $r(G_v(i))$ =1 : Seulement la région 1 sera considérée). Ensuite, l'indice de priorité de la sous-requête sera déterminé (*Voir* Algorithme 3.6: lignes 5 à 11). Le NG commence alors à chercher les ressources en CPU dans le secteur approprié selon la méthode expliquée dans l'approche hybride avec indice de priorité (*Voir* section 3.2.3.1).

Algorithme 3.6 Affectation des nœuds avec indice de priorité et indice de localisation

Entrées : $G_v(i) = (N_v(i), L_v(i), C_v^n(i), C_v^l(i))$ *// Un sous-réseau virtuel*
$G_s = (N_s, L_s, C_s^n, C_v^l)$ *// Le réseau physique*
Sortie: $G_s = (N_s, L_s, C_s^n, C_v^l)$ *// Le réseau physique*

1. **Upon** receiving $G_v(i)$ **do**
2. reg = $r(G_v(i))$ *// Indice de localisation de la sous-requête*
3. **while** $(N_v(i) \neq \emptyset)$ **do** *// Tant que il y a des nœuds virtuels non assignés*
4. **for** $\forall n_v \in N_v(i)$ **do**
5. **if (** $p(G_v(i)) = 1$) *// Indice de priorité = 1*
6. $j = 1$ *// Indice pour le secteur 1*
7. **else if** $(p(G_v(i)) = 2)$ *// Indice de priorité = 2*
8. $j = 2$ *// Indice pour le secteur 2*
9. **else** *// Indice de priorité = 3*
10. $j = 4$ *// Indice pour le secteur 4*
11. **end if**
// Trier les nœuds du secteur j de la région reg selon la valeur de CPU
12. **Sort** $(N_s(reg,j))$
// S'il y a assez de ressources dans le secteur j de la région reg
13. **if** $(c(\mathbf{Head}(N_s(reg, j))) \geq c(n_v))$
// Trouver l'indice du nœud physique avec la plus proche valeur de CPU
14. $\forall n_s \in N_s(reg, j)$
15. $k = 0$
16. **while** $(c(n_s)) \geq c(n_v))$
17. $k = k + 1$
18. **end while**
19. $n_v \uparrow n_s$ *// n_v est hébergé dans le nœud n_s d'indice k*
// Mettre à jour la valeur du CPU de n_s
20. $c(n_s) = c(n_s) - c(n_v)$
// Supprimer le nœud n_v de la liste des nœuds à affecter
21. $N_v(i) = N_v(i) - n_v$

```
22.                    else
23.                        j = j + 1  // Passer au prochain secteur
24.                        if (j = 5)  // Les 4 secteurs ont été balayés
25.                            exit  // Pas de ressources disponibles pour  n_v
26.                        else
27.                            go to  12  // Aller à la 12ème ligne
28.                        end if
29.                    end if

30.                end for
31.    end while
```

Afin de pouvoir évaluer les performances de ces trois approches proposées, nous les avons comparées avec deux autres algorithmes existants dans la littérature qui utilisent des stratégies d'affectation différentes. Ces derniers sont le *Baseline VN Embedding Algorithm* proposé par (Minlan Yu, 2008) et le *Distributed Virtual Network Mapping Algorithm* proposé par (Houidi, Louati et Zeghlache, 2008b). Dans le prochain chapitre, nous expliquerons en détail ces deux algorithmes et nous analyserons les résultats des simulations réalisées.

CHAPITRE 4

SIMULATIONS ET ANALYSE DES RÉSULTATS

4.1 Autres algorithmes

Afin d'évaluer les performances des trois approches présentées dans ce mémoire, nous les avons comparées aux performances de deux autres algorithmes qui ont été proposés dans des travaux de recherches antérieures. Le premier algorithme implémenté est le *Baseline VN Embedding Algorithm* proposé par (Minlan Yu, 2008). C'est un algorithme basé sur une approche centralisée. En effet, une seule entité est responsable à la réception des requêtes et à l'affectation de ses nœuds et liens virtuels au réseau physique. Le deuxième algorithme, le *Distributed Virtual Network Mapping Algorithm*, est proposé par (Houidi, Louati et Zeghlache, 2008b) et adopte une approche distribuée. En d'autres termes, plusieurs nœuds de l'infrastructure physique participent au processus de création du réseau virtuel.

Nous présentons, avec plus de détails, le fonctionnement de chacun de ces deux algorithmes afin de faciliter l'analyse des résultats des simulations.

4.1.1 Baseline VN Embedding Algorithm

L'approche proposée par (Minlan Yu, 2008) divise le processus d'affectation des requêtes au réseau physique en deux étapes : Affectation des nœuds des requêtes et affectation des liens des requêtes.

Affectation des nœuds

Les requêtes reçues sont d'abord triées dans un ordre décroissant selon la valeur de leur revenu (*Voir* Algorithme 4.1). Ce paramètre est équivalent au coût virtuel dans notre approche, calculé selon l'équation (3.9). Ensuite, pour chaque requête, l'algorithme extrait l'ensemble S des nœuds physiques dont le CPU disponible satisfait la plus grande valeur de

CPU demandé par cette requête. Enfin, chaque nœud virtuel est affecté au nœud physique qui a le plus de ressources disponibles et qui appartient à l'ensemble S (*Voir* équation 3.12).

Algorithme 4.2 Affectation des nœuds dans le Baseline VN Embedding Algorithm
Tiré de Minlan Yu (2008)

1. Trier les requêtes selon leur revenu.
2. S'il ne reste plus de requêtes, arrêter.
3. Extraire la requête avec la plus grande valeur de revenu.
4. Trouver l'ensemble S des nœuds physiques qui satisfont les exigences en CPU de la requête. Si $S = \emptyset$, garder la requête dans la file d'attente et aller à l'étape 2.
5. Pour chaque nœud virtuel, trouver le nœud physique dans S avec la valeur H la plus grande.
6. Aller à l'étape 2.

Affectation des liens

Une fois tous les nœuds des requêtes sont traités, l'algorithme passe à la deuxième étape qui est l'affectation des liens (*Voir* Algorithme 4.3). Dans cette étape, seules les requêtes dont tous les nœuds ont été affectés, seront considérées. Pour chaque lien virtuel, l'algorithme du k plus court chemin est appliqué afin de trouver le meilleur chemin physique qui peut l'héberger en utilisant le paramètre de la bande passante comme métrique.

Algorithme 4.4 Affectation des liens dans le Baseline VN Embedding Algorithm
Tiré de Minlan Yu (2008)

1. Trier les requêtes dont tous les nœuds ont été affectés avec succès selon leur revenue.
2. S'il ne reste plus de requêtes, arrêter.
3. Extraire la requête avec la plus grande valeur de revenu.
4. Pour chaque lien virtuel, appliquer le k plus cout chemin pour des valeurs de k croissantes jusqu' à trouver le chemin physique qui satisfait les contraintes en bande passante de ce lien.
5. Si aucun chemin physique n'est trouvé pour un lien donné, garder la requête dans la

file d'attente et aller à l'étape 2.
6. Aller à l'étape 2.

4.1.2 Distributed Virtual Network Mapping Algorithm

Dans cette approche proposée par (Houidi, Louati et Zeghlache, 2008b), les requêtes reçues sont d'abord divisées en plusieurs sous-requêtes. Ensuite, chaque sous-requête est affectée séparément au réseau physique.

Division des requêtes

Chaque requête reçue est divisée en plusieurs grappes. Chaque grappe est composée d'un nœud central (Hub) interconnecté en étoile à d'autres nœuds virtuels. La sélection de ces grappes se fait ainsi : Le nœud virtuel qui requiert la capacité la plus élevée est considéré comme Hub et tous les nœuds qui lui sont directement connectés sont les nœuds secondaires (Spoke).

Affectation des sous-requêtes

Pour chaque sous-requête, l'algorithme sélectionne le nœud physique dont la capacité disponible est la plus élevée. Ce dernier, nommé Root, est affecté au Hub de la sous-requête. Ensuite, l'algorithme du plus court chemin est appliqué afin de déterminer les chemins les plus courts entre le nœud Root et tous les autres nœuds physiques qui lui sont connectés. La métrique utilisée est le poids w qui correspond à un ensemble de paramètres d'un lien physique tels que le délai et le coût. Tous les chemins dont les nœuds ne disposent pas d'assez de ressources pour satisfaire les exigences des nœuds de la sous-requête sont éliminés. Aussi, tous les chemins dont les liens ne disposent pas d'assez de bande passante pour satisfaire les exigences des liens de la sous-requête sont éliminés. Parmi les chemins

restants, l'algorithme choisit les chemins qui ont le poids le moins élevé et affecte le Spoke qui requiert le plus de capacité au nœud physique qui en offre le plus (*Voir* Algorithme 4.5).

Algorithme 4.6 Affectation des requêtes dans le Distributed VN Mapping Algorithm
Tiré de Houidi, Louati et Zeghlache (2008b)

1. S'il ne reste plus de sous-requêtes, arrêter.
2. Trouver le Root (nœud physique qui dispose de la capacité) et l'assigner au Hub. Trouver les plus courts chemins entre le Root et les nœuds qui lui sont connectés en appliquant l'algorithme du plus court chemin.
4. Trouver l'ensemble S des chemins physiques qui satisfont les exigences en capacité de la sous-requête. Si le nombre de chemins trouvés est inférieur au nombre de liens de la sous-requête, aller à l'étape 1.
5. À partir de S, trouver l'ensemble S' des chemins physiques qui satisfont les exigences en bande passante de la sous-requête. Si le nombre de chemins trouvés est inférieur au nombre de liens de la sous-requête, aller à l'étape 1.
6. Garder les chemins les moins gourmands en termes de poids.
7. Affecter le Spoke qui requiert le plus de capacité au nœud physique qui en offre le plus.
8. Aller à l'étape 1.

4.2 Mesures de performances

Dans cette section, nous allons présenter en premier lieu les outils de simulation utilisés dans ce travail, à savoir GT-ITM comme générateur de topologies et Matlab comme environnement de programmation. Ensuite, nous allons décrire les paramètres utilisés dans cette simulation. Enfin, nous allons analyser les résultats obtenus.

4.2.1 Outils de simulation

Générateur de topologies : GT-ITM

Tout comme le soulignent (Dong et Zou, 2008), le réseau internet actuel adopte une structure hiérarchique. Il est divisé en plusieurs systèmes autonomes (AS). Chaque AS est composé de plusieurs grappes appelées *area*. La grappe centrale de chaque AS est appelée *backbone*. Chaque nœud du *backbone* est lié à d'autres grappes.

Dans la topologie illustrée par la Figure 4.1, le premier système autonome AS_1 est composé de cinq grappes A_1, A_2, A_3 etA_4. La grappe A_0 représente le *backbone*.

