UNIVERSITÉ DU QUÉBEC À MONTREAL

GOSSIP ET LA CONVERGENCE DANS LES RÉSEAUX D'ÉQUIPEMENTS VIRTUALISÉS

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

MARTIN HÉROUX

FÉVRIER 2012

REMERCIEMENTS

Un merci spécial à ma conjointe et mes deux enfants qui ont dû développer leur sens de l'autonomie et leur patience.

Merci au professeur Omar Cherkaoui qui a encadré mes travaux. Sa grande patience, sa disponibilité et sa vive intelligence ont certainement contribué à mon développement et à faire de moi une meilleure personne.

Finalement, merci à mes employeurs et à mes clients qui m'ont régulièrement permis de voyager et de m'absenter de mon travail afin que je puisse terminer mon programme d'études.

TABLE DES MATIÈRES

LISTE DES FIGURES

LEXIQUE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

Anglais	*Français*	*Définition*
MAC address	Adresse MAC	En réseau informatique, une adresse MAC (Media Access Control address) est un identifiant physique stocké dans une carte réseau ou une interface réseau similaire et utilisé pour attribuer mondialement une adresse unique au niveau de la couche de liaison (couche 2 du modèle OSI).
Architecture Building Block (ABB)	Bloc de construction d'architecture	Un bloc de construction a une limite définie et est généralement reconnaissable en tant que « quelque chose » par un expert du domaine. Il peut s'assembler à d'autres blocs et il peut lui-même composer d'autres blocs. Dans notre travail, un bloc de construction d'architecture représente un fondement, un modèle commun, un modèle d'implantation en industrie ou une composante adaptée à notre travail.
Convergence	Convergence	La convergence est reliée au temps requis pour qu'un système dans un état stable subissant un changement puisse redevenir stable.

DHT	Table de hachage distribuée	Une table de hachage distribuée (ou DHT pour Distributed Hash Table) est une technologie permettant l'identification et l'obtention, dans un système réparti, d'une information.
Fanout	Sortance	Nombre maximal de composants qui peuvent être alimentés simultanément.
Gossip	Rumeur	Mot anglais pour rumeur.
GRE	GRE	Protocole d'encapsulation de différentes couches du réseau dans le protocole IP. Cette technique permet de créer de lien point à point virtuel.
Hash	Hachage	Le hash (ou hachage) est le résultat d'une fonction cryptographique qui prend un bloc de données et qui retourne une chaîne de caractères de longueur fixe. Le résultat du hachage est toujours le même pour un même bloc de données.
Hypervisor	Hyperviseur	Un hyperviseur est une composante d'une architecture de virtualisation qui permet d'exécuter plusieurs systèmes d'exploitation simultanément sur un même ordinateur.
IAND	IAND	Acronyme pour Infect and Die, un protocole de type Gossip.
Intel x86 /Inet x86_64	Intel x86 /Inet x86_64	x86 et x86_64 réfèrent à une famille d'instruction-machine des processeurs Intel 8086. Les processeurs x86 composent aujourd'hui la quasi-totalité des ordinateurs personnels et serveurs dans la PME.
IOS	IOS	Internet Operating System : Systèmes d'exploitation spécialisée dans la gestion des ressources matérielles d'équipements de réseaux.

LIAND	LIAND	Acronyme pour Low Latency Infect and Die. Le résultat du travail de recherche présenté dans ce mémoire.
Meshing	Maillage	Architecture de réseau dans laquelle tous les nœuds qui correspondent aux points d'accès de l'infrastructure mise en place peuvent être reliés directement à d'autres nœuds situés dans leur entourage immédiat, afin de pouvoir relayer à l'un d'eux l'information qui leur est transmise à partir d'un terminal, de manière que celle-ci, à l'aide de routeurs, soit acheminée progressivement, de relais en relais, jusqu'à sa destination finale.
Omnet++	Omnet++	Omnet++ est un catalogue de fonctions de simulation modulaire, extensible fournissant un ensemble de composantes afin de réaliser des simulations informatiques.
OpenSource	OpenSource	L'OpenSource décrit des pratiques de production de développement de logiciels qui font la promotion de distribution des sources logicielles comme produit final et non des programmes compilés.
P2P	P2P	Acronyme pour peer-to-peer . Le P2P est un modèle d'architecture distribuée où les participants utilisent une partie de leurs ressources non utilisées (disque, CPU, bande passante) et les rend directement disponibles aux autres participants.
Resource slicing	Partitionnement de ressource	Processus de découpage dans le temps d'une ressource matérielle afin d'offrir des tranches de temps d'utilisation équitables entre plusieurs processus ou systèmes d'exploitation.

Ping	Ping	Ping est un outil d'administration servant à tester la disponibilité d'un nœud sur un réseau IP. Avec Ping, on peut aussi mesurer le temps que prend un paquet pour aller et revenir vers le nœud testé.
PlanetLab	PlanetLab	PlanetLab est un ordinateur disponible pour effectuer des expériences en réseautique et en système distribué. En 2010, le projet est composé de plus de 1 100 nœuds physiques repartis mondialement sur 500 sites.
Round	Itération	Utilisé dans le contexte de Gossip, ce terme représente une séquence de contaminations basée sur le *fanout*. En supposant un *fanout* de 2, un round représente les étapes que prennent 2 nœuds pour en contaminer 2 autres chacun.
Operating System (OS)	Système d'exploitation (SE)	Un système d'exploitation est un logiciel qui s'exécute sur un ordinateur et qui gère le matériel.
Decentralized system	Système décentralisé	La décentralisation est la mécanique par laquelle la prise de décision est dispersée afin de la rendre plus près de nœuds qui ont besoin de prendre des décisions. Dans le contexte P2P, un système décentralisé représente un réseau dont les prises de décision topologiques ne sont pas reliées à un serveur central, mais laissé aux nœuds participant au réseau.□
Unstructured System	Système non structuré	Un système non structuré est un système où les nœuds forment un *overlay* qui s'établit arbitrairement. Il n'y a pas de structure sous-jacente aidant à la gestion du système.
Topology	Topologie	Représentation schématique des différents nœuds d'un réseau et de leurs liaisons physiques.

TTL	TTL	Acronyme pour Time To Live. Le TTL est un champ dans un message. Chaque fois qu'un nœud traite le message, cette valeur est réduite de 1. Lorsque le TTL est à 0, alors le message est rejeté.
Tunelling	Tunnellisation	Les réseaux d'ordinateurs utilisent un protocole de tunnellisation quand le protocole de réseau (le protocole de livraison) encapsule un protocole différent. En utilisant un tunnel, on peut (par exemple) transporter les données d'un réseau incompatibles, ou de fournir un chemin d'accès sécurisé grâce à un réseau non sécurisé.
VIF	VIF	Acronyme pour Virtual Interface ou interface virtuelle de réseau.

RÉSUMÉ

L'Internet du futur devra certainement utiliser la virtualisation, car elle permet un passage flexible vers de nouveaux modèles de réseau, offre une abstraction des ressources, permet une meilleure utilisation de ces ressources ainsi que le partage de ces ressources.

La virtualisation des serveurs est déjà utilisée depuis plusieurs années dans les centres de données et sur Internet. L'abstraction du matériel, le partage des ressources et les facilités de déploiement ont permis d'évoluer vers le modèle des services infonuagiques.

On pense donc que l'Internet du futur doit passer par une virtualisation des équipements de réseau pour emprunter un chemin similaire à celui des serveurs vers l'informatique en nuage.

Pour faciliter la découverte des ressources d'un réseau d'équipements virtuels, il est préférable de ne pas contacter chaque équipement du réseau à tour de rôle. Pour accélérer la création d'un inventaire des ressources disponibles, il est nécessaire d'établir leur cartographie. Cette cartographie des ressources associées à leur nœud physique sera appelée topologie puisqu'elle tiendra compte des liaisons entre les ressources et les nœuds du réseau.

Le contexte de cette recherche est une approche de stabilisation rapide de la topologie des ressources mises en commun pour des réseaux dont les équipements ont été virtualisés.

La centralisation d'une topologie globale d'un réseau comme Internet ne pourrait être possible. La fréquence des mises à jour et la quantité d'opérations de lecture demanderaient une infrastructure incroyablement puissante pour supporter des millions de clients concurrents. La décentralisation est une approche qui permet de répondre à cette demande en puissance par la distribution massive de la charge de travail entre plusieurs ordinateurs. De plus, elle permet d'accroitre la tolérance aux

fautes, l'autoadaptation de la topologie, la réplication d'une large quantité de données et rapproche les informations vers les clients. À première vue, le modèle semble parfait, mais pour maintenir une telle topologie qui soit structurée autour d'un réseau, qui est lui-même en constante évolution, le modèle présente une complexité supplémentaire. En effet, pour maintenir une telle structure, il faut la mettre à jour à chaque changement. La décentralisation seule peut alors entrainer des connexions lentes entre deux nœuds qui sont relativement éloignés l'un de l'autre géographiquement et dont la latence entre ces liens peut être élevée. Ces connexions lentes peuvent ralentir les mises à jour de la topologie et donc ralentir la convergence (voir lexique) de l'information. L'utilisation d'une approche non structurée peut éliminer cette limitation. Chaque nœud participant à un système non structuré prend ses propres décisions. Ces décisions n'affectent pas les autres nœuds du système.

Le travail présenté dans ce mémoire utilise un réseau qui reflète une des directions d'exploration des projets de GENI [1], PlanetLab [2], VINI [3], Cabo [4], etc. Ce réseau utilise des routeurs virtuels dont les nœuds physiques partagent leurs ressources informatiques. Une approche de mises à jour décentralisées et non structurées sera utilisée dans le but de gérer la topologie et de répondre aux exigences de distribution, de robustesse, de croissance et d'extensibilité de cette dernière. Dans le réseau de routeurs virtuels du présent travail, chaque nœud physique possède sa propre copie de la topologie des ressources partagées par les autres nœuds physiques du réseau. Le défi principal que relève ce mémoire est la convergence rapide de cette topologie des ressources partagées, appliquée à un grand réseau.

Afin de réaliser les mises à jour de la topologie des ressources partagées, deux modèles d'architecture de système distribué ont été étudiés : le P2P (voir lexique) et le Gossip (voir lexique). Dans un premier temps, il sera démontré comment le modèle Gossip paraît être le mieux adapté au contexte du présent travail.

Dans un deuxième temps, l'expérience du protocole P2P Gnutella a fait ressortir qu'il est préférable de profiter de la topologie du réseau sur lequel on s'exécute. Le protocole Gossip sera amélioré en ce sens. Cette nouvelle version démontrera comment l'utilisation de la topologie du réseau physique peut être utilisée comme levier pour améliorer sa performance.

Finalement, il sera démontré en quoi l'amélioration apportée permet de stabiliser le temps de convergence d'une topologie décentralisée et non structurée indépendamment de la taille et de la latence d'un réseau, pourvu qu'il soit fortement maillé.

INTRODUCTION

La virtualisation des équipements de réseau est quelque chose qui est encore assez nouveau et mal défini. Nous ne connaissons pas réellement les impacts architecturaux sur les grands systèmes informatiques à plusieurs milliers de nœuds. Nous savons par contre que les modèles technologiques récents comme l'infonuagique (*Cloud Computing*) utilisent la virtualisation des équipements de réseau. Nous savons aussi qu'une des limitations de cette nouvelle architecture est la découverte des ressources de traitement et de la topologie par laquelle ces ressources sont organisées. L'état de ces ressources doit être connu afin de réaliser les nouvelles opérations de gestion dans le réseau d'équipements virtuels : instanciation d'équipements, arrêt des équipements, déplacement à chaud, augmentation/réduction de la capacité de traitement, etc.

Le présent travail propose une approche novatrice de découvertes de ressources dans le contexte d'un réseau d'équipements virtuels. C'est un problème important parce que l'état des ressources disponibles impacte directement les opérations de gestion et ultimement la topologie des routeurs. L'approche qui est proposée décentralise et distribue l'information afin que celle-ci ne possède pas de structure dépendante à la viabilité de l'information. L'effet de la décentralisation et de la distribution a un impact sur le protocole et la stratégie d'échange de l'information lors d'évènements dans le réseau. Dans le contexte d'un grand ensemble de nœuds, il n'est pas envisageable de les contacter un à un à tour de rôle afin d'effectuer la mise à

jour. Ce qui est décrit dans ce travail est l'algorithme d'un futur protocole qui pourra rapidement mettre à jour une grande topologie de nœuds partageant une immense cartographie de toutes les ressources de traitement disponibles sur un grand réseau.

Ce mémoire de recherche est structuré de la façon suivante : le chapitre 1 introduit le contexte du travail plus en détail, le chapitre 2 propose une revue de littérature expliquant d'où vient le besoin de virtualiser les équipements de réseau et la source de la problématique du présent travail, le chapitre 3 présente le P2P et le Gossip et fournit l'analyse qui nous a permis de choisir le protocole le plus approprié et, finalement, le chapitre 4 propose l'amélioration du modèle de protocole Gossip – Infect and Die.

CHAPITRE I

CONTEXTE DU TRAVAIL

La virtualisation des serveurs est une technologie qui permet d'exécuter plusieurs systèmes d'exploitation en même temps sur un même serveur physique [5]. La virtualisation est grandement utilisée dans les centres de données afin de maximiser l'utilisation des ressources matérielles et pour sa simplicité à déployer des serveurs virtuels.

Plusieurs travaux proposent d'appliquer cette technologie à des équipements de réseau [1] [2] [3] [6] [7]. La virtualisation dans les réseaux apporte des changements importants dans l'évolution de ces derniers. Plusieurs systèmes d'exploitation (Cisco IOS, Juniper, JunOS, Linux, etc.) peuvent cohabiter sur un même équipement physique. Cela permet par exemple d'utiliser ces systèmes d'exploitation simultanément pour tester de nouveaux protocoles sur des réseaux parallèles et en utilisant le même trafic [4]. Cela permet aussi d'héberger plusieurs topologies sur un même réseau physique [3]. Chacune de ces topologies étant indépendante l'une de l'autre, l'élaboration de ce type de réseau passe obligatoirement par l'utilisation de la virtualisation.

Un des éléments liés à la virtualisation est la capacité de partager des ressources matérielles non utilisées. Dans le cadre de la virtualisation des équipements de réseau,

le partage de ces ressources matérielles comme le CPU, la mémoire et la bande passante peuvent se faire sur une grande échelle. Les variations de ces ressources partagées peuvent être problématiques pour gestion globale des ressources du réseau et donc dans les décisions topologiques. Par exemple, sur un grand réseau, la gestion de ce type d'information, comme la quantité de processus ou la distribution de la charge de travail peut s'avérer complexe puisqu'il faut transmettre à tous les nœuds, les variations d'états des ressources du réseau.

L'approche qui a été retenue pour notre travail est de décentraliser l'information et de la gérer en réalisant une cartographie de ces ressources afin de pouvoir gérer les ressources de notre réseau sur une grande échelle. L'organisation de cette cartographie en topologie et l'efficience de la mise à jour de cette topologie sont les sujets qui composent ce mémoire.

Dans le contexte d'un parc de serveurs virtuels, les ressources sont gérées centralement par des consoles que les administrateurs utilisent. Une des opérations importantes dans la gestion des serveurs virtuels est la gestion de la capacité de traitement du parc de serveur [8] [9] physique qui héberge les serveurs virtuels [10]. Certains serveurs physiques deviennent surutilisés, alors que d'autres sont sous-utilisés. Afin d'assurer l'équilibrage des charges de travail, il suffit de déplacer l'exécution de certaines machines virtuelles d'un nœud physique à un autre. La centralisation de l'information permet de réaliser assez simplement cette opération.

Comme pour les serveurs, la gestion du réseau d'équipements virtuels implique de prendre en compte les fluctuations dans l'utilisation des ressources. Dans le but de répartir la charge de travail, les routeurs virtuels qui s'exécutent sur des routeurs physiques surutilisés peuvent être déplacés sur des routeurs physiques moins utilisés. Ce déplacement, qui peut être fréquent, des routeurs virtuels et l'instanciation de ces derniers font ressortir deux aspects importants : la taille du réseau et sa décentralisation rendent la sélection d'un nœud cible une opération minutieuse et

complexe. En effet, dans un contexte de réseau comme celui de la *National Science Foundation* (NSF), qui est un réseau de référence académique, on estimait, en 2002, à environs 13,000 le nombre routeurs périphériques (edge router) [64]. Dans un tel contexte, pour prendre des décisions comme le déplacement d'un routeur, il faut connaitre les types des interfaces du routeur cible ainsi que les capacités de traitement disponible, les capacités disponibles de mémoire et la bande passante pour accueillir le nouveau routeur physique. À la suite du déplacement des charges vers le routeur cible, il y aura la synchronisation de la topologie du réseau logique. Finalement, il y aura un relâchement des ressources informatique sur le routeur physique source. Ces variations dans l'utilisation des ressources doivent être diffusées vers tous les nœuds physiques du réseau.

Le développement d'une topologie qui associe les ressources disponibles à leurs nœuds physiques est donc nécessaire pour gérer les ressources disponibles aux machines virtuelles. La mise à jour de cette topologie est un élément important pour les réseaux routeurs virtuels à cause de l'échelle à laquelle elle s'applique. Cette problématique a été débattue à la conférence GRES 2010 dans un article [8] traitant d'une avenue possible pour améliorer le temps de convergence.

La principale motivation de ce mémoire consiste à proposer une approche pour améliorer le temps de convergence de l'information représentant les ressources partagées par différents routeurs physiques. La convergence est reliée au temps requis pour qu'un système dans un état stable subissant un changement puisse redevenir stable. Dans le contexte du réseau, le changement d'état des ressources vient des fluctuations dans l'utilisation des ressources informatiques partagées par les routeurs virtuels.

Les expérimentations qui ont été réalisées dans les réseaux de routeurs virtuels ont démontré un aspect important. Pour instancier un routeur virtuel ou pour déplacer son exécution, il est essentiel de connaitre les ressources CPU, la mémoire et la bande

passante que le routeur physique cible a de disponibles pour héberger ce routeur virtuel. Le défi est de garder cette information à jour sur chaque nœud malgré les changements dans la disponibilité des ressources.

1.1 La virtualisation dans les réseaux

Il a été démontré à plusieurs reprises qu'un ordinateur n'utilise en moyenne que 10% [11] de ses ressources matérielles. La virtualisation permet d'augmenter le ratio d'utilisation des ressources physiques et donc d'obtenir une meilleure utilisation du matériel et de l'infrastructure. Mais elle permet aussi toute une nouvelle gamme d'opérations qui sont reliées au fait que l'exécution d'un système d'exploitation n'est plus tributaire du matériel, par exemple la migration à chaud de machines virtuelles.

