Vers une architecture pair-à-pair pour l'informatique dans le nuage

Willy Malvaut-Martiarena

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

Willy Malvault-Martiarena

Thèse dirigée par **Jean-Bernard Stéfani** et
codirigée par **Vivien Quéma**

préparée au sein de l'INRIA
dans **l'École Doctorale MSTII**

Vers une architecture pair-à-pair pour l'informatique dans le Nuage.

Thèse soutenue publiquement le **4 Octobre 2011**,
devant le jury composé de :

Mme. Kermarrec Anne-Marie
Directeur de recherche à l'INRIA Rennes, Rapporteur.
M. Pierre Sens
Professeur à l'Université Paris 6, Rapporteur.
M. Pascal Felber
Professeur à l'Université de Neuchâtel, Examinateur.
M. Yves Denneulin
Professeur à l'ENSIMAG, Examinateur.
M. Jean-Bernard Stefani
Directeur de recherche à l'INRIA Rhônes-Alpes. Directeur.
M. Vivien Quéma
Professeur à l'INP Grenoble, Co-directeur.

à Elise, née pendant la rédaction de ce manuscrit.

Résumé

Avec l'émergence de l'informatique dans les nuages, une nouvelle approche consiste à externaliser des tâches de calcul, de façon à réduire les coûts d'hébergement et à augmenter la flexibilité des systèmes. L'infrastructure actuelle des services permettant cette externalisation repose sur l'utilisation de centres de traitement de données centralisés, qui sont dédiés à l'approvisionnement de ressources de calcul. Dans cette thèse, nous étudions la possibilité de fournir de tels services en utilisant une infrastructure pair-à-pair, c'est-à-dire une infrastructure totalement décentralisée pouvant être déployée sur une fédération de noeuds de calcul hétérogénes et de provenances diverses. Nous nous focalisons sur le problème de l'allocation des noeuds et présentons Salute, un service d'allocation de noeuds, qui organise les noeuds en réseaux virtuels non-structurés et repose sur des mécanismes de prédiction de disponibilité pour assurer, avec une grande probabilité, que les requêtes d'allocation sont satisfaites dans le temps, malgré le dynamisme de l'environnement hôte. Pour ce faire, le service Salute repose sur la collaboration de plusieurs protocoles pair-à-pair appartenant à la catégorie des protocoles épidémiques. Afin de valider nos propositions, nous évaluons Salute en utilisant des traces provenant d'un échantillonnage de plusieurs systèmes pair-à-pair de référence.

Abstract

With the emergence of Cloud computing, a new trend is to externalize computing tasks in order to decrease costs and increase flexibility. Current Cloud infrastructures rely on the usage of large-scale centralized data centers, for computing resources provisioning. In this thesis, we study the possibility to provide a peer-to-peer based Cloud infrastructure, which is a totally decentralized and can be deployed on any computing nodes federation. We focus on the nodes allocation problem and present Salute, a nodes allocation service that organizes nodes in unstructured overlay networks and relies on mechanisms to predict node availability in order to ensure, with high probability, that allocation requests will be satisfied over time, and this despite churn. Salute's implementation relies on the collaboration of several peer-to-peer protocols belonging to the category of epidemic protocols. To convey our claims, we evaluate Salute using real traces.

Remerciements

Tout d'abord je tiens à remercier Jean-Bernard et Vivien de m'avoir attribué le poste de doctorant qui m'a permi de réaliser la thèse décrite dans ce manuscrit. Les trois ans et demi que j'ai passé à leurs côtés ont été riches du point de vue scientifique, mais aussi du point de vue humain. Je pense avoir appris beaucoup de choses pendant mon doctorat grâce à ces deux personnes. Ensuite, je souhaite remercier Anne-Marie Karmarrec et Pierre Sens de m'avoir fait l'honneur de rapporter sur ma thèse, ainsi que Yves Denneulin et Pascal Felber pour avoir joué le rôle d'examinateur. Je remercie également l'ensemble des membres de l'équipe Sardes pour leur accueil chaleureux et pour leur disponibilité au quotidien. En particulier, la diversité des domaines étudiés dans cette équipe m'a permi d'entrevoir l'informatique à travers un large spectre, en partant de l'informatique formelle, jusqu'à la programmation de système d'exploitation, en passant par le développement d'intergiciel autoconfigurable. C'est ainsi que j'ai pu apprendre ce qu'était une monade (en réalité je ne suis toujours pas convaincu d'avoir compris) et que j'ai pu comprendre comment l'on procède pour développer un système d'exploitation (je ne suis pas sûr de savoir le refaire non plus). Je remercie également mes collègues de bureau Valerio et Alessio, à qui je dois le nom de la principale contribution de cette thèse : "Salute". Particulièrement, merci à Alessio pour son soutien logistique lors de ces derniers mois. Merci également à tous le "upper Sardes" pour leur bonne humeur au quotidien. Merci à Alan et Renaud pour leur disponibilité et leur efficacité à résoudre les problèmes de logistique. Merci à Fabienne et Noël de m'avoir trouvé du tavail pour l'après thèse. Merci à Olivier pour toutes les discussions de couloir que j'ai pu avoir avec lui. Merci à Diane pour sa patience et son efficacité. Et bien entendu, merci à tous mes proches. Merci à ma mère, de qui je pense avoir hérité la curiosité scientifique nécessaire pour faire une thèse et grâce à qui j'ai pu faire de longues études, et à Caroline (et plus récemment Elise), pour m'avoir supporté au quotidien (quel courage !), pendant les 4 années de mon doctorat.

Table des matières

Table des figures

Introduction

Depuis le début des années 1990, avec la démocratisation des accès à Internet et avec la création des standards du Web, la variété des services rendus par l'informatique ne cesse de croître. Aujourd'hui, bien que toutes les générations d'utilisateurs ne soient pas encore parfaitement familiarisées avec les outils informatiques, la consommation de services logiciels se généralise à de nombreux domaines de la vie courante. Pour ne citer que les fonctions les plus populaires, ces services permettent le réseautage social, la vente par correspondance, la diffusion de média, la réservation de billets de transports ou d'hôtels, la gestion de comptes bancaires ou encore la déclaration d'impôts. Leur aspect pragmatique et leur avantage économique sont tels qu'en une dizaine d'années d'existence ils sont déjà adoptés par plusieurs centaines de millions d'utilisateurs à travers le monde entier.

Ces services logiciels sont distribués via Internet. Ils utilisent des plateformes d'exécution et des canaux de communications hétérogènes, allant de la connexion domestique via ordinateur personnel (*PC*) à l'accès mobile via téléphone évolué (*smartphone*), en passant par la connexion à un réseau sans fil public via un ordinateur portable (*wifi*). Depuis quelques années, l'espace d'hébergement, de développement et de distribution de ces services logiciels s'est vu associer le concept de Nuage. Ce terme fait référence à l'ensemble des services logiciels et des moyens informatiques mis en oeuvre pour les exécuter. La vu du Nuage, du point de vue des utilisateurs de services logiciel, est illustrée sur la Figure 1.

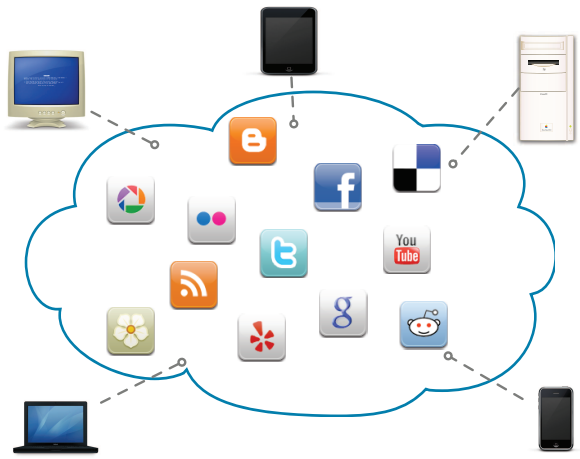


FIGURE 1 – Nuage vu par les utilisateurs de services logiciels.

Aussi, cette distribution de services logiciels à grande échelle pose des problèmes de gestion de ressources informatiques. Les différents taux de consommation de services logiciels étant fluctuants, les méthodes traditionnelles d'approvisionnement en ressources matérielles,

telles que les centres de traitement de données privés, montrent leurs limitations. En effet, les prestataires de services logiciels se heurtent à des problématiques complexes quant au calibrage de leurs applications et architectures informatiques : d'un côté, s'ils ne prévoient pas suffisamment de ressources, leurs services ne supportent pas certains pics d'utilisation et deviennent partiellement indisponibles, ce qui est économiquement catastrophique. D'un autre côté, s'ils prévoient trop de ressources pour l'hébergement de ces mêmes services, alors le taux d'utilisation des machines diminue et le retour sur investissement du matériel devient difficile. De plus, l'impact du calibrage de ces services logiciels est d'autant plus important que leurs échelles de déploiement sont grandes [29].

En 2006, une nouvelle gamme de services informatiques, vouée à résoudre ce problème de gestion de ressources, apparaît au sein du Nuage. C'est l'infrastructure en tant que service. Cette gamme de services propose de distribuer de la capacité de calcul et de stockage publiquement, via Internet. Elle permet, entre autres, aux prestataires de services logiciels d'adapter dynamiquement leur infrastructure informatique aux besoins fluctuants de leurs utilisateurs. Un nouveau modèle de consommation de ressources informatiques est alors crée. Ce modèle est économiquement viable dans le sens où il permet, d'une part au fournisseur d'infrastructure de rentabiliser ses machines inutilisées, et d'autre part au consommateur d'infrastructure d'assouplir son mode d'investissement en ressources matérielles. De façon simplifiée, l'infrastructure en tant que service constitue, avec les services logiciels, ce que l'on appelle aujourd'hui « l'informatique dans le Nuage ».

Avant l'apparition de l'informatique dans le Nuage, les premiers systèmes permettant une gestion économique des ressources informatiques, dans le cadre de services logiciels distribués à large échelle, étaient les systèmes pair-à-pair. Ces systèmes, apparus au début du siècle avec les applications de partage de fichiers telles que Napster [14], ont pour particularité de fonctionner selon un modèle d'exécution collaboratif. Ce modèle d'exécution met à contribution les ressources informatiques des clients et permet, de fait, d'étendre la capacité de calcul et de stockage nécessaire au fonctionnement du service au fur et à mesure que le nombre de clients augmente. Une autre caractéristique appréciée des systèmes pair-à-pair est leur autonomie de fonctionnement et la réduction de coût d'administration qui en découle. En effet, pour ajouter ou retirer une ressource d'un système pair-à-pair, il suffit simplement que cette ressource joigne ou quitte le réseau pair-à-pair associé au système. Le départ ou l'arrivée d'un noeud au sein d'un système pair-à-pair est géré de façon décentralisée par l'ensemble des noeuds, ce qui n'entraîne aucune intervention manuelle. Les applications pair-à-pair connaissent un grand succès dans certains domaines spécifiques tels que le partage de fichiers avec Bittorrent [50], la téléphonie via Internet avec Skype [35] ou encore le calcul scientifique avec la plateforme BOINC [25]. Cependant, la complexité de développement des protocoles pair-à-pair est telle qu'il n'existe aujourd'hui, à notre connaissance, aucun système pair-à-pair capable de fournir un service d'approvisionnement de ressources qui soit capable d'héberger plusieurs services indépendants.

Motivations et objectifs de la thèse

La majorité des offres d'infrastructure en tant que service se repose aujourd'hui sur l'utilisation de centres de traitement de données qui sont dédiés à cette tâche. Aussi, bien que le modèle de distribution et de consommation de ressources apporté par l'informatique dans le Nuage soit reconnu bénéfique par la majorité de la communauté informatique, les solutions proposées actuellement comportent un certain nombre de limitations qui peuvent être améliorées, notamment, par l'utilisation d'une structure pair-à-pair. Les motivations pour considérer

une organisation des ressources alternative au modèle proposé par les centres de traitement de données, dans le cadre de l'infrastructure en tant que service, sont les suivantes :

- Les coûts de construction et de maintenance des centres de traitement de données utilisés pour héberger les ressources matérielles de l'infrastructure en tant que service sont tels qu'on estime que ce type de solution n'est rentable qu'à partir d'une taille minimale de 60 000 serveurs [71, 29]. Bien qu'un tel type d'investissement permette de fournir une infrastructure fiable et hautement performante, la possibilité de fournir une infrastructure hébergée à moindre coût paraît être une piste intéressante pour le déploiement d'applications distribuées qui sont capables de se satisfaire de ressources non gérées. On trouve parmi ces applications : la diffusion de flux vidéo, la sauvegarde de données, la téléphonie via Internet ou encore le calcul distribué bénévole du type de celui implanté la par plateforme BOINC [25], laquelle est présentée au sein du Chapitre 2.

- En plus d'économiser les coûts de maintenance et d'administration de l'infrastructure en tant que service, l'utilisation d'un système pair-à-pair permet de gérer nativement une collection dynamique de noeuds composée de ressources hétérogènes et de provenances diverses. Cette caractéristique est particulièrement appréciée dans les scénarios de déploiement d'application sur ce que l'on appelle une fédération de nuages (c.-à-d. une collection de ressources appartenant à plusieurs entités administratives.). Ce type de déploiement est notamment motivé par la protection contre le verrouillage propriétaire, par la régulation économique de la distribution des ressources ou encore par la protection contre les événements catastrophiques.

- Les infrastructures distribuées en tant que service d'aujourd'hui ne sont pas capables de fournir un ensemble de ressources informatiques qui soit réparti géographiquement sur plus de trois ou quatre sites. Comme le montre l'étude de Church et coll. [48], cette limitation peut être pénalisante dans le cadre de certaines applications telles que la gestion de courrier électronique, la téléphonie via Internet, les jeux en lignes ou encore le déploiement d'applications sur les plateformes de calcul distribué de type MapReduce [56].

- La réalisation d'un service d'allocation de ressources pair-à-pair, tel que celui qui est proposé dans cette thèse, présente une opportunité pour la conception de services pair-à-pair composites, c'est-à-dire de services basés sur la collaboration de plusieurs sous-services implantés, eux aussi, par des protocoles pair-à-pair. Cette problématique a été initialement introduite par l'étude de Babaoglu et coll. [33], qui critique le fait que la plupart des protocoles pair-à-pair existants sont spécifiques à une application donnée et qu'il n'existe aucun service capable d'assigner des tâches à différents sous-ensembles de noeuds au sein d'un même système pair-à-pair.

En considérant les motivations énoncées ci-dessus, l'objectif du travail présenté dans cette thèse est de montrer qu'il est possible de fournir un service d'allocation de ressources informatiques s'appuyant sur une architecture pair-à-pair. Ce service doit être autonome, être implanté de façon totalement décentralisée et doit permettre d'allouer des noeuds non gérés de façon fiable, et ce, malgré la dynamique de l'environnement pair-à-pair hôte. Finalement, on accordera une attention particulière au passage à l'échelle du service en fonction du nombre de ressources qu'il peut accueillir, de la taille des collections de noeuds qu'il peut allouer et du nombre de requêtes d'allocations concurrentes qu'il peut gérer.

Contributions

La contribution proposée dans cette thèse est un service d'allocation de noeuds pair-à-pair s'inspirant du modèle de distribution de ressources de l'informatique dans le Nuage. Ce service est hébergé par une plateforme pair-à-pair appelée « Salute » qui est capable de collectionner un très grand nombre de ressources informatiques hétérogènes de provenances diverses. Le service d'allocation proposé est capable de sélectionner une sous-partie des ressources récoltées par la plateforme Salute et de les redistribuer sous forme de collection de noeuds de tailles multiples à un nombre potentiellement grand de clients. Dans cette thèse nous avons choisi d'articuler la description de ce service d'allocation de noeuds autour des contributions suivantes :

1. **La plateforme pair-à-pair Salute.** La conception de la plateforme pair-à-pair Salute utilise une nouvelle abstraction de collection de noeuds appelée « nuage pair-à-pair ». Un nuage pair-à-pair est spécifié principalement par le nombre de noeuds qu'il contient et par une durée de déploiement. C'est cette abstraction qui permet au service d'allocation de noeuds de Salute d'adopter le modèle de distribution de ressources de l'informatique dans le Nuage. Aussi la plateforme Salute est le seul système pair-à-pair, à notre connaissance, capable d'assigner un nombre potentiellement élevé de collections de noeuds à des applications ou à des services indépendants, de façon totalement décentralisée. Une des caractéristiques de cette plateforme est de posséder un certain nombre de ressources non allouables dites « de réserve », ce qui lui permet de maintenir la spécification des nuages qu'elle alloue au cours du temps, et ce, malgré le dynamisme crée par l'absence de contrôle sur la disponibilité des ressources provenant de l'environnement hôte.

2. **Le protocole de gestion de nuages pair-à-pair Sarcasm.** Sarcasm est un protocole de construction et de maintenance de nuages pair-à-pair basé sur l'utilisation de protocoles épidémiques. Dans cette thèse, nous montrons que Sarcasm est un service de gestion de nuages pair-à-pair fiable. En effet, ce protocole est capable de construire et de maintenir de nombreux nuages pair-à-pair malgré un degré de concurrence des requêtes potentiellement élevé et malgré le dynamisme des environnements pair-à-pair sur lesquels il est déployé. Ce protocole est totalement décentralisé et permet aux clients du service d'allocation de noeuds de la plateforme Salute de soumettre des requêtes d'allocations de nuages depuis n'importe quel noeud membre du système.

3. **Les contrôleurs d'admission et leurs politiques d'allocation de ressources.** Le service proposé par Salute permet d'utiliser un ensemble de ressources de provenance diverses, dont la disponibilité n'est pas contrôlée, afin de déployer des nuages pair-à-pair de taille stable pour une durée de déploiement prédéterminée. Le problème de prédire l'évolution du système de façon juste, afin de décider du nombre de nuages que l'ont peut maintenir au cours du temps, est géré par des contrôleurs d'admission. Comme tout service de Salute, ces contrôleurs fonctionnent de façon totalement décentralisée. Ils utilisent des techniques de prédiction de disponibilité afin d'évaluer si une requête d'allocation de noeuds soumise à la plateforme Salute peut être acceptée ou non. Plusieurs politiques d'allocations de noeuds, appliquées par les contrôleurs d'admission, sont étudiées dans cette thèse. Ces politiques d'allocation répondent au compromis suivant : faut-il allouer beaucoup de ressources ou garder beaucoup de noeuds en réserve afin de renforcer la garantie que les nuages alloués pourront être maintenu au cours du temps ? Intuitivement, la qualité du service d'allocation de noeuds de la plateforme Salute dépend directement des performances de ces politiques d'allocations.

Organisation du document

Le Chapitre 1 présente notre propre vue de l'informatique dans le Nuage. Le Chapitre 2 présente l'état de l'art des systèmes d'allocation de noeuds à large échelle, ainsi que les motivations qui nous ont poussé à concevoir un service d'allocation de noeuds pair-à-pair. Le Chapitre 3 présente une introduction aux systèmes pair-à-pair qui contient un rapide survol des protocoles existants. Ce chapitre insiste sur les protocoles épidémiques qui sont à la base de nos contributions. Le Chapitre 4 présente la plateforme pair-à-pair Salute hébergeant le service d'allocation de noeuds proposé dans cette thèse. Le Chapitre 5 décrit le protocole Sarcasm, responsable de la création et de la maintenance des nuages pair-à-pair. Le Chapitre 6 présente une étude des politiques d'allocations de noeuds les mieux adaptées aux environnements pair-à-pair. Finalement le Chapitre 7 présente les évaluations du service d'allocation de noeuds de la plateforme Salute.

Chapitre 1

Informatique dans le nuage

Sommaire

L'informatique dans le nuage est plus connue sous sa forme anglo-saxonne : « *Cloud Computing* », mais il existe de nombreux synonymes francophones tels que : « informatique dans *les* nuage*s* », « infonuagique » (Québec) ou encore « informatique dématérialisée ». C'est un domaine qui regroupe les technologies de distribution, à la demande et via Internet, de services informatiques logiciels et matériels. L'idée principale de ces technologies est de distribuer des ressources informatiques comme un service d'utilité publique, conformément à ce qui avait été imaginé par les pionniers de l'informatique moderne, il y a plus de 40 ans [129]. Ce principe de distribution publique de ressources informatiques anime également la communauté de la *grille informatique* [63], si bien qu'il est parfois difficile de distinguer la frontière entre « Grille » et « informatique dans le nuage » [69, 156, 127]. Cette difficulté est d'autant plus réelle que l'informatique dans les nuages est un concept jeune, dont les premières implantations datent de 2006, et dont le développement s'est accéléré durant ces dernières années.

Dans ce chapitre, nous proposons notre propre définition de « l'informatique dans le nuage », à partir d'une synthèse de définitions recueillies au sein de la littérature scientifique. La première section de ce chapitre présente cette synthèse, tout en détaillant le contexte technologique au sein duquel elle a été établie. La seconde section présente les différents types de

services qui sont proposés au sein de l'informatique dans le nuage, ainsi que les principaux modèles de déploiement associés. La troisième section présente le principe de fédération de nuages. Enfin, la dernière section conclut le chapitre en présentant une comparaison entre « grille informatique » et « informatique dans le nuage ».

Quelques citations :

> *« La chose intéressante à propos de l'informatique dans le nuage, c'est que nous l'avons redéfinie pour y inclure toutes les choses que nous faisons déjà... Je ne comprends pas ce que nous sommes censés faire de différent dans le contexte de l'informatique dans le nuage, hormis paraphraser certaines de nos publicités. »*
> Larry Ellison (PDG d'Oracle). *Wall Street Journal*, 26 Septembre 2008.

> *« Tout le monde s'empresse de monter dans le train du nuage, mais je n'ai pas entendu deux personnes dire la même chose à son sujet. Il existe de multiples définitions de ce fameux nuage. »*
> Andy Isherwood (Vice président du département commercial européen de Hewlett-Pacard). *Zdnet News*, 11 décembre 2008.

> *« Pire que de la bêtise : c'est un battage publicitaire abusif. On dit que l'informatique dans le nuage est inévitable et quand on entend quelqu'un dire ça, il est fort probable que cela vienne d'un ensemble de campagnes publicitaires, dont le but est de rendre cette nécessité réelle. »*
> Richard Stallman (fondateur de GNU et père spirituel du logiciel libre). *The Guardian*, 29 septembre 2008.

1.1 Concept de l'informatique dans le Nuage

La définition le plus souvent attribuée à l'informatique dans les nuages est sans doute celle du NIST : l'institut des standards et technologies des États-Unis [125]. Cette définition, corroborée par la plupart des grands acteurs de l'informatique dématérialisée [52], affirme que l'informatique dans les nuages est un paradigme en pleine évolution à l'heure d'aujourd'hui. C'est pourquoi la définition proposée dans la première partie de cette section est accompagnée d'un positionnement technologique présenté en seconde partie.

1.1.1 Vers une définition de l'informatique dans le nuage

Afin de comprendre les fondements de l'informatique dématérialisée, commençons par définir cette infrastructure qu'on appelle communément « le nuage ». De façon métaphorique le nuage en question est assimilé à Internet, ou plutôt à l'infrastructure matérielle et logicielle qui permet son fonctionnement [29]. Il est traditionnellement utilisé pour schématiser des réseaux de télécommunications, où il y représente la partie du réseau que l'on ne connaît pas, que l'on ne sait pas définir, mais qui permet d'interconnecter tous les réseaux déployés à travers le monde. L'exemple le plus courant est illustré par le schéma donné dans la Figure 1.1. Ce schéma représente un réseau informatique local (LAN), par exemple celui d'une petite entreprise. L'infrastructure gérée par l'entreprise en question (c.-à-d. ordinateurs, commutateurs et routeurs) y est détaillée, alors que celle associée à Internet est représentée par un nuage.

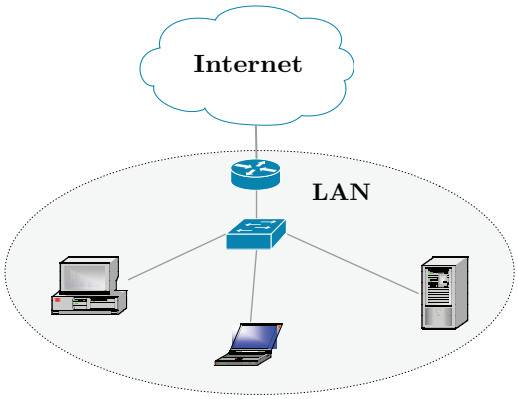


FIGURE 1.1 – Métaphore du nuage.

L'idée de cette représentation est d'identifier la partie du schéma ou de la solution à représenter qui est à la charge d'un tiers. Une question se pose alors : si cette partie du réseau est gérée par un tiers, alors pourquoi la détailler ? C'est l'idée soulevée par Velte et coll. dans le livre *« Cloud Computing, a practical approach »* [151] (une approche pratique de l'informatique dans le nuage), et c'est cette idée qui, probablement, correspond le mieux à la place de la métaphore du nuage au sein de l'informatique dématérialisée. Aussi l'informatique dématérialisée généralise ce concept de » partie du réseau que l'on n'administre pas, que l'on ne gère pas », à l'ensemble des ressources informatiques qui peuvent être accessibles via Internet en tant que service. Les détails techniques d'implantation de ces services, de leur hébergement ou de leur maintenance concernent alors uniquement le fournisseur, qui en contrepartie facture leur utilisation. Les services informatiques sont alors distribués publiquement, comme le serait l'eau, l'électricité, le gaz ou la téléphonie à travers leurs réseaux respectifs.

Ce modèle de distribution publique de l'informatique implique de déléguer la réalisation des services à des professionnels. Ainsi, leur expertise permet d'optimiser la gestion des ressources matérielles pour fournir des services hautement disponibles et d'une qualité digne des dernières avancées technologiques. En particulier, l'application de mécanismes évolués de virtualisation de ressources au sein des immenses centres de traitement de données modernes donne l'impression que le nuage dispose qu'une capacité infinie de ressources informatiques [29].L'élasticité d'allocation des ressources et le passage à l'échelle des services dépassent généralement les besoins des utilisateurs.

Les services fournis dans le cadre de l'informatique dématérialisée peuvent prendre de nombreuses formes, mais sont généralement catégorisés en trois sous-ensembles : les logiciels en tant que services, les plates-formes en tant que services et les infrastructures en tant que services. Quelque soit la catégorie de service demandée, l'accord entre le demandeur et le fournisseur se fait grâce à un contrat de niveau de service (ou SLA pour *« Software Level Agreement »*) dans lequel les conditions d'utilisation, ainsi que les limitations du service et son mode de facturation sont détaillés. La nature de ces contrats et services est étudiée plus en détail dans la Section 1.2.

L'informatique dans les nuages est un domaine vaste et dynamique, si bien qu'à l'heure d'aujourd'hui personne ne prétend en posséder une définition claire et absolue. En effet, la plupart des définitions attribuées à cette discipline semblent se concentrer seulement sur certains aspects de la technologie [29, 45, 67, 72, 115, 127], au point que Vaquero et coll. ont consacré un article entier [69] à sa définition globale. En guise de synthèse des différentes propositions données dans la littérature, nous proposons la définition suivante de l'informatique dans le nuage :

Définition de l'informatique dans le nuage :

> *L'informatique dans le nuage s'appuie sur une infrastructure (le nuage) composée d'un grand nombre de ressources virtualisées (par exemple : réseaux, serveurs, stockage, applications ou services), distribuées dans le monde entier. Ces ressources peuvent être allouées, puis relâchées rapidement, avec des efforts de gestion minimaux et avec peu d'interactions entre le client et le fournisseur. Aussi, cette infrastructure peut être dynamiquement reconfigurée pour s'ajuster à une charge de travail variable (passage à l'échelle). Finalement, les garanties de prestation offertes par l'informatique dans le Nuage prennent typiquement la forme de contrats de niveau de service.*

Cette définition est détaillée tout au long de ce chapitre, en commençant, dès le paragraphe suivant par expliquer son origine et le contexte technologique dans lequel l'informatique dans les nuages est apparue.

1.1.2 Contexte technologique

L'idée de l'informatique dans les nuages est apparue au courant de l'année 2006, avec le lancement du service de nuage extensible pour le calcul : « Amazon EC2 » [3]. Bien que ce dernier service soit aujourd'hui largement concurrencé, il demeure la référence de l'informatique dans le nuage en terme d'approvisionnement d'infrastructure. En particulier, un grand nombre de produits étiquetés « *Cloud Computing* » s'inspirent encore aujourd'hui des interfaces d'utilisation proposée par « Amazon EC2 ».

C'est le 25 août 2006 que la société Amazon, alors victime d'une spéculation excessive due à la bulle Internet, décide de rentabiliser son excédent de capacité de traitement de l'information. Elle donne un exemple commercial en proposant de louer ses ressources informatiques inutilisées sous la forme d'une gamme de services dits « extensibles ». C'est alors le lancement du premier service populaire explicitement rattaché à l'expression « Cloud Computing » : *EC2* (« *Elastic Compute Cloud* » [3]). Grâce à ces services, les clients d'Amazon EC2 peuvent louer des serveurs virtuels sur une base horaire et des capacités de stockage sur une base quotidienne. Les services sont de bonne qualité et l'idée plaît instantanément à un grand nombre d'entreprises qui y voient une source d'économie à différents niveaux (cf. Section 1.2.2). Les services proposés par Amazon ont aujourd'hui évolué et alimentent plusieurs dizaines de milliers de clients [2], dont les domaines principaux sont les suivants : commerce électronique, diffusion de contenu multimédia, hébergement de média, calcul haute performance, moteur de recherche, hébergement de site internet et plus généralement l'hébergement d'applications informatiques distribuées. En 2008, c'est-à-dire deux ans après l'apparition de l'informatique dans le nuage, la valeur de son marché s'est vue estimée à plus de 100 milliards de dollars par les spécialistes en investissement de la banque Merrill Lynch [131]. Finalement, d'après l'expertise de la société Gartner, spécialisée en investissement technologique, la mode de l'informatique dans le Nuage, en matière d'excitation médiatique, sera à son apogée aux alentours des années 2011 et 2010 [23].

L'informatique dans les nuages est un concept en phase de maturation, qui consiste à organiser la distribution publique et généralisée de ressources informatiques à travers le monde entier. Afin de mieux comprendre pourquoi les technologies relatives à l'informatique dans le nuage suscitent un fort engouement industriel et un fort intérêt économique, la section suivante détaille les différentes formes de services que l'ont peut trouver aujourd'hui au sein du Nuage.

1.2 L'informatique en tant que service

Traditionnellement, on distingue trois sous-ensembles de services au sein de l'informatique dans le nuage : le logiciel en tant que service (SaaS), la plateforme en tant que service (PaaS) et l'infrastructure en tant que service (IaaS). Chacun de ces types de service correspond à un niveau d'abstraction logiciel précis par rapport aux ressources informatiques matérielles accessibles via Internet, et donc hébergées au sein du nuage du point de vue de l'utilisateur. Il est commun de représenter l'ensemble de ces catégories de service sous la forme d'une pile, afin d'illustrer les différentes étapes potentiellement existantes au sein de la chaîne de production de services logiciels via le nuage. Cette pile est représentée sur la Figure 1.2, sa base, c.-à-d. son niveau inférieur, représente les ressources matérielles informatiques primaires, puis les trois étages intermédiaires représentent les trois types de services présentés dans cette section, et enfin l'étage supérieur représente les interactions entre les utilisateurs finaux des services proposés et le nuage.

C'est l'infrastructure en tant que service qui correspond au plus faible niveau d'abstraction que l'on peut obtenir par rapport aux ressources informatiques partagées via le nuage. En utilisant ce type de service, les utilisateurs peuvent directement administrer les ressources informatiques qu'ils consomment. Un exemple de service appartenant à cette catégorie est la location de serveurs virtuels proposée par Amazon EC2. Avec cette solution, les clients peuvent installer le système d'exploitation et les composants logiciels de leur choix au sein des espaces d'exécution virtualisés distribués par Amazon, comme ils le feraient sur une grappe de machines privée. A un niveau d'abstraction supérieur, on trouve la plateforme en tant

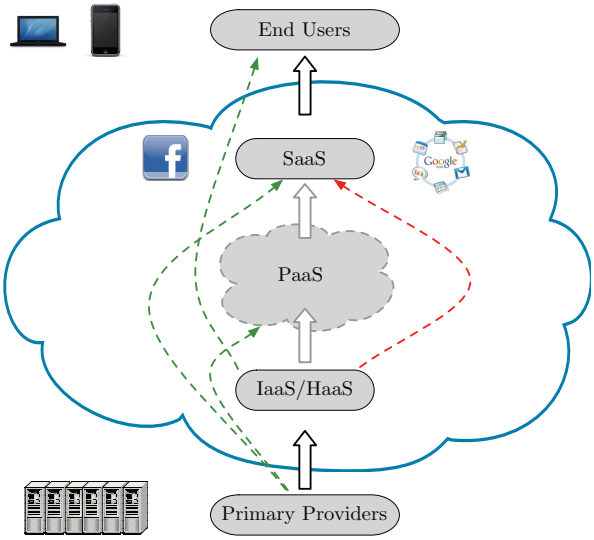


FIGURE 1.2 – Les services du nuage vus sous forme de pile.

que service qui étend la gamme de services proposés par l'infrastructure en tant que service avec des outils de développement d'applications Web qui sont totalement hébergés chez le fournisseur. Finalement, au plus haut niveau d'abstraction, on trouve les logiciels en tant que services, qui correspondent à des services logiciels classiques (bureautique, gestion de base de données simplifiée...etc.), dont la particularité est d'être hébergé au sein du nuage et non sur une infrastructure privée. A noter que ces différents types de services peuvent être combinés entre eux pour fournir des logiciels en tant que service. Leur utilisation est régie par l'existence de contrats de niveau de service. La section suivante détaille chacune de ces catégories de service, en terminant par la plateforme en tant que service. Une dernière partie présente également les différents modes de déploiement utilisés pour l'hébergement de ces services.

1.2.1 Logiciel en tant que service

Le logiciel en tant que service est un logiciel accessible à la demande, via Internet. Il est également connu sous l'appellation *« SaaS »*, dérivée de l'expression anglophone « Software as a Service ». Le logiciel en tant que service est un concept apparu au début du siècle [150]. Les logiciels en tant que service sont les services du nuage qui visent le plus grand nombre d'utilisateurs, car contrairement à l'infrastructure et à la plateforme en tant que service, leur utilisation ne demande aucune connaissance particulière en technologie de l'information et des télécommunications. Ces services sont accessibles via Internet, c'est-à-dire hébergés dans le nuage du point de vue de l'utilisateur, et sont généralement utilisables via un simple navigateur web. Une autre de leur particularité est d'être facturé par abonnement plutôt que par licence logicielle. Ils ont été initialement déployés pour automatiser les forces de ventes des entreprises, ainsi que la gestion de leur clientèle, comme ce fut le cas avec la solution Salesforce [18], considérée comme pionnier du logiciel en tant que service. Aujourd'hui, les logiciels en tant que services sont largement utilisés par les entreprises pour différentes tâches telles que la comptabilité, la facturation en ligne, la gestion de ressources humaines et les suites bureautiques de gestion de documents.

L'avantage des logiciels en tant que services est multiple pour les utilisateurs. En premier lieu, ils bénéficient d'un accès nomade et multiplateforme à leurs applications, grâce à l'hébergement dans le nuage et grâce à l'utilisation d'accès standardisés (accès via interface Web). Ensuite, la distribution de logiciel à la demande et le mode de facturation par abonnement permettent aux entreprises clientes d'assouplir leur mode d'investissement dans les technologies informatiques : ils peuvent dynamiquement adapter leur consommation logicielle en fonction de leur besoin. Finalement, ils jouissent d'une utilisation plus simple des services logiciels, sans avoir à se soucier de leur installation ou de leur mise à jour.

Il existe une catégorie particulière de service logiciel distribué via Internet qui n'est pas systématiquement considérée comme faisant partie du logiciel en tant que service, il s'agit des services gratuits à l'utilisation. Ces services sont indirectement financés par la publicité, ou par des produits dérivés de l'analyse statistique à grande échelle. Ils ciblent davantage les particuliers que les entreprises. Les exemples les plus courants sont les plateformes de réseautage social et les services de diffusion de média. Ces services sont déployés à grande échelle et typiquement hébergés sur plusieurs centres de traitement de données privés, comme c'est par exemple le cas pour Facebook [8] et Youtube [22]. Ces services sont des logiciels accessibles à la demande via Internet, nous prenons donc le parti dans cette thèse de classer ces « gratuiciels en tant que service » dans la catégorie des logiciels en tant que service.

La distribution du logiciel en tant que service est principalement freinée par deux problématiques : la dépendance technologique vis-à-vis du fournisseur et la confidentialité des données produites par les clients. Ces deux problèmes se généralisent à tous les types de services accessibles via le nuage. Ils sont davantage détaillés au sein de la Section 1.3.1. Cependant, il est important de comprendre que la confiance accordée au prestataire qui fournit le service logiciel est une composante importante du marché de la distribution logiciel, en particulier lorsque celle-ci est réalisée via le Nuage. Cette notion de confiance explique, en partie, la polarisation du marché vers un faible nombre de grands distributeurs tels que SalesForce [18] ou encore NetSuite [15].

En résumé, le logiciel en tant que service peut être gratuit ou payant, intégrer des notions de réseautage social ou encore de diffusion de média. Les services ainsi proposés, en particulier lorsqu'ils sont payants, sont régis par des contrats de niveau de service. Ces contrats définissent typiquement le dédommagement prévu pour les clients en cas d'indisponibilité du service vendu. Par exemple, les contrats de niveau de service fournis par Salesforce [18] prévoient de dédommager les clients sous forme de remise forfaitaire en cas d'indisponibilité du service. Le mode de facturation de ce service étant mensuel, lorsque l'abonnement d'un client arrive à échéance, les deux partis (le client et le prestataire) établissent un bilan au sein duquel le taux d'indisponibilité du service durant le moi écoulé est mesuré. Des remises forfaitaires, inscrites au contrat de niveau de service, sont alors appliquées en fonction de cette mesure.

1.2.2 Infrastructure en tant que service

L'infrastructure en tant que service est plus connue sous le nom d'*IaaS* pour « *Infrastrucure as a Service* ». Ce type de service consiste à distribuer des ressources informatiques telles que de la capacité de calcul, des moyens de stockage et de communication, de façon publique via Internet et sous une forme de paiement à l'utilisation. Les clients de l'infrastructure en tant que service peuvent donc exécuter et héberger leurs applications informatiques dans le nuage et ne paient que les ressources qu'ils consomment. Ces services d'approvisionnement en infrastructure peuvent servir, d'une part à héberger des logiciels ou plateformes en tant que services et, d'autre part, à être utilisés de façon plus générique en tant que ressources

informatiques pour des applications très variées, allant de la sauvegarde de données jusqu'au calcul haute performance, en passant par l'analyse statistique de données. La variété d'utilisation des services proposés par l'infrastructure en tant que service en fait la technologie de l'informatique dans le nuage la plus populaire. Aussi lorsque l'on parle d'informatique dans le nuage, on fait souvent référence à l'infrastructure en tant que service.

La particularité de l'infrastructure en tant que service est de fournir un approvisionnement en ressources informatiques de qualité, qui soit extensible sur commande et dont la capacité dépasse généralement la demande. Le nuage est alors vu par ses utilisateurs comme une source infinie de capacité de calcul, de stockage et de communication. Internet devient alors une place de marché où l'infrastructure informatique est distribuée, et consommée en tant que marchandise, selon le modèle illustré sur la Figure 1.3.

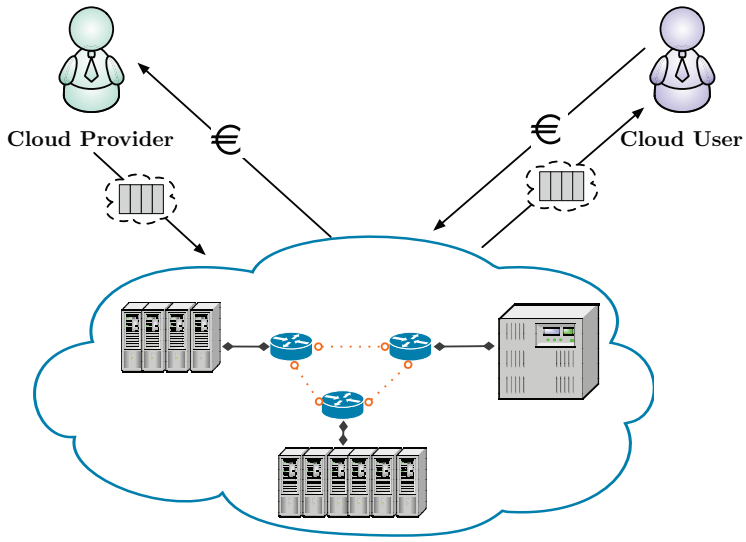


FIGURE 1.3 – Modèle de distribution de l'infrastructure en tant que service.

Ce modèle de distribution d'infrastructure assouplit le mode d'investissement en ressources matérielles et logicielles des industries, ouvrant des perspectives jusqu'alors inconnues pour les sociétés et organismes ayant besoin de disposer d'un accès direct à une infrastructure informatique. En effet, d'après les chercheurs de l'Université de Berkeley [29], les trois raisons majeures du succès de l'infrastructure en tant que service sont les suivantes :

1. *L'illusion d'une capacité de calcul infinie de la part des utilisateurs du nuage*, permettant aux clients de l'informatique dans les nuages de se reposer sur un unique service pour l'approvisionnement de ressources de calcul à long terme.

2. *L'assouplissement du mode d'investissement des utilisateurs du nuage*, permettant aux entreprises de commencer petit, puis d'augmenter les ressources informatiques matérielles seulement si le besoin s'en fait sentir.

3. *La possibilité de payer pour l'utilisation de ressources de calcul sur une base à court terme (c.-à-d. accès aux serveurs virtuels à l'heure)*, permettant une utilisation économe des capacités informatiques, en relâchant les ressources de calcul dès qu'elles deviennent inutilisées.

L'infrastructure en tant que service est fournie par des sociétés expertes en conception et gestion d'infrastructure informatique. Ces sociétés investissent dans des centres de traitement de données qui sont composés de plusieurs dizaines de milliers de machines. Ces centres sont construits de façon à optimiser les coûts de maintenance, d'alimentation, de refroidissement et d'administration de l'infrastructure matérielle [71]. Les ressources de calcul y sont distribuées sous forme de serveurs virtuels, souvent colocalisés sur une même machine physique. Le principe de virtualisation de serveurs utilisé dans les centres de traitement de données est illustré sur la Figure 1.4. Les mécanismes de virtualisation implantés au sein de ces centres

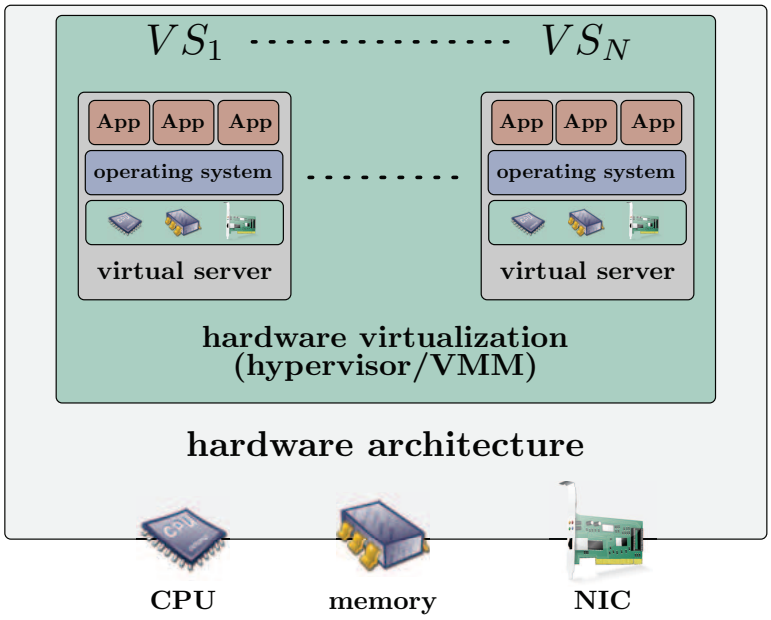


FIGURE 1.4 – Virtualisation de serveurs pour l'infrastructure en tant que service.

sont basés sur l'utilisation d'un moniteur de machine virtuel [34, 37, 30], également appelé hyperviseur, dont le but est de virtualiser l'accès aux ressources matérielles d'une machine physique (processeur, mémoire et autres périphériques). L'utilisation d'un hyperviseur permet de déployer plusieurs systèmes d'exploitation, exécutés de façon concurrente sur un même serveur. On parle alors de serveurs virtuels ou d'espaces virtuels d'exécution. En plus de permettre la colocalisation de plusieurs environnements d'exécution, l'intérêt d'un hyperviseur est de pouvoir ajouter ou retirer dynamiquement des instances de serveurs virtuels sur un même serveur physique. C'est cette propriété qui, à l'échelle d'un centre de traitement de données, permet de fournir un service d'approvisionnement en ressources de calcul qui est dynamique, extensible et qui possède de bonnes propriétés de passage à l'échelle.

La distribution de serveur virtuel se fait via un service d'allocation, généralement accessible depuis une interface web, qui coordonne l'instanciation de serveurs virtuels au sein du centre de traitement de données et leur allocation aux clients. Une fois qu'un ou plusieurs serveurs virtuels leur sont attribués, les utilisateurs peuvent y déployer des systèmes d'exploitation personnalisés, ou préconfigurés par les prestataires. Ces serveurs virtuels disposent d'une interface de connexion sécurisée à travers laquelle les clients de l'infrastructure peuvent administrer le système d'exploitation et y installer les applications de leur choix. Les serveurs virtuels disposent également d'une interface d'accès publique via Internet qui peut être utilisée par les applications qu'ils hébergent pour fournir des services publics, par exemple des logiciels

en tant que service. Á noter que les centres de traitements de données sont reliés entre eux et au reste d'Internet via des moyens de communication hautement performants, permettant aux applications qu'ils hébergent de bénéficier des meilleurs débits et temps d'accès disponibles à travers le monde entier.

Les services d'approvisionnement en capacité de stockage, qui sont proposés au sein de l'infrastructure en tant que service, sont généralement des services indépendants de l'approvisionnement en ressources de calcul. Ils sont accessibles via une interface publique (c.-à-d. accessible depuis Internet) qui prend généralement la forme d'une interface de base de données classique, comme c'est le cas pour le service S3 d'Amazon [66]. L'exécution des services de stockage est généralement effectuée sur un ensemble de machines dédiées au sein des centres de traitement de données. A noter que l'infrastructure matérielle exploitée par les prestataires de service est généralement optimisée pour que les applications déployées sur les serveurs virtuels puissent interagir de façon intensive avec les services de stockage de données, en particulier à travers des liens de communications dédiés, internes aux centres de traitement de données.

Les contrats de niveau de service utilisés au sein de l'infrastructure en tant que service définissent la tarification de l'utilisation des ressources. La location de serveur virtuel est par exemple facturée par heure d'utilisation. Le taux de disponibilité et le temps de reprise après panne de l'infrastructure sont également mentionnés au sein de ces contrats. A titre d'exemple, les contrats de service régissant l'utilisation d'EC2 [3] indiquent que le taux d'indisponibilité des services est de 99,95% par an. Des dédommagements forfaitaires, spécifiés dans les contrats, sont alors appliqués si le taux de disponibilité des services est inférieur à cette valeur. En ce qui concerne la performance des ressources informatiques distribuées, les fournisseurs d'infrastructure utilisent généralement leurs propres mesures. Par exemple, la gamme de service EC2 [3] définit plusieurs types d'instances dont le prix est proportionnel à la capacité de calcul et dont les performances sont appréciées empiriquement par les clients. En effet, Amazon conseille de faire des tests de déploiement sur plusieurs types d'instances afin d'évaluer les besoins de ressources d'une application. Ce même principe est généralement appliqué en ce qui concerne les performances de stockage et de communication de l'infrastructure en tant que service.

Pour conclure, l'infrastructure en tant que service est une technologie permettant l'approvisionnement élastique, dynamique et économique de tout type de ressources informatiques. Ces ressources peuvent être administrées de façon à ce que les clients de l'infrastructure en tant que service puissent y déployer les applications de leur choix. Finalement, il est important de noter l'influence da la gamme de services EC2 d'Amazon [3]. En effet, la majorité des fournisseurs en infrastructure proposent la même interface d'utilisation que celle initialement définie par EC2. C'est notamment le cas des gestionnaires d'infrastructure libres Nimbus [109, 92] et Eucalyptus [126], qui proposent des suites logicielles à déployer sur un ensemble de machines, afin d'y exécuter un service d'approvisionnement en infrastructure semblable à celui proposé par la société Amazon. Avant de présenter les différents modèles de déploiement permettant l'hébergement de ces services, la suite du document décrit la plateforme en tant que service.

1.2.3 Plateforme en tant que service

La plateforme en tant que service, plus connue sous l'appellation anglophone « Platform as a service » (PaaS) se trouve à mi-chemin entre le logiciel en tant que service et l'infrastructure en tant que service. Aussi, il est difficile de définir les frontières entre plateforme et infrastructure, aussi bien qu'entre plateforme et logiciel en tant que service [29]. Cependant, nous proposons de caractériser ce type de service comme suit : la plateforme en tant

que service offre un environnement de développement et de déploiement pour les logiciels en tant que service, qui est accessible et hébergé au sein du nuage. Les services offerts par cette technologie facilitent le déploiement des applications (souvent déployées comme logiciels en tant que service), en abstrayant à ses utilisateurs les coûts et la complexité de maintenance de l'infrastructure sous-jacente, et ce, pendant l'intégralité du cycle de vie des applications. Ainsi, ces services sont généralement adressés à des développeurs de logiciels souhaitant utiliser une même plateforme pour les cycles de développement et de déploiement de leurs applications.

Les plateformes en tant que services incluent généralement des services d'aide au développement tels que des applications de conceptions, de versionnement, de test, d'intégration de service Web ou de base de données... etc. Elles incluent également des services d'aide au déploiement tels que l'hébergement d'application, la surveillance des applications, le stockage de données, l'allocation dynamique de ressources, la gestion de la persistance des données...etc. Enfin, les services proposés par une plateforme en tant que service sont généralement délivrés sous la forme d'une solution intégrée accessible via des interfaces Web publiques. On peut distinguer deux types principaux de plateformes en tant que services : les plateformes de développement d'extension et les plateformes de développement d'applications autonomes.

Les plateformes de développement d'extension sont généralement mises à disposition par les grands éditeurs de logiciels en tant que services, dans le but de permettre à leurs utilisateurs d'ajouter des fonctions personnalisées aux services classiques. Les plateformes de développement d'extension les plus connues sont proposées par de grands éditeurs de logiciels en tant que services tels que Salesforce [18] et Netsuite [15]. Par exemple, SalesForce propose des outils de création assistée de base de données ou encore de personnalisation d'interface graphique qui sont conçus pour que les utilisateurs puissent personnaliser leur environnement de travail au sein de la plateforme en tant que service.

Les plateformes de développement d'applications autonomes sont, dans leur principe, plus proches de l'infrastructure en tant que service. Elles proposent généralement, en plus de la distribution d'infrastructure, des services de développement permettant de se reposer sur le Nuage pour l'intégralité du cycle de vie des applications clientes. C'est le cas, par exemple, du service Google Code [11] qui prend en charge l'hébergement du code d'une application ainsi que son versionnement. Les plateformes en tant que service favorisent généralement l'utilisation d'une technologie propriétaire ou de services ciblés, comme c'est le cas pour les offres Windows Azure [20] ou Google App Engine [9]. Par exemple, la plateforme Google App Engine met à disposition de ses clients des interfaces de programmation qui facilitent l'intégration des services logiciels distribués par Google, comme la gestion des comptes utilisateurs ou encore la gestion de partage des documents. La plateforme Windows Azure, elle, favorise l'utilisation des technologies et services appartenant à Microsoft, comme l'environnement de développement « .Net » ou la gamme de services « Live ».

Finalement, les contrats de niveau de service proposés par ces plateformes définissent également le taux de disponibilité des applications qu'ils hébergent. A titre d'exemples, Google App Engine et Windows Azure garantissent un taux de disponibilité de 99,9% mesurable sur une période d'un mois. Comme pour le logiciel ou pour l'infrastructure en tant que service, une remise forfaitaire est appliquée en guise de compensation si jamais le service ne respecte par les accords de disponibilité.

1.2.4 Modèles de déploiement

D'après la définition donnée dans la Section 1.1.1, un nuage correspond à une infrastructure distante, dont on ne connaît pas les détails architecturaux, et qui est connue pour les services

informatiques qu'elle offre. Aussi, il est courant d'utiliser le terme **un nuage** pour désigner l'infrastructure gérée par un prestataire donné. On pourra alors parler du nuage d'Amazon, de celui de Google, et ainsi de suite. On peut distinguer quatre types principaux de modèles de déploiement pour ces nuages : le nuage privé, le nuage communautaire, le nuage public et le nuage hybride.

Le nuage privé : l'infrastructure d'un nuage privé n'est utilisée que par un unique client. Elle peut être gérée par ce client ou par un prestataire de service et peut être située dans les locaux de l'entreprise cliente ou bien chez le prestataire, le cas échéant. L'utilisation d'un nuage privé permet de garantir, par exemple, que les ressources matérielles allouées ne seront jamais partagées par deux clients différents.

Le nuage communautaire : l'infrastructure d'un nuage communautaire est partagée par plusieurs organisations indépendantes et est utilisée par une communauté qui est organisée autour des mêmes besoins, vis-à-vis de son utilisation. Par exemple, dans le projet Open Cirrus [31], le nuage communautaire est partagé par plusieurs universités dans le cadre d'un projet scientifique commun. Son infrastructure peut être gérée par les organisations de la communauté qui l'utilise ou par un tiers et peut être située, soit au sein des dites organisations, soit chez un prestataire de service.

Le nuage public : l'infrastructure d'un nuage public est accessible publiquement ou pour un large groupe industriel. Son propriétaire est une entreprise qui vend de l'informatique en tant que service.

Le nuage hybride : l'infrastructure d'un nuage hybride est une composition de deux ou trois des types de nuages précédemment cités. Les différents nuages qui la composent restent des entités indépendantes à part entière, mais sont reliés par des standards ou par des technologies propriétaires qui permettent la portabilité des applications déployées sur les différents nuages. Une utilisation type de nuage hybride est la répartition de charge entre plusieurs nuages pendant les pics du taux d'utilisation.

Pour conclure, l'informatique dans les nuages permet la distribution de logiciel, d'infrastructure informatique et de plateforme de développement, à travers Internet. Il existe plusieurs modèles de déploiement de nuage, qui diffèrent par l'exclusivité de leur utilisation et par l'organisation de leur infrastructure. Dans la suite de ce document, on s'intéressera plus particulièrement au cas du modèle de déploiement de nuage hybride, dans le cadre de ce que l'on appelle « la fédération de nuages ».

1.3 Vers la fédération de nuages ou « Intercloud »

Comme expliqué précédemment dans ce chapitre, un nuage correspond à une infrastructure et à son domaine d'administration. De façon plus simple, il est courant d'associer un nuage à l'entreprise qui est responsable de la gestion de l'infrastructure associée. On parlera alors du nuage d'Amazon, de celui de Google, du nuage de Microsoft...etc. Cependant, du point de vue d'un utilisateur, cet ensemble de nuages accessibles publiquement via Internet peut être vu comme un méta-nuage, au sein duquel un certain nombre de services et de ressources informatiques sont disponibles. De la même façon qu'Internet est le réseau des réseaux. Ce

méta-nuage est le nuage des nuages et on l'appelle « Intercloud », son principe est illustré sur la Figure 1.6.

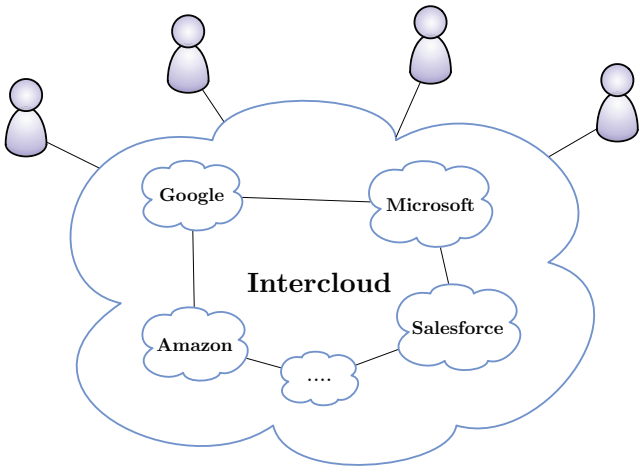


FIGURE 1.5 – Intercloud : le nuage des nuages.

Intercloud est un ensemble de services et de ressources informatiques qui sont publiquement disponibles via Internet. Comme décrit lors de la section précédente, on retrouve trois grandes catégories de services : le logiciel, la plateforme et l'infrastructure en tant que service. La section qui suit se focalisera sur l'infrastructure en tant que service. Plus précisément, cette section introduit l'idée du déploiement d'applications distribuées sur une fédération de nuages. Ce modèle de déploiement correspond au nuage hybride présenté plus tôt. Il a pour particularité d'utiliser plusieurs nuages au sein d'un même environnement de déploiement. Une première partie donne les motivations qui poussent les clients de l'informatique dématérialisée à vouloir déployer leurs applications sur une fédération de nuages, plutôt que sur une seule et même infrastructure. Une seconde partie présente les solutions techniques, les intergiciels pour nuages, qui rendent possible ce type de déploiement. Finalement, une troisième et dernière partie discute de l'intérêt d'une approche orientée marché au sein de l'architecture de l'Intercloud.

1.3.1 Motivations pour un déploiement multi-nuages

Cette section décrit les motivations qui poussent les clients de l'informatique dans le nuage à déployer leurs applications chez plusieurs fournisseurs d'infrastructure à la fois. Les trois principales raisons qui motivent un tel déploiement, à savoir : le verrouillage propriétaire, la confidentialité des données et la régulation économique de la distribution de ressource, sont développées ci-dessous.

Le verrouillage propriétaire : D'après un sondage de l'entreprise « Rightscale », leader sur le marché du logiciel intermédiaire de gestion d'infrastructure en tant que service (Cf. Section 1.3.2) : la plupart des entreprises refuseraient d'externaliser leur infrastructure informatique dans le nuage, de peur d'être victime de verrouillage propriétaire [78]. Le verrouillage propriétaire est un phénomène économique connu, qui apparaît lorsque l'activité d'une entreprise devient totalement dépendante des services d'un seul et même prestataire. Ses conséquences pour une entreprise cliente du nuage sont doubles : *i)* si le fournisseur vient

à disparaître où décide de suspendre ses services, alors elle doit re-concevoir toute sa gestion d'infrastructure, afin de l'adapter aux services d'un nouveau prestataire, et *ii)* une relation dominant-dominé s'installe à la faveur du fournisseur, qui est en bonne position pour imposer des choix technologiques à ses clients, par exemple : l'utilisation d'une technologie de virtualisation propriétaire. Aussi, les clients de l'infrastructure en tant que service ont intérêt à diversifier leur source d'approvisionnement autant que possible, afin de se protéger contre le verrouillage propriétaire et de préserver une plus grande liberté de choix technologique pour le développement de leurs applications.

Confidentialité des données : Un autre obstacle à l'adoption de l'informatique dans les nuages est le maintien de la confidentialité des données. En effet, toutes les données utilisées ou générées par les applications Web qui sont hébergées dans le nuage sont entièrement confiées aux prestataires de l'infrastructure en tant que service. Outre la confiance qu'un tel hébergement distant demande entre le client et le fournisseur d'infrastructure, un tel modèle de déploiement impose également une confiance totale dans les mécanismes de virtualisation utilisés pour la colocalisation de serveurs virtuels (cf. Section 1.2.2). Or, comme l'a montré Ristenpart et coll. [135], il est possible de localiser, puis d'observer l'activité d'une application, au sein d'un centre de traitement de données qui utilise ces techniques de virtualisation. Les informations ainsi observées peuvent ensuite servir à corrompre, en partie, la confidentialité de certaines données sensibles, y compris des informations chiffrées [135].

Afin de préserver la confidentialité des données hébergées au sein du nuage, Owens propose d'utiliser des solutions basées sur la cryptographie [128]. Cependant, ces solutions présentent encore de nombreux défis d'implantation et ne sont pas disponibles dans les infrastructures d'aujourd'hui. Une solution plus simple consiste à déployer des applications composites sur un nuage hybride. Les données sensibles sont alors stockées sur un nuage privé hautement sécurisé, et une partie des composants de l'application sont déployés sur des nuages publics. Bien que cette solution fonctionne pour certains types d'applications, elle nécessite d'une part de disposer d'un nuage privé correctement sécurisé, et d'autre part de disposer d'un système de gestion d'applications capable de déployer une application composite sur plusieurs nuages en même temps.

Régulation économique de la distribution de ressource : Les utilisateurs de l'informatique dans le nuage se reposent sur les fournisseurs d'infrastructure pour l'approvisionnement de la majeure partie de leurs ressources informatiques. Aussi ils sont susceptibles de demander des conditions d'utilisation et de qualité de service variées, en fonction de leurs besoins. Devant la variété des offres présentes sur le marché, ces clients ont tout intérêt à choisir un panel de services à la carte au sein des divers prestataires. Un exemple simple consiste à utiliser, au sein d'une même entreprise, un nuage fournissant des machines puissantes et chères, pour exécuter des calculs haute performance et un autre nuage dont les ressources sont moins chères et moins performantes, pour exécuter des calculs secondaires. Aussi, afin de répondre à la variété des demandes, les fournisseurs d'infrastructure en tant que service ont besoin de délivrer des offres diversifiées, c.-à-d. qui correspondent à différents contrats de niveau de service. Bien que certains grands fournisseurs d'infrastructure, comme Amazon, commencent à proposer plusieurs gammes de services, la variété des besoins en approvisionnement de ressources conduit inévitablement à la création d'un marché hautement concurrentiel. Aussi, les études préliminaires inspirées du milieu de la Grille informatique [44, 143] ont montré qu'une gestion de ressource « orientée marché » était nécessaire à la régulation de l'offre et de la demande. Cette régulation permet d'atteindre un équilibre du marché de la distribution de

ressources de calculs, ce qui est bénéfique pour les clients comme pour les prestataires. Un exemple d'infrastructure orientée marché pour l'informatique dans le nuage est présenté dans la Section 1.3.3.

Bien que non détaillées dans ce document, il existe d'autres motivations poussant les clients du nuage à déployer leurs applications chez de multiples fournisseurs. Parmi ces raisons, on trouve le besoin de distribution géographique relatif à des applications spécifiques, telles que la lutte contre la propagation des messages indésirables [48] ou encore le recouvrement d'infrastructure après des événements catastrophiques [147]. Pour toutes les raisons évoquées au sein de cette section, le déploiement d'applications sur une fédération de nuages, c'est-à-dire chez plusieurs fournisseurs d'infrastructure, a connu un véritable engouement ces dernières années, notamment à travers la conception d'intergiciels pour nuages.

1.3.2 Intergiciels pour nuages

L'absence de standard au sein des services d'approvisionnement en infrastructure rend la migration d'application entre les différentes structures très difficile et souvent non rentable pour les clients. En effet, lorsqu'une entreprise souhaite déployer une application distribuée sur plusieurs nuages, elle se confronte à un certain nombre de problèmes non triviaux tels que : la synchronisation des déploiements, la surveillance de l'application, la gestion d'un système de fichiers commun, la gestion des comptes permettant l'accès aux différentes infrastructures...etc.

L'intergiciel [1], dans le contexte des systèmes distribués, a été initialement décrit par Bernstein [39] comme étant un ensemble d'intermédiaires pour les différents composants logiciels d'un système informatique réparti. Ce concept a été intensivement utilisé avec la percée de l'architecture orientée service (SOA), où les services en question étaient distribués grâce à des intergiciels. Plus généralement, un intergiciel est utilisé pour abstraire les différences entre plusieurs systèmes hétérogènes et exposer une interface uniforme. C'est précisément le but des intergiciels pour nuages : abstraire les différentes interfaces et les différents types de contrats de niveaux de services et fournir une interface uniforme pour la distribution de ressources informatiques. Cette distribution uniforme de ressources est motivée par les raisons exposées précédemment (cf. Section 1.3.1) et se traduit techniquement par la réalisation d'un service intergiciel vu comme un « méta-nuage ». Ce méta-nuage est une abstraction de l'ensemble des services d'approvisionnement en infrastructure présents dans l'Intercloud. Le service intergiciel fournit, au minimum, les mêmes services que ceux proposés par les nuages classiques, à savoir : des services d'allocations élastiques de serveurs virtuels et des services de stockage de données. Un exemple de déploiement d'application sur une fédération de nuages est donné sur la Figure 1.6.

Cette illustration représente le déploiement de deux applications utilisant un intergiciel pour nuages. Les gérants des applications, ainsi que leurs utilisateurs ne voient que le nuage dans son ensemble (l'Intercloud) à travers les interfaces de l'intergiciel, alors que les applications sont déployées sur différentes infrastructures. En particulier, l'application dénotée « App 2 » est déployée sur plusieurs nuages, de façon transparente pour les personnes qui l'administrent.

Il existe plusieurs types d'intergiciels pour nuages qui sont différenciés par le niveaux de détails de leurs interfaces, par le type de nuages sur lesquels ils sont déployés, et par les méthodes de gestion de contrats utilisées.