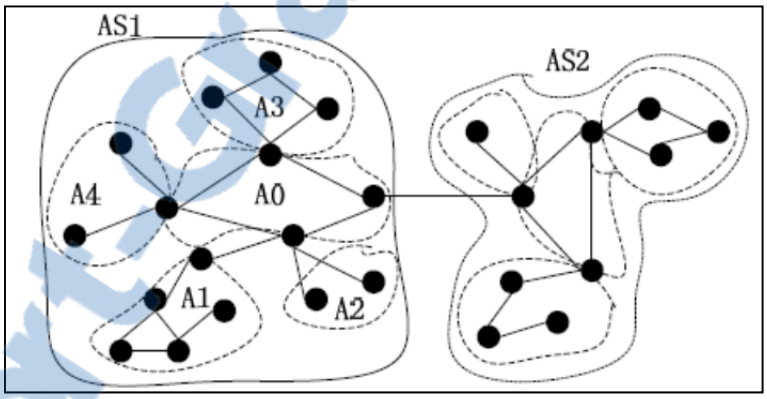

Figure 4.1Topologie d'Internet
Tirée de Dong et Zou (2008)

Il existe plusieurs générateurs de topologies de réseaux tels que BRITE (Medina et al., 2001), GT-ITM (Zegura, Calvert et Bhattacharjee, 1996), Inet (Suzuki et Yamamoto, 2000) et NTG (Melakessou et Engel, 2009). Nous avons opté pour le générateur GT-ITM car non seulement il permet de générer aléatoirement des réseaux de grande taille, mais aussi il offre la possibilité de reproduire le modèle hiérarchique du réseau internet actuel à l'aide de son modèle Transit-Stub (*Voir* Figure 4.2).

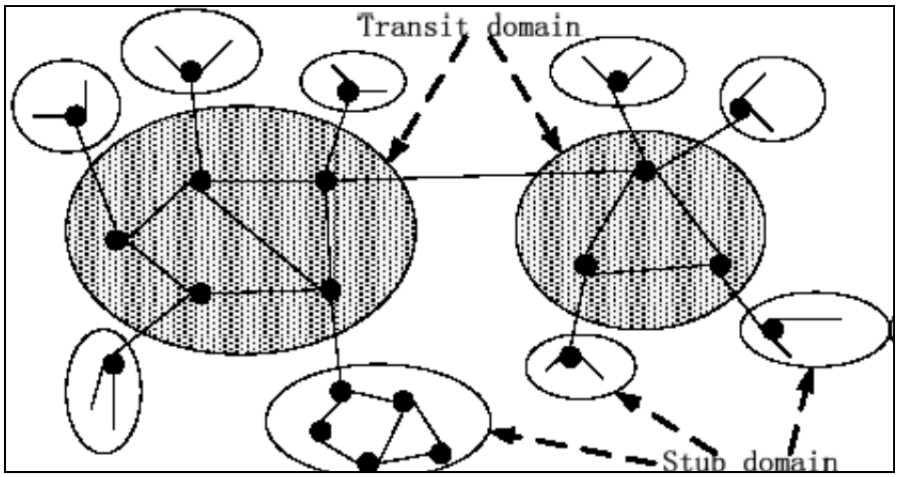

Figure 4.2 Le modèle de topologie Transit-Stub
Tirée de Dong et Zou (2008)

La Figure 4.2 illustre un exemple de topologie générée par GT-ITM. Nous remarquons que ce modèle reproduit parfaitement l'architecture hiérarchique du réseau internet. En effet, les ellipses sombres appelées *transit domain* représentent les réseaux cœur (*backbone)* et les ellipses blanches appelées *stub domain* représentent les zones (*area)* dans un AS.

Environnement de programmation : Matlab

Pour l'implémentation des algorithmes, nous avons choisi Matlab pour plusieurs raisons. Tous d'abord, c'est un logiciel très puissant, complet et facile à utiliser. En plus, c'est un environnement performant pour le calcul et la visualisation (Umarikar, Mishra et Umanand, 2006). Enfin, Matlab inclut tout un module comportant des fonctions prédéfinies traitant la théorie des graphes (Ex. : Graph Theory, Graph Vizualisation Methods).

4.2.2 Paramètre de simulation

Dans nos tests, la topologie de l'infrastructure physique est un réseau transit-stub de 100 nœuds généré à partir de l'outil GT-ITM. Le CPU d'un nœud physique est représenté par un nombre aléatoire compris entre 100 et 200 GHz. La tolérance de délai d'un lien physique est

un nombre aléatoire entre 100 et 350 ms. La tolérance de perte de paquets sur un lien est un nombre aléatoire entre 0 % et 5 %. Pour chaque lien, GT-ITM génère aléatoirement une valeur comprise entre 100 et 200 Mbps en guise de bande passante. Pour représenter les requêtes, GT-ITM génère 50 topologies aléatoires avec un CPU maximal de 20 GHz et une bande passante maximale de 20 Mbps. Ces requêtes arrivent avec un taux λ=2 requêtes/s.

Le réseau physique a été divisé en 4 secteurs. Chaque secteur a un facteur d'émission différent (Corp., 2010). Dans nos simulations, nous avons utilisé les valeurs décrites dans le Tableau 4.1.

Tableau 4.1 Facteurs d'émission pour chaque secteur

Secteur	Facteur d'émission
1	6.10^{-6}
2	0.25
3	0.4
4	0.8

4.2.3 Évaluation des performances

Afin de comparer les performances de notre approche avec celles de (Minlan Yu, 2008) et de (Houidi, Louati et Zeghlache, 2008b), nous avons analysé les sept paramètres suivants :

1) le nombre de nœuds utilisés : l'évolution du nombre des nœuds physiques utilisés par un algorithme pour héberger les nœuds virtuels des requêtes ;

2) le nombre de liens utilisés : l'évolution du nombre des liens physiques utilisés par un algorithme pour héberger les liens virtuels des requêtes ;

3) l'affectation des requêtes en fonction du temps : la variation du temps d'exécution des requêtes. Un algorithme est plus performant quand ce temps d'exécution est minimal ;

4) le revenu des requêtes en fonction du temps : la variation du revenu des requêtes (Voir équation 4.10) en fonction du temps. Plus ce revenu est élevé, plus les économies en argent réalisées par le client en choisissant le concept de virtualisation sont importantes ;

5) le coût des requêtes en fonction du temps : la variation du coût physique des requêtes (Voir équation 4.8) en fonction du temps. Ce coût représente les ressources physiques utilisées par un FIV pour héberger une requête. Le but d'un algorithme d'allocation de ressources est de choisir les ressources physiques telles que ce coût soit minimal ;

6) le coût en émission de CO_2 en fonction du temps : la variation du coût en CO_2 des requêtes (Voir équation 4.11) en fonction du temps. Ce coût est utile pour comparer l'impact écologique des trois approches.

4.2.4 Évaluation de l'approche hybride de base

Les Figures 4.3, 4.4 et 4.5 illustrent respectivement les 50 requêtes générées et affectées au réseau physique par l'approche centralisée, distribuée et hybride. Les nœuds colorés en cyan sont les nœuds physiques utilisés lors de l'affectation des différents nœuds virtuels. Les liens colorés en rose sont les liens physiques utilisés lors de l'affectation des différents liens virtuels. En plus, pour l'approche hybride le nœud coloré en rouge est le NC et les nœuds colorés en vert sont les NG (*Voir* Figure 4.5). Nous remarquons que la majorité des nœuds physiques utilisés par notre algorithme sont situés dans le secteur 1 qui est le secteur le plus écologique.

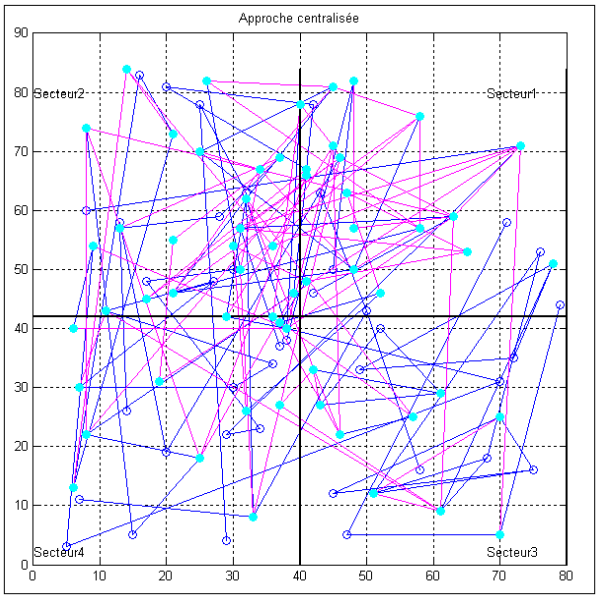

Figure 4.3 Topologie physique après exécution de l'approche centralisée

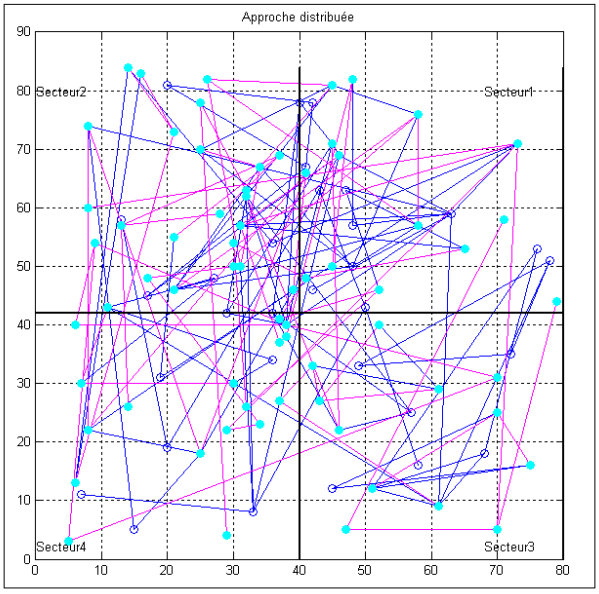

Figure 4.4 Topologie physique après exécution de l'approche distribuée

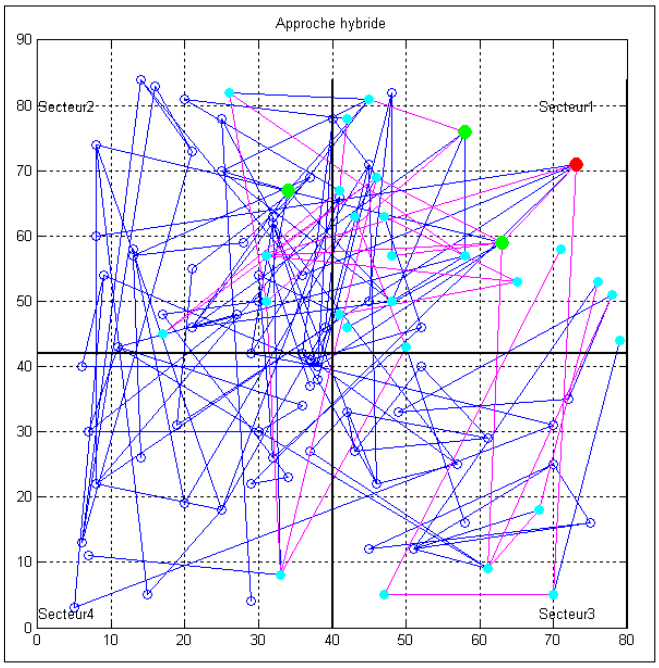

Figure 4.5 Topologie physique après exécution de l'approche hybride

En analysant les courbes obtenues, nous avons relevé les observations suivantes :

L'approche hybride de base a utilisé moins de nœuds et liens physiques que les deux autres approches :

Les Figures 4.6 et 4.7 illustrent respectivement le nombre de nœuds et de liens physiques utilisés pour traiter les 50 requêtes reçues pour les trois approches. Avec 28 nœuds et 30 liens physiques utilisés, l'algorithme hybride de base est celui qui utilise le moins de nœuds et liens comparativement aux approches centralisée et distribuée. Ce résultat est principalement dû à notre méthode d'affectation de nœuds qui vise à privilégier le choix des nœuds physique du secteur 1. Ceci augmente la probabilité d'affecter les requêtes aux mêmes nœuds et liens physiques.