De plus, il existe quelques dizaines d'hyperviseurs (voir lexique), certains même gratuits et ouverts, qui permettent de virtualiser des plateformes entières. Quoique la plateforme la plus commune à virtualiser soit le processeur x86 d'Intel (voir lexique), certains hyperviseurs comme Xen [12] peuvent virtualiser d'autres plateformes comme la plateforme Itanium, PowerPC ou ARM. Cette démocratisation de la virtualisation offre un écosystème technologique propice à l'expérimentation de nouveaux concepts. C'est dans ce contexte que le projet GENI [1] propose l'utilisation de la virtualisation pour l'expérimentation de nouveaux protocoles et de nouveaux designs qui pourraient composer des réseaux futurs.

La virtualisation sur des plateformes peu dispendieuses comme celles supportant le processeur Intel x86 a apporté un nouveau souffle au développement des réseaux et de nouveaux systèmes en général. La plateforme étant devenue accessible, plusieurs groupes de travail ont proposé des modèles de routeurs virtuels basés sur cette architecture matérielle [3] [13] [6].

L'instanciation des machines virtuelles facilite la création de nouveaux services et serveurs permettant de répondre à des besoins du design de l'Internet [1]. Cette

facilité a entre autres permis la mise sur pied d'environnements de développement importants comme PlanetLab [2].

1.2 Objectifs

Il y a un besoin évident d'un mécanisme pour effectuer la convergence de l'information d'une topologie de ressources disponibles d'un réseau d'équipements virtualisés. Le mécanisme de convergence doit être distribué et décentralisé et s'appliquer à des réseaux à grande échelle. L'objectif de ce mémoire est de proposer une réponse quant à la question du temps de convergence de la topologie des ressources partagées dans un réseau d'équipements virtuels.

1.3 Contribution du mémoire

L'approche proposée dans ce mémoire est basée sur le modèle de transmission épidémique Gossip Infect And Die (IAND) auquel nous avons apporté une modification dans la sélection des nœuds de propagation.

Le présent travail introduit les modifications qui ont été apportées à IAND pour former Low latency Infect and Die (LIAND). Cette approche permet de mettre à jour les données représentant les ressources partagées.

L'algorithme a été simulé de façon à représenter un protocole non structuré et décentralisé. La simulation a aussi démontré la mise à jour de la topologie en quelques secondes. Et cela indépendamment de la taille du réseau, de son maillage (voir lexique) et jusqu'à une certaine limite de ses latences.

Le travail présenté ici succède à la publication d'un chapitre dans un traité scientifique publié chez *Hermes* [9], un article publié à la conférence GRES 2010 [8], ainsi qu'une présentation à la même conférence. Les concepts du travail ont été débattus et critiqués et relèvent une certaine complexité par le fait qu'ils intègrent une multitude de technologies. Afin de comprendre la mise en contexte de ce travail, les

concepts généraux sur lesquels est basée l'intégration sont présentés dans cette section.

1.4 De la virtualisation des serveurs à la virtualisation des routeurs

La virtualisation est un des concepts centraux autour duquel les systèmes contemporains sont construits. Cette technologie offre plusieurs avantages, dont la flexibilité, par rapport aux implantations physiques. Comme mentionné précédemment, la virtualisation est la pierre angulaire de ce travail. L'ensemble des concepts, comme le partage des ressources, le déplacement à chaud de machines virtuelles et d'autres sujets qui seront présentés, sera les raisons de l'établissement d'une cartographie des ressources partagées.

1.4.1 Introduction à la virtualisation

La virtualisation est un mécanisme permettant l'isolation des ressources informatiques de façon à pouvoir exécuter plusieurs instances de systèmes d'exploitation [5] sur un même ordinateur physique.

Afin de réaliser la virtualisation, nous avons recours à un hyperviseur. Cette composante (logicielle ou matérielle) est en fait une couche de gestion entre les environnements logiciels et matériels. Cette couche de gestion permet le partitionnement du matériel du système et l'isolation des ressources entre les systèmes d'exploitation. En fait, en mode virtuel, un système d'exploitation devient un processus pour l'hyperviseur. Tout comme dans la plupart des systèmes d'exploitation modernes, chaque processus est en mesure d'agir comme s'il était la seule composante qui utilise le système à 100 % [5].

La virtualisation est un moyen efficace d'abstraire le matériel et donc de donner une grande flexibilité d'exécution aux machines virtuelles. L'abstraction du matériel est un mécanisme permettant l'instanciation de plusieurs machines virtuelles sur un seul nœud physique. Et comme une machine virtuelle est traitée comme des processus

de l'hyperviseur, l'exécution de cette dernière peut être déplacée sur d'autres machines physiques compatibles en copiant le contenu de son espace mémoire et l'état des CPU de cette machine virtuelle. Par ailleurs, il faut noter que dans un routeur, il n'y a pas de stockage à déplacer.

Finalement, les ressources CPU, mémoire et entrées/sorties peuvent être répertoriées et utilisées par d'autres nœuds du réseau. Ce partage des ressources est un élément important pour des projets d'envergure comme la grille informatique [14] ou SETI@Home [15] [16]. Ces projets mettent en commun les capacités de traitement de plusieurs ordinateurs afin de réaliser plusieurs calculs complexes en parallèle. La mise en commun des ressources dans le réseau est un concept qui attire l'attention de plusieurs groupes de travail par exemple Planetlab [2] et VROOM [7] [17]. Ces projets visent à partager le réseau avec d'autres nœuds de ce même réseau par différents mécanismes liés à la virtualisation. Tous ces projets ont en commun la capacité d'utiliser des ressources (matérielles ou logicielles) qui sont mises à la disposition d'une communauté ou d'un but commun.

1.4.2 La virtualisation dans le contexte du réseau

Nos besoins continuellement grandissants de l'Internet (téléphonie, vidéo sur demande, etc.) nous obligent à inventer et à faire évoluer les protocoles de communication, de tester de nouvelles topologies, de tester de nouvelles stratégies de résolution de problème, etc. Malheureusement, l'implantation de nouveaux protocoles passe par les systèmes d'exploitation spécialisés pour les équipements de réseau (IOS) et donc par les constructeurs d'équipements de réseaux puisque ces IOS (voir lexique) sont majoritairement propriétaires. Cela rend donc l'évolution du réseau difficile et réservée à une certaine élite de l'industrie.

La virtualisation permet d'assouplir ce modèle en permettant d'exécuter plusieurs systèmes d'exploitation simultanément sur un même équipement. On pourrait donc théoriquement voir cohabiter une version de Cisco IOS [18], de JunOS [17] et de

quelques instances Linux qui exécutent du code expérimental sur un même routeur. Ce rationnel nous servira d'argument de base pour utiliser la virtualisation au niveau du réseau et nous permettra d'explorer plusieurs facettes de la virtualisation appliquées au réseau.

A priori, un routeur virtuel est simplement une machine virtuelle qui fait tourner un système d'exploitation qui exécute un programme spécialisé de routage.

Lorsque la machine virtuelle est mise sous tension, elle charge un système d'exploitation. Ce dernier initie l'ensemble de son domaine d'exécution comme s'il était le seul à s'exécuter sur le nœud physique. C'est-à-dire qu'il configure sa plage d'adressage mémoire, ses partitions sur ses disques (virtuels ou non), il charge ses configurations réseau et configure ses interfaces de communication. Les interfaces de réseaux d'une machine virtuelle sont appelées des VIF pour Virtual Interfaces. Chaque VIF est associée à une interface physique, mais possède sa propre configuration : adresse IP, adresse MAC (voir lexique), etc. Le programme de routage utilisera ces interfaces virtuelles pour y associer les mécanismes de routage et les protocoles de routage.

Les composantes de ces routeurs virtuels sont isolées les unes des autres. Les machines virtuelles ne voient pas les ressources et les composantes des autres machines virtuelles sur un même équipement physique. Cette mécanique permet par le fait même d'isoler les différentes configurations des réseaux s'exécutant dans ces routeurs virtuels. Et cela permet d'exécuter plusieurs routeurs virtuels sur un même équipement physique sans que les réseaux ne s'entremêlent.

Un des bénéfices importants de la virtualisation est l'abstraction de l'architecture matérielle. En effet, le système d'exploitation s'exécutant dans une machine virtuelle n'accède pas véritablement aux composantes physiques, mais à une représentation de ces composantes qui est offerte par l'hyperviseur. Ce qui permet, entre autres, le déplacement des machines virtuelles pendant leur exécution. Le déplacement de ces

machines virtuelles oblige à connaitre l'état des ressources du nœud physique de destination avant le déplacement de l'exécution de la machine virtuelle source. Pour ce faire, les nœuds physiques doivent donc connaitre leurs spécifications d'exécution comme leurs bandes passantes, leurs liens physiques, leurs cartes d'interface réseau, les cycles CPU non utilisés, l'espace mémoire disponible et l'espace de stockage. Ces éléments sont nécessaires pour trouver un nœud physique compatible et ayant les ressources nécessaires pour assurer l'exécution de la future machine virtuelle.

Idéalement, les routeurs virtuels devraient être en mesure de s'exécuter sur tous les routeurs physiques. Par contre, certaines contraintes physiques influencent les contraintes virtuelles.

- La latence : La nouvelle localisation pour le routeur virtuel ne devrait pas changer de façon importante les délais des liens virtuels, afin d'éviter des dégradations importantes dans les applications. Par exemple, le déplacement d'un routeur virtuel à partir Montréal à Sherbrooke pourrait rajouter quelques millisecondes de latence sur le lien, mais le déplacement d'un routeur virtuel sur des milliers de kilomètres pourrait mener à augmentation importante de la latence.
- La capacité des liens : L'instanciation d'un routeur virtuel ajoute de la charge de trafic à une nouvelle série de liens sous-jacents. Ces liens doivent avoir suffisamment de capacité inutilisée pour accueillir le trafic supplémentaire.
- Les incompatibilités de plateformes : Les routeurs des différents fournisseurs peuvent ne pas supporter les mêmes systèmes d'exploitation, les mêmes protocoles de routages ou les mêmes techniques de migration. Il est difficile de déplacer un routeur virtuel d'une plateforme du fournisseur à l'autre. En tant que tel, un routeur virtuel peut être limité à la migration vers un autre routeur physique construit par le même fournisseur.
- Les capacités des routeurs : Même lorsque tous les routeurs viennent du même fournisseur, différents routeurs physiques pourraient avoir des

capacités différentes. En tant que tel, un routeur virtuel ne pouvait se déplacer vers des routeurs physiques qui supportent les fonctionnalités requises.

Il y a plusieurs facteurs pouvant faire globalement varier la disponibilité des ressources. Les prochaines sections introduisent ces facteurs sous la forme d'opérations dans les réseaux virtuels.

1.5 Les nouvelles opérations dans les réseaux virtuels

Au-delà des opérations de gestion standard comme la configuration des routeurs physiques, la virtualisation dans le réseau induit un certain nombre d'opérations qui sont présentées dans cette section.

Les opérations simples décrivent le démarrage, l'arrêt d'une machine virtuelle, les changements dans l'attribution des ressources et l'ajout ou le retrait de nœuds physiques. Ce ne sont pas des opérations liées au réseau comme tel, mais plutôt à l'administration des systèmes. Par contre, ce sont de nouvelles opérations qui sont introduites dans le réseau et celles-ci sont importantes puisque certaines d'entre elles influent sur la disponibilité et l'état des ressources. Cela déstabilise la topologie des ressources disponibles et la forcera à converger.

Les opérations complexes comme le déplacement à chaud de machines virtuelles ou le positionnement de machines virtuelles feront comprendre au lecteur toute la finesse de la mécanique requise pour faire fonctionner un réseau virtuel.

1.5.1 Opérations

La création de machines virtuelles consiste essentiellement à assigner ou à réserver des ressources matérielles (CPU, mémoire, espace disque, configuration des interfaces de réseau) qui seront utilisées pour son exécution.

Cette opération n'a pas d'impact sur la cartographie des ressources parce qu'elle ne fait que définir la configuration d'une machine virtuelle. Les ressources ne sont réellement utilisées que lors de l'instanciation de la machine virtuelle.

L'instanciation d'une machine virtuelle peut être comparée au démarrage d'un ordinateur physique. C'est-à-dire qu'après la mise sous tension, le BIOS doit charger et exécuter un Power-On Self Test (POST) [19]. Par la suite, la machine virtuelle charge le fichier de configuration qui décrit le matériel de l'environnement dans lequel elle s'exécutera. Ensuite, l'hyperviseur doit créer les composantes virtuelles comme les VIF et associer le pont de communication (bridge) pour le nouveau routeur virtuel de sorte qu'il soit possible de communiquer à partir de la machine virtuelle vers le réseau physique réel. Le fichier contient aussi l'emplacement du disque binaire (disque, NVRAM, EEPROM, etc.) qui contient le système d'exploitation (le disque de démarrage), la quantité de mémoire que le système d'exploitation est autorisé à prendre et la configuration du réseau. La prochaine étape consiste à amorcer la lecture des premiers secteurs du disque binaire. Cela a pour effet d'amorcer le système d'exploitation qui peut finaliser la séquence de démarrage. Pour finir, le système d'exploitation invité se déclenche et charge les pilotes des périphériques virtuels.

La finalisation du processus se réalise lorsque le système d'exploitation charge le programme de routage. Ce dernier configure le plan de contrôle du réseau afin qu'il puisse traiter les paquets du réseau auxquels il est associé. C'est-à-dire qu'il effectue le chargement des tables de routage, le démarrage des services et des protocoles qui sont associés aux interfaces, l'application des listes d'accès, etc.

L'impact de cette opération sur les ressources globales est la réduction des ressources disponibles du nœud physique. Ces ressources doivent être soustraites du groupe de ressources globales disponibles. Cette opération force une mise à jour de la topologie.

Dans le sens inverse, l'arrêt d'un routeur virtuel libère les ressources que sa machine virtuelle utilisait. L'hyperviseur libère l'espace mémoire, l'image des registres de CPU ainsi que les assignations de matériel. La machine virtuelle est alors détruite et sortie de la liste d'exécution. Au niveau des ressources, cette opération a pour effet de libérer des ressources sur le nœud physique et donc d'ajouter des ressources disponibles à l'ensemble des ressources partagées.

À mi-chemin entre le démarrage et l'arrêt de machines virtuelles se trouve le changement d'affectation de ressources, lequel est un élément nécessaire dans la gestion des machines virtuelles. Il existe plusieurs cas où, pour des raisons de performance ou par souci de récupération d'énergie [20] ou de ressources, il faut changer l'affectation des ressources. Changer l'attribution des ressources de traitement a de l'impact, positivement (ajout de ressource) ou négativement (retrait de ressource), sur l'ensemble des ressources disponibles. Par exemple, une augmentation de l'attribution de CPU à une machine virtuelle entraine une diminution des ressources CPU disponibles d'un nœud physique. À l'inverse, une diminution dans l'attribution des ressources a pour effet d'augmenter les ressources disponibles sur un nœud physique. Lors des changements dans l'affectation des ressources, il faut mettre à jour la topologie des ressources disponibles puisque la quantité de ressources disponibles a changé.

Finalement, un facteur influant de façon importante sur la quantité de ressources disponibles est le retrait ou l'ajout de nœuds physiques dans le système. En effet, lorsqu'un nœud physique s'ajoute ou devient indisponible, c'est un ensemble de ressources qui s'ajoute ou disparait. C'est aussi un nouveau nœud à considérer lors des mises à jour topologiques. Ces opérations sont spécifiques à la virtualisation et sont des évènements qui sont critiques dans le cadre du réseau. La fiabilité du réseau virtuel est en fonction du temps de propagation de ces fluctuations dans les ressources. Les décisions topologies seront réalisées en fonction de l'état de ces ressources.

1.5.2 Opérations complexes

Une des premières opérations complexes est le déplacement à chaud de machines virtuelles. Le déplacement à chaud de machines virtuelles consiste à déplacer une machine virtuelle d'un nœud physique à un autre, pendant que la machine virtuelle est en opération. Cela permet d'effectuer des changements topologiques sans impacter les services offerts par les machines virtuelles. Dans le cadre de ce travail, ce sont des routeurs virtuels qui sont utilisés. Il existe plusieurs cas où il peut être nécessaire de déplacer un routeur virtuel en cours d'exécution d'un nœud physique vers un autre nœud physique. Lors d'une maintenance du routeur physique, on peut déplacer les routeurs virtuels afin de ne pas impacter la topologie des réseaux qui s'exécutent sur ce routeur physique. Plusieurs cas existent où les protocoles de routage (par exemple BGP) peuvent être longs à resynchroniser. Une mise à jour d'un routeur requérant son redémarrage et le temps que le protocole de routage (par exemple BGP) se synchronise avec ses voisins peut occasionner des « pannes » allant de 10 à 15 minutes [7] [21]. Dans le cas d'une évolution de système comme par souci de sauvegarde énergétique [20], il est peut-être préférable de considérer le déplacement de l'exécution d'un routeur virtuel afin d'éviter les changements topologiques du réseau. L'entretien des routeurs physiques nécessite encore des modifications à la topologie du réseau logique, et exige des reconfigurations des protocoles de routage, ce qui implique généralement des erreurs de configuration et donc une augmentation de l'instabilité du réseau. Avec des mécanismes de déplacement de routeurs virtuels les administrateurs réseau peuvent tout simplement migrer tous les routeurs virtuels s'exécutant sur un routeur physique vers d'autres routeurs physiques avant de faire la maintenance et les faire migrer de retour, sans jamais avoir à reconfigurer les protocoles de routage ou de s'inquiéter de la perturbation du trafic ou de la convergence du protocole.

Ce mécanisme implique que la consommation de ressources passe de la machine source à la machine cible. Ceci implique deux mises à jour de l'état des ressources;

une première pour indiquer que les ressources de la machine source sont libérées et une seconde pour indiquer que les ressources de la machine cible sont maintenant attribuées à une nouvelle machine virtuelle.

A priori, lorsqu'il faut déplacer ou créer une nouvelle machine virtuelle, il faut trouver une machine physique permettant l'exécution de la machine virtuelle. Cette opération s'appelle le positionnement.

Cette opération n'a aucun impact sur les ressources puisqu'à cette étape, nous ne faisons que localiser une configuration potentielle pour une machine virtuelle. Par contre, pour trouver le bon candidat, il faut avoir toute l'information nécessaire pour prendre une décision judicieuse.

Que ce soit dans le déplacement ou la création de machines virtuelles, le choix de l'ordinateur de destination doit se faire en quelques secondes. Il faut donc que la topologie des ressources disponibles soit la plus précise possible en tout temps. Il faut comprendre que le déplacement d'un routeur implique aussi le déplacement topologique du réseau que ce routeur dessert. En effet, il faut déplacer les liens ainsi que leurs états et les flux de données qui sont traités par le routeur. Si la prise de décision est trop lente ou imprécise, il pourrait potentiellement en résulter une interruption momentanée de service, ce qui forcera le réseau à converger, ou encore il pourrait en résulter une panne.

Dans le cas contraire, les opérations complexes ne pourraient se réaliser avec la précision qui leur est nécessaire. C'est-à-dire que si l'on décide d'ajouter de la mémoire pour l'exécution d'une machine virtuelle, il faut que cette mémoire soit disponible au moment de l'utiliser. De la même façon, s'il est décidé de déplacer l'exécution d'un routeur virtuel sur un autre routeur physique, il faut que ce dernier soit disponible et avec les ressources nécessaires. Il faut que l'information traitant de l'état des ressources du réseau reflète la réalité, et ce, en tout temps. Dans le cas contraire, la qualité des décisions topologiques ne peut être garantie.