1. Le mot intergiciel est utilisé comme traduction de « middleware » dans ce document.

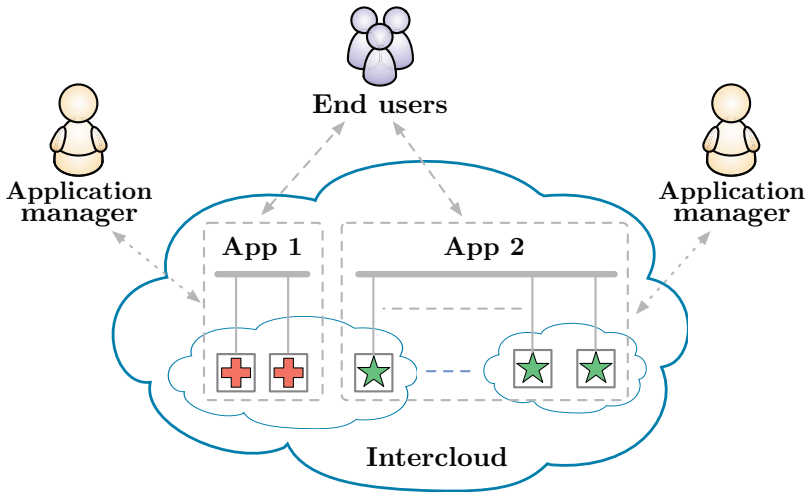


FIGURE 1.6 – Applications déployées via un intergiciel pour nuages.

Intergiciels d'agrégation de structures : Ce type d'intergiciel a pour but d'uniformiser une structure hébergée sur plusieurs nuages, de façon à la présenter à l'utilisateur comme une seule et même structure de déploiement. C'est l'approche présentée par Keahey et coll. dans leurs travaux sur « l'informatique dans le ciel » [92] (Sky Computing). Ces travaux présentent une extension du gestionnaire d'infrastructure Nimbus [91], qui utilise un réseau virtuel déployé sur toutes les structures à agréger afin de présenter l'ensemble des machines des différents nuages comme faisant partie d'un nuage virtuel unique. L'intergiciel a besoin de dédier un noeud (c.-à-d. un espace virtuel d'exécution) par nuage pour la gestion du réseau virtuel. Les auteurs montrent que l'exécution d'applications utilisant la plateforme de calcul distribué « Mapreduce » [56] est aussi performante lorsque les applications sont distribuées sur plusieurs nuages via l'intergiciel, que lorsqu'elles sont exécutées sur un seul nuage via un gestionnaire de ressources classique [92].

Intergiciels d'uniformisation d'interfaces Une seconde catégorie d'intergiciel pour les nuages est celle des solutions d'uniformisation d'interfaces de nuages. Ces intergiciels sont des systèmes qui tiennent le rôle d'intermédiaires durant les interactions entre clients et fournisseurs de nuages. Leur but est uniquement de fournir une interface uniforme pour les utilisateurs de nuages en interceptant et en traduisant toutes les communications entre les différents nuages et l'utilisateur. Cette uniformisation permet aux clients de concevoir leurs applications autour d'une interface uniforme, tout en gardant le choix du fournisseur d'infrastructure. Un exemple d'un tel intergiciel est le système Altocumulus d'IBM [112, 113], dont les trois principaux outils sont : un tableau de bord comportant les interfaces uniformes d'interaction avec les nuages, un répertoire d'images systèmes à déployer et une interface d'utilisation simplifiée visant à tester l'intergiciel dans le cadre de tutoriels. Notons qu'il existe également des travaux vers l'uniformisation des interfaces des nuages, qui sont concentrés sur la standardisation des formats de données et des protocoles de communications inter-nuages [38]. Des études académiques sont également en cours, concernant l'implantation d'intergiciel d'uniformisation d'interfaces pour l'utilisation du nuage au sein de la recherche scientifique. Le projet Open Cirrus [31] est un exemple d'une telle étude.

Intergiciels pour déploiement autonome : Ce type d'intergiciel fait l'hypothèse que les applications visant à être hébergées dans les nuages sont décrites comme des applications composites, dont chaque composant est déployé dans un espace virtuel d'exécution différent (serveur virtuel). Chaque application est alors associée à une description, au sein de laquelle sont décrites les spécifications de déploiement et d'interconnexion de chaque composant. L'intergiciel agit alors comme un gestionnaire autonome, capable de déployer et de maintenir une application sur plusieurs nuages. Par exemple, l'approche proposée par IBM dans le cadre du projet Reservoir [136] consiste à enrichir la description des applications composites en y incluant des politiques de reconfiguration, qui sont spécifiques à l'environnement de l'informatique dans les nuages. Il est alors possible de définir des politiques d'accords concernant les contrats de niveaux de services ainsi que des politiques de passage à l'échelle des applications [136]. L'intergiciel du projet Reservoir est capable, entre autres, d'utiliser ces politiques pour surveiller la charge de travail des applications déployées et réajuster leur consommation de ressources de façon autonome.

Intermédiaires et intergiciels propriétaires Devant le succès de l'informatique dans les nuages, un grand nombre d'entreprises proposent aujourd'hui des services de courtage ou de médiation entre les fournisseurs de ressources informatiques et les clients du nuage. Ces entreprises fournissent diverses valeurs ajoutées, comme la simplification de déploiement, par exemple dans les cas d'Elastra [7] ou de Heroku [13], ou encore en fournissant un intergiciel propriétaire d'uniformisation d'interfaces, comme c'est le cas pour RightScale [17]. RightScale est sans doute la plus populaire des entreprises intermédiaires, elle propose aujourd'hui des solutions de déploiement simples grâce à un intergiciel capable d'uniformiser l'accès aux ressources de deux nuages publics (Amazon AWS [4] et RackSpace [16]), d'un nuage communautaire (Eucalyptus [126]) et éventuellement d'un nuage privé propre au client. En plus de la valeur ajoutée par rapport aux services classiques de l'infrastructure en tant que service, ces entreprises profitent souvent de leur position d'expert et d'intermédiaire pour réaliser une spéculation sur l'utilisation des ressources informatiques. En effet, en redéfinissant leurs propres contrats de niveaux de services au dessus de ceux proposés par les fournisseurs classiques d'infrastructure, ces entreprises intermédiaires parviennent à virtualiser davantage les ressources informatiques du nuage en surveillant la consommation des applications qu'elles hébergent à grain fin. Elles réalisent ainsi une spéculation sur la charge de travail des applications hôtes, ce qui leur permet de potentiellement maximiser le profit des services proposés en tant qu'intermédiaire. On parle alors de courtage en ressources informatiques.

En résumé, il existe plusieurs types d'intergiciel qui permettent à leurs utilisateurs de déployer des applications distribuées sur une fédération de nuages. Ce nouveau type de déploiement fait appel à de nouvelles notions d'économies, comme la gestion autonome de consommation de ressources pour le passage à l'échelle automatique des applications. Ces nouvelles questions sont l'objet de l'étude appelée « le nuage orienté marché ».

1.3.3 Vers un nuage orienté marché

Afin de maximiser le profit apporté par la distribution de ressources informatiques, pour les clients comme pour les fournisseurs, il est nécessaire de prendre en compte l'aspect de marché concurrentiel dans le développement des technologies du nuage. Si on admet que le déploiement d'applications sur une fédération de nuage est techniquement possible, grâce aux intergiciels pour le nuage, la technologie actuelle ne permet pas de gérer l'aspect marchand

d'un tel déploiement. En particulier, il n'existe aucun service, au sein de l'informatique dans les nuages d'aujourd'hui, capable d'établir un contrat entre un client et un fournisseur d'infrastructure, de façon autonome. En conséquence, des mécanismes de gestion automatisée de l'aspect marchand de la distribution de ressources informatiques dans le nuage doivent être développés, afin de rendre possible le déploiement d'applications distribuées sur une fédération de nuages. Dans le cas contraire, la gestion des contrats de niveau de service devrait être manuelle et le nombre de nuages utilisés pour le déploiement d'une même application en serait restreint. C'est cette constatation qui a poussé Buyya et coll., en 2009, à proposer une architecture orientée marché pour l'organisation des services de l'informatique dans le nuage [45]. Cette architecture est décrite sur la Figure 1.7.

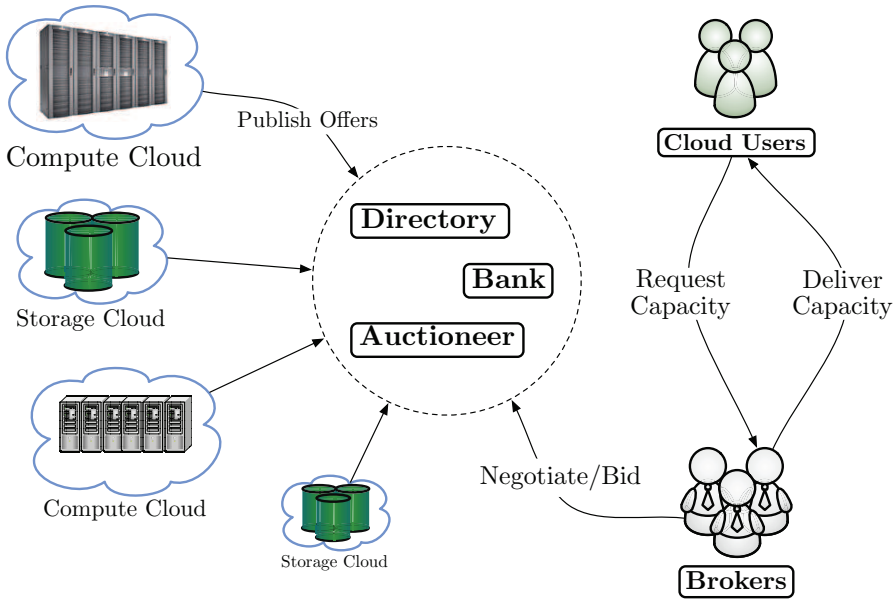


FIGURE 1.7 – Architecture pour un nuage orienté marché.

Le point le plus important, au sein de cette architecture, est la présence d'un nouvel agent intermédiaire : le courtier. Le rôle du courtier est de gérer la correspondance entre les contrats de niveau de service proposés par les fournisseurs et ceux demandés par les clients. Typiquement, un courtier va pouvoir acheter une grande quantité de ressources, puis revendre cette capacité de calcul à de nombreux clients en petite quantité. Ce système d'intermédiaire est avantageux pour les trois partis : les clients n'interagissent qu'avec un seul prestataire et ne paient qu'une seule facture, les fournisseurs d'infrastructure trouvent des clients plus facilement, et les courtiers peuvent réaliser des bénéfices en spéculant sur la revente des ressources informatiques.

Ce type d'architecture comporte une structure de marché composée d'un ensemble de banques, de registres de publications d'offres et de systèmes d'enchères automatisés. Les fournisseurs d'infrastructure publient alors leurs offres de services au sein du marché, dans les registres appropriés. Les courtiers surveillent ces offres et utilisent les systèmes d'enchères, puis les banques afin d'acheter les ressources informatiques correspondantes. D'après Buyya et coll., l'avantage d'une telle infrastructure est de profiter des travaux existants, tels que SHARP [65], Tycoon [100], ou encore Shirako [79]. Ces travaux, initialement conçus pour être utilisés dans le cadre de la grille informatique, peuvent être adapté à moindre coût, pour

produire l'infrastructure de nuage orienté marché proposée. En effet, matures pour la plupart, ils sont capables de gérer la correspondance entre les différents types de contrats de niveau de service, et ce, de façon automatisée. L'architecture présentée ci-dessus est décrite plus en détail dans l'article de Buyya et coll. [45].

Pour conclure, il existe de nombreuses motivations, telles que le verrouillage propriétaire ou le besoin de répartition géographique, pour déployer des applications distribuées sur une fédération de nuages. Un tel déploiement nécessite l'aide d'un intergiciel de gestion de déploiement, qui a pour but de fournir une interface d'accès et de contrôle uniforme au dessus d'une fédération de nuages. Bien que de nombreuses solutions existent aujourd'hui, le déploiement d'intergiciels pour nuages en est encore à un stade expérimental. En particulier, une structure de nuage orientée marché est nécessaire pour que ces intergiciels puissent être déployés plus largement et être utilisés par la majorité des clients de l'informatique dans le nuage. Cette structure de marché serait alors en charge d'enregistrer toutes les offres des fournisseurs d'infrastructure en tant que service, ainsi que les contrats de niveau de service associés, puis de les faire correspondre aux demandes des clients, et enfin de traiter les transactions liées à la distribution de ressources informatiques.

Conclusion L'informatique dans les nuages est un paradigme qui offre un nouveau modèle de distribution et de consommation de ressources informatiques à grande échelle. Les technologies associées à cette discipline permettent aux propriétaires de grands centres de traitement de données de louer les ressources inutilisées dont ils disposent, et de ce fait d'augmenter la rentabilité de leur investissement matériel. Les clients de l'informatique dans le nuage bénéficient également de ce modèle de distribution de ressources, car il leur permet d'assouplir leur mode d'investissement en ressources informatiques, par exemple en ajustant la capacité de traitement de leur infrastructure informatique au fur et à mesure que leurs besoins évoluent.

L'informatique dans le nuage est un concept jeune et en constante évolution. Une des évolutions les plus prometteuses d'après la communauté scientifique est le déploiement d'applications distribuées sur une fédération de nuages [45, 136, 31, 38, 92, 112]. En effet, ce mode de déploiement permet, entre autres, d'atténuer le risque de verrouillage propriétaire, de stimuler la concurrence des différents fournisseurs d'infrastructure dans le nuage, et de contrôler une partie de la confidentialité des données utilisées par les applications déployées dans le nuage. Néanmoins, la gestion des ressources informatiques provenant de plusieurs nuages est un défi technologique et scientifique d'actualité. En effet, la grande taille des fédérations de nuages et l'hétérogénéité des ressources qui les composent sont des aspects difficilement pris en compte par les solutions de l'informatique dans le nuage d'aujourd'hui. Dans ce contexte, cette thèse propose d'étudier l'intégration de systèmes pair-à-pair au sein des infrastructures actuelles de l'informatique dans le nuage.

Chapitre 2

Positionnement de la contribution

Sommaire

Dans le but de positionner la contribution proposée au sein de cette thèse, ce chapitre établit, dans une première section, un rapide état de l'art des services d'allocation de ressources à large échelle. Cet état de l'art compare notamment les services de l'informatique dans les nuages, présentés précédemment, à la grille informatique et aux plateformes de calcul de bénévole. Une seconde section présente les spécifications de Salute, une plateforme pair-à-pair hébergeant un service d'allocation de ressources non gérées, qui s'appuie sur le modèle d'allocation de ressources de l'infrastructure en tant que service. C'est la conception, l'étude, l'évaluation et l'analyse de cette plateforme qui constitue la contribution proposée au sein de cette thèse. Ce chapitre présente la plateforme Salute d'un point de vue utilisateur, c'est-à-dire qu'il présente la spécification du service d'allocation de noeuds qu'elle héberge. Finalement, une liste des motivations qui nous ont poussés à concevoir un tel service est donnée à la fin du chapitre.

2.1 Etat de l'art des services d'allocation de ressources à large échelle

Cette section établit un bref état de l'art des systèmes d'allocation de ressources à large échelle. Ce que nous caractérisons ici comme « système large échelle » est un système capable de gérer l'allocation de plusieurs centaines de machines, au minimum, et dont l'utilisation est partagée publiquement ou par une communauté d'utilisateurs appartenant à des domaines administratifs distincts. Cette section se focalise sur la présentation des trois principaux types

de service d'allocation à large échelle : l'infrastructure en tant que service, la grille informatique et les plateformes de calcul bénévole. Cette section décrit également le principe des infrastructures utilisées pour héberger ces trois types de service.

2.1.1 Infrastructure en tant que service

Comme il est expliqué dans le chapitre précédent, l'infrastructure en tant que service permet d'allouer des espaces d'exécutions virtualisés et des espaces de stockage. Cette allocation est faite à la demande du client, via Internet et en échange d'un paiement à l'utilisation qui est typiquement prédéfini par un contrat de niveau de service. Ce type de service est fourni grâce à une infrastructure matérielle qui est généralement hébergée au sein de centres de traitement de données, lesquels sont typiquement répartis à travers le monde entier. Pour des raisons économiques, ces centres de traitement de données sont de plus en plus grands. En effet, l'étude de Greenberg et coll. a montré que le coût moyen d'hébergement des ressources informatiques matérielles était, de façon simplifiée, inversement proportionnel à la taille des centres de traitement de données [71, 29]. Ce phénomène, appelé économie d'échelle, est principalement dû à la baisse du coût unitaire des ressources informatiques lorsque les tâches d'administration, de refroidissement et d'approvisionnement en électricité sont optimisées pour gérer un grand nombre de machines. Aussi, il est désormais admis que la taille minimale conseillée, pour qu'un centre de traitement de données soit économiquement compétitif, est de 60 000 serveurs [71, 29].

L'architecture des nuages fournissant de l'infrastructure en tant que service est aujourd'hui organisée selon une structure hiérarchique. Un schéma représentant ce type d'architecture est donné dans la Figure 2.1. Ce schéma est basé sur la description de l'infrastructure du nuage Eucalyptus [126], néanmoins son principe de structure hiérarchique est applicable à la majorité des infrastructures en tant que service.

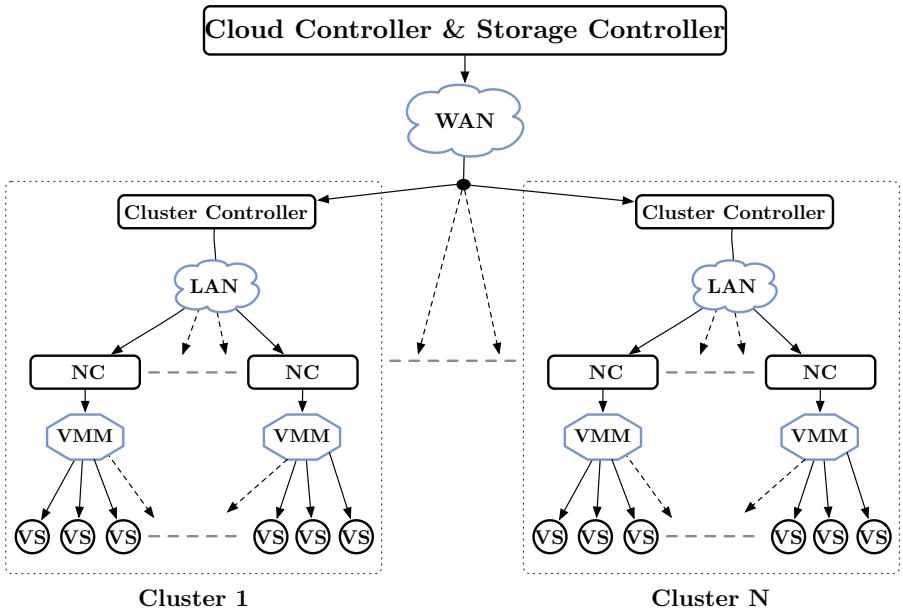


FIGURE 2.1 – Intergiciel pour nuages.

Premièrement, tout nuage proposant une infrastructure en tant que service dispose d'un

contrôleur de nuage, éventuellement couplé à un contrôleur de service de stockage. Il supervise l'ensemble des machines mis à disposition du public par le fournisseur. Ce contrôleur est exécuté sur un serveur central qui est éventuellement répliqué pour des raisons de tolérance aux fautes. Ce contrôleur de nuage coordonne un ensemble de contrôleurs de grappes de serveurs, lequel constitue le second étage de l'architecture hiérarchique du nuage. Ces contrôleurs de grappe supervisent à leur tour un ensemble de contrôleurs de noeuds, c'est le troisième étage de l'architecture du nuage. Enfin, ces contrôleurs de noeuds sont généralement implantés par des moniteurs de machines virtuelles [34, 37, 30], qui gèrent directement l'ensemble des espaces d'exécution virtualisés mis à la disposition des clients du nuage. A noter que pour les plus grandes infrastructures, un étage intermédiaire peut exister entre le contrôleur de nuage et les contrôleurs de grappes. Il s'agit de l'étage des contrôleurs de centres, dont le but est de superviser l'ensemble des grappes de machines déployées au sein d'un même centre de traitement de données. Les communications entre le contrôleur de nuage et les différents centres de traitements de données sont généralement établies via Internet, tandis que toutes les autres communications entre les différentes machines de l'infrastructure sont établies via des réseaux locaux privés. Finalement, c'est en utilisant cette structure en arbre que les fournisseurs d'infrastructure en tant que service sont capables de mettre à disposition de leur client un grand nombre de ressources informatiques, géographiquement réparties dans le monde entier.

2.1.2 Grille informatique

Le concept de grille informatique, ou grille de calcul, fait référence à un ensemble de ressources informatiques appartenant à des entités administratives différentes, qui sont rassemblées pour réaliser un projet commun en formant ce que l'on appelle une « organisation virtuelle ». Une grille informatique est un système distribué qui pourrait s'apparenter à un « supercalculateur virtuel », composé de machines potentiellement hétérogènes qui collaborent à la réalisation de tâches nécessitant un déploiement à grande échelle. Ce qui caractérise les grilles de calcul par rapport aux supercalculateurs conventionnels, comme les grappes de calcul haute performance, est leur tendance à reposer sur l'utilisation de ressources hétérogènes interconnectées avec un couplage faible et géographiquement distribuées. Bien que les grilles informatiques puissent être dédiées à une application donnée, la plupart d'entre elles sont utilisées comme ressource de calcul générique pour des besoins divers et variés. Les grilles de calcul sont souvent administrées à l'aide de boite à outils génériques et d'intergiciels tels que ceux proposés par les projets Globus [64] et Condor [105]. Finalement, la taille des grilles peut varier d'une centaine de machines à plusieurs milliers.

Les tentatives de comparaison entre grille et infrastructure en tant que service sont nombreuses [69, 156, 127]. Ces études font ressortir le fait que la grille est davantage utilisée dans le milieu académique et scientifique que l'est le nuage, ou encore que la grille fait généralement l'objet d'un déploiement communautaire, plutôt que d'un déploiement public. Néanmoins, nous insisterons dans cette thèse sur les modèles d'allocation de ressources proposés par ces deux types de systèmes :

- La Grille informatique est un environnement conçu pour que ses utilisateurs puissent exécuter des calculs haute performance, impliquant souvent un certain parallélisme. Pour des raisons de performance et de contrôle de l'environnement d'exécution des tâches, les machines sont allouées entièrement, c'est-à-dire qu'il n'existe pas de colocalisation de serveurs virtuels sur une même machine physique. Les requêtes d'allocation soumises sur une grille sont typiquement peu nombreuses, mais impliquent un grand nombre de

ressources. Dans certains cas, les utilisateurs sont même amenés à réserver l'ensemble des ressources d'une infrastructure pour l'exécution d'une seule tâche. Aussi, dans un tel système, peu de requêtes d'allocation sont généralement réalisées simultanément, pendant que les requêtes non prioritaires attendent la libération des ressources pour être satisfaites. Les grilles informatiques disposent donc d'algorithmes sophistiqués, visant à optimiser l'ordonnancement de tâches impliquant des calculs massivement parallèles.

- L'Informatique dans le nuage concerne plutôt un grand nombre d'allocations qui impliquent peu de ressources. Par exemple, le service EC2 d'Amazon utilisé comme référence dans le domaine de l'infrastructure en tant que service restreint (par défaut) le nombre maximum de ressources allouables pour un même utilisateur à 20 instances. Cependant, les allocations de ressources sont réalisées en temps réel et il n'existe aucun mécanisme de mise en attente des requêtes, pour gérer le cas où le service d'approvisionnement serait surchargé. Le paradigme d'allocation de ressource est donc totalement différent de celui de la Grille informatique, impliquant des modèles et méthodes d'utilisation des ressources différents.

L'infrastructure typique d'une grille informatique est composée d'un ensemble de grappes de calcul, qui sont interconnectées par un réseau étendu (WAN). Le schéma de cette infrastructure est illustré sur la Figure 2.2. A titre d'exemple, la grille de calcul Grid'5000 est composée

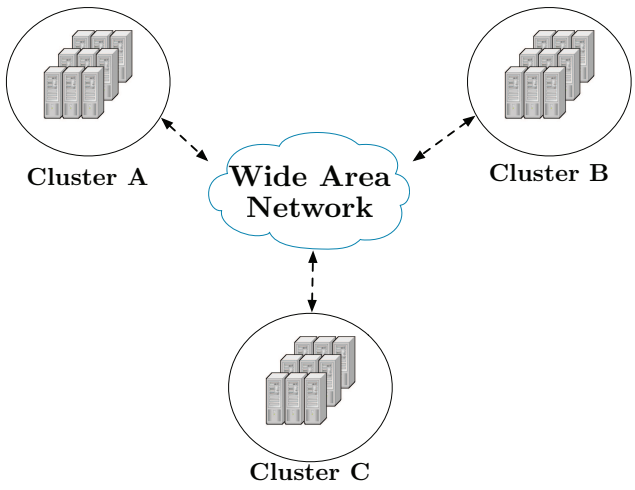


FIGURE 2.2 – Schéma d'une grille informatique.

de 20 grappes de calcul réparties sur 9 sites géographiques et interconnectées par un réseau étendu [12]. L'allocation des ressources d'une grille est réalisée par un ensemble de serveurs dédiés à cette tâche appelés serveurs de noeuds. Généralement, les grilles utilisent un serveur de noeuds par grappe de calcul. Ce serveur est alors chargé de gérer la réservation des noeuds de la grappe sur lequel il est déployé, de gérer les réservations émises par les clients du service d'allocation et aussi d'ordonnancer les allocations de noeuds en présence de concurrence des requêtes clientes. Grâce aux boites à outils développées pour les grilles de calcul, les serveurs d'allocation d'une même grille peuvent collaborer afin d'allouer des ressources provenant de différentes grappes.

2.1.3 Plateforme de calcul bénévole

Le calcul bénévole est un type de calcul distribué au sein duquel un grand nombre de propriétaires d'ordinateurs mettent leurs ressources informatiques (telle que les capacités de calcul et de stockage des dits ordinateurs) à disposition d'un ou plusieurs projets nécessitant une grande capacité de calcul. Le premier projet informatique utilisant le calcul bénévole fut le « Great Internet Mersenne Prime Search » ou « Grande recherche de nombres premiers de Mersenne via Internet », lancé en janvier 1996 par George Woltman. De nos jours, la plateforme de calcul bénévole la plus populaire est sans aucun doute la plateforme BOINC [25], hébergeant notamment les projets Einstein@home [6] et Seti@home [19].

BOINC, pour « *Berkeley Open Infrastructure for Network Computing* », est une plateforme permettant la distribution de tâche de calcul sur des ressources informatiques mises bénévolement à disposition de la communauté scientifique. La particularité de BOINC est d'utiliser un très grand nombre de ressources informatiques anonymes et non gérées, et de ne faire aucune hypothèse sur le comportement de ces noeuds ou même sur leur disponibilité. De façon simplifiée la distribution de calculs au sein de cette plateforme fonctionne comme suit : des serveurs dédiés à la coordination des calculs, appelés serveurs de tâches, sont déployés sur Internet et les noeuds souhaitant partager leurs ressources de calculs via BOINC s'enregistrent sur ses serveurs. Les calculs à effectuer sont proposés par les serveurs de tâches et sont toujours de type « sac de tâches », c'est-à-dire qu'ils ne nécessitent aucune communication entre les différents noeuds. Afin d'assurer l'intégrité des calculs effectués, les serveurs de BOINC utilisent simplement la redondance en demandant l'exécution d'une même tâche par plusieurs volontaires. Ils utilisent ensuite des heuristiques basées sur une analyse statistique des résultats pour déduire, avec une grande probabilité, le résultat d'un calcul.

La plateforme BOINC propose un service d'allocation de ressource original, dans le sens où ce sont les ressources qui viennent au projet de calcul en s'inscrivant sur les serveurs de tâches, alors que dans un service d'allocation de ressource traditionnel c'est le client qui commande l'allocation des ressources. A noter que ce mode d'allocation de ressources ne fonctionne que pour des calculs de types « sac de tâches », pour lesquels les tâches peuvent être exécutées indépendamment les unes des autres. La plateforme BOINC peut donc distribuer une tâche de calcul dès qu'une ressource se présente, sans avoir besoin de formuler des hypothèses sur la disponibilité des ressources bénévoles.

Finalement, un des aspects importants de cette plateforme de calcul est la présence d'un système de crédits visant à mesurer la contribution des bénévoles. Le principe de ce système de récompense est simple, lorsque l'ordinateur d'un contributeur bénévole exécute une tâche avec succès, le serveur responsable de cette tâche accorde à ce contributeur un montant de crédits proportionnel au nombre de calculs effectués. Le montant de crédits récolté par les contributeurs est ensuite périodiquement publié sur le Web, en guise de récompense. Une compétition s'installe alors entre les contributeurs bénévoles, les poussant à fournir davantage de ressources aux projets de calcul hébergés sur la plateforme BOINC. Pour la majorité des experts en calcul bénévole, le système de récompense de BOINC est un élément clef de son succès. Aujourd'hui, la plateforme BOINC compte plus d'un demi-million de bénévoles réguliers contribuant à plus d'une centaine de projets de calcul scientifique, ce qui lui permet d'atteindre une puissance de calcul moyenne, mesurée sur une période de 24 heures, de 3,775.51 TeraFLOPS.

En résumé, cette section présente trois types de services d'allocations de ressources à large échelle : l'infrastructure en tant que service, la grille informatique et les plateformes de calcul bénévole. Ces trois types de services se distinguent par leur modèle d'allocation de noeuds (à

la demande, par réservation, ou en fonction de la disponibilité de ressources bénévoles), mais aussi par la présence de colocalisation de serveurs virtuels, ou encore par les hypothèses faites sur la disponibilité et sur la capacité de traitement des ressources allouées. Dans le cadre de cette thèse, nous proposons un service d'allocation de ressources à large échelle considérant un nouvel élément de comparaison : la présence de serveurs dédiés au sein de l'infrastructure hébergeant le service d'allocation de noeuds.

2.2 Salute : un service pair-à-pair d'allocation de ressources

Cette section présente, dans une première partie, la contribution apportée au sein de cette thèse, à savoir : Salute, un service d'allocation de ressource à large échelle. Une seconde partie décrit les motivations qui nous ont poussés à proposer cette contribution.

2.2.1 Présentation

Salute est une plateforme pair-à-pair qui collecte des ressources bénévoles non gérées, hétérogènes et de provenances diverses, et qui héberge un service d'allocation capable de réallouer ces ressources à de potentielles applications clientes. Ces ressources, également appelées noeuds, correspondent typiquement aux ressources bénévoles utilisées par la plateforme BOINC, décrite précédemment. Elles proviennent de diverses entités administratives et leur comportement, en particulier leur disponibilité, n'est pas contrôlé par le service d'allocation. De plus, Salute fait l'hypothèse qu'un noeud correspond à une machine physique, c'est-à-dire que le service d'allocation n'utilise pas le principe de colocalisation de serveurs virtuels.

Le service d'allocation de la plateforme Salute offre une interface permettant de réallouer les ressources collectées pour l'exécution de tâches distribuées. Pour ce faire, ce service utilise une nouvelle abstraction appelée « nuage pair-à-pair », qui correspond à un ensemble de ressources bénévoles. La spécification des nuages pair-à-pair alloués par Salute comprend la taille du nuage, définie en nombre de noeuds, la durée de déploiement de ce nuages, définie en heures (c.-à-d. la durée d'allocation souhaitée), ainsi qu'un ensemble de filtres visant à sélectionner les noeuds composant le nuage pair-à-pair selon certains attributs. Les contraintes spécifiées au sein de ces filtres peuvent typiquement définir l'espace de stockage disponible, ou encore la capacité de calcul minimale requise qu'un noeud doit présenter pour qu'il fasse partie du nuage à alloué. Finalement, le modèle d'allocation de ressources utilisé par Salute s'inspire de l'infrastructure en tant que service. Effectivement, ce modèle permet d'allouer des nuages pair-à-pair à la demande, en spécifiant un nombre de ressources, une durée d'allocation et éventuellement des contraintes de capacité appliquées aux ressources à allouer.

L'architecture de Salute est schématisée sur la Figure 2.3. Elle possède la particularité d'être maintenue par des protocoles pair-à-pair, c'est-à-dire que ce sont les ressources collectées par Salute qui maintiennent cette architecture et qui exécutent, de façon collaborative, le service d'allocation de noeuds. Le principe des systèmes pair-à-pair est décrit au sein du Chapitre 3 et les protocoles spécifiques au maintien de l'architecture de Salute sont décrits dans les Chapitres 4 et 5. Ces protocoles pair-à-pair sont utilisés pour organiser l'architecture de Salute autour de réseaux pair-à-pair virtuels, également appelés « réseaux d'overlay ». Comme illustré sur la Figure 2.3, un réseau pair-à-pair virtuel est utilisé pour rassembler toutes les ressources collectées par Salute. C'est ce réseau, appelé réseau principal, qui définit les limites de la plateforme pair-à-pair. Salute utilise également un réseau virtuel par nuage pair-à-pair, afin de rassembler tous les noeuds qui y sont alloués. A noter que chaque noeud appartient au réseau virtuel de Salute et que les noeuds alloués au sein de nuages pair-à-pair

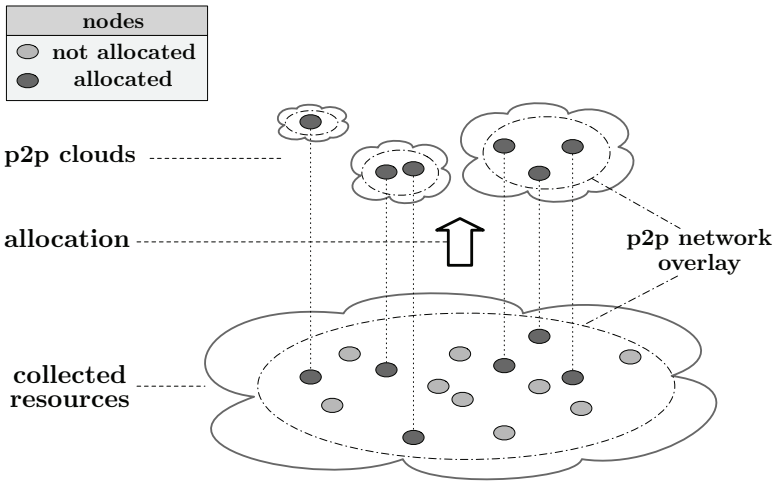


FIGURE 2.3 – Architecture de la plateforme Salute.

appartiennent à deux réseaux : le réseau principal de Salute et celui du nuage au sein duquel ils sont alloués. L'utilisation de ces réseaux virtuels pair-à-pair permet, entres autres, à Salute de se distinguer des services d'allocations de l'état de l'art par la propriété suivante :

> *L'architecture de Salute est maintenue de façon totalement décentralisée par les ressources bénévoles, si bien qu'aucun serveur dédié n'est utilisé pour le déploiement du service d'allocation.*

Les caractéristiques des services d'allocation de ressources décrits dans ce chapitre, y compris celles du service proposé par la plateforme Salute, sont rassemblées dans la Table 5.1.

2.2.2 Motivations

Le choix du modèle d'allocation de ressources utilisé dans la plateforme Salute, c'est-à-dire un modèle d'allocation de ressources à la demande, est motivé par le succès récent de l'infrastructure en tant que service. Comme l'explique le Chapitre 1, l'infrastructure en tant que service présente des avantages économiques certains. D'un coté, les prestataires de service peuvent rentabiliser leur infrastructure matérielle en louant leur capacité de traitement inutilisée. D'un autre coté, les clients de l'infrastructure en tant que service, peuvent bénéficier d'un mode d'approvisionnement en ressources informatiques fiable et élastique. Ce nouveau mode d'approvisionnement simplifie le dimensionnement des applications clientes en permettant d'adapter le nombre de ressources alloué pour leur exécution de façon dynamique, en fonction de la charge de travail subit par ces applications.

L'objet de cette section est de motiver le choix de l'implantation totalement décentralisée de la plateforme Salute. Les motivations qui sont détaillées dans les paragraphes suivants sont : l'absence de serveur dédié, l'opportunité de réaliser des services pair-à-pair composites, l'opportunité de fournir un service d'allocation générique pour les services assistés de pairs, la distribution géographique de l'infrastructure du nuage et l'opportunité d'hébergement d'une plateforme commune pour les fédérations de nuages.

	Nuage	Grille	BOINC	Salute
Accessibilité	publique et payante	communautaire	publique et gratuite	publique et gratuite
Type d'allocation	à la demande	sur réservation	bénévole	à la demande
Nombre de noeuds gérés	10^2 à 10^4	10^2 à 10^3	10^2 à 10^5	10^2 à 10^4
Virtualisation des ressources	oui	non	non	non
Disponibilité des ressources	haute et fiable	haute et fiable	variable et non contrôlée	variable et non contrôlée
Capacité de traitement par ressource allouée	proportionnelle au prix	haute et compatible avec le calcul haute performance	variable, mais généralement faible	variable, mais généralement faible
Présence de serveur dédié	oui (contrôleurs de grappe et de nuage)	oui (serveurs de noeuds)	oui (serveurs de tâches)	non

TABLE 2.1 – Caractéristiques des systèmes d'allocation de ressources à large échelle.

Absence de serveur dédié et réalisation de service pair-à-pair composite : La nature pair-à-pair de la plateforme Salute permet au service d'allocation de noeuds de se passer de la présence d'un serveur dédié. Tout d'abord, cette propriété permet d'économiser les coûts de maintenance et d'administration liés à la présence d'un serveur. Nous pensons également que le service d'allocation de nuages pair-à-pair offert par la plateforme Salute constitue un paradigme nouveau pouvant être utile au développement d'une nouvelle génération de service pair-à-pair reposant sur la collaboration de plusieurs sous-services. Cette remarque est inspirée d'une constatation émise par Babaoglu et coll. [33] en 2006 : la majorité des protocoles pair-à-pair sont spécifiques à une application donnée, et il n'existe pas d'architecture décentralisée permettant l'exécution de services composites pair-à-pair, c'est-à-dire de services basés sur la collaboration de plusieurs sous-services implantés, eux aussi, par des protocoles pair-à-pair. D'après ces auteurs, pour rendre possible le déploiement de services pair-à-pair composites, les protocoles pair-à-pair doivent offrir des interfaces d'administration plus complètes. Ces interfaces permettent notamment de démarrer, d'initialiser ou encore d'arrêter l'exécution de ces protocoles et d'allouer dynamiquement les ressources de l'environnement hôte aux différents sous-services. L'idée de Babaoglu et coll. est de voir l'ensemble des noeuds d'un système pair-à-pair comme un nuage et de créer des sous-nuages au sein de cet ensemble afin d'y déployer les sous-services d'un service pair-à-pair composite. Les pistes proposées par Babaoglu et coll. se basent sur l'utilisation d'un protocole de morcellement de réseau, lequel est présenté en détail dans le Chapitre 3. Cependant, aucun service d'allocation de noeuds pair-à-pair n'a été proposé par ces auteurs. En guise d'exemple de service pair-à-pair composite, Babaoglu et coll. propose un scénario dans lequel un nuage de noeuds exécute une application pair-à-pair classique, telle qu'un système de stockage distribué ou qu'une application décentralisée de messagerie instantanée, en collaboration avec un service de surveillance de charge de travail permettant typiquement d'évaluer l'espace de stockage moyen disponible

par pair ou encore le nombre moyen de communications effectué par noeud. Ce service composite utilise alors un sous-nuage pour déployer le service de surveillance et un autre sous-nuage pour exécuter l'application métier. Dans la continuité des travaux de Babaoglu et coll., nous pensons que le service d'allocation de noeuds de la plateforme Salute, basé sur l'utilisation de nuages pair-à-pair, constitue une étape intéressante vers la réalisation de services pair-à-pair composites.

Opportunité pour le déploiement de services assistés de pairs : Les services assistés de pairs sont des services distribués qui utilisent les ressources informatiques de leurs clients (les pairs) pour soulager la charge de travail de l'infrastructure hôte, laquelle est typiquement composée d'un ou de plusieurs serveurs dédiés. Ce modèle de partage de la charge de travail est apparu durant les cinq dernières années, dans les services de distribution de vidéos à la demande, afin d'utiliser la bande passante des clients pour participer à la diffusion des vidéos [77]. Plus récemment, certains travaux ont proposé une solution hybride visant à combiner le principe des services assistés de pairs avec l'infrastructure en tant que service. Ces services utilisent l'infrastructure de l'informatique dans le nuage pour assurer une qualité de service minimum, en se reposant sur un ensemble de machines fiables, et utilisent les pairs afin de réduire la charge de travail subit par l'infrastructure en tant que service, ce qui réduit également le coût d'hébergement du service. On peut compter parmi ces services, le service de diffusion de contenu de Sweha et coll. [146], qui utilise l'infrastructure en tant que service pour assurer une qualité de service minimum (c.-à-d. un temps de diffusion borné) et qui utilise la bande passante des pairs pour soulager la charge de travail subit par cette infrastructure. Le service de sauvegarde de données proposé par Toka et coll. [148] utilise également ce modèle de déploiement hybride. L'infrastructure en tant que service y est utilisée pour fournir un espace de stockage et une capacité de communications qui soient fiables et hautement disponibles, tandis que la capacité de stockage et la bande passante des pairs sont mises à contribution pour réduire le coût d'hébergement et d'exécution du service de sauvegarde.

Plus récemment, Andrzejak et coll. a proposé un modèle de déploiement hybride générique pour les services assistés de pairs étant hébergés dans le nuage [27]. Cette étude part du principe que les clients des services Web qui sont hébergés dans le nuages possèdent des ressources de calculs pouvant être mises à profit, pour réduire le coût de déploiement de ces services. Andrzejak et coll. établit deux restrictions sur les services candidats à un déploiement assistés de pairs :

- *Le service doit pouvoir s'accommoder du fait que les pairs sont des ressources non gérées dont la disponibilité n'est pas contrôlée et que, de fait, ceux-ci peuvent se déconnecter à n'importe quel moment.*

- *Le service doit prendre en compte le coût des transferts de données opérés entre les pairs, géographiquement répartis dans le monde entier, et les noeuds appartenant à l'infrastructure en tant que service hébergeant une partie du service.*

Selon Andrzejak et coll., les types d'applications suivants peuvent être facilement adaptés à ce modèle de déploiement : les applications utilisant l'infrastructure en tant que service uniquement pour traiter les pics de charge de travail, les calculs utilisant la plateforme MapReduce [56], ou encore les applications de stockage personnel (Dropbox [5], Wuala [21]). A noter que l'étude d'Andrzejak et coll. propose une méthode générique permettant de calculer le nombre de pairs à utiliser pour obtenir un rapport optimal entre le coût d'hébergement du service et la qualité de service. Cependant, Andrzejak et coll. ne propose aucun service

capable de collecter les ressources non gérées, afin de les redistribuer pour le déploiement de service assisté de pairs. Dans la continuité de ces travaux, nous pensons que le service d'allocation de noeuds de la plateforme Salute, constitue une opportunité pour l'allocation de noeuds aux services assistés de pairs. En effet, la plateforme Salute peut être hébergée par un ensemble de ressources non gérées, par exemple les pairs d'un service assisté, ce qui réduit le coût d'administration et de maintenance du service d'allocation de noeuds.

Distribution géographique du service d'allocation : Une des caractéristiques des infrastructures en tant que service existantes est leur faible répartition géographique. En effet, bien que la plupart d'entre elles disposent de plusieurs centres de traitement de données, répartis sur différents continents, les infrastructures des nuages d'aujourd'hui ne proposent généralement pas plus d'une dizaine de sites de déploiement. L'étude de Church et coll. oppose deux modèles de construction d'infrastructure en tant que service : le « mégamodèle », correspondant aux centres de traitement de données composés de cent mille à un million de serveurs hébergés sur un unique site géographique, avec le « micromodèle », correspondant à des centres de traitement de données composés de mille à dix mille machines [48]. L'idée de Church et coll. est qu'une infrastructure en tant que service peut être bâtie sur un des deux modèles de construction, indépendamment de son utilisation ou des services qu'elle propose. Dans un premier temps, cette étude évalue les coûts de construction et de maintenance d'une infrastructure en tant que service en fonction de son degré de répartition géographique, c'est-à-dire selon les deux modèles de construction présentés ci-dessus. Le résultat de cette évaluation fait apparaître que les coûts d'approvisionnement en électricité et les coûts liés à l'infrastructure réseau des centres de traitement de données sont plus faibles dans le cas du micromodèle. Ce résultat s'explique par le fait que lorsque les services d'approvisionnement en électricité et d'accès à Internet sont proportionnellement plus chers lorsque le nombre de serveurs dépasse cent mille unités [48]. Pour Church et coll. une infrastructure bâtie sur le micromodèle est donc moins chère à construire et à maintenir.

Dans un second temps, ces auteurs proposent d'évaluer l'impact de la décentralisation d'une infrastructure en tant que service sur l'efficacité de certaines applications distribuées. Pour ce faire, Church et coll. étudie le cas d'un service de gestion de courrier électronique avec filtrage de courriers indésirables. Dans ce service, les noeuds hébergés au sein de l'infrastructure du nuage sont utilisés pour acheminer les courriers des utilisateurs et pour filtrer les courriers indésirables. Il apparaît alors que lorsque l'infrastructure en tant que service est géographiquement distribuée, le service de filtrage de courriers indésirables est plus efficace. Ce résultat est expliqué par le fait que les noeuds utilisés pour filtrer les courriers sont plus proches des utilisateurs lorsque la topologie de l'infrastructure est géographiquement distribuée, ce qui permet de stopper la propagation des courriers indésirables plus rapidement. Le fait de bloquer la propagation des courriers indésirables, en étant proche de leurs sources, augmente l'efficacité du service de filtrage et permet d'économiser la bande passante utilisée par le service de gestion de courriel. Church et coll. conclut alors que la décentralisation d'une infrastructure en tant que service, en plus de réduire les coûts de construction, permet d'augmenter l'efficacité de certaines applications.

Finalement, Church et coll. pousse à l'extrême le principe de décentralisation d'une infrastructure en tant que service, c'est-à-dire en considérant une infrastructure pair-à-pair, et argumente qu'un certain nombre d'applications « hautement distribuées » peuvent facilement être déployées sur une infrastructure totalement décentralisée. Cet ensemble d'applications comprend notamment : la gestion de courrier électronique, la téléphonie via Internet, la sauvegarde de données, ou encore le déploiement d'applications sur les plateformes de calcul

populaires de type MapReduce [56]. Pour Church et coll., la particularité de ces applications est que leurs implantations ne nécessitent pas de communications intenses avec un ensemble de serveurs centralisés, ce qui les rend compatibles avec un déploiement sur une infrastructure largement répartie. Nous pensons que le travail présenté dans cette thèse autour de la plateforme Salute est une piste intéressante pour le développement d'infrastructure en tant que service présentant un haut degré de répartition géographique.

Opportunité pour les fédérations de nuages : Le déploiement d'applications distribuées sur une fédération de nuages est une solution envisagée aujourd'hui par de nombreux utilisateurs d'infrastructure en tant que service. La motivation principale de ce type de déploiement est de diversifier les fournisseurs d'infrastructure, afin d'assurer la protection des clients contre le verrouillage propriétaire (Cf Chapitre 1). On trouve aujourd'hui deux axes de recherches ayant pour but de permettre un tel déploiement d'applications au sein du nuage. Le premier axe de recherche est l'étude d'un marché global de ressources informatiques, permettant de gérer de façon automatique les aspects contractuels d'un déploiement impliquant des nuages indépendants [45, 38]. Le second axe de recherche est le développement d'intergiciels pour infrastructure en tant que service, permettant d'uniformiser les interfaces d'accès aux différents nuages [136, 31, 113].

Dans le cadre de cette thèse, nous proposons un troisième axe de recherche consistant à modéliser l'ensemble des ressources appartenant à une fédération de nuages sous la forme d'un réseau virtuel. Ce réseau virtuel est utilisé pour rassembler tous les noeuds mis à disposition des clients de l'infrastructure en tant que service, par différents nuages, au sein d'une fédération donnée. On dit alors que ce réseau virtuel « héberge » la fédération de nuages. En tenant compte de la popularité de l'infrastructure en tant que service, de l'échelle des déploiements qu'elle permet de réaliser et du nombre croissant de prestataires proposant aujourd'hui ce type de service, nous proposons de caractériser le réseau virtuel hébergeant une fédération de nuages, selon les trois propriétés suivantes :

- **Taille du réseau** : Une fédération de nuages est composée d'un nombre de noeuds potentiellement grand, c'est-à-dire un nombre de noeuds trop important pour que le réseau hébergeant les noeuds de la fédération soit géré manuellement.
- **Indépendance des ressources** : Les noeuds composant une fédération de nuages proviennent de différentes entités administratives. L'administration du réseau hébergeant les noeuds de la fédération doit donc faire l'objet d'un consensus de la part des différents propriétaires de nuage participant à cette fédération. En effet, si l'utilisation d'un serveur dédié est nécessaire pour remplir cette tâche, alors il faut désigner un responsable pour sa maintenance et pour son administration.
- **Dynamique de l'environnement** : Le comportement des noeuds participant à la fédération de nuages n'est pas géré par une autorité unique. En particulier, cette caractéristique pose le problème de la gestion de la disponibilité des ressources pour le déploiement des applications ciblées. En effet, si on considère qu'une fédération de nuage est composée d'un nombre suffisamment important de nuages indépendants, alors il devient raisonnable d'estimer que l'on ne peut pas contrôler la disponibilité de chaque ressource. La fédération de nuage représente alors un environnement dynamique au sein duquel la disponibilité des ressources n'est pas prévisible à un grain individuel. Le réseau virtuel hébergeant cette fédération devra donc tenir compte de cette dynamique.

La problématique posée par ces trois propriétés peu être résumée par la question suivante :

Comment organiser un réseau virtuel composé d'un nombre potentiellement grand de ressources, qui appartiennent à des entités administratives différentes et dont la disponibilité n'est pas contrôlée par une autorité unique ?

A notre connaissance, il n'existe aujourd'hui aucun service d'allocation de ressources capable d'allouer des noeuds provenant d'une fédération composée de plus d'une dizaine de nuages. Nous pensons que la plateforme Salute est un bon candidat pour pallier à ce manque. En effet, en déployant la plateforme Salute sur une fédération de nuages, l'administration du réseau virtuel hébergeant les noeuds mis à disposition des clients et l'exécution du service d'allocation seront effectués de façon collaborative par les noeuds de la fédération. Aucun serveur dédié n'est donc nécessaire au maintien du réseau virtuel hébergeant la fédération de nuages. De plus, l'utilisation de protocoles pair-à-pair pour le maintien de son réseau principal fait de la plateforme Salute un candidat idéal pour la gestion d'un environnement dynamique composé d'un grand nombre de noeuds, comme c'est le cas pour les fédérations de nuages. En effet, les systèmes pair-à-pair possèdent généralement de bonnes propriétés de passage à l'échelle et sont robustes à la dynamique des environnements sur lesquels ils sont déployés, comme il est expliqué dans le chapitre suivant.

Conclusion : Ce chapitre dresse un bref état de l'art des services d'allocation de ressources à large échelle et présente Salute : une plateforme pair-à-pair proposant un service d'allocation de noeuds. Ce service alloue des ressources non gérées, c'est-à-dire des ressources contrôlées par diverses entités administratives et dont la disponibilité n'est pas contrôlée par une autorité unique. La particularité de Salute est de proposer un service d'allocation de noeuds capable de s'exécuter de façon totalement décentralisée à l'aide des ressources non gérées qu'il collecte, et ce, sans aucun serveur dédié. Pour ce faire, l'architecture de Salute est basée sur l'utilisation de protocole pair-à-pair, plus particulièrement de protocoles dits « épidémiques ». Le chapitre suivant présente une vue d'ensemble des systèmes pair-à-pair et introduit les principaux protocoles épidémiques utilisés pas la plateforme Salute.

Chapitre 3

Les systèmes pair-à-pair

Sommaire

Les systèmes pair-à-pair, aussi appelés systèmes p2p pour « *peer-to-peer* », sont des systèmes distribués collaboratifs de grande taille, pouvant compter jusqu'à plusieurs million de participants. Au sein d'un système pair-à-pair, les participants mettent une partie de leurs ressources informatiques (capacité de calcul, de stockage ou de communication) à disposition des autres membres du réseau, dans le but d'établir un service collaboratif. Ces systèmes sont distribués par nature : ils ne comportent généralement ni hiérarchie, ni point de contrôle centralisé. Leur popularité a été bâtie par les réseaux de partage de fichiers du début du siècle. Aujourd'hui, les systèmes pair-à-pair sont utilisés à des fins plus variées, en partie au sein de la communauté scientifique.

Ce chapitre a pour but de présenter le domaine des systèmes pair-à-pair. Du fait que la contribution proposée au sein de cette thèse se repose sur l'utilisation d'un type spécifique de système pair-à-pair : les systèmes à réseau virtuel non structuré, ces derniers sont davantage détaillés dans l'état de l'art qui suit. La première partie du chapitre présente les systèmes pair-à-pair dans leur globalité, puis la seconde partie donne le détail de certains protocoles pair-à-pair, dont ceux qui sont utilisés dans la contribution de cette thèse. Finalement, une troisième partie décrit des protocoles pair-à-pair de gestion de réseau multicouches, qui proposent un service proche de celui fourni par la plateforme Salute.

3.1 Introduction aux systèmes pair-à-pair

Cette première section présente les systèmes pair-à-pair, leurs intérêts et utilisations principales, ainsi qu'une brève caractérisation des environnements dynamiques qui les hébergent. Elle présente également les deux principales classes de systèmes pair-à-pair : les systèmes structurés et les systèmes non structurés, ainsi que des exemples de protocole permettant le maintien des réseaux associés à ces systèmes.

3.1.1 Principe des systèmes pair-à-pair

Dans un système pair-à-pair, les noeuds, aussi appelés « pairs », sont à la fois consommateurs et fournisseurs de service. C'est ce double rôle de client/serveur qui distingue les systèmes pair-à-pair des systèmes distribués plus classiques, dans lesquels ces deux rôles sont généralement séparés. Les modèles architecturaux client/serveur et pair-à-pair sont décrits sur la Figure 3.1.

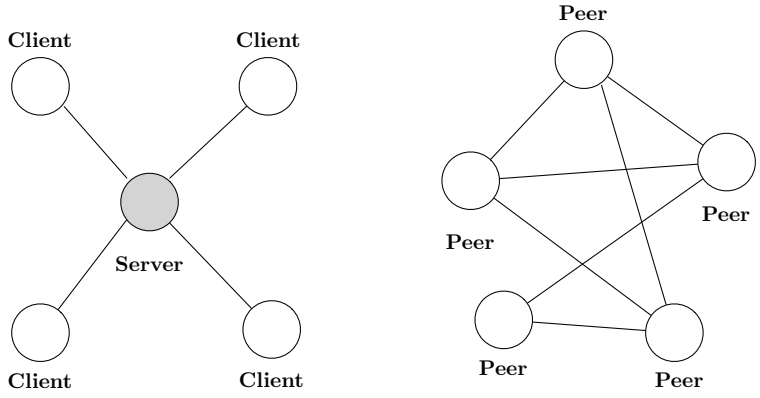


FIGURE 3.1 – Modèle d'architecture client/serveur et modèle pair-à-pair.

Il est très courant d'entendre, ou de lire dans la littérature, que les systèmes pair-à-pair se distinguent par leur capacité de passage à l'échelle, c'est-à-dire par leur capacité à servir un très grand nombre de clients (jusqu'à plusieurs millions), et ce, de façon simultanée. Dans cette thèse, nous ferons attention à nuancer ce type de caractérisation, car certains services informatiques basés sur une architecture de type client/serveurs parviennent à fournir des prestations équivalentes [10, 8]. Pour ce faire, ces services multiplient le nombre de serveurs déployés par service. Ces serveurs sont hébergés dans des centres de traitement de données, et leur allocation aux différents services fait l'objet d'optimisations, plus ou moins complexes, comme décrit dans le Chapitre 1. D'après la synthèse proposée par Rodrigues et Druschel [137], nous proposons de caractériser les systèmes pair-à-pair comme des systèmes distribués présentant les propriétés suivantes :

- **Haut degré de décentralisation :** Les pairs prennent le rôle de client et de serveur, et la majeure partie de l'état du système et des tâches sont dynamiquement distribués parmi les pairs. Il existe peu de noeuds, voire aucun, possédant un état centralisé. L'ensemble des besoins de calcul, de stockage et de communication liés à l'exécution du système est alors fourni de façon collaborative par les membres du système pair-à-pair.

- **Organisation automatique :** Une fois qu'un noeud est introduit au sein du système (typiquement en lui fournissant l'adresse IP d'un des membres), aucune configuration manuelle n'est nécessaire pour maintenir l'exécution du système pair-à-pair.
- **Entités administratives multiples :** Les noeuds participant au système pair-à-pair dépendent rarement d'une seule entité administrative. Dans la plupart des cas, chaque noeud appartient à un contributeur particulier souhaitant participer à la vie du système.
- **Faible coût de déploiement :** N'utilisant pas, ou peu, d'infrastructure dédiée, le coût de déploiement initial d'un service pair-à-pair est généralement faible en comparaison du coût de déploiement des services utilisant une architecture de type client/serveur.
- **Croissance organique :** Etant donné que la majeure partie des ressources exécutant le système sont fournies par les pairs, c'est-à-dire les clients du service, la croissance d'un système pair-à-pair ne nécessite pas de mises à jour franches de l'infrastructure hôte, par exemple le remplacement d'un serveur par un matériel plus puissant.
- **Robustesse aux pannes et aux attaques :** Les systèmes pair-à-pair sont généralement résistants aux pannes, car il existe peu, ou pas, de noeuds dont la présence est critique pour l'exécution du service associé. Pour attaquer ou rendre inaccessible un système pair-à-pair, un tiers malveillant doit prendre pour cible une grande partie des noeuds du système simultanément.
- **Abondance et hétérogénéité des ressources :** Les systèmes pair-à-pair les plus populaires rassemblent une quantité de ressources informatiques que peu d'entreprises peuvent prétendre acquérir individuellement. Les ressources sont hétérogènes en terme d'architecture matérielle et logicielle, de méthode d'accès à Internet, ou encore de situation géographique. Cette diversité contribue à réduire les pannes, les attaques, et même la censure rencontrée dans ce type de système distribué.

Les systèmes pair-à-pair possèdent également des limitations conséquentes, dues à leur nature totalement distribuée et au manque de fiabilité des environnements qui les hébergent. Par exemple, il est généralement admis que l'on ne peut pas connaître avec précision l'état exact ainsi que l'ensemble des membres d'un système pair-à-pair, en temps réel. Ces difficultés ont pour conséquences de restreindre le spectre des applications distribuées, qui sont facilement adaptables aux systèmes pair-à-pair. Aussi, un survol des principaux protocoles pair-à-pair est réalisé à travers les sections 3.1.2 et 3.2.

3.1.1.1 Exemples de systèmes pair-à-pair

Les systèmes pair-à-pair ont été rendus populaires par les applications de partage de fichiers, telles que Napster [14] dès 1999. Aujourd'hui, on distingue deux grandes catégories d'applications pair-à-pair en fonction des ressources qu'elles utilisent. On trouve alors les applications utilisant la capacité de communication des pairs, telles que Bittorrent [50], ou encore Skype [35, 73], et les applications utilisant la capacité de calcul des pairs, comme la plateforme BOINC [25] présentée plus tôt au sein du Chapitre 2.

Napster [14] : fut le premier système pair-à-pair largement utilisé (avec un pic à plus de 24 millions d'utilisateurs en février 2001). Il proposait un service de partage de fichiers musicaux entre tous les membres d'une même communauté pair-à-pair publique. Pour ce faire, Napster

utilisait un réseau pair-à-pair dit « hybride », au sein duquel un serveur central était chargé d'indexer les fichiers proposés par les utilisateurs du réseau et d'associer à chaque fichier les adresses IP des pairs qui en possédaient une copie. La nouveauté apportée par Napster à l'époque était l'utilisation de la bande passante des pairs, c'est à dire des clients du service, pour réaliser les échanges de fichiers de façon décentralisée. Aussi, Napster a été le premier service à utiliser les ressources de ses clients pour permettre à son service de passer au-delà de l'échelle des millions d'utilisateurs. Le service proposé par Napster a été utilisé de juin 1999 à juillet 2001, date à laquelle il a été arrêté pour des raisons de légalité et plus particulièrement de violations des droits d'auteurs.

Bittorrent [50] : est un système pair-à-pair de partage et de distribution de fichiers ayant vu le jour en 2001. Il vise à améliorer le principe des systèmes pair-à-pair hybrides tels que Napster en décentralisant les sites responsables de l'indexation des données. Son principe est basé sur l'utilisation de trois types de pairs : les sources (de l'anglais « seeder »), les traqueurs (de l'anglais « tracker ») et les sangsues (de l'anglais « leecher »). Comme dans le cas de Napster, la diffusion des fichiers à grande échelle est rendue possible par la contribution de tous les pairs, qui mettent leur bande passante à disposition du service. De façon simplifiée, le protocole Bittorrent fonctionne comme suit pour la diffusion d'un fichier : le fichier est découpé en plusieurs morceaux ; la source essaie de diffuser chaque morceau à une sangsue différente, puis les sangsues s'échangent les morceaux qui leur manquent entre eux. Ce principe permet de mettre la bande passante des sangsues à contribution lorsqu'ils téléchargent un fichier. Les traqueurs sont alors utilisés pour indexer les associations fichiers/sources et pour coordonner les échanges de fichiers entre tous les pairs.

Généralement, les utilisateurs publient les informations nécessaires pour joindre un téléchargement grâce à des métadonnées, qui sont regroupées au sein d'un fichier de description appelé « fichier torrent ». Les fichiers torrents contiennent, entre autres, l'adresse du traqueur responsable du téléchargement, ainsi que des informations de correction et de vérification d'intégrité concernant le fichier ou l'ensemble de fichiers à télécharger. Un utilisateur souhaitant diffuser un fichier via la protocole Bittorrent doit alors disposer d'au moins une source et d'un traqueur (généralement, une même machine peut remplir ces deux rôles, tant que le torrent est petit). Ensuite, il peut créer un fichier torrent puis le publier, typiquement sur un site Web, afin de solliciter la participation de nouveaux pairs et de faire grossir la capacité de téléchargement associée. Finalement, le succès de Bittorrent dans le début des années 2000 fut tel, que de nombreuses améliorations ont été apportées au protocole d'origine, par exemple pour diversifier son utilisation en diffusant des médias sous forme de flux de données. On estime en 2009 que le protocole Bittorrent est utilisé par un tiers du trafic Internet [1].

Skype [35, 73] : Skype est un protocole propriétaire qui permet aux clients qui l'utilisent de passer des appels vocaux via Internet. Les premières versions de ce protocole sont apparues en 2002. Skype utilise trois types d'agents pour implanter son service : un serveur d'identification, des « superpairs » et des pairs classiques. Comme son nom l'indique, le serveur d'identification est un point centralisé qui regroupe les données nécessaires pour enregistrer, puis identifier, un utilisateur de façon unique et sécurisée. Le principe du protocole Skype est d'utiliser la bande passante des utilisateurs pour établir les communications. Les connexions nécessaires à une communication vocale sont alors de type point à point, c'est-à-dire qu'elles n'impliquent que les deux pairs communicants.

La particularité de Skype est d'utiliser les ressources de calculs et de communication de certains pairs particuliers, les superpairs, pour maintenir un réseau virtuel de routage qui est

utilisé pour localiser les utilisateurs, et parfois même pour établir les communications. Les superpairs sont sélectionnés selon des critères de fiabilité et de disponibilité, afin de limiter la dynamique du réseau virtuel de routage. Aussi le réseau pair-à-pair maintenu par Skype est fiable et efficace, si bien que l'on compte en 2009 plus de 400 millions d'utilisateurs enregistrés et plus de 40 millions d'utilisateurs connectés par jour.

3.1.1.2 Dynamique des environnements pair-à-pair

Dans la suite de ce document, on appellera « environnement pair-à-pair » : l'ensemble des ressources de calcul (les machines) et des ressources de communications (les liens du réseau Internet) qui participent à l'exécution d'un système pair-à-pair. On définit alors la dynamique des environnements pair-à-pair, communément appelée « *churn* », comme étant le phénomène qui caractérise l'arrivée et le départ continu d'une partie des membres du réseau au sein du système. A noter que l'on assimile généralement le départ d'un noeud à une panne et que l'on considère qu'un noeud est en panne lorsqu'il cesse de communiquer avec le reste du système pendant une période prédéfinie, laquelle varie en fonction des besoins du système. Cette assimilation a pour but de modéliser l'évolution de la majorité des systèmes pair-à-pair, dans lesquels le comportement des noeuds n'est pas maîtrisé. Aussi, on ne sait pas quand un noeud décidera de quitter le réseau et on ne peut pas contraindre un pair à rester connecté au système contre sa volonté. En conséquence, un environnement pair-à-pair est un réseau composé d'un nombre de noeuds potentiellement grand, dont on ne maîtrise ni le comportement, ni la disponibilité.

De nombreuses études, visant à caractériser la dynamique des environnements pair-à-pair, ont été proposées au cours des sept dernières années [58, 40, 145, 117]. Les principales propriétés émanant de ces études peuvent être résumées comme suit : soit la disponibilité des noeuds est complètement aléatoire, soit elle suit des modèles de variation périodiques qui sont quotidiens ou hebdomadaires. Ces propriétés seront développées davantage dans le Chapitre 6.