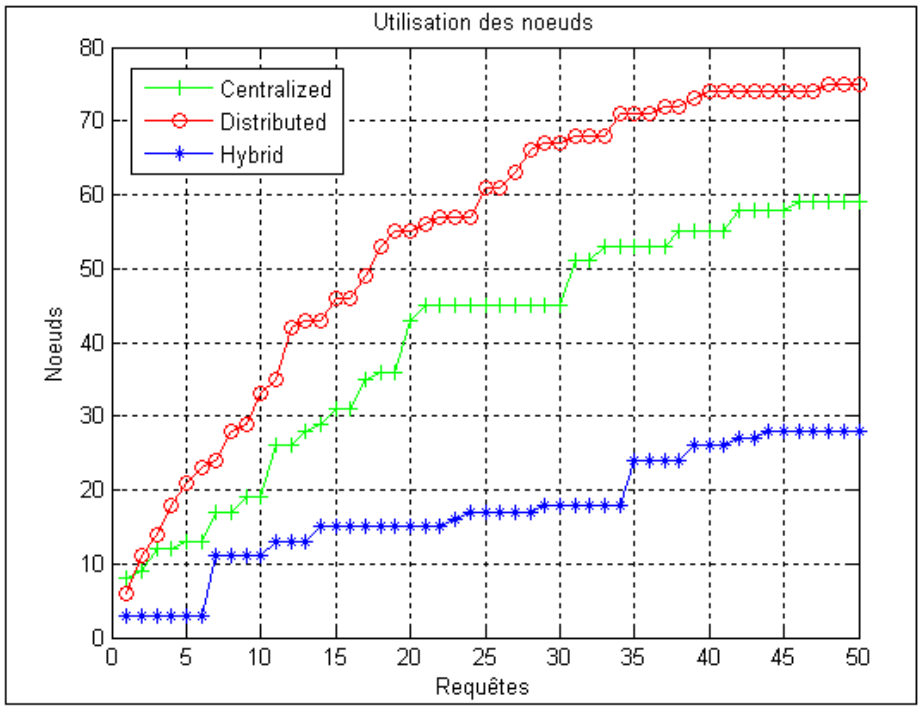

Figure 4.6 Nombre de nœuds physiques utilisés

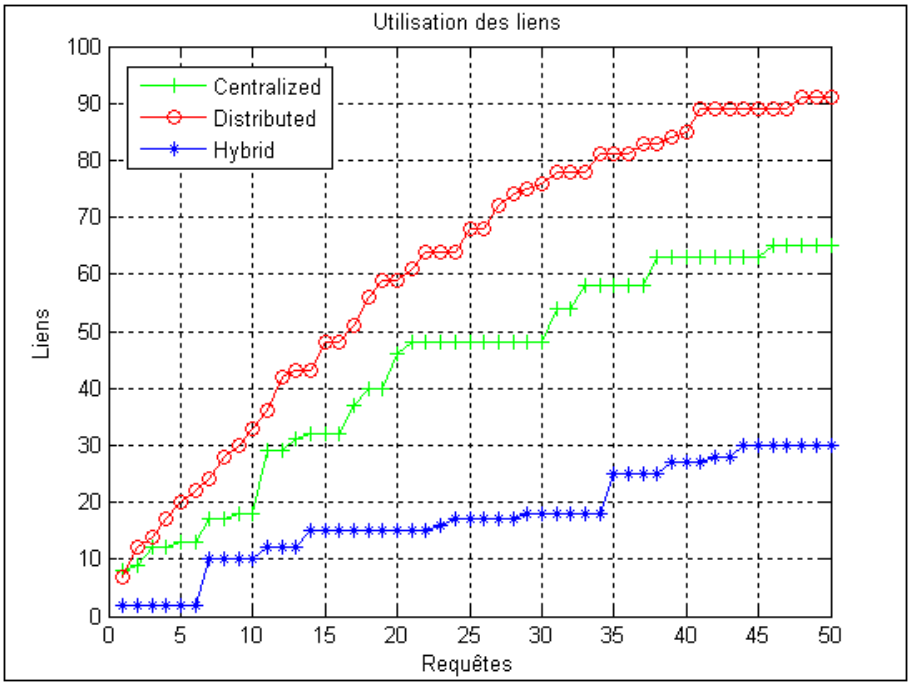

Figure 4.7 Nombre de liens physiques utilisés

Tableau 4.2 Nombre de nœuds et de liens utilisés

	Nœud utilisés	**Liens utilisés**
Approche hybride de base	28	30
Approche centralisée	60	65
Approche distribuée	75	91

L'approche distribuée a utilisé plus de nœuds et liens physiques que les deux autres approches :

Avec 75 nœuds et 91 liens physiques utilisés, l'approche distribuée est celle qui utilise le plus de nœuds et liens comparativement aux approches centralisée et hybride. En effet, lors de la phase d'affectation des sous-requêtes de l'approche distribuée, les nœuds virtuels d'une même sous-requête sont affectés obligatoirement à des nœuds physiques différents (Voir Algorithme 3.3). Par conséquent, le nombre de nœuds et liens physiques utilisés lors de cette phase sera plus important.

L'approche hybride de base est la plus rapide :

La Figure 4.8 illustre l'évolution du temps d'exécution des requêtes. Nous remarquons que notre algorithme est le plus rapide avec un temps d'exécution total de 59 secondes contre 142 secondes pour l'algorithme centralisé et 119 secondes pour le distribuer. L'utilisation de nœuds du même secteur facilite la tâche au NG pour trouver des chemins physiques qui satisfont les contraintes des liens des requêtes. En effet, plus les nœuds sont géographiquement proches, plus les chemins entre ces nœuds sont courts. En d'autres termes, le NG aura moins de liens physiques à vérifier.

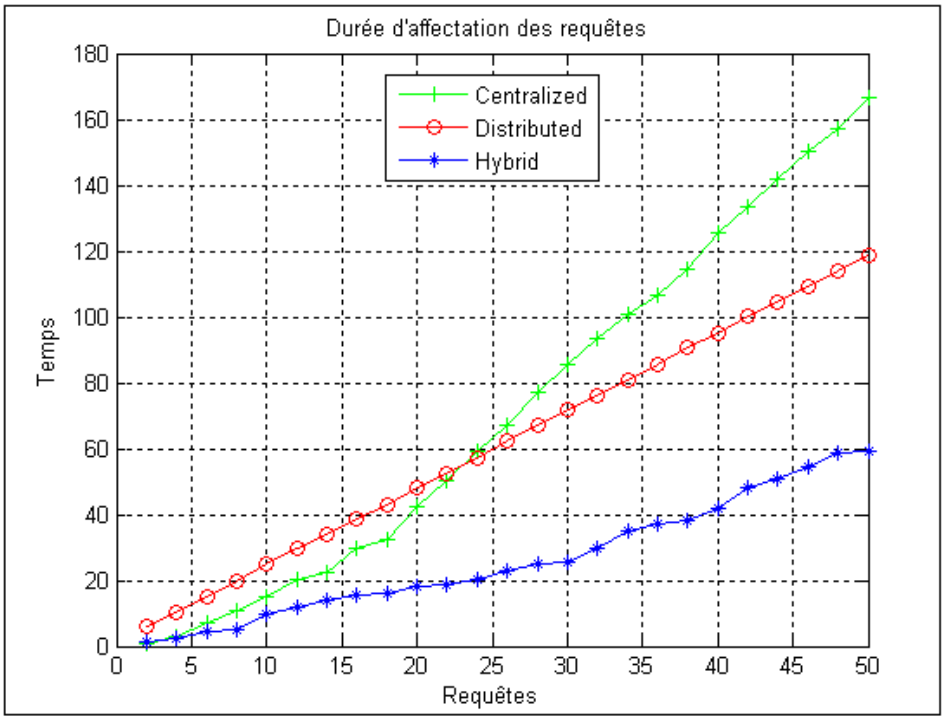

Figure 4.8 Affectation des requêtes en fonction du temps

Les ressources utilisées par l'approche hybride de base sont les moins polluantes : La Figure 4.9 illustre la variation du coût en émission de CO_2 des requêtes. Nous remarquons que ce coût relevé de notre algorithme est presque nul. En d'autres termes, la consommation d'énergie des différents nœuds physiques utilisés par notre algorithme est nettement inférieure à celle utilisée par les algorithmes existants. Ceci permet au FIV de préserver l'environnement en réduisant les gaz à effet de serre et d'avoir d'éventuelles primes ou subventions gouvernementales.

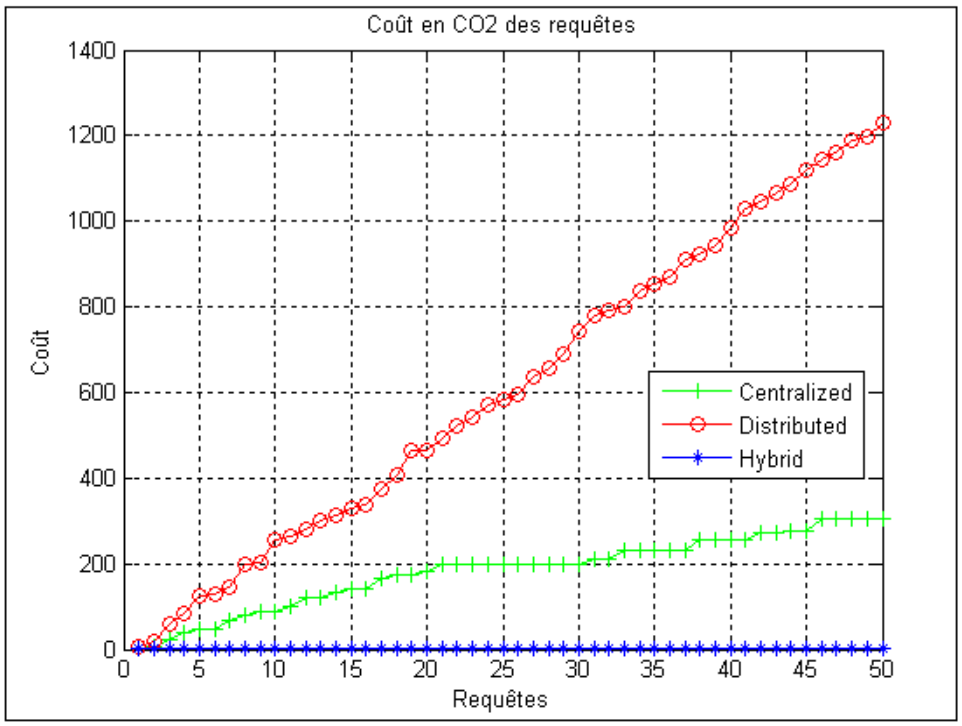

Figure 4.9 Coût en émission de CO_2 des requêtes

Revenu et coût :

Les Figures 4.10 et 4.11 illustrent respectivement la variation du coût physique et du revenu des requêtes. Notre algorithme génère le coût le moins élevé. En d'autres termes, il utilise moins de ressources physiques que les algorithmes existants. En effet, comme nous l'avons expliqué auparavant, les chemins choisis par notre algorithme sont des chemins courts. Donc la consommation de la bande passante sera moins importante. Aussi, nous remarquons que le revenu généré par notre approche est le moins élevé puisque ce dernier est proportionnel au coût physique (Voir équation 3.10).