1.6 La virtualisation des équipements de réseau et son impact sur l'organisation des ressources

La virtualisation des équipements de réseau modifie la mécanique d'élaboration d'une topologie réseau. Il faut en effet élargir la portée de la topologie au contexte virtuel. La virtualisation introduit une nouvelle couche de gestion qui doit être considérée dans la topologique générale d'un réseau composé de tels équipements. La Figure 1.1 présente le découpage en couche d'un réseau composé d'équipements virtualisés. On y retrouve la couche physique où s'exécutent les équipements physiques, la couche logique où s'exécutent les équipements virtuels et finalement la couche de gestion qui gère l'état des ressources, les équipements et d'où s'exécutent les opérations comme le déplacement à chaud de machines virtuelles.

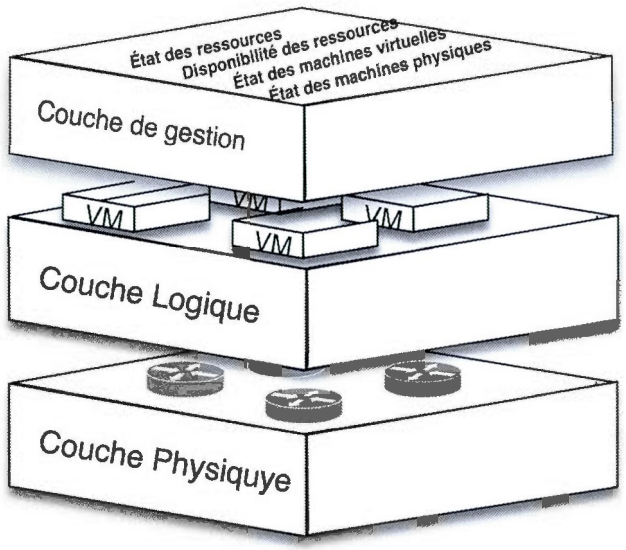

Figure 1.1 Architecture d'un réseau composé de routeurs virtuels

Lorsque l'état des ressources de la couche physique change, la couche de gestion doit être mise à jour. L'état des ressources de la couche physique peut changer à tout moment. Par exemple lors d'attribution ou de libération de ressources aux machines virtuelles.

De plus, nous savons que les équipements physiques ne consomment généralement pas toutes leurs ressources. Ils peuvent alors partager ces ressources afin de créer de nouvelles machines virtuelles ou encore héberger des routeurs lors des opérations de déplacement à chaud, etc.

C'est dans cette couche de gestion que la cartographie des ressources et l'organisation de ces ressources en topologie sont réalisées.

La Figure 1.2 présente un modèle architectural simplifié d'un routeur physique exécutant des routeurs virtuels. On retrouve dans cette figure l'intégration de ces différentes couches et à quel niveau elles se situent.

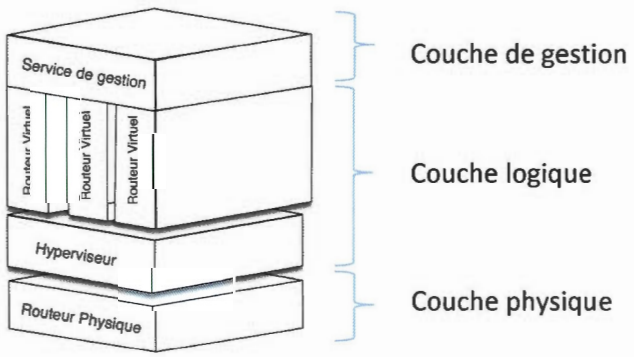

Figure 1.2 Architecture d'un routeur virtuel

On peut voir qu'on retrouve une partie du plan de gestion dans chacun des routeurs physiques. Dans notre modèle de routeur, la cartographie des ressources disponibles sur les autres routeurs physiques du réseau est gérée localement par le routeur physique lui-même.

La mise en réseau des services de gestion d'un tel routeur crée un réseau de service indépendant du réseau et des données traitées par les routeurs virtuels. Lorsque ces services sont mis en réseau, il est alors possible de découvrir quelles sont les ressources disponibles sur les différents routeurs physiques faisant partie de ce réseau. La Figure 1.3 montre la mise en réseau de cette couche de gestion.

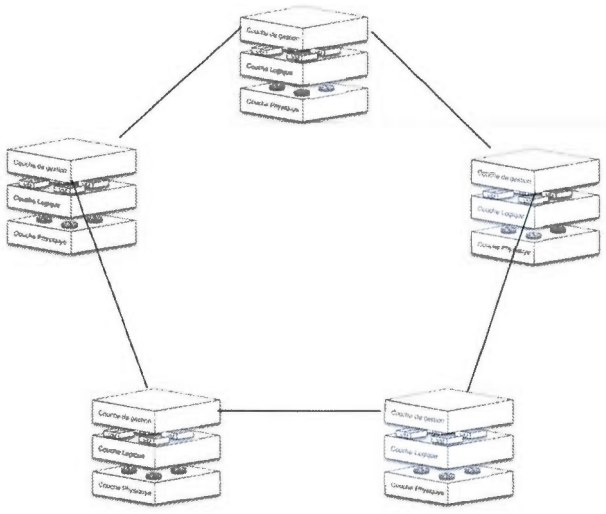

Figure 1.3 Découverte des ressources dans un réseau d'équipements virtualisés

A priori, il pourrait sembler naturel de centraliser l'information contenue dans ce plan de gestion. Ce faisant, il faudrait alors gérer une structure de données commune à tous les nœuds. Si on prend par exemple le système VCenter 4.0 de VMWare, qui joue le rôle de la couche de gestion pour les serveurs, on se rend vite compte de la limite de la centralisation. Par exemple, selon le document de spécification du produit [22], il ne peut y avoir plus de 96 opérations simultanées par instance de VCenter et il ne peut y avoir plus de 2000 machines virtuelles gérées par ce même VCenter pour un maximum de 300 hôtes physiques. De façon centralisée, la demande en traitement est telle que les limites de puissance sont rapidement rencontrées. Ces métriques ont été

prises en exemple parce qu'elles ont été démontrées et mesurées de façon commerciale et sont généralement entérinées par l'industrie et que ce sont des chiffres véritables sous de vraies charges de travail.

Pour ce travail, une approche décentralisée a été considérée pour gérer ces ressources et exécuter les différentes opérations.

1.7 Le partage des ressources et la topologie des ressources disponibles

La mise en commun des ressources n'est pas un concept réservé à la virtualisation des réseaux. La grille informatique [14] l'a largement exploité et des projets comme SETI@Home [23] mettent aussi le concept à profit. Le principe est relativement simple : il s'agit de rendre disponibles les ressources de traitement lorsque celles-ci ne sont pas utilisées. Dans un projet comme celui de SETI@Home, lorsque l'ordinateur entre en mode de veille, ce dernier contacte les serveurs de SETI@Home pour recevoir une parcelle d'un calcul complexe, par l'entremise d'un écran de veille. L'ensemble des ordinateurs participants (environ 2,5 millions [23]) mettent alors en commun leurs ressources pour résoudre des problèmes mathématiques complexes liés à l'astronomie.

Similairement, dans les réseaux d'équipements virtuels les ressources matérielles non utilisées des nœuds physiques seront mises à contribution pour former un pool de ressources de traitement utilisables pour tous les autres nœuds physiques du réseau. Cette mécanique est permise grâce à la virtualisation des équipements de réseau qui permet le découpage de l'utilisation des composantes matérielles.

Pour connaitre l'intégralité des configurations physiques potentielles, il faut rassembler l'ensemble de l'information des ressources disponibles et s'en faire une cartographie. La cartographie des ressources permet de classer, d'analyser et d'interroger l'état des ressources selon des caractéristiques qui sont définies par le contexte prévalant au moment d'exécuter les opérations. Cette cartographie des

ressources disponibles servira à identifier des configurations potentielles de machines virtuelles et permettra de positionner rapidement un routeur virtuel. Le contenu de la cartographie est l'association entre les ressources disponibles et les nœuds physiques.

Afin de réaliser la cartographie des ressources d'un réseau tel qu'Internet, il faut tout d'abord prévoir de la décentraliser. Il existe plusieurs contraintes dans un système global : il y a une multitude d'opérateurs et la cartographie doit être indépendante des multiples architectures matérielles ou logicielles ainsi que des différentes topologies ou organisations des réseaux. Ces contraintes forcent alors à opter pour une déstructuration du modèle et donc de forcer la distribution d'une carte à chaque nœud. C'est un modèle qui a fait ses preuves avec les protocoles de routage où chaque décision prise sur un paquet est locale à chaque routeur. De la même façon, la découverte des ressources et les mises à jour de l'état des ressources sont décentralisées et non structurées.

Pour relever les défis qu'impliquent la décentralisation et la déstructuration, deux types de protocoles seront étudiés et présentés dans ce mémoire : les protocoles P2P [24] et les protocoles Gossip [25].

Les réseaux de type P2P sont définis par deux types de topologie : les topologies faiblement maillées et les topologies fortement maillées. Le maillage de la topologie est un élément important pour ce travail parce que le maillage implique des possibilités plus ou moins grandes de chemins possibles pour effectuer les mises à jour.

Les topologies de réseaux [26] comme l'anneau (*ring*), maillés (*mesh*), en étoile (*star*), en arbre (*tree*), en ligne (*line*) ou finalement en bus sont des topologies qui sont considérées faiblement maillées puisque chaque nœud n'a pas plus de 2 voisins.

Les topologies fortement maillées sont des topologies où chaque nœud possède plus de deux voisins. La topologie de réseau maillé qualifie les réseaux dont tous les

hôtes sont connectés en P2P sans hiérarchie centrale, formant ainsi une structure en forme de filet. Cette topologie permet d'éviter d'avoir des points de défaillance unique, qui en cas de panne, coupent la connexion d'une partie entière du réseau.

L'établissement de la limite du maillage du réseau de la simulation est basé sur une moyenne du nombre de ports que possède un routeur périphérique. Ces routeurs de moyenne puissance [53, 54] son composé d'en moyenne quatre (4) ports Ethernet. C'est en effet ce que le réseau NSF [64] nous démontre en supposant que 13,000 routeurs supportent environ 30,000 arêtes du graphe de leur réseau. On compte en moyenne 2.25 voisins par routeurs. Puisqu'un routeur physique moyen possède quatre (4) ports, on peut donc dire qu'ils peuvent avoir entre un (1) et quatre (4) voisins. Le maillage de la simulation sera établi en fonction du nombre de voisins moyen d'un routeur, c'est-à-dire entre 2 et 4. Pour les besoins de la simulation, la limite a été poussée jusqu'à 5 afin de voir si l'algorithme s'améliorait au-dessus d'un maillage de 4.

Le maillage du réseau est un élément de la distribution des réseaux qui permet d'éviter les points de défaillance unique. Les changements qui seront apportés à IAND tenterons de fait ressortir s'il y a un gain à profiter du maillage du réseau pour diffuser de l'information.

1.8 Les stratégies de gestion d'environnements décentralisés et distribués

Cette section présente une brève introduction aux deux types de protocoles étudiés. Les deux types de protocoles sont le P2P [24] et le Gossip [25].

1.8.1 Introduction au modèle P2P

A priori, ce qui définit les réseaux P2P purs est la particularité de ne fournir aucun algorithme pour l'organisation ou l'optimisation des connexions réseau. C'est-à-dire que chaque nœud qui se joint au réseau P2P participe de façon égalitaire avec tous les autres nœuds du système. Il n'y a pas de hiérarchie ou de nœuds centraux.

Mais l'application du modèle P2P a forcé quelques dérogations à ce modèle de base. Par exemple, certains systèmes utilisent des nœuds centraux pour gérer les index ou pour optimiser les recherches. Ces systèmes sont appelés structurés puisqu'ils dépendent d'une structure centrale pour fonctionner. On peut se rappeler le système Napster [27].

De la même façon, d'autres systèmes définissent une structure de données sous-jacente au réseau P2P pour fonctionner. Cette structure de données est utilisée pour décrire les ressources partagées dans le réseau et faciliter leur localisation. Cette structure de données est dynamique et décentralisée puisqu'elle ne dépend pas du réseau P2P. Ce modèle de protocole P2P est celui utilisé par BitTorrent [28] par exemple, lequel utilise une table de hachage distribuée (DHT voir lexique). Dans ce cas, la DHT est la structure de données sous-jacente au réseau. C'est-à-dire que la DHT contient les informations structurelles associées au système qu'elle dessert. Cette structure de données est généralement globale et a pour rôle de décrire les ressources disponibles par le système et donc de facilité leurs localisations.

Le père des systèmes P2P de partage de fichiers est Gnutella [29]. Ce système fonctionne dans un environnement distribué. Un utilisateur a deux rôles : (1) il est client et (2) en même temps il est serveur. Ce qui permet aux utilisateurs possédant des fichiers de les partager avec d'autres utilisateurs tout en téléchargeant des fichiers d'autres utilisateurs. Les services de serveur et les fonctionnalités clients sont toujours actifs en même temps dans un nœud Gnutella.

Dans les réseaux Gnutella, une des caractéristiques d'un nœud est la bande passante offerte pour le téléchargement de fichiers. Il a été remarqué que l'attribution de cette qualité aux nœuds peut influencer la topologie. En effet, plus un nœud possède une grande bande passante, plus il est préféré à d'autres, ce qui forme des regroupements sous la forme d'un anneau central de raccordement au réseau

Gnutella. Ce regroupement de nœuds par caractéristique sera réutilisé par d'autres groupes de travail [28] [29] afin de créer des regroupements par intérêt.

Dans le réseau Gnutella, lorsqu'il faut trouver des ressources, il faut inonder le réseau de la requête afin que tous les nœuds qui correspondent à la recherche répondent. La propagation de ces messages fonctionne comme le modèle connu sous le nom de « propagation virale ». Un client envoie un message à un nœud, et celui-ci doit l'envoyer à nouveau à tous les nœuds auxquels il est connecté et ainsi de suite.

Une étude [30] a démontré que l'inondation est une technique qui utilise beaucoup de bande passante et qu'elle comporte des limites importantes pour une mise à l'échelle.

Un des éléments importants de l'étude est que la topologie du réseau Gnutella ne concorde pas avec la topologie d'Internet, réseau sur lequel il s'exécute. Ce qui fait que le réseau Gnutella utilise inefficacement les ressources du réseau sur lequel il s'exécute. Les auteurs croient que ces deux caractéristiques sont aussi applicables à un vaste modèle de protocoles P2P puisque ce sont des caractéristiques qui définissent le P2P.

Voici en quelques statistiques un survol des propriétés du réseau Gnutella :

1. 40 % des nœuds se débranchent dans une période de 4 heures ou moins
2. 25 % des nœuds sont disponibles 24 heures ou plus
3. 36 % du trafic seulement est utilisé pour des requêtes
4. 55 % du trafic est utilisé pour des messages topologiques
5. Une estimation de 330TB/mois est proposée pour seulement gérer et maintenir la topologie. Ceci représente environ 1,7 % de la dorsale de l'Internet aux États-Unis.

Ce qu'il faut retenir de ces statistiques, c'est que maintenir une topologie à l'échelle de l'Internet coûte cher en termes de bande passante et que cette topologie change régulièrement.

Comme il a été démontré dans l'étude, 36 % de ce trafic est utilisé pour l'acheminement des requêtes de recherche de fichiers. Ces requêtes sont les requêtes qui génèrent le plus de stress sur les nœuds puisqu'elles demandent d'effectuer du traitement.

Une des techniques populaires utilisées pour pallier à l'inondation de Gnutella est le recours à la table de hachage distribuée (DHT) [31]. Cette technique est utile pour les applications à basse latence qui demandent l'accès à de grands volumes de données. En effet, la répartition géographique des nœuds signifie que les latences entre les nœuds sont susceptibles d'être élevées. Pour fournir une réponse à une opération de lecture (get) à basse latence, il faut que le système puisse trouver une copie des données à proximité sans avoir à traverser des liens à haute latence. Les liaisons d'un réseau étendu sont susceptibles d'être moins fiables et ont des capacités inférieures que les liens du réseau local. Donc, afin de fournir durabilité et efficacité, le système doit minimiser le nombre de copies d'éléments de données qu'il envoie sur ces liens de capacité limitée. Pour ce faire, chaque nœud participant à la DHT possède une partie de la table de hachage (voir lexique). Plusieurs nœuds peuvent posséder la même information pour des raisons de distribution ou de robustesse. À cette information est associée une clé unique obtenue par hachage, par exemple avec SHA-1. Une table de hachage permet de retourner une valeur associée à une clé en O(1) [32]. C'est donc un modèle très performant par rapport à Gnutella où il faut inonder le réseau.

La Figure 1.4 [33] démontre le principe d'une DHT. On retrouve les données à rechercher, les clés et un réseau de nœuds formant la table de hachage distribuée. Le principe qui y est démontré est que, par exemple pour trouver le mot Whiskey, il faut

le hacher. Par la suite, il suffit de localiser les nœuds qui possèdent l'index pour le résultat du hachage de Whiskey. On contacte ensuite directement les nœuds contenus dans l'index afin d'accéder au contenu de Whiskey. On peut aussi utiliser des variantes comme le démontre la Figure 1.4. C'est un modèle de stockage de données assez populaire sur Internet pour des bases de données très volumineuses. Des logiciels comme Cassandra [34] sont utilisés dans de grands systèmes comme Facebook [35].

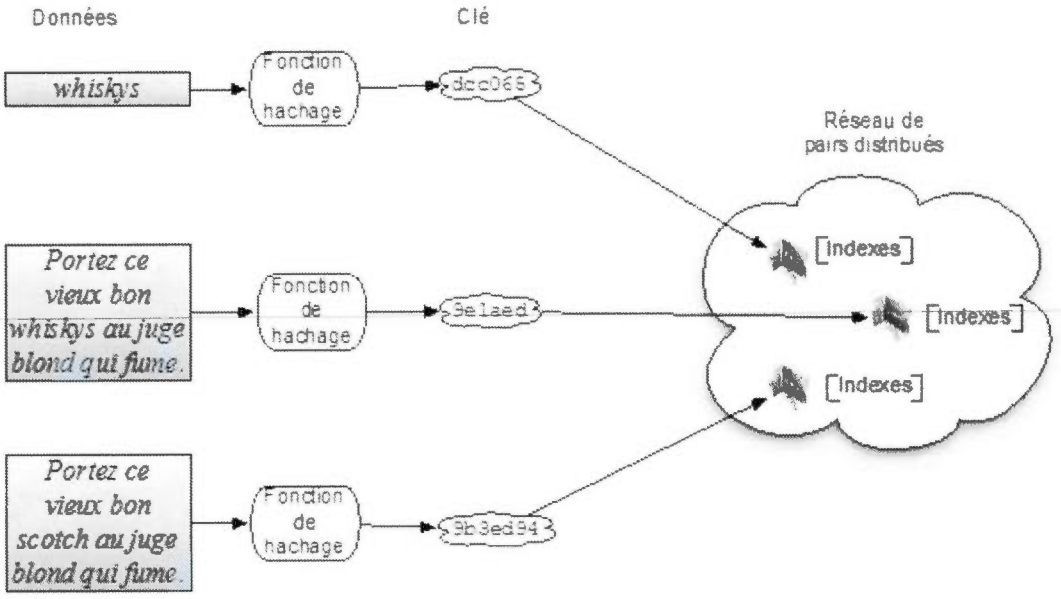

Figure 1.4 Table de hachage distribuée

Cette mécanique élimine le concept d'inondation lors des requêtes dans les systèmes distribués. Au lieu de soumettre une requête à tous les nœuds d'un système, il suffit de soumettre une clé de recherche et spontanément les nœuds possédant l'information sont localisés sur le réseau. Cette localisation spontanée de l'information est possible parce que le résultat du hachage est unique pour chaque chaîne de caractères et que chaque nœud possède un index local des nœuds

desservant l'information pour ladite clé. Il suffit ensuite d'effectuer le get nécessaire pour obtenir l'information. Une des applications de ce mécanisme est le protocole BitTorrent.

