GRAPP&S, une Architecture Multi-Échelle pour les Données et le Services

Thierno Ahmadou Diallo†‡
thierno.diallo.sn@gmail.com

Samba N'Diaye†
samba.ndiaye@ucad.edu.sn

Olivier Flauzac‡
olivier.flauzac@univ-reims.fr

Luiz-Angelo Steffenel‡
luiz-angelo.steffenel@univ-reims.fr

† Département d'Informatique, LMI, Université Cheikh Anta Diop, 5005 Dakar-Fann, Sénégal
‡ Laboratoire CReSTIC - Équipe SYSCOM, Université de Reims Champagne-Ardenne, BP 1039, 51687 Reims Cedex 2, France

RÉSUMÉ

Cet article présente GrAPP&S (Grid APPlication & Services), une spécification d'une architecture multi-échelle pour le stockage et l'indexation unifiée de différents formats de données tels que les fichiers, les flux de données, les requêtes sur les bases de données mais aussi l'accès à des services distants (des web services sur un serveur ou le cloud, ou des tâches de calcul dans un cluster HPC). GrAPP&S offre et orchestre une architecture hiérarchique de routage pour accéder aux différents types de données, et permet l'interconnexion de différentes communautés (réseaux locaux) grâce aux principes des systèmes multi-échelle. Les données sont présentées de façon transparente à l'utilisateur par l'intermédiaire de proxies spécifiques à chaque type de donnée. La transparence offerte par les proxies porte sur la localisation des sources de données, le traitement des requêtes, la composition des résultats mais aussi la gestion de la consistance des données. De plus, l'architecture de GRAPP&S a été conçue pour permettre l'application de politiques de sécurité et confidentialité, tant à l'intérieur d'une communauté qu'entre les communautés.

ABSTRACT

In this paper we introduce GRAPP&S (Grid APPlication & Services), a multi-scale framework for an unified storage and indexing of both data and services. Indeed, GRAPP&S was designed to support different data formats such as files, stream or database queries, but also to support the access to distant services (web services, cloud services or HPC computing services, for example). GRAPP&S coordinates an hierarchical routing architecture that allows data indexing and access, thanks to a multi-scale network of local-area communities. Transparent access is provided by a network of specialized proxies, which handle most aspects related to data location, request handling, data preprocessing/summary and also data consistence. Furthermore, GRAPP&S infrastructure can be easily enhanced to ensure security and confidentiality inside a GRAPP&S community and among different communities.

UbiMob 2013, 5–6 juin 2013, Nancy, France.

1. INTRODUCTION

La gestion de données à grande échelle est un problème récurrent autant dans les domaines scientifiques que dans le monde de l'entreprise. Malgré les constantes avancées en matière de capacité des mémoires et disques, l'utilisation d'un seul dispositif de stockage n'est plus une option vu que l'accès concurrent, la fiabilité, la consommation énergétique et le coût sont des obstacles au développement des systèmes. C'est pour cette raison que les chercheurs et développeurs se sont tournés depuis longtemps vers le développement de solutions de stockage distribué, afin de contourner ces limitations.

Dans les cas où les données peuvent être représentés sous la forme de fichiers, les solutions de type NAS/SAN, réseaux P2P et aussi le stockage en nuage (*clouds*) représentent des choix technologiques capables d'offrir un stockage à grande échelle pour un coût raisonnable. Ces choix concernent aussi les bases de données de type relationnelle ou NoSQL, mais dans ce cas l'accès aux données requiert toujours une entité (pseudo)centralisée capable d'agréger et de présenter les données (cela n'exclut pas le traitement parallèle des requêtes).

À travers différentes stratégies, ces solutions distribuées proposent des solutions transparentes à l'augmentation des besoins de stockage, tout en offrant suffisamment de garanties pour assurer la consistance et la pérennité des données. Aujourd'hui, l'utilisation de solutions de stockage de fichiers sur NAS/SAN ou cloud est devenue aussi courante que l'utilisation de disques ou clés USB, une fois que les ressources potentiellement illimités offerts par les solutions P2P ou cloud présentent plusieurs avantages en ce qui concerne le coût, la disponibilité et l'utilisation des ressources physiques. Cependant, ces solutions peuvent aussi présenter des inconvénients liés à la vitesse d'accès et à la sécurité des données ; la solution à ces inconvénients est encore loin d'être garantie et dépend majoritairement des solutions propriétaires propo-

sées par les fournisseurs des services de stockage. Un autre aspect à considérer est la compatibilité entre les systèmes : si certaines APIs rendent la manipulation des fichiers relativement simple, il est moins évident l'intégration d'autres représentations de données telles que les requêtes sur une base de données, des flux de données ou encore l'exécution de services.