Finalement, les systèmes pair-à-pair sont des systèmes distribués conçus spécialement pour gérer des applications regroupant un grand nombre de participants, dont aucun n'a de rôle particulier. Ces participants sont appelés des « pairs », ce sont eux qui contribuent à l'exécution et au maintien du système, et aucune hypothèse n'est faite sur leur disponibilité. De plus, les systèmes pair-à-pair sont caractérisés par un réseau virtuel, que l'on appelle également *« réseau d'overlay »*. Ce réseau est composé de liens logiques, qui permettent aux membres du système de communiquer entre eux dans le cadre de l'application pair-à-pair à laquelle ils participent. C'est alors l'appartenance à ce réseau virtuel qui définit les limites d'un système pair-à-pair. Bien que l'étude et le développement des systèmes pair-à-pair ne datent que d'une dizaine d'années, il existe une frontière franche qui sépare ces systèmes en deux catégories : ceux qui utilisent un réseau virtuel structuré et ceux utilisant un réseau virtuel non structuré. La suite de cette section est consacrée au détail de ses deux catégories de systèmes pair-à-pair. Certaines implantations y sont présentés, mais il existe des survols plus complets au sein de la littérature scientifiques [107, 26].

3.1.2 Réseaux logiques structurés et tables de hachage distribuées

Les systèmes pair-à-pair structurés tirent leur nom de la topologie qui caractérise leur réseau virtuel. En effet, l'architecture des réseaux pair-à-pair dits structurés est généralement inspirée d'une topologie géométrique bien définie, telle qu'un cercle ou un tore. En utilisant un système de coordonnées reposant sur cette topologie, les pairs membres de ce réseau sont

capables de créer un espace d'adressage, qui est distribué sur la totalité des membres du réseau de façon déterministe. Les espaces d'adressages définis par les systèmes pair-à-pair structurés sont souvent utilisés dans le but de créer des tables de hachage distribuées, dont le principe est défini ci-dessous.

Table de hachage distribué : Une table de hachage distribuée est une classe de système distribué qui fournit un service d'indexation et de recherche de données similaire à celui fourni par une table de hachage classique. Des paires (clé,valeur) sont stockées sur la table de hachage distribuée, et n'importe quel noeud participant au réseau pair-à-pair hébergeant cette table peut retrouver la valeur associée à une clé donnée, de façon efficace. La responsabilité du maintien de l'association des clés et des valeurs est distribuée sur l'ensemble des noeuds du système, de sorte qu'une modification de l'ensemble des participants au réseau ne cause que de légères perturbations dans la structure de la table. Cela permet aux tables de hachage distribuées de se déployer de façon autonome sur des réseaux de très grande taille, tout en gérant les arrivées, départ et pannes continues des noeuds composant l'environnement pair-à-pair hôte.

Les tables de hachage distribuées permettent de bâtir des services distribués plus complexes à partir des réseaux pair-à-pair, tels que des systèmes de fichiers, des plateformes de partage de fichiers, des services de gestion de caches Web ou plus simplement des services de diffusion d'information. Ce document présente les principes des quatre protocoles considérés comme piliers de la gestion des tables de hachage distribuées, à savoir : CAN [133], Chord [141]), Pastry [138] et Tapestry [158]. Un bref survol des principaux services réalisés à partir de ces protocoles est présenté dans les paragraphes suivants.

CAN [133] : (pour « Content Addressable Network ») est un protocole permettant l'implantation d'une table de hachage distribuée sur un réseau pair-à-pair. L'architecture du réseau logique construit par CAN se repose sur un espace virtuel de dimension d muni d'un système de coordonnées cartésiennes. La topologie du réseau virtuel correspond alors à un tore de dimension d, fournissant un espace d'adressage virtuel qui est complètement indépendant de la localisation ou de la connectivité des noeuds du réseau. Des points, correspondant aux clefs des données à enregistrer dans la table de hachage, sont alors définis grâce à des coordonnées exprimées dans cet espace d'adressage. Ces coordonnées sont calculées par une fonction de hachage classique de type SHA-1 [140] à partir d'un mot clef, par exemple le nom du fichier contenant l'information à indexer dans la table. Dans CAN, l'espace de coordonnées est dynamiquement partitionné entre tous les membres du système, de façon à ce que chaque noeud soit responsable d'une zone parmi cet espace. Un exemple d'un tel partitionnement est donné sur la Figure 3.2 pour un réseau à topologie planaire composé de 5 noeuds.

Au sein de la table de hachage distribuée, la gestion d'une paire (clef,valeur) (ajout, mise à jour ou retrait d'une paire) se fait à partir du noeud qui est responsable de la zone comprenant le point associé à la clef de cette paire. Afin de permettre un routage de requêtes au sein du système distribué, chaque noeud maintient une table de routage contenant les adresses IP, ainsi que les coordonnées virtuelles de zones, de chacun de ses voisins dans l'espace torique. De cette façon, un noeud peut router une requête vers son point de destination en suivant les coordonnées de l'espace virtuel. Le noeud détermine, dans un premier temps, quel est le voisin le plus proche du point de destination, puis recherche l'adresse IP de ce noeud au sein de sa table de routage. Ce protocole est alors répété, jusqu'à ce que la requête atteigne le noeud

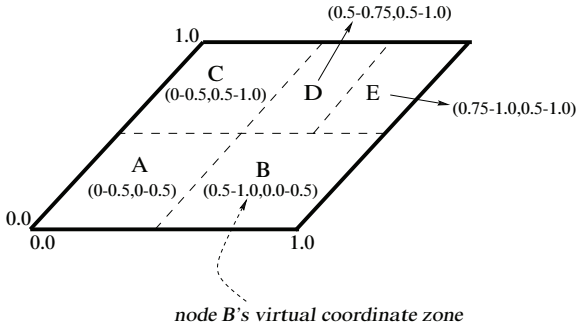


FIGURE 3.2 – Schéma de CAN (d'après Ratnasamy et coll. [133]).

responsable de la zone de destination. Cet algorithme de routage simple et glouton possède une complexité en $O(d.N^{\frac{1}{d}})$, où d est la dimension de l'espace torique.

Lorsqu'un nouveau noeud souhaite joindre le réseau virtuel géré par CAN, il doit : *i)* trouver un noeud faisant déjà partie de ce réseau virtuel, *ii)* puis identifier une zone qui peut être découpée (pour attribuer au nouveau noeud la responsabilité d'une sous-partie de cette zone), *iii)* et enfin veiller à la mise à jour des différentes tables de routage des voisins de cette zone, de manière à prendre en compte le nouveau découpage de l'espace virtuel de coordonnées. Lorsqu'un noeud tombe en panne ou quitte volontairement le réseau, CAN doit : *i)* identifier le noeud qui fait défaut, *ii)* s'assurer de la prise en charge de la zone attribuée à ce noeud par un de ses voisins, *iii)* et mettre à jour les tables de routage au sein du réseau.

En résumé, CAN est un protocole qui permet l'implantation d'une table de hachage distribuée sur un environnement pair-à-pair. La particularité de ce système est d'attribuer la responsabilité des zones d'adressage aux noeuds de façon arbitraire. En conséquence, l'ajout ou le retrait d'un noeud n'implique qu'une modification locale de la structure de la table (au niveau des voisins uniquement). Cependant, CAN est dépourvu de mécanisme permettant la répartition de charge (nombre moyen de clefs par noeud) au sein de la table de hachage distribuée. Aussi il se peut qu'une zone soit surchargée, pendant qu'une autre est sous-utilisée. Ce problème constitue la motivation principale pour considérer d'autres protocoles d'implantation de tables de hachage distribuées, tel que Chord [141], Pastry [138] ou encore Tapestry [158].

Chord [141] est un protocole pair-à-pair implantant une table de hachage distribuée. Ce protocole utilise une fonction de hachage dite « cohérente » [90] pour attribuer des identifiants aux pairs du réseau et aux clés de la table de hachage distribuée de façon uniforme, tout en respectant une topologie circulaire. Ces identifiants comportent m bits et sont calculés grâce la fonction de hachage SHA-1 [140], qui offre de bonnes propriétés de résistance aux collisions, si le paramètre m est suffisamment grand. La fonction de hachage cohérente peut produire $2m-1$ identifiants pour chaque espace d'adressage (le système peut donc contenir, au maximum, $2m-1$ noeuds et $2m-1$ clefs). Typiquement, cette fonction utilise l'adresse IP pour produire les identifiants des noeuds et une chaîne de caractère telle qu'un nom de fichier, pour produire les identifiants des clés. Ces identifiants sont ensuite utilisés pour organiser l'architecture du réseau selon un anneau comportant des positions, correspondant aux noeuds numérotés de 0 à $2m-1$.

D'après la topologie en anneau de Chord, chaque noeud et chaque clé possèdent un successeur et un prédécesseur. Le successeur d'un noeud N_i (ou d'une clé C_i) est le noeud N_j

suivant l'identifiant de N_i (resp. C_i) au sein de l'espace circulaire des identifiants, et ce, dans le sens horaire. Dans le cas où il existe un noeud pour chaque identifiant possible, et si les identifiants sont ordonnés suivant l'ordre croissant des entiers naturels, alors le successeur du noeud N_2 (d'identifiant 2) est le noeud N_3 (d'identifiant 3) et le prédécesseur du N_1 est le noeud N_0. Cependant, il est courant qu'il existe des trous au sein de la séquence des identifiants effectivement attribués aux noeuds de Chord. Un tel cas se produit lorsque le nombre de pairs est inférieur au nombre d'identifiants possibles. Par exemple, le successeur du noeud N_{153} peut être le noeud N_{167}, si les identifiants 154 à 166 ne sont attribués à aucun noeud.

Selon cette topologie, une clef est attribuée au successeur de son identifiant. Par exemple, une clef d'identifiant C_i sera prise en charge par le noeud successeur de l'identifiant i dans l'espace d'adressage des noeuds existants. Aussi, la fonction de hachage cohérente qui détermine l'attribution des identifiants des clés et des noeuds joue un rôle majeur dans la performance et la robustesse du protocole, car elle permet de distribuer uniformément l'ensemble des clefs sur les noeuds existants. Si un système est composé de N noeuds et de C clefs, alors chaque noeud sera responsable, en moyenne, de $\frac{C}{N}$ clés [141].

Chord requiert également que chaque noeud maintienne une table d'index contenant m entrées, de façon à connecter des sous-ensembles de noeuds entre eux. Cette table est construite de façon à ce que l'entrée de position i corresponde au noeud successeur de l'identifiant $N+2^i$, où N est l'identifiant du noeud local. L'utilité de cette table est de permettre des recherches de clefs au sein d'un réseau de taille N avec une complexité de $O(LogN)$ [141]. Un schéma illustrant un exemple d'architecture maintenue par Chord est disponible sur la Figure 3.3. Cette figure illustre également un exemple de routage pour une requête concernant la clef

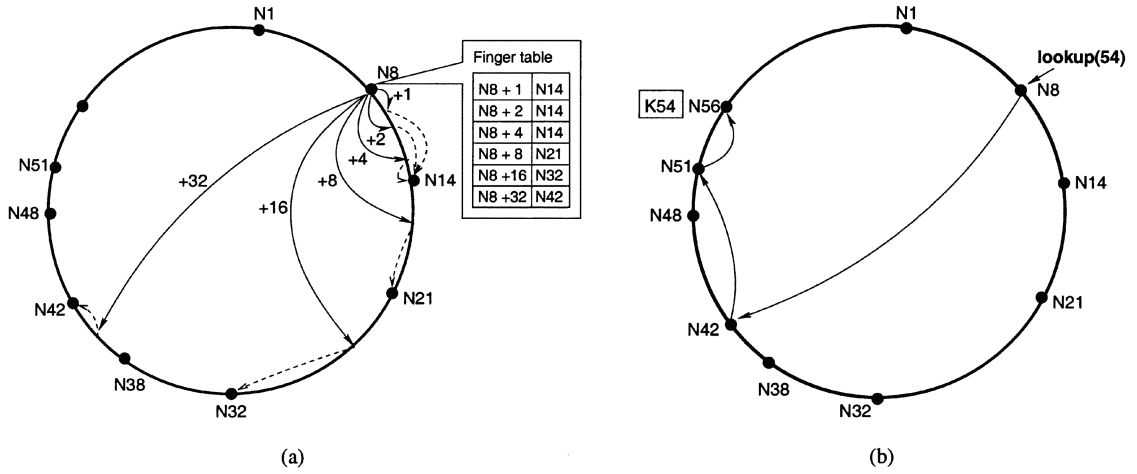


FIGURE 3.3 – Schéma de Chord (d'après Stoica et coll. [141]).

de valeur 54. Comme on peut le voir sur cet exemple, la table d'index des noeuds de Chord permet de retrouver le noeud responsable de cette valeur rapidement, c.-à-d. sans avoir à parcourir l'intégralité des noeuds du système.

Le dernier point à mentionner dans cet état de l'art est la gestion du dynamisme du système par Chord. Puisque le successeur (ou le prédécesseur) d'un noeud peut disparaître du réseau pair-à-pair (à cause d'une panne ou d'un départ volontaire), chaque noeud doit enregistrer les identifiants des noeuds qui lui sont adjacents sur un segment entier de l'anneau virtuel. Par exemple, un noeud doit connaître les identifiants de ses R prédécesseurs et de ses R successeurs. De cette façon, il pourra correctement identifier son successeur et son

prédécesseur, avec une grande probabilité, malgré un taux de panne potentiellement élevé. Á noter finalement que lorsqu'un noeud joint ou quitte le réseau, un équilibrage de charge d'environ $\frac{C}{N}$ clés est opéré pour maintenir ou pour redistribuer la responsabilité des clefs, le cas échéant.

Plusieurs services, directement basés sur Chord, ont été réalisés avec succès. Les plus populaires sont : une implantation distribuée du service de nommage DNS [51] et un système collaboratif d'hébergement de fichiers [54].

Pastry [138] : est un protocole pair-à-pair implantant une table de hachage distribuée. La table de hachage distribuée implantée par Pastry possède une fonctionnalité similaire à celles proposées par Chord [141] et CAN [133]. Cependant, Pastry se distingue de ses concurrents par l'utilisation d'un réseau virtuel de routage des requêtes, qui est indépendant du réseau hébergeant l'infrastructure de la table de hachage. Cela permet à Pastry d'hériter des propriétés de passage à l'échelle et de tolérance aux fautes des tables de hachage distribuées, tout en réduisant le coût de routage des requêtes d'un noeud à l'autre, en évitant d'impliquer tous les noeuds du réseau dans ce routage. Cette idée est inspirée de l'algorithme de Plaxton [130, 103], qui introduit la séparation des rôles de serveur (noeud implantant la logique métier de l'application distribuée) et de routeur (noeud dédié au routage des communications) au sein d'un même système distribué. Ainsi, l'indépendance du protocole de routage des requêtes permet l'utilisation de métriques visant à optimiser un aspect spécifique des communications. Par exemple, ces métriques peuvent viser à optimiser le nombre de sauts réalisés lors du routage d'une requête, ou alors la latence, le débit ou même une combinaison de plusieurs de ces métriques.

L'espace d'adressage des clefs de la table est circulaire, comme celui utilisé dans Chord, et les noeuds sont identifiés à l'aide d'entiers naturels non signés chiffrés sur 128 bits. Ces identifiants représentent les positions des noeuds sur l'anneau virtuel qui définit la structure du réseau. Comme dans Chord, les identifiants des noeuds sont choisis aléatoirement et uniformément, si bien que les noeuds adjacents sur l'anneau virtuel possèdent des coordonnées géographiques diverses. Le réseau virtuel de routage est superposé au réseau virtuel utilisé pour maintenir la structure de la table de hachage. Chaque noeud possède trois structures de données contenant les informations utiles au maintien de l'état du système : une table des pages de noeuds (appelée « leaf set » dans la version originale anglophone), une liste de voisins et une table de routage. La table des pages de noeuds contient les $\frac{L}{2}$ noeuds les plus proches dans chaque direction sur l'espace d'adressage circulaire utilisé pour les identifiants des pairs. La liste des voisins contient les M noeuds les plus proches selon la métrique de routage choisie. Cette liste est utilisée pour maintenir une notion de localité au sein de la table de routage [138]. Finalement, la table de routage contient des blocs d'adresses de clefs.

Pour former les blocs d'adresses, Pastry coupe les identifiants des clefs (chiffrés sur 128 bits) en digits de longueur b, exprimés dans un système de numérotation de base 2^b. Ce mécanisme partitionne les adresses des clés en différents niveaux, du point de vue de l'utilisateur. Le niveau 0 représente l'égalité entre deux adresses, le niveau 1 représente un préfixe commun de 1 bit, puis le niveau 2 représente un préfixe commun de 2 bits, et ainsi de suite. La table de routage d'un pair contient l'adresse du pair le plus proche de chaque digit, pour chaque niveau d'adressage, à l'exception du digit qui appartient au pair local pour un niveau particulier. La table de routage contient donc $2b-1$ contacts par niveaux, avec un nombre de niveaux évoluant en $O(Log_{2^b} N)$.

De façon simplifiée, le protocole de routage consiste, pour un noeud, à transmettre la requête à traiter, soit au noeud destinataire si celui-ci est contenu dans la table des pages

locales, soit au noeud de la table de routage qui partage le plus grand préfixe commun avec la destination de la requête. En suivant ce principe, la latence du routage d'une requête d'un noeud à l'autre, en terme de nombre de sauts, est bornée par la valeur $Log_{2^b}N$ [138]. Aussi, des valeurs typiques pour les paramètres de Pastry sont : $b = 4$, $L = 2^b = 16$ et $M = 2^b = 16$. De façon similaire à Chord, l'arrivée ou le départ d'un noeud dans le réseau géré par Pastry, engendre des modifications sur les tables et listes d'un sous-ensemble des noeuds du système [138].

On peut compter aujourd'hui plusieurs dizaines de services construits au-dessus de Pastry, comme le système de fichier distribué PAST [59], la plateforme de services de types « Publish/-Subscribe » Scribe [139], le service collaboratif de diffusion de flux de média Splitstream[47] ou encore le service collaboratif de gestion de cache Web Squirrel [81].

Tapestry [158] : est un protocole se basant également sur l'algorithme de Plaxton [130, 103] pour implanter une table de hachage distribuée. Les principales différences entre Tapestry [158] et Pastry [138] résident dans la gestion de la réplication d'objets et de la prise en compte de la proximité des noeuds dans le protocole de routage des requêtes. Les performances de routages sont équivalentes pour les deux protocoles. Tapestry est notamment connu pour être le support du système distribué de stockage de fichier Oceanstore [98]. Oceanstore est un système largement distribué et hautement disponible, qui est notamment déployé au sein de la plateforme de test distribuée PlanetLab.

Parmi les implantations connues de tables de hachage distribuées, on peut également citer Kademlia [114] et Viceroy [108], qui ne sont pas détaillées dans ce document. Néanmoins, en considérant les principales études concernant les tables de hachage distribuées [26, 46, 107], on retiendra que ces systèmes distribués sont efficaces pour la recherche d'objets indexés et qu'ils supportent bien le passage à l'échelle et la dynamique de certains environnements pair-à-pair. De ce fait, ils permettent l'élaboration de services distribués complets, tels que des systèmes de fichiers distribués ou encore des systèmes collaboratifs de gestion de cache Web. Enfin, ces systèmes pair-à-pair ne satisfont pas tous les types d'applications, car leur programmation peut être difficile. En effet, il apparaît que les implantations des tables de hachage distribuées nécessitent des évolutions, parfois complexes par rapport aux protocoles de référence, lorsque l'on considère des aspects de sécurité [26, 107] ou de performance sur des déploiements réels [46]. Cette dernière remarque explique, en partie, que d'autres architectures soient considérées comme base de construction des systèmes pair-à-pair, en particulier les réseaux logiques non structurés.

3.1.3 Réseaux logiques non structurés et protocoles épidémiques

On appelle réseau logique non structuré tout réseau logique dont la topologie n'est pas contrainte par une structure. Par extension, la localisation des données et des services proposés au sein de réseaux logiques non structurés ne sont pas non plus contraints par une structure particulière. Aussi, comme il est détaillé davantage au cours de cette section, la topologie de ces réseaux s'apparente souvent à celle d'un graphe aléatoire.

Les réseaux non structurés ont été les premiers à voir le jour au sein de la communauté des systèmes pair-à-pair, en particulier pour leur utilisation dans les applications largement distribuées de partage de fichiers. Les réseaux non structurés les plus populaires utilisés pour la diffusion de média ou pour le partage de fichiers sont : Freenet [107, 49], Gnutella [107, 144], Bittorrent [107, 50], FastTrack [107, 26, 104], Overnet [107, 99] et le réseau Edonkey [107, 157]. Contrairement aux tables de hachage distribuées, ces réseaux ont été initialement conçus

par des groupes d'utilisateurs experts, avant d'être étudiés en détail par la communauté scientifique. Ce phénomène explique l'absence de descriptions de référence pour ces systèmes. Néanmoins, chacun de ces systèmes a fait l'objet d'études approfondies, comme en témoignent les références bibliographiques citées ci-dessus.

Les réseaux non structurés sont particulièrement appréciés pour leur résistance à la dynamique des environnements pair-à-pair, ainsi que pour la simplicité des protocoles qui les maintiennent. En effet, l'absence de contraintes liées à une structure quelconque permet de relâcher certaines tâches complexes, comme le maintien d'index ou de structures de données réparties. La gestion des liens logiques entre les noeuds peut donc s'opérer de façon non déterministe et simple, ou en fonction d'une métrique qui est indépendante de la structure du réseau globale (par exemple la proximité géographique des noeuds). Cependant, par souci d'optimisation, la plupart des protocoles de gestion de réseaux logiques non structurés imposent certaines règles quant à la gestion des liens entre les noeuds. Ces règles garantissent que la topologie du réseau garde de bonnes propriétés malgré l'absence de structure et malgré la dynamique de l'environnement hôte. Un exemple de règle appliquée à la gestion de réseaux logiques non structurés est le maintien d'un degré de connectivité moyen pour chaque noeud du réseau. Cette règle garantit avec une grande probabilité, que le graphe représentant la topologie du réseau reste complètement connecté au cours du temps.

La gestion de réseau non structuré, dans le cadre des systèmes pair-à-pair, est généralement réalisée à l'aide de protocoles épidémiques. Ces protocoles servent de base à la contribution proposée dans cette thèse. C'est pourquoi nous avons décidé de développer leur présentation dans la suite de cet état de l'art.

Protocoles épidémiques : Dans le domaine des systèmes distribués, un protocole épidémique fait référence à un mode de communication, qui implique un échange non déterministe et répété d'informations entre les membres d'un même réseau. Le choix aléatoire est un élément clé des protocoles épidémiques et concerne, en général, le choix d'une paire de noeuds entre lesquelles une communication est établie. La répétition en est également un élément clé, car le principe d'un protocole épidémique consiste en un processus ininterrompu de choix de paires de noeuds et d'échange d'informations. Ce modèle de communication a pour effet de permettre à une information d'être propagée de façon épidémique, c'est à dire rapidement et sur un très grand nombre d'entités, comme le serait une maladie contagieuse au sein d'une grande population. L'information est tout d'abord distribuée à un petit groupe de noeuds, qui à leur tour infectent d'autres noeuds, jusqu'à ce qu'une grande partie du réseau soit infecté par cette information.

Ces protocoles sont apparus en 1987 lors des travaux de Demers et coll. [57]. Ils étaient alors utilisés pour maintenir la cohérence des informations entre les différents répliquas d'une base de données distribuée. Depuis les travaux de Demer et coll., les protocoles épidémiques ont été largement étudiés et utilisés par la communauté scientifique pour produire une gamme de service très large, allant de la construction et du maintien de réseaux logiques non structurés [24, 60, 76, 84, 95, 149, 153] à la détection de panne [134, 132], en passant par la synchronisation de pairs [80, 32] (Cf. Section 3.2). Ces protocoles sont particulièrement appréciés pour leur simplicité et pour leur capacité à diffuser une grande quantité d'informations à travers une communauté potentiellement très large, et ce, malgré une dynamique forte de l'environnement hôte.

En 2007 Kermarrec et coll. a défini un algorithme générique pour la représentation de protocoles épidémiques [96]. Cet algorithme décrit les actions exécutées par deux pairs, P et Q, lors d'un échange d'informations initié par le pair P. Son pseudo-code est donné dans la

Figure 3.4. Cet algorithme décrit une routine, c'est-à-dire une séquence d'instructions exé-

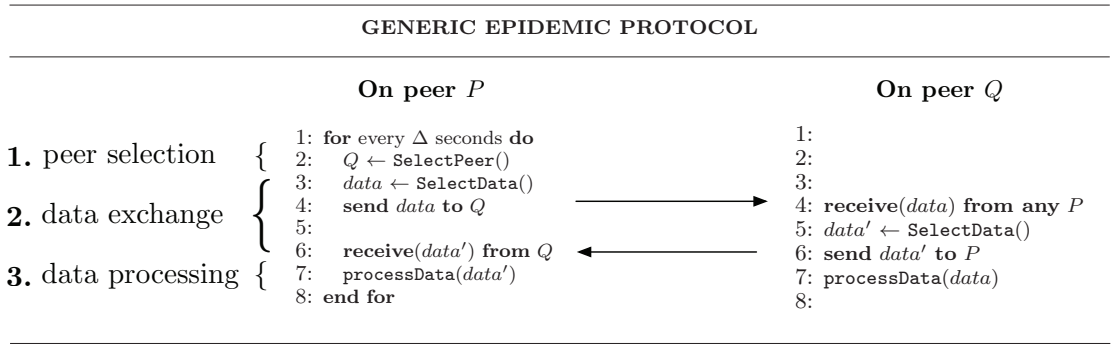


FIGURE 3.4 – Pseudo-code d'un protocole épidémique générique.

cutée périodiquement selon une période donnée Δ par le pair P. Il décrit la façon dont les communications sont établies entre deux pairs dans un système utilisant un protocole épidémique. Ce protocole comporte trois phases : tout d'abord, la sélection du partenaire par P (ligne 2), puis l'échange d'informations (lignes 3 à 6) et enfin, le traitement des informations échangées (ligne 6). La méthode `SelectPeer()` est généralement la même pour tous les protocoles épidémiques, son rôle est de retourner l'identifiant d'un pair choisi aléatoirement parmi les membres du système pair-à-pair sur lequel le protocole épidémique est exécuté. Cette méthode est typiquement implantée par un protocole d'échantillonnage de pairs (Cf. paragraphe suivant). Les méthodes `selectData()` et `processData(`*data*`)`, respectivement responsables de l'échange et du traitement d'informations entre deux noeuds donnés, sont spécifiques à l'application réalisée par le protocole épidémique en question. La suite de cette section présente une catégorie de protocoles épidémiques appelés protocoles d'échantillonnage de pairs.

Service d'échantillonnage de pairs : Dans un système pair-à-pair non structuré, le service d'échantillonnage de pairs est en charge de construire, puis de maintenir un réseau virtuel non structuré. C'est un service collaboratif, qui est réalisé de façon totalement distribuée par tous les pairs du système. Il consiste à maintenir la connectivité du réseau en fournissant aux noeuds des échantillons uniformes d'identifiants, qui correspondent aux autres membres du réseau. Le principe de ce service est de créer de nouveaux liens logiques au sein du réseau virtuel pour intégrer les pairs qui joignent le système, et de retirer les liens obsolètes qui pointent vers des pairs défaillants ou déconnectés du réseau. Ce document se focalisera sur la présentation des services d'échantillonnage de pairs basés sur le principe d'échange de vues locales, en raison de leur capacité reconnue à fournir des échantillons de pairs uniformes [24, 60, 76, 84, 95, 149, 153]. Au sein de ces protocoles, chaque pair maintient une vue locale du système (c.-à-d. une liste contenant un sous-ensemble des noeuds appartenant au réseau) et mélange cette vue périodiquement avec celle d'un partenaire choisi au hasard. La topologie résultante est proche d'un graphe aléatoire, si bien qu'elle hérite de leurs propriétés de haute connectivité et de faible diamètre. En conséquence, les réseaux non structurés construits par ces protocoles sont très résistants à la dynamique des environnements hôtes, et ce, indépendamment de la taille du système.

Les protocoles d'échantillonnage de pairs basés sur le principe d'échange de vues sont des protocoles épidémiques. En ce sens, ils utilisent un mode de communication non déterministe

et répété. Un pseudo-code générique illustrant le principe d'un tel protocole est donné dans la Figure 3.5. De façon périodique, chaque noeud choisit un partenaire au hasard au sein

GENERIC PEER SAMPLING PROTOCOL

```
1: task Sampling()
2:   for every Δ seconds do
3:     Q ← SelectPeer(view)
4:     list ← SelectSubset(view)
5:     send(list) to Q
6:   end for
7: end

8: upon receive(list) do
9:   updataLocalView(view, list)
10: end upon
```

FIGURE 3.5 – Pseudo-code générique d'un protocole d'échantillonnage de pairs.

de sa vue locale (ligne 3). Puis il extrait un sous-ensemble de sa vue, grâce à la méthode `SelectSubset(`*view*`)` et envoi cette liste au partenaire (lignes 4 et 5). Lors de sa réception, le partenaire intègre cette liste d'identifiants au sein de sa vue locale, à l'aide de la fonction `UpdateLocalView`(*view*, *list*). On dit alors qu'il y a mélange des vues. Aussi, les différentes versions de services d'échantillonnage de pairs [24, 60, 76, 84, 95, 149, 153] diffèrent par leurs implantations des méthodes `SelectSubset(`*view*`)` et `UpdateLocalView`(*view*, *list*). Ces différentes implémentations peuvent, par exemple, adapter le service d'échantillonnage de pairs à la présence de noeuds connectés via des routeurs en utilisant le principe de translation d'adresse réseau (NAT [1]), comme c'est le cas pour le protocole Nylon [95] qui maintient des tables de routage sur chaque noeud en plus des vues locales. Ces tables de routage contiennent typiquement l'identifiant de pairs qui peuvent servir de relais pour établir une communication avec un pair connecté via une translation d'adresse réseau. Aussi dans le cas de Nylon, le contenu des listes utilisées pendant le protocole d'échange de vues locales est enrichi avec des informations utiles au maintien des tables de routages, par exemple la présence de pairs pouvant servir de relais.

Par souci de simplification, l'échange décrit sur la Figure 3.5 représente une communication unilatérale et non un échange (le partenaire ne répond pas à l'envoi de la liste). Bien que certains protocoles d'échantillonnage utilisent des communications unilatérales, il apparaît que les protocoles utilisant des échanges bilatéraux offrent de meilleures performances [84]. Dans ces derniers protocoles, un noeud recevant une liste d'identifiants répond à l'expéditeur de cette liste en lui renvoyant un sous-ensemble de sa liste locale. On dit alors que les deux noeuds procèdent à un échange de vues partielles. Afin de fixer les idées sur le fonctionnement des services d'échantillonnage de pairs, un exemple graphique d'échange de vues partielles entre deux pairs P et Q est donné sur la Figure 3.6.

Ce dessin représente un protocole comportant trois étapes :

1. Le pair P sélectionne un noeud au hasard parmi sa vue locale (c'est le noeud Q).

2. Les deux pairs procèdent à l'échange de vues partielles.

3. La topologie du réseau non structuré est modifiée en fonction des mises à jour engendrées par cet échange (les vues sont mélangées).

Pour joindre un réseau maintenu par un protocole utilisant l'échange de vues, un noeud doit préalablement connaître au moins un membre de ce réseau. On appelle ce membre le

1. Network Translation Address.

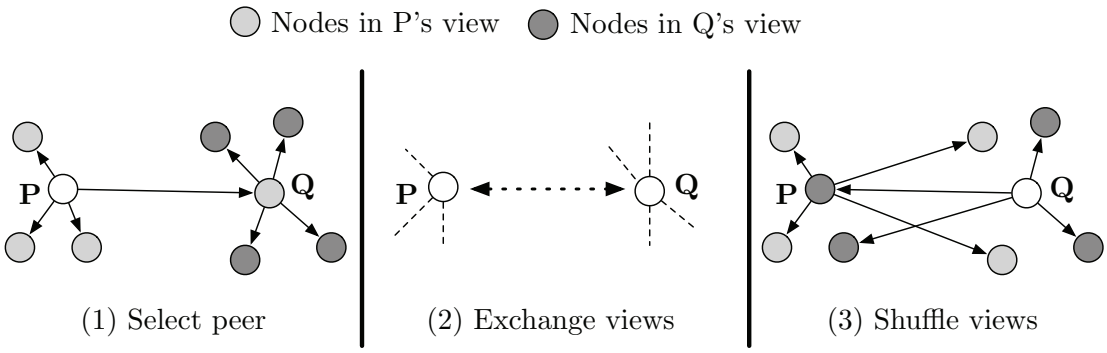


FIGURE 3.6 – Echange de vues partielles dans un protocole d'échantillonnage de pairs.

pair de démarrage. C'est ce pair de démarrage qui va pouvoir alimenter la vue du noeud souhaitant joindre le réseau. Une fois cette vue alimentée, l'introduction du nouveau noeud dans le réseau se fait naturellement, en suivant le protocole d'échange de vues. De façon opposée, lorsqu'un noeud quitte le réseau ou tombe en panne, les mécanismes de mises à jour des vues garantissent que son identifiant disparaîtra des vues locales des autres noeuds, en un temps borné. Lorsqu'un noeud souhaite créer un nouveau réseau virtuel, à l'aide d'un service d'échantillonnage de pairs, il exécute le protocole avec une vue locale vide, puis y ajoute les noeuds souhaitant joindre son réseau au fur et à mesure.

Les protocoles d'échantillonnage de pairs basés sur l'échange de vues ont fait l'objet d'analyses formelles analytiques [24, 76]. Ces analyses visent à valider les propriétés suivantes :

1. **Distribution uniforme des identifiants :** Le degré entrant de chaque noeud (c.-à-d. le nombre de vues qui contiennent son identifiant) est borné et la variance de ce degré sur l'ensemble des noeuds du système est faible.

2. **Echantillonnage uniforme :** La présence de l'identifiant d'un noeud au sein d'une vue est équiprobable pour tous les noeuds.

3. **Indépendance des échantillons :** Les ensembles d'identifiants contenus dans les vues sont indépendants dans le temps et dans l'espace. Intuitivement, cela signifie qu'un noeud P n'a pas plus de chance de faire partie de la vue d'un noeud Q, s'il a fait partie de la vue d'un des voisins de Q dans le passé.

Plus simplement, le maintien de ces propriétés au sein d'un système pair-à-pair assure la connectivité du réseau virtuel, et ce, sur des environnements très larges, non fiables et dynamiques. Les différentes études, analytiques et empiriques, menées sur ces services d'échantillonnage de pairs, ont montré que la connectivité des réseaux pair-à-pair non structurés était garantie avec une forte probabilité. Les hypothèses classiques de cette garantie sont un taux de pannes des communications compris entre 1% et 10% et une taille de vue locale logarithmiquement proportionnelle à la taille du système. Notons finalement, que les protocoles d'échantillonnage de pairs sont capables de maintenir la connectivité d'un réseau non structuré, avec une grande probabilité, en dépit d'événements catastrophiques entraînant une panne simultanée de près de 70% des noeuds du réseau [84].

En résumé, cette section a présenté les différents types de réseaux virtuels pair-à-pair, ainsi que les protocoles utilisés pour les maintenir. Ces réseaux sont divisés en deux grandes

catégories : les réseaux virtuels structurés et les réseaux virtuels non structurés. L'avantage des réseaux virtuels structurés est de pouvoir implanter, sur les systèmes qu'ils définissent, des mécanismes de routage et d'indexation performants. Cependant, le coût et la complexité de maintien de leur structure, au sein de larges environnements dynamiques, augmentent la difficulté de conception d'applications pair-à-pair. Les réseaux non structurés, eux, offre une bonne résistance à la dynamique des environnements pair-à-pair, et ce, à moindre coût en utilisant des protocoles épidémiques. La section suivante décrit les principaux services réalisés à l'aide de protocoles épidémiques.

3.2 Principaux services à base de protocoles épidémiques

Cette section établit un rapide survol des services pair-à-pair, qui sont construits sur la base de protocoles épidémiques. Ces protocoles sont déployés sur des réseaux virtuels non structurés et utilisent un service d'échantillonnage de pairs, similaire à ceux décrits dans la section précédente. A noter qu'une des particularités de ces services est de fournir des garanties statistiques sur les résultats attendus. En effet, la caractéristique non déterministe des protocoles épidémiques permet, en général, de garantir la réalisation d'un service donné avec une grande probabilité. Les protocoles épidémiques présentés dans cette section sont utilisés par la plateforme Salute (Cf. Chapitre 5). En particulier, les services de diffusion d'informations, d'agrégation de données et de synchronisation des noeuds sont décrits en détail.

3.2.1 Protocoles de diffusion épidémique

Les protocoles de diffusions épidémiques sont les premiers services bâtis sur le principe des protocoles épidémiques apparus au sein des systèmes pair-à-pair [57, 60, 42, 94]. Ils sont conçus pour diffuser des informations à travers de larges réseaux virtuels non structurés et dynamiques, typiquement ceux maintenus par un service d'échantillonnage de pairs.

Le principe d'une diffusion épidémique peut être résumé comme suit : lorsqu'un pair veut diffuser un message de façon épidémique, il envoie ce même message à un nombre constant $\mathcal{F}$ de pairs, choisis au hasard parmi les membres de son réseau virtuel. Lorsqu'un pair reçoit un message pour la première fois, on dit qu'il est infecté. Il réexpédie alors, lui aussi, ce message à $\mathcal{F}$ pairs choisis aléatoirement. Si un noeud reçoit un message par lequel il a déjà été infecté, alors il l'ignore et ne le retransmet pas ; c'est ce qu'on appelle le principe « infecter et mourir » (de l'anglais « infect and die »). Le message est donc propagé de façon épidémique, en suivant le modèle de réexpédition conditionnelle énoncé ci-dessus. La diffusion s'achève naturellement, lorsque les réexpéditions de messages ne touchent plus de pairs n'étant pas encore infectés. A noter qu'une diffusion est dite atomique lorsque tous les pairs du réseau de diffusion ont reçu ce message. La Figure 3.7 présente un exemple de diffusion épidémique pour un système composé de 14 noeuds, dont chaque pair possède trois connexions vers les autres membres de son réseau.

Dans les protocoles de diffusion épidémique, le seul élément de configuration est le paramètre $\mathcal{F}$, appelé *fanout*, qui indique le nombre de pairs aléatoirement choisis lors de la réexpédition d'un message. En 2004, Eugster et coll. proposent d'évaluer analytiquement, puis expérimentalement, les performances des protocoles pair-à-pair de diffusion épidémique [61]. Dans un premier temps Eugster et coll. s'inspirent d'études appartenant au domaine de la théorie des graphes pour établir les relations suivantes, en considérant un protocole de diffusion épidémique sur un système pair-à-pair non structuré de taille N :

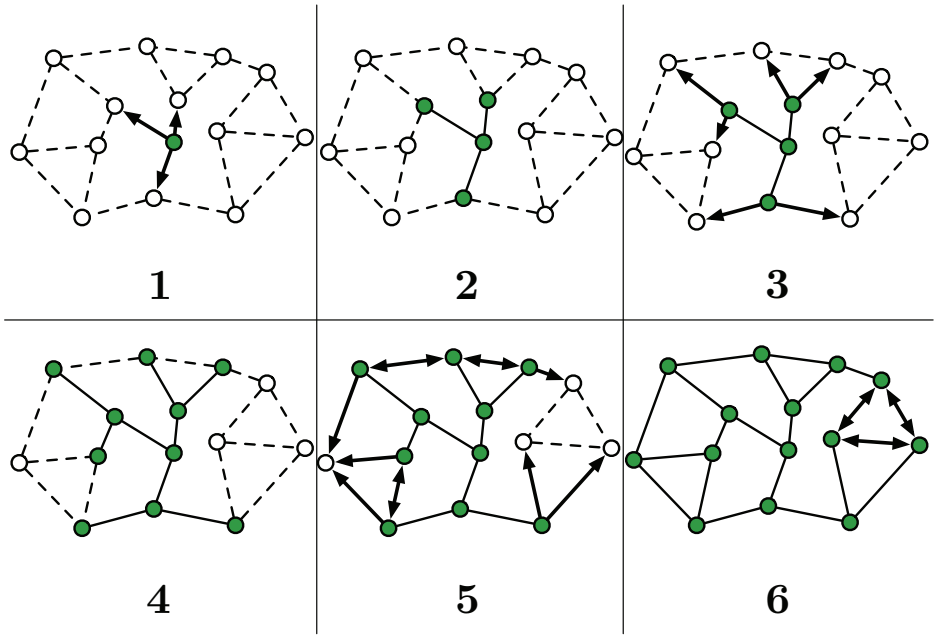


FIGURE 3.7 – Exemple de diffusion épidémique sur un réseau virtuel non structuré.

- La proportion de noeuds Π ayant reçu l'information, une fois la diffusion épidémique terminée, satisfait l'équation :

$$\Pi = 1 - e^{-\Pi F}$$

  Cette relation indique que, pour une valeur fixe du fanout, la proportion de noeuds touchés par une diffusion épidémique est indépendante de la taille du système.

- La probabilité P_a qu'une diffusion épidémique soit atomique satisfait la relation suivante :

$$P_a = e^{e^{-c}}, \text{ où } F = \log(N) + c$$

  Cette relation montre notamment qu'il existe un changement de comportement lorsque le paramètre c devient positif, c'est à dire lorsque la valeur du fanout dépasse $\log(N)$.

- Soit une diffusion épidémique atomique réalisée en utilisant un fanout F de l'ordre de $\log(N)$. La latence de cette diffusion en nombre de réexpédition R satisfait l'équation :

$$R = \frac{\log N}{\log(\log(N))} + \mathcal{O}(1)$$

  Ce résultat indique qu'une diffusion épidémique a besoin, au plus, d'un nombre logarithmique d'étapes par rapport à la taille du système, pour atteindre tous les pairs.

Les premiers résultats expérimentaux confirmant ces relations théoriques sont ceux présentés dans les travaux de Kermarrec et coll. [94]. Ces résultats proviennent de simulations de diffusion épidémique sur des réseaux comportant 10 000 et 50 000 pairs. Pour des raisons de simplification, nous ne présenterons que les résultats correspondant au réseau de 50 000 pairs au sein de ce document. Les résultats de cette évaluation sont reportés sur la Figure 3.8. Les barres noires représentent le nombre, parmi 100 exécutions, de diffusions atomiques. Les

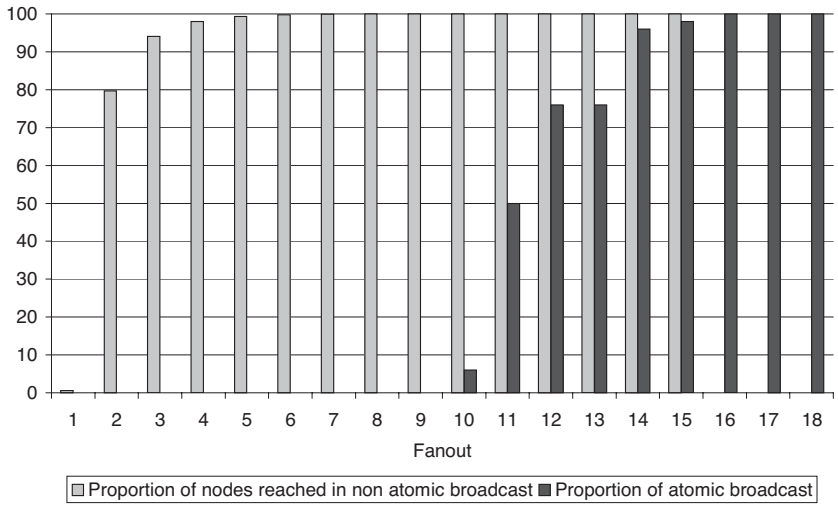


FIGURE 3.8 – Evaluation d'une diffusion épidémique en fonction du fanout, sur un réseau pair-à-pair non structuré composé de 50000 noeuds. D'après l'étude de Kermarrec et coll. [94].

barres claires représentent le pourcentage de pairs infectés lorsque la diffusion n'était pas atomique. La variance des mesures n'est pas représentée sur le graphique, car elle considérée, par les auteurs, comme négligeable. On peut alors distinguer trois régions sur le graphique représenté au sein de la Figure 3.8. Premièrement, pour les valeurs faibles du fanout (de 1 à 4), aucune diffusion n'est atomique et le pourcentage de noeuds touchés par la diffusion varie subitement d'une valeur quasi nulle à une valeur proche de 100%. Ensuite, pour les valeurs intermédiaires du fanout (de 5 à 15) le pourcentage de noeuds touchés reste proche de 100% et le nombre de diffusions atomiques varie de 0 à 100. Finalement, si le fanout est suffisamment grand (supérieur à 15), alors toutes les diffusions sont atomiques. Cas résultats sont cohérents avec l'analyse théorique présentée précédemment.

Dans un second temps, Kermarrec et coll. a procédé à la même évaluation, en fixant le fanout à une valeur de 15 et en intégrant un nombre variable de pairs tombant en panne pendant la diffusion. Les pairs dits « en panne » ne réexpédient pas les messages reçus lors de la diffusion épidémique. Cette nouvelle évaluation a pour but d'étudier les performances d'une diffusion épidémique lorsqu'elle est exécutée sur des environnements dynamiques au sein desquels les noeuds peuvent présenter des défaillances, par exemple des environnements pair-à-pair. Les résultats de cette évaluation sont illustrés sur la Figure 3.9. Ces résultats montrent qu'une diffusion épidémique est fiable en dépit des pannes potentiellement rencontrées lors de son exécution. Le nombre de diffusions atomiques décroît de façon abrupte lorsque le nombre de pairs défaillants dépasse les 20%. Néanmoins, la diffusion épidémique réussit à toucher une très large proportion des membres du réseau de diffusion (supérieure à 99%), même si la moitié des pairs du système présentent des défaillances. Pour conclure, les protocoles de diffusion épidémique sont des moyens fiables de diffuser une information sur des réseaux pair-à-pair non structurés de grande taille, en dépit de la dynamique de l'environnement hôte.

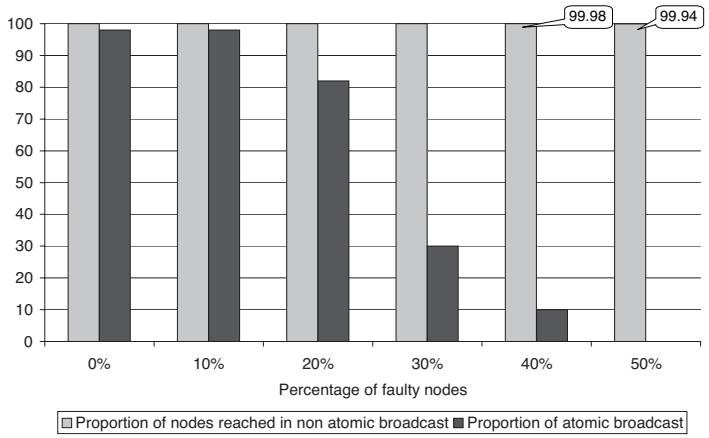


FIGURE 3.9 – Evaluation d'une diffusion épidémique en présence de noeuds défaillants. D'après l'étude de Kermarrec et coll. [94].

3.2.2 Agrégation de données et estimation de taille de réseaux

Les protocoles d'agrégation de données sont utilisés pour calculer des propriétés globales sur des réseaux pair-à-pair. On parle d'agrégation, car les résultats de ces calculs proviennent de l'agrégation de valeurs qui sont prélevées sur un échantillon généralement grand des pairs du système. Par exemple, un protocole d'agrégation de données peut estimer la charge de calcul moyenne ou maximale exercée sur les pairs d'un réseau, ou encore le nombre moyen de noeuds présents au sein du système. Ces valeurs sont typiquement utilisées pour surveiller l'évolution d'un système pair-à-pair, par exemple pour surveiller la variation de la charge de travail moyenne. Cette section présente également des adaptations de protocoles d'agrégation de données qui permettent d'évaluer la taille d'un réseau pair-à-pair non structuré de façon fiable et en dépit de la dynamique de l'environnement hôte.

Il existe de nombreux protocoles d'agrégation de données pour les systèmes pair-à-pair [93, 86, 97, 110, 116]. Certains de ces protocoles sont focalisés sur le calcul d'une valeur particulière, par exemple la taille moyenne du réseau pair-à-pair [97, 110, 116]. D'autres sont plus génériques et permettent de calculer des valeurs statistiques telles que des valeurs maximales ou des moyennes sur un attribut donné des pairs du réseau [93, 86]. Au sein de cet état de l'art nous allons détailler le protocole épidémique d'agrégation de données proposé par Jelasity et Montresor en 2004 [86], que nous utilisons pour la contribution proposée dans le Chapitre 5.

Ce protocole possède deux particularités : premièrement, c'est un protocole épidémique et ensuite, lorsqu'il est exécuté sur un réseau pair-à-pair non structuré, il fournit les résultats des calculs d'agrégation à tous les membres du système. Le fonctionnement de ce protocole peut être décrit en suivant le schéma d'exécution d'un protocole épidémique générique (Cf. Figure 3.4). Pour ce faire, nous allons considérer un scénario dans lequel le protocole d'agrégation calcule la valeur moyenne $\overline{v}$ d'un attribut donné des pairs du système, par exemple l'espace de stockage disponible de chacun. Initialement, la valeur v_i correspondant à l'estimation de $\overline{v}$ par le pair P_i est fixée telle que $\overline{v} = s_i$, où s_i est la valeur correspondant à l'espace de stockage disponible sur le pair P_i. Le protocole d'agrégation de données s'exécute alors de la façon suivante :

– **Sélection d'un partenaire :** en suivant le modèle d'exécution des protocoles épidémiques et dans le cadre d'un échange d'information périodique, chaque noeud choisit

un partenaire au hasard parmi les membres du système. Ce choix peut typiquement être fait en utilisant un protocole d'échantillonnage de pairs exécuté parallèlement au protocole épidémique d'agrégation de données, sur le même réseau virtuel.

- **Echange de données :** deux pairs, par exemple P_i et P_j, s'échangent les valeurs v_i et v_j correspondant à l'évaluation de la moyenne $\overline{v}$ respectivement réalisée par P_i et P_j, jusqu'au moment de l'échange en cours.

- **Traitement des données :** les valeurs v_i et v_j sont ensuite mises à jour en suivant la relation suivante : $v_i, v_j \leftarrow (v_i + v_j)/2$.

Dans leurs travaux, Jelasity et Montresor donnent une analyse théorique de ce protocole d'agrégation [86]. Cette analyse établit que le facteur de convergence ρ_k, de la valeur $\overline{v}$ après l'exécution de k étapes du protocole épidémique, satisfait l'équation suivante pour tout nombre k d'étapes supérieur à 1 : $\rho_i \approx 1/(2\sqrt{e})$. En d'autres termes, lors de chaque cycle la variance espérée pour une estimation locale v_i de la valeur moyenne $\overline{v}$ est réduite par un facteur $2\sqrt{e}$. Il apparaît alors que le protocole converge de façon exponentielle et qu'une approximation précise de $\overline{v}$ peut être atteinte en quelques cycles, indépendamment de la taille du réseau. Ce qui justifie, de façon théorique, les bonnes propriétés de passage à l'échelle du protocole.

Ces auteurs ont proposé un protocole d'estimation de taille du réseau, également appelé protocole de comptage, qui est basé sur le protocole d'agrégation décrit ci-dessus. On utilise le nom « *Average* » comme référence à ce protocole dans la suite du document. Le principe de ce protocole est inspiré par la réflexion suivante : s'il existe strictement un noeud possédant la valeur 1 et que tous les autres possèdent la valeur 0, alors la moyenne exacte de ses valeurs sur un réseau de N pairs est $1/N$. La taille du réseau peut donc être déduite directement de la moyenne de ces valeurs, laquelle pouvant être estimée à l'aide du protocole épidémique d'agrégation de données. L'implantation de ce protocole de comptage implique que l'on soit capable de choisir un noeud particulier possédant la valeur unique 1, ce qui va à l'encontre du principe de décentralisation généralement appliqué aux protocoles pair-à-pair. Pour remédier à ce problème, Jelasity et Montresor ont autorisé l'exécution concurrente de plusieurs instances du protocole épidémique. En suivant ce principe, chaque pair déclenche une nouvelle exécution du protocole *Average* selon une probabilité faible, de l'ordre de $1/100$. Les pairs du système répètent cette exécution conditionnelle selon une période Δ_{Count} telle que $\Delta_{Count} >> \Delta$. Il existe alors, en moyenne, $N/100$ exécutions concurrentes du protocole de comptage sur le réseau ciblé, où N est la taille de ce réseau. Les estimations résultant des exécutions concurrentes peuvent ensuite être utilisées pour affiner l'approximation statistique de la valeur N.

En 2009, une amélioration a été apportée au protocole *Average*, cette nouvelle version du protocole de comptage est appelée *T-Size* [121]. Elle consiste à combiner l'exécution du protocole *Average* avec celle du protocole épidémique *T-Man* [83], initialement conçu pour gérer la topologie du réseau au sein des systèmes pair-à-pair non structurés. Le but de cette amélioration est de stabiliser la vitesse de convergence et la précision du protocole de comptage en présence d'une dynamique forte de l'environnement hôte. Le principe du protocole *T-Size* est décrit dans la suite sous forme de quatre étapes successives :

1. Lors de l'initialisation du protocole, chaque noeud possède un identifiant unique p et choisit une valeur aléatoire v_p au sein de l'espace d'adressage circulaire $\mathcal{I} = \{0, ..., M-1\}$. Pour un couple d'identifiant arbitraire $x, y \in \mathcal{I}$, la distance de x à y notée $d(x, y)$ est définie par la relation suivante, où la valeur M est un paramètre du protocole :

$$d(x, y) = (y - x) \mod M$$

2. T-Man est utilisé pour construire un anneau virtuel ordonné selon l'ensemble des valeurs v_p, en utilisant la fonction de distance d, si bien qu'une fois l'anneau construit, chaque pair p connaît l'identifiant q de son successeur sur l'anneau virtuel ainsi que sa valeur v_q. D'après les propriétés du protocole T-Man [83], non détaillées ici par soucis de concision, un pair q est le successeur d'un pair p si et seulement si il n'existe aucun pair r suivant le noeud p et étant plus proche de p que q. En d'autres termes, la relation entre un pair et son successeur sur l'anneau satisfait la relation suivante :

$$succ(p) = q \Leftrightarrow \nexists r \in \mathcal{P} : d(v_p, v_r) < d(v_p, v_q)$$

3. Chaque pair p initialise une variable locale a_p telle que $a_p = d(v_p, v_{succ(p)})$. En considérant qu'aucun noeuds de tombe en panne, la relation suivante est alors vérifiée :

$$\sum_{p \in \mathcal{P}} a_p = M$$

4. Finalement, le protocole *Average* est exécuté pour calculer la valeur moyenne $\overline{a}$:

$$\overline{a} \approx (\sum_{p \in \mathcal{P}} a_p)/N = M/N$$

   A l'issue de l'exécution de ce protocole, tous les membres du réseau connaissent une approximation de la valeur $\overline{a}$. La valeur M étant un paramètre du protocole *T-Size*, chaque pair est alors en mesure de déduire une estimation de la taille N du système à partir de la valeur $\overline{a}$ d'après l'équation $N = M/\overline{a}$.

De façon intuitive, *T-Size* résiste mieux à la dynamique des systèmes pair-à-pair, car le calcul de la moyenne $\overline{a}$ est peu impacté lorsqu'un des noeuds du système tombe en panne ou quitte le réseau. En revanche dans la version initiale du protocole *Average*, si le noeud possédant la valeur unique 1 tombe en panne ou quitte le réseau avant la fin de l'exécution du protocole d'agrégation, alors l'estimation de la taille du système peut être sévèrement faussée. Dans un cas extrême, si le noeud possédant la valeur unique tombe en panne lors du premier échange d'information exécuté par le protocole épidémique d'agrégation, alors l'estimation de la taille du système sera inexacte. Afin d'alimenter cette théorie, Montresor et coll. a réalisé une série d'évaluations à l'aide du simulateur PeerSim [89] et en utilisant une couche de transport qui émule les délais point à point entre chaque couple de noeuds d'après les données recueillies au sein des traces de King [74]. Chacune des expériences présentées ici est répétée 50 fois en utilisant des graines différentes pour le générateur de nombre aléatoire du simulateur. Ce générateur aléatoire est utilisé pour simuler l'exécution des protocoles épidémiques, par exemple pour choisir un partenaire au hasard lors des échanges d'informations qui sont au coeur de ces protocoles. Lorsque cela est possible, le résultat de chaque simulation est illustré par un point ; si trop de points se superposent, ils sont séparés par un décalage léger selon l'abscisse du graphique en question. Finalement, les évaluations présentées dans ce document concernent l'erreur d'estimation des protocoles de comptage en fonction de la dynamique du système pair-à-pair, ainsi que leur temps de convergence et leur surcoût en bande passante généré, le tout en fonction de la taille du système.

La Figure 3.10 donne les performances des protocoles *T-Size* et *Average* en fonction de la dynamique de l'environnement hôte. Le système utilisé pour produire ces résultats est composé de 2^{16} noeuds. La dynamique des pairs est donnée en pourcentage de défaillance par pair par seconde. Ce pourcentage varie de 0% à 1% par incrément de 0,1%. Les auteurs soulignent que ce scénario est pessimiste dans le sens ou la dynamique moyenne mesurée au sein des systèmes

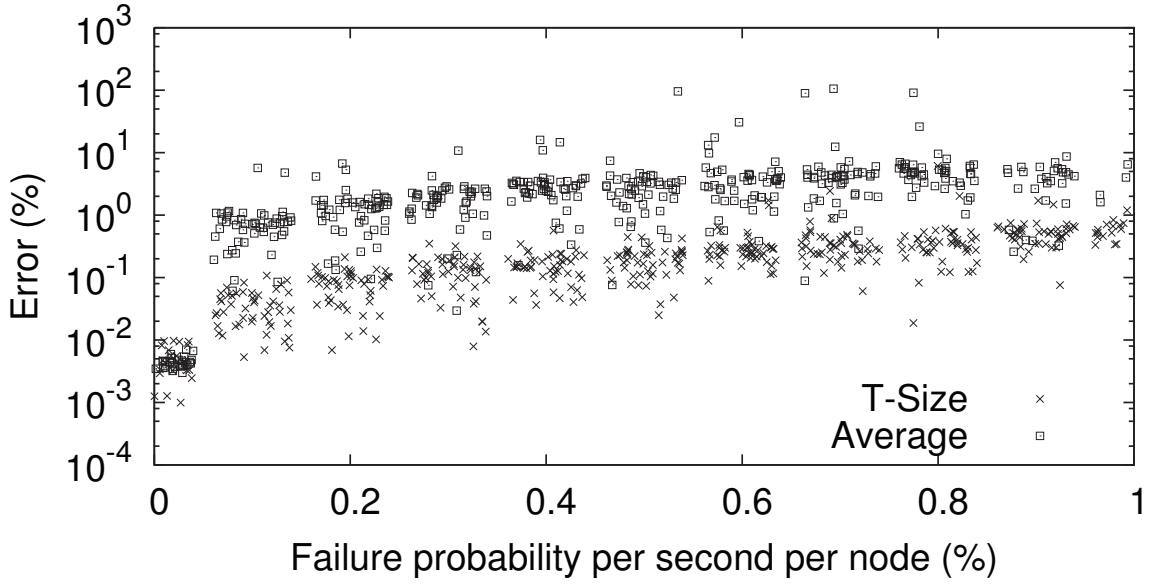


FIGURE 3.10 – Evaluation de l'erreur d'approximation de la taille d'un réseau pair-à-pair, par les protocoles de comptage *T-Size* et *Average*. D'après les travaux de Montresor et coll. [121].

pair-à-pair est généralement proche de 0,01%. En effet, un pourcentage de défaillance de 1% par pair et par seconde implique que la durée moyenne de session espérée est de 99s. En guise de comparaison, la durée moyenne de session espérée dans le système Skype [73] est de 10 heures, soit environ 360 fois supérieure à celle simulée dans le pire des scénarios utilisé dans cette expérience. Néanmoins, face à cette forte dynamique le protocole *T-Size* parvient à maintenir la valeur de l'erreur d'estimation en dessous de 7% pour les cas extrêmes et en dessous de 2% dans la majorité des cas. Il est important de noter que la réduction de l'erreur moyenne par le protocole *T-Size* est nette par rapport à l'erreur moyenne obtenue par le protocole *Average*. On notera également qu'en présence d'un fort taux de churn, le protocole *Average* présente sporadiquement des erreurs pouvant atteindre 100% de la taille réelle. Finalement, lorsque la dynamique de l'environnement hôte est faible, les deux protocoles parviennent à évaluer la taille du réseau pair-à-pair avec une précision de l'ordre de 0,01%.

La Figure 3.11 montre le temps de convergence et le surcoût en bande passante des protocoles *T-Size* et *Average* en fonction de la taille du système pair-à-pair sur lequel ils sont exécutés. Il apparaît alors que le protocole *T-Size* permet un passage à l'échelle logarithmique pour des tailles de système variant de 2^{10} à 2^{18}, du point de vue du temps de convergence comme du point de vue de la bande passante consommée par le protocole de comptage. Le surcoût en bande passante associé au protocole *T-Size* varie de 3 KO/s pour une taille de réseau de 2^{10} à 5 KO/s pour une taille de réseau de 2^{18}. Le temps de convergence associé au protocole *Average* est plus long, cependant ce dernier présente un coût constant en bande passante qui plafonne à environ 1 KO/s, indépendamment de la taille du système. Ces deux protocoles offrent donc des services de comptage de pairs fiables pour un coût raisonnable, et ce, malgré la dynamique de l'environnement hôte.

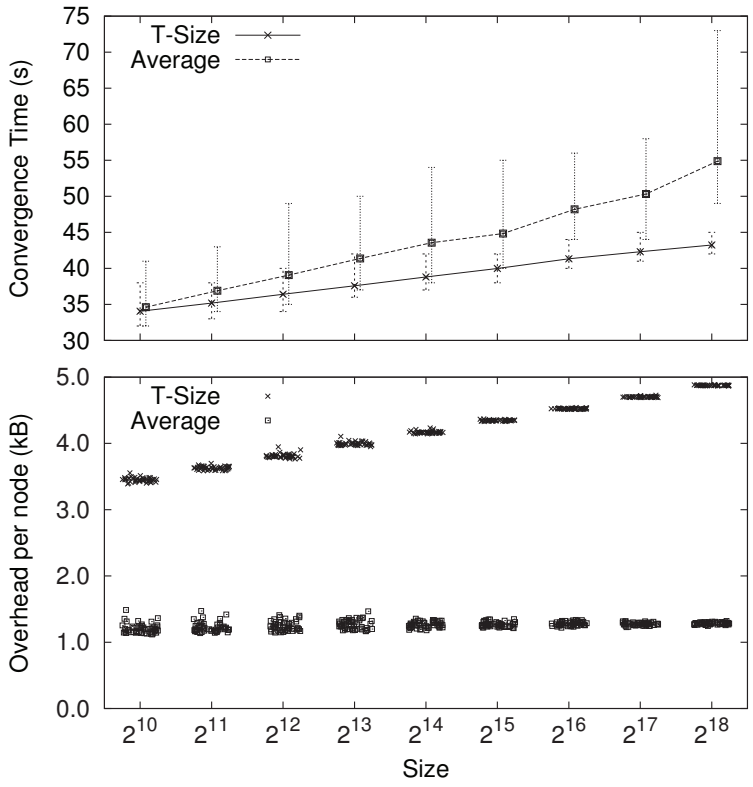


FIGURE 3.11 – Evaluation du temps de convergence et du surcoût en bande passante des protocoles de comptage *T-Size* et *Average*. D'après les travaux de Montresor et coll. [121].

3.2.3 Synchronisation de l'horloge des noeuds

La synchronisation des noeuds dans un système distribué est un service des plus utiles. En particulier, Babaoglu et coll. [32] remarque que la plupart des protocoles épidémiques font l'hypothèse que les noeuds du système évoluent de façon synchronisée, c'est-à-dire que les actions périodiques qui caractérisent ces protocoles ont lieux simultanément par rapport à la période Δ qui s'écoule entre deux communications. Or, peu de protocoles sont capables de fournir cette garantie sur des réseaux pair-à-pair non structurés, qui comportent un grand nombre de noeuds et qui sont souvent non fiables et très dynamiques. Pour les systèmes distribués plus classiques, par exemple pour des grappes de machine, on utilise généralement le protocole NTP (*« Network Time Protocol »*) [120, 118], afin de garantir que les horloges locales des noeuds sont synchronisées. Cependant, ce protocole de synchronisation utilise une structure hiérarchique qui comprend un certain nombre de serveurs fiables, c'est-à-dire de serveurs hautement disponibles. Néanmoins, dans le cadre d'environnements pair-à-pair hautement dynamiques, cette structure hiérarchique est difficile à maintenir avec des protocoles simples.