Un des principaux défauts de la table de hachage distribuée est justement le hachage. En fait, c'est une opération coûteuse en traitement qui n'est pas nécessairement fiable. Par exemple, la chaîne « abcdef » chiffrée avec SHA-1 donne le hachage suivant « 1f8ac10f23c5b5bc1167bda84b833e5c057a77d2 ». Mais s'il faut chercher toutes les ressources débutant avec « abc », la chaîne « abc » donne « a9993e364706816aba3e25717850c26c9cd0d89d ». Il n'y a pas de concordance d'une chaîne à l'autre. Ce qui rend difficile la recherche par critère.

Une autre approche que la table de hachage distribuée propose d'organiser est les nœuds par intérêt [36]. En effet, un des défis fondamentaux est de trouver quelle est la stratégie appropriée pour la localisation de contenu. Le contenu peut être reproduit sans cesse à de nombreux endroits dans le système P2P. Mais si le contenu ne peut être localisé de manière efficace, il y a peu d'espoir pour l'utilisation de la technologie P2P à des fins de distributions de contenu.

Une des prémisses du regroupement par intérêt est que si un pair a un élément particulier de contenu pour lequel on est intéressé, il est probable qu'il y aura d'autres pièces du contenu pour lequel on soit aussi intéressé. C'est ce modèle de localisation de ressources qui présente une organisation des pairs basés sur les intérêts. Les pairs qui partagent les mêmes intérêts créent des raccourcis entre eux. Les pairs utilisent ensuite ces raccourcis pour localiser le contenu. Lorsque l'un des raccourcis est en échec, le système se replie sur le protocole Gnutella afin de trouver un autre pair. Le raccourci fournit une structure souple au-dessus de Gnutella.

La Figure 1.5 présente un exemple de regroupement de nœuds dans Gnutella et un regroupement par intérêt avec raccourcis.

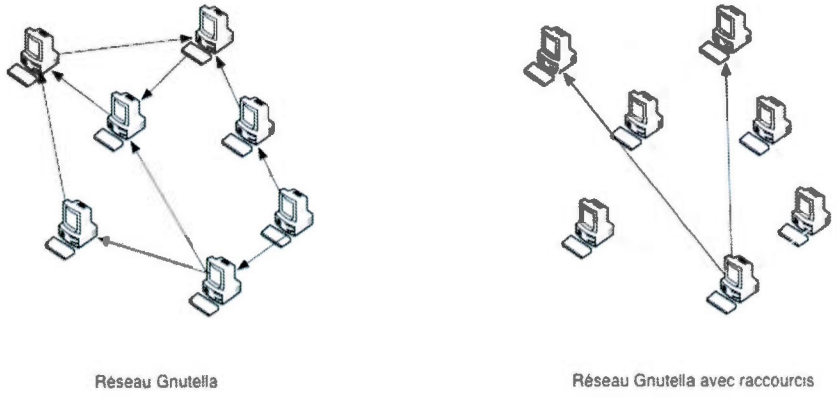

Figure 1.5 Création de raccourcis dans un réseau Gnutella

Le regroupement par intérêt permet de réduire de 75 % (4 pour 1) la quantité de requêtes nécessaires à la localisation de contenu dans le réseau Gnutella. C'est d'ailleurs ce qui a fait la popularité de KaZaa [37], un très populaire système de partage de fichiers.

Mais pour la distribution à grande échelle, c'est encore trop peu, car la création de raccourcis prend trop de temps. Microsoft et l'Université Rice du Texas ont développé un système regroupant la DHT et le principe de raccourci pour la localisation des ressources et former Pastry : « un système générique, décentralisé de contenu P2P et qui possède un système de routage pour les très grands réseaux, du recouvrement et de l'autoconfiguration des nœuds connectés via Internet » [38].

Pastry est un protocole P2P où les nœuds tendent à s'organiser selon ce qu'ils partagent, selon leurs affinités [36] et dont chaque nœud fait partie d'une table de hachage distribuée. La recherche des ressources se base sur cette table de hachage distribuée.

Le modèle de Pastry répond à tous les critères d'un système P2P robuste, décentralisé et passant les tests de mise à l'échelle. Il subsiste un seul problème : le besoin de maintenir une structure sous-jacente au réseau et au réseau P2P sur lequel Pastry s'exécute.

La création et le maintien de cette structure ne sont pas gratuits et posent le problème de la centralisation/décentralisation et du maintien de cette structure. En effet, la création et le maintien de la table de hachage distribuée sont un élément structurel. Quoique la table de hachage soit elle-même décentralisée, elle n'en est pas moins une structure sous-jacente au réseau P2P qu'il faut maintenir.

1.8.2 Introduction au modèle Gossip

Les systèmes Gossip [25] possèdent des propriétés techniques similaires aux systèmes P2P comme le fait qu'ils sont décentralisés, auto-organisés et auto-adaptatifs. Mais en plus, ils sont réellement et entièrement non structurés, c'est-à-dire qu'ils n'ont pas besoin de structure de données sous-jacentes pour fonctionner.

Le principe de diffusion de l'information s'appuie sur une technique qui reproduit la dispersion de messages lors de rumeur. Nous parlons alors de diffusion par rumeur (Gossip Dissemination). Le principe d'une rumeur est qu'elle est émise d'une source et qu'elle est répétée par d'autres instances que la source originale, et ce, jusqu'à ce que tous les éléments d'un ensemble connaissent la rumeur. C'est une différence importante par rapport au modèle du P2P en ce sens que l'information est émise vers les nœuds du réseau et non l'inverse. Ceci implique que les nœuds possèdent l'information localement et que seulement les changements sont émis vers les nœuds du réseau. Dans un modèle Gnutella, les clients parcourent les nœuds du réseau pour connaître leurs ressources partagées. Avec Gossip, les clients connaissent qui possède quoi. Il s'agit de garder l'information à jour entre les nœuds.

Pour s'assurer qu'un message émis d'un nœud source dans le réseau soit acheminé de façon rapide et sûre, un modèle de propagation épidémique a été proposé [25]. La propagation épidémique permet de transmettre un message dans un système entièrement non structuré de façon sûre.

Le principe général est que lorsqu'il y a une épidémie, une quantité minimale critique d'individus doit être contaminée afin qu'à un moment dans le temps, l'ensemble des individus le soit. Afin de cerner le parallèle avec le modèle épidémique, nous définissons que l'infection est le message à diffuser et que la contamination est l'effet d'avoir reçu le message. A priori, tous les nœuds d'un système (population finie) sont sains. C'est-à-dire qu'aucun des nœuds de l'ensemble initial n'est contaminé. Lorsqu'un message doit être transmis, le nœud source infectera ses voisins aléatoirement. La contamination totale d'un système se produit par phase. À chacune des phases, un nombre de nœuds tentera de contaminer un nouveau nœud sain. Après un certain nombre d'itérations, que nous appellerons round, l'ensemble du système est entièrement infecté.

Les protocoles Gossip utilisent le principe de *membership* afin de définir leurs réseaux de diffusion d'information. La découverte des membres et l'adhésion de ces membres au réseau Gossip est un problème encore ouvert et qui fait l'objet de plusieurs travaux [62] [63]. Brièvement, le problème se résume par comment les nœuds se découvrent les uns les autres et combien de nœuds chacun doit connaitre. Dans un grand ensemble, tous les nœuds ne peuvent pas se connaitre. Il y a un compromis à faire entre la mise à l'échelle et la fiabilité. Pour notre simulation, tous les nœuds connaissent l'ensemble du réseau. De plus, nous avons opté pour une liste centralisée des nœuds qui se créée au fur et à mesure de l'instanciation des routeurs. Donc lorsqu'un nœud se branche au réseau, il télécharge cette liste et bâti sa topologie.

Au niveau de la diffusion du message, chaque nœud qui transmet l'information inclut la liste des nœuds à qui il a transmis l'information. De cette manière on s'assure éventuellement de choisir des nœuds qui n'ont pas reçu le message. Le fait que le graphe soit connecté assure que le message sera diffusé à tous les nœuds du réseau.

Il y aura des nœuds qui recevront le même message plus d'une fois. Dans ce cas, le nœud qui reçoit une duplication de message ne fait rien avec le message. Comme la liste de nœuds visités est transmise à chaque transmission avec message, éventuellement il y a une proportion de nœuds qu'on sait qu'il ne faut pas choisir, car ils ont déjà été visités.

Il y a réellement deux approches dans le modèle Gossip pour la diffusion d'information. Une approche où lorsqu'un nœud est infecté par un message il le répète jusqu'à ce que le système soit entièrement infecté, et il y a l'approche où lorsqu'un nœud est infecté par un message il le transmet un seul round et s'arrête. Il faut aussi savoir qu'avec Gossip le maillage est considéré fort parce qu'il n'y a pas de structure et que tous les nœuds sont considérés comme voisins.

Les algorithmes d'épidémies sont reconnus pour être un moyen robuste et évolutif afin de diffuser l'information dans des milieux à grande échelle [25]. L'information est diffusée de façon fiable dans un système distribué de la même façon qu'une épidémie serait propagée dans un groupe de personnes. Chaque processus du système choisit au hasard un ou des pairs à qui il transmet les informations qu'il a reçues. Le paradigme de la communication P2P est la clé de l'évolutivité du système de diffusion. La diffusion d'un nouvel élément d'information dans le système n'est pas de l'envoyer à un serveur (ou une grappe de serveurs) responsable de retransmettre cette information, mais plutôt de transmettre l'information à un ensemble de pairs, choisis au hasard. À son tour, chacun de ces nœuds répète le processus jusqu'à ce que tous les nœuds du réseau aient reçu le message.

Deux algorithmes épidémiques ont été étudiés : Infect Forever et Infect and Die. Ces deux algorithmes produisent des résultats similaires, mais leur approche est complètement différente.

1. Infect Forever est un modèle qui suppose que le nœud infecté continuera à infecter d'autres nœuds après avoir transmis son message original. Donc, à chaque itération, il y a de plus en plus de nœuds qui transmettent l'infection.
2. Infect and Die suppose que le nœud infecté ne contaminera pas d'autres nœuds à nouveau même après avoir transmis son message original. Donc, à chaque itération, il y a un nombre fixe de nœuds qui transmettent le message.

Cette section du mémoire a présenté l'ensemble des blocs de construction d'une architecture qui sera utilisée dans ce travail. C'est-à-dire, la structure que prendra le système d'information. Selon le cadre d'architecture d'entreprise TOGAF version 9 [40], cette architecture est décomposée en couche logique que nous appelons blocs de construction (voir lexique). Les couches inférieures supportent les couches supérieures. La synthèse de ces blocs sont représentés par la Figure 1.6 et sont agencés logiquement afin de créer un système d'information logique pour supporter les fonctionnalités énoncées.

La virtualisation est la fondation de cette architecture. La virtualisation de routeurs s'appuie sur cette fondation. Le partage des ressources à grande échelle utilise la virtualisation des routeurs. Pour réaliser ce partage des ressources, une topologie de gestion de ces ressources est positionnée au-dessus du partage des ressources. Finalement, cette topologie des ressources disponibles peut être assurée, soit par le P2P ou par le Gossip.

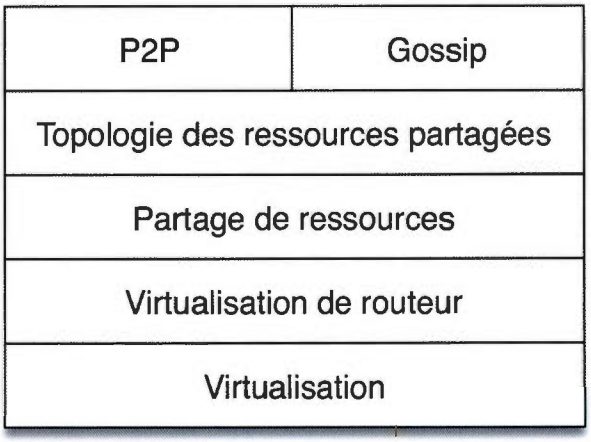

Figure 1.6 Bloc de construction de l'architecture

Ce chapitre a introduit la virtualisation et le concept de la virtualisation dans les équipements de réseaux. Par la suite, les nouvelles opérations, comme le déplacement à chaud des routeurs virtuels et la gestion de la capacité de traitement, ont été abordées. Nous avons vu l'impact de la virtualisation et de ses nouvelles opérations sur l'architecture de réseau conventionnel. Il a été démontré que pour implanter ce type d'architecture, il est nécessaire de développer une nouvelle couche de gestion de réseau ainsi qu'à quel niveau de l'architecture il faudrait implanter une couche de gestion de ressources. Les arguments de la décentralisation pour la réalisation d'opérations de façon décentralisée ont été présentés. Deux stratégies pour la décentralisation de cette nouvelle couche de gestion ont été proposées : le modèle P2P ou le modèle Gossip. Finalement, la première ébauche des blocs de construction de l'architecture a été proposée. Ces blocs technologiques ont été assemblés selon les différentes couches d'une éventuelle solution à notre problème.

CHAPITRE II

REVUE DE LITTÉRATURE

Le présent chapitre fournit une revue de littérature. Celle-ci est composée de deux types de travaux : (1) les travaux fondamentaux qui ont été utilisés pour réaliser l'environnement expérimental de nos travaux et (2) les principaux travaux similaires et les différences de nos approches.

2.1 Un Internet pour le futur

« La possibilité d'utiliser des environnements d'expérimentation à l'échelle de l'Internet permet de créer des débouchés importants pour la compréhension et l'innovation des réseaux mondiaux et de mieux comprendre leurs interactions avec la société. » [1] Tel est l'argument de base pour la création du groupe de travail GENI (Global Environment for Network Innovation). C'est un projet ambitieux qui a nécessité l'injection de 12 millions de dollars américains en 2008. Il a également proposé un design particulier et qui a nécessité plusieurs années d'analyse. Les premières estimations de coût de mise en œuvre d'un tel projet s'élèvent à plus de 350 millions de dollars américains. C'est donc un projet majeur.

C'est en 2006 qu'une douzaine de chercheurs ont proposé un nouveau design pour répondre aux besoins des réseaux de demain. La proposition définit trois objectifs importants que la solution doit respecter.

1. GENI doit permettre de réaliser des expérimentations qui répondront à des questions liées à des réseaux complexes, par exemple leur dynamisme, leur stabilité ou encore leurs comportements émergent;
2. GENI doit permettre d'évaluer d'autres structures architecturales de réseaux;
3. GENI doit permettre d'évaluer les compromis de conception, d'ingénierie ainsi que de tester les différentes théories qui pourraient être conçues.

Le système résultant des activités du groupe de travail devrait aussi être construit pour répondre à deux types d'activités : (1) déployer un prototype de réseau et dont l'apprentissage se fait par l'observation du prototype sous du trafic réseau réel, acheminé depuis l'Internet, (2) effectuer des expérimentations contrôlées afin d'évaluer les designs, l'implémentation et les choix de conception.

En gardant en tête les trois objectifs et les deux catégories d'activités, GENI doit répondre à plus d'une douzaine d'exigences (*Sliceability, Generality, Fidelity, User Access, Controlled Isolation, Diversity and extensibility, Wide Deployment, Observability, Federation and sustainability, ease of user, Security*). Pour la compréhension du présent mémoire, seules trois de ces exigences seront retenues :

1. La possibilité d'allouer une tranche (*Sliceablility*)

Le système proposé par GENI doit être partagé et capable de supporter plusieurs expériences simultanément de plusieurs groupes de recherche indépendants. La virtualisation est une technologie-clé qui permet d'atteindre cet objectif, car elle permet des installations multiplexées et indépendantes.

2. L'isolation entre les tranches

La solution doit supporter l'isolation forte entre les tranches de façon à ce que les expériences n'interfèrent pas entre elles. De plus, le mécanisme d'isolation doit fournir assez de rétroaction sur une ressource pour que les chercheurs puissent valider

leurs résultats. L'isolation doit permettre de traiter du trafic réel sans nécessairement nuire à l'Internet.

3. Grand déploiement

La solution doit avoir une portée aussi large que possible. Ceci est nécessaire pour pouvoir supporter les expérimentations à grande échelle. Un large déploiement implique aussi une interconnexion abondante entre les facilités de l'Internet. Quelques connexions sur la dorsale ne seront pas suffisantes puisque l'échantillonnage et les cas d'utilisation ne seraient pas représentatifs de la réalité.

Ces trois critères (virtualisation, isolation et grand déploiement) sont à la base des travaux qui ont mené à l'élaboration du sujet présenté dans ce mémoire.

2.2 Partage de l'infrastructure matérielle

Le partage de l'infrastructure physique, l'attribution d'une tranche de réseau et l'isolation des ressources sont des sujets couverts par VINI [3] (VIrtual Network Infrastructure). Le travail de VINI permet d'héberger plusieurs réseaux d'équipements virtuels sur une même infrastructure physique. Pour ce faire, les travaux utilisent la virtualisation comme mécanique d'isolation des ressources. En fait, VINI définit les sept (7) critères auxquels la solution doit répondre afin de réaliser le partage de l'infrastructure matérielle :

1. L'établissement d'une connectivité point à point virtuelle;
2. Une interface unique par expérimentation;
3. Chaque routeur doit posséder une table de *forwarding* distinct l'un de l'autre afin d'assurer l'isolation du trafic;
4. Chaque routeur doit posséder son propre processus de routage;
5. Permettre à n'importe quel nœud d'injecter du trafic dans VINI;
6. Isoler les ressources entre les différentes expérimentations;
7. Attribution de tranches de réseau de façon équitable.