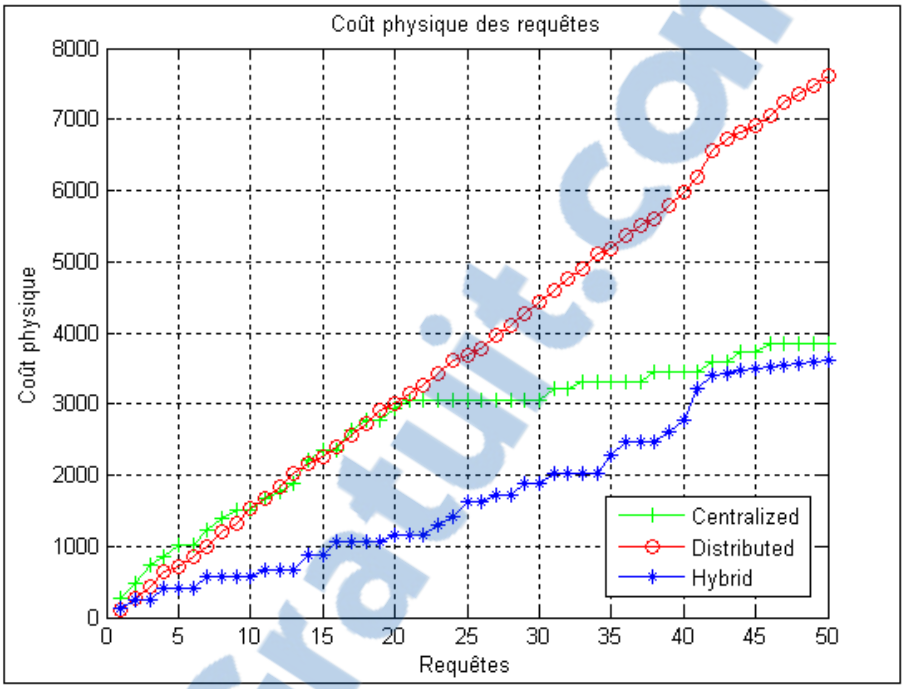

Figure 4.10 Coût des requêtes en fonction du temps

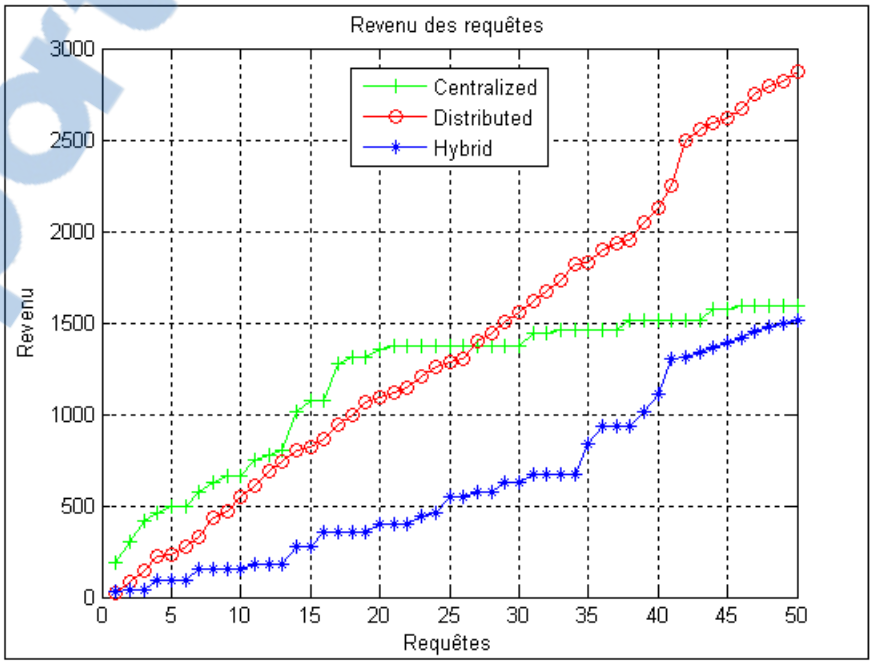

Figure 4.11 Revenu des requêtes en fonction du temps

4.2.5 Évaluation de l'approche hybride avec indice de priorité

Nous rappelons que dans l'approche hybride avec indice de priorité les requêtes munies d'un indice de priorité élevé sont non seulement traitées plus rapidement, mais aussi elles seront affectées dans les secteurs les plus écologiques (*Voir* Figure 3.9).

En analysant les courbes obtenues, nous avons relevé les observations suivantes :

L'approche hybride avec indice de priorité utilise plus de nœuds et liens physiques que l'approche hybride de base :

Les Figures 4.12 et 4.13 illustrent respectivement le nombre de nœuds et de liens physiques utilisés pour traiter les 50 requêtes reçues pour les trois approches. En ajoutant l'indice de priorité, seules les requêtes dont l'indice est égal à un sont affectées au premier secteur (Voir Figure 3.9). En effet, cette nouvelle approche oblige le FIV à chercher des ressources disponibles dans d'autres secteurs d'où l'augmentation du nombre de nœuds et de liens physiques utilisés (Voir Tableau 4.3).

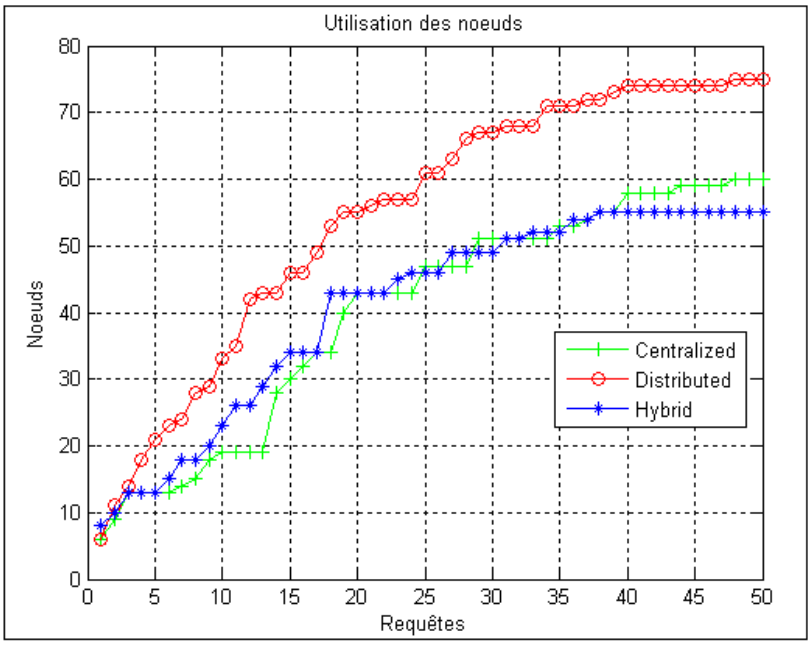

Figure 4.12 Nombre de nœuds physiques utilisés

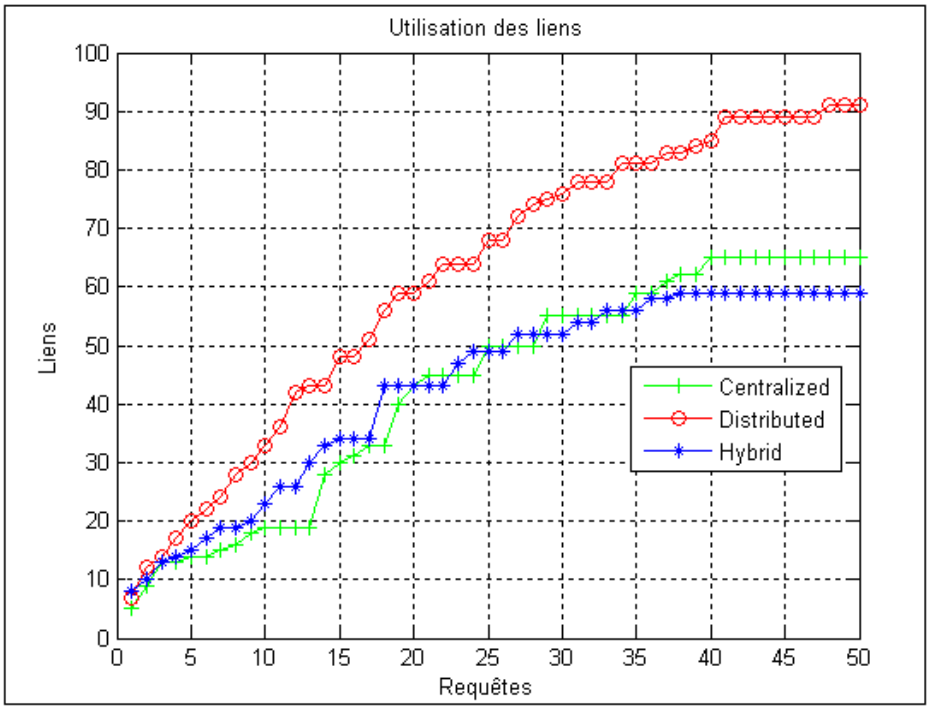

Figure 4.13 Nombre de liens physiques utilisés

Tableau 4.3 Nombre de nœuds et de liens utilisés

	Nœud utilisés	**Liens utilisés**
Approche hybride de base	28	30
Approche hybride avec indice de priorité	55	59

Le temps d'exécution des requêtes pour l'approche hybride a augmenté :

La Figure 4.14 illustre l'évolution du temps d'exécution des requêtes. Ce temps est passé de 59 secondes pour l'approche hybride de base à 155 secondes pour l'approche hybride avec indice de priorité. En effet, plus les nœuds physiques choisis sont éparpillés dans le réseau

physique, plus les chemins physiques utilisés risquent d'être longs. Cette situation est de plus en plus présente lorsque le nombre de requêtes augmente. Ceci engendre un temps de traitement plus important pour le FIV afin qu'il trouve le meilleur chemin.