Pour remédier à ce problème, Iwanicki et coll. a proposé un protocole épidémique [80], appelée GTP pour *« Gossiping Time Protocol »*, qui est capable de synchroniser les horloges locales de chaque pair d'un réseau non structuré avec une horloge de référence appelée « source ». Le principe de ce protocole peut être décrit comme suit, en suivant le schéma d'exécution d'un protocole épidémique générique (Cf. Figure 3.4) :

– **Sélection d'un partenaire :** en suivant le modèle d'exécution des protocoles épidémiques et dans le cadre d'un échange d'information périodique, chaque noeud choisit un partenaire au hasard parmi les membres du système. Ce choix peut typiquement être fait en utilisant un protocole d'échantillonnage de pairs exécuté parallèlement au protocole épidémique de synchronisation, sur le même réseau virtuel.

– **Echange de données :** chaque couple de noeuds échange des estampilles horaires, produites par leurs horloges locales. Cet échange d'horloge suit l'algorithme de synchronisation de Cristian [53], qui permet à un noeud A de synchroniser sont horloge avec celle d'un noeud B. Le protocole de synchronisation de Cristian est décrit sur le schémas de la Figure 3.12. Dans un premier temps, le noeud A enregistre l'estampille T_1^A

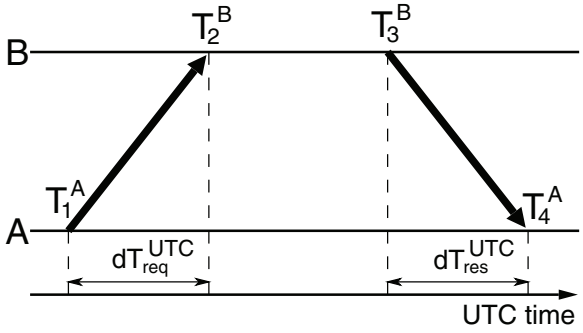


FIGURE 3.12 – Schéma d'échange d'estampilles horaires, entre deux noeuds A et B, dans le protocole « Gossiping Time Protocol ». D'après les travaux d'Iwanicki et coll. [80].

correspondante à la valeur de son horloge locale, puis il envoie un message contenant la valeur T_1^A au noeud B. Le noeud B enregistre l'estampille T_2^B, d'après son horloge locale, dès la réception du message, puis commence à préparer un message de réponse contenant l'estampille enregistrée. Lorsque ce message est près, B enregistre une nouvelle estampille T_3^B, l'incorpore dans le message de réponse et envoie le tout au noeud A. A la réception du message de réponse, A enregistre l'estampille T_4^A. Grâce à l'ensemble d'estampilles $\{T_1^A, T_2^B, T_3^B, T_4^A\}$, appelé échantillon de synchronisation, le noeud A peut calculer le délai δ de l'échange de message :

$$\delta = T_4^A - T_1^A - (T_3^B - T_2^B) = T_4^A - T_1^A + T_2^B - T_3^B, \qquad (1)$$

ainsi que le décalage θ de l'horloge du noeud B :

$$\theta = T_3^B + \tfrac{\delta}{2} - T_4^A = \tfrac{T_2^B - T_1^A + T_3^B - T_4^A}{2}. \qquad (2)$$

En utilisant la valeur θ, A peut synchroniser son horloge avec celle de B, soit en modifiant temporairement la fréquence de son horloge (ajustement graduel), soit en redémarrant directement son horloge à la date appropriée (ajustement immédiat). Dans le protocole GTP, chaque noeud initie un tel échange d'estampille de façon périodique, conformément au modèle des protocoles épidémiques. Cependant, ces mêmes noeuds n'utilisent les valeurs calculées d'après cet échange, si et seulement s'ils estiment que ces valeurs peuvent améliorer la précision d'une de leurs horloges par rapport à l'horloge source.

– **Traitement des données :** si un noeud pratique un échange d'estampilles horaires avec la source, alors il synchronise systématiquement son horloge sur celle de la source. On dit que ce noeud possède une synchronisation distante d'un saut par rapport à la source de temps. La distance de synchronisation est stockée au sein de la variable locale H_A. Cette variable prend la valeur 1 lorsque le noeud A synchronise son horloge directement avec la source. Si au contraire le noeud A synchronise son horloge sur celle d'un noeud B quelconque, alors cette variable est mise à jour en suivant la relation $H_A = H_B + 1$. Lorsqu'un noeud A procède à un échange d'estampille avec un noeud B différent de la source de temps, les deux noeuds utilisent une métrique de dispersion pour évaluer laquelle des deux horloges, entre celle de A et celle de B, possède la meilleure synchronisation, c'est-à-dire laquelle de ces horloges possède le plus petit décalage par rapport à l'horloge de la source. En supposant que chaque noeud connaît la précision et la tolérance de fréquence de son horloge, le noeud A calcule la dispersion λ de l'échantillon obtenu via l'échange épidémique avec le noeud B selon l'équation suivante :

$$\lambda = \epsilon^B + \rho^B + |\theta^B| + (T^B_{NOW} - T^B_{LU}).\phi^B + \tfrac{\delta}{2}, \qquad (3)$$

où ϵ^B est la dispersion estimée du noeud B, ρ^B la précision de l'horloge de B, θ^B la correction en cours de l'horloge de B (si B utilise la méthode d'ajustement graduel), T^B_{NOW} correspond à la valeur actuelle de l'horloge de B,T^B_{LU} est la date de la dernière synchronisation de l'horloge de B, ϕ^B la tolérance de fréquence de l'horloge de B, et où δ est le délai de l'échange de message. Ainsi, la dispersion de l'échantillon comprend plusieurs types d'erreurs pouvant causer la désynchronisation d'une horloge par rapport à la source. Le noeud A décide alors d'utiliser l'échantillon de synchronisation si et seulement si la condition suivante est respectée :

$$\lambda \leq \epsilon^A + \rho^A + (T^A_{NOW} - T^A_{LU}).\phi^A \textbf{ et } (\theta^A = 0 \textbf{ ou } H^A > H^B) \qquad (4)$$

La première inégalité détermine si l'utilisation de l'échantillon peut améliorer la synchronisation du noeud A avec la source de temps. La seconde heuristique permet de démarrer la synchronisation seulement s'il n'existe aucun ajustement d'horloge en cours et si la distance de synchronisation de B est inférieure à celle de A. A noter que la dispersion ϵ^A est arbitrairement considérée comme infinie lors de l'initialisation du protocole et que la dispersion de l'horloge de la source de temps est nulle.

Le protocole GTP a été évalué par Iwanicki et coll. à l'aide d'une technique d'émulation de réseau permettant de recréer un environnement de type pair-à-pair sur une grappe composée de 72 machines [80]. Cette évaluation considère trois tailles de système variant de 1 500 à 64 500 noeuds. Les résultats intéressants pour le cadre de cette thèse sont illustrés sur la Figure 3.13. Le temps de convergence du protocole en fonction de la taille du réseau est donné sur la sous-figure de gauche, alors que le décalage moyen des horloges des noeuds par rapport à celle de la source est donné sur la sous-figure de droite. Il apparaît alors que le protocole GTP permet de synchroniser les horloges d'un nombre important de noeuds, selon un temps logarithmiquement proportionnel à la taille du système, et ce, avec une précision de l'ordre d'une dizaine de millisecondes.

Pour conclure, un certains nombres de services basés sur l'utilisation d'un protocole épidémique ont été détaillés dans cette section. Cependant, il est important de noter que les protocoles épidémiques sont également utilisés pour développer une large gamme de services génériques, non détaillés dans ce document, tels que des ramasse-miettes distribués [75], ou

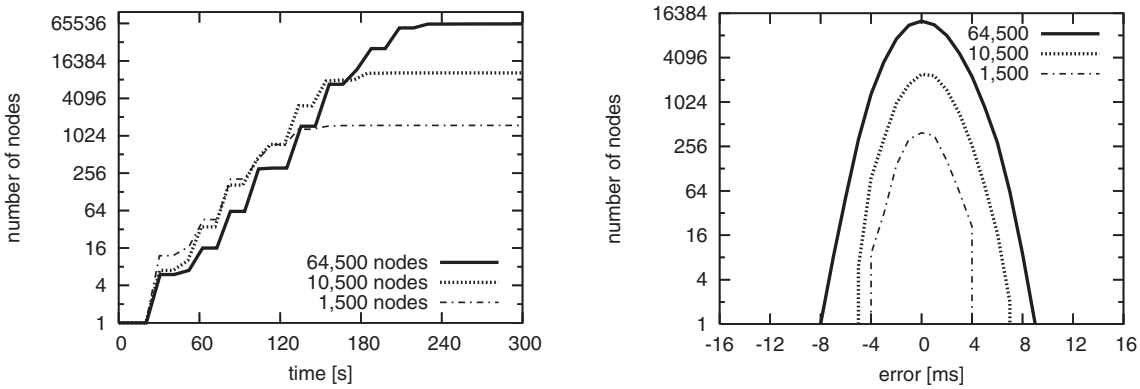


FIGURE 3.13 – Evaluation du temps de convergence et de l'erreur moyenne du protocole « Gossiping Time Protocol ». D'après les travaux d'Iwanicki et coll. [80].

encore des détecteurs de pannes [134, 132, 55]. La section suivante présente également un sous-ensemble des protocoles épidémiques, les protocoles de construction de réseau virtuel superposé et les protocoles de morcellement.

3.3 Protocoles épidémiques et gestion de réseaux multicouches

Cette section présente tous les protocoles épidémiques dont la vocation est de gérer la maintenance et la répartition de pairs, au sein de ce que l'on appelle des réseaux virtuels multicouches. Dans un système pair-à-pair, un réseau virtuel multicouche est un réseau composé d'un réseau principal, qui définit l'appartenance au système, et d'une ou plusieurs couches de réseaux superposés. Ces réseaux superposés définissent des sous-ensembles du système pair-à-pair sur lequel ils sont déployés. Leur utilisation permet, par exemple, d'assigner plusieurs tâches indépendantes aux membres d'un système pair-à-pair. Un exemple de réseau virtuel composé de deux couches est donné sur la Figure 3.14. On notera que tous les pairs

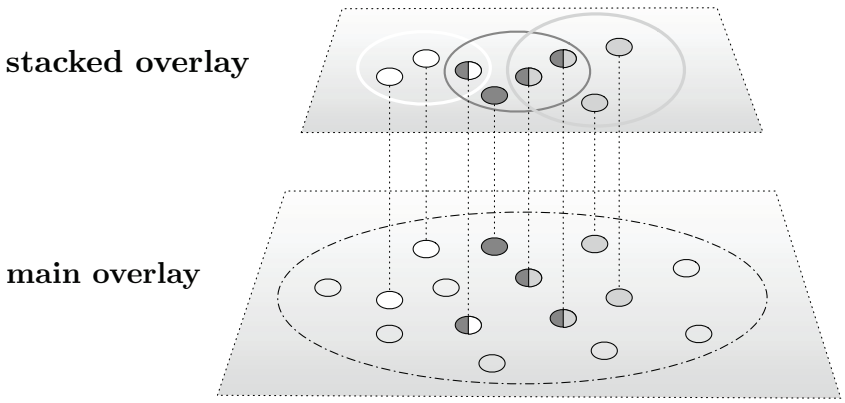


FIGURE 3.14 – Schémas d'un réseau virtuel multicouche.

membres d'un système pair-à-pair font obligatoirement partie du réseau principal, puisque c'est ce même réseau qui définit les limites du système. Cependant, un pair peut n'appartenir

à aucun réseau superposé, comme il peut appartenir à plusieurs de ces réseaux, y compris au sein de la même couche. Dans la suite de cette section, on distinguera deux sous catégories de protocoles de gestion de réseau virtuel multicouche : les protocoles de construction de réseau virtuel superposé, dont le but est de maintenir plusieurs réseaux virtuels superposés au sein d'un même système pair-à-pair, et les protocoles de morcellement de réseau virtuel, dont le but est de répartir les pairs du système entre les différents réseaux d'une même couche.

3.3.1 Construction de réseau virtuel superposé

L'objectif d'un protocole de construction de réseau virtuel superposé est de créer un réseau virtuel dit « secondaire » au dessus du réseau virtuel « principal » d'un système pair-à-pair, et ce, afin d'y rassembler un sous-ensemble des membres du système. Dans le cadre des protocoles épidémiques, ces protocoles font l'hypothèse que les réseaux virtuels principaux et secondaires sont maintenus grâce à des protocoles d'échantillonnage de pairs.

Les deux principaux protocoles épidémiques utilisés pour construire des réseaux virtuels multicouches sont Vicinity [154, 155, 152] et T-Man [83, 88]. Ces deux protocoles reposent sur l'utilisation d'un service d'échantillonnage de pairs. Vicinity utilise le service d'échantillonnage de pairs Cyclon [153], et T-Man utilise le service Newscast [149]. Ces deux services d'échantillonnage de pairs sont basés sur le principe de l'échange périodique de vue partielle présenté dans la Section 3.1.3. Pour simplifier, leur exécution consiste à maintenir à jour une vue locale composée d'un échantillon des pairs du système, et ce, malgré la dynamique de l'environnement hôte. L'utilisation des protocoles Cyclon et Newscast assure, avec une grande probabilité, que les identifiants des membres du système sont uniformément répartis au sein des vues de tous les pairs.

Dans les protocoles Vicinity et T-Man, les pairs possèdent une vue pour chaque couche du système, c'est-à-dire une vue pour le réseau principal et une vue pour le réseau secondaire. Le principe de ces protocoles est d'approvisionner la seconde vue (c.-à-d. la vue du réseau secondaire) avec les identifiants de noeuds stockés au sein de la vue principale. Lors de cet approvisionnement, certains filtres peuvent être appliqués de façon à regrouper les pairs du réseau principal en différents réseaux virtuels secondaires. Par exemple, considérons un système pair-à-pair au sein duquel on souhaite séparer les noeuds en six sous-réseaux distincts selon le continent sur lesquels les pairs sont géographiquement situés. Le résultat attendu est la création d'un réseau multicouche composé d'un réseau principal et de six réseaux secondaires, hébergeant respectivement les pairs provenant d'Afrique, d'Amérique, d'Antarctique, d'Asie, d'Europe et d'Océanie. Pour ce faire, durant l'exécution des protocoles de construction de réseaux virtuels superposés, un pair ajoute l'identifiant d'un autre pair dans sa seconde vue, si et seulement si ce second pair appartient au même continent que lui.

Les protocoles Vicinity et T-Man diffèrent, d'une part par le protocole d'échantillonnage de pairs qu'ils utilisent, et d'autre part par la stratégie qu'ils appliquent lors de la mise à jour des vues secondaires. Nous ne souhaitons par rentrer dans les détails de ces deux implantations dans ce document, cependant la comparaison de ces deux protocoles par Voulgaris [152] laisse apparaître que T-Man permet de construire les réseaux secondaires plus rapidement, mais consomme également plus de bande passante par noeud que le protocole Vicinity. Ces deux protocoles proposent un service de construction de réseau virtuel superposé qui est résistant à la dynamique des environnements pair-à-pair et permet de construire un ensemble de sous réseaux secondaires en un nombre d'étapes logarithmiquement proportionnel à la taille du système. A noter qu'une étape de ces protocoles correspond à un échange de vues, conformément au modèle d'exécution périodique des protocoles épidémiques présentés au sein la

Section 3.1.3.

Dans les études de Vicinity [152] et de T-man [88], les auteurs présentent des filtres, utilisés pour l'approvisionnement de la vue secondaire, qui sont plus complexes que celui présenté dans cette section. Ces filtres ont pour but de construire des topologies de réseau secondaire structurées telles qu'un anneau ou qu'une table de hachage distribuée. Ces structures n'étant pas utilisées au sein des travaux présentés dans cette thèse et par souci de concision, ces filtres ne sont pas présentés dans ce document. Cependant, il est important de noter que les protocoles de construction de réseau superposé ne gèrent pas l'attribution des valeurs qui sont utilisées pour filtrer les identifiants des pairs à mettre ou non dans une vue secondaire. En d'autres termes, ces protocoles sont capables de créer une topologie de réseau en couche en suivant un partitionnement prédéfini, mais ils ne présentent aucun service capable de calculer ce partitionnement. En effet, les seuls protocoles épidémiques capables de partitionner les pairs d'un système en plusieurs réseaux superposés sont les protocoles de morcellement de réseau virtuel.

3.3.2 Protocoles de morcellement de réseau

Les protocoles de morcellement de réseau virtuel, plus connus sous leur appellation anglo-saxonne « slicing protocols », sont utilisés pour trier et diviser les membres d'un réseau virtuel non structuré en sous-groupes. La motivation du découpage qu'ils réalisent est de définir plusieurs espaces de déploiement, afin de faire cohabiter plusieurs applications ou services indépendants au sein d'un même système pair-à-pair. On distingue deux types de protocole de morcellement : les protocoles de morcellement ordonné et les protocoles de morcellement absolu. Ces deux types de protocoles se différencient par la spécification guidant la division du réseau.

3.3.2.1 Protocole de morcellement ordonné

Les protocoles de morcellement ordonné utilisent un ordre, qui est établi sur les attributs des pairs du réseau, afin de morceler le système pair-à-pair sur lequel ils sont exécutés [85, 62, 70]. La spécification de découpage du réseau qu'ils proposent contient deux paramètres : *i)* l'attribut sur lequel l'ordre doit être basé et *ii)* le nombre de morceaux à produire lors de la division du réseau. Le fonctionnement de ces protocoles peut être décrit comme suit, en suivant le schéma d'exécution d'un protocole épidémique générique (Cf. Figure 3.4) :

- **Sélection d'un partenaire :** en suivant le modèle d'exécution des protocoles épidémiques et dans le cadre d'un échange d'information périodique, chaque noeud choisit un partenaire au hasard parmi les membres du système. Ce choix peut typiquement être fait en utilisant un protocole d'échantillonnage de pairs exécuté parallèlement au protocole épidémique de morcellement ordonné, sur le même réseau virtuel.
- **Echange de données :** les deux partenaires échangent la valeur de leurs attributs respectifs.
- **Traitement des données :** la valeur échangée est utilisée pour calculer une estimation du morceau auquel chacun des deux partenaires doit appartenir.

De façon simplifiée, un noeud possédant le $k$ème plus petit attribut au sein d'un réseau de taille n, doit évaluer sont propre index normalisé k/n. Il est ensuite facile pour ce noeud de déduire l'index du morceau auquel il appartient, en comparant son index normalisé au nombre de sous-groupes décrit dans la spécification de morcellement. La première méthode d'estimation [85]

consiste à attribuer un index aléatoire à chaque noeud lors de l'initialisation du protocole, puis d'ajuster cet index en fonction des comparaisons réalisées avec les valeurs collectées lors des échanges épidémiques. La seconde méthode, appelée estimation du rang [62, 70], consiste à évaluer l'index normalisé k/n des pairs de façon statistique, en comparant cette valeur à celles accumulées lors des échanges épidémiques. L'avantage de cette dernière méthode est que la distribution des noeuds parmi les différents morceaux ne dépend pas de l'uniformité des valeurs distribuées aux pairs, lors de l'initialisation du protocole.

Un exemple de morcellement ordonné est illustré sur la Figure 3.15, il correspond à une spécification de division, dans laquelle le nombre de morceaux à produire est de 4. Les valeurs entières données sur le schéma correspondent aux attributs selon lesquels les pairs sont ordonnés.

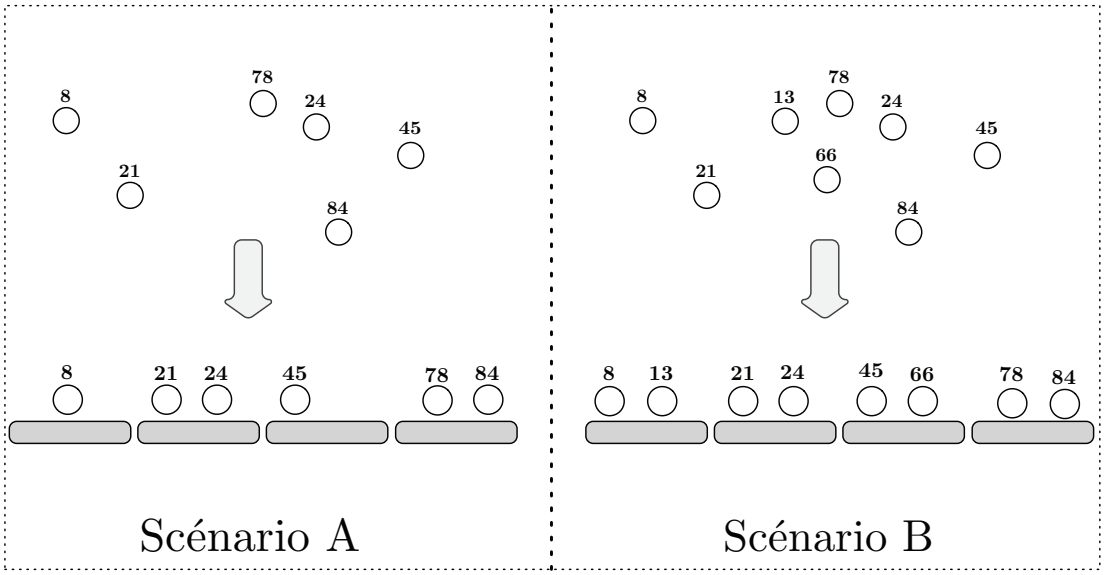


FIGURE 3.15 – Morcellement ordonné d'un réseau pair-à-pair.

Dans cet exemple, le scénario A correspond à un système composé de six pairs, si bien que le protocole de morcellement répartit ces pairs en 2 sous-groupes de 2 pairs et deux sous-groupes d'un seul pair. Dans le scénario B, le réseau est composé de 8 pairs, et le protocole répartit ces pairs en 4 sous-groupes de 2 pairs. La spécification de morcellement ordonné est respectée dans les deux cas, cependant on remarque qu'un protocole de morcellement ordonné ne présente aucune garantie concernant la taille des morceaux. Finalement, notons que les protocoles de morcellement ordonné sont capables, de par leur nature épidémique, de trier les noeuds et de les ordonner en un nombre donné de morceaux, sur des réseaux potentiellement grands (plus d'un million de membres d'après [62]), et ce, malgré la dynamique d'un environnement pair-à-pair. Cependant il existe d'autres protocoles de morcellement proposant d'autres spécifications pour diviser un réseau pair-à-pair non structuré, comme par exemple : les protocoles de morcellement absolu.

3.3.2.2 Protocole de morcellement absolu

Afin de proposer une spécification de morcellement de réseau pair-à-pair plus souple que celle fournie par les protocoles de morcellement ordonné, Montresor et coll. a proposé un

protocole de morcellement absolu [124]. Ce protocole propose de créer un sous-réseau au sein d'un réseau virtuel non-structuré en suivant une spécification de la forme $\mathcal{S}(c, \mathcal{T})$, dans laquelle c est un ensemble de contraintes que les pairs doivent respecter pour faire partie du sous-réseau et où $\mathcal{T}$ est la taille du morceau demandé. Un exemple de morcellement absolu est illustré sur la Figure 3.16. Dans cet exemple, on demande la création d'un sous réseau comportant 2 pairs, dont les attributs ont une valeur supérieure à ou égale à 24. Pour ce faire, les pairs

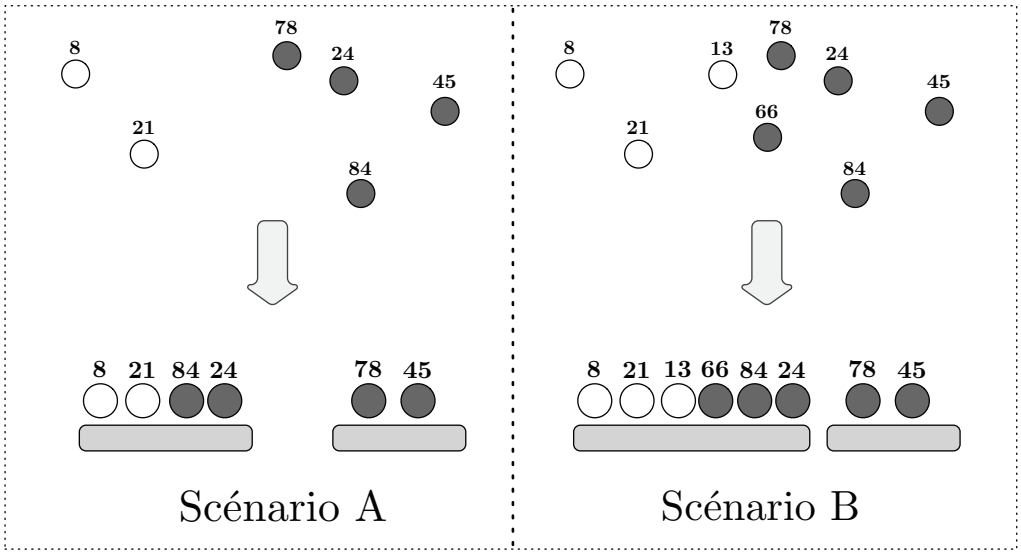


FIGURE 3.16 – Morcellement absolu d'un réseau pair-à-pair.

sont séparés en deux catégories : les pairs satisfaisant la contrainte de morcellement c (en gris sur la figure) et les pairs non désirés. La taille du sous-réseau créé à l'aide du protocole de morcellement absolu est toujours conforme à la spécification, quelque soit la taille du réseau principal. Le morcellement ainsi réalisé correspond alors à l'extraction d'un sous-ensemble des noeuds du réseau, dont la taille est fixe et sur lequel le filtre défini par la contrainte c a été appliqué.

Le protocole proposé par Montresor et coll. utilise une combinaison de protocoles épidémiques pour implanter le service de morcellement absolu [124]. Plus précisément, ce protocole utilise un service d'échantillonnage de pairs, un service de construction de réseau virtuel superposé, un service de diffusion épidémique et un service de comptage de pairs, le tout sur une infrastructure pair-à-pair composée de trois couches, comportant chacune un seul réseau virtuel. Cette architecture est décrite sur la Figure 3.17. Le principe de construction et de maintenance du morceau est le suivant : dans un premier temps, la spécification $\mathcal{S}(c, \mathcal{T})$ du morceau à construire est diffusée à l'ensemble des membres du système grâce au protocole de diffusion épidémique Lpbcast [60]. Ensuite, tous les noeuds satisfaisant la condition c sont regroupés au sein d'un réseau virtuel secondaire grâce à un protocole conçu par les auteurs qui est proche de T-Man [83, 88]. L'ensemble des pairs appartenant à ce réseau est appelé groupe potentiel. Un troisième réseau virtuel est également construit pour héberger le morceau cible, lequel est initialement vide. Une instance du protocole de comptage de pairs de Jelasity et coll. [122] est ensuite exécutée sur le réseau du groupe potentiel et sur le réseau du morceau cible, afin d'estimer les tailles réelles T_P du groupe potentiel et T_S du morceau cible. Finalement, le mouvement des pairs entre le réseau potentiel et le réseau du morceau est conditionné par les deux règles suivantes :

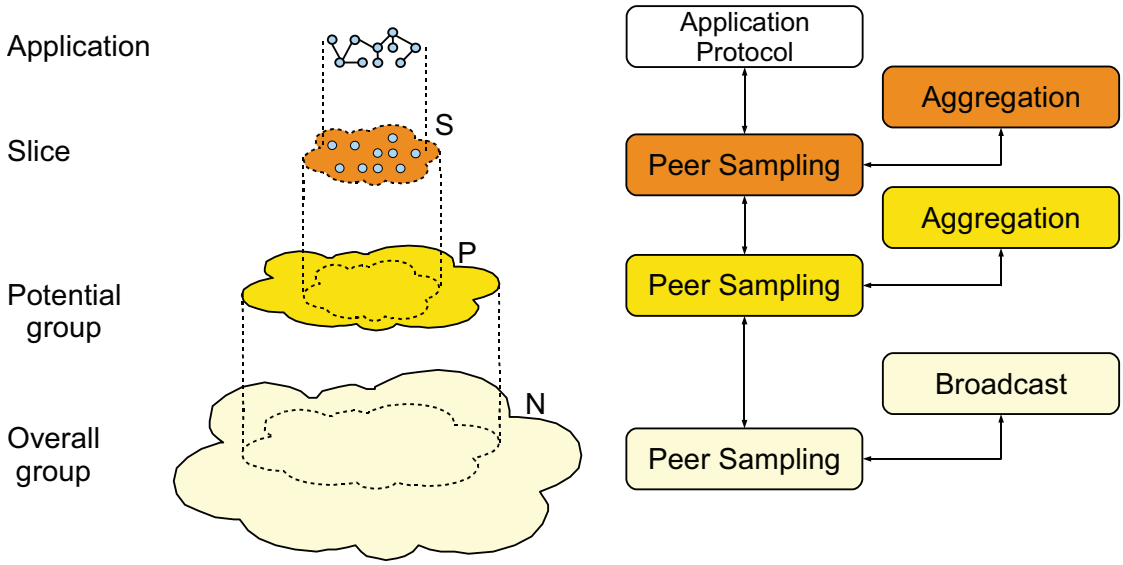


FIGURE 3.17 – Architecture du morcellement absolu. D'après l'étude de Montresor et coll. [124]

- Si la taille T_S du morceau cible est inférieure à la taille requise $\mathcal{T}$, alors chaque noeud appartenant au réseau du groupe potentiel décide de joindre le morceau avec une probabilité P_{join} telle que :

$$P_{join} = \frac{\mathcal{T} - T_S}{T_P - T_S}$$

- Si la taille T_S du morceau est supérieure à la taille requise $\mathcal{T}$, alors chaque noeud appartenant au morceau décide de quitter le réseau virtuel qui l'héberge avec une probabilité P_{leave} telle que :

$$P_{leave} = \frac{T_S - \mathcal{T}}{T_S}$$

Finalement, l'étude de Montresor et coll. montre que ces deux conditions permettent de construire un morceau de grande taille (jusqu'à 5000 noeuds d'après les expériences réalisées) en quelques minutes, et ce, dans les conditions réelles d'un système pair-à-pair, c'est-à-dire malgré la dynamique de l'environnement hôte [124].

3.3.3 Morcellement de réseau et allocation de pairs

La plateforme Salute présentée dans le chapitre précédent est un système pair-à-pair qui collecte des ressources bénévoles pour les redistribuer à des applications clientes. Les allocations réalisées par Salute prennent la forme de nuages pair-à-pair, lesquels correspondent à des ensembles de noeuds alloués à la demande pour une période prédéfinie, et dont la taille est maintenue stable tout au long de l'allocation. Une des particularités de Salute par rapport aux services d'allocation de noeuds existants est sa nature pair-à-pair, c'est-à-dire qu'elle utilise les ressources bénévoles pour maintenir son infrastructure et pour exécuter le service d'allocation de noeuds. Cependant, Salute n'est pas le premier système pair-à-pair à proposer de diviser son environnement hôte en sous-groupes, afin d'y déployer plusieurs applications distribuées, qui sont exécutées indépendamment les unes des autres. Ce problème de division,

mieux connu sous le nom de morcellement de réseau pair-à-pair, a fait l'objet des études présentées précédemment. Cette section explique en quoi les solutions proposées dans ces études sont insuffisantes pour permettre l'implantation d'un service d'allocation de noeuds totalement décentralisé qui repose sur le modèle d'allocation de l'infrastructure en tant que service, c'est-à-dire dont le but est d'allouer des noeuds à la demande.

La problématique du partage de ressources entre plusieurs applications, au sein d'un même système pair-à-pair, a été initialement soulevée par Babaoglu et coll. [33]. Dans cette étude préliminaire, dont les motivations sont données au sein du chapitre 2, les auteurs proposent d'utiliser un protocole de morcellement ordonné afin de définir les différents sous-groupes de noeuds alloués à chaque application déployée au sein d'un système pair-à-pair. Pour ce faire, Babaoglu et coll. propose d'utiliser une unique description de déploiement d'applications, gérée de façon centralisée et périodiquement diffusée à l'ensemble des pairs du système. Cette description contient la spécification du morcellement ordonné et associe chacun des morceaux à une application donnée. Cependant, un des inconvénients principaux de cette solution est que l'utilisation d'un protocole de morcellement ordonné ne permet pas de spécifier la taille des groupes de noeuds à allouer. Cette limitation empêche les protocoles de morcellement ordonné de satisfaire les conditions d'un service d'allocation à la demande, au sein duquel la quantité de ressources à allouer doit pouvoir être spécifiée. En conséquence, la solution proposée par Babaoglu et coll. [33] n'est pas adaptée pour implanter un service d'allocation de noeuds à la demande.

Cette dernière remarque a inspiré Montresor et coll. [124] pour la conception d'un protocole de morcellement absolu, capable de construire un sous-groupe de noeuds de taille donnée, au sein d'un environnement pair-à-pair. Cette solution propose un morcellement de réseau pair-à-pair hautement configurable, qui permet de définir la taille de l'ensemble de noeuds à allouer et de filtrer ces noeuds en fonction de contraintes établies sur leurs attributs (capacité de calcul ou de stockage, situation géographique,...etc). Ce protocole de morcellement satisfait le modèle d'allocation de l'infrastructure en tant que service, car il permet d'allouer un ensemble de noeuds de taille donnée et à la demande. Néanmoins, la conception de ce protocole ne prend pas en compte la possibilité de construire plusieurs morceaux au sein d'un même système pair-à-pair. Plus particulièrement, nous avons identifié dans cette thèse deux problèmes, liés à la volonté d'allouer plusieurs morceaux de taille fixe au sein d'un système pair-à-pair, qui ne sont pas pris en compte dans les travaux de Montresor et coll. :

- **La gestion de la concurrence des requêtes d'allocation :** l'étude de Montresor et coll. [124] ne précise pas quel est le comportement du système lorsque plusieurs requêtes de morcellement sont soumises simultanément au système. En particulier, ce protocole de morcellement ne propose pas de solution décentralisée permettant de départager deux requêtes lorsqu'elles sont en concurrence pour le recrutement d'un même noeud. Comme il est décrit plus loin dans ce document, un tel cas peut poser des problèmes non triviaux, comme des scénarios de famine générale, lorsque la concurrence des requêtes est gérée de façon totalement décentralisée (Cf. Chapitre 5).

- **Le contrôle d'admission des requêtes d'allocation :** dans l'étude de Montresor et coll. [124], la disponibilité des pairs, qui sont des ressources non gérées, n'est pas prise en compte lors du morcellement du réseau. En d'autres termes, lorsque le système construit un morceau de taille T, il n'existe aucun mécanisme en charge d'évaluer si le système comportera suffisamment de ressources, c'est-à-dire au moins T noeuds, pour satisfaire la spécification du morceau à construire au cours du temps. Si on considère des environnements de déploiement dynamiques tels que des systèmes pair-à-pair, dont la

taille est amenée à varier dans le temps avec une forte probabilité, alors il est important de contrôler l'admission des requêtes d'allocation. En effet, les travaux réalisés dans cette thèse montrent qu'un phénomène appelé « surrallocation » peut corrompre le morcellement d'un système pair-à-pair, lorsqu'aucun contrôle d'admission n'est appliqué sur les requêtes de morcellement (Cf. Chapitre 7). Par conséquent, nous considérons dans cette thèse que pour qu'un service d'allocation de noeuds pair-à-pair soit fiable, il faut que les requêtes d'allocation acceptées obtiennent la garantie qu'elles puissent être maintenues au cours du temps.

Pour résumer, les protocoles de morcellement existants ne sont pas capables de fournir un découpage des ressources suffisamment configurable ou suffisamment fiable pour fournir un service d'allocation de nuages pair-à-pair similaire à celui décrit dans le Chapitre 2. Un exemple illustrant les limitations de ces derniers protocoles est donné au sein de la Figure 3.18. Dans cet exemple, on veut construire deux sous-groupes comportant respectivement 2 et 3

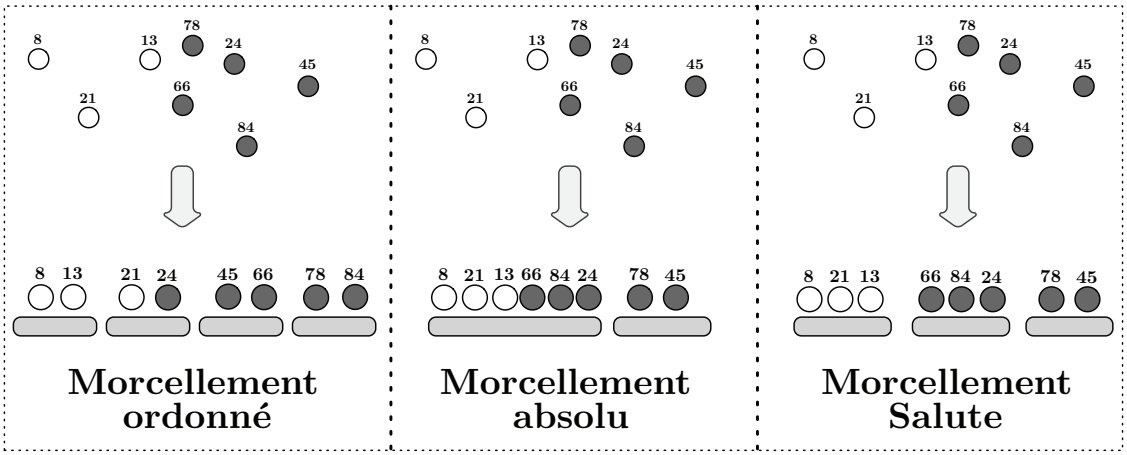


FIGURE 3.18 – Exemple de morcellement différents.

noeuds, dont l'attribut associé aux pairs (qui est arbitraire dans cet exemple) est supérieur à 21. En utilisant le protocole de morcellement ordonné, cette configuration est difficilement réalisable, car le seul paramètre de configuration pour le découpage est le nombre de morceaux à produire dans le système. Le morcellement se rapprochant le plus du morcellement souhaité est celui décrit sur la sous-figure de droite, il comporte 4 morceaux de 2 noeuds chacun. En utilisant le protocole de morcellement absolu, on ne peut construire qu'un seul morceau de façon fiable, comme illustré sur la sous-figure du centre. Il est donc nécessaire d'utiliser un nouveau protocole de morcellement, afin de produire le découpage souhaité, illustré sur la sous-figure de droite. Cet exemple simple démontre les limitations des protocoles de morcellement existants, et donc l'utilité de concevoir un nouveau protocole capable de créer des sous-groupes configurables au sein d'un système pair-à-pair, afin d'implanter un service d'allocation de noeuds à la demande.

Conclusion Les systèmes pair-à-pair sont des systèmes distribués autonomes capables de gérer un très grand nombre de ressources informatiques hétérogènes. Ces systèmes sont généralement faciles à déployer et à maintenir et tolèrent la dynamique de larges environnements non gérés, qui sont composés par des machines dont la disponibilité n'est pas contrôlée par le

système auquel elles participent. Les systèmes pair-à-pair parviennent à fournir ces caractéristiques en utilisant soit des tables de hachage distribuées, soit des protocoles épidémiques, qui sont exécutés en collaboration par tous les membres du système.

La plupart des systèmes pair-à-pair basés sur l'utilisation de protocoles épidémiques sont réputés être plus robustes à la dynamique des environnements pair-à-pair que les tables de hachage distribuées. Ces protocoles épidémiques sont capables de fournir toute une gamme de services génériques, tels que la diffusion fiable d'information, l'estimation de la taille d'un réseau virtuel non structuré, ou encore la synchronisation des noeuds collaborant à l'exécution du système. Une tendance récente consiste à utiliser des protocoles épidémiques de morcellement pour rassembler les membres d'un système pair-à-pair dit multicouche en sous réseau virtuels indépendants, de façon à pouvoir assigner différentes tâches aux membres du système. Néanmoins, les protocoles de morcellement de réseau existants offrent des services insuffisants pour l'implantation d'un service d'allocation de noeuds à la demande. En particulier, ces protocoles ne gèrent pas la concurrence et ne contrôlent pas l'admission des requêtes d'allocation ou de morcellement.

Chapitre 4

Salute : un système d'allocation de ressources pair-à-pair

Sommaire

Salute [1] est une plateforme distribuée, totalement décentralisée, capable d'allouer des collections de noeuds de tailles multiples en étant déployée sur un environnement large échelle, lequel est composé de ressources informatiques potentiellement hétérogènes et non gérées. Ce type d'environnement correspond, entre autres, à l'Intercloud précédemment défini dans le Chapitre 1 de ce document. La plateforme Salute héberge deux types de collections de ressources informatiques : une entrepôt, dans lequel des ressources informatiques disponibles sont collectées, et des nuages pair-à-pair au sein desquels ces ressources sont redistribuées à des applications clientes. Le contenu des nuages pair-à-pair est configurable et possède un nombre fixe de noeuds. Une durée de déploiement est associée à chaque allocation de nuage pair-à-pair. Ce modèle d'allocation de ressource est inspiré de l'infrastructure en temps que service, dans laquelle les ressources sont allouées à la demande.

Ce chapitre présente la plateforme Salute hébergeant le service d'allocation de nuages pair-à-pair situé au coeur de la contribution de cette thèse. Le concept de nuage pair-à-pair est présenté au sein de la Section 4.1. La Section 4.2 présente la problématique principale de Salute, à savoir les problèmes soulevés par l'allocation de ressources informatiques, dont la disponibilité n'est pas contrôlée, au sein de différentes collections de noeuds. Finalement, le principe de fonctionnement de la plateforme Salute est décrit au sein de la Section 4.3.

1. De l'anglais : « Scalable Allocation in Large Unmanaged disTributed Environments ».

4.1 Nuages pair-à-pair

Un nuage pair-à-pair est un ensemble de ressources informatiques, également appelées « noeuds », qui sont rassemblées au sein d'un même réseau virtuel pair-à-pair. C'est une nouvelle abstraction utilisée pour l'allocation de ressources au sein de l'informatique dans le nuage. La collection de noeuds qu'il contient est dédiée au déploiement d'une unique application distribuée. Plus précisément, la spécification d'un nuage pair-à-pair contient trois paramètres : *i)* la taille du nuage, c.-à-d. le nombre de noeuds qu'il contient, *ii)* la durée de déploiement du nuage et *iii)* une liste de filtres portant sur les attributs des noeuds qui le composent.

Au sein des travaux présentés dans cette thèse, les nuages pair-à-pair sont construits sur des réseaux virtuels non structurés, lesquelles sont maintenus par des protocoles épidémiques d'échantillonnage de pairs (Cf. Chapitre 3). Par conséquent, ces nuages sont des structures autonomes, qui sont maintenues par la collaboration des ressources qui les composent. Ils sont capables de tolérer un grand dynamisme de la part de l'environnement hôte, même dans le cas ou celui-ci est composé de ressources informatiques dont la disponibilité n'est pas contrôlable.

Grâce à l'utilisation de protocoles pair-à-pair, la taille des nuages peut varier d'une dizaine à plusieurs millions de noeuds. La durée de déploiement est exprimée dans une métrique commune à toutes les noeuds du système, par exemple l'heure. Finalement, l'ensemble de filtres contenus dans la spécification des nuages permet de définir des contraintes sur les attributs des noeuds alloués. Par exemple, un utilisateur peut exiger que tous les noeuds compris dans son nuage pair-à-pair disposent d'une puissance de calcul supérieure à 2GHz et d'un espace de stockage supérieur ou égale à 10Go. L'utilisation de ces filtres est davantage détaillée au sein du Chapitre 5.

Au sein de la plateforme Salute, les nuages pair-à-pair cohabitent avec un entrepôt de noeuds, qui rassemble toutes les ressources informatiques disponibles pour le déploiement des applications clientes. Aussi l'architecture de Salute peut être divisée en deux couches de réseaux virtuels, comme illustré sur la Figure 4.1. La couche inférieure de cette architecture correspond au réseau principal, qui regroupe tous les membres de la plateforme pair-à-pair Salute. La couche supérieure, elle, contient un nombre potentiellement important de nuages pair-à-pair et un unique entrepôt de noeuds.

Ainsi, le déploiement de la plateforme Salute consiste à créer le réseau virtuel principal à l'aide d'un noeud de démarrage unique et arbitraire, puis de collecter les ressources à allouer au sein de l'entrepôt. Ce déploiement présente un faible coût d'administration, car l'ajout d'une nouvelle ressource au sein de la plateforme Salute ne nécessite aucune configuration manuelle. La ressource ajoutée est introduite au sein de l'infrastructure pair-à-pair de Salute et participe directement à sa maintenance. La plateforme Salute peut être déployée au sein des grands centres de traitement de données, sur une fédération de nuages, ou encore sur n'importe quel environnement distribué composé de ressources potentiellement hétérogènes et de provenances diverses. Le seul prérequis pour joindre la plateforme pair-à-pair est de connaître l'adresse IP d'au moins un de ses membres (en supposant que la plateforme soit déployée sur un réseau utilisant le protocole IP). Lorsqu'un utilisateur souhaite soumettre une requête de réservation de ressources au service d'allocation de Salute, il lui suffit de transmettre la spécification du nuage pair-à-pair demandé à un des membres du système. Si cette requête est acceptée par le service d'allocation, alors un nuage pair-à-pair correspondant à la spécification requise sera créé au sein de la plateforme Salute, puis un ensemble d'adresses de « noeuds de contacts » seront communiquées à l'utilisateur. Ces noeuds de contact sont des noeuds membres du réseau virtuel sur lequel le nuage pair-à-pair est hébergé, ils sont

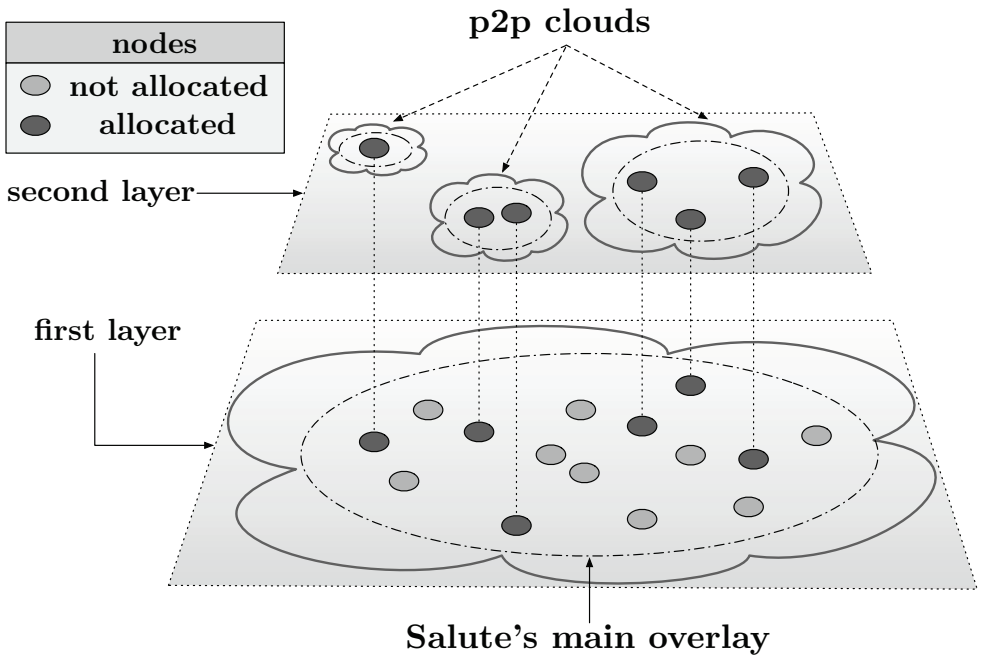


FIGURE 4.1 – Principe de l'architecture de la plateforme Salute.

utilisés par les clients afin de déployer leurs applications sur le nuage pair-à-pair qui leur est alloué. Aucune méthode générique de déploiement d'applications sur un nuage pair-à-pair n'est proposée dans cette thèse, car nous estimons que les protocoles de déploiement sont spécifiques à chaque type d'application distribuée.

Il est néanmoins important de noter que dans le cadre de travaux ayant pour but de promouvoir le déploiement de services pair-à-pair composites, plusieurs protocoles épidémiques capables de démarrer « rapidement » une application pair-à-pair ont été proposés [83, 123, 87, 88]. Ces protocoles permettent de construire, à partir d'un service d'échantillonnage de pairs, un réseau virtuel structuré ou non, en un temps logarithmiquement proportionnel à la taille du dit réseau. Par exemple, ces protocoles sont capables de déployer la table de hachage distribuée Chord [141], en utilisant un service d'échantillonnage de pairs. Ces protocoles permettent également de fusionner ou de séparer temporairement des réseaux virtuels secondaires appartenant à une architecture pair-à-pair multicouche (Cf. Chapitre3). Puisque les nuages pair-à-pair sont construits sur des réseaux pair-à-pair non structurés, qui sont maintenus par des protocoles d'échantillonnage de pairs, la plateforme Salute est compatible avec l'utilisation de ces protocoles de démarrage rapide d'applications pair-à-pair.

Dans la suite de ce document, on considérera qu'un noeud correspond à une machine physique, plutôt qu'à un espace d'exécution virtualisé, comme c'est souvent le cas au sein de l'informatique dans les nuages. Ce modèle a pour simple but de simplifier les hypothèses de disponibilité des noeuds, afin de pouvoir comparer nos travaux aux études existantes concernant les systèmes pair-à-pair.

4.2 Allocation de noeuds et flux de churn

Le churn, ou dynamique des environnements pair-à-pair, correspond au phénomène créé par les départs et arrivées continus des membres d'un système pair-à-pair. En effet, la disponibilité des noeuds composant un réseau pair-à-pair varie de façon aléatoire et n'est prévisible que dans certains cas particuliers, et ce, pour une période de très courte durée [117]. De fait, l'ensemble des membres d'un système pair-à-pair évolue de façon aléatoire, imprévisible et dynamique. Cependant, il existe de nombreux systèmes pair-à-pair qui parviennent à fournir des services fiables, malgré la dynamique et l'hétérogénéité des ressources qui les composent. Pour ce faire, ces systèmes utilisent des protocoles distribués qui sont spécifiques à un service donné, mais qui sont capables de gérer la dynamique des réseaux qui les hébergent, en étant conçus pour utiliser autant de ressources que possible indépendamment de leur disponibilité. Néanmoins, aucun de ces protocoles n'est capable de diviser une collection de noeuds, quelle que soit sa taille, de façon à déployer des services exécutés de façon indépendante sur ces ressources.

Contrairement aux systèmes pair-à-pair traditionnels, Salute est capable de gérer plusieurs collections de noeuds hébergées sur un même environnement pair-à-pair, dont les tailles sont stables et auxquelles est associée une durée de déploiement. Ces collections s'appellent des nuages pair-à-pair. Les motivations pour utiliser ce type de collection sont les suivantes : *i)* cela permet le déploiement de plusieurs services indépendants au sein d'un même ensemble non géré de ressources informatiques, et *ii)* cela permet au système d'ordonnancer l'allocation des noeuds qui le composent en fonction de leur disponibilité. C'est en utilisant ce dernier principe que Salute parvient à allouer des nuages pair-à-pair à la demande à des applications qui s'exécutent de façon indépendante, comme le ferait un nuage classique en allouant des noeuds provenant d'un centre de traitement de données. La contrepartie du système proposé dans cette thèse est l'apparition d'un nouveau problème : la maintenance des différentes allocations de noeuds en dépit de la dynamique de l'environnement hôte. En effet, au fur et à mesure que les pairs tombent en panne, quittent ou joignent Salute, les nuages pair-à-pair alloués doivent être continuellement surveillés et nourris avec de nouveaux noeuds, si besoin est, afin de maintenir leur spécification pendant toute la durée de leur déploiement.

L'utilisation de « flux de churn » est particulièrement bien appropriée à la description de ce problème, puisque les collections de noeuds existantes au sein de Salute sont en évolution constante, du fait de la dynamique créée par les pairs du système. Ces flux de churn peuvent prendre la forme de « flux d'arrivée », de « flux de départ », ou encore de « flux d'approvisionnement ». La Figure 4.2 schématise les différents flux de churn qui existent dans un scénario où trois nuages pair-à-pair sont alloués, puis maintenus par Salute, malgré le churn. Les pairs joignant le système créent un flux d'arrivée qui remplit l'entrepôt de noeuds. Ensuite, Salute utilise les ressources contenues dans cet entrepôt pour créer, puis maintenir, les nuages pair-à-pair (respectivement noté Pc_i, Pc_j et Pc_k sur la figure), en créant un flux d'approvisionnement par nuage. La fonction d'un flux d'approvisionnement est de transférer des noeuds depuis l'entrepôt jusqu'au nuage donné. Dans un premier temps, ce flux remplit le nuage en question jusqu'à ce que la taille spécifiée soit atteinte, puis il fournit des noeuds supplémentaires à ce nuage, de façon à compenser le potentiel flux de départ qui s'y exécute. Effectivement, alors que les noeuds tombent en panne ou quittent volontairement le système de façon continue, plusieurs flux de départ apparaissent : dans l'entrepôt et dans les nuages pair-à-pair déployés. Ces flux de départ on pour conséquence de diminuer la taille des nuages alloués de façon ininterrompue. Les flux d'approvisionnement doivent donc constamment transférer de nouveaux noeuds au sein des différents nuages pair-à-pair, de façon à maintenir leur spécification,

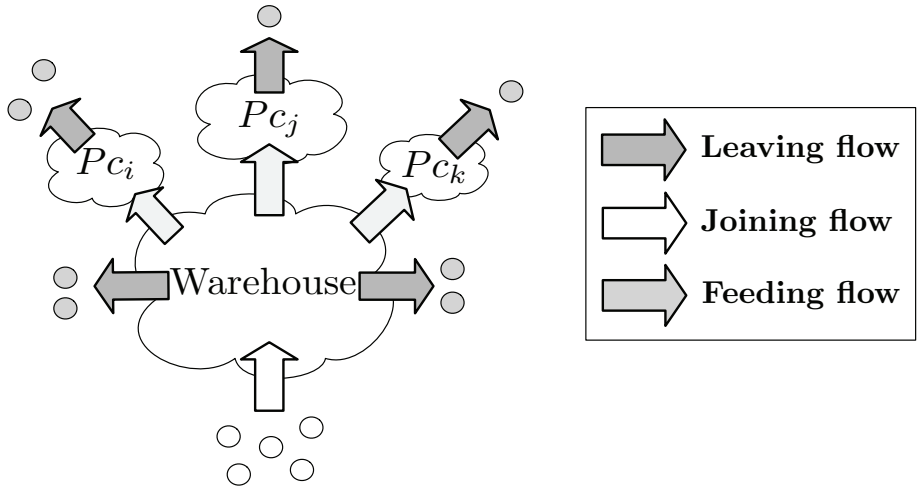


FIGURE 4.2 – Flux de churn au sein de la plateforme Salute.

malgré les flux de churn crées par la disponibilité des ressources de l'environnement hôte.

Dans ce contexte dynamique, le rôle de Salute est d'évaluer, puis de décider, si une requête d'allocation peut être satisfaite et maintenue au cours du temps de façon fiable, en fonction de l'état des différents flux de churn présents dans l'ensemble du système. Si l'entrepôt devient vide ou trop petit par rapport à l'ensemble des ressources allouées au sein des nuages pair-à-pair, alors les flux d'approvisionnement vont être interrompus et les spécifications des différents nuages déployés ne pourront plus être maintenues. Typiquement, ce problème apparaît lorsque tous les noeuds présents dans l'entrepôt sont alloués dans des nuages pair-à-pair et que le flux d'arrivée de l'entrepôt devient moins important que la somme des flux de départ présents dans le système. Un tel événement peut par exemple se produire si la taille globale du système diminue, c'est-à-dire que la tendance globale des pairs est de quitter le système plutôt que de le joindre. Or, d'après la caractéristique périodique de la disponibilité des ressources dans les systèmes pair-à-pair, un tel événement est probable au sein d'une large communauté de ressources informatiques non gérées [43, 58, 73, 117]. En particulier, les environnements pair-à-pair de référence choisis pour évaluer les travaux présentés dans cette thèse présentent une taille de système qui varie périodiquement (Cf. Chapitre 7). Par conséquent, lorsque Salute autorise l'allocation d'une collection de ressource, il doit exercer un contrôle d'admission visant à assurer que cette collection, ainsi que toutes celles précédemment allouées, pourront être maintenues au cours de leurs déploiements. En d'autres termes Salute soit s'assurer que l'environnement hôte comportera suffisamment de ressources dans le futur pour satisfaire l'ensemble des allocations de nuages pair-à-pair en cours.

Afin d'éviter que trop de noeuds ne soient alloués au sein des nuages pair-à-pair, Salute divise les ressources contenues au sein de l'entrepôt en deux sous catégories : le magasin et la réserve. Un statut d'allocation est également attribué à chaque pair membre de la plateforme Salute, en fonction de la collection de noeuds à laquelle il appartient. Les pairs qui sont regroupés au sein des nuages pair-à-pair possèdent le statut « alloué ». Un noeud possédant le statut « alloué » ne peut appartenir qu'à un seul nuage pair-à-pair et n'appartient pas à l'entrepôt. Les pairs appartenant à la réserve possèdent le statut « de réserve ». Ces pairs sont strictement dédiés à compenser les flux de départ des nuages pair-à-pair alloués et ne peuvent pas être considérés comme disponibles au moment d'allouer un nouveau nuage. Les

pairs regroupés au sein du magasin possèdent le statut « libre », ils peuvent être utilisés pour créer de nouveaux nuages ou pour compenser le départ des ressources au sein des nuages en cours de déploiement. L'organisation de l'architecture multicouche de Salute et la répartition des pairs en fonction de leurs statuts sont illustrées sur la Figure 4.3.

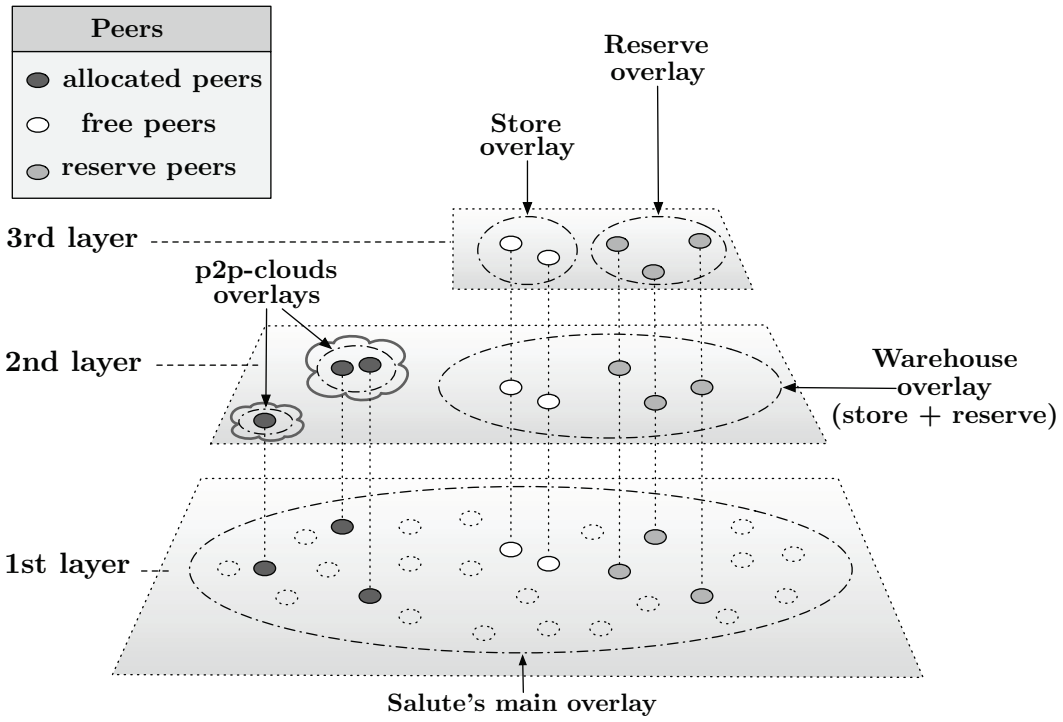


FIGURE 4.3 – Illustration complète de l'architecture multicouche de Salute.

4.3 Principe de fonctionnement de la plateforme Salute

Cette section présente une vue d'ensemble du fonctionnement de la plateforme Salute et du service d'allocation de noeuds qu'elle héberge. Une première partie présente comment différents agents, dont l'exécution est totalement décentralisée, collaborent à la réalisation du service d'allocation de nuages pair-à-pair à la demande. Une seconde partie présente le service de gestion de réseau virtuel qui est utilisé pour maintenir l'architecture multicouche de Salute. Finalement une troisième partie présente le protocole de gestion de l'entrepôt de noeuds.

4.3.1 Vue d'ensemble du service d'allocation de noeuds

Le service d'allocation de noeuds proposé par la plateforme Salute est un service pair-à-pair collaboratif exécuté par tous les membres du système. Afin de simplifier la description de ce service, on peut distinguer trois agents principaux correspondant chacun à une sous-partie de ce service : *i)* les contrôleurs d'admission, dont le rôle est de décider si une requête doit être satisfaite ou non, en fonction de l'évolution de l'environnement hôte, *ii)* Sarcasm, le gestionnaire construisant et maintenant les nuages pair-à-pair et *iii)* le protocole de gestion de l'entrepôt, qui collecte les ressources informatiques disponibles et leur distribue les statuts de pair de réserve ou de pair libre. Cette section présente le fonctionnement du service d'allocation

de noeuds à travers le schéma donné sur la Figure 4.4. Ce schéma retrace les différentes étapes du traitement d'une requête d'allocation de ressources.

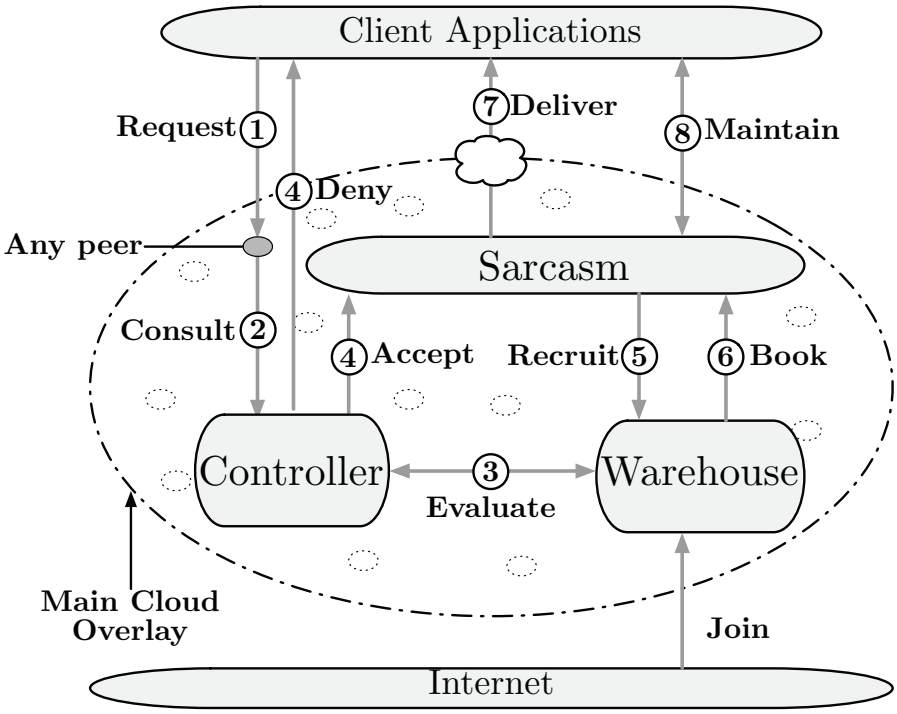


FIGURE 4.4 – Schémas de fonctionnement du service d'allocation de noeuds de Salute.

Une requête peut être émise par un utilisateur depuis n'importe quel pair membre du réseau principal de Salute (**1**). Lorsqu'un pair reçoit une requête d'allocation, il consulte les contrôleurs d'admission afin de décider s'il faut autoriser la propagation de cette requête dans le système ou non (**2**). Ensuite, en fonction de l'état de l'entrepôt (**3**) et des prédictions sur l'évolution de l'environnement hôte, les contrôleurs d'admission décident d'accepter ou de rejeter la requête d'allocation (**4**). Si la requête est rejetée, le processus de construction de nuage pair-à-pair est abandonné et l'utilisateur est notifié. Dans le cas contraire, le pair ayant reçu cette requête crée un flux d'approvisionnement qui va nourrir le nuage demandé (**5**). A noter que la création de flux d'approvisionnement est réalisée selon le protocole Sarcasm, qui est décrit au sein du Chapitre 5. Lorsque le nuage en question est complet, les pairs qui le composent changent leur statut de « libre » à « alloué » (**6**), de façon à marquer leur réservation pour ce nuage. L'utilisateur est alors alerté que le nuage pair-à-pair demandé est complet (**7**) et reçoit des points d'entrés sur le réseau virtuel contenant les ressources demandées, le nuage est livré. Finalement, durant l'intégralité du déploiement d'un nuage, Sarcasm surveille les noeuds qui le composent et recrute de nouveaux pairs, si cela est nécessaire au maintien de la spécification du nuage en question (**8**). Ce dernier recrutement est réalisé en répétant les étapes **5** et **6** décrites ci-dessus, à l'exception que le nuage peut maintenant recruter des pairs dits « de réserve ». Les protocoles et algorithmes utilisés par Sarcasm pour construire et maintenir les nuages pair-à-pair sont décrits dans le Chapitre 5. Les heuristiques utilisées par les contrôleurs d'admission pour prédire l'évolution du système et décider si une requête doit être autorisée ou non sont détaillées dans le Chapitre 6.