Afin de rencontrer ces exigences de système, le groupe de recherche de VINI a réalisé l'implantation d'un tel modèle en assemblant plusieurs composantes créées pour la recherche en réseautique et par la communauté OpenSource (voir lexique) : Click [41] comme mécanisme de *forwarding*, XORP [42] comme routeur de paquets et comme plan de contrôle, OpenVPN [43] comme mécanisme de détournement de paquets et iptables/nat [44] comme mécanisme de traduction d'adresse afin de rendre le routeur transparent à l'infrastructure. Le tout s'exécute dans une machine virtuelle sous Linux, ce qui définit un modèle de routeur virtuel. Chaque nœud physique peut exécuter un ou plusieurs de ces routeurs virtuels. La création de ces routeurs virtuels introduit le besoin de découverte des ressources du réseau composé de routeurs virtuels.

2.3 L'urbanisation des routeurs virtuels et le besoin de localiser les nœuds cibles

Le déplacement de l'exécution des routeurs d'un nœud physique vers un autre [7] se bute à la nécessité de maintenir la cohérence entre le lien physique et la configuration logique des routeurs. Et cela présente un enjeu important pour l'urbanisation des routeurs virtuels. La virtualisation des équipements de réseau qui composera les prochaines générations de réseaux devra rompre le couplage fort entre le lien physique et la configuration logique en permettant aux routeurs (virtualisés) de se déplacer librement d'un nœud à un autre, sans changer la topologie de la couche IP.

Le groupe de travail de VROOM [7] propose un prototype d'expérimentation de routeurs virtuels basés sur le modèle de VINI et qui s'exécute sur un réseau qui supporte l'utilisation du protocole GRE (voir lexique). Ce protocole permet la tunnellisation (voir lexique) du protocole IP [45], et donc de découpler le lien physique de la configuration logique.

La réalisation de découplage a permis de faire ressortir l'établissement de quatre grands critères de sélection à rencontrer avant de programmer un déplacement de routeur virtuel d'un nœud à un autre :

- La latence du lien : délai que prend un paquet pour atteindre sa destination;
- La capacité du lien : quantité de paquets que peut transporter un lien pour un temps donné;
- La comptabilité de la plateforme, est-ce que l'ordinateur cible offre toutes les mêmes fonctionnalités que l'ordinateur source. Dans le contexte de routeur, est-ce que le routeur physique cible possède par exemple les mêmes interfaces physiques que le routeur source (ex. : 1000BaseT pour des liens gigabits);
- La capacité du routeur cible, est-ce que le routeur cible possède la capacité de traitement ou d'espace pour héberger le routeur source.

Ces informations seront importantes pour notre travail puisque ces quatre critères seront les informations utilisées pour créer la topologie des ressources disponibles.

2.4 Approches de découverte de ressources

Plusieurs groupes de travail [46] [47] ont proposé des techniques de découverte de ressources ou de convergence de l'information dans les réseaux P2P ou dans la Grille [48] [49]. Ces approches tentent de résoudre le temps de découverte d'une ressource spécifique dans le réseau. Plusieurs propositions ont été faites, dans un cas [49], les auteurs proposent la fédération de services, comme la création d'un bus de messages fédéré pour transporter les messages topologiques. Ce bus de messages est un réseau P2P structuré. Dans notre contexte, cette approche ne pourrait s'appliquer puisque le bus de messages utilise justement le réseau. Et ce que nous cherchons à faire c'est de transporter des messages d'une topologie de ressources de réseaux disponibles.

Dans le deuxième cas [46], les auteurs proposent plusieurs modèles de découverte de ressources : centralisé, par anneau, recherche aléatoire, par annonce de services, par rendez-vous [50] et de positionner correctement ces nœuds dans le réseau. Nous savons déjà que le modèle centralisé ne se met pas à l'échelle. Le travail a démontré que les modèles par anneau, de recherche aléatoire et par annonce de service ne peuvent pas être utilisés autrement que localement. Le modèle par rendez-vous semble être le seul modèle viable qui s'appliquerait dans notre contexte. Mais pour réaliser le modèle rendez-vous, il faut de facto connaître le nombre de nœuds qui composent notre réseau afin d'élire un nombre de nœuds rendez-vous. Cela implique une organisation préalable des nœuds afin de les compter et de les configurer.

Notre travail adresse donc un problème bien particulier qui semble original et qui n'a pas été étudié par ces différents groupes de travail. Notre approche de découverte de ressources est distribuée et décentralisée et chaque nœud physique du réseau possède sa propre copie de la topologie, ce qui permet la déstructuration de cette dernière. Ce concept a été emprunté aux protocoles de routage dans le réseau, ce qui permet à chaque nœud physique du réseau de prendre ses propres décisions topologiques. Par contre, cela implique le défi des mises à jour de chacun des nœuds.

CHAPITRE III

LE CHOIX DU PROTOCOLE

Dans le contexte présenté, il y a encore plusieurs éléments qui peuvent varier. Il y a des questions auxquelles nous sommes contraints de répondre avant de statuer sur l'algorithme que nous devons développer pour réaliser notre topologie non structurée et décentralisée.

En premier lieu, à moins de vouloir tendre vers un modèle d'inondation de l'information comme avec Gnutella, le modèle de protocole P2P implique la mise sur pied d'une structure sous-jacente au système qu'il veut supporter, que ce soit une DHT, l'établissement de raccourcis entre les nœuds ou encore une organisation par intérêt. Il y a un coût opérationnel à maintenir cette structure. Nous savons que le modèle Gossip n'a pas cette structure à gérer.

En deuxième lieu, le modèle Gossip propose deux protocoles : Infect Forever et Infect and Die. Le protocole Infect and Die est plus stable et plus prédictible. Nous démontrerons qu'il est mieux adapté à notre contexte.

En troisième lieu, nous démontrons en quoi le maillage et la taille du réseau peuvent faire varier la performance des protocoles présentés.

Afin de présenter la proposition de solutions de la problématique de notre travail, il faut répondre à ces questions essentielles. Tout d'abord, abordons les performances

des protocoles Gossip. Par la suite, le protocole que nous retiendrons sera confronté au modèle P2P. Chacune de ces comparaisons sera effectuée en fonction de la taille du réseau et de son maillage.

3.1 Comparaison entre Infect Forever et Infect and Die

La comparaison des deux algorithmes Gossip se calcule en nombre de rounds. C'est à dire le nombre d'itérations nécessaire afin que le message infecte tout un système. L'évaluation de ces deux algorithmes a été réalisée avec le simulateur Omnet++ (voir lexique). Pour la simulation, la génération automatique des réseaux se réalise avec trois paramètres : (i) un nombre de nœuds variant de façon constante, selon le scénario, entre chaque simulation, (ii) un maillage variant entre 2 et 5 liens par nœuds selon la simulation, (iii) une latence aléatoire pour chaque lien variant entre 10ms et 100ms.

Donc pour une simulation donnée, il faut spécifier le nombre de nœuds, le maillage et les limites minimales et maximales de latence des liens.

Par la suite, Omnet++ génère la quantité nœuds spécifiés. Les nœuds ont ensuite été reliés selon leurs maillages. Donc un nœud sera relié aux autres nœuds selon le maillage spécifié. Finalement, une latence aléatoirement générée (entre 10ms et 100ms) est attribuée à chacun de ces liens. De cette façon, on s'assure que l'algorithme et les résultats seront relativement indépendants d'une structure qui pourrait être volontairement prédéfinie dans le réseau. En effet, il peut subvenir des évènements aléatoirement dans un réseau et ces évènements peuvent faire changer la latence des liens ou des topologies (maillage).

Finalement, les chiffres ont été produits par le module de traçage de paquets fourni par Omnet++. Ce traçage comprend le chemin complet, (noeuds, ports, liens, ...) que prend chacun des paquets avant d'atteindre sa destination. Au total, plus de 90 simulations ont été réalisées pour une moyenne d'environ 1,8M d'entrées par

simulation. Nous avons effectué cinquante (50) itérations de chacune de ces simulations sans changer aucun paramètre. Seule la latence aléatoire sur les liens change entre les itérations pour une même configuration. Ceci a permis d'établir une moyenne de temps de convergence que nous jugeons représentative.

La mesure des rounds de propagation s'est faite en utilisant un marqueur unique pour chaque round. Chaque paquet de la simulation possède un compteur et chaque fois qu'un nœud physique retransmet le paquet, ce compteur est incrémenté de 1. À la fin de la propagation, il suffit de récupérer les marqueurs et ainsi obtenir le nombre de rounds de propagation.

L'évaluation du temps de convergence est le temps que s'exécute la simulation. Omnet++ fournit ce genre mécanisme.

L'évaluation des protocoles GOSSIP se fait par le biais des rounds, car c'est la métrique utilisée pour calculer la probabilité que chaque individu d'un système soit contaminé après un certain nombre de rounds. Cette quantité de rounds est importante pour nous afin d'évaluer l'effort requis pour garder la topologie de notre réseau à jour.

La Figure 3.7 présente les résultats de la comparaison entre Infect and Die et Infect Forever. La sortance (voir lexique) utilisée pour Infect Forever a été de 7 puisque c'est le TTL utilisé dans Gnutella et qu'il représente la distance maximale moyenne d'un nœud par rapport à l'information qu'il recherche. La Figure 3.7 présente les valeurs moyennes des résultats de la simulation.

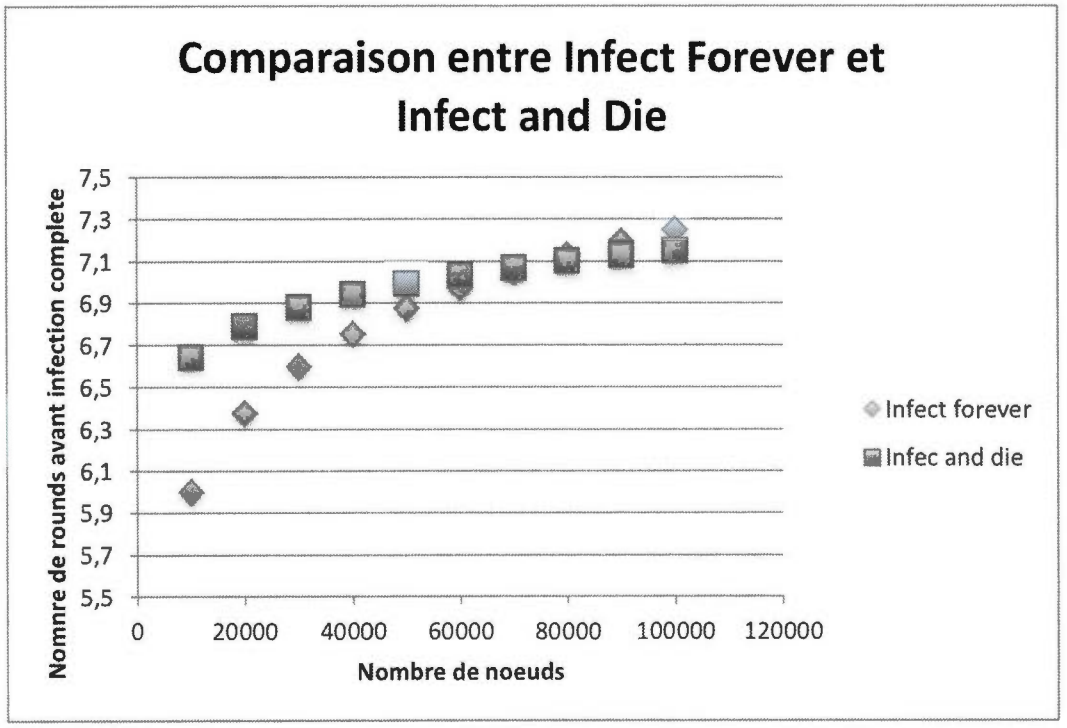

Figure 3.7 Comparaison du nombre de rounds avant infection complète d'Infect and Die et Infect Forever

Ce qu'il faut remarquer c'est qu'Infect and Die tend à se stabiliser rapidement par rapport à la taille du système. Cette stabilisation d'Infect and Die est due au fait que la sortance évolue avec la taille du système. Puisque le contexte de la virtualisation du réseau présente un potentiel pour les grands réseaux pouvant atteindre plus de 60,000 nœuds, nous préférons Infect and Die par rapport à Infect Forever. En effet, Infect and Die est moins probabiliste et plus simple à contrôler du fait que sa sortance est liée à la taille du réseau et n'est calculée qu'une seule fois, lors de l'amorce du protocole. À l'inverse, Infect Forever doit calculer sa sortance chaque fois qu'il transmet un message. Finalement, le problème avec Infect Forver est qu'à chaque round de propagation, tous les nœuds infectés rediffusent le message. Il y a donc, par design, un danger important de saturer les liens.

3.2 P2P versus Gossip

Pour comprendre l'impact d'utiliser le modèle P2P avec Pastry ou Gossip avec Infect and Die sur les réseaux actuels, nous avons calculé la quantité de paquets supplémentaires que le réseau physique devra traiter. Afin de mesurer cette nouvelle charge de travail, nous avons créé un environnement où les comportements du réseau sont contrôlables, prévisibles et mesurables. De plus, nous assumons que les paquets sont toujours acheminés vers leurs destinations et que le réseau ne présente aucune perte de paquets. De cette façon, il a été possible de mesurer précisément la charge de chacun des protocoles en mesurant la quantité de paquets générée par un évènement du réseau ou encore pour localiser des ressources dans le réseau.

Nous avons mesuré l'impact d'un évènement dans les réseaux P2P ou Gossip : l'ajout/retrait d'un nœud. Cet évènement présente la propriété de réaliser une mise à jour complète. C'est-à-dire de réaliser les ajustements à la DHT, détecter les link-up et link-down, l'ajout ou le retrait de ressources dans la topologie, etc. Nous avons aussi mesuré l'opération de localisation de nœuds pour rechercher les ressources CPU, Mémoire, de bande passante, de type de média et de type de liaison, comme présenté par VROOM [7].

La première simulation a été réalisée avec un maillage de 2. Nous avons obtenu qu'Infect and Die et Pastry se comportent de façon identique avec ce faible maillage. La Figure 3.8 montre le résultat :

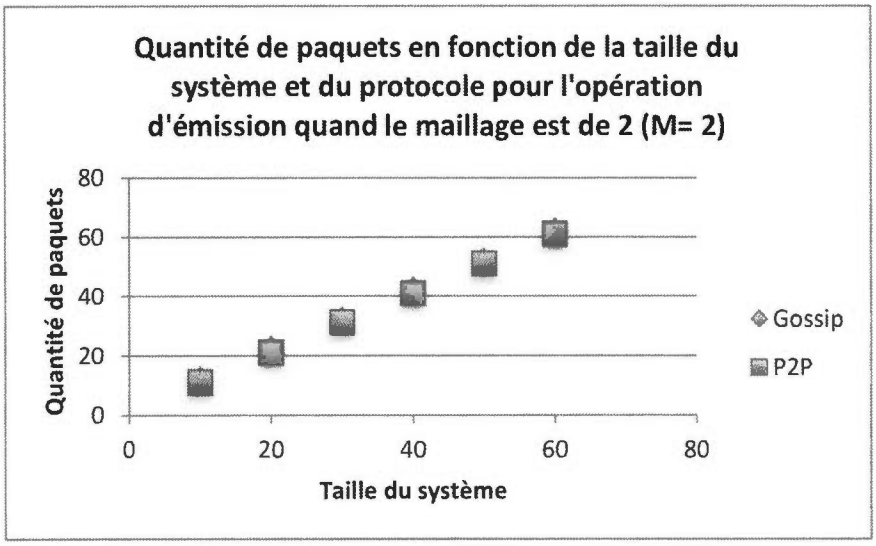

Figure 3.8 P2P versus Gossip lorsque M=2

Pour les moyennes de paquets des nœuds de 10 à 60, on retrouve des quantités de messages similaires pour P2P ou Gossip. Les écarts entre ces moyennes (que nous appellerons variance) d'un scénario de simulation à l'autre sont relativement constants et ils progressent en fonction de la taille du système. Ce qui est clair ici, c'est qu'aucun des protocoles n'a l'avantage dans un maillage de 2 puisque la quantité de chemins pour contacter tous les nœuds de la topologie est toujours de 1. Il est donc impossible de trouver un meilleur chemin pour effectuer les mises à jour. Une topologie avec un maillage de 2 se représente souvent par un anneau, comme présenté par la Figure 3.9.

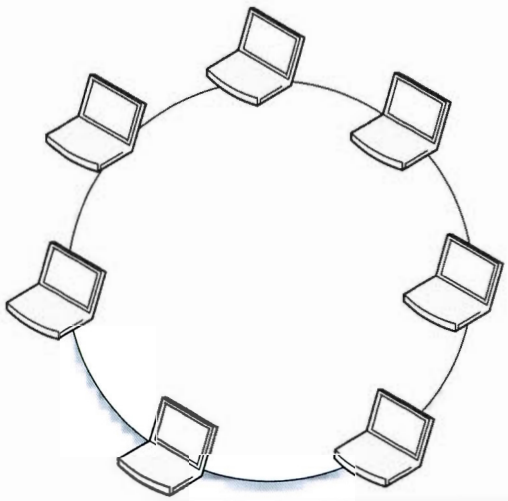

Figure 3.9 Topologie en anneau

Dans cette topologie, chaque nœud n'a qu'un chemin possible pour contacter son voisin, donc une seule possibilité de chemin.

Une topologie maillée plus fortement augmentera la possibilité d'obtenir plus de chemins. La Figure 3.10 représente un réseau avec un maillage 3. Pour contacter ses voisins, chaque nœud peut emprunter deux chemins.

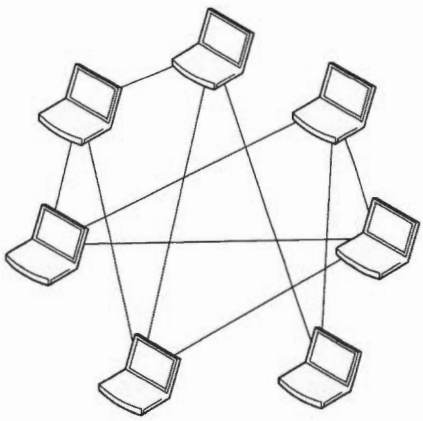

Figure 3.10 Topologie avec M=3

De la même façon, un maillage de 4 augmente la possibilité de chemins possibles pour contacter les voisins. Pour contacter ses voisins, chaque nœud peut emprunter trois chemins.

En étudiant la Figure 3.11, on peut comprendre qu'avec l'augmentation du maillage, on augmente les chemins vers les nœuds du réseau, et donc les chemins reliant les voisins augmentent aussi. Ce qui accroît les chances d'avoir un lien à faible latence.

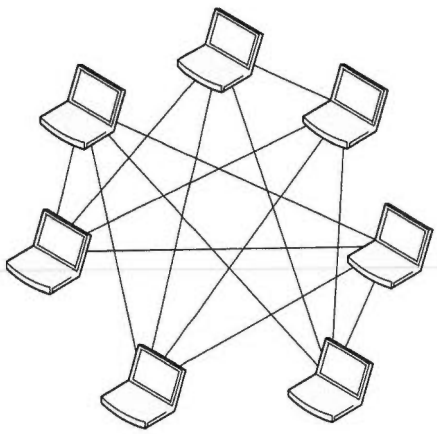

Figure 3.11 Topologie avec M=4

Ce phénomène sera démontré dans les prochains résultats. Lorsque le maillage augmente, les résultats sont nettement différents en raison de ces possibilités d'emprunter des chemins différents et à latences différentes. Cela a un impact négatif sur le P2P, mais positif pour le Gossip.