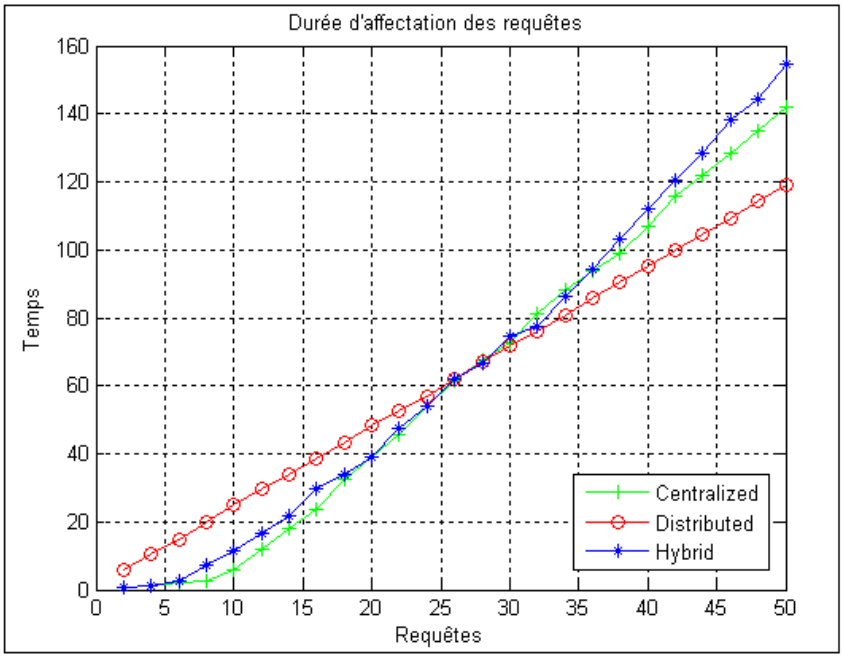

Figure 4.14 Affectation des requêtes en fonction du temps

Le coût en CO_2 engendré par l'approche hybride a augmenté :

La Figure 4.15 illustre la variation du coût en émission de CO_2 des requêtes. Nous remarquons que le coût en CO_2 engendré par notre algorithme n'est plus nul parce que les nœuds sont affectés dans différents secteurs dépendamment de l'indice de priorité des requêtes.

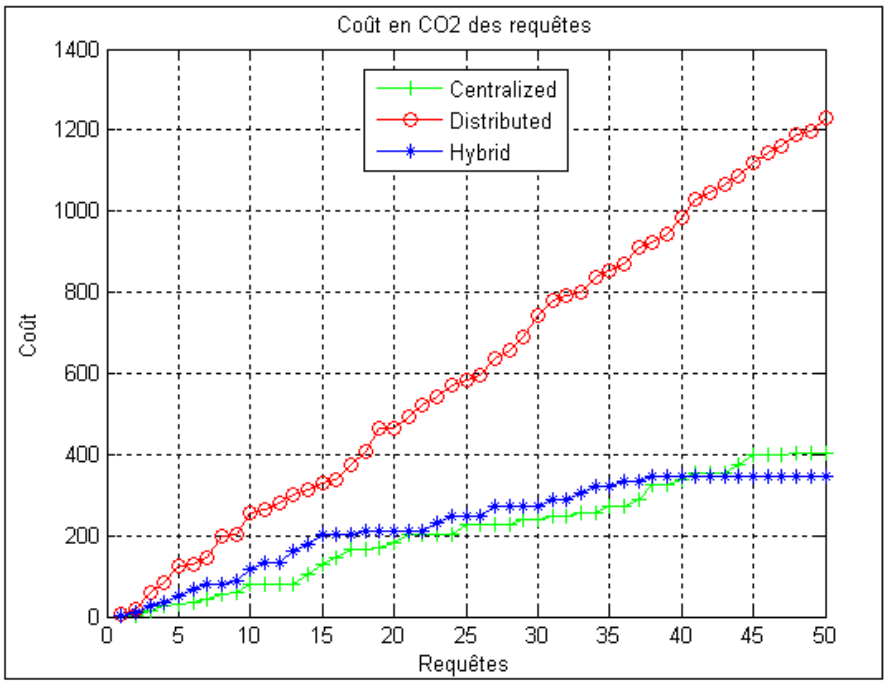

Figure 4.15 Coût en émission de CO_2 des requêtes

Revenu et coût plus importants :

Les Figures 4.16 et 4.17 illustrent respectivement la variation du coût physique et du revenu des requêtes. Nous remarquons que ces valeurs ont augmenté avec la nouvelle approche hybride parce que la consommation en bande passante des requêtes est plus importante.

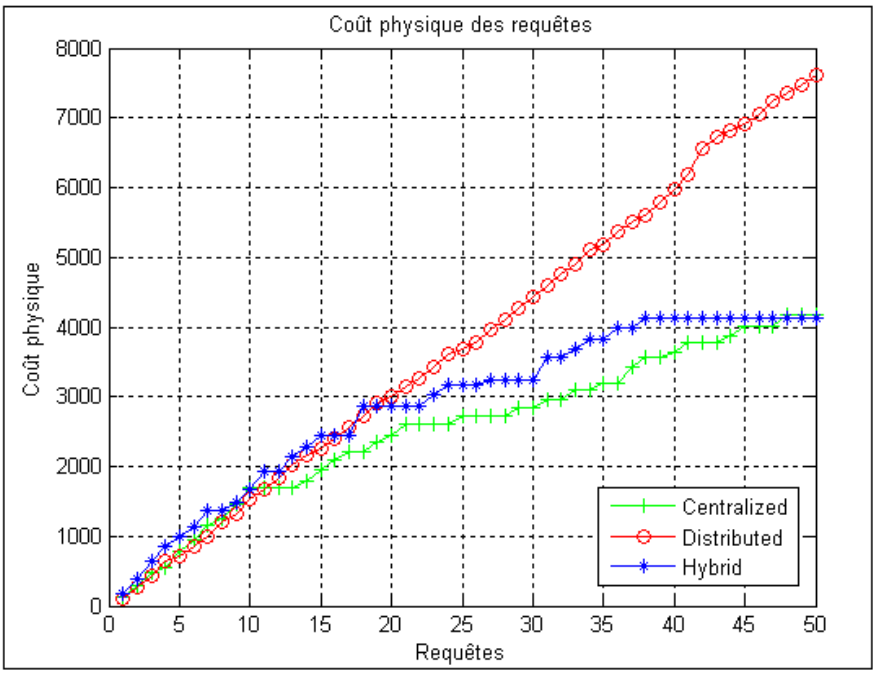

Figure 4.16 Coût des requêtes en fonction du temps

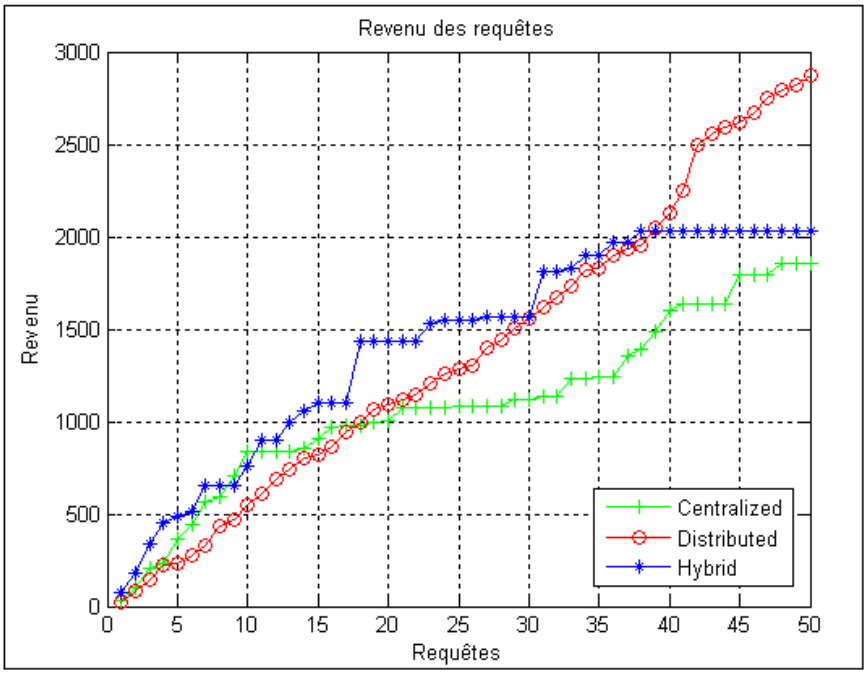

Figure 4.17 Revenu des requêtes en fonction du temps

4.2.6 Évaluation de l'approche hybride avec indice de priorité et indice de localisation

Nous rappelons que dans l'approche hybride avec indice de priorité et indice de localisation, chaque requête est affectée à l'une des trois régions du réseau physique dépendamment de son indice de localisation (*Voir* Figure 3.10).

En analysant les courbes obtenues, nous avons relevé les observations suivantes :

L'approche hybride avec indice de priorité et indice de localisation utilise plus de nœuds et liens physiques que l'approche hybride de base :

Les Figures 4.18 et 4.19 illustrent respectivement le nombre de nœuds et liens physiques utilisés pour traiter les 50 requêtes reçues pour les trois approches. En ajoutant l'indice de priorité ainsi que l'indice de localisation, les requêtes seront affectées dans l'une des trois régions de l'infrastructure physique dépendamment de leur indice de localisation. Le FIV se basera ensuite sur l'indice de priorité de chaque requête afin de déterminer le secteur prioritaire dont les ressources seront utilisées par les nœuds et liens de cette requête. Cette nouvelle approche permet au FIV l'accès aux ressources d'un nombre plus important de nœuds et de liens dépendamment des deux indices de la requête (Voir Tableau 4.4).

Tableau 4.4 Nombre de nœuds et de liens utilisés dans la simulation 3

	Nœud utilisés	**Liens utilisés**
Approche hybride de base	28	30
Approche hybride avec indice de priorité	55	59
Approche hybride avec indice de priorité et indice de localisation	55	56