4.3.2 Service de gestion de réseau virtuel

La plateforme Salute utilise un service de gestion de réseau virtuel pair-à-pair pour maintenir son architecture multicouche. Ce service de gestion de réseau virtuel peut être divisé en quatre sous-services : le service d'échantillonnage de pairs, le service de comptage de pairs, le service de construction de réseau superposé et le service de synchronisation des noeuds. Une instance du service d'échantillonnage de pairs et une instance du service de comptage de pairs sont exécutées pour maintenir les réseaux hébergeant chacune des collections de noeuds suivantes : le réseau principal de Salute, l'entrepôt, le magasin, la réserve et chaque nuage pair-à-pair. Le service de synchronisation des noeuds, lui, n'est exécuté que sur le réseau principal de Salute, ce qui suffit à synchroniser tous les noeuds du système. Finalement, le service de construction de réseau superposé est exécuté sur le réseau principal de Salute, afin de construire le réseau virtuel hébergeant l'entrepôt. Il est également exécuté sur le réseau virtuel de l'entrepôt pour créer les réseaux virtuels hébergeant le magasin et la réserve de noeuds. Ces quatre services sont décrits dans les paragraphes suivants :

- ***Echantillonnage de pairs et maintien de réseau virtuel*** *:* Comme il est expliqué plus tôt dans ce chapitre, les différents réseaux virtuels hébergés au sein de l'architecture de Salute sont maintenus grâce à un protocole d'échantillonnage de pairs. Le protocole choisi pour implanter le service d'échantillonnage de pairs de Salute est Cyclon [153]. Ce protocole fournit un service d'échantillonnage fiable malgré la dynamique de l'environnement hôte et, de plus, est nativement compatible avec le protocole Vicinity [154, 152, 155], qui est également utilisé au sein de la plateforme Salute pour construire des réseaux virtuels superposés. Cyclon peut également être remplacé, si besoin est, par le protocole Nylon [95], qui correspond à une version de Cyclon capable de fournir un service d'échantillonnage de pairs, dans des scénarios pour lesquels une large proportion de noeuds utilise des techniques de translation d'adresse.

- ***Comptage des pairs d'un réseau virtuel*** *:* Ce service permet d'estimer la taille d'un réseau virtuel. L'évaluation de Salute présentée au sein du Chapitre 7, a été réalisée en utilisant le protocole *Average* [86], dont le fonctionnement est détaillé au sein du Chapitre 3. Cependant, le protocole *T-Size* [121] peut également être utilisé à la place du protocole *Average*. Ces deux protocoles présentent des interfaces de service similaires : ils reposent tous deux sur l'utilisation d'un service d'échantillonnage de pairs et ils calculent la taille du réseau virtuel hôte de façon totalement décentralisée. En particulier, lors de l'utilisation d'un de ces deux protocoles, tous les membres du réseau virtuel connaissent l'estimation de la taille du réseau. Finalement, ces deux protocoles se différencient par leurs performances. Typiquement le protocole *Average* est plus lent, moins résistant à la dynamique des pairs, et consomme moins de bande passante que le protocole *T-Size* [121]. Le choix initial du protocole *Average* est dû à sa simplicité d'implantation. L'impact de ce choix sur les performances du service d'allocation de noeuds de Salute est discuté au sein du Chapitre 7.

- ***Synchronisation des noeuds*** *:* Tous les pairs collaborant à l'exécution de la plateforme Salute sont supposés être synchronisés, au moins à l'échelle de la seconde. Cette synchronisation peut être réalisée à l'aide du protocole d'ajustement d'horloge en réseau NTP [119]. Ce protocole très largement répandu utilise une infrastructure distribuée, composée de serveurs de temps. De plus, la majorité des ordinateurs aujourd'hui connectés à Internet utilisent le protocole NTP pour synchroniser leur horloge sur une horloge

de référence correspondant à un fuseau horaire donné. Dans le cadre de Salute, le protocole NTP peut fournir une synchronisation suffisante, si tous les noeuds sont synchronisés sur un fuseau horaire commun. Dans des scénarios de déploiement pour lesquels le protocole NTP ne pourrait pas être utilisé, par exemple pour déployer Salute sur un ensemble de machines déconnecté d'Internet, le protocole GTP [80] peut être utilisé pour assurer la synchronisation des noeuds. A noter que ce protocole nécessite l'utilisation d'une horloge fiable de référence, tel qu'un serveur NTP public ou privé [119], afin d'assurer la synchronisation des noeuds autour d'une horloge de référence [80]. Ce protocole est conçu pour être déployé sur un réseau virtuel maintenu par un protocole d'échantillonnage de pairs et permet de synchroniser les membres de ce réseau avec une précision d'environ 10ms, en dépit d'une forte dynamique de l'environnement hôte. Afin de synchroniser les pairs collaborant à la maintenance de la plateforme Salute, ce protocole doit être déployé sur le réseau virtuel principal de Salute, afin de toucher l'intégralité des membres de la plateforme. Ainsi, l'utilisation des protocoles NTP ou GTP est suffisante pour assurer que les noeuds de la plateforme Salute sont synchronisés à l'échelle de la seconde, c'est-à-dire qu'ils sont capables d'utiliser le même fuseau horaire et de déclencher une action, datée à la seconde prés, avec un décalage maximum d'une seconde entre deux pairs.

- ***Construction de réseau superposé :*** Le protocole de construction de réseau superposé utilisé dans Salute est Vicinity [154, 152, 155]. Ce protocole est déployé sur le réseau principal de Salute afin de construire le réseau virtuel hébergeant l'entrepôt de noeuds. Il est également déployé sur le réseau virtuel de l'entrepôt afin de définir les réseaux virtuels associés au magasin et à la réserve de pairs. Vicinity a été conçu pour être combiné avec l'utilisation du protocole d'échantillonnage de pairs Cyclon [153]. Il est notamment capable de maintenir une topologie de sous-réseaux virtuels donnée, sur des systèmes pair-à-pair déployés à large échelle, et ce, malgré la dynamique de l'environnement hôte.

Un schéma récapitulatif de l'ensemble des services pair-à-pair de gestion de réseau virtuel, qui sont déployés au sein de la plateforme Salute, est illustré sur la Figure 4.5.

4.3.3 Protocole de maintenance de l'entrepôt

La gestion de l'entrepôt de noeuds suit un protocole simple. On peut distinguer deux cas dans lesquels un noeud rejoint cet entrepôt : dans le premier cas, ce noeud arrive dans le système pour la première fois ou après un temps significatif de déconnexion. Dans le second cas, ce noeud revient dans l'entrepôt après avoir été relâché, à la fin du déploiement d'un nuage pair-à-pair. Dans ces deux cas, ce noeud doit, soit recevoir le statut de pair libre et rejoindre le magasin, soit recevoir le statut de pair de réserve et rejoindre la réserve de noeuds. La règle guidant l'attribution des statuts aux pairs rejoignant l'entrepôt de noeuds est la suivante : si le pourcentage de pairs de réserve dans le système est en dessous d'un seuil $\mathcal{S}$ prédéfini, alors le noeud en question prend le statut de pair de réserve et rejoint la réserve de noeuds. Dans le cas contraire, ce noeud prend le statut de pair libre et rejoint le magasin. A noter que les noeuds sont capables d'évaluer la taille de la réserve et la taille du système, grâce au service de gestion de réseau virtuel présenté plus tôt dans cette section.

Ce simple protocole permet d'assurer que le nombre de pairs de réserve garde une valeur stable proche du pourcentage $\mathcal{S}$, indépendamment de la taille globale du système. Ce pourcentage fixe de pairs de réserve a pour but de permettre le maintien des nuages pair-à-pair

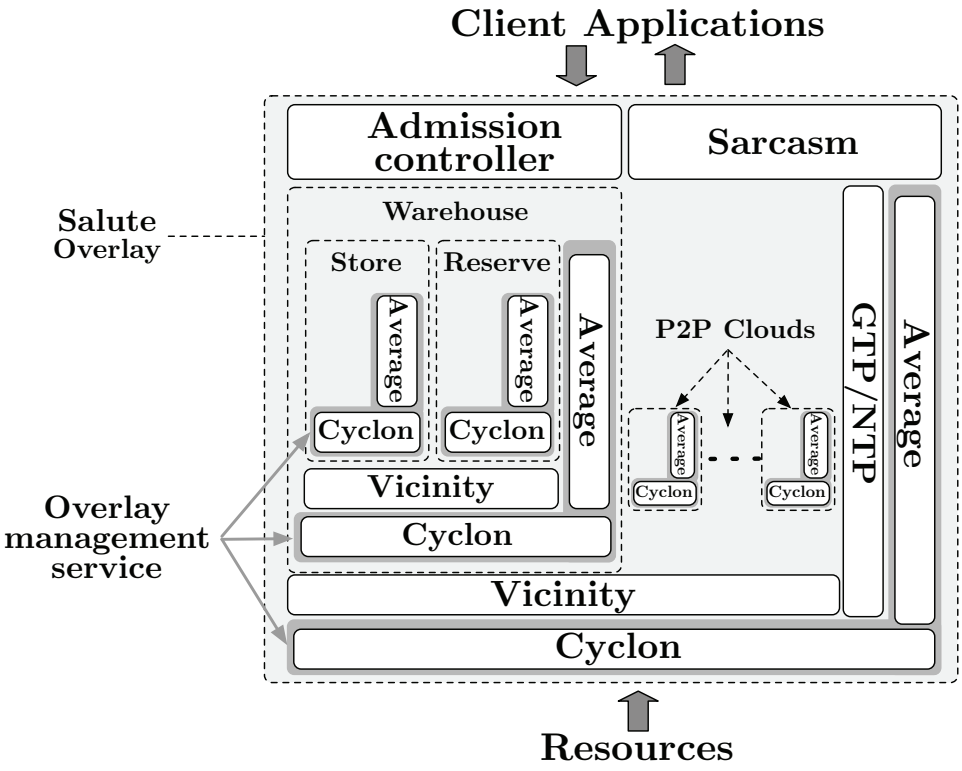


FIGURE 4.5 – Schéma des services de gestion de réseaux virtuels de la plateforme Salute.

déployés dans le cas où le flux d'arrivée des pairs au sein de l'entrepôt deviendrait temporairement inférieur à la somme des flux de départ dans tout le système. Cette valeur dépend fortement de la dynamique de l'environnement hôte. Intuitivement, si le pourcentage $\mathcal{S}$ de pair de réserve est trop grand, alors le système disposera d'un nombre réduit de ressources à allouer pour la création de nouveaux nuages pair-à-pair. En revanche, si le pourcentage $\mathcal{S}$ est trop petit, alors certains nuages pair-à-pair risquent de ne plus pouvoir être approvisionnés pendant une certaine période, et de ce fait de violer leur spécification de déploiement. L'étude d'une évaluation du protocole d'allocation de noeuds de Salute est donnée au sein de du Chapitre 7. Cette étude décrit le comportement du service d'allocation de nuages pair-à-pair en suivant différentes configurations du pourcentage $\mathcal{S}$ de pairs de réserve, et ce, sur trois environnements distribués distincts. Nous avons alors établi, de façon empirique, qu'un nombre de pairs de réserve variant de 5% à 10% était généralement satisfaisant pour maintenir une bonne stabilité des nuages pair-à-pair déployés.

Conclusion : Dans ce chapitre, l'architecture et le fonctionnement global de la plateforme pair-à-pair Salute sont présentés. Cette plateforme utilise des réseaux virtuels multicouches afin de collectionner des ressources informatiques hétérogènes et de provenance variée, puis de les réallouer sous forme de nuages pair-à-pair à des applications clientes. Après la description de haut niveau de la plateforme Salute réalisée dans ce chapitre, les chapitres suivants sont axés sur le détail des deux contributions principales, au sein de la collaboration de protocoles composant le service d'allocation de nuages pair-à-pair, à savoir le protocole de gestion de nuages pair-à-pair « Sarcasm » et les politiques d'allocation de noeuds appliquées par les contrôleurs d'admission de requêtes.

Chapitre 5

Gestion de nuages pair-à-pair avec le protocole Sarcasm

Sommaire

Sarcasm [1] est le protocole de gestion autonome de nuages pair-à-pair. Plus précisément, ce protocole est chargé de la construction, puis du maintien, des allocations de nuages pair-à-pair au sein de Salute. Aussi, son rôle peut être assimilé à la gestion du flux d'approvisionnement de noeuds de chaque nuage. Le travail réalisé par Sarcasm peut être divisé en trois tâches principales : *i)* créer un nuage à partir de sa spécification, lorsque une requête d'allocation de noeuds est acceptée, *ii)* surveiller la taille des nuages alloués et *iii)* fournir, si nécessaire, des noeuds supplémentaires aux nuages pour le maintient de leur spécification, en dépit de la dynamique de l'environnement sous-jacent. Afin de respecter les principes de déploiement de Salute en tant que plateforme pair-à-pair, les contraintes imposées à la réalisation du gestionnaire autonome Sarcasm sont les suivantes :

1. L'exécution du protocole de gestion de nuages doit être totalement décentralisée.
2. Le service proposé doit pouvoir gérer jusqu'à plusieurs milliers de nuages en parallèle.
3. La taille des nuages doit pouvoir varier d'une dizaine à plusieurs milliers de noeuds.

La première section de ce chapitre décrit une vue d'ensemble du fonctionnement de Sarcasm, tandis que la seconde section présente les détails algorithmiques des protocoles implantés par Sarcasm.

1. Scalable Allocation of Resources and Concurrency Aware p2p-cloudS Management.

5.1 Présentation et vue d'ensemble de Sarcasm

Le protocole Sarcasm est utilisé pour créer, puis pour maintenir les allocations de ressources depuis l'entrepôt de noeuds vers les différents nuages pair-à-pair déployés au sein de la plateforme Salute. Cette section donne une vue d'ensemble du service de gestion de nuages pair-à-pair, en présentant le principe de fonctionnement de Sarcasm, et en détaillant les interfaces du service de gestion de réseau virtuel (Cf. Chapitre 4) sur lesquelles Sarcasm s'appuie pour assurer son rôle de gestionnaire de nuages pair-à-pair.

5.1.1 Principe de gestion des nuages pair-à-pair

Dans un but pédagogique, cette section présente les principes de construction et de maintenance des nuages pair-à-pair, avant que leurs implantations respectives ne soient détaillées au sein de la Section 5.2. A noter que le protocole Sarcasm est conçu pour être utilisé au sein de la plateforme Salute, c'est-à-dire que ce protocole de gestion de nuages pair-à-pair fait l'hypothèse que l'architecture du système sur lequel il s'exécute est composée d'un entrepôt de noeuds, d'un magasin, d'une réserve et d'un nombre potentiellement grand de nuages pair-à-pair.

Construction de nuage pair-à-pair Un nuage pair-à-pair est créé à la demande d'un utilisateur. L'attente de cet utilisateur est d'obtenir un réseau virtuel, composé des noeuds correspondants à la spécification du nuage demandé, et ce, dans les plus brefs délais. Aussi Sarcasm doit être capable de contacter rapidement un très grand nombre de pairs libres au sein du magasin de noeuds, afin de les recruter pour construire le nuage demandé. Compte tenu de la nature du réseau virtuel hébergeant ce magasin (réseau virtuel non structuré), Sarcasm utilise un protocole de diffusion épidémique pour accomplir cette tâche.

Le processus exécuté par Sarcasm pour la création d'un nuage pair-à-pair est décrit par le schéma de la Figure 5.1. Son principe est le suivant : quand un pair, disons P_i, demande une allocation de noeuds, il soumet dans un premier temps la spécification de ce nuage, appelons le C_i, aux contrôleurs d'admission du système. Comme il est détaillé dans le chapitre 6,

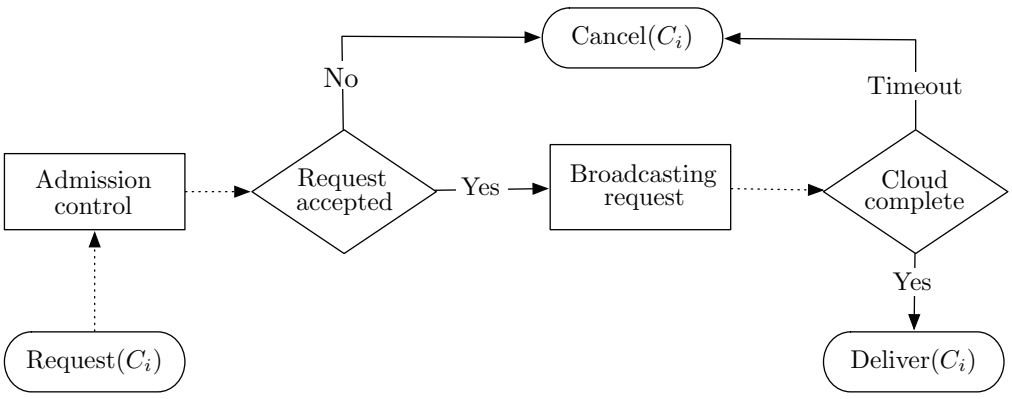


FIGURE 5.1 – Schéma du processus de construction de nuage.

ces contrôleurs d'admission sont exécutés de façon totalement décentralisée et peuvent être consultés localement depuis chaque pair membre de Salute. En utilisant des mécanismes de prédiction de l'évolution du système, ils vont estimer si le nuage demandé peut être alloué

avec sûreté, en fonction de la taille et du temps de déploiement souhaité. S'ils acceptent la requête d'allocation, alors P_i initialisera la construction du réseau virtuel O_i, hébergeant le nuage C_i, et diffusera la requête d'allocation associée à travers le réseau du magasin de noeuds, de façon à recruter des pairs libres pour ce nuage. Parallèlement à la propagation de la requête d'allocation, les pairs libres rejoignent le réseau virtuel O_i et changent leur statut de « libre » à « alloué ». La diffusion de la requête s'achève quand le nuage associé atteint la taille requise. Une fois que le nuage C_i est complet, le pair P_i diffuse une notification au sein de ce même nuage pour indiquer aux pairs qui le composent que la réservation de noeuds commence. Finalement, l'application cliente associée peut être déployée sur le nuage. Si, au contraire, le nuage ne parvient pas à atteindre la taille requise après une période prédéterminée (Cf. Chapitre 7), alors la réservation de noeuds est annulée : le pair demandeur est notifié de l'annulation et les noeuds ayant rejoint le réseau O_i de façon anticipée sont libérés (c.-à-d. reprennent le statut « libre ») et rejoignent l'entrepôt de noeuds. Le protocole décrit ci-dessus est détaillé au sein de la Section 5.2.1.

Maintenance de nuage pair-à-pair La maintenance d'un nuage pair-à-pair comprend deux étapes : la surveillance du nuage en question, puis le réapprovisionnement en ressources, en cas de besoin. Le processus exécuté par Sarcasm pour maintenir la spécification d'un nuage pair-à-pair est décrit au par le schéma de la Figure 5.2. Son principe est le suivant : les noeuds appartenant au nuage surveillent la taille de ce dernier, en consultant périodiquement le service de comptage exécuté sur le réseau virtuel associé au nuage en question. Lorsque les

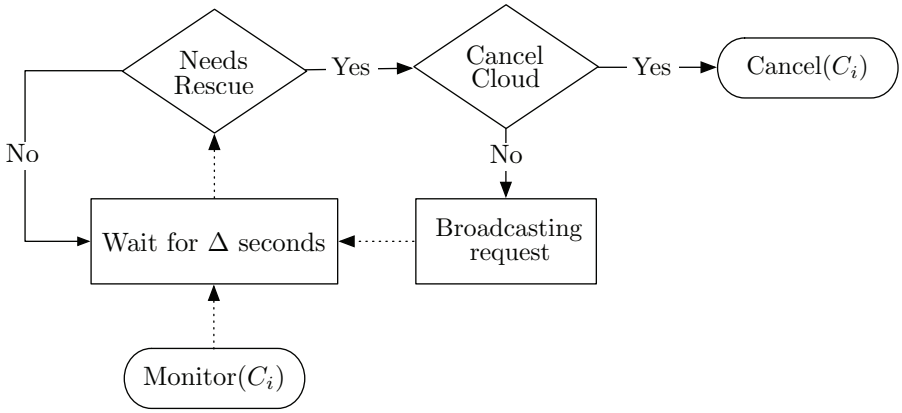


FIGURE 5.2 – Schéma du processus de maintenance de nuage.

pairs détectent que la taille du nuage est inférieure à la taille spécifiée, deux solutions s'offrent à eux : soit ils jugent que le nuage peut être correctement réapprovisionné par les ressources du magasin de noeuds, auquel cas ils émettent une requête de réapprovisionnement dans le but de recruter des pairs libres et des pairs de réserve, soit ils estiment que le déploiement du nuage doit être prématurément interrompu et le nuage pair-à-pair est abandonné. Le protocole décrit ci-dessus est détaillé au sein de la Section 5.2.3.

En résumé, Sarcasm est un protocole pair-à-pair, c'est-à-dire qu'il utilise la collaboration des pairs du système pour remplir sa fonction, et ce, de façon totalement décentralisée. Le problème de la concurrence des requêtes d'allocation est abordé en détail au sein de la Section 5.2.1. Les interfaces du service de gestion de réseau virtuel utilisées par Sarcasm sont décrites dans les paragraphes suivants.

5.1.2 Service de gestion de réseau virtuel

De façon à simplifier la description des algorithmes implantant le protocole Sarcasm, les interfaces d'accès au service de gestion de réseau virtuel présenté dans le Chapitre 4 sont décrites ci-dessous :

- ***Echantillonnage de pairs :*** Afin d'obtenir l'adresse IP d'un pair aléatoirement choisi parmi les membres d'un réseau virtuel O_i, n'importe quel noeud membre de ce réseau peut utiliser la méthode `getRandomPeer`(O_i), fournie par le service d'échantillonnage de pairs exécuté sur le réseau O_i. Ce service d'échantillonnage de pairs est disponible sur tous les réseaux virtuels composant l'architecture multicouche de Salute, c'est-à-dire sur les réseaux suivants : le réseau principal de Salute, l'entrepôt de noeuds, le magasin, la réserve et chaque nuage pair-à-pair.

- ***Joindre un réseau virtuel :*** Pour joindre un réseau pair-à-pair maintenu par un protocole d'échantillonnage de pairs, il suffit de démarrer l'exécution de ce protocole en fournissant en paramètre l'adresse d'un des membres du réseau à joindre. Ce membre est appelé « pair d'introduction ». Dans le cas de Salute, le protocole d'échantillonnage de pairs Cyclon assure qu'une fois que cette étape de démarrage est effectuée, le pair ayant joint le réseau virtuel O_i est intégré de façon fiable au sein du réseau virtuel malgré la dynamique de l'environnement hôte [153]. Au sein de la plateforme Salute, un noeud peut rejoindre un réseau virtuel O_i en utilisant la méthode `join`($O_i, \mathcal{N}_i$), où le paramètre $\mathcal{N}_i$ correspond à un ensemble d'adresses de noeuds d'introduction contenant au moins une entrée. Si l'ensemble $\mathcal{N}_i$ contient plusieurs adresses, alors le pair d'introduction est choisi au hasard parmi cet ensemble. La construction de l'ensemble d'adresses $\mathcal{N}_i$ peut être réalisée de différentes façons au sein de la plateforme Salute. Par exemple, lorsqu'un pair joint le système pour la première fois, c'est le protocole de construction de réseau superposé Vicinity qui est en charge de trouver des noeuds d'introductions pour que ce pair puisse joindre l'entrepôt de noeuds. Pour la création de nuages pair-à-pair, c'est le protocole Sarcasm qui est en charge de construire cet ensemble de noeuds d'introduction. La gestion du paramètre $\mathcal{N}_i$ par Sarcasm est détaillée au sein de la section suivante.

- ***Quitter un réseau virtuel :*** La méthode `leave`(O_i) permet de quitter volontairement un réseau virtuel O_i. Dans le contexte particulier de Salute, cette méthode est appelée lorsqu'un noeud quitte un nuage pair-à-pair ou quitte l'entrepôt de noeuds. Ce pair cesse donc l'exécution du service d'échantillonnage de pairs associé au réseau virtuel O_i. A noter que l'impact du départ d'un noeud au sein d'un réseau virtuel non structuré est traité de façon native par les protocoles épidémiques d'échantillonnage de pairs, et en particulier par le protocole Cyclon qui implante ce service au sein de la plateforme Salute [153]. Plus précisément, Cyclon assure qu'au bout d'un temps borné, les identifiants des pairs ayant quitté le réseau virtuel disparaissent des vues locales utilisées par les pairs pour échantillonner ce même réseau (Cf. Chapitre 3).

- ***Estimer la taille d'un réseau virtuel :*** Lorsqu'un pair souhaite connaître la taille d'un réseau virtuel, il utilise la méthode `getSize`(O_i), où O_i est l'identifiant du réseau virtuel ciblé. Si un noeud souhaite évaluer la taille d'un réseau dont il est membre, il lui suffit de consulter le service pair-à-pair de comptage de pairs qui est exécuté sur ce réseau. Nous faisons également l'hypothèse que grâce à l'exécution du protocole Vicinity sur le réseau principal de Salute, chaque noeud peut facilement obtenir l'identifiant d'au

moins un pair appartenant à chacune des collections suivantes : l'entrepôt de noeuds, le magasin et la réserve. De cette façon, si un pair P_i souhaite évaluer la taille d'une de ces collections de noeuds, alors qu'il n'en fait pas partie, il peut émettre une requête de consultation à un pair membre du réseau ciblé, dont l'adresse sera fournie par le protocole Vicinity. Ainsi, à l'aide d'un simple échange de message, le pair P_i peut connaître l'estimation de la taille du réseau O_i.

- ***Récupérer une estampille horaire synchronisée :*** Chaque pair membre de la plateforme pair-à-pair Salute peut obtenir une estampille horaire locale qui est synchronisée avec l'ensemble des autres membres. La méthode utilisée pour produire une telle estampille est la méthode `getSynchTimeOfDay`(), elle utilise directement le service pair-à-pair de synchronisation pour produire une estampille composée de la date d'invocation de la méthode, synchronisée à la seconde prés, et de l'identifiant du pair ayant invoqué cette méthode. Ce type d'estampille peut être utilisé par un pair pour dater ses propres actions, par exemple pour dater ses requêtes d'allocation de nuage pair-à-pair. Il est important de noter que ces estampilles horaires ont pour seule vocation de dater des événements relatifs à un pair donné, et en aucun cas d'établir un ordre logique entre des événements physiquement distribués sur plusieurs machines, comme c'est par exemple le cas des horloges logiques de Lamport [101].

Pour résumer, chaque pair membre de la plateforme Salute est capable de **joindre**, de **quitter** et de **compter** la composition, en terme de noeuds, de n'importe quel réseau virtuel faisant partie de l'architecture multicouche du système. De plus, ces pairs sont **synchronisés** à la seconde près et sont capables d'établir des estampilles horaires locales, tout cela, grâce au service de gestion de réseau virtuel. Les interfaces d'utilisation de ce service sont résumées au sein de la Table 5.1. Ces mêmes interfaces sont utilisées pour la description détaillée du protocole Sarcasm dans la Section suivante.

Nom de la méthode	**Paramètre(s)**	**Service(s) p2p impliqué(s)**
`getRandomPeer(`O_i`)`	O_i : réseau virtuel cible	échantillonnage
`join(`O_i`,`$\mathcal{N}_i$`)`	O_i, $\mathcal{N}_i$: ensemble d'adresses de pairs d'introduction	échantillonnage
`leave(`O_i`)`	O_i	échantillonnage
`getSize(`O_i`)`	O_i	comptage
`getSynchTimeOfDay()`	$\emptyset$	synchronisation

TABLE 5.1 – Interfaces du service pair-à-pair de gestion de réseau virtuel.

Pour conclure, Sarcasm est un protocole pair-à-pair permettant de construire, puis de maintenir des nuages pair-à-pair au sein de la plateforme Salute. Ce protocole est basé sur l'utilisation d'un service de gestion de réseau virtuel, dont les interfaces d'utilisation sont décrites ci-dessus. La section suivante décrit les algorithmes implantant le protocole Sarcasm et étudie les impacts de la concurrence des requêtes d'allocation sur le service de gestion de nuages pair-à-pair tel qu'il a été présenté jusqu'ici.

5.2 Algorithmes d'implantation de Sarcasm

Cette section décrit l'implantation du protocole Sarcasm en utilisant un pseudo-code inspiré des langages de programmation orientés objet. En particulier, les algorithmes présentés dans cette section font l'hypothèse que chaque pair peut accéder à un objet `OV-MAN`, qui représente le service de gestion de réseau virtuel dont les interfaces sont décrites dans la section précédente. De plus, le fonctionnement des protocoles décrits au sein de cette section dépend des hypothèses présentées dans le paragraphe suivant.

Hypothèses :

- Les noeuds participant à l'exécution du protocole Sarcasm ne subissent que des pannes franches. Plus particulièrement, la résistance du protocole à des comportements malicieux, de types égoïstes ou byzantins, n'est pas abordée dans cette thèse.

- L'environnement hébergeant la plateforme Salute est composé d'un réseau dont la topologie peut être modélisée par un graphe connecté, du type D'Internet. En particulier, nous faisons l'hypothèse qu'il suffit qu'un noeud n_i connaisse l'adresse IP d'un autre noeud n_j pour qu'un message puisse être routé depuis n_i jusqu'à n_j.

- Les noeuds de Salute sont synchronisés selon une période Δ_{Sync} (Cf. Chapitre 4).

- La borne supérieure sur le temps nécessaire au protocole de comptage de pairs pour évaluer la taille d'un réseau virtuel est connue et correspond à la constante Δ_{Count}.

- La durée de transfert d'un message entre deux noeuds, c'est-à-dire d'une communication de type point à point, est bornée par la constante $\Delta_{Unicast}$.

- La constante Δ_{Alloc} définit l'unité de temps utilisée pour spécifier la durée de déploiement des nuages alloués par Salute.

- La dynamique moyenne de l'environnement hôte, en nombre de départs et d'arrivées de noeuds par minute, est suffisamment basse pour que les protocoles épidémiques utilisés par Salute puisse fonctionner correctement.

- Pour la construction de chaque nuage C_i, le pair P_i étant à l'origine de la diffusion de la requête d'allocation de noeuds associée, ne tombe jamais en panne.

Cette dernière hypothèse constitue le seul élément de centralisation de la plateforme Salute. En effet, le pair P_i est utilisé à deux reprises par le protocole de construction de nuages pair-à-pair : c'est lui qui déclare le début du déploiement du nuage C_i, et c'est lui qui déclare l'annulation de ce déploiement. Dans les deux cas, la décision prise par le pair P_i est diffusée à l'ensemble des pairs alloués au sein du nuage C_i dans le but de synchroniser l'ensemble de ses membres sur l'une de ces deux actions. Pour simplifier les algorithmes utilisés par le protocole Sarcasm, nous supposons que le pair P_i ne tombe jamais en panne. Cependant, nous pensons que cette hypothèse peut être facilement levée en considérant un mécanisme de réplication simple, dans lequel le pair P_i choisit aléatoirement un nombre prédéfini de pairs mandataires pour prendre les décisions de synchronisation ou d'annulation à sa place, s'il tombe en panne.

Exemple de scénario de déploiement : Pour donner une idée de la valeur des constantes utilisées dans les hypothèses décrites ci-dessus, nous proposons de considérer un scénario dans lequel la plateforme Salute est déployée comme système pair-à-pair public, à travers Internet. Pour ce scénario la borne $\Delta_{Unicast}$ peut être fixée à 1 seconde. D'après l'étude de Ledlie et coll. [102], cette hypothèse est valable pour plus de 99,5% des communications réalisées via Internet.

Ensuite, la borne Δ_{Count} peut être fixée à une valeur de 40 secondes. Cette borne est donnée par l'évaluation du protocole Average [121], qui est utilisé comme protocole de comptage de pairs par Salute. Cette évaluation est présentée au sein du Chapitre 3 et concerne des systèmes dont la taille est inférieure ou égale à 2^{18}.

En considérant la latence du service de comptage, on peut considérer qu'une bonne unité de temps pour l'allocation des nuages posséderait une valeur de l'ordre de la minute. Par conséquent, nous proposons de fixer la valeur de la constante Δ_{Alloc} à une minute.

Finalement, le taux de churn moyen par unité de temps est inférieur à 0,05% de la taille du système. Cette hypothèse est validée par les traces collectées par Brighten Godfrey [68], qui servent de références en matière de simulation de la dynamique des environnements pair-à-pair dans cette thèse. Ces traces, présentées en détail au sein du Chapitre 7, représentent le système Skype [35, 73], l'environnement de machines de bureau de l'entreprise Microsoft utilisée dans l'étude de Bolosky et coll. [43] et le système PlanetLab [36, 142]. A titre d'exemple, le taux de churn moyen estimé par minute pour ces systèmes est de : 0,04% pour Skype, de 0,01 pour le système de Microsoft et de 0,005 pour PlanetLab (Cf. Chapitre 7). Ces dernières mesures sont donc conformes à l'hypothèse formulée sur le taux de churn des membres de Salute.

Dans une première partie, cette section donne le détail du protocole de construction de nuage de noeuds. Une seconde partie présente les protocoles utilisés par Sarcasm pour gérer la concurrence des requêtes d'allocation. Une troisième partie présente le protocole de maintenance des nuages pair-à-pair. Il est important de noter que le protocole de filtrage des pairs, en fonction de leurs attributs, n'est pas traité dans les algorithmes présentés lors de ces trois premières parties, pour des raisons pédagogiques. Aussi, le traitement des filtres appliqués aux nuages de noeuds est présenté dans une quatrième partie de cette Section, en tant que raffinement des algorithmes présentés dans les trois premières sections.

5.2.1 Détails du protocole de construction de nuage de noeuds

Comme expliqué précédemment, lorsqu'un utilisateur souhaite réserver un nuage pair-à-pair au sein de la plateforme Salute, il diffuse la spécification de ce nuage sur le réseau virtuel hébergeant le magasin de noeuds. Le traitement de cette requête est alors assuré de façon collaborative par les pairs membres du magasin. Ce traitement peut être divisé en deux étapes : premièrement, les pairs du magasin de noeuds diffusent la requête d'allocation de façon à recruter suffisamment de pairs libres pour remplir le nuage demandé. Dans un second temps, les noeuds alloués pour ce nuage en construction se synchronisent de façon à déclarer le démarrage du déploiement en question. Les paragraphes suivants décrivent les algorithmes utilisés par les pairs du magasin de noeuds pour réaliser ces deux étapes.

Emission d'une requête d'allocation de nuage pair-à-pair : Lorsqu'un pair P_i souhaite construire un nuage pair-à-pair C_i, il initialise le protocole d'échantillonnage de pairs qui est chargé de maintenir le réseau virtuel O_i hébergeant ce nuage. Initialement, P_i est le seul

noeud appartenant au réseau O_i. Afin de recruter d'autres noeuds pour le nuage à construire, le pair P_i diffuse un message M_i au sein du magasin de noeuds. Le protocole utilisé pour initialiser cette diffusion est illustré au sein de la Figure 5.3. Dans un premier temps, le pair P_i construit le message de requête M_i. Ce message contient l'identifiant P_i, une estampille horaire τ_i obtenue grâce au service de gestion de réseau virtuel, la taille S_i du nuage C_i à construire, un ensemble $\mathcal{N}_i$ d'adresses de pairs d'introduction et le numéro E de l'étape de diffusion (lignes 1 à 7). L'ensemble $\mathcal{N}_i$ ne contient initialement que l'adresse de P_i. Lorsque le message M_i est reçu par des pairs libres recrutés pour le nuage C_i, P_i est utilisé comme pair d'introduction pour permettre à ces pairs libres de rejoindre le réseau O_i (Cf. Section 5.1.2). Au fur et à mesure que la requête d'allocation de noeuds progresse dans le magasin, des noeuds intermédiaires ajoutent leur adresse au sein de l'ensemble $\mathcal{N}_i$ (Cf. protocole de recrutement de noeuds). A noter que l'identifiant du réseau virtuel O_i correspond à l'identifiant du nuage C_i. Cet identifiant est calculé à partir de l'identifiant du pair P_i et de l'estampille horaire τ_i, si bien que chaque nuage est identifié de manière unique au sein du système. Comme pour une

CLOUD CONSTRUCTION INITIALIZATION PROTOCOL

```
1: procedure RequestCloud(S_i)
2:     N_i ← {P_i}
3:     E ← 1
4:     τ_i ← OV-MAN.getSynchTimeOfDay()
5:     M_i ← < P_i, τ_i, S_i, N_i, E >
6:     propagate(Store, M_i)
7: end
8: procedure propagate(O_i, M_i)
9:     repeat
10:        P_j ← OV-MAN.getRandomPeer(O_i)
11:        send M_i to P_j
12:    until F messages have been sent
13: end
```

FIGURE 5.3 – Pseudo-code du protocole de requête de construction de nuage pair-à-pair.

diffusion épidémique classique, la propagation de la requête se fait en expédiant le message M_i à un nombre prédéterminé F de pairs, tous choisis au hasard par le service d'échantillonnage de pairs (lignes 8 à 13). Finalement, d'après les études présentées dans le Chapitre 3, pour qu'une diffusion épidémique touche un nombre S_i de noeuds en F étapes, avec une grande probabilité, la valeur du fanout doit être fixée telle que $F = Log(S_i)$.

Les paragraphes suivants décrivent les algorithmes utilisés par les pairs du magasin de noeuds pour traiter cette requête et construire le nuage demandé. Afin de simplifier la description du protocole Sarcasm, seule la taille du nuage demandé est prise en compte dans la requête d'allocation. Le traitement de requêtes plus complexes, incluant un ensemble de filtres pour les attributs des pairs à recruter, est détaillé au sein de la Section 5.2.4.

Recrutement de noeuds : La diffusion utilisée par Sarcasm pour recruter les pairs libres diffère des diffusions épidémiques classiques, telles que décrites au sein du Chapitre 3. En effet, sa progression au sein du magasin de noeuds est « synchronisée » de façon à permettre au protocole d'évaluer la taille du nuage en construction lors de chaque étape de la diffusion. Ainsi, lorsque le nuage à construire est complet, la diffusion est stoppée.

La diffusion épidémique d'un message m est un processus itératif, au sein duquel les noeuds ayant reçu le message m pour la première fois lors de l'étape E, réexpédient chacun ce message à un nombre fixe F de destinataires lors de l'étape $E+1$ (Cf. Chapitre 3). Dans une diffusion épidémique classique, les pairs réexpédient le message m dès qu'ils le reçoivent, sous réserve qu'ils le reçoivent pour la première fois. Dans la diffusion épidémique utilisée par Sarcasm, les pairs libres recevant un message M_i de requête d'allocation, pour la première fois lors de l'étape E, réagissent différemment. Dans un premier temps, ils joignent le réseau virtuel O_i

hébergeant le nuage à construire, et ce, dès réception du message. Ensuite, ces mêmes pairs attendent une date T_E prédéterminée pour déclencher la réexpédition du message M_i. Cette attente a pour but d'estimer la taille du nuage en construction, en tenant compte des pairs ayant potentiellement joint ce nuage lors du début de l'étape E. De cette façon, les pairs peuvent évaluer si la diffusion doit être poursuivie en déclenchant une nouvelle étape $E+1$ pour compléter ce nuage ou si la diffusion doit être stoppée. La date T_E est calculée selon l'expression suivante :

$$T_E = \tau_i + E * \Delta_{Count},$$

où τ_i est la date à laquelle la diffusion a été initiée, où E est le nombre d'étapes exécutées depuis l'initialisation de la diffusion et où Δ_{Count} est une période prédéterminée de l'ordre de la minute. Le calcul de la valeur de la période Δ_{Count} dépend de résultats empiriques, il est donc présenté avec les évaluations du protocole Sarcasm, au sein du Chapitre 7. En particulier, la période Δ_{Count} est supposée être largement supérieure au temps nécessaire pour transmettre un message à F destinataires et au temps nécessaire à ces destinataires pour rejoindre le réseau virtuel O_i. Toutes les étapes de la diffusion épidémique utilisée par Sarcasm sont alors synchronisées sur les dates $T_2, T_3, ..., T_E$, de façon à ce que les pairs déclenchent les étapes de réexpédition $2, 3, ..., E$, au même moment, à la seconde prés. Le processus de réexpédition du message M_i est répété jusqu'à ce que le nuage à construire ait atteint une taille supérieure ou égale à la taille S_i demandée, ou jusqu'à ce que tous les pairs du magasin aient reçu ce message. Si la taille du nuage est largement supérieure à la taille S_i, un protocole de déchargement est exécuté pour libérer l'excédent de noeuds. Finalement, les évaluations de Sarcasm, présentées au sein du Chapitre 7, montrent que ce protocole de construction de nuages a rarement besoin de plus de 5 ou 6 étapes de diffusion pour construire des nuages pair-à-pair de taille inférieure ou égale à 5000 noeuds.

Le pseudo-code du protocole utilisé par Sarcasm pour construire un nuage de noeuds C_i est donné dans la Figure 5.4. Il décrit les différentes procédures exécutées lorsqu'un pair P_R, membre du magasin de noeuds, reçoit un message M_i portant une requête d'allocation pour le nuage C_i. Ce message M_i contient l'identifiant P_i du pair ayant émis la requête d'allocation, l'estampille horaire τ_i indiquant la date d'émission de la requête, la taille S_i demandée pour le nuage C_i, un ensemble $\mathcal{N}_i$ de noeuds de contacts qui appartiennent au réseau virtuel O_i et le paramètre E correspondant au nombre d'étapes de diffusion déjà réalisé par Sarcasm au moment de la réception du message.

CLOUD CONSTRUCTION PROTOCOL

```
1: upon receive(M_i :< P_i, τ_i, S_i, N_i, E >) do
2:    if M_i ∉ receivedMessage then
3:       receivedMessage ← receivedMessage ∪ M_i
4:       joinOverlay(O_i, N_i)
5:       forward(M_i)
6:    end if
7: end upon

8: procedure joinOverlay(O_i, N_i)
9:    OV-MAN.join(O_i, N_i)
10:   status ← booked
11: end

11: procedure forward(M_i :< P_i, τ_i, S_i, N_i, E >)
12:   T_E ← τ_i + E * Δ_Count
13:   waitUntill(T_E)
14:   S'_i ← OV-MAN.getSize(C_i)
15:   if S'_i < S_i then
16:      E ← E + 1
17:      N_i ← N_i ∪ P_R
18:      M'_i ← < P_i, τ_i, S_i, N_i, E >
19:      propagate(Store, M'_i)
20:   end if
21: end
```

FIGURE 5.4 – Pseudo-code du protocole de construction de nuage pair-à-pair de Sarcasm.

Comme lors d'une diffusion épidémique classique, le pair P_R qui suit le protocole Sarcasm,

ne traite un message de requête d'allocation M_i que s'il le reçoit pour la première fois (lignes 1 et 2). Si le message est traité, alors P_R rejoint le réseau virtuel hébergeant le nuage C_i, puis change sont statut de « libre » à « alloué » (lignes 4 et 8 à 11). Ensuite, ce pair invoque la méthode `forward`($M_i :< P_i, \tau_i, S_i, \mathcal{N}_i, E >$), qui est chargée de la réexpédition du message de requête d'allocation. Cette méthode débute son exécution par une attente active, qui dure jusqu'à la date T_E (ligne 12 et 13). Comme expliqué précédemment, cette attente a pour but de permettre au service de comptage de pairs de prendre en compte le nombre de pairs ayant rejoint le nuage C_i lors du début de l'étape de diffusion E. Une fois cette période écoulée, le pair P_R estime la taille S'_i du nuage à construire, à l'aide du service de gestion de réseau virtuel (ligne 14). Si cette taille S'_i est supérieure ou égale à la taille S_i requise pour le déploiement du nuage C_i, alors la diffusion du message M_i est terminée. Au contraire, si le nuage n'est pas encore complet, c'est-à-dire si la taille S'_i est inférieure à la taille S_i, alors P_R continue la diffusion en réexpédiant un message M'_i au sein du magasin de noeuds (ligne 19 et 20). Ce message M'_i est une copie du message initial M_i, dans lequel la valeur de l'étape de diffusion courante est mise à jour (ligne 16) et dans lequel l'adresse de P_R a été ajoutée à l'ensemble d'adresses de noeuds de contact $\mathcal{N}_i$.

Le protocole décrit ci-dessus assure que la diffusion du message de requête progresse jusqu'à ce que la taille du nuage C_i soit strictement supérieure à la taille spécifiée S_i, ou jusqu'à ce que tous les pairs libres du magasin de noeuds aient reçu le message de requête. Le paragraphe suivant décrit le protocole de déchargement appliqué pour stabiliser la taille du nuage C_i autour de la valeur S_i et explique comment les pairs ayant rejoint le nuage C_i se synchronisent pour déclarer le début du déploiement de ce nuage.

Gestion du déploiement d'un nuage pair-à-pair : Le paragraphe suivant décrit comment Sarcasm surveille l'évolution d'un nuage pair-à-pair tout au long de son cycle de vie. Cette surveillance a pour but de détecter la complétude du nuage durant son processus de construction et de déclarer le début de sa période de déploiement, mais aussi de surveiller l'évolution de sa taille et de déclencher des actions appropriées au maintien de cette taille, en dépit de la dynamique de l'environnement hôte.

Tout d'abord, une fois que le nuage C_i est complet, Sarcasm doit déclarer le début de sa période de déploiement, puis communiquer cette date à l'ensemble des pairs qui y sont alloués. Pour ce faire, il utilise une routine qui surveille périodiquement l'évolution du nuage en construction, et qui décrète le début du déploiement du nuage C_i, lorsqu'elle détecte que celui-ci est complet. Pour simplifier la description des protocoles qui va suivre, on considérera dans un premier temps, que cette routine est localement exécutée par le pair P_i (c.-à-d. le pair ayant émis la requête d'allocation et qui est supposé ne jamais tomber en panne). Une méthode permettant de distribuer une partie de l'exécution de cette routine à l'ensemble des noeuds du nuage, est discutée au sein de la Section 5.2.2. Le pseudo-code de la routine de surveillance est donné sur la Figure 5.5, au sein de laquelle elle est représentée par la tâche `monitorCloud(<`C_i, S_i, L_i`>)`. Cette tâche est exécutée périodiquement en suivant la période Δ_{Count}, relative au temps nécessaire pour que le service pair-à-pair de comptage de pairs puisse mettre à jour l'estimation de la taille du nuage ciblé. Cette routine comporte trois paramètres : l'identifiant C_i du nuage à surveiller, sa taille de référence S_i, ainsi que sa durée de déploiement L_i. Elle est exécutée pendant tout le cycle de vie du nuage C_i, c'est-à-dire depuis le déclenchement de sa requête d'allocation, jusqu'à la date τ_{end} indiquant la fin de son déploiement.

Lors de chaque exécution, la routine de surveillance évalue la taille S'_i du nuage C_i à l'aide du service de gestion de réseau virtuel (ligne 3). Elle utilise une variable booléenne

CLOUD STARTING AND ENDING PROTOCOL

```
1: task monitorCloud(< C_i, S_i, L_i >)
2:   for every Δ_Count seconds do
3:     S'_i ← OV-MAN.getSize(C_i)
4:     if ¬started ∧ S'_i ≥ S_i then
5:       started ← true
6:       τ ← OV-MAN.getSyncTimeOfDay()
7:       τ_start ← τ + Δ_Bdcast
8:       τ_end ← τ_start + L_i
9:       propagate(O_i, < START, τ_start, τ_end >)
10:    else if started ∧ S'_i < S_i then
11:      lightRequest(....)
12:    else if started ∧ S'_i > S_i + offset then
13:      propagate(O_i, < UNLOAD, S_i, S'_i >)
14:    end if
15:  end for
16: end

17: procedure receive(UNLOAD, S_i, S'_i)
18:   nbToRelease ← S'_i − S_i
19:   rdnValue ← getRndValue(0,S'_i)
20:   if rdnValue ≤ nbToRelease then
21:     release()
22:   end if
23: end

24: upon receive(< START, τ_start, τ_end >) do
25:   for every minute do
26:     τ_j ← OV-MAN.getSyncTimeOfDay()
27:     if τ_j > τ_end then
28:       release()
29:     end if
30:   end for
31: end upon

32: upon release() do
33:   OV_MAN.leave(C_i)
34:   OV_MAN.join(Warehouse)
35:   NbSafe ← OV-MAN.getSize(Reserve)
36:   if NbSafe < minimumSafe then
37:     status ← safe
38:     OV_MAN.join(Reserve)
39:   else
40:     status ← free
41:     OV_MAN.join(Store)
42:   end if
43: end upon
```

FIGURE 5.5 – Pseudo-code du protocole de démarrage d'allocation de Sarcasm.

started, afin de sauvegarder l'état du nuage ciblé, c'est-à-dire pour mémoriser si sa période de déploiement a commencé ou non. On peut alors distinguer trois cas selon les valeurs des variables S_i, S'_i et *started* :

1. La taille du nuage C_i est supérieure à sa taille de référence et le déploiement n'a pas démarré (ligne 4). Dans ce cas, le pair P_i qui exécute la routine de surveillance marque le début du déploiement en activant la variable *started* (ligne 5). Ce même pair définit ensuite la date τ_{start} correspondant au début de la période de déploiement du nuage C_i, et la date τ_{end} correspondant à la fin de la période de déploiement de ce même nuage (lignes 6 à 8). Puis il diffuse une notification de démarrage de déploiement contenant ces deux dates à l'ensemble des pairs alloués au sein du nuage C_i. La date τ_{start} est définie à partir de la date locale τ, correspondant au moment où la routine a détecté la complétude du nuage C_i, à laquelle est ajoutée une période Δ_{Bdcast}. Cette dernière période assure que la diffusion des deux dates sera terminée lorsque la période de déploiement débutera. A titre d'exemple, la valeur de Δ_{Bdcast} est fixée empiriquement à 6 secondes dans les évaluations présentées au sein du Chapitre 7. Finalement, la date τ_{end} est calculée en additionnant la durée de déploiement L_i à la date τ_{start}.

2. La taille du nuage C_i est inférieure à sa taille de référence et le déploiement a démarré (ligne 10). C'est le moment pour le protocole Sarcasm de recruter des pairs supplémentaires pour maintenir la spécification du nuage pair-à-pair C_i. Pour ce faire, la routine de surveillance invoque la méthode `lightRequest`(....) dont l'exécution est détaillée au sein de la Section 5.2.3.

3. La taille du nuage C_i est supérieure à sa taille de référence et le déploiement a démarré (ligne 12). Ce cas signifie que le nuage C_i possède plus de pairs que nécessaire. A noter que le service d'allocation de noeuds offert par Salute tolère ce cas de figure, jusqu'à

un certain seuil. En effet, on admet que la taille d'un nuage en cours de déploiement soit supérieure à sa taille de référence, tant que le nombre de pairs en excédant ne dépasse pas une valeur prédéfinie appelée $offset$. Cette valeur est calculée à partir d'un pourcentage de la taille de référence du nuage, et il est approprié de l'établir en fonction de la dynamique de l'environnement hôte. Par exemple, la valeur par défaut de cet excédant toléré est fixée à 5% dans les évaluations présentées au sein du Chapitre 7. Finalement, si la routine détecte que le nuage en question possède un excédant de pairs, alors elle diffuse une requête de déchargement au sein du réseau virtuel hébergeant ce nuage. Le traitement de cette requête est décrit dans les paragraphes suivants, il a pour but de relâcher une partie des pairs alloués au nuage à surveiller, afin que sa taille se stabilise autour de la valeur de référence S_i.

Lorsqu'un pair membre du nuage C_i reçoit la notification de démarrage du déploiement (ligne 24), il commence l'exécution d'une nouvelle routine exécutée toutes les minutes, qui a pour but de détecter la fin du déploiement du nuage C_i (lignes 25 à 30). Lorsqu'un noeud détecte la fin du déploiement du nuage au sein duquel il est alloué, il invoque la méthode `release()` chargée de piloter le retour des pairs au sein de l'entrepôt de noeuds. Conformément au protocole de gestion de l'entrepôt décrit au sein du Chapitre 4, lorsqu'un pair rejoint l'entrepôt de noeuds, il choisit sont nouveau statut (« libre » ou « de réserve ») en fonction du pourcentage de pairs de réserve présent au sein du système (lignes 32 à 43). Si la réserve de noeuds contient suffisamment de pairs, alors ce noeud prend le statut « libre », et dans le cas contraire il prend le statut de pair « de réserve ».

Lorsqu'un pair membre du nuage C_i reçoit une requête de déchargement (ligne 17), il calcule le nombre $nbToRelease$ de noeuds excédants présents au sein de ce nuage (ligne 18). Ensuite, ce même pair choisit un nombre entier au hasard compris entre zéro et la valeur S'_i, grâce à la méthode `getRndValue(0,`S'_i`)` proposée par un générateur de nombre aléatoire. La valeur ainsi obtenue est stockée au sein de la variable $rdnValue$ (ligne 19). Si cette valeur est inférieure ou égale au nombre de pairs en excédant $nbToRealse$, alors ce pair quitte le nuage C_i (lignes 20 à 22). Cette méthode d'action distribuée, conditionnée par une variable aléatoire, est inspirée du protocole de morcellement absolu de Montresor et coll. [124]. Si elle est exécutée par l'intégralité des S'_i pairs présents au sein du nuage à décharger, elle assure qu'un nombre de pairs proche de la valeur $nbToRelease$ choisit de quitter ce nuage. L'exécution de ce protocole de déchargement a donc pour effet de faire converger la taille du nuage C_i vers sa taille de référence S_i. Le générateur aléatoire de nombres utilisé pour implanter cette méthode est le Merseme Twister [111].

En utilisant les protocoles décrits au sein de cette section, Sarcasm est capable de construire des nuages pair-à-pair de tailles multiples, tout en héritant des performances accordées aux protocoles de diffusion épidémique classiques. Le service de construction de nuage en résultant est donc rapide, robuste à la dynamique des réseaux non fiables et permet de construire des nuages de tailles potentiellement grandes. Une évaluation de ce protocole, confirmant ces performances, est proposée au sein du Chapitre 7.

5.2.2 Gestion de la concurrence des requêtes d'allocation

En raison de la grande taille attendue pour un système pair-à-pair tel que Salute, l'existence de nombreuses requêtes d'allocation de nuages *concurrentes* est un événement hautement probable que Sarcasm doit prendre en compte. Dans la description qui suit, deux (ou plus) requêtes sont dites *concurrentes* si les processus de création de leurs nuages s'entrecroisent, c'est-à-dire si deux instances du protocole de diffusion épidémique utilisé par Sarcasm sont

exécutées au sein du magasin de noeuds en même temps. Nous définissons également le degré de concurrence $\mathcal{D}c$ des requêtes d'allocation comme le nombre moyen de requêtes de recrutement par noeud libre. En d'autres termes, le degré de concurrence est calculé selon l'équation suivante :

$$\mathcal{D}c = \frac{N_{requested}}{N_{free}},$$

où $N_{requested}$ est la somme des nombres de noeuds demandés par toutes les requêtes concurrentes et où N_{free} est le nombre de noeuds libres disponibles dans le magasin. En tenant compte de la concurrence potentielle des requêtes d'allocation, il existe des cas pour lesquels Sarcasm ne peut allouer qu'une partie des noeuds libres à tous les nuages en construction, sans garantir qu'un seul d'entre eux puisse être complété.

Problème de concurrence : Afin d'illustrer le phénomène problématique engendré par la concurrence des requêtes au sein de Salute, la Figure 5.6 décrit un scénario dans lequel deux requêtes concurrentes sont diffusées en utilisant le protocole de diffusion épidémique de Sarcasm, tel qu'il est défini dans la section précédente. Le problème décrit dans la partie gauche

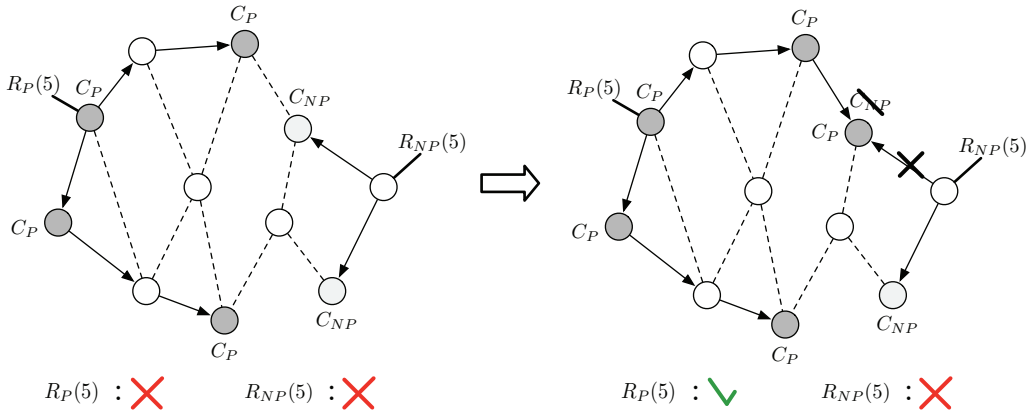


FIGURE 5.6 – Illustration du problème des requêtes d'allocation concurrentes.

de cette figure est l'échec de la construction de deux nuages concurrents C_P et C_{NP}, alors que chacun d'eux aurait pu être déployé avec succès en l'absence de l'autre. Dans ce scénario, deux requêtes d'allocation de noeuds, respectivement nommée R_P et R_{NP}, essaient chacune de collecter cinq noeuds pour construire un nouveau nuage pair-à-pair, alors que le système ne contient que six pairs libres. Ce scénario de requête présente un degré de concurrence $\mathcal{D}c = \frac{5}{3}$. A noter que lorsque le degré de concurrence des requêtes dépasse la valeur 1 (la somme des noeuds demandés dépasse le nombre de noeuds libres), il existe obligatoirement au moins une requête qui ne peut pas être satisfaite. Aussi, dans le scénario décrit sur la Figure 5.6, seule une des deux requêtes peut être théoriquement satisfaite. En utilisant le protocole de construction de nuage de Sarcasm décrit dans la Figure 5.4, R_P a réussi à réserver quatre pairs (ceux étiquetés C_P), alors que R_{NP} n'a réussi à réserver que deux pairs (ceux étiquetés C_{NP}). Par conséquent, les deux nuages en construction sont incomplets et abandonnent leur déploiement après le délai d'annulation prédéterminé. On parlera alors d'interblocage des deux requêtes concurrentes, dans le sens où leur exécution concurrente a conduit à un double échec, alors que chaque requête aurait pu être satisfaite si elle avait été diffusée seule au sein du magasin de noeuds.

A noter que le caractère stochastique de la diffusion épidémique utilisée par Sarcasm rend un tel événement imprévisible. Ce phénomène a été découvert lors de plusieurs simulations du protocole de construction de nuage de Sarcasm, dans le cadre des évaluations présentées au sein du Chapitre 7. En particulier, ces évaluations constatent de façon empirique que le succès des requêtes d'allocation de noeuds est inversement proportionnel au degré de concurrence. Comme illustré précédemment, il suffit que quelques requêtes d'allocations soient en concurrence, pour qu'une situation d'interblocage apparaisse. La solution choisie pour résoudre ce problème de concurrence, de façon décentralisée et sans faire aucune hypothèse sur l'ordonnancement des requêtes d'allocation, est le vol de noeuds par priorité.

Vol de noeuds par priorité : Dans le but d'optimiser les performances du protocole d'allocation de noeuds, Sarcasm fournit un service de construction de nuage adapté à la concurrence potentielle des requêtes d'allocation. Ce service utilise une version alternative du protocole de diffusion épidémique présenté dans la Figure 5.4 qui autorise le vol de noeud par priorité. Le principe de ce dernier protocole est d'associer des priorités déterministes et absolues aux requêtes d'allocations de noeuds, afin de toujours pouvoir départager deux requêtes étant en concurrence pour la réservation d'un même ensemble de noeuds. De cette façon, si deux (ou plus) requêtes concurrentes sont propagées au sein du magasin de noeuds, celle qui est prioritaire, disons R_P, est autorisée à voler des noeuds potentiellement déjà recrutés par d'autres nuages, tant que ces mêmes nuages n'ont pas terminé leur processus de construction. Un tel vol de ressource permet à Sarcasm de satisfaire au moins une des requêtes d'allocation, en cas de problème de concurrence. Par exemple, dans le scénario illustré précédemment par la Figure 5.6, la requête prioritaire R_P parvient à compléter le nuage demandé, en volant un noeud préalablement recruté par la requête non prioritaire R_{NP}. Plus généralement, dans un scénario où deux requêtes concurrentes peuvent être théoriquement satisfaites (c.-à-d. qu'il existe suffisamment de noeuds libres pour satisfaire les deux requêtes), le vol de noeuds a pour effet d'accélérer la complétion du nuage C_P associé à la requête prioritaire R_P, au détriment du nuage C_{NP}. Néanmoins, dans ce scénario, les deux requêtes d'allocation sont satisfaites avec une grande probabilité. Dans un scénario où une seule des deux requêtes peut être satisfaite, le nuage prioritaire C_P est créé, tandis que la requête non prioritaire R_{NP} est finalement annulée. Bien entendu, le scénario de requête d'allocation considéré précédemment est généralisable pour un nombre important de requêtes concurrentes. Le paragraphe suivant décrit l'adaptation du protocole de construction de nuage, tel qu'il est présenté dans la première partie de cette section, au vol de noeuds par priorité.

Adaptation de Sarcasm au vol de noeuds par priorité : Afin de pouvoir implanter le vol de noeuds par priorité au sein du protocole de construction de nuage pair-à-pair, un nouvel état de disponibilité des pairs est utilisé, il s'agit de l'état « en attente ». Cet état est attribué à un pair qui est membre d'un nuage pair-à-pair en construction, c'est à dire d'un nuage qui n'est pas encore complet. Les pairs dits « en attente » peuvent donc être volés par un nuage prioritaire, contrairement aux pairs alloués. Comme décrit précédemment dans la première version du protocole, lorsqu'un nuage est complet, un message M_{Start} indiquant les dates de début et de fin de déploiement du nuage à construire est diffusé à l'ensemble des noeuds recrutés. Lorsque des pairs possédant le statut « en attente » reçoivent un tel message, ils changent leur statut pour la valeur « alloué », de façon à interdire le vol de noeuds pour les nuages pair-à-pair en cours de déploiement. A noter que des noeuds possédant le statut « en attente » peuvent être volés pendant la propagation du message M_{Start}. Aussi, si un nuage ne possède plus suffisamment de noeuds pour satisfaire sa spécification au début de sa période

de déploiement, conformément à la routine de surveillance décrite dans la section suivante, ce nuage émettra des requêtes de secours afin de recruter des pairs de l'entrepôt de noeuds pour compléter sa taille.

Le pseudo-code du protocole de création de nuage de Sarcasm adapté au vol de noeuds par priorité est donné dans la Figure 5.7. Ce protocole décrit les modifications apportées aux

CONCURRENCY AWARE CLOUD CONSTRUCTION PROTOCOL

```
1: upon receive(M_i :< P_i, τ_i, S_i, N_i, E >) do
2:   if M_i ∉ receivedMessage then
3:     receivedMessage ← receivedMessage ∪ M_i
4:     if status = free then
5:       joinCloud(N_i, C_i)
6:     else if status = pending then
7:       if C_i ≺ myCloud then
8:         OV-MAN.leave(myCloud)
9:         joinCloud(N_i, C_i)
10:      end if
11:    end if
12:    forward(M_i)
13:  end if
14: end upon

15: procedure joinCloud(N_i, C_i)
16:   OV-MAN.join(C_i, N_i)
17:   myCloud ← C_i
18:   status ← pending
19: end

20: upon receive(< START, τ_start, τ_end >) do
21:   status ← booked
22:   for every Δ_Alloc do
23:     τ_j ← OV-MAN.getSyncTimeOfDay()
24:     if τ_j > τ_end then
25:       release()
26:     end if
27:   end for
28: end upon
```

FIGURE 5.7 – Pseudo code d'un protocole de diffusion robuste à la concurrence.

deux traitants de réception de messages : `receive`$(M_i :< P_i, \tau_i, S_i, \mathcal{N}_i, E >)$ et `receive`$(< START, \tau_{start}, \tau_{end} >)$. Aussi, dans cette nouvelle version du protocole de construction de nuage pair-à-pair, lorsqu'un pair P_R du magasin de noeuds reçoit un message M_i de requête d'allocation, il ne traite ce message que s'il le reçoit pour la première fois (lignes 2 et 3). Si ce pair est un pair libre, alors il joint le nuage C_i en construction (lignes 4 à 5), change son statut de « libre » à « en attente », puis sauvegarde l'identifiant du nuage C_i dans la variable locale *myCloud* (lignes 15 à 19). Si P_R possède le statut « en attente », alors il ne joint le nuage C_i que si celui si possède une priorité supérieure à son nuage actuel, c.-à-d. celui identifié par la variable *myCloud* (lignes 7 et 8). Si C'est le cas, le pair P_R quitte son ancien nuage (*myCloud*) avant de joindre le nouveau (C_i) (lignes 6 à 11). Ensuite, conformément au protocole de construction de nuage décrit au sein de la Figure 5.4, le message M_i est réexpédié à l'aide de la méthode `forward(`M_i`)`, qui vérifie que le nuage à construire est incomplet avant de continuer la diffusion de son message de requête. Finalement, lorsqu'un pair reçoit la notification de début de déploiement du nuage auquel il est attribué, il change son statut de « en attente » à « alloué » (ligne 21), et garde ce dernier statut jusqu'à la fin de la durée de déploiement de son nuage. Une routine, exécutée selon la période Δ_{Alloc}, est alors démarrée afin de détecter la fin du déploiement du nuage (lignes 22 à 27). Lorsque la fin du déploiement est détectée par un pair grâce à cette routine, alors le pair en question retourne au sein de l'entrepôt de noeuds.