Nous voyons entre autres qu'il y a un point d'inflexion dans le P2P lorsque le réseau comprend plus de 30 nœuds. D'une manière générale, le mot inflexion désigne l'action de fléchir, de courber ou de plier. Ce point représente donc le moment où

pour le même nombre de nœuds, la quantité de paquets échangés par le protocole P2P devient beaucoup plus importante qu'avec le protocole Gossip afin de stabiliser le système.

La Figure 3.12 montre clairement une augmentation de la quantité de paquets avec le protocole P2P avec l'augmentation du nombre de nœuds dans le réseau. Alors qu'avec Gossip, la quantité est relativement stable.

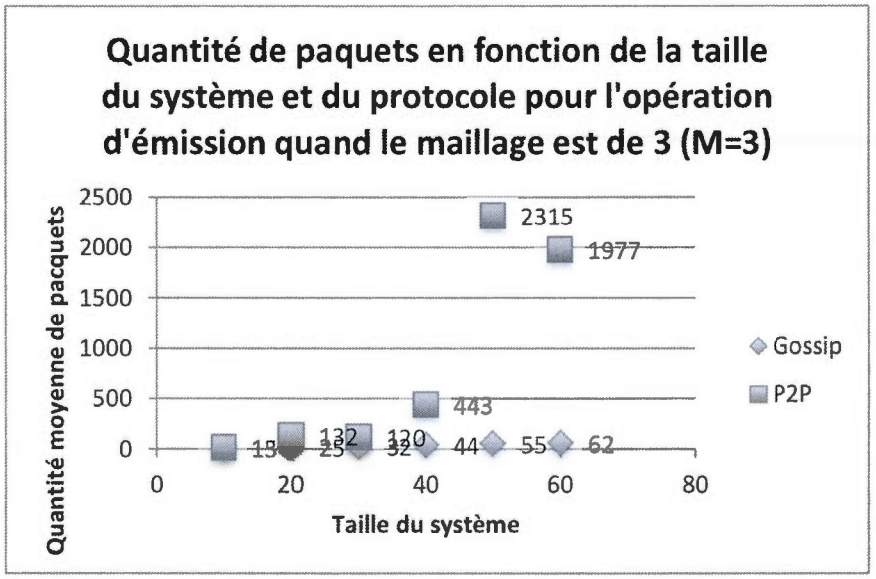

Figure 3.12 Gossip et P2P lorsque M=3

Il y a une inflexion dans les résultats où la taille dépasse 30. On peut voir la quantité de paquets échangés avec P2P exploser en fonction de la taille du système. On peut facilement constater que la quantité de paquets variera d'un maximum de 65 pour Gossip à près de 2300 pour le P2P. La quantité de paquets pour Gossip se situe autour de 45 et 65, alors que pour le P2P, il passe de 500 à plus de 2000.

La Figure 3.13 reprend la même expérience lorsque M=4. On voit que la valeur pour le protocole P2P explose rapidement. Gossip oscille toujours entre 35 et 55 paquets, alors que le P2P est au-delà de 5000 paquets dès 50 nœuds. La quantité de paquets échangés dans le P2P peut sembler élevée par rapport à Gossip. Il faut noter que la mise à jour de la DHT est une opération relativement complexe et que le routage des messages a aussi été comptabilisé. De plus, le réseau physique de la simulation a été généré aléatoirement, ce qui prouve une des limitations d'adaptation du modèle P2P.

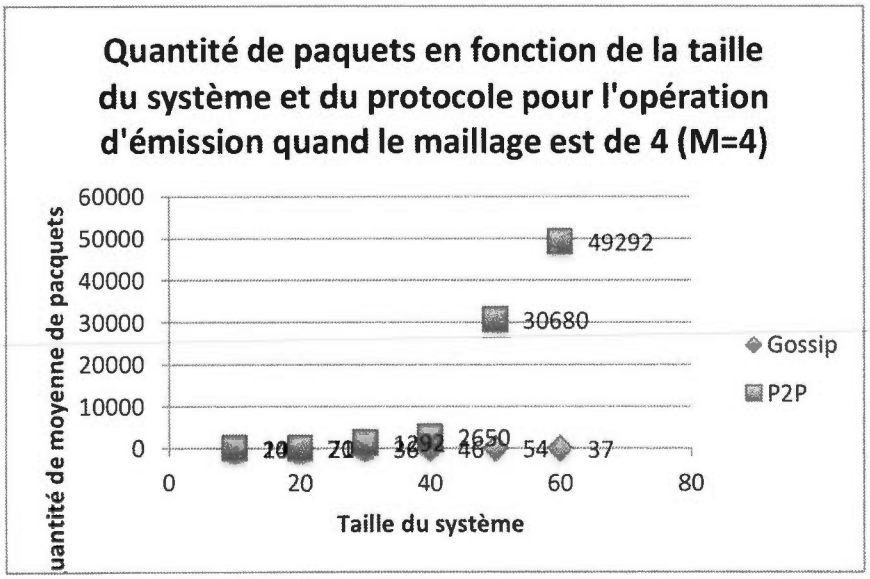

Figure 3.13 Gossip et de P2P lorsque M=4

La Figure 3.14 nous permet de statuer qu'indépendamment du maillage, le protocole P2P échange toujours plus de paquets pour réaliser la mise à jour de la topologie.

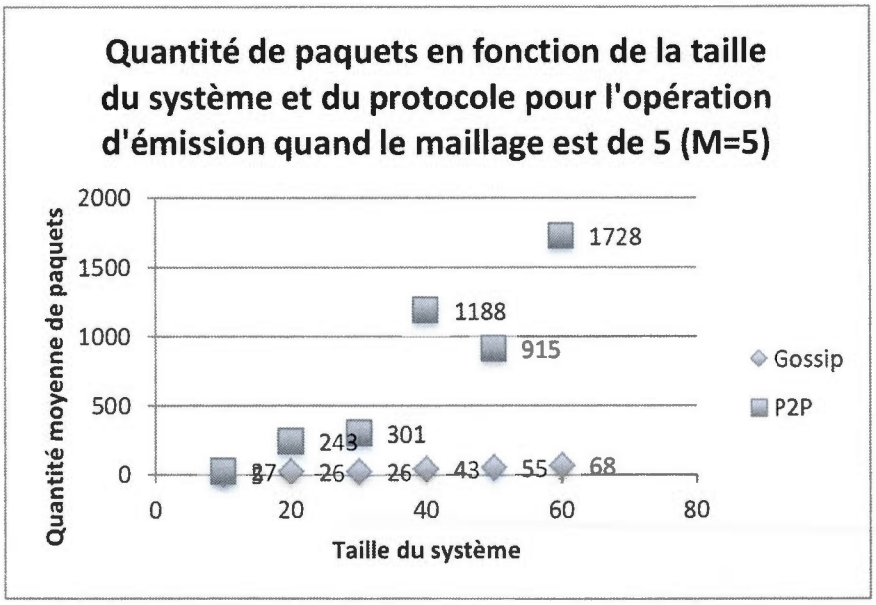

Figure 3.14 Gossip et P2P lorsque M=5

Similairement aux précédentes figures, la Figure 3.15 démontre qu'au point d'inflexion où la taille est de 40 ou plus, la quantité de paquets échangés avec le P2P est de loin supérieure qu'avec le Gossip. En effet, on peut remarquer que la quantité de paquets échangés avec Gossip se situe toujours autour de 50 et 70, alors que le P2P se situe autour de 1200 et de 1800. Donc Gossip est plus efficient que P2P pour stabiliser notre système quel que soit les latences aléatoires placées sur les différents liens.

Ces résultats font ressortir qu'indépendamment du niveau de maillage (M) au-dessus de 2 et qu'indépendamment de la taille du réseau, la quantité de messages échangés dans le protocole Pastry est largement supérieure qu'avec IAND. La raison est qu'avec P2P, pour mettre à jour les nœuds de la DHT, il faut inonder le réseau, même avec une DHT.

La comptabilisation des paquets fait ressortir que la quantité de messages échangés augmente de façon importante en fonction du nombre de nœuds et du maillage. Avec le Gossip, les nœuds sont choisis aléatoirement et les messages ne

sont émis qu'une seule fois par nœud, ce qui maintient la quantité de paquets relativement près de la quantité de nœuds du réseau, soit entre 65 et 75 pour 60 nœuds.

La supériorité de Gossip dans ce domaine le positionne comme le protocole de choix dans le modèle technologique de ce travail. Cette décision se reflète sur le diagramme des blocs de construction de la façon suivante :

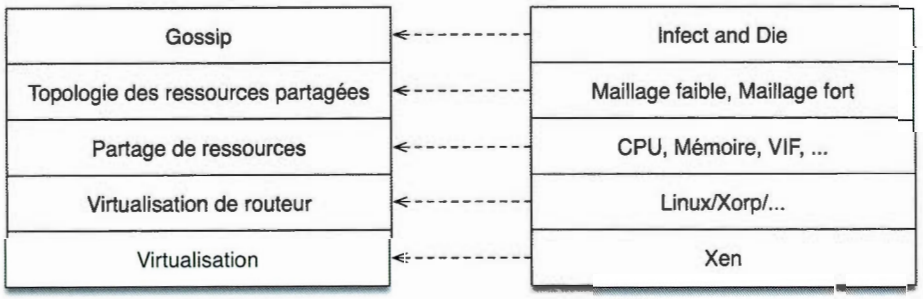

Figure 3.15 Bloc de construction de l'approche proposée

Dans ce chapitre, nous avons présenté les avantages du protocole Infect and Die (IAND) pour les grands systèmes. Il a été démontré que de façon tendancielle, le protocole subsistait mieux à l'accroissement du nombre de nœuds dans le réseau qu'Infect Forever. C'est la raison pour laquelle IAND a été préféré à Infect Forever. Par la suite, IAND a été comparé à Pastry, un protocole P2P structurel populaire afin de déterminer si IAND, qui est un protocole Gossip, pouvait être supérieur à Pastry dans un contexte de découverte des ressources dans un réseau d'équipements virtuels. Il a été établi qu'IAND était mieux adapté pour notre contexte.

La prochaine section présente les travaux qui ont été réalisés afin de démontrer comment il est possible d'utiliser la topologie du réseau physique comme levier de performance. Il sera aussi démontré qu'il est possible de faire converger la topologie des ressources disponibles en tirant avantage du maillage du réseau.

CHAPITRE IV

CHOIX DE NŒUDS BASÉ SUR LA LATENCE

Afin de répondre à la question s'il est possible de mieux utiliser le réseau qui supporte le protocole IAND, un scénario simple de dix nœuds est présenté. Ce réseau est composé de nœuds qui représentent la couche de gestion dans un réseau virtuel. Chacune des figures suivantes décrit le fonctionnement du protocole Infect and Die à chacune des étapes.

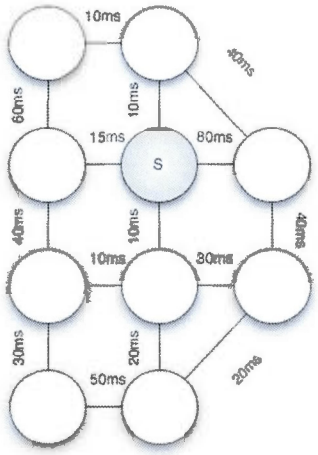

Figure 4.16 Gossip état initial du système

Le nœud marqué par S est le nœud source d'où le message sera émis. Tous les autres nœuds du réseau sont sains.

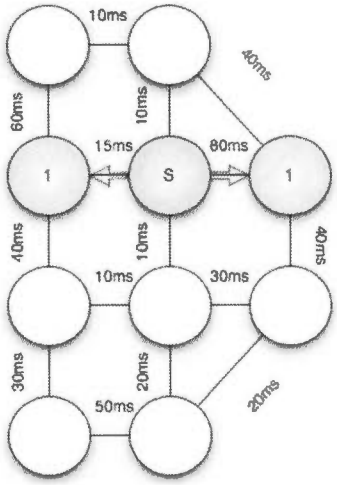

Figure 4.17 Gossip Round 1

Le nœud source choisit deux nœuds cibles aléatoirement et transmet son message.

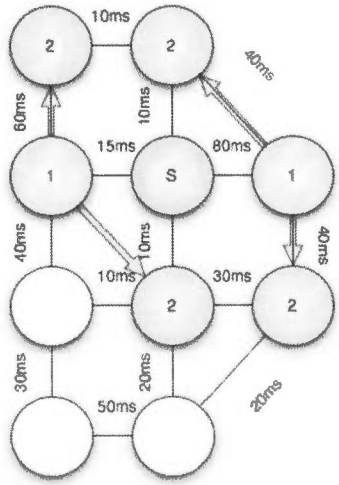

Figure 4.18 Gossip Round 2

Les deux nœuds reçoivent le message et choisissent à nouveau aléatoirement deux autres nœuds chacun, et ils retransmettent le message.

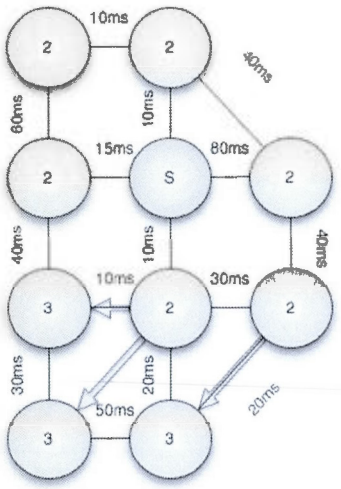

Figure 4.19 Gossip Round 3

Répétition du round 2 avec les nœuds qui ont été contaminés au round 2.

Cette version du protocole ne tient pas compte des latences entre les nœuds. Il se met à jour selon les trois rounds suivants où nous calculons l'effort de propagation à chacun des rounds :

Round 1 : 15ms + 80ms = 95ms

Round 2 : 60ms + 35ms + 40ms + 40ms = 175ms

Round 3 : 10ms + 40ms + 20ms = 70ms

Le message a donc parcouru le graphe pour un total de 340ms. Attention, ce chiffre ne représente pas le temps de parcours, mais plutôt l'effort pour parcourir le graphe puisque les mises à jour se font parallèlement.

Afin d'améliorer le temps de convergence de ce type de système, nous avons proposé de n'utiliser que les relations entre les nœuds dont les latences sont les plus basses. C'est-à-dire que le choix n'est plus aléatoire, mais privilégie les nœuds ayant une plus basse latence, ce qui a permis de définir le protocole Low Latency Infect and Die.

En effet, nous avons évalué que si, plutôt que de choisir des voisins aléatoirement, nous utilisions les nœuds avec lesquels nous avons le moins de latence, et donc de ne pas congestionner les liens les plus lents du réseau, il serait possible d'améliorer le temps de convergence de notre système.

Les figures suivantes reprennent la démonstration précédente, mais en tenant compte des latences entre les nœuds.

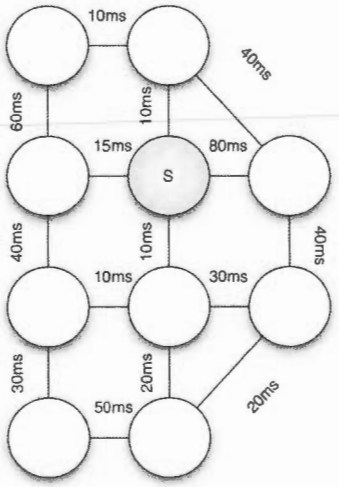

Figure 4.20 IAND état initial du système

Le nœud marqué par S est le nœud source d'où le message sera émis. Tous les autres nœuds du réseau sont sains.

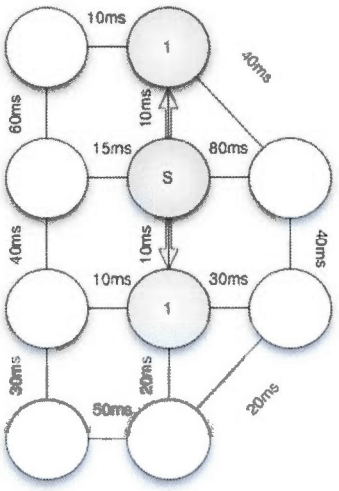

Figure 4.21 IAND Round 1

Le nœud source choisit deux nœuds cibles selon les latences les plus basses par rapport à lui et transmet son message.

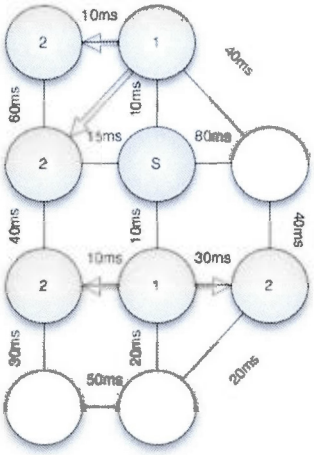

Figure 4.22 IAND Round 2

Les deux nœuds reçoivent le message et choisissent à nouveau aléatoirement deux autres nœuds qui sont rapprochés en termes de latence et chacun retransmet le message.

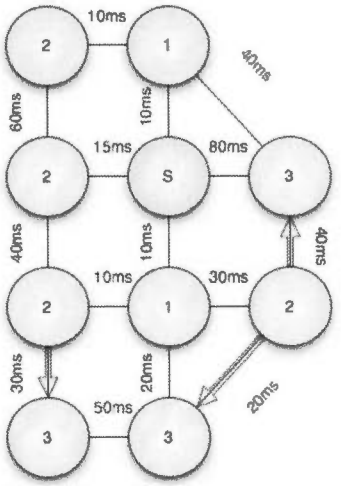

Figure 4.23 IAND Round 3

Répétition du round 2 avec les nœuds qui ont été contaminés au round 2.

Ce système tient compte des latences entre les nœuds. Il se met à jour selon les trois rounds suivants où nous calculons l'effort de propagation à chacun des rounds :

Round 1 : 10ms + 10ms = 20ms

Round 2 : 10ms + 35ms + 10ms + 30ms = 85ms

Round 3 : 30ms + 20ms + 40ms = 90ms

Le message a donc parcouru le graphe pour un total de 195ms, ce qui représente environ 1,7 fois moins d'effort de parcours que pour IAND. Cette hypothèse présente

un potentiel d'amélioration non négligeable. De plus, nous l'avons vérifié à l'aide d'une simulation que nous présentons à la prochaine section.

4.1 Création de l'algorithme LIAND

Il a été possible de mesurer précisément le temps de convergence du modèle en isolant trois variables : la taille du réseau, le maillage du réseau et un temps de latence aléatoire entre les nœuds Gossip. La simulation a été implantée avec Omnet++. Et pour les besoins de la simulation, nous avons fait varier la latence de façon aléatoire entre 10ms et 100ms et ensuite de 100ms et 5000ms entre les nœuds, selon une distribution normale. Nous avons aussi fait varier le nombre de nœuds entre 100 et 3600 et le maillage entre 3 et 5, ce qui nous donne un échantillon malléable et représentatif des mesures. Ces dernières ont été réalisées par la même technique que la comparaison de Gossip et de P2P.