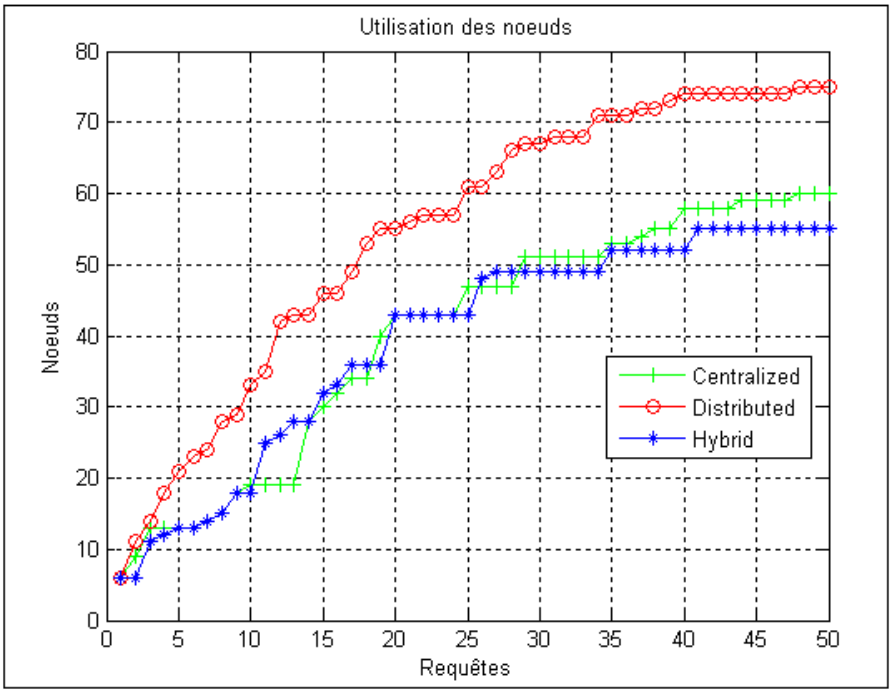

Figure 4.18 Nombre de nœuds physiques utilisés

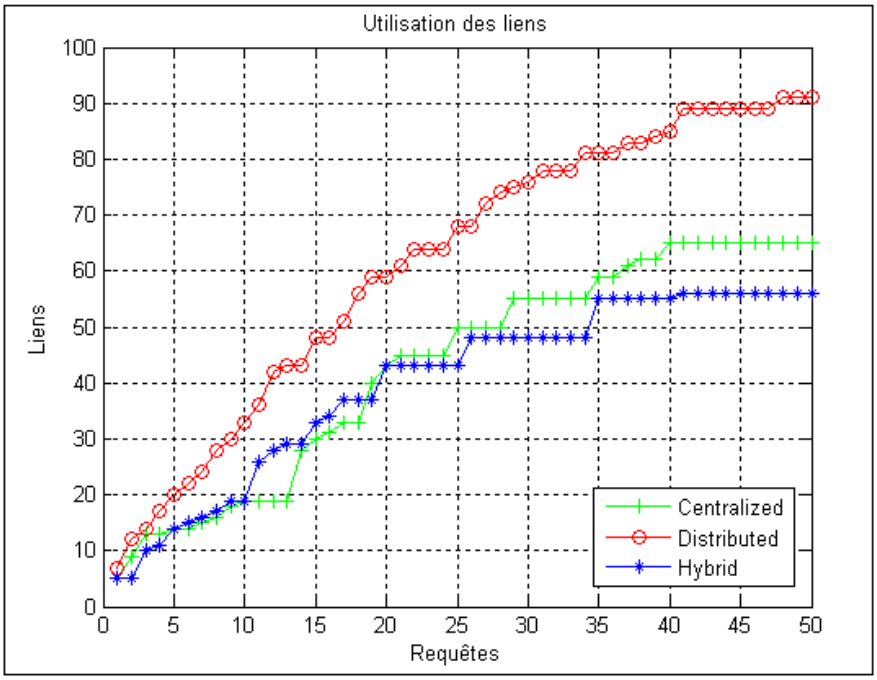

Figure 4.19 Nombre de liens physiques utilisés

Le temps d'exécution des requêtes pour l'approche hybride a encore augmenté :

La Figure 4.20 illustre l'évolution du temps d'exécution des requêtes. Ce temps a augmenté de 13 secondes par rapport à la précédente simulation. En effet, le temps d'exécution des requêtes a passé de 155 secondes pour l'approche hybride avec indice de priorité à 168 secondes pour l'approche hybride avec indice de priorité et indice de localisation. En effet, le FIV est contraint d'affecter chaque requête dans l'une des trois régions de l'infrastructure physique. Cette approche limite les ressources disponibles pour les requêtes. Cette limitation rend le temps de traitement des requêtes plus important, spécifiquement lorsque le nombre de requêtes augmente.

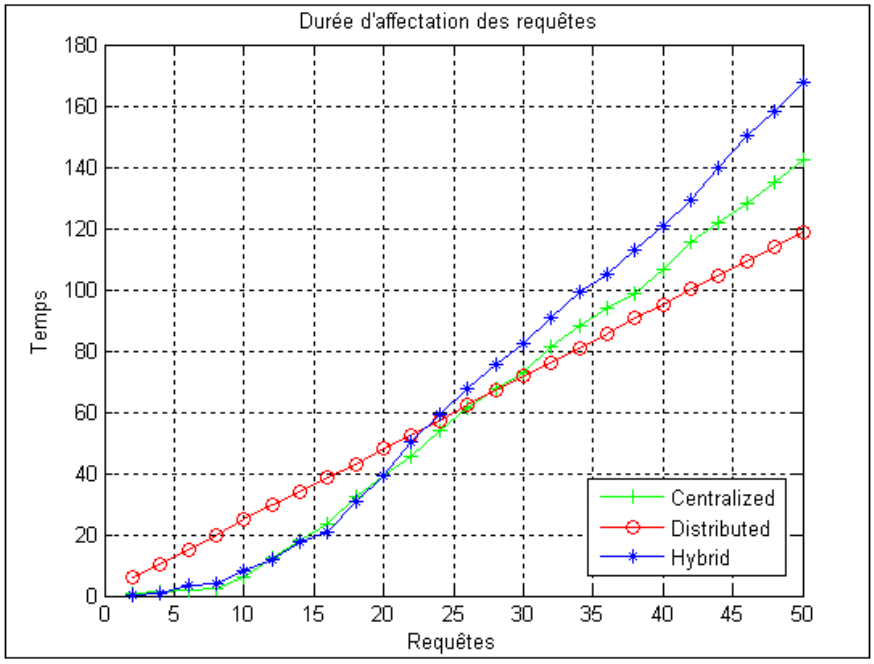

Figure 4.20 Affectation des requêtes en fonction du temps

Le coût en CO_2 engendré par l'approche hybride est aléatoire:

La Figure 4.21 illustre la variation du coût en émission de CO_2 des requêtes. Le choix du secteur dans lequel la requête sera affectée dépend directement de son indice de priorité. Cet indice est aléatoire d'où l'allure de la courbe de la Figure 4.21.

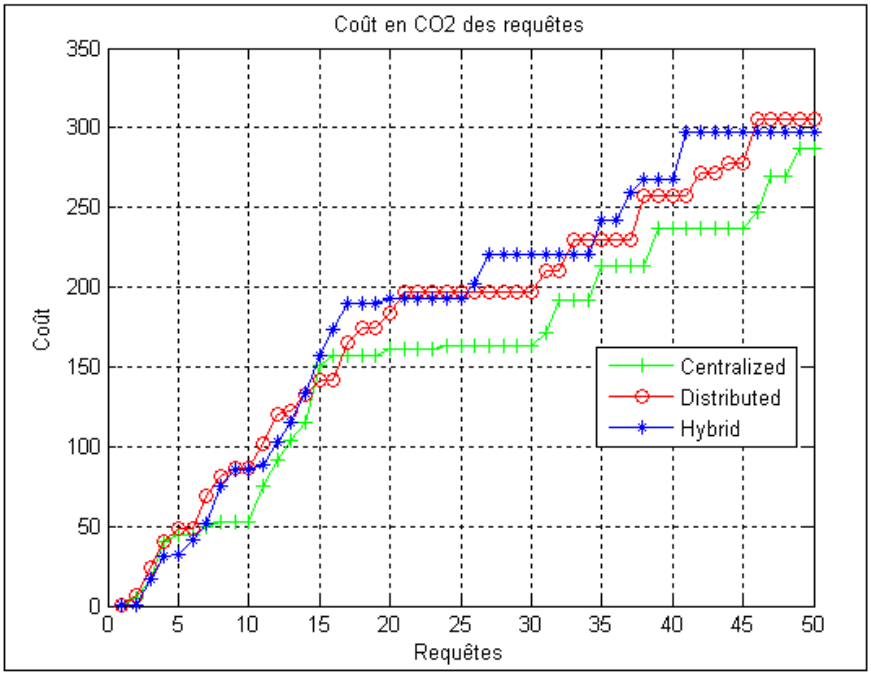

Figure 4.21 Coût en CO_2 des requêtes en fonction du temps

Rejet plus important des requêtes :

Les courbes du coût et de revenu contiennent plusieurs sections où la pente est nulle (Voir Figures 4.22 et 4.23). Ceci veut dire que plusieurs requêtes sont rejetées vu le manque de ressources physiques disponibles. En effet, le FIV est contraint à limiter ces recherches à l'une des trois régions du réseau physique dépendamment de l'indice de localisation de l'indice de localisation de la requête.

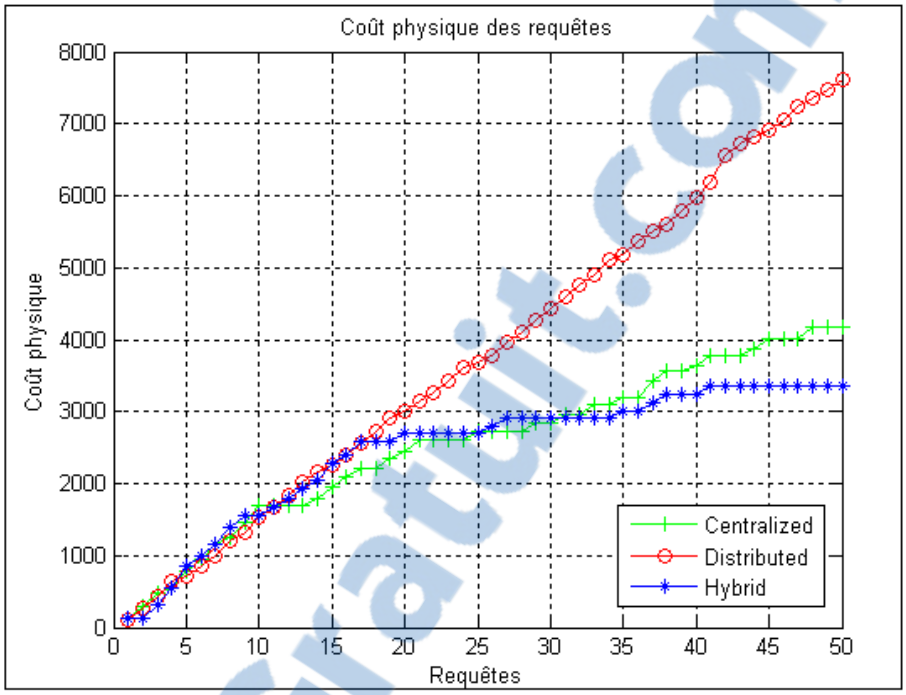

Figure 4.22 Coût des requêtes en fonction du temps

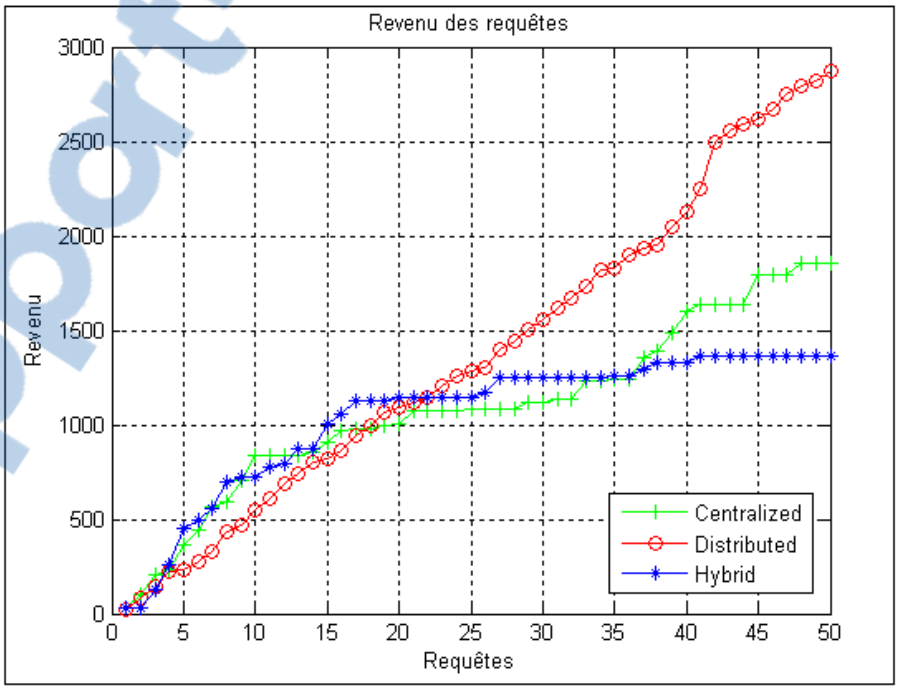

Figure 4.23 Revenu des requêtes en fonction du temps

4.2.7 Analyse sommaire

Dans l'approche hybride de base, le NC se charge de décomposer les requêtes reçues en sous-requêtes et de les envoyer au NG correspondant, dépendamment des contraintes de liens de ces dernières. Chaque NG vise à affecter les sous-requêtes dans les secteurs dont les ressources sont les moins polluantes. Ce traitement se fait selon l'ordre d'arrivée des requêtes. Les résultats obtenus ont montré que le temps d'exécution de cette approche est plus rapide que celui des deux autres approches proposées par (Minlan Yu, 2008) et par (Houidi, Louati et Zeghlache, 2008b). Aussi, les ressources utilisées par l'approche hybride de base sont les moins polluantes et le coût généré par cette dernière est le plus faible. Ces résultats sont dus entres autres à la méthode de sélection de ressources qui favorise les ressources disponibles dans le secteur avec le plus petit facteur d'émission de CO_2, à savoir le secteur 1.