Ce nouveau protocole de construction de nuage pair-à-pair assure, qu'avec une attribution des priorités de requêtes cohérente, qu'au moins un nuage sera construit avec succès en présence de concurrence. Des évaluations de la performance de Sarcasm en fonction du degré de concurrence des requêtes d'allocation sont présentées dans le Chapitre 7.

Attribution des priorités : L'ordre utilisé par Sarcasm pour établir les priorités des requêtes d'allocation utilise trois critères : l'identifiant du pair ayant émis la requête, la période à laquelle cette requête a été émise et un critère générique. De façon intuitive, le critère

générique est utilisé pour rendre la politique d'attribution de priorité paramétrable. La période d'émission des requêtes est utilisée pour assurer que les requêtes d'allocations aboutissent à la création d'au moins un nuage pair-à-pair, en un temps raisonnablement court. Les identifiants des pairs, eux, sont utilisés pour assurer que l'ordre définissant les priorités est total, c'est-à-dire que tout couple de priorités de requête peut être comparé en utilisant cet ordre.

Plus précisément, l'ordre établissant les priorités des requêtes suit la règle suivante pour comparer deux requêtes R_i et R_j : si une des deux requêtes a été initiée avant l'autre (Cf. concept d'époque de diffusion, défini dans le paragraphe suivant), alors elle est systématiquement prioritaire. Si les requêtes R_i et R_j ont été initiées durant la même période, alors le critère générique est utilisé pour départager ces deux requêtes. Finalement, si la période de diffusion des requêtes R_i et R_j est la même et que le critère générique est incapable de départager ces requêtes, alors on utilise les identifiants des pairs ayant initié les diffusions de R_i et de R_j pour établir la comparaison.

Le critère de priorité portant sur les dates d'émission des requêtes utilise le principe d'époque de diffusion. Une époque de diffusion est une période de durée Δ_{Epoch} qui est définie par rapport à une référence horaire commune à tous les pairs du système. Par exemple, on peut considérer qu'une nouvelle époque de diffusion démarre toutes les deux minutes, en prenant pour référence le début de l'heure en cours. De cette façon, si une époque E démarre à un instant t, alors tout instant t' tel que $t \leq t' < t' + \Delta_{Epoch}$ appartient à cette époque. Une requête d'allocation est donc strictement prioritaire si elle est estampillée par une époque de diffusion antérieure aux époques associées aux autres requêtes. Pour résumer ce mécanisme de comparaison de priorités, le pseudo-code de la fonction `compare`(R_i, R_j), qui compare les priorités de deux requêtes R_i et R_j, est donné sur la Figure 5.8. A noter que ce pseudo-code suppose que les deux objets R_i et R_j, représentant les deux requêtes à comparer, possèdent chacun un attribut *epoch* correspondant à l'époque de diffusion de la requête en question, un attribut *gen* correspondant à la valeur du critère générique associé, et un attribut *id* correspondant à l'identifiant du pair ayant émis cette requête. L'algorithme de comparaison

PRIORITY COMPARISON ALGORITHM

```
 1: function compare(R_i, R_j)
 2:    if R_i.epoch = R_j.epoch then
 3:       if R_i.gen = R_j.gen then
 4:          return  R_i.id ≺ R_j.id
 5:       else
 6:          return  R_i.gen ≺ R_j.gen
 7:       end if
 8:    else
 9:       return  R_i.epoch ≺ R_j.epoch
10:    end if
11: end
```

FIGURE 5.8 – Pseudo-code de l'algorithme de comparaison de priorités de requêtes.

de priorités représenté sur la Figure 5.8 utilise, dans un premier temps, les époques de diffusion pour comparer deux requêtes d'allocation de nuage (ligne 2 et ligne 9). Si ces époques sont différentes, alors c'est systématiquement la requête possédant l'estampille la plus ancienne qui sera prioritaire. Si au contraire les deux époques sont égales, alors l'algorithme utilise le critère générique pour comparer ces deux requêtes (ligne 3 et ligne 6). Finalement, si ces deux requêtes sont également équivalentes du point de vue du critère d'équité, alors on utilise les identifiants

des pairs ayant émis ces requêtes pour départager les deux priorités (ligne 4). Ce mécanisme de comparaison en trois étapes assure que chaque membre de Salute est capable de comparer tous les couples possibles de requêtes d'allocation concurrentes. Il permet également de garantir que si le nombre de requêtes émises durant une même époque de diffusion est borné, alors une de ces requêtes est strictement prioritaire sur toutes les requêtes potentiellement générées pendant et après cette époque. Cette dernière garantie permet d'assurer que Salute est capable de construire au moins un nuage pair-à-pair en temps borné, malgré la présence d'un degré de concurrence des requêtes potentiellement élevé. De plus, ce mécanisme permet d'intégrer des critères paramétrables au sein des priorités associées aux requêtes d'allocation. Ces critères sont systématiquement respectés pour la comparaison de deux requêtes émises lors de la même époque de diffusion.

Pour conclure, l'utilisation du protocole adapté à la concurrence permet à Sarcasm d'assurer, avec une grande probabilité, qu'en cas de concurrence des requêtes au moins un nuage pair-à-pair, à priori celui possédant la priorité la plus élevée, est construit avec succès. Une discussion sur l'évaluation du protocole de construction de nuages pair-à-pair et sur la façon d'utiliser le critère générique d'attribution des priorités, pour préserver l'équité du service d'allocation de noeuds, est présentée dans le Chapitre 7.

5.2.3 Maintenance des nuages pair-à-pair

Une fois qu'un nuage pair-à-pair est construit avec succès et que sa période de déploiement commence, le protocole de gestion de nuage Sarcasm doit maintenir la spécification de ce nuage pendant toute sa durée de déploiement. Ce protocole de maintenance peut être divisé en deux tâches : la surveillance de la taille du nuage et la création de flux d'approvisionnement pour secourir ce nuage, si sa taille vient à baisser en raison de la dynamique de l'environnement hôte. Aussi ces deux tâches doivent être réalisées de façon totalement décentralisée par les pairs alloués au sein du nuage en question. La Figure 5.9 donne le pseudo-code du protocole exécuté par les pairs d'un nuage C_i pour assurer la maintenance de sa taille.

SARCASM CLOUD MAINTENANCE PROTOCOL

```
1: task monitorCloudSize(< C_i, S_i >)
2:   for every Δ_Count seconds do
3:     S'_i ← OV-MAN.getSize(C_i)
4:     if S'_i < S_i then
5:       N_i ← OV-MAN.getRandomPeer(O_i)
6:       M_r ← < RESCUE, C_i, S_i, N_i, τ_end >
7:       lightRequest(Warehouse, M_r, S_i, R)
8:     else if S'_i > S_i + offset then
9:       unload(S_i, S'_i)
10:    end if
11:  end for
12: end

13: procedure unload(S_i, S'_i)
14:   nbToRelease ← S'_i − S_i
15:   rdnValue ← getRndValue(0,S'_i)
16:   if rdnValue ≤ nbToRelease then
17:     release()
18:   end if
19: end

20: procedure lightRequest(O_i, M_r, S_i, R)
21:   nbToBroadcast ← R x S_i
22:   rdnValue ← getRndValue(0,S_i)
23:   if rdnValue ≤ nbToBroadcast then
24:     propagate(O_i, M_r)
25:   end if
26: end
```

FIGURE 5.9 – Pseudo code du protocole de maintenance des nuages de noeuds.

Chaque pair membre d'un nuage C_i exécute la routine `monitorCloudSize`$(< C_i, S_i >)$, où S_i est la taille de référence de ce nuage. Cette routine est exécutée périodiquement selon la valeur Δ_{Count}, qui représente le temps nécessaire pour que le service de comptage de pairs puisse détecter un changement de taille du nuage C_i, de façon fiable. Cette routine observe la

taille actuelle S'_i du nuage C_i et vérifie si cette taille correspond bien à la taille de référence S_i (lignes 3 et 4). Si c'est le cas, le protocole de maintenance n'a rien à faire et l'exécution de la routine se suspend pendant une période Δ_{Count}. En revanche, si la taille S'_i est inférieure à la taille de référence, alors les pairs « alloués » préparent un message de requête de secours M_r, en vue de recruter des pairs supplémentaires pour rétablir la taille spécifiée. Ce message contient l'identifiant du nuage C_i, sa taille de référence S_i, un ensemble de noeuds de contact $\mathcal{N}_i$ à utiliser pour rejoindre ce nuage, puis la date τ_{end} correspondant à la fin du déploiement du nuage C_i (lignes 5 et 6). Le message M_r est diffusé au sein de l'entrepôt de noeuds en utilisant la méthode `lightRequest`$(O_i, M_r, S_i, \mathcal{R})$, qui a pour but de limiter le nombre de requêtes de secours propagées simultanément. Cette méthode fait en sorte que seul un pourcentage fixe $\mathcal{R}$ des membres du nuage diffuse le message M_r (lignes 20 à 26). La valeur par défaut du pourcentage $\mathcal{R}$ est fixée à 5%, ce qui permet d'assurer qu'une requête de secours sera propagée avec une grande probabilité, tout en réduisant le nombre de messages générés par la surveillance décentralisée du nuage C_i. Finalement, si la routine de surveillance détecte que la taille du nuage C_i est trop grande (ligne 8), elle exécute le même processus de déchargement que celui utilisé lors du démarrage du déploiement d'un nuage (Cf. Figure 5.5). Néanmoins, ce processus de déchargement n'a pas besoin qu'un message soit diffusé, il est simplement exécuté de façon décentralisée par tous les membres du nuage (lignes 13 à 19).

La création du flux d'approvisionnement de pairs réalisée par la diffusion d'un message de secours M_r est très proche de celle réalisée lors de la création d'un nuage (Cf. Figure 5.4). La différence entre ces deux flux d'approvisionnement est que lorsqu'il s'agit d'un message de secours, les pairs de réserve peuvent également joindre le nuage à secourir.

5.2.4 Filtrage des ressources composant les nuages

Lors d'une demande d'allocation de nuage pair-à-pair, le protocole Sarcasm permet de spécifier un ensemble de contraintes sur les attributs des noeuds à allouer. Effectivement, en plus de décrire sa taille et sa durée de déploiement, la spécification d'un nuage contient un ensemble de filtres, potentiellement vide, qui définit un type de noeud. Le nuage à construire contiendra uniquement des noeuds satisfaisant ce type. De façon plus précise, un filtre spécifie une contrainte sur un des attributs des noeuds à allouer, tels que leur capacité de calcul ou de stockage minimum. Un type de noeud contient, au maximum, un filtre par attribut. Le type ne contenant aucun filtre désigne tous les noeuds potentiellement disponibles au sein de la plateforme Salute. Un exemple de liste d'attributs à considérer peut être : la capacité de calcul, la capacité de stockage ou encore la bande passante des noeuds à allouer.

De façon à implanter ce filtrage de pairs au sein du protocole de construction de nuage établi par Sarcasm, l'ensemble de filtres définissant le type de noeuds à allouer est ajouté aux message de requêtes d'allocation. Lorsqu'un pair libre reçoit un tel message, il ne joint le nuage associé que s'il correspond au type de pair spécifié par cette liste de filtres. Le même principe est appliqué pour les messages de secours, de façon à maintenir la spécification du nuage en question. Cette propriété de filtrage des ressources devient plus complexe lorsque l'on considère des requêtes d'allocation demandant plusieurs types de pairs. Effectivement, la construction d'un nuage pair-à-pair contenant plusieurs types de noeuds nécessite l'utilisation d'un service de comptage de pairs qui est capable de compter le nombre de différents types de pairs au sein du même réseau virtuel. Une proposition d'implantation de ce service est décrite dans le paragraphe suivant :

Service de comptage de pairs typés : Dans Salute, les pairs sont capables de compter le nombre de noeuds correspondant à un type T donné, au sein des différents réseaux vir-

tuels composant les nuages pair-à-pair hébergés par la plateforme pair-à-pair. Pour ce faire, ils utilisent la méthode `getNumber`$(\mathcal{T}, O_i, \mathcal{N}_i)$ capable de compter le nombre de noeuds correspondant au type $\mathcal{T}$ au sein du réseau virtuel O_i. Cette méthode utilise une technique particulière qui consiste à créer un réseau virtuel supplémentaire pour chaque type de noeud susceptible d'être compté (le nombre de types est connu d'après la spécification du nuage), puis exécute le service pair-à-pair de comptage sur ce dernier réseau. Lorsque le réseau virtuel hébergeant le nuage est créé (Cf. Section 5.2.1), le protocole de construction de réseau superposé est également exécuté. Ce dernier protocole est configuré de sorte qu'il existe un réseau virtuel superposé pour chaque type T de pairs à héberger au sein du nuage demandé. D'après les évaluations du protocole Vicinity [154, 152, 155], qui implante le service de construction de réseau superposé de Salute, la maintenance de l'architecture d'un tel réseau virtuel multicouche est garantie malgré la dynamique du système, tant que le nombre de réseaux superposés ne dépasse pas une centaine. En conséquence, le nombre de types de pairs différents que peut contenir un nuage pair-à-pair alloué par la plateforme Salute peut être de l'ordre d'une centaine d'unités.

Une méthode simple pour satisfaire de telles demandes d'allocation, en utilisant le service de comptage décrit ci-dessus, serait de construire un nuage pour chaque type de pair, puis de fusionner ces nuages. Malheureusement, cette solution poserait de sérieux problèmes de synchronisation lorsque le nuage correspondant à un des types de noeud serait complet, alors que d'autres nuages répondant à la même demande d'allocation seraient encore en attente de construction. Effectivement, en considérant les problèmes liés à la concurrence des requêtes et aux priorités associées, cette synchronisation des sous-nuages deviendrait vite difficile. La méthode alors proposée dans cette thèse est de diffuser un message de requête d'allocation M_i^p pour chaque type p de noeuds requis au sein de la spécification du nuage demandé. Ces messages contiennent le même identifiant de nuage et le même ensemble initial de pairs de contacts, de façon à collecter tous les pairs recrutés par ces différents nuages au sein du même réseau virtuel. La diffusion d'un message de requête M_i^p est alors stoppée lorsque le nuage contient suffisamment de noeuds du type p. Finalement, le déploiement du nuage commence lorsqu'il contient suffisamment de pairs pour chaque type contenu dans sa spécification. A noter que cette technique est réalisable grâce à la présence d'un service de comptage de pairs qui est capable de compter plusieurs types de pairs au sein du même réseau virtuel. En utilisant ce service de comptage, le protocole de surveillance et de secours des nuages pair-à-pair n'est presque pas modifié. En effet, les requêtes de secours sont déclenchées indépendamment pour chaque type de pairs manquant au sein du nuage en cours de déploiement. Finalement, il est important de noter que le filtrage des attributs des ressources proposé par la plateforme Salute est peu précis. En effet, la vocation de ce filtrage n'est pas de proposer un service d'allocation de nuages dont la composition peut être choisie « à la carte », mais plutôt d'effectuer un tri grossier sur les ressources qui seraient potentiellement inutilisables par une application cliente.

Conclusion : Ce chapitre présente le protocole Sarcasm qui est responsable de la construction et de la maintenance des nuages pair-à-pair au sein de la plateforme Salute. Ce protocole constitue la première solution totalement décentralisée qui permet de faire cohabiter plusieurs collections de noeuds, dont la durée de déploiement et la taille sont configurables, au sein d'un environnement pair-à-pair. La conception de cette solution a soulevé des problèmes de gestion décentralisée de la concurrence des requêtes d'allocation. La proposition faite dans cette thèse pour pallier à ce problème de concurrence consiste à associer systématiquement des priorités aux requêtes émises par les clients du service d'allocation de noeuds afin de procéder à un éventuel vol de noeuds par priorité, si besoin est. Les évaluations présentées dans le Chapitre 7

montrent que le vol de noeuds est nécessaire afin de préserver la fiabilité du service d'allocation de nuages de la plateforme Salute. Ces évaluations montrent également qu'il est possible de définir différentes politiques d'attribution de priorités permettant d'assurer l'équité du service d'allocation selon la métrique voulue.

Chapitre 6

Contrôle d'admission des requêtes d'allocation

Sommaire

La problématique traitée par les contrôleurs d'admission est d'accepter et d'ordonnancer les allocations de nuages pair-à-pair en fonction de l'évolution de l'environnement hôte, et ce, de façon totalement décentralisée. En effet, dans les scénarios de déploiement de la plateforme Salute considérés dans cette thèse, et contrairement à de nombreux services de l'informatique dans les nuages, on suppose que le nombre de ressources disponibles n'est pas extensible à l'infini. Dans ce cas, il est impératif de confronter la faisabilité des allocations demandées à la disponibilité des ressources, en appliquant un contrôle d'admission, de façon à décider si une requête donnée doit être satisfaite ou non.

L'étude présentée dans ce chapitre consiste à comparer les différentes stratégies et politiques d'allocation pouvant être utilisées pour accepter les requêtes soumises à la plateforme Salute. Une des solutions proposées est d'étudier la disponibilité des ressources afin d'établir des prédictions qui pourront être utilisées pour décider si une requête d'allocation peut être satisfaite de façon sûre ou non. On distingue alors deux techniques de prédiction de disponibilité : la première technique est dite locale, car elle s'intéresse à la disponibilité des pairs de façon individuelle. La seconde technique est dite globale, elle consiste à observer et à prédire l'évolution de la taille du système. Finalement, l'évaluation de plusieurs contrôleurs d'admission, dans des conditions de déploiement différentes, montre qu'il est utile de considérer plusieurs stratégies d'allocation. Par exemple, notre étude nous a conduits à valider le fait que l'utilisation d'une politique d'admission triviale, consistant à accepter toutes les requêtes d'allocation, pouvait être la solution optimale lorsque la durée de déploiement des nuages demandés était courte. A l'inverse, les techniques utilisant une prédiction locale, que l'ont peut trouver dans la littérature, semblent s'adapter difficilement au contexte de l'allocation de nuages sur des environnements pair-à-pair.

La première section de chapitre détaille la problématique posée par l'implantation d'un

contrôle d'admission totalement décentralisé, au sein de la plateforme Salute. La seconde section présente les trois familles de contrôleurs d'admission considérées dans cette thèse, à savoir : les contrôleurs optimistes, les contrôleurs locaux et les contrôleurs globaux.

6.1 Problématique de l'admission des requêtes

Soit une requête d'allocation R, associée à un nuage pair-à-pair C_R de taille S_R et de durée de déploiement L_R. Si cette requête est soumise à la plateforme Salute au temps t_R, alors le rôle d'un contrôleur d'admission est d'évaluer si le système contiendra suffisamment de pairs pour maintenir la spécification du nuage C_R entre les temps t_R et $t_R + L_R$. Pour comprendre la problématique rencontrée par les contrôleurs d'admission, il faut faire l'hypothèse que ceux-ci ont accès à deux fonctions $\mathcal{T}(t)$ et $\mathcal{A}(t)$, qui représentent respectivement l'évolution de la taille du système et l'évolution des allocations de nuages pair-à-pair en cours, pour un temps futur t. A noter que la fonction $\mathcal{A}(t)$ est strictement décroissante, car elle représente l'évolution du nombre de noeuds alloués en ne tenant compte que des nuages alloués avant le temps t_R. Intuitivement, cette fonction représente le nombre de noeuds alloués au temps t, en ne considérant que les allocations de nuages acceptées avant le temps t_R. Considérons maintenant une discrétisation du temps à l'échelle de la seconde, telle que $t \in \mathbb{N}$, et telle que t_0 représente l'état initial du système. La règle établissant que la requête R peut être acceptée de façon sûre est la suivante :

$$\forall t \in [t_R, t_R + 1, .., t_R + L_R] : \mathcal{A}(t) + S_R \leqslant \mathcal{T}(t) \qquad (1)$$

Pour simplifier cette règle, les contrôleurs d'admission utilisent une troisième fonction $\mathcal{O}(t)$, telle que $\mathcal{O}(t) = \mathcal{A}(t) + S_R$. Cette fonction décrit l'évolution du nombre de noeuds alloués entre le temps t_R et le temps $t_R + L_R$, dans le cas où la requête R est acceptée. La Figure 6.1 illustre deux scénarios au sein desquels cette règle est utilisée pour décider si la requête R doit être acceptée ou non. Dans le scénario décrit sur la partie gauche de cette figure, l'application de la règle *(1)* entraîne le rejet de la requête R. En effet, il existe un temps

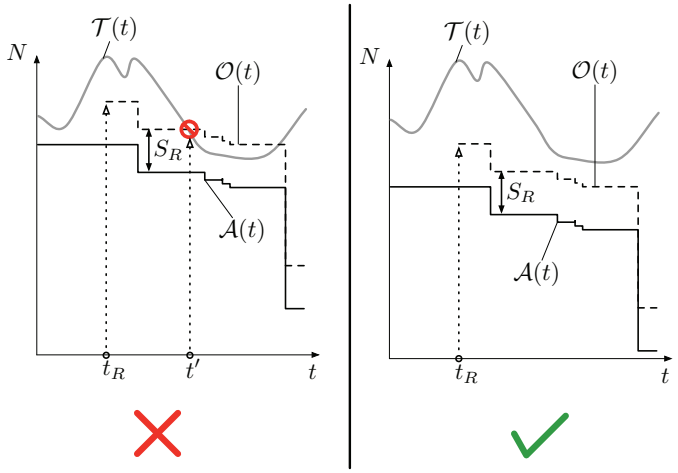


FIGURE 6.1 – Règle appliquée par les contrôleurs d'admission pour accepter une requête R.

t' pour lequel $\mathcal{O}(t') > \mathcal{T}(t')$, c.-à-d. pour lequel la taille globale du système est inférieure

à la somme des noeuds alloués au sein des divers nuages pair-à-pair. L'interprétation des contrôleurs d'admission est que le nombre de noeuds présents dans le système sera insuffisant après ce temps t' pour maintenir les spécifications de tous les nuages déployés, si la requête R est acceptée. Dans le scénario décrit sur la partie gauche de la Figure 6.1, le contrôleur n'est pas capable de trouver un temps t' pour lequel le nombre de noeuds alloués soit supérieur à la taille du système. En conséquence, le contrôleur d'admission estime qu'il existera suffisamment de noeuds au sein de Salute pour nourrir tous les nuages pair-à-pair jusqu'à la fin de leur déploiement respectif, y compris si la requête R est acceptée.

Bien que la règle *(1)* décrive parfaitement le comportement que doit avoir un contrôleur d'admission parfait, elle doit être adaptée au contexte des environnements pair-à-pair, au sein desquels aucune estimation exacte de la fonction théorique $\mathcal{T}(t)$ n'existe. En effet, l'hétérogénéité et le grand nombre des ressources composant ce type d'environnement rendent difficile la réalisation d'un service capable de prédire parfaitement et exhaustivement la disponibilité des pairs du système. L'approche alors choisie par Salute consiste à utiliser des techniques de prédiction de disponibilités des noeuds, dans le but de réaliser une approximation de la fonction $\mathcal{T}(t)$. Le rôle des contrôleurs d'admission est alors d'utiliser ces prédictions approximatives, combinées à des politiques d'allocations de nuages, afin de contrôler le déclenchement des requêtes d'allocations au sein de la plateforme Salute. La section suivante présente les techniques de prédiction de disponibilité, étudiée dans cette thèse, ainsi que les politiques d'allocation qui en découlent.

La prédiction de disponibilité des ressources au sein des environnements pair-à-pair est une problématique difficile, qui donne lieux à plusieurs catégories de solutions bien distinctes. En effet, la disponibilité des noeuds composant un environnement pair-à-pair, tel qu'une fédération de nuages, peut être prédite de façon locale, c'est-à-dire en observant la disponibilité des pairs un par un, ou bien de façon globale, c'est à dire en observant l'évolution de la taille du système. Aussi, ces différentes techniques de prédiction donnent lieux à différentes politiques d'allocation de nuages. Les politiques que nous avons étudiées dans cette thèse sont présentées dans la suite du chapitre.

6.2 Politique d'allocation optimiste

Un politique d'allocation optimiste consiste à accepter l'exécution d'une requête, si le nombre de noeuds libres disponible au moment où la requête est soumise est suffisamment important pour la satisfaire. Cette stratégie est dite « optimiste », car elle ne tient compte d'aucune prédiction quant à l'évolution de la taille du système. Le contrôleur d'admission associé fonctionne donc comme s'il répondait systématiquement « oui » à la question : « Y aura t-il suffisamment de noeuds au sein du système pour maintenir tous les nuages pair-à-pair au cours du temps ? ». La règle associée à cette politique d'allocation est la suivante :

$$\mathcal{A}(t_R) + S_R \leqslant \mathcal{T}(t_R) \qquad (2)$$

Cette règle correspond à la stratégie adoptée par défaut au sein de Salute, et elle est particulièrement simple à implanter au sein de la plateforme pair-à-pair : lorsqu'un pair est sur le point de diffuser une requête d'allocation de noeuds, il vérifie que le nombre de pairs libres contenus par le magasin de noeuds est supérieur à la taille du nuage à construire. Dans l'affirmative, la requête est acceptée par « le contrôleur d'admission optimiste », dans le cas contraire, elle est rejetée et la requête d'allocation n'est pas diffusée. Bien que cette stratégie

paraisse triviale, il s'avère qu'elle est optimale lorsque la plateforme Salute est déployée sur un environnement très stable, comme c'est par exemple le cas pour le système PlanetLab [36, 142] (Cf. Section 7.4).

Effectivement, si le nombre de noeuds présents au sein du système ne varie pas, alors la prédiction de disponibilité devient inutile. Cependant, conformément à ce qu'a montré l'étude de Douceur [58], il existe de nombreux environnements pair-à-pair au sein desquels la disponibilité d'une grande partie des noeuds est conditionnée par des cycles quotidiens et hebdomadaires. C'est le cas, par exemple, pour l'environnement des pairs du système de communication Skype [35, 73, 68] et pour ceux du système composé de stations de travail étudié par Bolosky et coll. [43, 68], que l'on appellera système « de Microsoft » dans notre étude. Ces deux traces, analysées plus en détail au sein du Chapitre 7, sont données sur la Figure 7.1. La sous-figure de gauche représente l'évolution de la taille du système de Microsoft

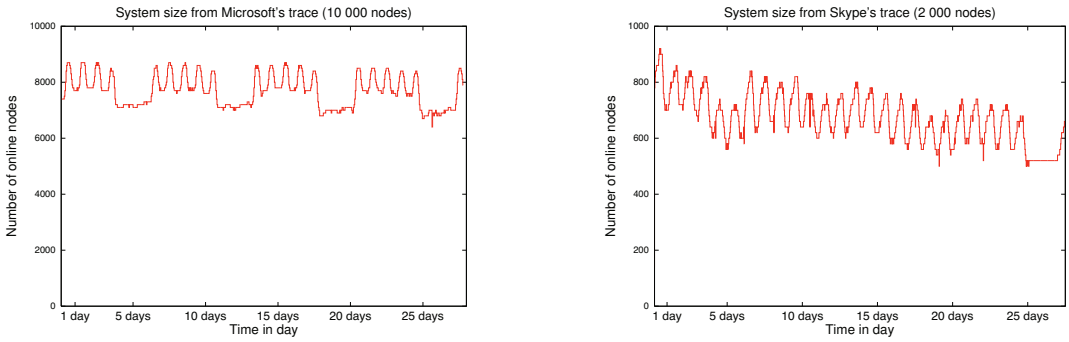


FIGURE 6.2 – Exemples de variation de la disponibilité des membres d'un système pair-à-pair.

et la sous-figure de droite représente l'évolution de la taille du système Skype. Aussi, l'influence des cycles quotidiens et hebdomadaires est nettement visible sur ces deux traces : la taille du système de Microsoft varie quotidiennement selon une amplitude maximale de 1500 noeuds, soit 15% de la taille du système, pendant les jours ouvrés. Puis, cette taille est provisoirement stabilisée pendant les week-ends. Dans la trace représentant l'évolution du système Skype, la variation hebdomadaire est moins nette et l'amplitude maximale de variation quotidienne correspond à 10% de la taille totale du système.

En considérant de telles variations, il existe des scénarios pour lesquels une politique d'allocation de nuages pair-à-pair optimiste n'est pas adaptée. En effet, si un grand nombre de requêtes sont soumises à Salute lorsque l'environnement pair-à-pair atteint sa taille maximale quotidienne, le système aura tendance à allouer plus de nuages que ce qu'il pourra maintenir au cours du temps, et ce, parce qu'aucune prédiction de disponibilité des noeuds n'est utilisée. Par exemple, si le système alloue 95% des ressources disponibles et qu'il connaît une baisse soudaine du nombre de pairs qui le compose, de plus de 15%, alors certains nuages pair-à-pair ne pourront plus être maintenus. Ce problème est appelé la « surallocation ». Le résultat d'une série d'évaluation ayant pour but d'illustrer ce phénomène est présenté au sein du Chapitre 7. Aussi, afin de prévenir ce problème de surallocation, Salute propose deux types de contrôleur d'admission utilisant des techniques de prédiction de disponibilité : les contrôleurs d'admission locaux et les contrôleurs d'admission globaux.

6.3 Contrôleurs locaux

Le premier type de contrôleur d'admission non trivial utilisé dans Salute est le contrôleur local. Ce type de contrôleur d'admission utilise des techniques de prédiction de disponibilité qui sont basées sur l'étude de l'historique des pairs, comme celles proposées par Mickens et coll. [117] ou encore celles proposées par Bhagwan et coll. [41, 28, 82]. Ces techniques de prédiction appliquent une analyse statistique qui est basée sur des lois probabilistes, reconnues comme étant des modèles pertinents du comportement des noeuds dans un environnement pair-à-pair [41, 28, 82], ou encore sur l'utilisation d'heuristiques empiriques validées par l'expérimentation [117]. On parle alors de contrôleurs locaux, car les prédictions sont réalisées localement pour chaque pair, à partir de son propre historique de disponibilité. Une des limitations importantes de ces contrôleurs d'admission réside dans le fait qu'ils ont besoin d'une quantité importante de données pour nourrir l'analyse statistique qui est à la base de leurs prédictions. Par exemple, les heuristiques de prédiction proposées par Mickens et coll. sont entraînées avec un historique contenant des données récoltées sur une période de quinze jours, et ce, pour prévoir la disponibilité des pairs sur une période s'étalant d'une heure à une semaine [117].

Dans le service d'allocation proposé par Salute, les contrôleurs d'admission locaux sont utilisés par les pairs libres, au moment où ceux-ci reçoivent une requête d'allocation de nuage. Ces mêmes pairs décident alors de joindre un nuage en construction uniquement s'ils estiment être disponibles durant toute la durée de déploiement du nuage en question. Cela suppose que chaque pair membre de Salute exécute localement un contrôleur d'admission, chargé d'enregistrer son historique de disponibilité, et chargé d'appliquer les techniques de prédiction locale au moment de satisfaire une requête d'allocation. A noter que si les contrôleurs d'admission locaux ont accès à des méthodes de prédiction parfaites, alors Salute peut définir une politique d'allocation de noeuds selon laquelle tout nuage construit avec succès aura une taille strictement stable durant l'intégralité de son déploiement. Pour ce faire, il suffit qu'une requête d'allocation n'accepte que des pairs strictement sûrs d'être disponibles durant toute la durée de déploiement du nuage demandé. Dans ce cas, les pairs alloués pour le déploiement d'un nuage ne quitteront pas le système avant la fin de la période d'allocation spécifiée, et les mécanismes d'approvisionnement continu du protocole de maintenance Saracasm seront inutiles.

Malheureusement, comme l'ont montré les travaux de Mickens et coll. [117], la précision des prédictions de disponibilité concernant les noeuds d'un environnement pair-à-pair diminue fortement au fur et à mesure que la période de prédiction est allongée. Par exemple, ces études montrent qu'il est très difficile de prévoir la disponibilité des noeuds de façon fiable, sur un système pair-à-pair relativement stable comme PlanetLab [36, 142], pour des durées de déploiement supérieures à quelques heures [117]. De plus, seule une partie des noeuds de l'environnement pair-à-pair présente un modèle de disponibilité compatible avec les heuristiques de prédictions (50% pour le système de Microsoft et 10% pour PlanetLab [117]). En conséquence, si Salute utilise uniquement des contrôleurs d'admission locaux pour conduire l'allocation des nuages pair-à-pair, seule une partie des noeuds du système pourra être allouée. Aussi, afin d'optimiser cette quantité maximale de ressources allouables, il est intéressant de considérer d'autres types de contrôleur d'admission. A noter que l'évaluation d'une politique d'allocation basée sur l'utilisation de contrôleurs d'admission locaux parfaits est étudiée au sein du Chapitre 7. Cette politique d'allocation est comparée aux autres politiques présentées dans ce chapitre, en particulier les politiques d'allocation utilisant des prédictions globales.

6.4 Contrôleurs globaux

Contrairement aux contrôleurs d'admission locaux, les contrôleurs qui suivent une stratégie de prédiction globale ne s'intéressent pas directement à la disponibilité des pairs, mais se focalisent plutôt sur l'évolution de la taille du système au cours du temps. L'intérêt de ce type de contrôleur d'admission est double : premièrement, il s'attaque directement au problème de la surallocation en focalisant ses prédictions de disponibilité sur la taille du système. Ensuite, ce type de contrôleur peut être utilisé indépendamment des modèles de disponibilité des pairs, c'est-à-dire que, contrairement aux contrôleurs d'admission locaux, il prend en compte tous les pairs du système. Dans cette thèse deux contrôleurs globaux, sont proposés : le contrôleur d'admission à minimum strict et le contrôleur d'admission à échantillonnage périodique du minimum.

Contrôleur d'admission à minimum stricte : Un contrôleur utilisant la politique du minimum strict n'acceptera de satisfaire une requête d'allocation de noeuds que si le nombre de noeuds alloués, en comptant la taille de la requête en question, ne dépasse pas la taille minimale du système. Cette taille minimale est calculée par rapport à l'historique de l'évolution du système. La règle alors appliquée par un contrôleur à minimum strict pour décider si une requête d'allocation doit être satisfaite ou non est la suivante :

$$\forall t \in [t_0, t_R] : \mathcal{O}(t) \leqslant \min \mathcal{T}(t) \quad (3)$$

La valeur du minimum est mise à jour au fur et à mesure que la taille du système évolue. L'idée de cette stratégie est de dire que si la taille du système est toujours supérieure à une valeur minimale Min, alors Salute peut allouer un nombre Min de noeuds de façon sûre. Cette stratégie est particulièrement efficace si la taille du système reste raisonnablement stable au cours du temps. En revanche, on peut distinguer deux scénarios pour lesquels un contrôleur d'admission à minimum strict peut présenter de sévères limitations :

1. Si la taille du système décroît subitement en dessous de la valeur Min connue par le contrôleur, alors il sera difficile de maintenir les nuages pair-à-pair déjà déployés. Néanmoins, cette limitation n'a que peu d'effet sur la fiabilité du service d'allocation de noeuds, car la plateforme Salute est naturellement capable de compenser une légère baisse de la taille du système à l'aide des pairs de réserve. Cette capacité de compensation est valable tant que la baisse de la taille du système ne correspond pas à un événement catastrophique.

2. Si la valeur minimale de la taille du système est largement inférieure à sa taille moyenne, alors le système n'allouera qu'une faible partie des ressources disponibles. Les performances du service d'allocation résultant seront alors faibles.

Le première de ces deux limitations est dépendante d'événements catastrophiques que nous considérons comme rares dans cette thèse. La gestion d'un tel phénomène n'est alors pas prise en compte par les contrôleurs d'admission de Salute. Cependant, on peut définir un nouveau type de contrôleur afin de pallier à la seconde limitation associée à la politique du minimum fixe. Il s'agit du contrôleur d'admission à échantillonnage périodique du minimum.

Contrôleur d'admission à échantillonnage périodique du minimum : L'idée du contrôleur d'admission à échantillonnage périodique du minimum est de détecter des cycles hebdomadaires ou quotidiens dans le profil d'évolution de la taille du système, afin d'améliorer la stratégie du minimum strict. Pour ce faire, ce dernier type de contrôleur d'admission construit un modèle $\mathcal{W}(t)$ représentant l'évolution du système par rapport à une période hebdomadaire. Ce modèle est calculé à partir d'une analyse statistique simple de l'historique de l'évolution du système. Ainsi, pour chaque moment de la semaine, le modèle hebdomadaire contient une série de valeurs statistiques mesurées par le passé, comme la taille moyenne du système à ce moment de la semaine et la variance associée. Si la variance des valeurs mesurées est faible (inférieure à 5%), alors le modèle hebdomadaire est considéré comme fiable et peut être utilisé par un contrôleur d'admission global en suivant la règle *(4)*, dans laquelle la fonction $\mathcal{W}(t)$ représente la taille moyenne du système sur une période hebdomadaire :

$$\forall t \in [t_R, ..., t_R + L_R] : \mathcal{O}(t) \leqslant \mathcal{W}(t) \quad (4)$$

Le modèle $\mathcal{W}(t)$ peut être construit simultanément par tous les pairs du système, la quantité de donnéed à stocker étant minime. On suppose alors que cette valeur est échangée par les pairs à l'aide d'un protocole épidémique d'agrégation de données, tel que le protocole de Jelasity et coll. [86], qui est utilisé par le service de comptage de pairs déployé sur la plateforme Salute. De cette façon, le modèle $\mathcal{W}(t)$ est construit et stocké de façon fiable par l'ensemble des pairs du système, et ce, malgré la dynamique de l'environnement hôte.

Conclusion

Dans ce chapitre, nous avons présenté une liste non exhaustive des types de contrôleur d'admission pouvant être utilisé au sein de la plateforme pair-à-pair Salute, pour ordonnancer les allocations de noeuds de façon fiable et totalement décentralisée. On distingue alors trois catégories de contrôleur : les contrôleurs optimistes, qui n'utilisent aucune prédiction de disponibilité pour accepter les requêtes soumises à Salute, les contrôleurs locaux qui utilisent des techniques de prédiction exécutées localement à chaque noeud et les contrôleurs globaux qui analysent et prédisent l'évolution du nombre de ressources disponibles au sein du système. L'évaluation de ces contrôleurs d'admission est présentée dans le Chapitre suivant.

Chapitre 7

Evaluation de Salute

Sommaire

Ce chapitre regroupe les évaluations de la plateforme Salute et du service d'allocation de noeuds qu'elle héberge, ainsi que les analyses qui en découlent. Tout d'abord, le protocole expérimental utilisé pour réaliser ces évaluations est présenté dans la Section 7.1. Puis la Section 7.2 illustre l'imp act du pourcentage de pairs de réserve de Salute sur des environnements dynamiques différents. La Section 7.3 présente les évaluations du protocole Sarcasm, utilisé pour construire et maintenir les nuages pair-à-pair au sein de la plateforme Salute. Finalement, la Section 7.4 donne les évaluations détaillées des contrôleurs d'admission et des politiques d'allocations de nuages qui leur sont associées.

7.1 Protocole expérimental

Dans le but d'évaluer le service d'allocation de noeuds et de création de nuages pair-à-pair proposé dans cette thèse, nous avons implanté l'ensemble des protocoles formant la plateforme Salute au sein d'un environnement de simulation fortement inspiré du simulateur Peersim [89]. Ces deux simulateurs utilisent un modèle de programmation orienté objet pour recréer un environnement distribué abstrait et un modèle d'exécution événementiel pour simuler l'exécution de protocoles pair-à-pair. Ces environnements de simulation permettent ainsi d'étudier des protocoles pair-à-pair de façon analytique, en simulant des interactions entre les noeuds du système. Le simulateur utilisé dans le cadre de cette thèse utilise également

des traces de churn afin de recréer la dynamique de certains environnements pair-à-pair de référence.

Environnements pair-à-pair simulés : Les environnements pair-à-pair simulés dans le cadre de l'évaluation de Salute sont modélisés à l'aide de traces collectées par Brighten Godfrey [68]. Ces traces ont été construites en échantillonnant la disponibilité des noeuds composant les trois systèmes pair-à-pair de référence suivant : le système Skype [35, 73], l'environnement de machines de bureau de l'entreprise Microsoft, utilisée dans l'étude de Bolosky et coll. [43] et le système PlanetLab [36, 142]. Elles décrivent la disponibilité des pairs avec une précision de l'ordre de l'heure, sur une période de 28 jours. Le protocole expérimental, visant à évaluer les différents aspects de la plateforme Salute, s'appuie sur ces traces de churn afin de construire différents environnements simulés, au sein desquels la plateforme Salute peut être déployée et étudiée. Dans la série d'évaluations proposée au sein de cette thèse, ces environnements sont respectivement composés de : 650 pairs pour la trace modélisant l'environnement PlanetLab, 2000 pour la trace modélisant l'environnement du système Skype et 10000 pour la trace modélisant l'environnement du système de Microsoft. L'intérêt d'utiliser plusieurs traces est de disposer de plusieurs environnements de simulation, qui diffèrent par leur taille, leur taux de dynamisme et le modèle de variation de la disponibilité de leurs noeuds. Les caractéristiques principales de ces trois environnements sont données sur la Figure 7.1.

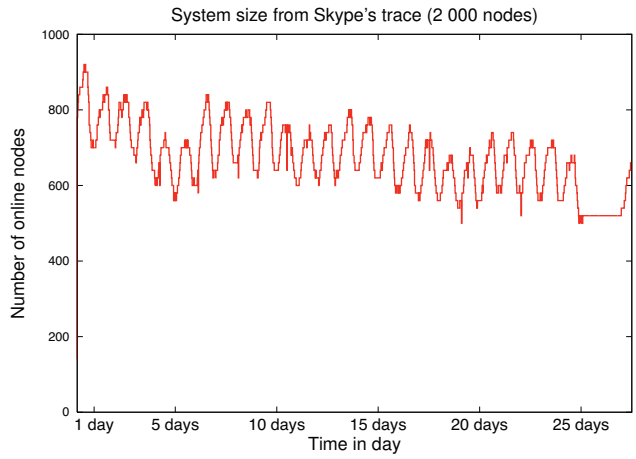
A noter que la disponibilité des noeuds est émulée par le simulateur, en respectant la trace de churn associée, de façon à produire une dynamique du système dont la fréquence est supérieure à un événement par heure. En effet si l'on suit strictement les traces fournies par Brighten Godfrey [68], les départs et arrivées des différents noeuds sont synchronisés selon une période d'une heure, ce qui ne reflète pas la dynamique réelle des environnements pair-à-pair. Cette synchronisation est due à la période d'échantillonnage utilisée pour récolter ces traces [68]. Aussi, de façon à répartir le churn caractérisé par ces traces de façon uniforme au cours du temps, le simulateur procède à une émulation de la dynamique du système pour chaque heure simulée. Pour ce faire, au début de chaque heure le simulateur consulte la trace de churn et en déduit, pour chaque noeud, si celui-ci doit effectuer une action de churn (c.-à-d. rejoindre ou quitter le système) dans l'heure à venir. Si c'est le cas, une durée inférieure ou égale à une heure est tirée aléatoirement pour exécuter l'arrivée ou le départ du noeud concerné. Cette durée peut varier de 1 à 3600 secondes, de sorte que la dynamique des pairs puisse être simulée à l'échelle de la seconde. Lorsqu'elle est appliquée à l'ensemble des pairs du système, cette méthode permet de répartir uniformément les départs et arrivées de noeuds dans le temps. Le comportement du système est alors plus proche d'un système réel, dans le sens où les départs et arrivées de noeuds sont distribués dans le temps au lieu d'être exécutés simultanément toutes les heures.

Déploiement de la plateforme Salute sur le simulateur : Le protocole expérimental adopté pour l'évaluation de Salute consiste à déployer la plateforme pair-à-pair sur un des environnements simulés décrits précédemment, puis de jouer différents scénarios de requêtes d'allocation de noeuds, afin d'évaluer les services d'allocation et de gestion de nuages pair-à-pair. Pour ce faire, chaque noeud simulé suit les protocoles définis par la plateforme Salute, c'est-à-dire le protocole de maintenance de son architecture, le protocole Sarcasm et les protocoles éventuellement utilisés pour implanter les contrôleurs d'admission.

Chaque évaluation se décompose en une série de simulations. La configuration d'une simulation comprend quatre paramètres principaux : la trace correspondant à l'environnement pair-à-pair à simuler, le pourcentage de pairs de réserve présent dans le système, un scénario

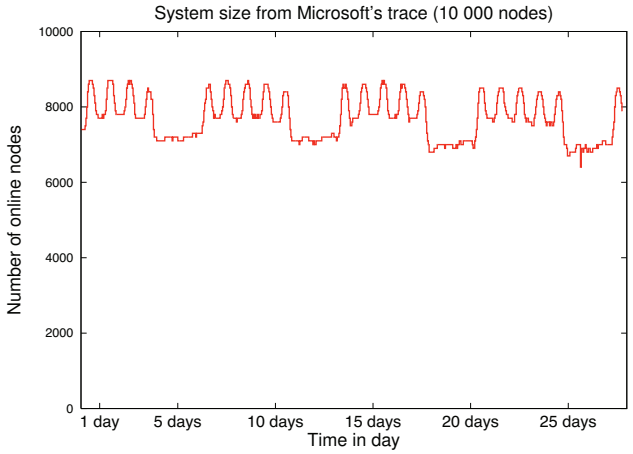
Skype

Nombre de noeuds : 2000
Taille moyenne : 692 (± 75)
Churn horaire moyen : 2,5% ($\pm 2,5\%$)
Taille minimale : 504
Taille maximale : 933
Durée moyenne de session : 10h
Nombre moyen de sessions : 11,2



Microsoft

Nombre de noeuds : 10000
Taille moyenne : 7730 (± 570)
Churn horaire moyen : 0,7% ($\pm 1\%$)
Taille minimale : 6428
Taille maximale : 8792
Durée moyenne de session : 60h
Nombre moyen de sessions : 11,2



Planetlab

Nombre de noeuds : 650
Taille moyenne : 243 (± 14)
Churn horaire moyen : 0,3% ($\pm 3\%$)
Taille minimale : 204
Taille maximale : 280
Durée moyenne de session : 1 semaine
Nombre moyen de sessions : 33

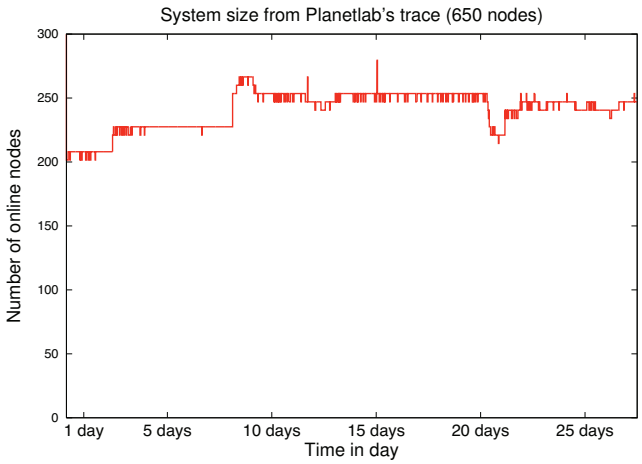


FIGURE 7.1 – Caractéristiques des environnements simulés pour une période de 28 jours.

de requêtes spécifique à l'évaluation en cours et une graine pour le générateur aléatoire utilisé par le simulateur. Lors du démarrage d'une simulation, l'historique de disponibilité est attribué individuellement à chaque noeud en chargeant la trace correspondant au système à simuler. Initialement, tous les noeuds disponibles font partie de l'entrepôt de noeuds, c'est-à-dire que lorsqu'une simulation démarre, aucun nuage n'est déployé. Les pairs démarrent donc la simulation avec le statut « libre » ou « de réserve ». Ces statuts sont attribués aux noeuds en utilisant le générateur aléatoire, tout en vérifiant que le pourcentage de pairs de réserve est initialement égal à celui spécifié dans la configuration de la simulation. Les topologies initiales des réseaux virtuels hébergeant la plateforme Salute sont également configurées à l'aide du générateur aléatoire, de façon à produire une topologie conforme à celles données par un protocole d'échantillonnage de pairs [84]. Ainsi, l'exécution des différentes simulations dépend de la graine utilisée pour configurer le générateur aléatoire chargé d'établir les liens du réseau virtuel initial, d'émuler le churn et de distribuer les statuts de disponibilité aux pairs. Le générateur aléatoire utilisé lors de chaque simulation est le « Mersenne Twister » [111].

Une fois la simulation initialisée, les protocoles épidémiques chargés de la maintenance de la plateforme Salute, ainsi que les services pair-à-pair de gestion de réseau virtuel décrits au sein de la Section 5.1.2, sont démarrés. Tous les noeuds du système procèdent donc à des échanges de messages périodiques, dans le cadre de l'exécution de ces protocoles épidémiques. Afin de recréer des conditions de communication réelles, un taux de perte de messages constant de 2% est appliqué. Pour ce faire, le générateur aléatoire est utilisé pour choisir et annuler environ 2% des envois de messages, de façon à simuler une panne du réseau, c'est-à-dire sans avertir la source ou la destination de l'occurrence de cette panne. Finalement, de façon à garantir l'équité des communications, on considère que la latence des envois de message est égale pour chaque paire de noeuds.

Evaluation par scénarios de requêtes : Les différentes évaluations proposées dans cette thèse sont conduites en jouant différents scénarios de requêtes d'allocation de nuages pair-à-pair au sein de la plateforme simulée. Dans ces scénarios, une liste de « demandeurs » choisis aléatoirement est fixée durant l'initialisation de la simulation. Ces demandeurs sont des pairs particuliers, qui soumettent des requêtes d'allocations de noeuds à la plateforme Salute. Les requêtes d'allocation sont émises de façon périodique, en suivant un modèle prédéfini par le scénario de requêtes qui guide la simulation. Dans ce type de scénario, le générateur aléatoire est typiquement utilisé pour produire un ordonnancement aléatoire des requêtes. L'idée de ce mode d'évaluation est d'observer le comportement de la plateforme simulée en exécutant différentes simulations qui suivent le même scénario de requêtes. En initialisant le générateur aléatoire avec une graine différente pour chaque simulation, on obtient une série d'exécutions différentes, chacune suivant le même scénario de requête. Des valeurs statistiques sont ainsi calculées à partir d'une quinzaine de simulations et le comportement de la plateforme Salute, pour un scénario de requêtes donné, est analysé à partir de ses valeurs. Finalement, un récapitulatif des paramètres de simulations génériques de la plateforme Salute est donné dans la Table 7.1.

Le protocole expérimental décrit au sein de cette section est utilisé pour évaluer le coût, en terme de bande passante consommée, et l'équité du traitement des requêtes d'allocations gérées par le protocole Sarcasm, sous différents degrés de concurrence (Cf. Section 7.3). Il est également utilisé pour comparer l'efficacité de plusieurs contrôleurs d'admission et politiques d'allocation (Cf. Section 7.4). Finalement, la section suivante présente une étude de l'impact du pourcentage de pairs de réserve sur les performances du service d'allocation de noeuds proposé par la plateforme Salute.

Nom du paramètre	état	valeur(s)
durée de la simulation	fixe	28 jours
pourcentage de perte de messages	fixe	2%
trace de churn	variable	PlanetLab, Skype, Microsoft
pourcentage de pairs de réserve	variable	1% à 40%
graine du générateur aléatoire	variable	valeur entière
scénario de requêtes	variable	protocole

TABLE 7.1 – Paramètres génériques de simulation.

7.2 Configuration du pourcentage de pairs de réserve

Comme expliqué précédemment dans le Chapitre 4, le pourcentage de pairs de réserve est un paramètre de configuration sensible pour la plateforme Salute. En effet, si un nombre excessif de pairs est alloué sous le statut « de réserve », alors l'excédent de pairs de réserve constituera un ensemble de ressources disponibles inutilisées. En revanche, si le pourcentage de pairs de réserve est trop faible, alors le maintien du déploiement des nuages sera plus difficile et la plateforme Salute sera probablement contrainte d'annuler le déploiement de certains nuages. Par conséquent, la clef d'une bonne configuration pour la plateforme Salute est de trouver le pourcentage minimal de pairs de réserve qui soit suffisant pour couvrir la plupart des déploiements de nuages pair-à-pair de façon fiable.

Afin de déterminer une valeur par défaut pour le pourcentage de pairs de réserve, nous avons conduit une évaluation dans laquelle on tente périodiquement de construire le maximum de nuages pair-à-pair, et ce, pour des valeurs du pourcentage de réserve variant de 1% à 40%. Le scénario de requêtes utilisé est le suivant : on choisit 10 pairs qui soumettent simultanément une requête d'allocation toutes les 10 heures. Le nuage demandé est le même pour chaque requête, sa taille correspond à 10% de la taille totale du système et sa durée de déploiement est de 8 heures, de sorte que la plateforme ne puisse satisfaire que 9 de ces requêtes au maximum (le pourcentage minimal de pair de réserve est de 1%). Ce scénario de requêtes est joué sur chacun des environnements simulés (Skype, Microsoft et PlanetLab), de façon à confronter la capacité de Salute à allouer des nuages pair-à-pair à des systèmes de taille et de composition différentes. Ce scénario de requête utilise différentes graines pour alimenter le générateur de nombre aléatoire, ce qui détermine notamment la façon dont les requêtes d'allocations sont propagées par Sarcasm (Cf. Chapitre 5). Les paramètres de simulations de cette évaluation sont similaires à ceux définis dans la Table 7.1 pour la configuration de la plateforme Salute. Les paramètres de configuration du scénario de requêtes sont fixes. Ils sont récapitulés dans la Table 7.2.

Scénario de requêtes :

Nom du paramètre	état	valeur(s)
nombre de demandeurs	fixe	10
période entre deux vagues de requêtes	fixe	10 heures
durée de déploiement des nuages demandés	fixe	8 heures
taille des nuages demandés	fixe	10% de la taille du système

TABLE 7.2 – Paramètres de simulation pour l'évaluation de l'impact du pourcentage de pairs de réserve sur les performances du service d'allocation de noeuds, sans contrôle d'admission.

On mesure alors le nombre total de nuages déployés avec succès et le nombre de déploiements annulés lors d'une même simulation, et ce, pour différentes valeurs du pourcentage de pairs de réserve. A noter que le déploiement d'un nuage est annulé lorsque celui-ci ne peut plus maintenir sa taille de référence, du fait d'un nombre insuffisant de pairs de réserve. Les résultats de cette évaluation sont donnés sur les Figures 7.2 et 7.3, dans lesquelles chaque point correspond à la moyenne de mesures effectuées sur 15 simulations.

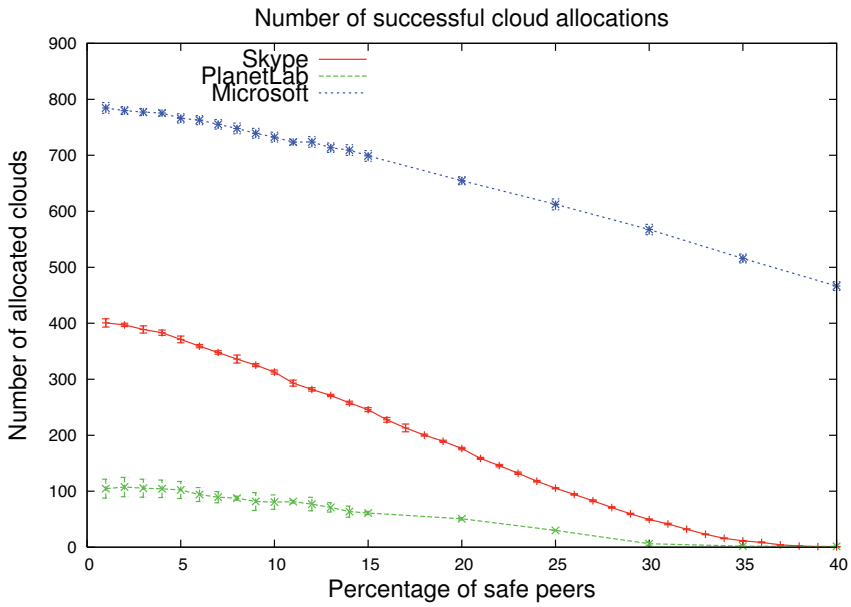


FIGURE 7.2 – Evaluation du service d'allocation de noeuds en fonction du pourcentage de pairs de réserve : nombre de nuages déployés avec succès.

Tout d'abord, on remarque que quel que soit l'environnement considéré, le nombre de nuages déployés avec succès est inversement proportionnel au pourcentage de pairs de réserve, ce qui confirme nos attentes. Si le système comporte davantage de de pairs de réserve, alors il comporte également moins de pairs libres et, de ce fait, peut allouer moins de nuages pair-à-pair. Le nombre de pairs de réserve doit donc être le plus petit possible, de façon à optimiser le nombre de ressources allouables par la plateforme Salute. Ensuite, on peut constater que le nombre de nuages annulés est proche de zéro pour une valeur faible du pourcentage de pairs de réserve, sur les environnements Microsoft et Planetlab. Par contre, pour l'environnement Skype il faut que le système possède au moins 30% de pairs de réserve pour qu'aucun nuage ne soit annulé. Cette observation s'explique par le fait que le dynamisme de l'environnement Skype est beaucoup plus important que ceux des autres systèmes considérés. Le pourcentage de pairs de réserve doit donc être configuré en fonction du dynamisme du système.

Afin de fixer un pourcentage de pairs de réserve par défaut pour ces environnements, il est important de noter que ces évaluations ont été conduites sans utiliser de contrôleur d'admission. En effet, le rôle des contrôleurs d'admission est de diminuer le nombre de nuages pairs à pairs annulés pour cause d'insuffisance de pairs de réserve. Pour ce faire, les contrôleurs

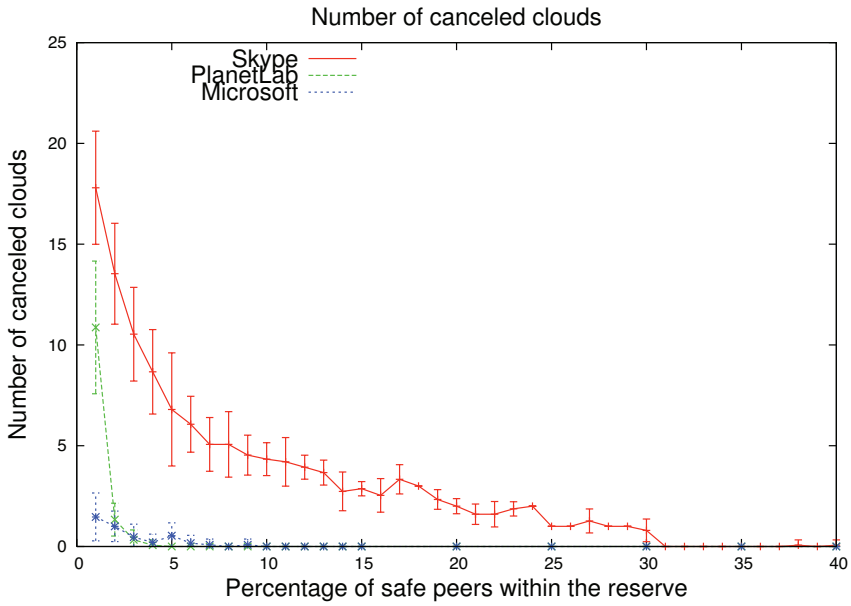


FIGURE 7.3 – Evaluation du service d'allocation de noeuds en fonction du pourcentage de pairs de réserve : nombre de déploiements de nuage annulés.

d'admission utilisent des prédictions sur l'évolution de la taille du système, auxquelles sont couplées des politiques d'allocations décrites dans le Chapitre 6. Pour les systèmes Microsoft et PlanetLab, nous avons choisi de fixer ce pourcentage à 5%, car d'après les résultats de l'évaluation précédente, le nombre de nuages annulés pour cette configuration est stabilisé autour d'une valeur presque nulle. Le rôle des contrôleurs d'admission sera alors de veiller à ce que le nombre d'annulations soit réellement nul en appliquant des politiques d'allocations appropriées. Pour le système Skype, fixer un pourcentage par défaut à 30% serait pénalisant pour les performances du service d'allocation de noeuds. Nous avons choisi de fixer cette valeur à 10%, valeur pour laquelle le nombre d'annulations semble se stabiliser. En effet, on peut observer sur les résultats de la Figure 7.2 que le nombre d'annulations décroît de façon exponentielle pour des valeurs comprises entre 1% et 7%, puis décroît de façon linéaire pour des valeurs supérieures à 7%. On peut donc espérer qu'avec des politiques d'allocations adaptées, le comportement de la plateforme Salute puisse être stabilisé pour une configuration du nombre de pairs de réserve supérieure à 10%. Des évaluations plus complètes et plus détaillées du service d'allocation de noeuds de Salute sont données dans les sections suivantes.

7.3 Evaluation de Sarcasm

L'évaluation du protocole de gestion de nuages pair-à-pair « Sarcasm » est divisée en trois parties. La première partie de cette évaluation est une étude comparative des deux protocoles de construction de nuage proposés dans le Chapitre 5, à savoir le protocole utilisant le vol de noeuds par priorité et le protocole ne l'utilisant pas. La seconde partie de cette évaluation vise à étudier l'équité du protocole de construction de nuages en présence de concurrence, et en particulier l'impact de la taille des nuages demandés sur l'équité du service d'allocation de noeuds. Finalement, la troisième partie de cette évaluation étudie le passage à l'échelle de Sarcasm en proposant une étude statistique des diffusions épidémiques utilisées par le protocole Sarcasm pour construire les nuages pair-à-pair. Cette étude statistique permet notamment de déduire une estimation de la bande passante moyenne consommée par la plateforme Salute, ainsi qu'une estimation du temps moyen nécessaire pour construire un nuage pair-à-pair, grâce au protocole Sarcasm.

7.3.1 Evaluation du vol par priorité en présence de concurrence

Le but de cette première évaluation est de montrer que l'utilisation des mécanismes de priorité décrits dans la section précédente permet à Sarcasm de fournir un service de construction de nuage fiable en présence de concurrence des requêtes d'allocation. Pour ce faire, nous avons réalisé une évaluation comparant les performances de deux versions du protocole Sarcasm. La première version correspond à celle décrite au sein de la Figure 5.4 du Chapitre 5, c'est la version du protocole de construction de nuage qui n'utilise pas les mécanismes de priorité. La seconde version est celle décrite au sein de la Figure 5.7 du Chapitre 5, elle correspond à la version du protocole de construction de nuage qui utilise les priorités attribuées aux requêtes pour implanter le vol par priorité.

Ces deux versions de Sarcasm sont comparées à l'aide d'un scénario de requêtes, dont le but est d'évaluer les performances du service d'allocation de noeuds en fonction du degré de concurrence des requêtes. Ce scénario est défini par les points suivants :

- L'environnement pair-à-pair simulé correspond au système Microsoft défini dans la Section 7.1. Cet environnement est composé de 10000 noeuds et permet d'effectuer des simulations d'une durée de 28 jours. La disponibilité des noeuds composant ces environnements pair-à-pair est simulée avec une précision d'une heure, puis émulée à l'échelle de la seconde, selon le protocole décrit au sein de la Section 7.1. Les évaluations conduites sur les autres environnements pair-à-pair simulés conduisent sensiblement au même type de résultat. Par souci de concision, seuls les résultats des évaluations réalisées sur l'environnement Microsoft sont présentés dans cette section. Les résultats du système simulé Microsoft ont été choisis prioritairement, car la grande taille de ce système permet au scénario de requêtes présenté ci-après de se placer dans un contexte large échelle, au sein duquel il demande l'allocation de nuages dont la taille est de l'ordre de plusieurs centaines de noeuds.

- Les pairs représentant les clients du service d'allocation sont choisis aléatoirement durant la procédure d'initialisation de la plateforme Salute. Ces clients sont au nombre de 25, ce qui permet de simuler une concurrence des requêtes potentiellement forte, tout en gardant des tailles de nuage supérieures à une centaine de noeuds.

- Les requêtes d'allocation sont déclenchées par vague, avec un intervalle de 2 heures entre deux vagues successives. Le temps de déploiement des nuages est de 1 heure.

Cette configuration assure que les déploiements des nuages demandés lors d'une vague V sont terminés lorsque la vague suivante $V+1$ est déclenchée. Ces durées sont choisies de façon à maximiser le nombre de vagues déclenchées par simulation, lequel est de 336 vagues de requêtes par simulation.

- Le scénario de requêtes joué lors de cette évaluation comporte deux paramètres principaux : le degré de concurrence des requêtes D_C et la période L_P durant laquelle les requêtes concurrentes d'une même vague sont émises.

  – Le degré de concurrence D_C est défini par la relation :

  $$D_C = \frac{N_{Req}}{N_{Free}},$$

  où N_{Req} est le nombre total de noeuds demandés et N_{Free} le nombre de noeuds libres disponibles dans le système, lorsque la requête est soumise à Salute. C'est ce paramètre qui va déterminer la taille des nuages demandés par les clients du système d'allocation. En effet, le nombre de clients étant fixe, la taille des nuages T_N est alors calculée, lors de chaque vague d'émission de requêtes, en utilisant la relation suivante :

  $$T_N = \frac{N_{Free} * D_C}{N_{Client}},$$

  où N_{Client} correspond au nombre de clients (c.-à-d. possède la valeur 25). La valeur du degré de concurrence varie de 1 (le nombre total de noeuds demandés est égal au nombre de noeuds libres) à 2 (le nombre total de noeuds demandés est égal au double du nombre de noeuds libres), avec un incrément de 0,1. La taille des nuages demandés par ce scénario de requêtes peut varier de 230 à 640 noeuds, en fonction du degré de concurrence et en fonction du nombre de pairs libres au moment où les requêtes sont émises.