Le temps de convergence varie selon le choix des nœuds pour la sortance d'un message et l'algorithme choisi IAND ou LIAND. L'algorithme IAND utilisé est représenté par la Figure 4.24. C'est un algorithme cyclique qui se termine lorsqu'il n'y a plus de nœuds à sélectionner à l'étape de sélection de la destination.

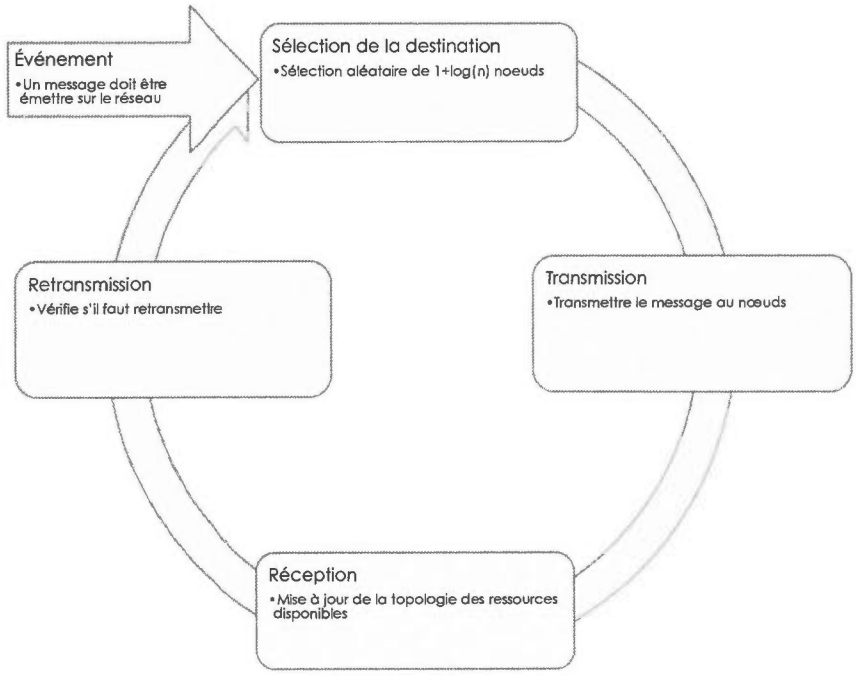

Figure 4.24 Algorithme Infect and Die

Les étapes 1 à 3 sont exécutées sur chaque nœud à chacun des rounds de transmission. Pour cet algorithme, le temps de convergence est soumis à la latence aléatoire du système. Ce temps de latence n'étant pas prévisible, il est fort possible que ce dernier impacte négativement la convergence. Afin d'optimiser l'algorithme IAND par LIAND, nous proposons l'ajustement du choix de nœud aléatoire par les nœuds avec lesquels il y a le moins de latence. La Figure 4.25 représente cet ajustement. Il faut remarquer qu'à l'étape de la sélection de la destination, le choix des nœuds n'est plus aléatoire. Il est maintenant basé sur la faible latence avec les voisins.

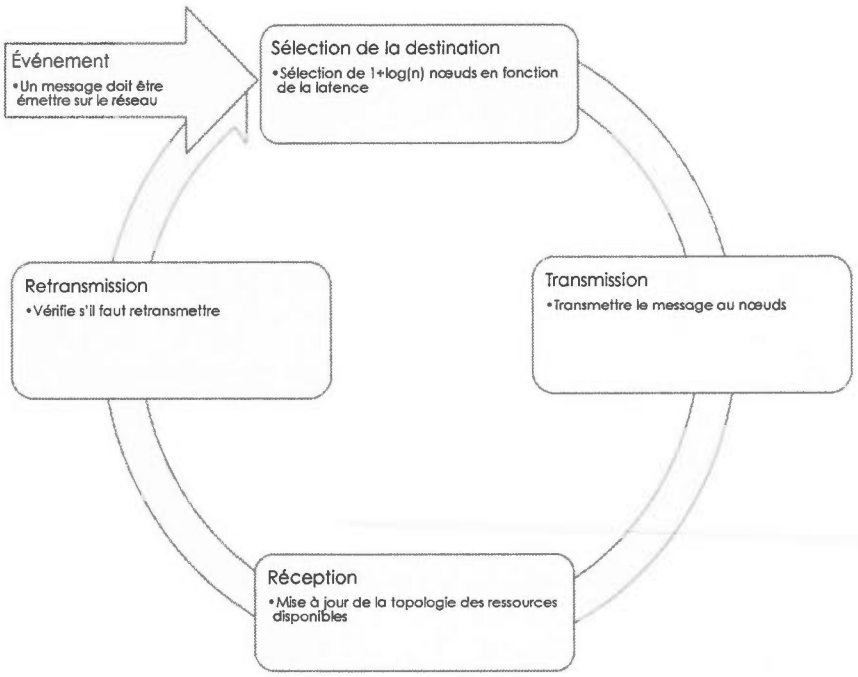

Figure 4.25 Algorithme Low Latency Infect and Die

En sélectionnant des nœuds ayant une basse latence, le temps de convergence est toujours directement lié au nombre de nœuds, mais n'est plus soumis à la latence aléatoire du système puisque le message prendra toujours le chemin optimal (avec le moins de latence) pour se propager. De plus, il sera démontré que le maillage jouera ici un rôle important.

4.2 Résultats

Les résultats que nous avons obtenus présentent une nette amélioration de LIAND par rapport à IAND : le temps de convergence est relativement constant, indépendamment du maillage ou de la latence. Pour un maillage de 3 avec IAND, le temps de convergence varie de 0,8s à 1,6s. En introduisant le choix des vecteurs de propagation selon une basse latence,

le temps de convergence varie alors de 0,05s à 0,1s. La Figure 4.26 compare le temps de convergence des deux algorithmes.

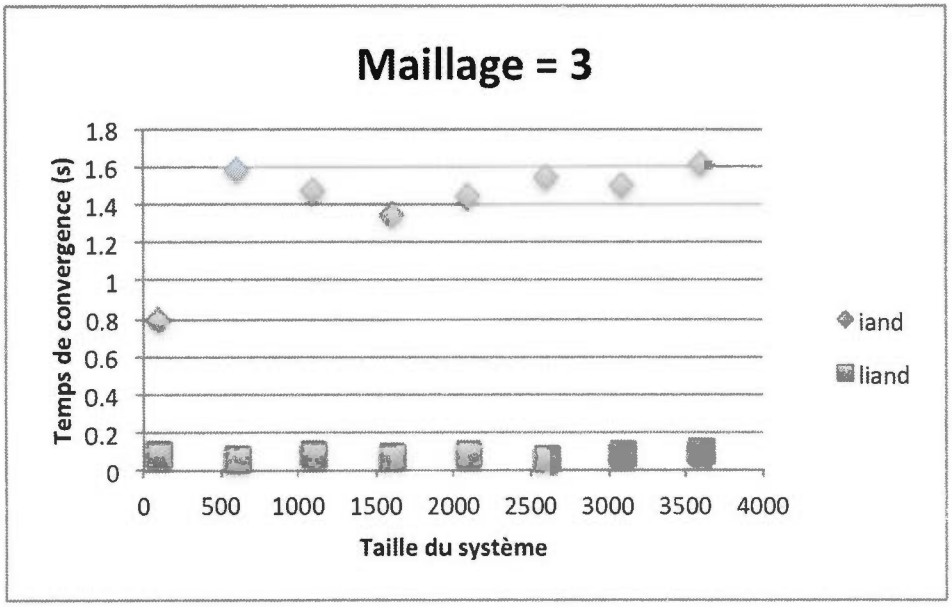

Figure 4.26 LIAND vs IAND pour un maillage de 3 (latence entre 10ms et 100ms)

Pour un maillage de quatre (4), la comparaison peut sembler moins probante. La raison est que plus il y a de liens (maillage), moins il y a de chance de sélectionner le lien avec la plus haute latence. En fait, nous avons plus de chance de choisir sur un lien avec des latences moyennes par rapport aux autres. La Figure 4.27 démontre qu'en général LIAND a permis une convergence plus rapide du système, mais on ne parle ici que d'une variance de 0,02s à 0,04s. Il faut prendre en compte qu'il y a un certain temps requis pour trouver le bon lien avec LIAND. Ceci était plus long que de choisir un lien aléatoire. Il faut aussi noter que le temps moyen de convergence de IAND est par nature plus bas que lorsque le maillage est de 3. Encore une fois, cela est dû aux possibilités plus grandes d'avoir un lien à faible latence.

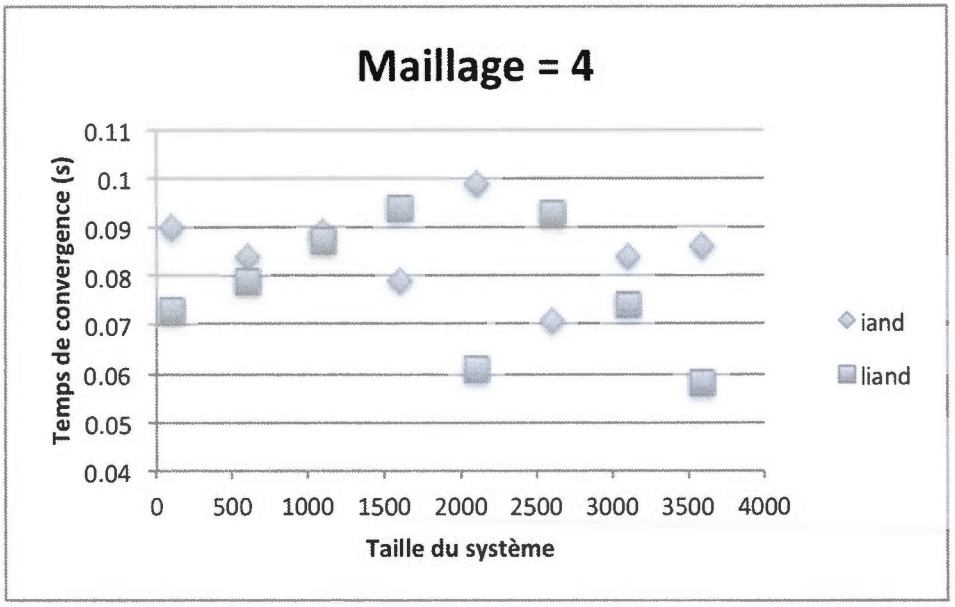

Figure 4.27 LIAND vs IAND pour un maillage de 4 (latence entre 10ms et 100ms)

Finalement, la Figure 4.28 démontre les résultats dans le cas où le maillage est de cinq (5). On y retrouve le même comportement que lorsque le maillage est de quatre (4). C'est-à-dire qu'IAND peut sembler plus rapide pour la convergence, mais sa variance (les écarts des moyennes entre les différentes simulations) est plus grande. Il faut comprendre que la nature d'IAND implique cette variance, donc lorsque les latences varient, le temps de convergence variera aussi, car la latence a un impact sur le temps de diffusion de l'information. Les pointes dans les figures le démontrent. Comme il a été introduit précédemment, un routeur moyen possède en moyenne quatre (4) interfaces, donc un maillage entre 3 et 5, nous pouvons considérer ces résultats comme étant représentatifs.

Nous pouvons donc déduire qu'il est possible de faire converger un réseau de façon précise et rapidement en utilisant LIAND quel que soit le maillage, alors que IAND est moins constant. La mise à l'échelle de ce modèle est assurée puisque le temps de convergence tend à moins varier selon la latence aléatoire du réseau. Cette

stabilité du temps de convergence est importante afin d'éventuellement prévoir les seuils de tolérances adéquats pour les mécanismes de gestion topologique des réseaux virtuels. En se basant sur les métriques de BGP [57], il est possible d'envisager qu'il est possible qu'il y aille plus de 2000 mises à jour topologique à l'heure. Il est donc important de prévoir le temps de convergence du système indépendamment de la latence moyenne des liens du réseau ou du maillage.

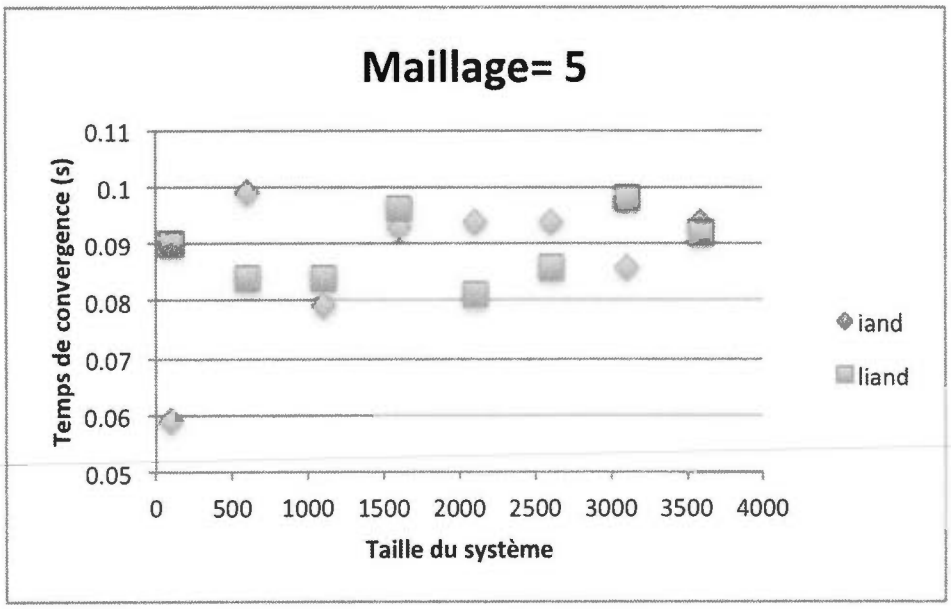

Figure 4.28 LIAND vs IAND pour un maillage de 5 (latence entre 10ms et 100ms)

Afin de démontrer cette affirmation, reprenons les tests en faisant varier la latence plus largement et aléatoirement entre 100ms à 5000ms. Le but de changer la latence est de prouver que LIAND est relativement constant indépendamment de la variance de la latence et de la taille du réseau.

En effet, la Figure 4.29 fait ressortir que la performance de LIAND, lorsque le maillage est de trois (3), est déjà plus constante qu'IAND. Au contraire, pour IAND, le temps de convergence varie de 40 à 70 secondes et cette tendance sera à la hausse en fonction du nombre de nœuds.

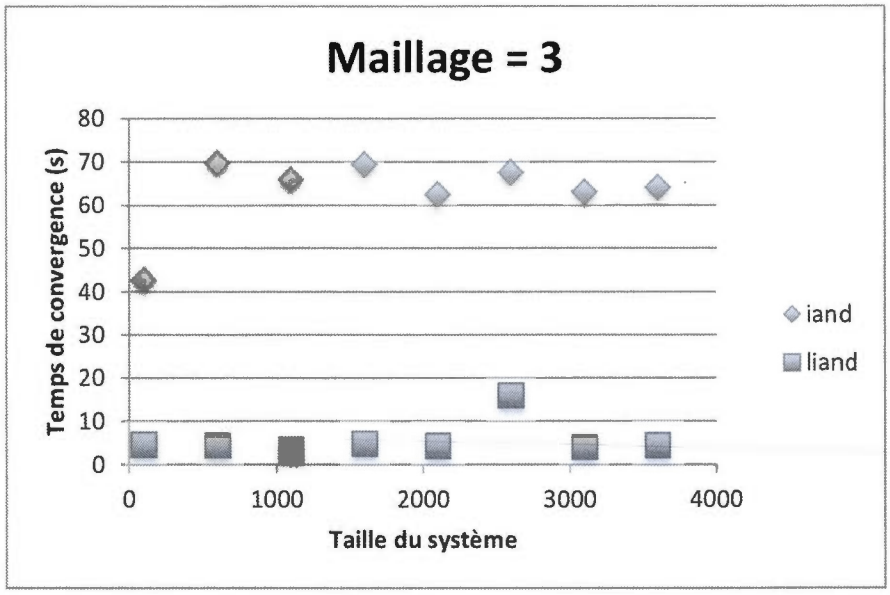

Figure 4.29 LIAND vs IAND pour un maillage de 3 (latence entre 100ms et 5000ms)

La Figure 4.30 présente le même réseau, mais avec un maillage de quatre (4). Bien qu'on remarque que la performance des protocoles est relativement similaire, LIAND est plus rapide avec l'augmentation du nombre de nœuds.

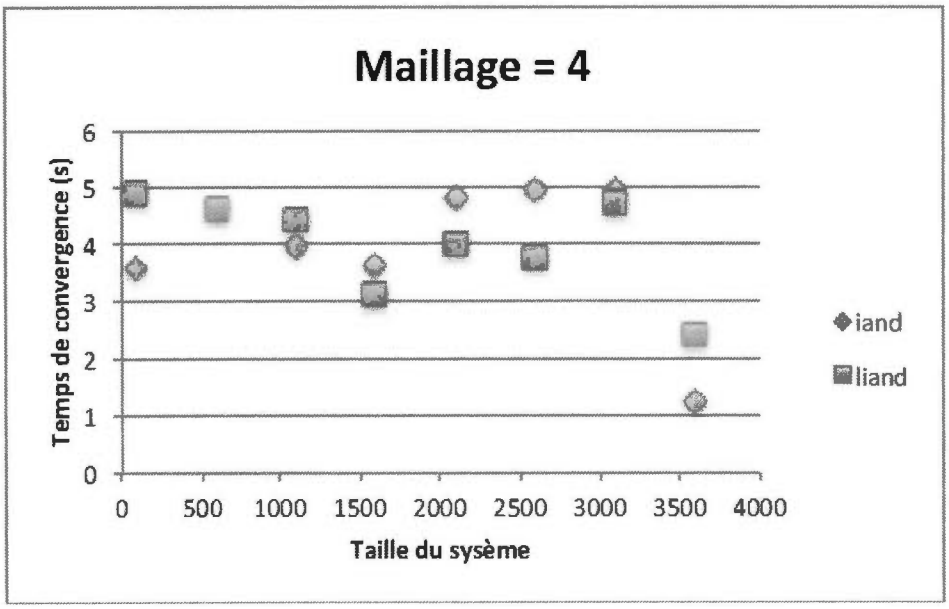

Figure 4.30 LIAND vs IAND pour un maillage de 4 (latence entre 100ms et 5000ms)

Finalement, la Figure 4.31 confirme l'hypothèse que LIAND peut s'adapter et performer pratiquement de façon identique selon le système et la latence. D'ailleurs, on voit bien cet effet avec un maillage de cinq (5) dans la Figure 4.31 où LIAND est généralement plus performant qu'IAND et dont la variance est d'environ 1 seconde. Cette variance dans la latence est importante parce qu'elle démontre que l'approche que nous avons choisie est bonne pour la majorité de nos cas d'utilisation, c'est-à-dire que LIAND performe aussi bien dans des réseaux de latence entre les nœuds variant entre 10ms et 100ms qu'entre 100ms et 5000ms.