Certes, les résultats obtenus sont parfaits, mais les conditions utilisées sont loin d'être réalistes. En effet, dans cette approche nous avons considéré que les requêtes arrivent avec la même priorité. Cependant, l'accessibilité aux ressources doit dépendre de la nature du contrat signé entre le FS et le FIV. Par exemple, le FS doit payer plus pour accéder aux ressources les moins polluantes, car ce choix peut lui apporter d'éventuelles primes et/ou subventions gouvernementales. Pour cette raison, nous avons ajouté un indice de priorité à chaque requête envoyée. Cet indice permet de classer les requêtes dans trois files d'attente avec des classes de priorités différentes. Les performances de cette approche sont avérées moins bonnes que celles de l'approche hybride de base. En effet, nous avons relevé une nette augmentation du temps de traitement des requêtes, mais il demeure moins important que celui des approches centralisée et distribuée. Aussi, le coût en CO_2 engendré a augmenté vu l'aspect aléatoire de l'indice de priorité affecté à chaque requête. Le coût physique des requêtes a aussi augmenté, car les nœuds physiques utilisés sont plus dispersés géographiquement et donc la bande passante requise pour satisfaire les contraintes de liens virtuels est plus importante.

Par ailleurs, les FS peuvent exiger, pour des raisons de sécurité par exemple, que leurs réseaux virtuels soient déployés à travers des infrastructures physiques situées dans des zones géographiques bien déterminées. Pour cette raison, nous avons ajouté un indice de localisation pour chaque requête. Cet indice exige du NG de limiter ses recherches à une région bien déterminée. Cette contrainte a engendré un rejet plus important des requêtes par manque de ressources.

Ainsi, les performances de l'approche hybride proposée dans le cadre de ce projet en termes de coût de ressources physiques, de revenu pour le réseau virtuel, de temps de traitements des requêtes et de coût en CO_2 peuvent variées selon les contraintes exigées par les fournisseurs de services telles que la priorité d'accès aux ressources les plus verts et la localisation géographique de ces dernières.

CONCLUSION

La conception du futur réseau Internet doit se baser sur une approche permettant aux différentes architectures de réseaux de communiquer et partager le même réseau physique. La notion de virtualisation des réseaux peut jouer un rôle très important dans ce scénario parce qu'elle permet à plusieurs réseaux virtuels de coexister et de partager les ressources d'une ou de plusieurs infrastructures physiques.

Dans une architecture de virtualisation de réseaux, les trois principaux acteurs sont : le fournisseur de service (FS), le fournisseur d'infrastructure physique (FIP), le fournisseur d'infrastructure virtuelle (FIV). Le FS est l'entité qui offre des services aux utilisateurs. Le FIP est l'entité responsable du déploiement de l'infrastructure physique et de la gestion de ses ressources. Le FIV est l'entité qui loue les ressources physiques offertes par les FIP afin de déployer des réseaux virtuels.

Un des défis les plus importants de ce concept est d'optimiser l'utilisation des ressources du réseau physique en appliquant des algorithmes d'allocation de ressources efficaces. Plusieurs stratégies de gestion de ressources ont été proposées et testées dans des travaux de recherches antérieurs. Certaines méthodes s'appuient sur une approche centralisée. En d'autres termes, un seul nœud du réseau physique sera responsable de la sélection et de l'allocation des différentes ressources disponibles. Si le réseau est très dynamique, ce nœud peut être rapidement surchargé et ainsi il peut engendrer un goulot d'étranglement. D'autres approches, dites distribuées, répartissent cette tâche entre plusieurs nœuds. Cette méthode peut s'avérer très gourmande en ressources.

Notre solution consiste à combiner les deux approches en utilisant un nœud central (NC) pour la réception et la décomposition des requêtes en sous-requêtes dépendamment des contraintes de liens de ces dernières et trois autres nœuds de gestion (NG) pour l'intégration des sous-requêtes dans l'infrastructure physique. Contrairement à tous les algorithmes présents dans la littérature, notre solution vise aussi à minimiser l'émission du CO_2 résultant

de la consommation d'énergie des différents réseaux virtuels en favorisant les ressources des nœuds se trouvant dans des secteurs verts desservis par des réseaux utilisant des énergies propres.

Nous avons proposé trois approches hybrides différentes : approche hybride de base, approche hybride avec indice de priorité et approche hybride avec indice de priorité et indice de localisation. Dans l'approche hybride de base, le FS envoie au FIV une ou plusieurs requêtes de création de réseaux virtuels. Chaque requête est reçue par le NC du FIV en question. Afin d'alléger le traitement de ces requêtes, Le NC divise la requête en plusieurs sous-requêtes selon le type de contrainte identifiée au niveau des liens virtuels. En effet, les requêtes sont décomposées en trois types de sous-requêtes : avec contrainte de délai, avec contrainte de perte de paquet et sans contraintes. Ces sous-réseaux sont ensuite envoyés aux NGs correspondants. Ces derniers sont responsables de l'affectation des sous-requêtes aux nœuds et liens de l'infrastructure physique en favorisant les secteurs les plus écologiques.

Dans l'approche hybride de base, les requêtes sont traitées selon leur ordre d'arrivée. Dans l'approche hybride avec indice de priorité, nous avons affecté un indice de priorité à chaque requête. Cet indice permet de classer les requêtes dans trois files d'attente avec des classes de priorités différentes. Dans l'approche hybride avec indice de priorité et indice de localisation, nous avons ajouté un indice de localisation pour chaque requête qui spécifie dans quelle région de l'infrastructure physique la requête doit être intégrée.

Nous avons comparé les performances de nos trois approches avec deux autres algorithmes présents dans la littérature utilisant des méthodes d'affectation de nœuds et de liens différentes. Le premier algorithme, basé sur une approche centralisée, est le *Baseline VN Embedding Algorithm* proposé par (Minlan Yu, 2008). Le deuxième algorithme, le *Distributed Virtual Network Mapping Algorithm*, est proposé par (Houidi, Louati et Zeghlache, 2008b) et adopte une approche distribuée.

Les résultats obtenus montrent que l'approche hybride de base est la plus performante en termes de rapidité de traitement. En plus, elle génère le coût en termes d'émission de CO_2 et le coût physique les plus faibles. L'ajout de l'indice de priorité a engendré une augmentation des valeurs de ces coûts. L'indice de localisation a quant à lui limiter les ressources valables pour chaque requête d'où le rejet important des requêtes relevé.

Ainsi, les performances de l'approche hybride proposée varient en fonction des contraintes exigées par les fournisseurs de service. Plus, ces dernières sont importantes plus les performances sont moins bonnes. Cependant, il est important de nos jours de concevoir des réseaux plus verts afin de préserver notre environnement et notre planète. Par ailleurs, la priorité d'accès aux ressources les plus verts est une contrainte importante, voire indispensable, dans la conception du réseau internet du futur.

RECOMMANDATIONS

Dans les travaux futurs, plusieurs améliorations peuvent être apportées aux approches proposées dans ce mémoire.

Premièrement, pour chaque réseau virtuel qui a été accepté par un FIV, nous recommandons d'affecter une durée de vie qui correspond, dans des conditions réelles, à la durée du contrat entre le FS et le FIV. Après l'échéance de cette durée, le FIV libèrera les ressources allouées à ce dernier. Ainsi, une partie des requêtes mises en file d'attente par le FIV par manque de ressources peuvent être intégrées dans l'infrastructure physique.

Deuxièmement, nous recommandons de proposer et d'implémenter un système de gestion de défaillance dans les architectures de virtualisation de réseaux. En effet, plusieurs pannes possibles peuvent affecter les nœuds et/ou les liens de l'infrastructure physique. Ce système doit donc proposer des solutions en termes de gestion des réseaux virtuels en cas de défaillance des nœuds et/ou liens hébergeant ces réseaux.

Troisièmement, nous recommandons la réalisation des tests de performance avec un banc de test tel que PlanetLab. En effet, l'utilisation de l'environnement Matlab ne permet pas d'évaluer l'apport de l'architecture hybride en termes de traitement parallèle des requêtes. Nous rappelons que le concept hybride est basé sur le partage du traitement des requêtes entre le NC et les trois NG. De plus, l'utilisation d'un banc de test permettra de tester les approches proposées sur le réseau internet « réel » et de déceler la moindre anomalie qui a pu échapper à la simulation réalisée avec Matlab.

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

Veuillez sélectionner un type de document autre que « Generic » afin de faire afficher la référence bibliographique.

Anderson, Thomas, Larry Peterson, Scott Shenker et Jonathan Turner. 2005. « Overcoming the Internet Impasse through Virtualization ». *Computer,* vol. 38, n^{o} 4, p. 34-41.

Aun Haider, Richard Potter, Akihiro nakao. 2009. « Challenges in Resource Allocation in Network Virtualization ». In *20th ITC Specialist Seminar*. (Vietnam).

Beitollahi, Hakem, et Geert Deconinck. 2008. « An overlay protection layer against denial-of-service attacks ». In *IPDPS 2008 - 22nd IEEE International Parallel and Distributed Processing Symposium, April 14, 2008 - April 18, 2008*. (Miami, FL, United states), p. IEEE Computer Society Technical Committee on Parallel Processing. Coll. « IPDPS Miami 2008 - Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium, Program and CD-ROM »: Inst. of Elec. and Eng. Computer Society <http://dx.doi.org/10.1109/IPDPS.2008.4536157>.

Biswas, J., A. A. Lazar, J. F. Huard, Lim Koonseng, S. Mahjoub, L. F. Pau, M. Suzuki, S. Torstensson, Wang Weiguo et S. Weinstein. 1998. « The IEEE P1520 standards initiative for programmable network interfaces ». *Communications Magazine, IEEE,* vol. 36, n^{o} 10, p. 64-70.

Bo, Lv, Wang Zhenkai, Huang Tao, Chen Jianya et Liu Yunjie. 2010. « A Hierarchical Management Architecture for Virtual Network Mapping ». In *Internet Technology and Applications, 2010 International Conference on*. (20-22 Aug. 2010), p. 1-4.

Bonnet, Renaud. 2002. « Virtualisation du stockage : fédérer les volumes en une unique ressource ». <http://www.01net.com/editorial/189258/virtualisation-du-stockage-federer-les-volumes-en-une-unique-ressource/>.