  – Les requêtes concurrentes d'une même vague sont émises dans une fenêtre de temps de période L_P. La valeur de cette période varie de 1 à 10 secondes. Cette fenêtre de temps est utilisée pour répartir le déclenchement des requêtes concurrentes, de façon uniforme, sur une période plus ou moins longue. Pour ce faire, lors de chaque vague d'émission de requêtes, un temps d'attente dont la valeur est choisie aléatoirement entre 1 et L_P seconde(s) est attribué à chaque client. Les clients émettent chacun une requête d'allocation par vague, lorsque le délai d'attente est écoulé. Notons que lorsque la valeur du paramètre L_P est fixée à 1 seconde, alors les clients émettent leurs requêtes d'allocation simultanément.

- Le simulateur utilisé pour cette évaluation assure que les requêtes concurrentes sont propagées avec la même latence au sein du réseau virtuel du magasin de noeuds, si bien qu'aucune requête n'est privilégiée par la qualité des canaux virtuels de communication.

- Les priorités associées aux requêtes sont uniquement basées sur les identifiants des pairs, c'est-à-dire que les requêtes sont équivalentes du point de vue du critère générique et que les requêtes d'une même vague sont estampillées par la même époque de diffusion (Cf. paragraphe « attribution des priorités » du Chapitre 5).

- Ce scénario de requête utilise un contrôleur d'admission qui accepte systématiquement toutes les requêtes d'allocation soumises au système.

- Pour chaque simulation, on retient le pourcentage de requêtes traitées avec succès (c'est-à-dire dont le traitement conduit au déploiement du nuage demandé), par rapport au

nombre de requêtes d'allocation soumises au système. Pour une meilleure lisibilité des résultats, ce pourcentage est normalisé par le degré de concurrence. Cette normalisation se justifie par le fait que lorsque toutes les requêtes d'allocation sont propagées en même temps, le contrôleur d'admission les accepte toutes. Par conséquent, le nombre maximal de requêtes satisfaites N_{Max} suit la relation suivante :

$$N_{Max} = \frac{N_{Free}}{T_N} = \frac{N_{Client} * N_{Free}}{D_C * N_{Free}} = \frac{N_{Client}}{D_C},$$

où N_{Free} est le nombre de noeuds libres disponibles, N_{Client} le nombre de requêtes concurrentes, T_N la taille des nuages demandés et D_C le degré de concurrence des requêtes. Ainsi, afin de pouvoir comparer les performances du service d'allocation indépendamment du degré de concurrence des requêtes, les résultats sont normalisés par le paramètre D_C.

Les paramètres de ce scénario de requêtes sont récapitulés au sein de la Table 7.3.

Plateforme Salute :

Nom du paramètre	**état**	**valeur(s)**
durée de la simulation	fixe	28 jours
pourcentage de perte de messages	fixe	2%
trace de churn	fixe	Microsoft
pourcentage de pairs de réserve	fixe	5%
graine du générateur aléatoire	variable	valeur entière

Scénario de requêtes :

Nom du paramètre	**état**	**valeur(s)**
nombre de demandeurs	fixe	25
période entre deux vagues de requêtes	fixe	2 heures
durée de déploiement des nuages demandés	fixe	1 heure
Degré de concurrence	variable	de 1 à 2
Période de répartition des requêtes	variable	de 1 à 10 secondes

TABLE 7.3 – Paramètres de simulation pour l'évaluation des mécanismes de vol par priorité utilisés par le protocole Sarcasm pour construire les nuages pair-à-pair.

Les résultats donnés sur la Figure 7.4 illustrent les performances de Sarcasm, avec et sans le vol de noeuds, pour des degrés de concurrence variant de 1 à 2, et lorsque les requêtes d'allocation sont diffusées simultanément (LP = 1s) ou dans un intervalle de 2 secondes (LP = 2s). D'après ces résultats, lorsque Sarcasm n'utilise pas le vol de noeuds et que les requêtes sont propagées simultanément, il est incapable d'allouer le moindre nuage lorsque le degré de concurrence des requêtes dépasse la valeur de 1.5. Ce résultat est dû au phénomène d'interblocage des requêtes qui est détaillé au sein de la Section 5.2.2. En effet, si le vol de noeuds n'est pas utilisé alors que le degré de concurrence est largement supérieure à 1, chaque requête d'allocation parvient à recruter une partie des noeuds dont elle a besoin pour construire le nuage demandé, mais aucune ne parvient à compléter le nuage ciblé. Dans les mêmes conditions d'utilisation, lorsque Sarcasm utilise le vol de noeuds, le service d'allocation fournit des performances quasiment optimales.

Dans la même configuration, lorsque les requêtes sont propagées dans un intervalle de 2 secondes, la version de Sarcasm n'utilisant pas le vol de noeuds présente une efficacité

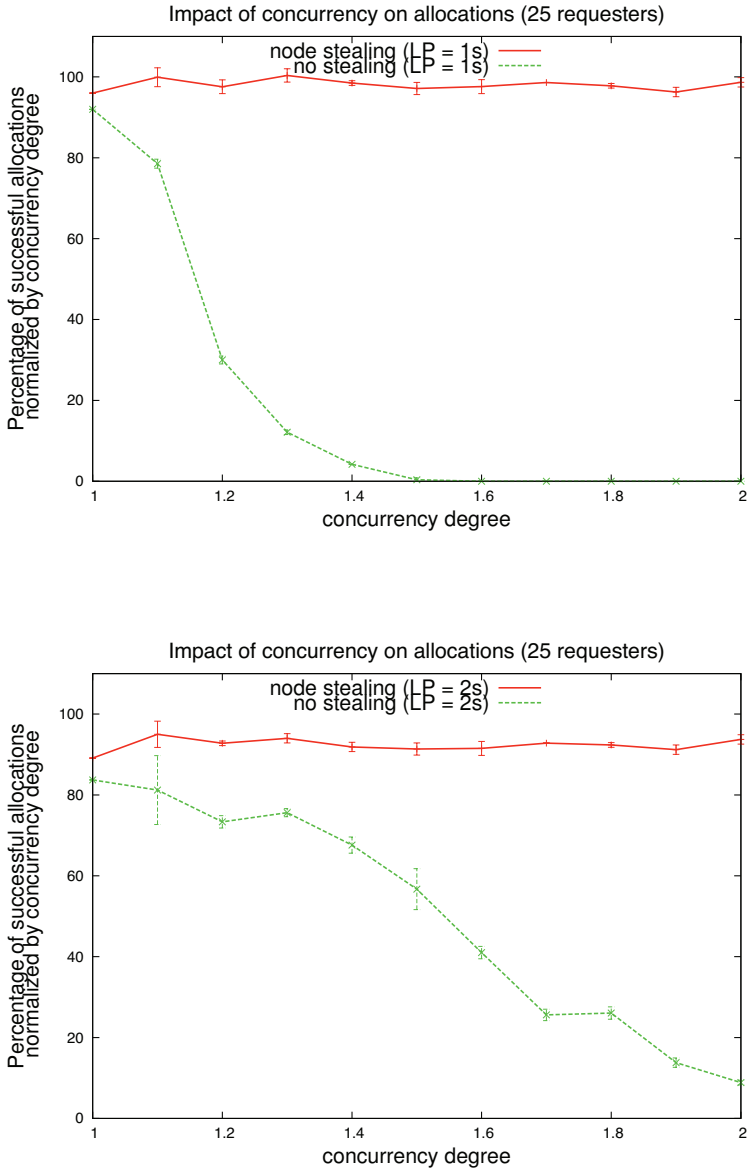


FIGURE 7.4 – Évaluation de Salute en fonction du degré de concurrence des requêtes, pour des émissions de requêtes simultanées (version normalisée) et réparties sur intervalle de temps de 2 secondes.

inversement proportionnelle au degré de concurrence. Ce résultat s'explique par le fait que le phénomène d'interblocage des requêtes est nuancé par la durée de l'intervalle pendant lequel les requêtes concurrentes sont émises. Afin de déterminer l'impact de cet intervalle de temps, nous avons simulé le scénario de requête décrit précédemment pour différentes valeur de la période L_P définissant la longueur de cet intervalle. Les résultats donnés sur la Figure 7.5 illustrent les performances de Sarcasm, avec et sans le vol de noeuds, pour des degrés de concurrence variant de 1 à 2 et pour des valeurs de la période L_P de 5 et 10 secondes.

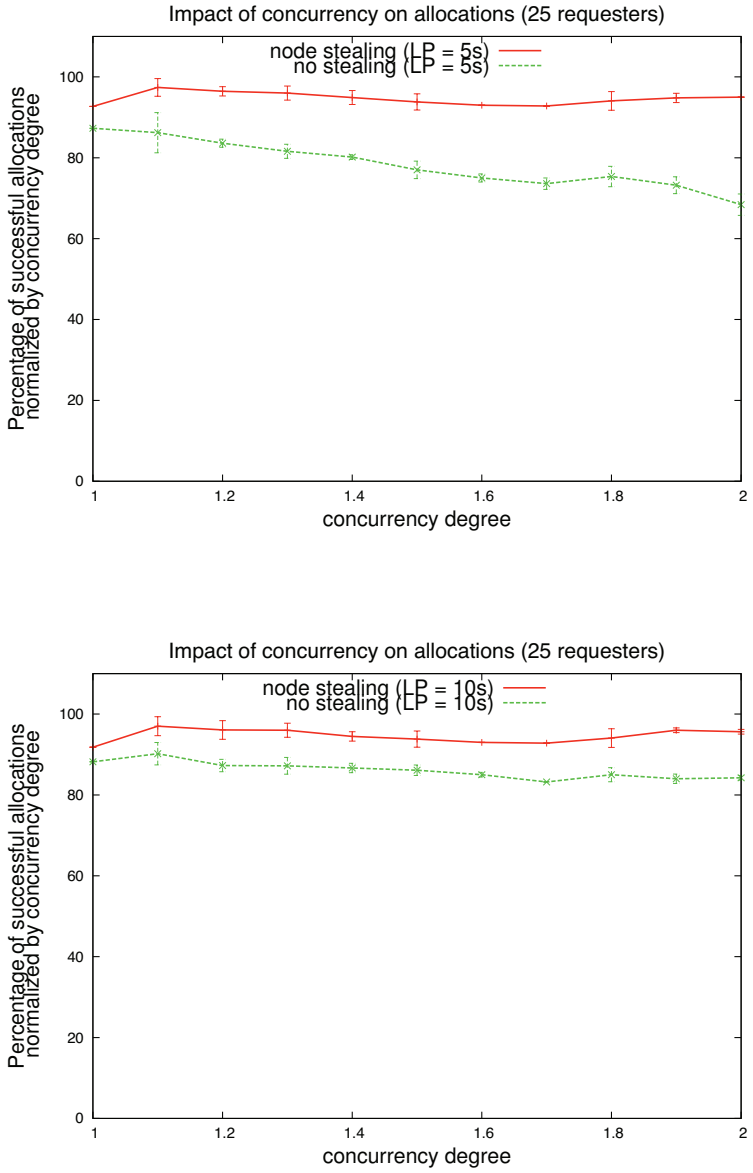


FIGURE 7.5 – Évaluation de Salute en fonction du degré de concurrence des requêtes, pour des émissions de requêtes réparties sur un intervalle de temps de 5 et de 10 secondes.

Il apparaît alors, que lorsque des requêtes concurrentes sont diffusées dans un intervalle de temps supérieur à 5 secondes, alors les performances du protocole Sarcasm, sans vol de noeuds, sont nettement moins dégradées que lorsque ces mêmes requêtes sont diffusées simultanément. En effet, plus l'intervalle de temps de diffusion des requêtes concurrentes est grand, plus le phénomène d'interblocage décrit dans le Chapitre 5 est nuancé. Des simulations prenant en compte des valeurs du paramètre L_P supérieures à 10 secondes ont également été effectuées. Il apparaît alors que lorsque la durée de l'intervalle de temps, pendant lequel les

requêtes concurrentes sont diffusées est grand, plus le phénomène d'interblocage des requête est nuancé. Ces dernières simulations ne sont pas illustrées dans ce document, car les résultats correspondants sont très proches de ceux présentés dans la Figure 7.5.

Finalement, la version de Sarcasm utilisant le vol par priorité fournit un service de construction de nuage pair-à-pair fiable. En effet, quelque soit le degré de concurrence des requêtes et l'intervalle de temps durant lequel elles sont déclenchées, l'utilisation de ce protocole permet de maximiser le nombre de nuages alloués. L'utilisation du vol de noeuds par priorité est donc une méthode suffisante et efficace pour construire des nuages pair-à-pair en présence de concurrence des requêtes d'allocation de noeuds. Cette méthode semble également résoudre certains cas pathologiques de concurrence des requêtes, notamment lorsque de nombreuses requêtes sont diffusées simultanément.

7.3.2 Equité du traitement des requêtes d'allocation

Cette section montre comment utiliser le critère générique de la politique d'attribution de priorités des requêtes d'allocation, pour assurer l'équité du service d'allocation de noeuds. En particulier, l'évaluation présentée ici cherche à montrer que l'on peut établir des politiques d'attribution de priorité qui assurent, sous certaines conditions, que Salute ne favorise aucune requête d'allocation de noeuds, en particulier lorsque de nombreuses requêtes concurrentes et hétérogènes sont soumises au service d'allocation de nuages. Comme expliqué dans la Section 5.2.2, Sarcasm doit attribuer des priorités aux requêtes d'allocation de façon à assurer la convergence du système vers un état stable, c'est-à-dire vers un état où la présence de requêtes d'allocation conduit à la création d'au moins un nuage pair-à-pair en un temps raisonnable. L'utilisation de ces mécanismes de priorité nécessite de vérifier que Salute ne favorise pas certaines requêtes au détriment des autres, au cours du temps. Dans cette évaluation, on s'intéresse particulièrement à l'impact de la taille des requêtes sur l'équité du service d'allocation de nuages.

Utilisation d'un critère d'équité : Le critère d'équité choisi dans cet évaluation est appelé le *ratio Allocations/Demandes*. Ce ratio représente la proportion de noeuds reçus par un client au cours du temps, en fonction du nombre de noeuds qu'il demande. Ce critère d'équité noté R_{Alloc} est défini, à un instant t donné, par la relation suivante :

$$R_{Alloc}(t) = \frac{N_{Demand}(t)}{N_{Received}(t)},$$

où $N_{Demand}(t)$ est le nombre total de noeuds demandés par un client entre son arrivée dans le système et l'instant t et où $N_{Received}(t)$ est le nombre total de noeuds reçus par ce même client jusqu'à l'instant t. On considère alors que le service d'allocation de Salute est équitable si, après une période suffisamment longue, tous les clients du système possèdent un ratio *Allocations/Demandes* équivalent.

Politique d'attribution de priorités équitable : De façon à étudier l'équité du service d'allocation de nuages, nous proposons de définir une politique d'attribution de priorités qui favorise les requêtes émises par des clients ayant reçu moins de noeuds que les autres, dans le passé, du point de vue du critère d'équité basé sur le ratio *Allocations/Demandes*. On rappelle que l'ordre des priorités utilisé par Sarcasm suit la règle suivante pour comparer deux requêtes R_i et R_j :

> Si une des deux requêtes a été initiée avant l'autre, alors elle est systématiquement prioritaire. Si les requêtes R_i et R_j ont été initiées durant la même période, alors

le critère d'équité est utilisé pour départager ces deux requêtes. C'est alors la requête émise par le client ayant le plus faible ratio *Allocations/Demandes* qui est prioritaire. Finalement, si la période de diffusion des requêtes R_i et R_j est la même et que le critère d'équité est incapable de départager ces requêtes, alors on utilise les identifiants des pairs ayant initié les diffusions de R_i et de R_j pour établir la comparaison.

A noter que l'historique du nombre de noeuds alloués à un pair donné dans le passé, et du ratio *Allocations/Demandes* qui en découle, est stocké localement par tous les pairs au sein de la variable locale *gen*, utilisée pour stocker les informations relatives au critère générique d'attribution des priorités (Cf. Figure 5.8 du Chapitre 5). De cette façon, lorsqu'un pair initie la diffusion d'une requête d'allocation de nuage, il peut y inclure la valeur de son ratio *Allocations/Demandes*. En procédant de la sorte, les pairs libres peuvent comparer deux requêtes d'allocation du point de vue du critère d'équité, en utilisant la valeur de l'historique fourni par les clients.

L'évaluation présentée dans la suite de ce document propose d'étudier l'équité du service d'allocation de noeuds de la plateforme Salute, en opposant la politique d'attribution de priorités présentée ci-dessus et une politique d'attribution de priorités dans laquelle le critère générique n'est pas utilisé, c'est-à-dire dans laquelle les priorités sont attribuées uniquement en fonction de la date à laquelle les requêtes ont été émises.

Scénario de requêtes et évaluation : Le scénario de requêtes joué pour évaluer l'équité du service d'allocation de noeuds est configuré comme suit :

- L'environnement pair-à-pair simulé correspond au système Microsoft défini dans la Section 7.1. Cet environnement est composé de 10000 noeuds et permet d'effectuer des simulations d'une durée de 28 jours. La disponibilité des noeuds composant cet environnement pair-à-pair est simulée avec une précision d'une heure, puis émulée à l'échelle de la seconde, selon le protocole décrit au sein de la Section 7.1. Les évaluations conduites sur les autres environnements pair-à-pair simulés conduisent sensiblement au même type de résultat. Par souci de concision, seuls les résultats des évaluations réalisées sur l'environnement Microsoft sont présentés dans cette section. Les résultats du système simulé Microsoft ont été choisis prioritairement, car la grande taille de ce système permet au scénario de requêtes présenté ci-après de se placer dans un contexte large échelle, au sein duquel il demande l'allocation de nuages dont la taille varie de 500 à 5000 noeuds.
- Les requêtes d'allocation sont regroupées par vague. Chaque vague correspond à l'exécution de 20 requêtes concurrentes, chacune étant émise par un pair différent.
- Les pairs représentant les clients du service d'allocation sont choisis aléatoirement durant la procédure d'initialisation de la plateforme Salute. Une taille de nuage est associée à chacun de ces pairs, de sorte qu'un pair soumette systématiquement la même requête pendant toute une même simulation. Les tailles des nuages sont aléatoirement choisies durant la procédure d'initialisation parmi un ensemble de 10 possibilités, chacune étant un multiple de 500 et étant comprise entre 500 et 5000 noeuds. Le temps de déploiement des nuages demandés est de 2 heures.
- Les vagues de requêtes sont déclenchées selon un intervalle de 10 heures, si bien que 2 vagues sont déclenchées par jour simulé. Aussi, les déploiements de nuages précédents, s'ils existent, sont terminés lorsqu'une nouvelle vague de requêtes est déclenchée.

- Les requêtes concurrentes d'une même vague sont émises dans une fenêtre de temps de période L_P. La valeur de cette période varie de 1 à 3600 secondes. Cette fenêtre de temps est utilisée pour répartir le déclenchement des requêtes concurrentes, de façon uniforme, sur une période plus ou moins longue. Pour ce faire, lors de chaque vague d'émission de requêtes, un temps d'attente dont la valeur est choisie aléatoirement entre 1 et L_P seconde(s) est attribué à chaque client. Les clients émettent chacun une requête d'allocation par vague, lorsque le délai d'attente est écoulé. Notons que lorsque la valeur du paramètre L_P est fixée à 1 seconde, alors les clients émettent leur requêtes d'allocations simultanément.

- Le simulateur utilisé pour cette évaluation assure que les requêtes concurrentes sont propagées avec la même latence au sein du réseau virtuel du magasin de noeuds, si bien qu'aucune requête n'est privilégiée par la qualité des canaux virtuels de communication.

- Le ratio *Allocations/Demandes* reçu par chaque client est mesuré pour chaque simulation. Ces mesures sont ensuite regroupées par rapport à la taille des nuages demandés par les clients. On retient alors la moyenne de 15 mesures pour chacune des deux politiques d'attribution de priorité, chaque mesure étant établie sur une simulation différente. Finalement, le générateur aléatoire est configuré avec une graine différente lors de chaque simulation, si bien que la composition des vagues de requêtes est différente d'une exécution à l'autre.

Les paramètres de ce scénario de requêtes sont récapitulés au sein de la Table 7.4.

Plateforme Salute :

Nom du paramètre	**état**	**valeur(s)**
durée de la simulation	fixe	28 jours
pourcentage de perte de messages	fixe	2%
trace de churn	fixe	Microsoft
pourcentage de pairs de réserve	fixe	5%
graine du générateur aléatoire	variable	valeur entière

Scénario de requêtes :

Nom du paramètre	**état**	**valeur(s)**
nombre de demandeurs	fixe	20
période entre deux vagues de requêtes	fixe	10 heures
durée de déploiement des nuages demandés	fixe	2 heures
taille des nuages demandés	variable	500 à 5000
valeur du paramètre L_P	variable	1 à 3600 secondes

TABLE 7.4 – Paramètres de simulation pour l'évaluation de l'équité de Sarcasm.

La moyenne des mesures du ratio *Allocations/Demandes* est représentée sur la Figure 7.6, pour des scénarios de requêtes au sein desquels les diffusions des requêtes sont déclenchées, soit simultanément, soit uniformément pendant une période de 10 secondes. Les mêmes mesures sont représentées sur la Figure 7.7, pour des scénarios de requêtes au sein desquels les diffusions des requêtes sont déclenchées uniformément pendant une période de 15 secondes pour un premier scénario, et pendant une période d'une heure pour le second.

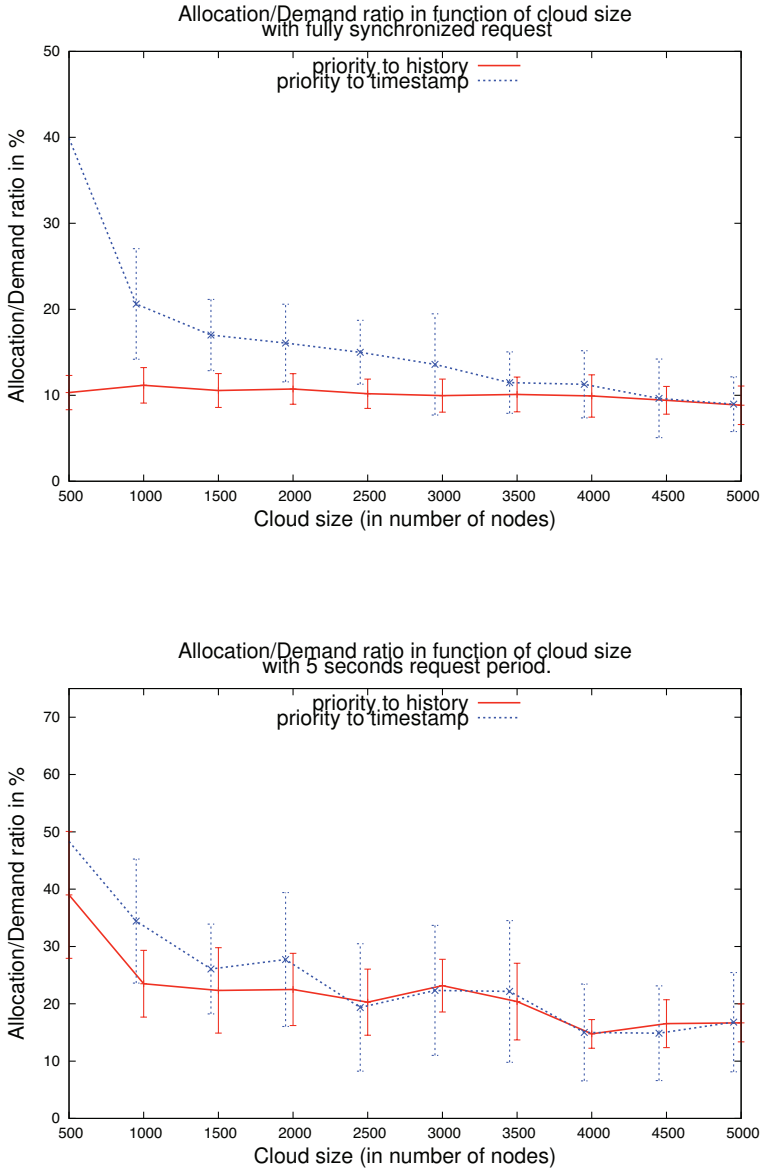


FIGURE 7.6 – Evaluation du ratio *Allocations/Demandes* pour des périodes de diffusion de requêtes d'une seconde et de 5 secondes.

Les résultats reportés sur la Figure 7.6 montrent que les clients demandant des petits nuages sont naturellement favorisés, lorsque la politique d'attribution des priorités ne tient pas compte du critère d'équité et que les requêtes d'allocation sont diffusées simultanément. En contrepartie, lorsque le critère d'équité est utilisé pour attribuer les priorités aux requêtes d'allocations, les clients reçoivent le même ratio *Allocations/Demandes*, indépendamment de la taille des nuages qu'ils demandent. Cette évaluation, montre que le service d'allocation de noeuds de la plateforme Salute peut être équitable, si les requêtes d'allocation de noeuds sont

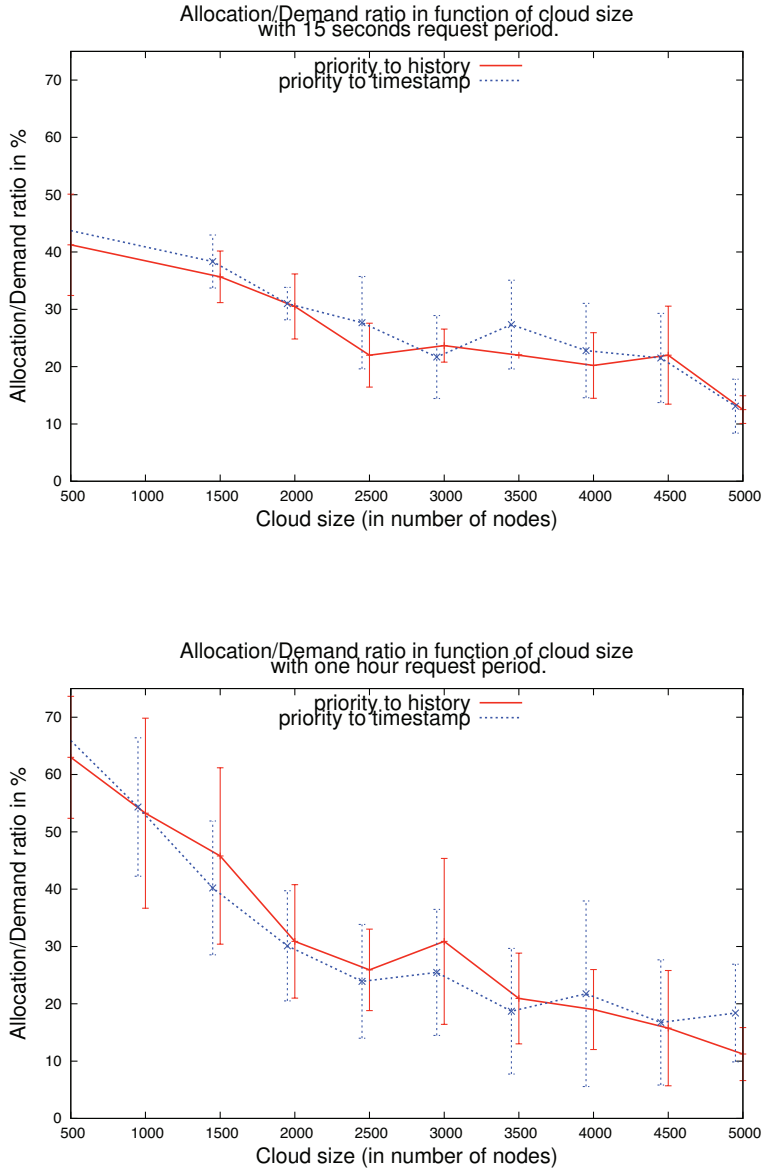


FIGURE 7.7 – Evaluation du ratio *Allocations/Demandes* pour des périodes de diffusion de requêtes de 15 secondes et d'une heure.

déclenchées simultanément. Les limitations de cette propriété d'équité apparaissent lorsque l'on augmente la durée de la fenêtre de temps pendant laquelle les requêtes d'allocation sont déclenchées.

Comme on peut le voir sur la Figure 7.6, lorsque les requêtes concurrentes sont déclenchées pendant une période s'étalant sur 5 secondes, les requêtes les plus petites sont de nouveau privilégiées. Intuitivement, ce phénomène s'explique simplement par le fait que les nuages les plus petits sont les plus faciles à construire. Ainsi, selon une multitude de scénarios de

requêtes, lorsqu'une large proportion des noeuds disponibles est allouée à un moment donné (par exemple 75%), seules les petites requêtes (c.-à-d. celles qui ne demandent pas plus de 25% du total des noeuds disponibles) peuvent être satisfaites.

Pour compléter cette évaluation, la Figure 7.7 illustre les résultats produits par le scénario de requêtes décrit dans cette section, selon deux configurations. Dans la première configuration, les déclenchements respectifs des 20 requêtes d'allocation sont répartis uniformément sur une période de 15 secondes. Dans la seconde configuration, ils sont répartis uniformément sur une période d'une heure. Il apparaît alors que lorsque la période de diffusion des requêtes mises en concurrence est supérieure ou égale à 15 secondes, le ratio *Allocations/Demandes* moyen reçu par les clients est inversement proportionnel à la taille des nuages qu'ils demandent. De plus, cette tendance à favoriser les requêtes demandant de petits nuages semble s'accentuer au fur et à mesure que la durée de la fenêtre de déclenchement des requêtes augmente. A noter que l'impact de la durée de cette fenêtre, pour des valeurs supérieures à une heure, est similaire à celui représenté par la seconde courbe de la Figure 7.7, correspondant au scénario dans lequel les déclenchements de requêtes sont répartis sur une période d'une heure. Dans le cas où la durée de la fenêtre de déclenchement des requêtes est supérieure ou égale à une heure, le ratio *Allocations/Demandes* moyen reçu par les clients est inversement proportionnel à la taille des nuages qu'ils demandent.

Pour conclure, cette évaluation montre que le service d'allocation de noeuds proposé par la plateforme Salute peut être équitable, y compris en présence de mécanismes basés sur l'utilisation de priorités. Cependant, cette propriété d'équité n'est satisfaite que sous certaines conditions, en particulier lorsque les requêtes d'allocation sont déclenchées simultanément. Le service d'allocation de noeuds de la plateforme Salute doit donc contraindre les requêtes d'allocation à être diffusées de façon synchronisée. Nous pensons que cette contrainte est facilement respectable, du fait qu'il existe un service de synchronisation des pairs au sein de la plateforme Salute. La condition de simultanéité des requêtes peut alors être satisfaite, par exemple, en forçant les pairs à synchroniser le déclenchement de leurs requêtes d'allocation selon une période fixe $\Delta Request$. Aussi, un paramètre intéressant, pour fixer cette période $\Delta Request$ de façon empirique, est la latence moyenne de la construction d'un nuage via le protocole Sarcam. L'évaluation de cette latence est présentée au sein de la section suivante.

7.3.3 Coût du protocole Sarcasm :

Les deux critères étudiés dans cette thèse pour évaluer le coût d'exécution du protocole Sarcasm et de la plateforme Salute sont : *i)* la bande passante moyenne consommée par les pairs et *ii)* le temps moyen nécessaire pour construire un nuage pair-à-pair. Ces deux critères sont notamment mesurés en fonction de la taille du système et en fonction de la taille des nuages demandés.

Ces évaluations dépendent, en partie, de certains protocoles de l'état de l'art utilisés par la plateforme Salute (Cf. Nyclon [95], Average et T-Size [121] du Chapitre 3), mais dépendent également des caractéristiques de Sarcasm. En particulier, nous proposons une évaluation du nombre moyen d'étapes de diffusion nécessaire pour construire un nuage avec le protocole Sarcasm et une évaluation du nombre moyen de messages de requêtes d'allocation reçus par les membres de Salute, et ce, en fonction du nombre de requêtes demandées au cours du temps. Combinée aux évaluations des protocoles de l'état de l'art utilisés par Salute, cette évaluation permet d'estimer la bande passante moyenne consommée par les membres de Salute, ainsi que le temps moyen requis pour construire un nuage pair-à-pair avec le protocole Sarcasm. Ces estimations permettent notamment d'établir des valeurs empiriques pour la liste des bornes

temporelles utilisées dans la description du protocole Sarcasm (Cf. Chapitre 5). Un rappel de la nature de ces bornes est donné dans la Table 7.5 :

Nom	**Fonction associée**
$\Delta_{Unicast}$	Borne le temps d'une communication unitaire
Δ_{Bdcast}	Borne le temps d'une diffusion épidémique
Δ_{Count}	Borne le temps du comptage de pairs
Δ_{Build}	Borne le temps d'une construction de nuage

TABLE 7.5 – Bornes temporelles sur les actions des pairs via le protocole Sarcasm.

Comme dans tout système distribué, l'hypothèse de synchronie à la base de toute communication est une borne supérieure sur le temps nécessaire pour effectuer une communication unitaire, c'est-à-dire un simple échange de message entre deux noeuds. Dans cette thèse, nous considérons que cette borne, nommée $\Delta_{Unicast}$, vaut 1 seconde. D'après l'étude de Ledlie et coll. [102], cette hypothèse est valable pour plus de 99,5% des communications réalisées via Internet. A partir de cette borne sur une communication simple, on peut définir la borne Δ_{Bdcast} associée au temps nécessaire pour effectuer une diffusion épidémique fiable. Elle est définie par la relation suivante, où le paramètre Δ_{Steps} représente le nombre d'étapes nécessaires pour que la diffusion épidémique se termine :

$$\Delta_{Bdcast} = \Delta_{Steps} * \Delta_{Unicast} \qquad (1)$$

La borne Δ_{Count} borne le temps nécessaire pour évaluer la taille d'un réseau virtuel non structuré. La période Δ_{Build} représente une borne sur le temps nécessaire à Sarcasm pour réaliser la construction d'un nuage pair-à-pair, à partir du moment où la requête associée est diffusée au sein du magasin de noeuds. D'après le protocole Sarcasm, décrit au sein du Chapitre 5, cette borne satisfait la relation suivante :

$$\Delta_{Build} = \Delta_{Steps} * (\Delta_{Unicast} + \Delta_{Count}) + \Delta_{Bdcast} \qquad (2)$$

En effet, on rappelle que la diffusion épidémique utilisée par le protocole sarcasm attend une période Δ_{Count} entre chaque étape de diffusion, de façon à prendre en compte la variation de taille du nuage à construire au fur et à mesure que la diffusion se propage. Une fois le nuage complet, le protocole Sarcasm émet une diffusion épidémique simple, afin de notifier les pairs alloués au sein de ce nuage du démarrage de son déploiement. Un des buts de l'évaluation présentée dans cette section est alors de fournir des valeurs expérimentales du paramètre Δ_{Steps}, afin de pouvoir définir des valeurs empiriques pour ces bornes temporelles.

Le scénario de requêtes utilisé pour évaluer le coût du protocole Sarcasm est défini par les points suivants :

- Les environnements pair-à-pair simulés utilisés dans cette évaluation sont les systèmes Microsoft, PlanetLab et Skype. Ils permettent tous d'effectuer des simulations d'une durée de 28 jours. La disponibilité des noeuds composant ces environnements pair-à-pair est simulée avec une précision d'une heure, puis émulée à l'échelle de la seconde, selon le protocole décrit au sein de la Section 7.1.

- Le pourcentage de pairs de réserve est fixé à 5% pour les systèmes simulés Microsoft et PlanetLab, puis à 10% pour le système Skype.

- Les pairs représentant les clients du service d'allocation sont choisis aléatoirement durant la procédure d'initialisation de la plateforme Salute. Le nombre de clients $N_{Clients}$ est

un des paramètres variables du scénario de requêtes. Sa valeur varie de 1 à 55, de façon à produire des simulations dans lesquelles le degré de concurrence des requêtes varie de 0,1 à 2 requêtes par noeud.

- Une configuration de simulation comprend deux paramètres : l'environnement pair-à-pair simulé et le nombre de clients. Chaque configuration est exécutée 10 fois de façon à produire des valeurs statistiques. Pour ce faire, on nourrit le générateur aléatoire utilisé par le scénario de requêtes avec une graine différente lors de chacune de ces 10 exécutions. L'identité des pairs représentant les clients du service d'allocation, l'ordonnancement des allocations et la propagation de chaque requête d'allocation, différent donc d'une simulation à l'autre.

- La taille des nuages demandés est fixée en fonction de l'environnement de simulation. Par conséquent, la valeur F du $fanout$ qui calibre la diffusion épidémique réalisée par Sarcasm et qui varie en fonction de la taille des nuages à construire (Cf. Chapitre 5), varie elle aussi en fonction de l'environnement de simulation. Les configurations suivantes sont alors appliquées :

  - Pour le système Microsoft, la taille des nuages demandés est de 500 noeuds et la valeur du $fanout$ est fixée à 12.
  - Pour le système Microsoft, la taille des nuages demandés est de 70 noeuds et la valeur du $fanout$ est fixée à 7.
  - Pour le système Microsoft, la taille des nuages demandés est de 10 noeuds et la valeur du $fanout$ est fixée à 5.

- Les requêtes d'allocation sont déclenchées par vague, avec un intervalle de 2 heures entre deux vagues successives. Toutes les diffusions sont déclenchées simultanément. Le temps de déploiement des nuages est de 1 heure. Cette configuration assure que les déploiements des nuages demandés lors d'une vague V sont terminés lorsque la vague suivante $V+1$ est déclenchée. Ces durées sont choisies de façon à maximiser le nombre de vagues déclenchées par simulation, lequel est de 336 vagues de requêtes par simulation.

- Ce scénario de requête utilise un contrôleur d'admission qui accepte systématiquement toutes les requêtes d'allocation soumises au système.

- Pour chaque simulation on mesure le nombre moyen de messages reçu par noeud et par seconde pour toutes les diffusions de requêtes simulées, ainsi que la latence moyenne de ces diffusions.

Les paramètres de ce scénario de requêtes sont récapitulés au sein de la Table 7.6.

Plateforme Salute :

Nom du paramètre	état	valeur(s)
durée de la simulation	fixe	28 jours
pourcentage de perte de messages	fixe	2%
trace de churn	variable	Microsoft, Skype, PlanetLab
pourcentage de pairs de réserve	variable	5% ou 10% (Skype)
graine du générateur aléatoire	variable	valeur entière

Scénario de requêtes :

Nom du paramètre	état	valeur(s)
période entre deux vagues de requêtes	fixe	2 heures
durée de déploiement des nuages demandés	fixe	1 heure
nombre de client ($N_{Clients}$)	variable	de 1 à 55

TABLE 7.6 – Paramètres de simulation pour l'évaluation du coût de Sarcasm en termes de bande passante consommée et de latence de construction de nuage.

Latence de construction d'un nuage La latence moyenne de construction d'un nuage, en nombre d'étapes de diffusion et en fonction du degré de concurrence des requêtes, est donnée dans la Figure 7.8. Dans un premier temps, on remarque que cette latence est fixe

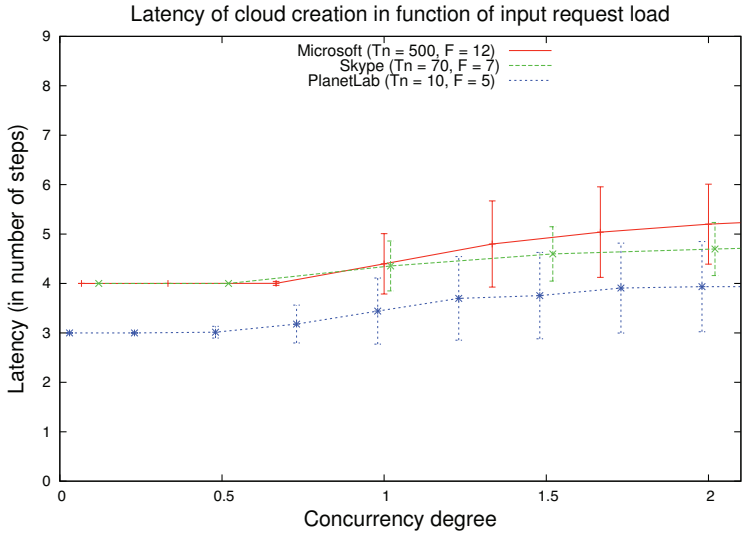


FIGURE 7.8 – Evaluation de la latence du protocol Sarcasm.

pour un degré de concurrence inférieur ou égal à 0,5, puis que le nombre d'étapes moyen augmente de façon logarithmique pour des valeurs du degré de concurrence supérieures à 0,5, quelque soit l'environnement simulé. Cette latence dépend également de la taille des

nuages à construire. A noter, que ces résultats sont cohérents avec les résultats théoriques d'Eugster et coll. [61] présentés dans la Section 3.2.1 de ce document, qui affirment que le nombre d'étapes nécessaire pour qu'une diffusion épidémique touche tous les membres d'un réseau donné est logarithmiquement proportionnelle à la taille de ce réseau. La latence de construction d'un nuage est cependant bornée à 6 étapes de diffusion pour le scénario de requêtes choisi. Par conséquent, nous pouvons établir de façon empirique que la latence de construction d'un nuage $\Delta Steps$, mesurée en nombre d'étapes de diffusion, peut être bornée à six étapes pour la construction de nuages de taille inférieure ou égale à 5000 noeuds, avec un fanout de 12 et pour un degré de concurrence inférieur ou égal à 2. Aussi, d'après les équations *(1)* et *(2)*, données plus tôt dans cette section, et d'après les latences des protocoles de comptage de pairs, présentés au sein de l'état de l'art de ce document (Cf. Section 3.2), nous pouvons estimer les valeurs des bornes temporelles relatives à l'exécution du protocole Sarcasm. La valeur de ces bornes, exprimée en seconde, est donnée dans la Table 7.7. D'après

Nom	**Fonction associée**	**Durée réelle**
$\Delta_{Unicast}$	Communication unitaire	1s
Δ_{Bdcast}	Diffusion épidémique ($S_i \leq 5000$ noeuds)	6s
Δ_{Count}	Comptage de pairs (T-Size)	45s
Δ_{Count}	Comptage de pairs (Average)	60s
Δ_{Build}	Construction de nuage pair-à-pair (avec T-Size)	282s
Δ_{Build}	Construction de nuage pair-à-pair (avec Average)	372s

TABLE 7.7 – Bornes temporelles sur le temps de construction de nuages, avec le protocole Sarcasm, pour un degré de concurrence des requêtes inférieur ou égal à 2 requêtes par noeud.

les résultats de l'évaluation présentée dans cette section, le protocole Sarcasm a besoin d'une période d'environ 6 minutes, au plus, pour construire un nuage d'une taille inférieure ou égale à 5000 noeuds et avec un degré de concurrence des requêtes inférieur ou égal à 2. En guise de comparaison, l'infrastructure en tant que service d'Amazon EC2 présente une latence moyenne de 2 minutes pour l'allocation d'une unique machine virtuelle [3, 66], au sein d'un centre de traitement de données. Nous considérons alors que la latence moyenne du service d'allocation de la plateforme Salute est raisonnablement basse, compte tenu de l'environnement dynamique sur lequel ce service s'exécute et compte tenu du nombre important de noeuds alloués.

Bande passante moyenne consommée pour la construction d'un nuage Afin d'exprimer la bande passante moyenne consommée par Sarcasm en kilo octets par seconde, nous avons évalué la taille moyenne des messages de requêtes T_{Req}. Le détail des poids associés à chaque champ d'un message de requête d'allocation (Cf. Section 5.2) est donné dans la Table 7.8. La taille moyenne T_{Req} d'un message de requête d'allocation est donc de 125 octets. En combinant cette taille de message, avec les mesures du nombre moyen de messages reçu par les pairs de Salute, lorsque le scénario de requêtes défini plus tôt dans cette section est joué, on peut évaluer la bande passante moyenne consommée par le protocole Sarcasm. La valeur de cette bande passante moyenne consommée est donnée dans la Figure 7.9, pour des degrés de concurrence des requêtes variant de 0,1 à 2 requêtes par noeud. Il apparaît alors que la bande passante moyenne consommée est faible et stable lorsque le degré de concurrence est inférieur à 0,75 requêtes par noeud (bande passante inférieure à 2 Ko/s). Puis cette bande passante consommée croît proportionnellement au degré de concurrence, à partir d'un degré de concurrence de 0,75 pour la simulation réalisée sur le système Microsoft, et à partir d'un degré de concurrence de 1 pour les autres systèmes. La croissance de cette bande passante

Nom du champ	Taille en octet
Entête UDP (IPV6)	44 o
Identifiant de pair (adresse IP + port)	5 o
Estampille horaire de l'époque	4 o
Spécification du nuage (taille plus filtres)	22 o
Ensemble de noeuds de contact	50 o
Taille totale T_{Req}	125 o

TABLE 7.8 – Détails des champs compris dans les messages de requête d'allocation.

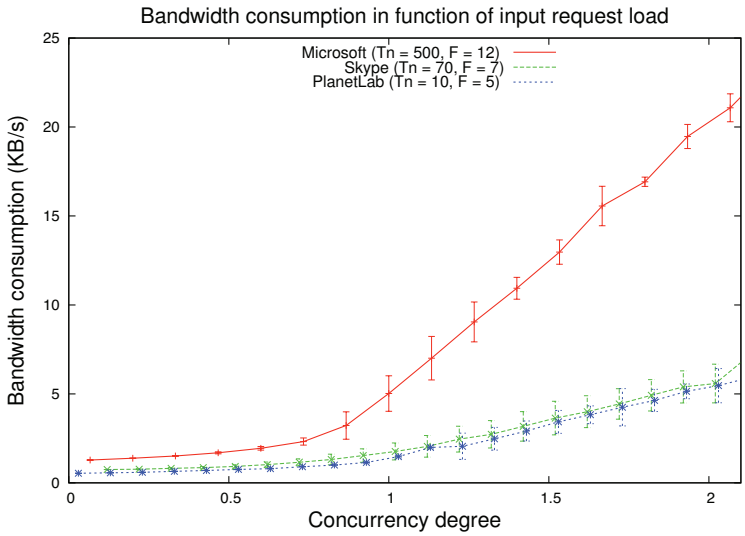


FIGURE 7.9 – Evaluation de la bande passante consommée par le protocole Sarcasm.

consommée, en fonction du degré de concurrence, est proportionnelle à la taille des nuages demandés. En effet, les résultats correspondants aux simulations réalisées sur le système Microsoft affichent une croissance de la bande passante consommée plus rapide que sur les autres systèmes. Ceci s'explique par le fait que, plus les nuages à construire sont grands, plus la valeur du fanout est grande. On rappelle que le fanout est le nombre de destinataires que choisit chaque noeud, lorsqu'il réexpédie un message lors d'une étape de diffusion épidémique (Cf. Section 3.2). Par conséquent, le nombre de messages générés lors d'une diffusion épidémique est directement proportionnel au fanout choisi pour exécuter cette diffusion. La bande passante consommée par Sarcasm est alors proportionnelle à la taille des nuages à construire et proportionnelle au degré de concurrence des requêtes.

En tenant compte de ces évaluations, on peut établir que la bande passante consommée par Sarcasm pour construire des nuages est inférieure ou égale à 5 Ko/s pour des scénarios dans lesquels la concurrence des requêtes est contrôlée, c'est-à-dire dans lesquels le degré de concurrence est inférieur ou égale à une requête par noeud. Cependant, lorsque le degré de concurrence dépasse cette valeur d'une requête par noeud, la bande passante consommée peut atteindre une valeur de 25 Ko/s. A noter que cette bande passante consommée est à

additionner à la bande passante moyenne consommée par les protocoles de l'état de l'art chargés du maintien de la plateforme Salute. En effet, les protocoles d'échantillonnage de pairs, de synchronisation et de comptage de pairs consomment, eux aussi, de la bande passante. D'après l'architecture de Salute, présentée au sein du Chapitre 4, les pairs qui maintiennent la plateforme Salute exécutent au plus trois services d'échantillonnage de pairs (un pour le réseau principal, un pour l'entrepôt et un pour le magasin ou la réserve), deux services de comptage (un pour le réseau principal et un pour le magasin, la réserve, où le nuage) et une instance du service de synchronisation. Aussi, d'après les travaux réalisés par leurs auteurs respectifs, les études de ces protocoles de l'état de l'art montrent que la bande passante moyenne consommée peut être bornée de la façon suivante :

- Le service d'échantillonnage de pairs consomme, au plus, une bande passante de 1Ko/s [95].
- Le service de synchronisation des pairs consomme, au plus, une bande passante de 1Ko/s [80].
- Le service de comptage de pairs $T - Size$ consomme, au plus, une bande passante de 5Ko/s [121].
- Le service de comptage de pairs *Average* consomme, au plus, une bande passante de 1Ko/s [121].

D'après ces données, on peut déduire la bande passante maximum consommée par le maintien de la plateforme Salute. Les estimations de consommation de bande passante sont données, par service, au sein de la Table 7.9.

Nom	**Bande passante consommée**
Echantillonnage de pairs (x3)	3 Ko/s
Synchronisation	1 Ko/s
Comptage de pairs (Average) (x2)	2 Ko/s
Comptage de pairs (T-Size) (x2)	10 Ko/s
Total (Average)	6 Ko/s
Total (T-Size)	14 Ko/s

TABLE 7.9 – Bande passante maximum consommée par les services maintenant la plateforme Salute.

Si on additionne la bande passante maximale consommée par la plateforme Salute à la bande passante maximale consommée par Sarcasm, lorsque le degré de concurrence des requêtes est fort, on obtient une valeur d'environ 40Ko/s. Pour conclure, la bande passante consommée par la plateforme Salute peut être importante (jusqu'à 40Ko/s dans les scénarios extrêmes), mais reste supportable par la majorité des connexions Internet modernes. Il faut noter également, qu'une grande partie de cette bande passante est consommée par le protocole de comptage de pairs (jusqu'à 10 Ko/s avec la protocole T-Size). Un compromis doit alors être réalisé. Si on utilise le protocole T-Size, alors la latence de construction d'un nuage est au maximum 282 secondes et la bande passante moyenne consommée par les noeuds est de 14Ko/s (hors construction de nuage). En revanche, si on utilise le protocole de comptage Average, alors la latence de construction d'un nuage est au maximum 372 secondes, mais la bande passante moyenne consommée, hors construction de nuage, n'est que de 6Ko/s.

7.4 Evaluation des contrôleurs d'admission

Cette section présente l'évaluation des contrôleurs d'admission et des politiques d'allocation déployés au sein de la plateforme Salute. Comme il a été mentionné dans le Chapitre 4, les décisions des contrôleurs d'admission ont un impact majeur sur la qualité du service d'allocation de nuages. D'un côté, si les contrôleurs refusent trop d'allocations, alors une partie des ressources disponibles sera inutilisée et donc gâchée. D'un autre côté, si ces mêmes contrôleurs acceptent trop d'allocations, alors il existe un risque de famine pouvant potentiellement conduire à l'annulation de certains nuages déployés. On considère alors que les ressources monopolisées par un nuage qui est annulé avant la fin de son déploiement sont également gâchées. En conséquence, cette évaluation considère qu'une politique d'allocation est optimale lorsqu'elle alloue le plus grand nombre de ressources possible, tout en assurant qu'aucun des nuages pair-à-pair construits avec succès n'est annulé avant la fin de la période de déploiement spécifiée. Plus précisément, on considère que l'allocation d'un nuage est réussie si le déploiement du nuage en question arrive à son terme sans que sa taille ne soit descendue en dessous de 95% de la taille spécifiée.

Dans cette évaluation, on étudie les performances de trois contrôleurs d'admission différents : un contrôleur optimiste, un contrôleur local et un contrôleur global, et ce, afin de déterminer l'impact des politiques d'allocations qui leurs sont associées sur les performances du service d'allocation. Les caractéristiques de ces trois contrôleurs sont les suivantes :

- **Le contrôleur d'admission optimiste** utilisé dans cette évaluation accepte une requête d'allocation de noeuds sous la simple condition qu'il existe suffisamment de noeuds libres au sein du magasin de noeuds pour la satisfaire, au moment où cette requête est soumise.

- **Le contrôleur d'admission local** utilisé dans cette évaluation est dit « parfait », dans le sens où il utilise des techniques de prédiction ayant une connaissance parfaite de la disponibilité des noeuds du système. Ce contrôleur d'admission utilise la trace de churn déterminant l'évolution des pairs au sein du simulateur décrit dans la Section 7.1, dans le but de calculer des prédictions de disponibilité parfaites. Ainsi, lorsqu'une requête d'allocation est reçue par un pair libre, ce pair ne joint le nuage associé que s'il est réellement disponible pour toute la durée de déploiement spécifiée.

- **Le contrôleur d'admission global** utilisé lors de cette évaluation est un contrôleur à minimum strict. Ce contrôleur est initialisé avec la valeur minimale de la taille du système, laquelle est déterminée d'après la trace de churn utilisé pour les simulations.

De manière à évaluer la capacité des contrôleurs d'admission à accepter le nombre optimal de requêtes d'allocation, le scénario de requêtes choisi pour réaliser cette évaluation fait en sorte de saturer le service d'allocation de noeuds proposé par la plateforme Salute. Des requêtes d'allocation sont donc soumises continuellement à la plateforme simulée. Aussi, d'après nos expérimentations, il apparaît qu'il est intéressant d'évaluer les performances de ces contrôleurs en fonction de la durée de déploiement des nuages demandés. Le scénario de requêtes utilisé pour évaluer les contrôleurs d'admission, à l'aide de simulations, est alors défini comme suit :

- Les environnements pair-à-pair simulés utilisés dans cette évaluation sont les systèmes Microsoft, PlanetLab et Skype. Ils permettent tous d'effectuer des simulations d'une durée de 28 jours. La disponibilité des noeuds composant ces environnements pair-à-pair est simulée avec une précision d'une heure, puis émulée à l'échelle de la seconde, selon le protocole décrit au sein de la Section 7.1.

- Le pourcentage de pairs de réserve est fixé à 5% pour les systèmes simulés Microsoft et PlanetLab, puis à 10% pour le système Skype.

- Les pairs représentant les clients du service d'allocation sont choisis aléatoirement durant la procédure d'initialisation de la plateforme Salute ; ils sont au nombre de 20. La durée des déploiements demandés est la même pour chaque nuage, elle est comprise entre 2 heures et 10 jours et correspond au paramètre de simulation L. La taille des nuages demandés est choisie aléatoirement lors de l'émission de la requête, elle est comprise entre 5% et 15% de la taille du système. En début de chaque heure, lorsque la disponibilité des noeuds est mise à jour par le simulateur à l'aide de la trace de churn, chacun des 20 pairs représentant les clients du service émet une nouvelle requête d'allocation, sauf si un nuage en cours de déploiement lui est associé. En d'autres termes, lorsqu'un pair, parmi ceux choisis pour émettre des requêtes d'allocation, réussit à réserver un nuage avec succès, il cesse de soumettre des requêtes à la plateforme Salute pendant toute la durée de déploiement de ce nuage. Ce scénario de requêtes assure que le service d'allocation est saturé, dans le sens où la somme des requêtes d'allocation qui lui sont demandées est constamment supérieure au nombre de ressources disponibles.

- Une configuration de simulation comprend trois paramètres : l'environnement pair-à-pair simulé, le type de contrôleur d'admission à évaluer et une valeur pour le paramètre L représentant la durée de déploiement des nuages à allouer. Chaque configuration est exécutée 10 fois de façon à produire des valeurs statistiques. Pour ce faire, on nourrit le générateur aléatoire utilisé par le scénario de requêtes avec une graine différente lors de chacune de ces 10 exécutions. L'identité des pairs représentant les clients du service d'allocation, ainsi que les tailles des nuages demandés et l'ordonnancement des allocations différent donc d'une simulation à l'autre.

- La durée de simulation est d'une semaine pour les configurations dans lesquelles la durée de déploiement est comprise entre 1 et 24 heures. Pour les configurations correspondant à des durées de déploiement comprises entre 1 et 10 jours, la durée de la simulation est de 28 jours.

- Pour chaque configuration de simulation, on mesure le nombre moyen de noeuds alloués avec succès et le nombre moyen de nuages annulés. On rappelle qu'un nuage est annulé si sa taille réelle descend en dessous de 95% de sa taille de référence.

Les paramètres du scénario de requêtes utilisé pour évaluer les contrôleurs d'admission sont récapitulés dans la Table 7.10

Les résultats de ces séries de simulation sont donnés sur la Figure 7.10 pour l'environnement simulé Microsoft, sur la Figure 7.11 pour l'environnement simulé PlanetLab et sur la Figure 7.12 pour l'environnement simulé Skype. Sur ces figures, la sous-figure du haut comprend les résultats correspondant aux simulations courtes (c.-à-d. d'une semaine) et la sous-figure du bas représente les résultats correspondant aux simulations longues (c.-à-d. de 28 jours). Au sein de ces figures, les contrôleurs d'admission évalués sont dénommés comme suit : l'appellation « aw » (pour *« Always Yes »*) correspond au contrôleur optimiste, l'appellation « lpo » (pour *« Local Perfect Oracle »*) correspond au contrôleur local parfait et l'appellation « min » correspond au contrôleur d'admission à minimum strict. De façon à rendre la comparaison équitable, la proportion de ressources allouées lors de chaque simulation est exprimée dans la métrique « *nombre de noeuds* $*$ *heure* », si bien que l'on considère qu'un nuage composé de 100 noeuds qui est déployé pendant 2 heures consomme autant de

Plateforme Salute :

Nom du paramètre	état	valeur(s)
durée de la simulation	variable	7 ou 28 jours
pourcentage de perte de messages	fixe	2%
trace de churn	variable	Microsoft, Skype, PlanetLab
pourcentage de pairs de réserve	variable	5% ou 10% (Skype)
graine du générateur aléatoire	variable	valeur entière

Scénario de requêtes :

Nom du paramètre	état	valeur(s)
contrôleur d'admission	variable	optimiste, local ou minimum
nombre de demandeurs	fixe	20
période entre deux vagues de requêtes	fixe	1 heure
durée de déploiement des nuages demandés (L)	variable	2 heures à 10 jours
taille des nuages demandés	variable	5% à 15% de la taille du système

TABLE 7.10 – Paramètres de simulation pour l'évaluation des contrôleurs d'admission.

ressources qu'un nuage composé de 200 noeuds qui est déployé pendant 1 heure. Le service d'allocation est donc optimal lorsque la valeur exprimée dans la métrique « *nombre de noeuds* $*$ *heure* » est à son maximum. Finalement, les valeurs données sur les Figures 7.10,7.11 et 7.12 sont normalisées par rapport au nombre total de nuages demandés lors des simulations, de façon à tenir compte du nombre de requêtes d'allocation annulées après avoir été acceptées par les contrôleurs d'allocation.

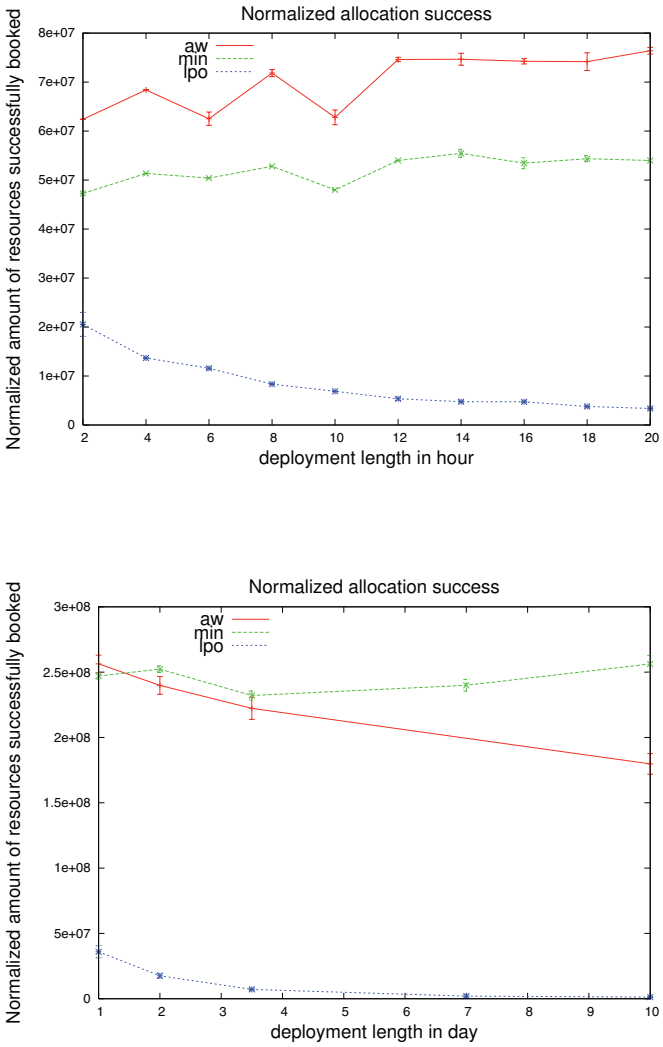


FIGURE 7.10 – Efficacité moyenne des contrôleurs d'admission en fonction de la durée des déploiements, sur l'environnement simulé Microsoft.

Dans un premier temps, nous allons donner l'interprétation brute de ces résultats. Ils seront ensuite analysés plus en profondeur, contrôleur par contrôleur, dans les parties suivantes de cette section. Le premier des résultats à noter est la bonne performance du contrôleur optimiste pour les allocations de nuages à période de déploiement courte. En effet, pour des périodes de déploiement inférieures ou égales à un jour, la configuration de Salute utilisant le contrôleur d'admission optimiste est celle qui parvient à allouer le plus grand nombre de noeuds sur l'environnement Microsoft, et parvient à allouer autant de noeuds que le contrôleur à minimum sur les autres environnements. Pour les périodes de déploiements plus longues, c'est-à-dire supérieures à un jour, le nombre de ressources allouées avec succès par le contrôleur optimiste diminue de façon linéaire, tandis que le nombre de ressources allouées grâce au contrôleur à minimum strict reste stable, et ce, pour tous les environnements simulés.

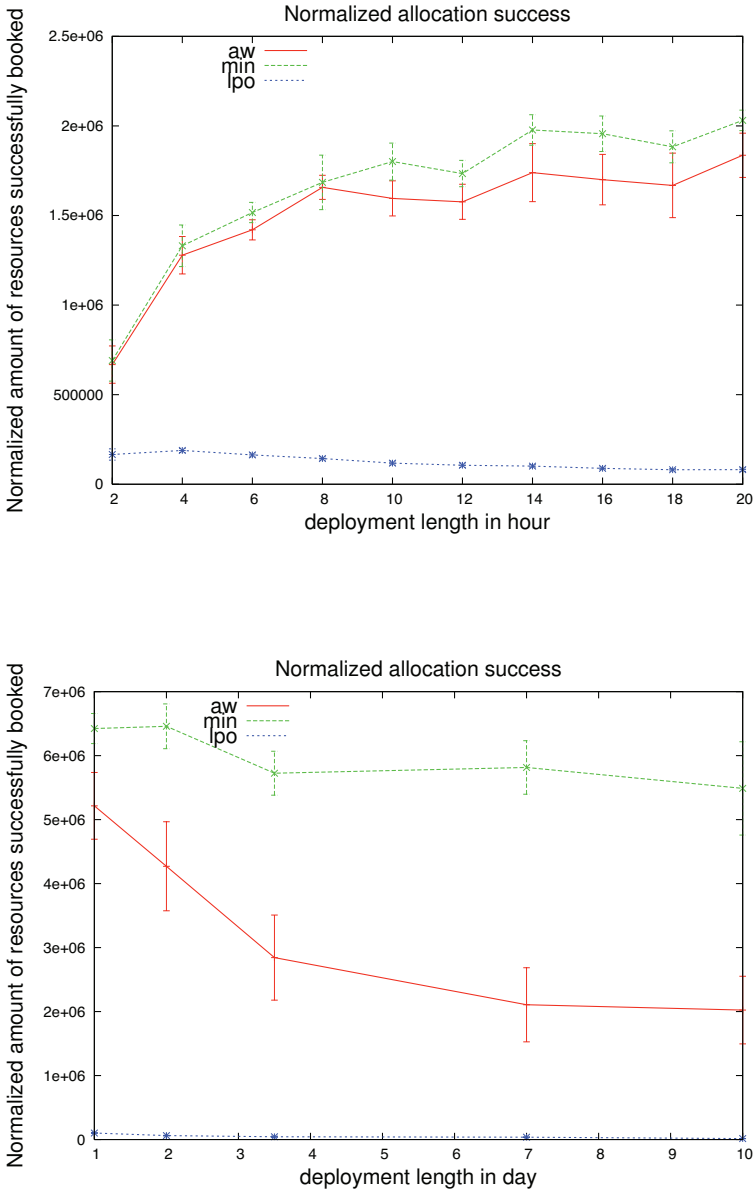


FIGURE 7.11 – Efficacité moyenne des contrôleurs d'admission en fonction de la durée des déploiements, sur l'environnement simulé PlanetLab.