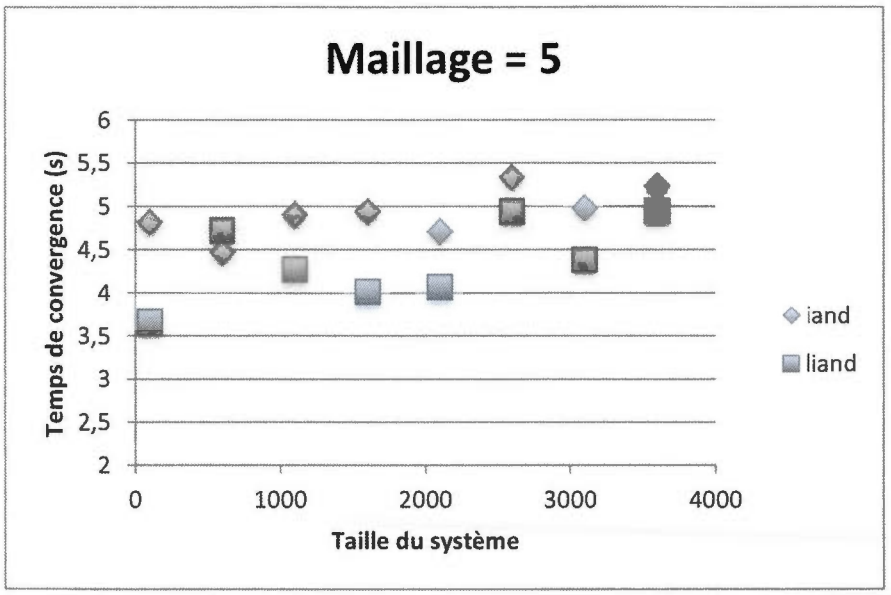

Figure 4.31 LIAND vs IAND pour un maillage de 5 (latence entre 100ms et 5000ms)

Par ce travail, nous avons évalué le temps de convergence d'un protocole Gossip que nous voulons utiliser dans une couche de gestion d'équipements de réseaux virtualisés. Nous avons introduit une amélioration au protocole Infect and Die en basant le choix des nœuds de propagation (sortance) non pas aléatoirement comme originalement proposé, mais en fonction de la latence la moins élevée. Cette proposition a permis d'améliorer le temps de convergence de façon significative et nous avons nommé cette nouvelle version du protocole Low Latency Infect and Die. La solution finale de notre travail est représentée par le diagramme de solutions suivant :

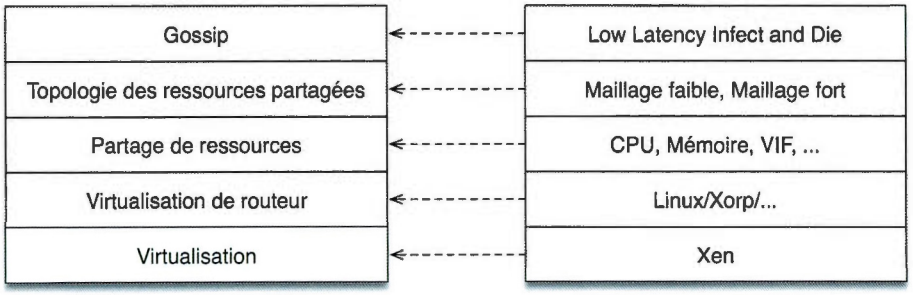

Figure 4.32 Blocs de construction de la version finale

Dans ce chapitre, il a été démontré que si nous utilisions un des paramètres du réseau physique sur lequel le réseau virtuel s'exécute, il est possible d'améliorer le protocole IAND. Le travail présenté ici utilise la latence entre les voisins comme vecteur d'optimisation. Le choix des nœuds par basse latence versus l'envoi aléatoire de paquets présente une charge de travail supplémentaire, mais l'utilisation des meilleurs liens du réseau afin de contacter les nœuds offre un avantage non négligeable, surtout dans les réseaux à haute latence. Nous avons aussi démontré que plus le maillage entre les nœuds est élevé, plus LIAND est pertinent. Finalement, nous pouvons conclure en affirmant que LIAND est un modèle de protocole qui peut s'adapter à de grands réseaux et qu'il passera la mise à l'échelle.

CONCLUSION

Ce mémoire, qui donne suite à nos travaux publié chez *Hermes* [8] et à la conférence GRES 2010 [9], a présenté un algorithme qui permet de mettre à jour une topologie de ressources disponibles pour un réseau d'équipements virtualisés. Pour ce faire, quatre éléments ont été mis en contexte : (1) le protocole Infect and Die (IAND) a été positionné et préféré à Infect Forever, (2) le protocole P2P Pastry a été déclassé au profit d'IAND, (3) une amélioration d'IAND a formé Low Latency Infect and Die (LIAND), (4) l'ajout de latence dans le réseau a permis de démontrer que LIAND est assez constant pour réaliser la convergence rapidement malgré cette latence.

Une des contributions de ce mémoire est que puisque l'expérience du protocole P2P Gnutella a fait ressortir qu'il est préférable de profiter de la topologie du réseau sur lequel on s'exécute, le protocole Gossip a été amélioré. Cette nouvelle version démontre comment l'utilisation de la topologie du réseau physique peut être un levier pour l'amélioration de sa performance.

Une autre contribution est la démonstration que l'amélioration apportée permet de stabiliser le temps de convergence d'une topologie décentralisée et non structurée indépendamment de la taille et de la latence d'un réseau, pourvu qu'il soit fortement maillé.

La contribution principale de ce travail est un algorithme permettant des mises à jour topologiques décentralisées, déstructurées, et qui ne sont pas lourdes pour le

réseau qui l'héberge. De plus, cet algorithme passe les tests de mise à l'échelle vers un grand réseau.

Le protocole LIAND est un modèle puissant qui pourrait aussi être utilisé dans d'autres contextes que celui présenté dans ce mémoire. Il suffit de penser à l'informatique dans le nuage (*Cloud Computing*) où les besoins d'infrastructures varient régulièrement selon les besoins des utilisateurs. Les ressources disponibles par les fournisseurs d'infrastructures varient à chaque instanciation ou mise en déroute d'un service. Le modèle de LIAND est parfaitement adapté pour gérer ce genre de problématique.

ANNEXE A

Les configurations de Omnet++ qui ont été utilisées pour produire les résultats

On remarque qu'il y a deux configurations : P2PLightlyMeshed et P2PStronglyMeshed. Pour chacune de ces configurations, il y a 3 paramètres communs :

- numRouters qui définit la quantité de routeurs qui seront créés dans la simulation;
- gossipNodes qui définit le *fanout* de Gossip;
- routingProtocol qui définit les protocoles de routage qui seront utilisés.

Pour la configuration P2PStronglyMeshed :

- mesh : définit le maillage qui sera utilisé dans la simulation;
- linkLag : une latence aléatoire choisit entre deux valeurs;
- minLinkDelay, minMaxDelay : limite les valeurs de linkLag;

- pRoutingVersion : version du protocole Gossip qui sera utilisé dans la simulation.

```
Omnet.ini
[General]

cmdenv-express-mode = true
fname-append-host = false
parallel-simulation = false
parsim-filecommunications-preserve-read = true
**.aiTimer = intuniform(1,100)
**.delayTime = 1000ms

[Config P2PLightlyMeshed]
network = Network
eventlog-file = ${resultdir}/${configname}-${runnumber}.elog
record-eventlog = true
**.partition-id =
**.numRouters = ${n=10..60 step 10}
**.gossipNodes = log10(${n}) + 1
**.routingProtocol = ${"P2P", "Gossip"}

[Config P2PStronglyMeshed]
network = P2PNetworkMeshed
eventlog-file = ${resultdir}/${configname}-${runnumber}-meshed.elog
record-eventlog = true
**.partition-id =
**.numRouters = ${n=100..6500 step 500}
**.gossipNodes = log10(${n}) + 1
**.routingProtocol = ${"Gossip"}
**.mesh = ${m=3..5}
**.linkLag = truncnormal(100ms,5000ms)
**.proutingVersion = ${"iand", "liand"}
**.minLinkDelay = 100
**.maxLinkDelay = 5000
```

Les simulations ont été lancées de façon à couvrir l'ensemble des cas. Par exemple, un réseau de 50 nœuds, utilisant Gossip Infect and Die avec un maillage de 3 dont les latences entre les nœuds varient de 100 à 5 000.

À titre informatif, la configuration Config P2PLightlyMeshed a produit 12 sets de résultats, la configuration Config P2PStronglyMeshed a produit 78 sets de résultats. Pour un total de 210MB de données, ces résultats ont pris près de 128 heures pour être produits.

RÉFÉRENCES

[1] L. Peterson, T. Anderson, D. Blumenthal,, D. Casey, D. Clark, D. Estrin, J. Evans, D. Raychaudhuri, M. Reiter, J. Rexford, S. Shenker and J. Wroclawski, "GENI Design Principles," *Computer,* vol. 39, no. 9, pp. 102--105, 2006.

[2] L. Peterson, A. Bavier, Marc Fiuczynski and .. S. Muir., "Experiences Building PlanetLab," *Proceedings of the Seventh Symposium on Operating System Design and Implementation (OSDI),* 2006.

[3] A. Bavier, N. Feamster, M. Huang, L. Peterson and J. Rexford, "In VINI veritas: realistic and controlled network experimentation," *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications,* p. 14, 2006.

[4] N. Feamster, L. Gao and J. Rexford, "How to lease the Internet in your spare time," *ACM,* vol. 2007, pp. 61--64.

[5] D. Chisnall, The definitive guide to the xen hypervisor, Prentice Hall Press Upper Saddle River, NJ, 2007.

[6] N. Feamster, L. Gao and J. Rexford, "How to lease the Internet in your spare time," *ACM SIGCOMM Computer Communication Review,* vol. 37, pp. 61--64, 2007.

[7] W. Yi, M. V. D. Jacobus and J. Rexford, "VROOM: Virtual routers on the move," in *Proc. ACM SIGCOMM Workshop on Hot Topics in Networking*, 2007.

[8] M. Héroux and O. Cherkaoui, "LIAND - Convergence à basse latence," in *Gestion de Réseaux Et de Services*, 2010.

[9] M. Heroux and O. Cherkaoui, "Chapitre 9. P2P et Gossip pour la gestion des réseaux d'équipements virtuels.," in *Evolution des technologies Pair-à-Pair: optimisation, sécurité et application*, Hermès/Lavoisier, 2010.

[10] V. Infrastructure, "Resource management with VMware DRS," *VMware Whitepaper*, 2006.

[11] M. Laverick, "IT Handbook - Server Consolidation," Dell, 2010.

[12] X. Community, "Xen Community," 2010. [Online]. Available: http://xen.org.

[13] C. Clark, K. Fraser, S. Hand, J. Hansen, E. Jul, C. Limpach, I. Pratt and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, USENIX Association, 2005, pp. 273-286.

[14] Wikipedia, 16 04 2010. [Online]. Available: http://fr.wikipedia.org/wiki/Grille_informatique.

[15] SETI@Home, 1 1 2010. [Online]. Available: http://setiathome.ssl.berkeley.edu/.

[16] S. Koren, "BOINC: A System for Public-Ressource Computing and Storage," California, 2005.

[17] J. N. Inc. [Online]. Available: http://www.juniper.net/us/en/products-services/nos/junos/.

[18] C. S. Inc.. [Online]. Available:

http://www.cisco.com/en/US/products/ps6537/products_ios_sub_category_home.html.

[19] svrops. [Online]. Available: http://www.svrops.com/svrops/documents/ciscoboot.htm.

[20] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee and N. McKeown, "ElasticTree: Saving energy in data center networks," *Submitted to ACM SIGCOMM,* 2009.

[21] M. Agrawal, S. Bailey, A. Greenberg, J. Pastor, P. Sebos, S. Seshan, J. v. d. Merwe et J. Yates., «RouterFarm: Towards a dynamic, manageable network edge.,» chez *Proceedings of the 2006 SIGCOMM workshop on Internet network management*, 2006.

[22] VMware. Inc, «Configuration Maximums VMware® vSphere 4.0 and vSphere 4.0 » 2009. [Online]. Available: http://www.vmware.com/pdf/vsphere4/r40/vsp_40_config_max.pdf

[23] Wikipedia, 2010. [Online]. Available: http://en.wikipedia.org/wiki/SETI@home.

[24] Wikipedia, 2010. [Online]. Available: http://en.wikipedia.org/wiki/Peer-to-peer.

[25] P. Eugster, R. Guerraoui, A. Kermarrec and L. Massoulie, "From epidemics to distributed computing," *IEEE computer,* vol. 37, no. 5, pp. 60--67, 2004.

[26] Wikipedia, 2010. [Online]. Available: http://en.wikipedia.org/wiki/Network_topology.

[27] Wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/Napster.

[28] Wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/BitTorrent_(protocol).

[29] F. Bordignon and G. Tolosa, "Gnutella: Distributed System for Information Storage and Searching Model Description," *Journal of Internet Technology, Taipei (Taiwan),* vol. 2, no. 5, 2002.

[30] M. Ripeanu, I. Foster and A. Iamnitchi, "Mapping the gnutella network: Properties of

large-scale peer-to-peer systems and implications for system design," *IEEE Internet Computing,* vol. 6, pp. 50-57, 2002.

[31] F. Dabek, "A distributed hash table," 2005.

[32] Wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/Big_O_notation.

[33] Wikipedia, 2010. [Online]. Available: http://en.wikipedia.org/wiki/Distributed_hash_table.

[34] T. A. S. Foundation, «The Apache Cassandra Project,» 2009. [En ligne]. Available: http://cassandra.apache.org/.

[35] A. Lakshman, August 2008. [En ligne]. Available: http://www.facebook.com/note.php?note_id=24413138919.

[36] K. Sripanidkulchai, B. Maggs and H. Zhang, "Efficient content location using interest-based locality in peer-to-peer systems," *DEF,* vol. 3, no. 3, 2002.

[37] KaZaa, 2010. [Online]. Available: http://www.kazaa.com/.

[38] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware),* vol. 11, pp. 329--350, 2001.

[39] S. Verma et W. T. Ooi, «Controlling Gossip Protocol Infection Pattern Using Adaptive Fanout,» chez *ICDCS '05 Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, 2005.

[40] The OpenGroup, TOGAF Version 9, Van Haren, 2009.

[41] E. Kohler, R. Morris, B. Chen, J. Jannotti and M. Kaashoek, "The Click modular router," *ACM Transactions on Computer Systems,* vol. 18, pp. 263--297, 2000.

[42] XORP, [Online]. Available: http://www.xorp.org/. [Accessed 2009].

[43] OpenVPN, [Online]. Available: http://openvpn.net/.

[44] Wikipedia, 2010. [Online]. Available: http://en.wikipedia.org/wiki/Iptables.

[45] Wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/Tunneling_protocol.

[46] A. Iamnitchi, I. Foster and D. Nurmi, "A peer-to-peer approach to resource location in grid environments," *INTERNATIONAL SERIES IN OPERATIONS RESEARCH AND MANAGEMENT SCIENCE,* pp. 413--430, 2003.

[47] D. Zhou and V. Lo, "Cluster Computing on the Fly: resource discovery in a cycle sharing peer-to-peer system," *CCGrid,* pp. 66--73, 2004.

[48] I. Foster and A. Iamnitchi, "On death, taxes, and the convergence of peer-to-peer and grid computing," *Peer-to-Peer Systems II,* pp. 118--128, 2003.

[49] G. Fox, D. Gannon, S. Ko, S. Lee, S. Pallickara, M. Pierce, X. Qiu, X. Rao, A. Uyar, M. Wang and others, "Peer-to-peer Grids," *Grid Computing-Making the Global Infrastructure a Reality. John Wiley \& Sons Ltd,* pp. 471--490, 2003.

[50] V. Lo, D. Zhou, Y. Liu, C. GauthierDickey and J. Li, "Scalable supernode selection in peer-to-peer overlay networks," in *Hot Topics in Peer-to-Peer Systems, 2005. HOT-P2P 2005. Second International Workshop on*, IEEE, 2005, pp. 18--25.

[51] M. a. N. T. Matsumoto, «Mersenne twister: A 623-dimensionally equidis- tributed uniform pseudo-random number generator.,» *ACM Transactions on Modeling and Computer Simulation,* vol. 8, p. 3–30, 1998.

[52] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt and A. Warfield, "Xen and the art of virtualization," *Proceedings of the nineteenth ACM symposium on Operating systems principles,* p. 177, 2003.

[53] Cisco "3900", [Online]. Available: http://www.cisco.com/en/US/prod/collateral/routers/ps10536/data_sheet_c78_553924.html. [Accessed 2011].

[54] Juniper, "J SERIES SERVICES ROUTERS." [Online]. Available: http://www.juniper.net/us/en/local/pdf/datasheets/1000206-en.pdf [Accessed: 2011].

[55] A. Turtschi, "Complete List of Class A and Class B Networks." [Online]. Available: http://www.aturtschi.com/whois/networks.html. [Accessed: 2011].

[56] I. Research, "@Infonetics: Cisco storms back in IP Edge market, Huawei posts biggest loss." [Online]. Available: http://www.infonetics.com/pr/2011/3Q11-Service-Provider-Routers-Switches-Market-Highlights.asp. [Accessed: 2011].

[57] "BGP Routing Table Analysis Reports." [Online]. Available: http://bgp.potaroo.net/. [Accessed: Dec-2011].

[58] O. Babaoglu, M. Marzolla, and M. Tamburini, "Design and Implementation of a P2P Cloud System.", *Technical Report UBLCS-2011-10*

[59] M. Marzolla and S. F. G. D'Angelo, "Dynamic scalability for next generation gaming infrastructures," in *Proc. 4th ACM/ICST International Conference on Simulation Tools and Techniques (SIMUTools 2011)*, 2010, pp. 1–8.

[60] R. Yanggratoke, F. Wuhib, and R. Stadler, "Gossip-based Resource Allocation for Green Computing in Large Clouds (long version)," *KTH Royal Institute of Technology, https://eeweb01. ee. kth. se/upload/publications/reports/2011/TRITA-EE*, vol. 36, 2011.

[61] Gartner Research, "Google: one million servers and counting," Jul-2007. [Online]. Available: http://www.pandia.com/sew/481-gartner.html. [Accessed: 2011].

[62] A. Allavena, A. Demers, and J. E. Hopcroft, "Correctness of a gossip based membership protocol," in *Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, 2005, pp. 292–301.

[63] M. Gurevich and I. Keidar, "Correctness of gossip-based membership under message loss," in *Proceedings of the 28th ACM symposium on Principles of distributed computing*, 2009, pp. 151–160.

[64] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On Power-Law Relationships of the Internet Topology," in *In SIGCOMM*, CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE, 1999, p. 251--262.