Byers, J. W., J. Considine, M. Mitzenmacher et S. Rost. 2004. « Informed content delivery across adaptive overlay networks ». *Networking, IEEE/ACM Transactions on,* vol. 12, n^{o} 5, p. 767-780.

Campbell, Andrew T., Herman G. De Meer, Michael E. Kounavis, Kazuho Miki, John B. Vicente et Daniel Villela. 1999. « A survey of programmable networks ». *SIGCOMM Comput. Commun. Rev.,* vol. 29, n^{o} 2, p. 7-23.

Carbone, Marta, et Luigi Rizzo. 2009. « Adding emulation to planetlab nodes ». In *Proceedings of the 5th international student workshop on Emerging networking experiments and technologies*. (Rome, Italy), p. 41-42. 1659020: ACM.

Chowdhury, Mosharaf Kabir. 2008. « Identity Management and Resource Allocation in the Network Virtualization Environment ». University of Waterloo.

Chowdhury, N. M. M. K., M. R. Rahman et R. Boutaba. 2009. « Virtual Network Embedding with Coordinated Node and Link Mapping ». In *INFOCOM 2009, IEEE*. (19-25 April 2009), p. 783-791.

Chowdhury, N. M. Mosharaf Kabir, et Raouf Boutaba. 2010. « A survey of network virtualization ». *Computer Networks,* vol. 54, n° 5, p. 862-876.

Chun, Brent, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak et Mic Bowman. 2003. « PlanetLab: an overlay testbed for broad-coverage services ». *SIGCOMM Comput. Commun. Rev.,* vol. 33, n° 3, p. 3-12.

Corp., Planet Energy. 2010. « Calculate your carbon footprint ». <http://www.livclean.ca/index.php>.

da Silva, S., Y. Yemini et D. Florissi. 2001. « The NetScript active network system ». *Selected Areas in Communications, IEEE Journal on,* vol. 19, n° 3, p. 538-551.

Dekeris, B., T. Adomkus et A. Budnikas. 2006. « Analysis of qos assurance using weighted fair queueing (WQF) scheduling discipline with low latency queue (LLQ) ». In *Information Technology Interfaces, 2006. 28th International Conference on*. (0-0 0), p. 507-512.

Dhaini, A. R., Ho Pin-Han et Jiang Xiaohong. 2010. « Performance Analysis of QoS-Aware Layer-2 VPNs over Fiber-Wireless (FiWi) Networks ». In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*. (6-10 Dec. 2010), p. 1-6.

Diab, Wafaa Bou, Samir Tohme et Carole Bassil. 2008. « VPN Analysis and New Perspective for Securing Voice over VPN Networks ». In *Proceedings of the Fourth International Conference on Networking and Services*. p. 73-78. 1396690: IEEE Computer Society.

Dong, Renfei, et Kaiqi Zou. 2008. « A novel approach for simulating peer-to-peer based ALM using ns2 ». In *IT in Medicine and Education, 2008. ITME 2008. IEEE International Symposium on*. (12-14 Dec. 2008), p. 1073-1078.

El Barachi, M., N. Kara et R. Dssouli. 2010. « Towards a service-oriented network virtualization architecture ». In *Kaleidoscope: Beyond the Internet? - Innovations for Future Networks and Services, 2010 ITU-T*. (13-15 Dec. 2010), p. 1-7.

Elaoud, M., A. McAuley, G. Kim et J. Chennikara. 2005. « Cross-layer optimized unicast and multicast routing on overlay networks ». In *Military Communications Conference, 2005. MILCOM 2005. IEEE*. (17-20 Oct. 2005), p. 1122-1126 Vol. 2.

Gurr, Coby. 2008. *Microsoft Windows Vista Migration Through Application Virtualization*. <http://www.dell.com/downloads/global/power/ps1q08-20080154-LANDesk.pdf>.

Houidi, I., W. Louati et D. Zeghlache. 2008a. « A Distributed and Autonomic Virtual Network Mapping Framework ». In *Autonomic and Autonomous Systems, 2008. ICAS 2008. Fourth International Conference on*. (16-21 March 2008), p. 241-247.

Houidi, I., W. Louati et D. Zeghlache. 2008b. « A Distributed Virtual Network Mapping Algorithm ». In *Communications, 2008. ICC '08. IEEE International Conference on*. (19-23 May 2008), p. 5634-5640.

Houidi, I., W. Louati, D. Zeghlache et S. Baucke. 2009. « Virtual Resource Description and Clustering for Virtual Network Discovery ». In *Communications Workshops, 2009. ICC Workshops 2009. IEEE International Conference on*. (14-18 June 2009), p. 1-6.

IBM, Systèmes et technologie. 2011. « IBM System Storage SAN Volume Controller ». <http://public.dhe.ibm.com/common/ssi/ecm/fr/tsd00254frfr/TSD00254FRFR.PDF>.

Jannotti, J., D. K. Gifford, K. Johnson, M. F. Kaashoek et J. W. O'Toole, Jr. 2000. « Overcast: reliable multicasting with an overlay network ». In *Proceedings of OSDI 2000. 4th Symposium on Operating Systems Design and Implementation, 23-25 Oct. 2000*. (Berkeley, CA, USA), p. 197-212. Coll. « Proceedings of the Fourth Symposium on Operating Systems Design and Implementation (OSDI 2000) »: USENIX Assoc.

Joe Touch , Steve Hotz 1998. « The X-Bone ». In *Proc. Global Internet Mini-Conference / Globecom*.

Kawahara, R., S. Harada, N. Kamiyama, T. Mori, H. Hasegawa et A. Nakao. 2011. « Traffic Engineering Using Overlay Network ». In *Communications (ICC), 2011 IEEE International Conference on*. (5-9 June 2011), p. 1-6.

Kawahara, Ryoichi, Satoshi Kamei, Noriaki Kamiyama, Haruhisa Hasegawa, Hideaki Yoshino, Eng Keong Lua et Akihiro Nakao. 2009. « A method of constructing QoS overlay network and its evaluation ». In *Proceedings of the 28th IEEE conference on Global telecommunications*. (Honolulu, Hawaii, USA), p. 2059-2064. 1811722: IEEE Press.

Kyriakos Zarifis, Georgia Kontesidou. 2009. « Openflow Virtual Networking: A Flow-Based Network Virtualization Architecture ». Royal Institute of Technology.

L.Kristiansen. 1998. « The TINA Architecture ». *Telektronikk,* vol. 94, p. 95-106.

Leon-Garcia, A., et L. G. Mason. 2003. « Virtual network resource management for next-generation networks ». *Communications Magazine, IEEE,* vol. 41, n° 7, p. 102-109.

Medina, A., A. Lakhina, I. Matta et J. Byers. 2001. « BRITE: an approach to universal topology generation ». In *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2001. Proceedings. Ninth International Symposium on*. (2001), p. 346-353.

Meempat, G., et M. K. Sundareshan. 1993. « Optimal channel allocation policies for access control of circuit-switched traffic in ISDN environments ». *Communications, IEEE Transactions on,* vol. 41, n° 2, p. 338-350.

Melakessou, F., et T. Engel. 2009. « Network Traffic Simulator 2.0: Simulating the internet traffic ». In *Open-source Software for Scientific Computation (OSSC), 2009 IEEE International Workshop on*. (18-20 Sept. 2009), p. 139-147.

Michael Till Beck, Andreas Fischer, and Hermann de Meer. 2012. « Distributed Virtual Network Embedding ». In *7th GI/ITG KuVS Workshop on Future Internet*

Minlan Yu, Yung Yi, Jennifer Rexford and Mung Chiang 2008. « Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration ». *Computer Communication Review,* vol. 38, p. 19-29.

Mohapatra, P., C. Metz et Cui Yong. 2007. « Layer 3 VPN Services over IPv6 Backbone Networks: Requirements, Technology, and Standardization Efforts ». *Communications Magazine, IEEE,* vol. 45, n° 4, p. 32-37.

Olivier Festor, Isabelle Chrisment, Eric Fleury. 2000. « Les réseaux programmables 1.0 ». Institut National de Recherche en Informatique et en Automatique.

Peterson, Larry, Tom Anderson, David Culler et Timothy Roscoe. 2003. « A blueprint for introducing disruptive technology into the Internet ». *SIGCOMM Comput. Commun. Rev.,* vol. 33, n° 1, p. 59-64.

Pradeep Padala, Xiaoyun Zhu, Zhikui Wang, Sharad Singhal, Kang G. Shin. 2007. « Performance Evaluation of Virtualization Technologies for ServerConsolidation ». *HPL,* vol. 2007.

Rooney, S., J. E. van der Merwe, S. A. Crosby et I. M. Leslie. 1998. « The Tempest: a framework for safe, resource assured, programmable networks ». *Communications Magazine, IEEE,* vol. 36, n° 10, p. 42-53.

Schmitt, Damien. 2008. « Introduction à la virtualisation du stockage ». <http://www.virtualisation-news.com/2008/09/dossier-introduction-a-la-virtualisation-du-stokage.html>.

Spyropoulos, Thrasyvoulos, Serge Fdida et Scott Kirkpatrick. 2007. « Future internet: fundamentals and measurement ». *SIGCOMM Comput. Commun. Rev.,* vol. 37, n° 2, p. 101-106.

Suzuki, J., et Y. Yamamoto. 2000. « iNet: an extensible framework for simulating immune network ». In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*. (2000) Vol. 1, p. 119-124 vol.1.

Turner, J. Lu and J. 2006. « Efficient mapping of virtual networks onto a shared substrate ». In *WUCSE*.

Turner, J. S., et D. E. Taylor. 2005. « Diversifying the Internet ». In *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*. (2-2 Dec. 2005) Vol. 2, p. 6 pp.-760.

Umarikar, Amod C., Tusharkant Mishra et L. Umanand. 2006. « Bond graph simulation and symbolic extraction toolbox in MATLAB/SIMULINK ». *Journal of the Indian Institute of Science,* vol. 86, n° 1, p. 45-68.

VMware. 2011. « VMware ESX and VMware ESXi The Market Leading Prodution-Proven Hypervisors ». <http://www.vmware.com/files/pdf/VMware-ESX-and-VMware-ESXi-DS-EN.pdf>.

You, Guohua, et Ying Zhao. 2011. « A weighted-fair-queuing (WFQ)-based dynamic request scheduling approach in a multi-core system ». *Future Generation Computer Systems*.

Yu, Minlan, Yung Yi, Jennifer Rexford et Mung Chiang. 2008. « Rethinking virtual network embedding: substrate support for path splitting and migration ». *SIGCOMM Comput. Commun. Rev.,* vol. 38, n° 2, p. 17-29.

Yuen, Marco. 2006. « GENI in the Cloud ». University of Victoria.

Zegura, E. W., K. L. Calvert et S. Bhattacharjee. 1996. « How to model an internetwork ». In *INFOCOM '96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE*. (24-28 Mar 1996) Vol. 2, p. 594-602 vol.2.

Zeng, Xiyang, et Chuanqing Cheng. 2009. « Research on VLAN Technology in L3 Switch ». In *Intelligent Information Technology Application, 2009. IITA 2009. Third International Symposium on*. (21-22 Nov. 2009) Vol. 3, p. 722-725.

Zhu, Y., et M. Ammar. 2006. « Algorithms for Assigning Substrate Network Resources to Virtual Network Components ». In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*. (April 2006), p. 1-12.