Intuitivement, cette observation induit que le problème de surallocation, introduit lors de la description des contrôleurs d'admission dans le Chapitre 6, apparaît pour des périodes de déploiement de nuages supérieures à un jour, lorsque le contrôleur optimiste est utilisé. Le second résultat important est la mauvaise performance, en comparaison de celles des autres contrôleurs évalués, du contrôleur d'admission local parfait. Ce résultat s'explique principalement par le grand nombre d'annulations subit par les nuages en cours de construction lorsque

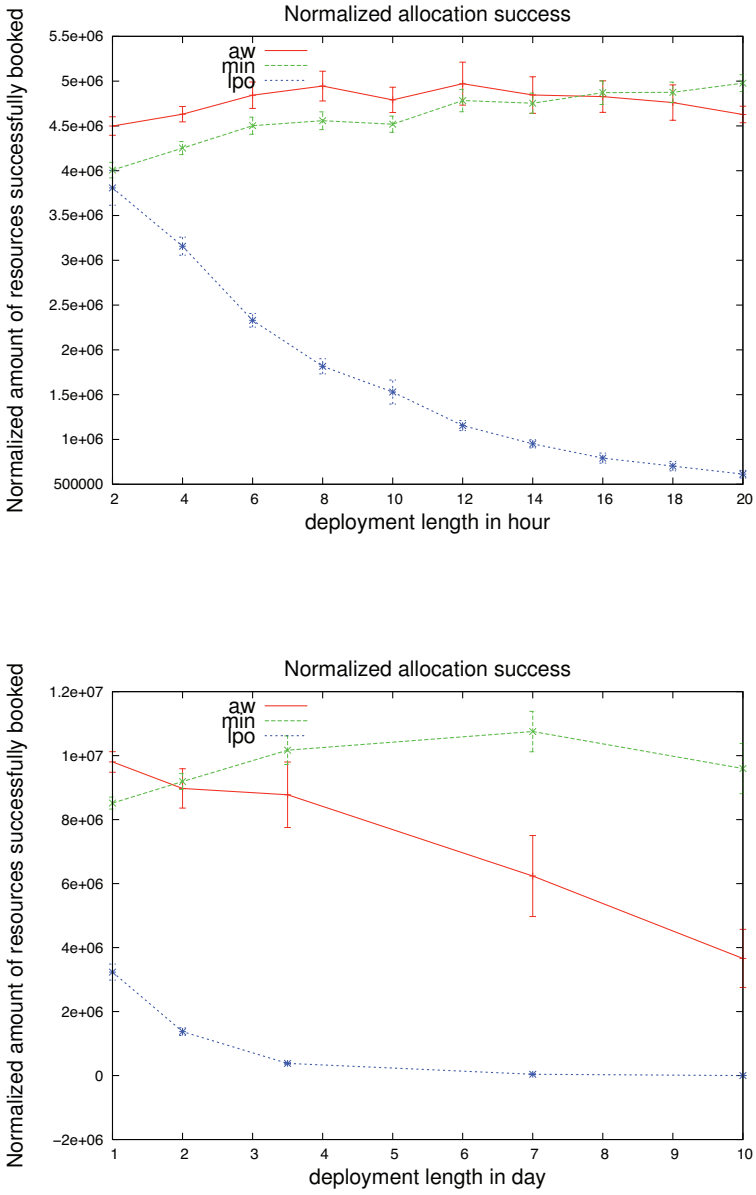


FIGURE 7.12 – Efficacité moyenne des contrôleurs d'admission en fonction de la durée des déploiements, sur l'environnement simulé Skype.

ce contrôleur est utilisé seul. Finalement, on notera la stabilité des performances obtenues par Salute lorsque le service d'allocation utilise un contrôleur d'admission à minimum strict.

Dans la suite de cette section, on utilise des traces générées par les simulations réalisées lors de l'évaluation des contrôleurs d'admission. Ces traces ont pour but d'illustrer le comportement des pairs, lors de certaines simulations de référence, et ce, pour chaque contrôleur étudié. Elle représente l'évolution de la taille du système ainsi que l'évolution du nombre

total de pairs alloués, du nombre de pairs alloués avec succès et, lorsque cela est utile, du nombre de pairs de réserve. En particulier, la comparaison de l'évolution du nombre total de pairs alloués avec celle du nombre de pairs alloués avec succès, nous permet d'identifier les moments où certains nuages déployés sont annulés. Ce type d'analyse permet également d'illustrer le problème de surallocation qui est potentiellement rencontré lorsque Salute utilise un contrôleur d'admission optimiste.

7.4.1 Evaluation du contrôleur d'admission optimiste

L'analyse des performances du contrôleur d'admission optimiste est divisée en deux parties. Dans une première partie, le problème de la surallocation est illustré, tandis que la seconde partie présente une analyse complète des performances du contrôleur d'admission optimiste.

Illustration du problème de surallocation : Afin d'illustrer clairement le problème de surallocation présenté dans cette thèse, nous avons choisi d'exécuter un scénario de requêtes dédié à la production d'une trace dans laquelle ce phénomène apparaît distinctement. Ce scénario de requêtes possède les caractéristiques suivantes :

- L'environnement pair-à-pair simulé correspond au système Microsoft. Cet environnement est composé de 10000 noeuds et permet d'effectuer des simulations d'une durée de 28 jours. La disponibilité des noeuds composant ces environnements pair-à-pair est simulée avec une précision d'une heure, puis émulée à l'échelle de la seconde, selon le protocole décrit au sein de la Section 7.1.

- Sept pairs sont aléatoirement choisis lors de l'initialisation de la plateforme Salute pour représenter les clients du service d'allocation de noeuds. Chacun de ces pairs émet une requête d'allocation toutes les heures. Les nuages demandés ont une taille de 1000 noeuds, ce qui correspond à 10% de la taille totale du système, et la durée de déploiement de ces nuages est de 4 jours.

- Ce scénario utilise un contrôleur d'admission optimiste pour gérer les requêtes d'allocation.

- Le pourcentage de pairs de réserve est fixé à 10%.

La trace produite par l'exécution de ce scénario de requête au sein de notre environnement de simulation est donnée sur la Figure 7.13. Comme on peut le constater sur cette trace, le scénario de requêtes joué donne lieux à l'allocation de 6 nuages pair-à-pair, ce qui correspond à 60% du total des noeuds disponibles alloués avec succès, pendant la période correspondante aux trois premières semaines du déploiement simulé. Durant ces trois premières semaines, le nombre de noeuds alloués avec succès est égal au nombre total de noeuds déployés, ce qui signifie qu'aucun nuage n'a été annulé. Cependant, pendant la période correspondante à la dernière semaine du déploiement simulé, on peut constater que le nombre de noeuds alloués est inférieur au nombre de noeuds alloués avec succès. Cette différence implique que certains nuages déployés ont été annulés.

En observant la trace de la Figure 7.13, on remarque que pendant la période correspondante aux trois premières semaines, le nombre de noeuds alloués correspond à 60% de la taille totale du système, c'est-à-dire à 6000 noeuds. Sachant que le système comporte 10% de pairs de réserve, c'est à dire 1000 pairs de réserve et que la taille moyenne du système est de 7730 noeuds (Cf. Figure 7.1 de la Section 7.1), on peut conclure que le nombre maximal de nuages

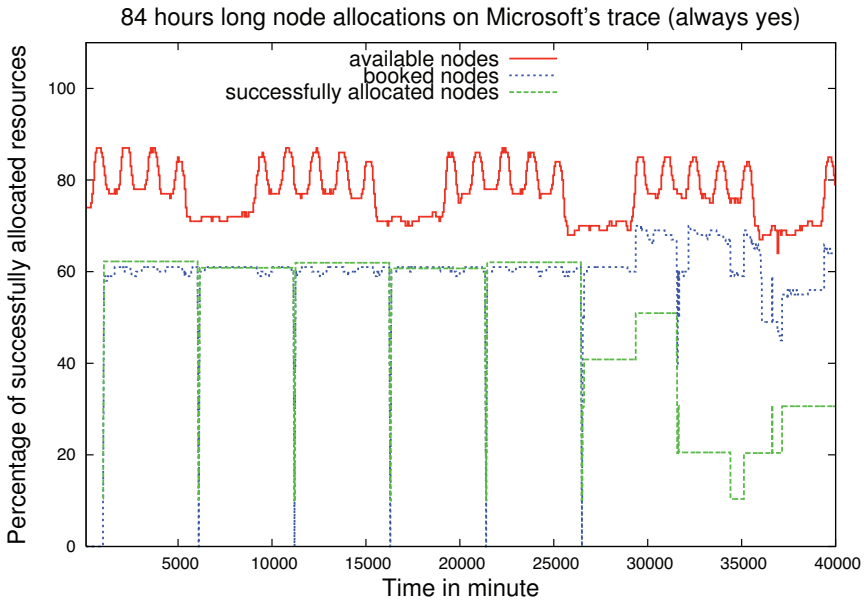


FIGURE 7.13 – Illustration du problème de surallocation.

de 1000 noeuds est alloué pendant cette période. Le problème de surallocation apparaît au moment correspondant au début de la quatrième semaine du déploiement simulé, lorsque le nombre de noeuds alloués atteint 70% de la taille totale du système. A ce moment, le contrôleur optimiste détecte que la taille actuelle du système dépasse les 8000 noeuds, pendant une période correspondant au début des heures de bureau et donc à l'arrivée au sein du système d'un grand nombre de machines supplémentaires [43]. Il décide alors d'accepter une requête d'allocation supplémentaire et le nombre de pairs alloués atteint les 7000 noeuds. Néanmoins, après une période correspondant à une demi-journée, le nombre de machines disponibles au sein de l'environnement hôte diminue rapidement, pendant ce qui correspond à la période de fin des heures de bureau. A ce moment là, aucun des sept nuages n'a terminé son déploiement. Le système manque alors de pairs de réserve pour maintenir ces nuages et certains d'entre eux sont inévitablement annulés. On parle alors d'un problème de surallocation, car si le contrôleur d'admission avait eu connaissance à priori de cette variation de la taille du système, il n'aurait pas autorisé le déploiement du septième nuage. Finalement, ce phénomène se reproduit plusieurs fois au cours de la dernière semaine du déploiement simulé, causant de nouvelles annulations de nuages et dégradant les performances du service d'allocation de noeuds.

Analyse complète des performances du contrôleur optimiste : Bien que le scénario présenté ci-dessus montre que l'utilisation d'un contrôleur d'admission optimiste peut conduire à un problème de surallocation, ce problème n'apparaît que dans des circonstances particulières, ce qui explique que les performances du contrôleur d'admission optimiste soient globalement bonnes. Afin d'analyser ces performances, nous allons utiliser les traces données dans les Figures 7.14 et 7.15. Ces deux traces ont été générées lors de simulations exécu-

tées dans le cadre de l'évaluation décrite au début de cette section. La trace donnée dans la Figures 7.14 correspond à un scénario au sein duquel la durée de déploiement des nuages demandés est de 8 heures, tandis que la trace donnée dans la Figures 7.15 correspond à un scénario au sein duquel la durée de déploiement des nuages demandés est de 4 jours.

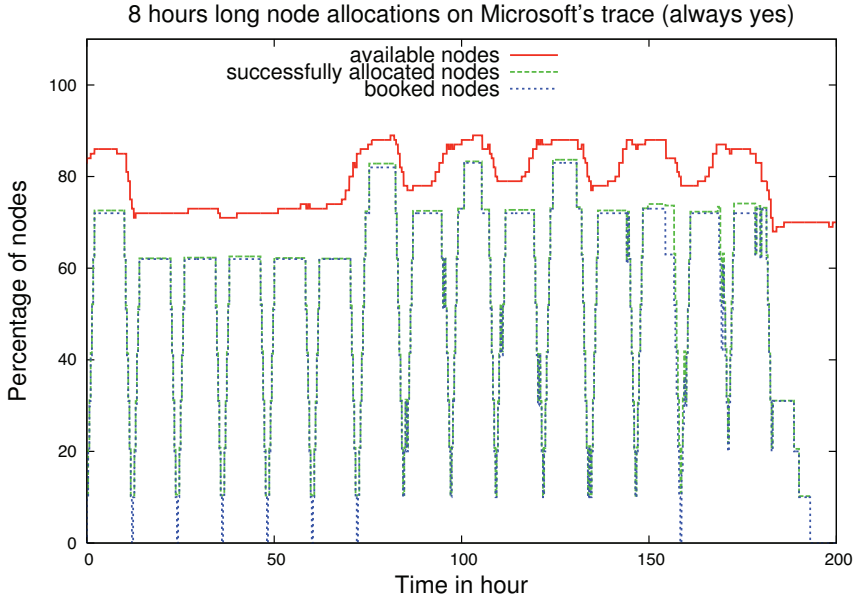


FIGURE 7.14 – Traces d'analyse de la politique d'allocation optimiste.

D'après ces traces, on peut observer que lorsque les temps de déploiement des nuages sont cours, le service d'allocation parvient à allouer jusqu'à 8000 noeuds durant les pics de la journée, en utilisant le contrôleur d'admission optimiste. En effet, lorsque la durée de déploiement est inférieure à 1 jour, le contrôleur optimiste ne provoque pas de surallocation. Plus précisément, cette surallocation est compensée avec succès par les pairs de réserve. Cette compensation est possible, car la durée de déploiement des nuages à secourir est courte. De fait, pendant que la taille du système décroît, certains déploiements de nuages se terminent, libérant ainsi suffisamment de noeuds pour alimenter les nuages dont le déploiement continue, et ce, malgré la diminution de la taille du système. Lorsque la durée de déploiement des nuages est de 4 jours, le problème de surallocation cause au moins une annulation de nuage par semaine simulée. Ces annulations peuvent être observées sur la trace donnée dans la Figure 7.15, elles correspondent aux moments où le nombre de pairs alloués est supérieur au nombre de pairs alloués avec succès. De plus, cette figure montre que les pairs de réserve sont vivement sollicités pendant les courtes périodes correspondantes aux diminutions quotidiennes et brutales de la taille du système. Cela signifie que de nombreuses requêtes de secours sont déclenchées pendant ces périodes. Aussi, lorsque la durée de déploiement des nuages est nettement supérieure à 1 jour, la surallocation n'est plus parfaitement compensée, car la probabilité que le déploiement d'un nuage se termine pendant les périodes critiques (c.-à-d. pendant les périodes où la taille du système diminue brutalement) est trop faible.

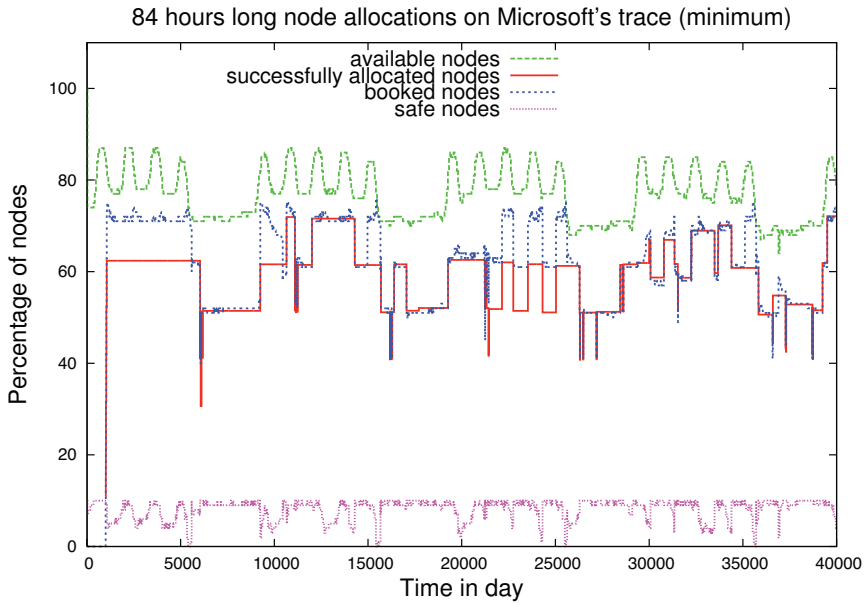


FIGURE 7.15 – Traces d'analyse de la politique d'allocation optimiste.

Pour conclure, la probabilité que le problème de surallocation apparaisse, lorsqu'un contrôleur optimiste est utilisé, est proportionnelle à la durée moyenne des déploiements de nuages. Ce problème entraîne un certain nombre d'annulations de nuages non prévisibles, ce qui est problématique pour le service d'allocation de noeuds proposé par la plateforme Salute. On notera néanmoins que, généralement, l'annulation d'un nuage permet le maintien des autres collections de noeuds. Finalement, on pourrait penser qu'une des solutions pour résoudre le problème de surallocation pourrait consister à augmenter le nombre de pairs de réserve. Cette solution n'est pas considérée dans cette thèse, car d'une part, elle augmenterait le nombre de ressources inutilisées dans le système. En effet, on peut constater sur la trace donnée dans la Figure 7.15 que le nombre de pairs de réserve n'est que sporadiquement proche de zéro. D'autre part, il existe d'autres contrôleurs d'admission capables de fournir de bonnes performances lorsque la durée de déploiement des nuages est longue. Notre étude se focalise alors sur le développement de ces contrôleurs d'admission.

7.4.2 Evaluation du contrôleur d'admission local parfait

Afin d'analyser les performances du service d'allocation de noeuds lorsque la plateforme Salute utilise des contrôleurs locaux parfaits, les Figures 7.16 et 7.17 donnent deux traces représentant l'exécution d'une simulation allouant des nuages dont la durée de déploiement est de 8 heures pour la Figure 7.16, et de 4 jours pour la Figure 7.17.

Sur ces deux traces, on peut observer que le nombre de pairs alloués est strictement égal au nombre de pairs alloués avec succès, quelque soit la durée de déploiement des nuages demandés. Cette observation s'explique simplement par le fait que l'utilisation de prédictions de disponibilité parfaites assure que les noeuds alloués ne quittent pas le système, ni ne

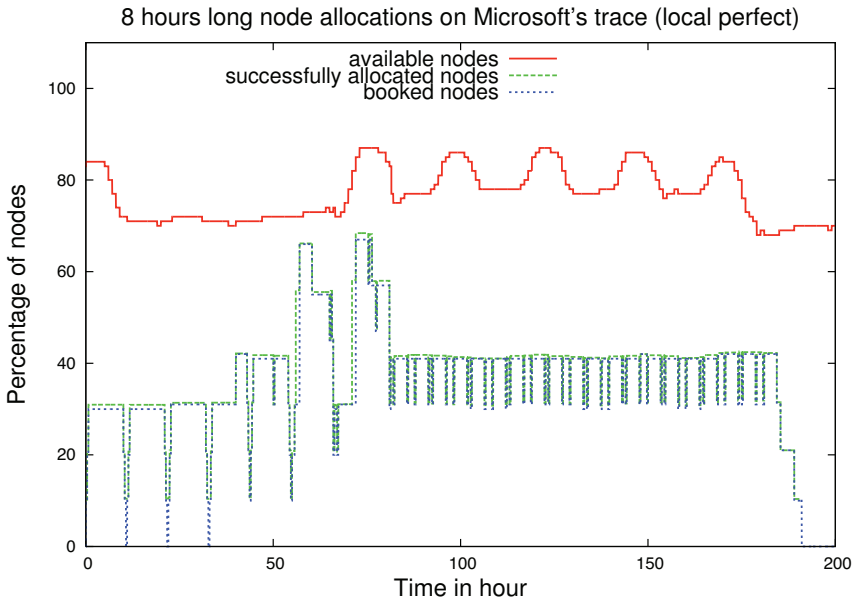


FIGURE 7.16 – Traces d'analyse des performances du contrôleur d'admission local parfait.

tombent en panne, avant la fin du déploiement des nuages demandés. Cependant, le nombre de tentatives de construction de nuage peut être important avant que l'une d'entre elles soit satisfaite. En effet, la contrainte imposée aux pairs libres pour rejoindre un nuage en construction est très difficile à satisfaire lorsque les contrôleurs d'admission locaux sont utilisés. Les pairs libres doivent en effet être disponibles pendant toute la durée de déploiement du nuage demandé. De fait, il est difficile pour un nuage en construction de recruter suffisamment de pairs pour atteindre sa taille de référence. Ceci explique pourquoi le service d'allocation de noeuds ne parvient à allouer, en moyenne, que la moitié des noeuds disponibles, lorsqu'il utilise des contrôleurs d'admission locaux parfaits.

Lorsque la durée de déploiement des nuages demandés augmente, la contrainte imposée aux noeuds devant être recrutés est d'autant plus difficile à satisfaire et le nombre d'allocations réussies diminue. Si on normalise le nombre de nuages alloués avec succès par le nombre de nuages annulés durant leur tentative de construction, les performances du contrôleur d'admission local parfait sont nettement inférieures à celles des autres contrôleurs. Néanmoins, on notera que l'utilisation de contrôleurs d'admission locaux permet de diminuer le churn expérimenté par les nuages pair-à-pair durant leur déploiement. En utilisant un contrôleur parfait, ce churn est nul, mais aucune technique de prédiction de disponibilité n'est parfaite en dehors d'un environnement de simulation. Finalement, en utilisant des techniques de prédiction de disponibilité locale, telles que celles proposées dans les études de Mickens et coll. [117] ou de Bhagwan et coll. [41], on peut s'attendre à ce que le churn expérimenté par les nuages pair-à-pair soit moins important que lorsque l'on utilise d'autres contrôleurs d'admission pour conduire leur allocation. Des perspectives de travail, visant à combiner l'utilisation d'oracles locaux et d'oracles globaux sont discutées dans le chapitre de conclusion de ce document.

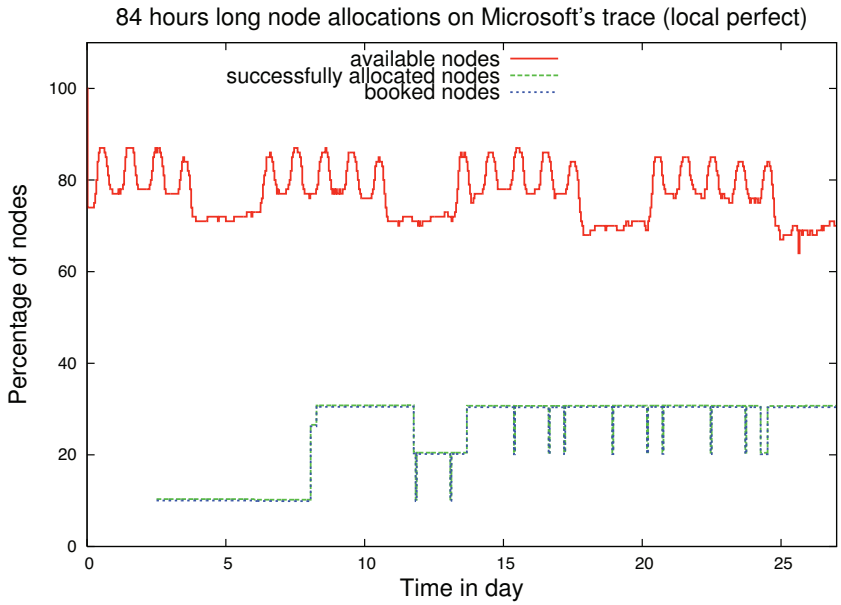


FIGURE 7.17 – Traces d'analyse des performances du contrôleur d'admission local parfait.

7.4.3 Evaluation du contrôleur d'admission à minimum strict

L'analyse des performances du contrôleur d'admission à minimum strict est basée sur les deux traces données dans les Figures 7.18 et 7.19. La trace de la Figure 7.18 représente l'exécution d'une simulation au sein de laquelle la durée de déploiement des nuages demandés est de 8 heures. La trace de la Figure 7.19 représente une simulation dans laquelle les nuages possèdent une durée de déploiement de 4 jours.

Comme on peut le constater sur ces deux traces, le nombre de noeuds alloués grâce au contrôleur d'admission à minimum strict est le même quelque soit la durée de déploiement des nuages demandés. Le contrôleur à minimum strict parvient à allouer un nombre constant de 6000 noeuds au sein d'un système contenant environ 500 pairs de réserve. Bien que cette performance puisse être sporadiquement dépassée par celle d'un contrôleur optimiste, la différence reste faible. La force du contrôleur d'admission à minimum strict est de fournir un service fiable, avec lequel aucun nuage n'est annulé, et dont les performances sont bonnes et stables indépendamment de la durée de déploiement des nuages demandés.

Pour être parfaitement équitable, on se doit de noter que 2 nuages ont été annulés à la fin de l'exécution représentée par la trace de la Figure 7.19. Cette annulation correspond à un événement catastrophique, aussi appelé churn massif dans la communauté des systèmes pair-à-pair. Aussi nous ne considérons pas dans cette thèse que ces annulations sont à imputer au comportement du contrôleur d'admission à minimum strict. En effet, si le service d'allocation de noeuds veut se prémunir contre les événements catastrophiques, qui sont rares par nature, alors il doit être configuré pour contenir une grande proportion de noeuds de réserve. Or comme expliqué précédemment dans ce chapitre, le nombre de ressources potentiellement alloué par la plateforme Salute décroît au fur et à mesure que le nombre de pairs de réserve

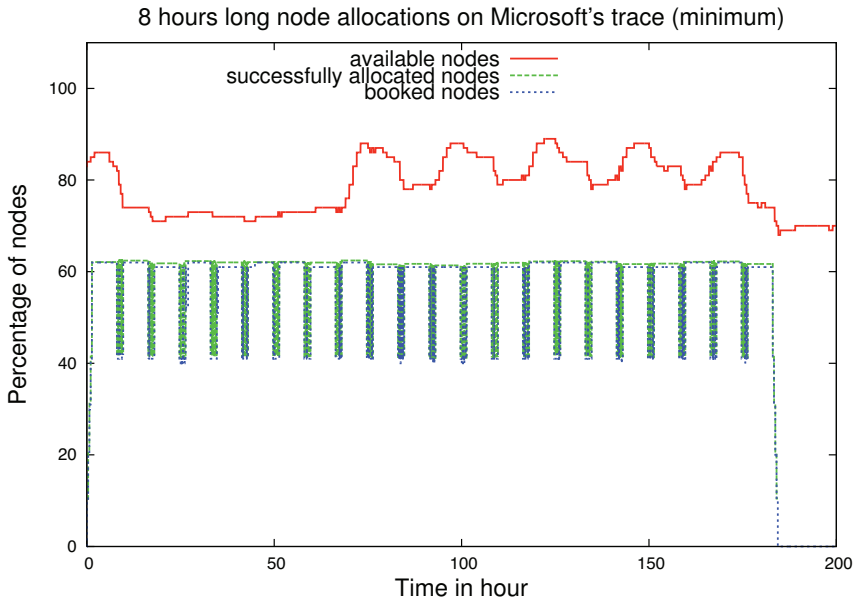


FIGURE 7.18 – Traces d'analyse des performances du contrôleur d'admission à minimum strict.

augmente. Il existe donc un compromis à réaliser entre la fiabilité du système face à des événements catastrophiques et ses performances moyennes en temps normal.

Discussion sur les événements catastrophiques : Une des caractéristiques des environnements pair-à-pair, qui n'est pas étudiée en détail dans cette thèse, est la présence d'événements catastrophiques. De tels événements sont appelés « churn massif » au sein de la communauté des systèmes pair-à-pair, ils correspondent, la plupart du temps, à une défaillance temporaire d'un appareil de routage d'Internet qui peut engendrer la déconnexion d'une grande partie des noeuds d'un système pair-à-pair. D'après les études connues à ce jour, un tel événement est très rare, et de façon générale : la taille des systèmes pair-à-pair varie très rarement de façon abrupte [68]. Aussi, bien que les travaux présentés dans cette thèse ne s'intéressent pas à optimiser le service d'allocation de noeuds proposé en fonction de ce type d'événement, nous considérons que la résistance de la plateforme Salute au churn massif reste correcte. Effectivement, il faut noter que l'utilisation de pairs de réserve permet à Salute de supporter naturellement une baisse temporaire de la taille du système, si celle-ci ne correspond pas à un événement catastrophique, c'est-à-dire si elle concerne un nombre de noeuds d'un ordre inférieur à 10% de la taille du système. En revanche, dans le cas où un événement catastrophique amène inévitablement à l'annulation de certains nuages pair-à-pair (le système ne comporte pas suffisamment de pairs de réserve pour supporter le churn massif), les mécanismes de gestion de concurrence employés au sein du protocole Sarcasm permettent de minimiser le nombre de nuages annulés. Dans un tel cas, les requêtes de secours sont déclenchées quasiment simultanément et connaissent donc un degré de concurrence important. Les mécanismes de priorités utilisés par Sarcasm permettent alors de sauver le déploiement

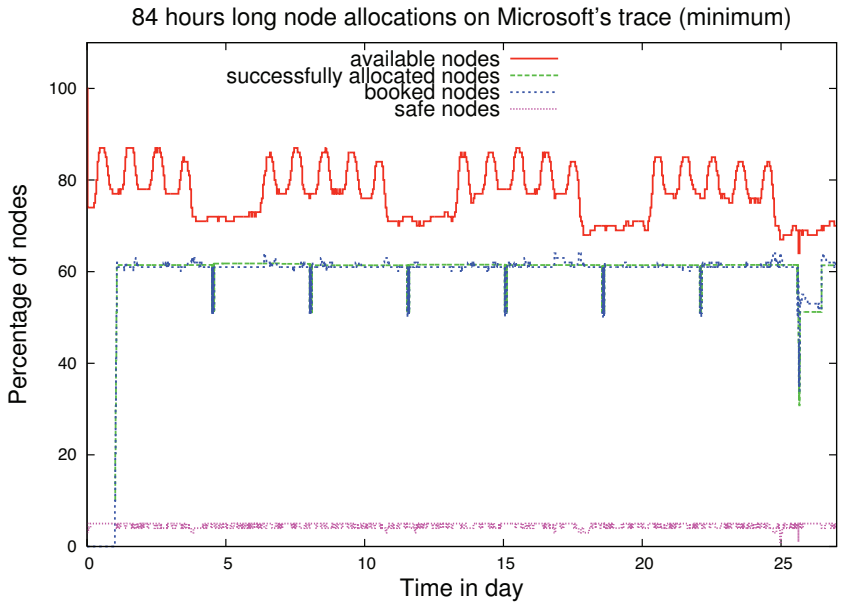


FIGURE 7.19 – Traces d'analyse des performances du contrôleur d'admission à minimum strict.

des nuages prioritaires au détriment d'autres nuages, ce qui d'après nous est une façon fiable de gérer les événements catastrophiques, lorsque cela est possible.

Conclusion

L'évaluation des contrôleurs d'admission montre que chacune de ces catégories présente des avantages et des inconvénients. Les contrôleurs optimistes sont faciles à implanter, mais leurs performances dépendent des conditions d'utilisation, c'est-à-dire du rythme auquel les requêtes sont soumises à la plateforme Salute et de la dynamique du système. Les contrôleurs d'admission locaux, eux, possèdent l'avantage de diminuer la dynamique des nuages déployés, mais ne permettent d'allouer qu'une faible partie des ressources disponibles. Finalement, les contrôleurs d'admission globaux, et en particulier les contrôleurs à minimum strict, possèdent de bonnes performances qui sont stables indépendamment de l'évolution du système.

Finalement, il nous tient à coeur de noter que l'étude des contrôleurs d'admission présentée dans cette thèse est limitée par le cadre expérimental, en particulier par la durée des traces utilisées pour reproduire la dynamique des environnements de référence. En effet, parmi les nombreux autres types de contrôleurs d'admission que l'on peut imaginer pour diriger l'ordonnancement des requêtes au sein de la plateforme Salute, nous avons étudié la possibilité d'utiliser des algorithmes « d'apprentissage par renforcement », pour définir un nouveau type de contrôleur d'admission. Le principe de ce nouveau type de contrôleur d'admission est d'observer l'évolution d'un environnement pair-à-pair pour apprendre empiriquement et automatiquement à diriger les allocations de noeuds. Cependant, ces techniques d'apprentissage nécessitent une grande quantité de données que nous n'avons pas pu collecter à temps pour la réalisation de cette thèse.

Conclusion

Bilan de la thèse

Dans cette thèse, nous nous sommes intéressés à l'informatique dans les nuages ou plutôt à son concept de distribution de ressources, qui possède la particularité de rentabiliser les ressources informatiques inutilisées en proposant de les allouer temporairement à des clients en ayant besoin. Bien que ce paradigme de distribution de ressources soit largement adopté par la communauté informatique, les moyens mis en oeuvre pour fournir cette distribution de ressources possèdent encore quelques limitations. En effet, les services de l'informatique dans les nuages sont aujourd'hui hébergés au sein de centres de traitement de données dont les ressources sont strictement utilisées sous la forme d'infrastructure en tant que service. Comme nous l'avons souligné dans cette thèse, ce mode de production d'infrastructure est coûteux et n'est pas adapté au déploiement de toutes les applications distribuées, en particulier celles tirant profit d'une large répartition géographique.

Devant ce constat, nous avons proposé l'élaboration d'une plateforme pair-à-pair : Salute, qui est capable de fournir un service d'allocation de noeuds qui utilise le même principe que l'infrastructure en tant que service, c'est-à-dire qui récolte des ressources informatiques inutilisées et les redistribue à des applications clientes. L'avantage principal de la plateforme Salute, par rapport aux centres de traitement de données, est que ses coûts de maintenance et d'administration sont significativement réduits et distribués sur un grand nombre de noeuds. De plus, l'architecture pair-à-pair utilisée pour héberger cette plateforme permet de recueillir un grand nombre de ressources hétérogènes, de provenances diverses et dont la disponibilité n'est pas contrôlée par le système. Le service d'allocation de Salute utilise une nouvelle abstraction de collection de noeuds appelée « nuage pair-à-pair ». Ces nuages sont composés d'un nombre stable de noeuds et possèdent une durée de déploiement précise. De fait, la plateforme Salute est capable d'ordonnancer le déploiement des nuages en fonction de la disponibilité des noeuds, ce qui en fait le premier système pair-à-pair, à nôtre connaissance, à savoir assigner plusieurs collections de noeuds à des applications ou services indépendants.

Finalement, parmi les contributions permettant l'implantation du service d'allocation de noeuds cités précédemment, on trouve le protocole Sarcasm qui est chargé de construire et de maintenir les nuages pair-à-pair en dépit de la dynamique de l'environnement hôte. Ce protocole possède notamment la particularité de garantir la construction d'au moins un nuage, lorsqu'il existe suffisamment de ressources, malgré un taux de concurrence très élevé des requêtes d'allocations. Il permet également d'appliquer des politiques garantissant l'équité des allocations de noeuds parmi un ensemble de clients. On trouve également au centre du service d'allocation de nuages des contrôleurs d'admission et des politiques d'allocation de noeuds qui permettent, à partir de l'observation du système, de garantir la fiabilité des déploiements de nuages pair-à-pair et d'optimiser le nombre de noeuds alloués au cours du temps. Ces contrôleurs d'admission sont implantés de façon décentralisée et permettent d'ordonnancer le

déploiement des nuages pair-à-pair sur un environnement fédéré, c'est-à-dire un environnement distribué qui est composé de ressources de provenances diverses et dont la disponibilité n'est pas contrôlée.

Perspectives

Cette section présente les pistes et les perspectives de travail que nous avons retenues lors de la réalisation du travail présenté dans cette thèse. Bien que, par manque de temps ou par manque de ressources, ces pistes ne sont pas développées davantage dans ce document, nous pensons qu'elles constituent des axes de recherche intéressants.

Collecter des traces plus longues et plus précises : Comme nous l'avons mentionné au sein de l'évaluation présentée dans le Chapitre 7, la limitation principale que nous avons rencontrées dans nos travaux étaient liées au manque de données concernant les caractéristiques des environnements pair-à-pair. Les données rassemblées par Brigthen Godfrey [68] nous ont néanmoins permis de réaliser plusieurs environnements de simulation réalistes pour la conception et pour l'étude du service d'allocation de noeuds de la plateforme Salute. A partir de ces environnements simulés, nous avons pu évaluer notre approche en validant le fonctionnement de l'architecture de la plateforme Salute, en validant le fonctionnement du protocole Sarcasm et en évaluant certaines politiques d'allocation de ressources simples. En possédant davantage de données sur les environnements pair-à-pair étudiés ici, en particulier des traces de disponibilité plus précises et s'étalant sur une période d'au moins un an, nous pensons qu'il est intéressant d'approfondir nos travaux en étudiant les points suivants :

- La possibilité de construire des contrôleurs d'admission intelligents qui sont basés sur l'utilisation d'algorithmes d'apprentissage par renforcement.
- L'implantation de contrôleurs d'admission locaux utilisant les techniques de prédiction existantes [41, 117, 28, 82], et leur impact sur le dynamisme des nuages alloués.
- La possibilité de combiner plusieurs types de contrôleur d'admission.

Contrôleurs d'admission intelligents : Une des perspectives de recherche temporairement mise de côté est la construction de contrôleurs d'admission globaux dits « intelligents », dont le principe est d'utiliser un algorithme d'apprentissage par renforcement tel que celui proposé par Liu et coll. [106]. Ce type d'algorithme est en effet capable de prédire l'évolution d'une série temporelle complexe, par exemple le cours de certaines actions boursières, et de majorer en temps réel l'erreur attribuée à chaque prédiction. La perspective d'utiliser cet algorithme pour concevoir un contrôleur d'admission global qui soit capable de prédire l'évolution de la taille d'un système pair-à-pair nous a paru être une piste intéressante. Néanmoins, après certains tests, nous nous sommes aperçus que la quantité de données nécessaire pour entraîner les algorithmes d'apprentissage par renforcement était bien plus importante que la quantité de données dont nous disposions au sein de nos environnements simulés. En effet, dans l'étude de Liu et coll. [106], les algorithmes d'apprentissage par renforcement sont entraînés avec un historique d'évolution de la série temporelle comportant 2500 points, sur une trace composée de 5000 points. Or les traces utilisées dans nos travaux ne comportent qu'environ 800 points, ce qui est insuffisant pour configurer un contrôleur d'admission intelligent potentiel. De façon

plus intuitive, on peut dire qu'un contrôleur d'admission basé sur l'apprentissage par renforcement aurait du mal à assimiler le concept de variation hebdomadaire sur une période totale de simulation de 28 jours.

Combinaisons de contrôleurs d'admission et réduction de churn : Comme nous l'avons montré dans cette thèse, les contrôleurs d'admission locaux, c'est-à-dire les contrôleurs utilisant des prédictions de disponibilité locales aux pairs qui composent le système, ne sont pas capables de gérer l'ordonnancement des allocations de nuages de façon performante. En effet, les contraintes appliquées par ce type de contrôleur d'admission semblent être trop strictes pour qu'une large proportion de noeuds puisse être allouée. En revanche, l'utilisation de ce type de contrôleur permet de réduire le dynamisme des nuages pair-à-pair alloués, puisque la contrainte de recrutement des noeuds leur impose d'être présent pendant toute la durée de déploiement du nuage demandé. Dans notre étude, nous avons utilisé des contrôleurs d'admission locaux ayant accès à des prédictions parfaites. Une des perspectives liées aux contrôleurs d'admission parfaits est d'implanter et d'évaluer le même type de contrôleur en utilisant les techniques de prédiction de l'état de l'art [41, 117, 28, 82]. D'après ces études, lorsqu'un noeud est assigné à une tâche en utilisent ces techniques de prédiction, la probabilité qu'il quitte le système ou qu'il tombe en panne avant la fin de cette tâche est très faible. Par conséquent, nous pensons que l'utilisation de ces méthodes de prédictions de disponibilité pourrait réduire le dynamisme des nuages déployés par la plateforme Salute. Finalement, étant donné que les contrôleurs d'admission locaux ne peuvent pas diriger l'ordonnancement des déploiements de nuages de façon fiable, la seconde piste à étudier est le couplage de contrôleurs d'admission locaux avec des contrôleurs globaux, afin de compenser le manque de performances ressenti lorsque les contrôleurs locaux sont utilisés.

Intégration d'un service de localisation de ressources : Nous avons expliqué dans cette thèse qu'il était possible de filtrer les noeuds à allouer au sein d'un nuage en définissant des types de pairs. Ce mécanisme de filtrage est destiné à montrer que l'on peut facilement construire, avec les protocoles présentés dans cette thèse, des nuages pair-à-pair qui contiennent plusieurs types de pairs. Néanmoins, il existe une problématique concernant cette caractéristique du service d'allocation de noeuds que nous n'avons pas détaillée dans ce document, et qui constitue selon nous une bonne perspective de recherche :

> Comment peut-on inclure la contrainte des types de pairs au sein des politiques d'allocations utilisées dans la plateforme Salute ? En effet, les contrôleurs d'admission étudiés ici ne tiennent pas compte des attributs des pairs, c'est-à-dire qu'ils n'estiment pas la disponibilité à long terme d'un type de pairs donné. De cette façon, si Salute parvient à allouer un nuage uniquement composé de pairs appartenant à un type A, le système n'est pas capable de garantir qu'il existera suffisamment de pair du type A, au cours du temps, pour maintenir la spécification du nuage demandé.

Implémentation d'un prototype de Sarcasm : Afin de conforter les résultats empiriques récoltés lors de nos simulations, nous avons réalisé un prototype implantant une version simplifiée du protocole Sarcasm. Ce prototype à pour but d'évaluer la latence et la bande passante consommée par Sarcasm, dans des conditions de déploiement réelles. Une des perspectives de cette thèse est alors d'évaluer les performances de ce prototype sur la grille Grid5000 [12] et sur le système PlanetLab [36].

Bibliographie

[1] http ://www.ipoque.com/resources/internet-studies/internet-study-2008_2009.

[2] Amazon case studies. http ://aws.amazon.com/solutions/case-studies/.

[3] Amazon elastic compute cloud : Ec2. http ://aws.amazon.com/ec2/.

[4] Amazon web services. http ://aws.amazon.com/.

[5] Dropbox. http ://www.dropbox.com/.

[6] Eainstein@home project. http ://www.einsteinathome.org/.

[7] Elastra. http ://www.elastra.com/.

[8] Facebook. http ://www.facebook.com/.

[9] Google app engine. http ://code.google.com/intl/fr-FR/appengine/.

[10] Google apps. http ://www.google.com/apps/.

[11] Google code. http ://code.google.com/.

[12] Grid'5000. https ://www.grid5000.fr/.

[13] Heroku. http ://heroku.com/.

[14] Napster. http ://www.napster.com.

[15] Netsuite. http ://www.netsuite.com/.

[16] Rackspace. http ://www.rackspace.com/index.php.

[17] Rightscale. http ://www.rightscale.com/.

[18] Salesforce. https ://www.salesforce.com/.

[19] Seti@home project. http ://setiathome.berkeley.edu/.

[20] Windows azure. http ://www.microsoft.com/windowsazure/windowsazure/.

[21] Wuala. http ://www.wuala.com/.

[22] Youtube. http ://www.youtube.com/.

[23] Gartner's 2009 hype cycle special report. http ://www.gartner.com/, August 2009.

[24] André Allavena, Alan J. Demers, and John E. Hopcroft. Correctness of a gossip based membership protocol. In *PODC*, pages 292–301, 2005.

[25] David P. Anderson. Boinc : A system for public-resource computing and storage. In *GRID*, pages 4–10, 2004.

[26] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, 36(4) :335–371, 2004.

[27] A. Andrzejak, D. Kondo, and D.P. Anderson. Exploiting non-dedicated resources for cloud computing. In *Network Operations and Management Symposium (NOMS), 2010 IEEE*, pages 341–348. IEEE, 2010.

[28] Artur Andrzejak, Derrick Kondo, and David P. Anderson. Ensuring collective availability in volatile resource pools via forecasting. In *DSOM*, pages 149–161, 2008.

[29] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andy Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4) :50–58, 2010.

[30] John Arrasjid, Shridhar Deuskar, and Irfan Ahmad. Vmware virtual infrastructure tools and techniques. In *USENIX Annual Technical Conference*, 2007.

[31] Arutyun Avetisyan, Roy H. Campbell, Indranil Gupta, Michael T. Heath, Steven Y. Ko, Gregory R. Ganger, Michael A. Kozuch, David R. O'Hallaron, Marcel Kunze, Thomas T. Kwan, Kevin Lai, Martha Lyons, Dejan S. Milojicic, Hing Yan Lee, Yeng Chai Soh, Ng Kwang Ming, Jing-Yuan Luke, and Han Namgoong. Open cirrus : A global cloud computing testbed. *IEEE Computer*, 43(4) :35–43, 2010.

[32] Özalp Babaoglu, Toni Binci, Márk Jelasity, and Alberto Montresor. Firefly-inspired heartbeat synchronization in overlay networks. In *SASO*, pages 77–86, 2007.

[33] Özalp Babaoglu, Márk Jelasity, Anne-Marie Kermarrec, Alberto Montresor, and Maarten van Steen. Managing clouds : a case for a fresh look at large unreliable dynamic networks. *Operating Systems Review*, 40(3) :9–13, 2006.

[34] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Timothy L. Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *SOSP*, pages 164–177, 2003.

[35] Salman Baset and Henning Schulzrinne. An analysis of the skype peer-to-peer internet telephony protocol. In *INFOCOM*, 2006.

[36] Andy C. Bavier, Mic Bowman, Brent N. Chun, David E. Culler, Scott Karlin, Steve Muir, Larry L. Peterson, Timothy Roscoe, Tammo Spalink, and Mike Wawrzoniak. Operating systems support for planetary-scale network services. In *NSDI*, pages 253–266, 2004.

[37] Fabrice Bellard. Qemu, a fast and portable dynamic translator. In *USENIX Annual Technical Conference, FREENIX Track*, pages 41–46, 2005.

[38] David Bernstein, Erik Ludvigson, Krishna Sankar, Steve Diamond, and Monique Morrow. Blueprint for the intercloud - protocols and formats for cloud computing interoperability. In *ICIW*, pages 328–336, 2009.

[39] Philip A. Bernstein. Middleware : A model for distributed system services. *Commun. ACM*, 39(2) :86–98, 1996.

[40] Ranjita Bhagwan, Stefan Savage, and Geoffrey M. Voelker. Understanding availability. In *IPTPS*, pages 256–267, 2003.

[41] Ranjita Bhagwan, Kiran Tati, Yu-Chung Cheng, Stefan Savage, and Geoffrey M. Voelker. Total recall : System support for automated availability management. In *NSDI*, pages 337–350, 2004.

[42] Kenneth P. Birman, Mark Hayden, Öznur Özkasap, Zhen Xiao, Mihai Budiu, and Yaron Minsky. Bimodal multicast. *ACM Trans. Comput. Syst.*, 17(2) :41–88, 1999.

[43] William J. Bolosky, John R. Douceur, David Ely, and Marvin Theimer. Feasibility of a serverless distributed file system deployed on an existing set of desktop pcs. In *SIGMETRICS*, pages 34–43, 2000.

[44] R. Buyya, D. Abramson, and S. Venugopal. The grid economy. *Proceedings of the IEEE*, 93(3) :698–714, 2005.

[45] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms : Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Comp. Syst.*, 25(6) :599–616, 2009.

[46] Miguel Castro, Manuel Costa, and Antony I. T. Rowstron. Performance and dependability of structured peer-to-peer overlays. In *DSN*, pages 9–18, 2004.

[47] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony I. T. Rowstron, and Atul Singh. Splitstream : high-bandwidth multicast in cooperative environments. In *SOSP*, pages 298–313, 2003.

[48] K. Church, A. Greenberg, and J. Hamilton. On delivering embarrassingly distributed cloud services. *Hotnets VII*, 2008.

[49] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet : A distributed anonymous information storage and retrieval system. In *Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66, 2000.

[50] Bram Cohen. The official specification of the bittorrent protocol. http ://www.bittorrent.org/beps/bep_0003.html, 2008.

[51] R. Cox, A. Muthitacharoen, and R. Morris. Serving dns using a peer-to-peer lookup service. *Peer-to-Peer Systems*, pages 155–165, 2002.

[52] Mache Creeger. Cto roundtable : cloud computing. *Commun. ACM*, 52(8) :50–56, 2009.

[53] Flaviu Cristian. Probabilistic clock synchronization. *Distributed Computing*, 3(3) :146–158, 1989.

[54] F. Dabek, M.F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with cfs. In *Proceedings of the eighteenth ACM symposium on Operating systems principles*, page 215. ACM, 2001.

[55] Abhinandan Das, Indranil Gupta, and Ashish Motivala. Swim : Scalable weakly-consistent infection-style process group membership protocol. In *DSN*, pages 303–312, 2002.

[56] Jeffrey Dean and Sanjay Ghemawat. Mapreduce : Simplified data processing on large clusters. In *OSDI*, pages 137–150, 2004.

[57] Alan J. Demers, Daniel H. Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard E. Sturgis, Daniel C. Swinehart, and Douglas B. Terry. Epidemic algorithms for replicated database maintenance. In *PODC*, pages 1–12, 1987.

[58] John R. Douceur. Is remote host availability governed by a universal law ? *SIGMETRICS Performance Evaluation Review*, 31(3) :25–29, 2003.

[59] Peter Druschel and Antony I. T. Rowstron. Past : A large-scale, persistent peer-to-peer storage utility. In *HotOS*, pages 75–80, 2001.

[60] Patrick Th. Eugster, Rachid Guerraoui, Sidath B. Handurukande, Petr Kouznetsov, and Anne-Marie Kermarrec. Lightweight probabilistic broadcast. *ACM Trans. Comput. Syst.*, 21(4) :341–374, 2003.

[61] Patrick Th. Eugster, Rachid Guerraoui, Anne-Marie Kermarrec, and Laurent Massoulié. Epidemic information dissemination in distributed systems. *IEEE Computer*, 37(5) :60–67, 2004.

[62] Antonio Fernández, Vincent Gramoli, Ernesto Jiménez, Anne-Marie Kermarrec, and Michel Raynal. Distributed slicing in dynamic systems. In *ICDCS*, page 66, 2007.

[63] I. Foster and C. Kesselman. *The grid : blueprint for a new computing infrastructure*. Morgan Kaufmann, 1999.

[64] Ian T. Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid : Enabling scalable virtual organizations. *IJHPCA*, 15(3) :200–222, 2001.

[65] Yun Fu, Jeffrey S. Chase, Brent N. Chun, Stephen Schwab, and Amin Vahdat. Sharp : an architecture for secure resource peering. In *SOSP*, pages 133–148, 2003.

[66] Simson L. Garfinkel. Technical report tr-08-07 : An evaluation of amazon's grid computing services : Ec2, s3 and sqs. Technical report, Harvard, 2007.

[67] Jeremy Geelan. Twenty one experts define cloud computing. Electronic Magazine, article available at http ://virtualization.sys-con.com/node/612375, August 2008.

[68] Brighten Godfrey. http ://www.cs.illinois.edu/ pbg/availability/.

[69] Luis Miguel Vaquero Gonzalez, Luis Rodero Merino, Juan Caceres, and Maik Lindner. A break in the clouds : towards a cloud definition. *Computer Communication Review*, 39(1) :50–55, 2009.

[70] Vincent Gramoli, Ymir Vigfusson, Ken Birman, Anne-Marie Kermarrec, and Robbert van Renesse. A fast distributed slicing algorithm. In *PODC*, page 427, 2008.

[71] Albert G. Greenberg, James R. Hamilton, David A. Maltz, and Parveen Patel. The cost of a cloud : research problems in data center networks. *Computer Communication Review*, 39(1) :68–73, 2009.

[72] Galen Gruman and Eric Knorr. What cloud computing really mean. InfoWorld, Electronic Magazine available at http ://www.infoworld.com/d/cloud-computing/what-cloud-computing-really-means-031, April 2008.

[73] Saikat Guha, Neil Daswani, and Ravi Jain. An Experimental Study of the Skype Peer-to-Peer VoIP System. In *Proceedings of The 5th International Workshop on Peer-to-Peer Systems (IPTPS '06)*, Santa Barbara, CA, February 2006.

[74] P. Krishna Gummadi, Stefan Saroiu, and Steven D. Gribble. King : estimating latency between arbitrary internet end hosts. *Computer Communication Review*, 32(3) :11, 2002.

[75] Katherine Guo, Mark Hayden, Robbert Van Renesse, Werner Vogels, and Kenneth P. Birman. Gsgc : An efficient gossip-style garbage collection scheme for scalable reliable multicast. Technical report, Cornell University, NY, USA, 1997.

[76] Maxim Gurevich and Idit Keidar. Correctness of gossip-based membership under message loss. In *PODC*, pages 151–160, 2009.

[77] C. Huang, J. Li, and K. Ross. Peer-assisted vod : Making internet video distribution cheap. In *Proc. of IPTPS*, volume 7. Citeseer, 2007.

[78] RightScale Inc. The skinny on cloud lock-in. http ://blog.rightscale.com/2009/02/19/the-skinny-on-cloud-lock-in/, 2009.

[79] David E. Irwin, Jeffrey S. Chase, Laura E. Grit, Aydan R. Yumerefendi, David Becker, and Ken Yocum. Sharing networked resources with brokered leases. In *USENIX Annual Technical Conference, General Track*, pages 199–212, 2006.

[80] Konrad Iwanicki, Maarten van Steen, and Spyros Voulgaris. Gossip-based clock synchronization for large decentralized systems. In *SelfMan*, pages 28–42, 2006.

[81] Sitaram Iyer, Antony I. T. Rowstron, and Peter Druschel. Squirrel : a decentralized peer-to-peer web cache. In *PODC*, pages 213–222, 2002.

[82] Bahman Javadi, Derrick Kondo, Jean-Marc Vincent, and David P. Anderson. Mining for statistical models of availability in large-scale distributed systems : An empirical study of seti@home. In *MASCOTS*, pages 1–10, 2009.

[83] Márk Jelasity and Özalp Babaoglu. T-man : Gossip-based overlay topology management. In *Engineering Self-Organising Systems*, pages 1–15, 2005.

[84] Márk Jelasity, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten van Steen. The peer sampling service : Experimental evaluation of unstructured gossip-based implementations. In *Middleware*, pages 79–98, 2004.

[85] Márk Jelasity and Anne-Marie Kermarrec. Ordered slicing of very large-scale overlay networks. In *Peer-to-Peer Computing*, pages 117–124, 2006.

[86] Márk Jelasity and Alberto Montresor. Epidemic-style proactive aggregation in large overlay networks. In *ICDCS*, pages 102–109, 2004.

[87] Márk Jelasity, Alberto Montresor, and Özalp Babaoglu. The bootstrapping service. In *ICDCS Workshops*, page 11, 2006.

[88] Márk Jelasity, Alberto Montresor, and Özalp Babaoglu. T-man : Gossip-based fast overlay topology construction. *Computer Networks*, 53(13) :2321–2339, 2009.

[89] Márk Jelasity, Alberto Montresor, Gian Paolo Jesi, and Spyros Voulgaris. The Peersim simulator. `http://peersim.sf.net`.

[90] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin. Consistent hashing and random trees : Distributed caching protocols for relieving hot spots on the world wide web. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 654–663. ACM, 1997.

[91] Katarzyna Keahey, Ian T. Foster, Timothy Freeman, Xuehai Zhang, and Daniel Galron. Virtual workspaces in the grid. In *Euro-Par*, pages 421–431, 2005.

[92] Katarzyna Keahey, Maurício O. Tsugawa, Andréa M. Matsunaga, and José A. B. Fortes. Sky computing. *IEEE Internet Computing*, 13(5) :43–51, 2009.

[93] David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *FOCS*, pages 482–491, 2003.

[94] Anne-Marie Kermarrec, Laurent Massoulié, and Ayalvadi J. Ganesh. Probabilistic reliable dissemination in large-scale systems. *IEEE Trans. Parallel Distrib. Syst.*, 14(3) :248–258, 2003.

[95] Anne-Marie Kermarrec, Alessio Pace, Vivien Quéma, and Valerio Schiavoni. Nat-resilient gossip peer sampling. In *ICDCS*, pages 360–367, 2009.

[96] Anne-Marie Kermarrec and Maarten van Steen. Gossiping in distributed systems. *Operating Systems Review*, 41(5) :2–7, 2007.

[97] Dionysios Kostoulas, Dimitrios Psaltoulis, Indranil Gupta, Ken Birman, and Alan J. Demers. Decentralized schemes for size estimation in large and dynamic groups. In *NCA*, pages 41–48, 2005.

[98] John Kubiatowicz, David Bindel, Yan Chen, Steven E. Czerwinski, Patrick R. Eaton, Dennis Geels, Ramakrishna Gummadi, Sean C. Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Y. Zhao. Oceanstore : An architecture for global-scale persistent storage. In *ASPLOS*, pages 190–201, 2000.

[99] Kendy Kutzner and Thomas Fuhrmann. Measuring large overlay networks - the overnet example. In *KiVS*, pages 193–204, 2005.

[100] Kevin Lai, Lars Rasmusson, Eytan Adar, Li Zhang, and Bernardo A. Huberman. Tycoon : An implementation of a distributed, market-based resource allocation system. *Multiagent and Grid Systems*, 1(3) :169–182, 2005.

[101] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7) :558–565, 1978.

[102] Jonathan Ledlie, Peter R. Pietzuch, and Margo I. Seltzer. Stable and accurate network coordinates. In *ICDCS*, page 74, 2006.

[103] Xiaozhou (Steve) Li and C. Greg Plaxton. On name resolution in peer-to-peer networks. In *POMC*, pages 82–89, 2002.

[104] Jian Liang, Rakesh Kumar, and Keith W. Ross. The fasttrack overlay : A measurement study. *Computer Networks*, 50(6) :842–858, 2006.

[105] Michael J. Litzkow, Miron Livny, and Matt W. Mutka. Condor - a hunter of idle workstations. In *ICDCS*, pages 104–111, 1988.

[106] F. Liu, Geok See Ng, and Chai Quek. Rldde : A novel reinforcement learning-based dimension and delay estimator for neural networks in time series prediction. *Neurocomputing*, 70(7-9) :1331–1341, 2007.

[107] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials*, 7(1-4) :72–93, 2005.

[108] Dahlia Malkhi, Moni Naor, and David Ratajczak. Viceroy : a scalable and dynamic emulation of the butterfly. In *PODC*, pages 183–192, 2002.

[109] Keahey K. Freeman T. Marshall, P. Elastic site : Using clouds to elastically extend site resources. In *CCGrid*, 2010.

[110] Laurent Massoulié, Erwan Le Merrer, Anne-Marie Kermarrec, and Ayalvadi J. Ganesh. Peer counting and sampling in overlay networks : random walk methods. In *PODC*, pages 123–132, 2006.

[111] Makoto Matsumoto and Takuji Nishimura. Mersenne twister : A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8(1) :3–30, 1998.

[112] E. Michael Maximilien, Ajith Ranabahu, Roy Engehausen, and Laura C. Anderson. Ibm altocumulus : a cross-cloud middleware and platform. In *OOPSLA Companion*, pages 805–806, 2009.

[113] E. Michael Maximilien, Ajith Ranabahu, Roy Engehausen, and Laura C. Anderson. Toward cloud-agnostic middlewares. In *OOPSLA Companion*, pages 619–626, 2009.

[114] Petar Maymounkov and David Mazières. Kademlia : A peer-to-peer information system based on the xor metric. In *IPTPS*, pages 53–65, 2002.

[115] Paul McFedries. The cloud is the computer. IEEE Spectrum Online, Electronic Magazine available at http ://spectrum.ieee.org/computing/hardware/the-cloud-is-the-computer, August 2008.

[116] Erwan Le Merrer, Anne-Marie Kermarrec, and Laurent Massoulié. Peer to peer size estimation in large and dynamic networks : A comparative study. In *HPDC*, pages 7–17, 2006.

[117] James W. Mickens and Brian D. Noble. Exploiting availability prediction in distributed systems. In *NSDI*, 2006.

[118] D Mills. Network time protocol (version 3) specification, implementation and analysis. Technical Report RFC 1305, March 1992, 1992.

[119] D. Mills, J. Martin, J. Burbank, and W. Kasch. Network time protocol version 4 : Protocol and algorithms specification (rfc 5905). RFC 5905 (Proposed Standard), 2010.

[120] David L. Mills. Improved algorithms for synchronizing computer network clocks. *IEEE/ACM Trans. Netw.*, 3(3) :245–254, 1995.

[121] Alberto Montresor and Ali Ghodsi. Towards robust peer counting. In *Peer-to-Peer Computing*, pages 143–146, 2009.

[122] Alberto Montresor, Márk Jelasity, and Özalp Babaoglu. Robust aggregation protocols for large-scale overlay networks. In *DSN*, pages 19–28, 2004.

[123] Alberto Montresor, Márk Jelasity, and Özalp Babaoglu. Chord on demand. In *Peer-to-Peer Computing*, pages 87–94, 2005.

[124] Alberto Montresor and Roberto Zandonati. Absolute slicing in peer-to-peer systems. In *IPDPS*, pages 1–8, 2008.

[125] NIST. Nist definition of cloud computing v15. http ://csrc.nist.gov/groups/SNS/cloud-computing/, May 2010.

[126] Daniel Nurmi, Richard Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, and Dmitrii Zagorodnov. The eucalyptus open-source cloud-computing system. In *CCGRID*, pages 124–131, 2009.

[127] Members of EGEE-II. An egee comparative study : Grids and clouds - evolution or revolution. Technical report, Enabling Grids for E-SciencE projects, available at https ://edms.cern.ch/document/925013/, 2008.

[128] D. Owen. Securing elasticity in the cloud. *Communications of the ACM*, 53(6) :46–51, 2010.

[129] D.F. Parkhill. *The Challenge of the Computer Utility.* Addison-Wesley Educational Publishers Inc., US, 1966.

[130] C. Greg Plaxton, Rajmohan Rajaraman, and Andréa W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *SPAA*, pages 311–320, 1997.

[131] K. Rangan. The cloud wars : $100+ billion at stake. Technical report, Merrill Lynch, May 2008.

[132] Sridharan Ranganathan, Alan D. George, Robert W. Todd, and Matthew C. Chidester. Gossip-style failure detection and distributed consensus for scalable heterogeneous clusters. *Cluster Computing*, 4(3) :197–209, 2001.

[133] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard M. Karp, and Scott Shenker. A scalable content-addressable network. In *SIGCOMM*, pages 161–172, 2001.

[134] Robbert Van Renesse, Yaron Minsky, and Mark Hayden. A gossip-style failure detection service. In *Service," Proc. Conf. Middleware*, pages 55–70, 1996.

[135] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud : exploring information leakage in third-party compute clouds. In *ACM Conference on Computer and Communications Security*, pages 199–212, 2009.

[136] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, et al. The reservoir model and architecture for open federated cloud computing. *IBM Systems Journa*, 53(4), 2009.

[137] Rodrigo Rodrigues and Peter Druschel. Peer-to-peer systems. *Commun. ACM*, 53(10) :72–82, 2010.

[138] Antony I. T. Rowstron and Peter Druschel. Pastry : Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware*, pages 329–350, 2001.

[139] Antony I. T. Rowstron, Anne-Marie Kermarrec, Miguel Castro, and Peter Druschel. Scribe : The design of a large-scale event notification infrastructure. In *Networked Group Communication*, pages 30–43, 2001.

[140] S.H. Standard. Sha-1. fips publication 180-1. *Apr*, 17 :1–24, 1995.

[141] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord : a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.*, 11(1) :17–32, 2003.

[142] J. Stribling. All-pairs planetlab ping data. http ://www.pdos.lcs.mit.edu/~strib/pl_app/.

[143] Gunther Stuer, Kurt Vanmechelen, and Jan Broeckhove. A commodity market algorithm for pricing substitutable grid resources. *Future Generation Comp. Syst.*, 23(5) :688–701, 2007.

[144] Daniel Stutzbach and Reza Rejaie. Characterizing the two-tier gnutella topology. In *SIGMETRICS*, pages 402–403, 2005.

[145] Daniel Stutzbach and Reza Rejaie. Understanding churn in peer-to-peer networks. In *Internet Measurement Conference*, pages 189–202, 2006.

[146] R. Sweha, V. Ishakian, and A. Bestavros. Angels in the cloud : A peer-assited bulk-synchronous content distribution service. Technical report, Boston University, Computer science department., 2010.

[147] K.K. Ramakrishnan Prashant Shenoy Jacobus van der Merwe Timothy Wood, Emmanuel Cecchet and Arun Venkataramani. Disaster recovery as a cloud service : Economic benefits and deployment challenges. In *Proceedings of the 2nd Workshop on Hot Topics in Cloud Computing. HotCloud 2010*, 2010.

[148] László Toka, Matteo Dell'Amico, and Pietro Michiardi. Online data backup : A peer-assisted approach. In *Peer-to-Peer Computing*, 2010.

[149] Norbert Tölgyesi and Márk Jelasity. Adaptive peer sampling with newscast. In *Euro-Par*, pages 523–534, 2009.

[150] Mark Turner, David Budgen, and Pearl Brereton. Turning software into a service. *IEEE Computer*, 36(10) :38–44, 2003.

[151] T. Velte, A. Velte, and R. Elsenpeter. *Cloud Computing : A Practical Approach*. McGraw-Hill Osborne Media, 2009.

[152] S. Voulgaris. *Epidemic-Based Self-Organization in Peer-to-Peer Systems*. PhD thesis, Vrije Universiteit in Amsterdam, 2006.

[153] Spyros Voulgaris, Daniela Gavidia, and Maarten van Steen. Cyclon : Inexpensive membership management for unstructured p2p overlays. *J. Network Syst. Manage.*, 13(2), 2005.

[154] Spyros Voulgaris and Maarten van Steen. Epidemic-style management of semantic overlays for content-based searching. In *Euro-Par*, 2005.

[155] Spyros Voulgaris, Maarten van Steen, and Konrad Iwanicki. Proactive gossip-based management of semantic overlay networks. *Concurrency and Computation : Practice and Experience*, 19(17) :2299–2311, 2007.

[156] A. Weiss. Computing in the clouds. *ACM. NetWorker*, 11(4) :16–25, 2007.

[157] Jia Yang, Hao Ma, Weijia Song, Jian Cui, and Changling Zhou. Crawling the edonkey network. In *GCC Workshops*, pages 133–136, 2006.

[158] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John Kubiatowicz. Tapestry : a resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1) :41–53, 2004.