C'est dans le but de proposer une architecture unifiée pour les données et les services que nous présentons GRAPP&S (GRid Applications and Services), une architecture multi-échelle pour l'agrégation de données et services. Ce framework a été conçu de manière à intégrer de manière transparente les données de type fichier mais aussi les bases de données, les flux (audio, vidéo), les services Web et le calcul distribué. À travers une structuration hiérarchique basée autour du concept de "communautés", GRAPP&S permet l'intégration de sources de d'information disposant de protocoles d'accès hétérogènes et des règles de sécurité variés.

D'autres contributions de GRAPP&S sont la mise en place d'une spécification détachée du middleware de communication (P2P ou autre), et la définition d'une solution de stockage générique. Dans le premier cas, ceci est possible car la spécification de GRAPP&S est basée sur des propriétés telles que la gestion de la connexité de la communauté et la gestion du routage entre les nœuds. Dans le deuxième cas, l'utilisation de nœuds "proxy data" permet d'unifier l'accès à des ressources variées telles que des fichiers, des flux, des services (FTP, mail) ou des données créées à la volée par des tâches de calcul ou des capteurs.

Enfin, l'accès aux données n'est pas dépendant d'une interconnexion directe entre les nœuds, comme c'est le cas de la plupart des réseaux P2P. Dans GRAPP&S, il est toujours possible de router le transfert des données par le chemin utilisé lors de la recherche, au cas où une connexion directe entre les nœuds n'est pas possible.

La suite de cet article est organisée comme suit : la section 2 introduit brièvement l'état de l'art concernant les systèmes multi-échelle et les réseaux P2P. La section 3 présente les principaux éléments qui constituent l'architecture GRAPP&S et comment ces éléments s'interconnectent, alors que la section 4 détaille les principales opérations liées à la gestion et à la recherche d'informations dans GRAPP&S. La section 5 détaille l'état actuel du développement de GRA PP&S et présente le module GAÏA destiné à l'optimisation du stockage multi-échelle. Finalement, la section 6 propose nos conclusions.

2. ÉTAT DE L'ART

2.1 Systèmes Multi-Échelle

D'une manière générale, les systèmes multi-échelle sont donc structurés autour de services déployés sur plusieurs niveaux et qui se complètement afin de répondre aux besoins plus ou moins immédiats des clients. Rottenberg *et al* [14] formalisent cette définition sur la forme :

> Un système multi-échelle est un système réparti sur plusieurs niveaux de tailles différentes dans une ou plusieurs dimensions (équipement, réseau, géographie, etc.)

Satyanarayanan présente un exemple de système multi-échelle dans ses travaux autour des "cloudlets" [16, 17]. Dans ces travaux, Satyanarayanan se penchent sur les limitations des équipements mobiles et sur l'inadéquation des solutions actuellement mises en place pour l'externalisation des services mobiles (notamment le transfert de ces services sur le *cloud computing*). La solution proposée est la mise en place de serveurs de proximité sans état et connectés à Internet (*cloudlets*) auxquels les équipements mobiles peuvent se connecter directement via un réseau Wi-Fi. Ces équipements sont déployés comme des hotspots Wi-Fi dans des cafés, magasins etc. Ils possèdent une forte puissance de calcul et permettent aux équipements mobiles d'externaliser les calculs trop complexes sur une machine proche, ce qui résout les problèmes de latence du cloud computing.

De tels systèmes ont forcément des besoins de coordination très spécifiques. Ainsi, l'impact de l'hétérogénéité des technologies impliquées dans un système multi-échelle est étudié par Blair et Grace [3] alors que Kessis et al. [9] analyse le caractère malléable de ce systèmes, capables de changer d'architecture et de granularité au cours de l'exécution. En effet, les systèmes multi-échelle requièrent un système de gestion flexible et adaptable capable de contrer la complexité et la volatilité de ces systèmes.

De même, Rottenberg *et al.* [14] cite la gestion de la confidentialité comme l'une des caractéristiques importantes des systèmes multi-échelle. À gestion fine de la confidentialité à chaque niveau de l'échelle vient s'ajouter au besoin d'une couche intergicielle dédiée à la sécurité de l'ensemble d'un réseau [6], tout en masquant l'hétérogénéité technique des systèmes multi-échelle.

2.2 Réseaux P2P

Différentes architectures hiérarchique P2P ont été proposée dans [12, 8, 7, 13]. Dans [8], l'auteur propose une architecture hiérarchique à deux niveaux. Au niveau inférieur il regroupe les pairs d'une même région, organisés sur un anneau de Chord et coordonnés par un super nœud. Au niveau supérieur se trouvent les super-nœuds de chaque région. [8] propose un algorithme de recherche régionale basé sur les super-nœuds, qui gardent une table de routage bidirectionnelle dans le but de réduire efficacement la redondance de la table de routage originale Chord.

Toutefois, même si l'architecture passe à l'échelle, elle ne résiste pas dans un environnement dynamique. Dans [7] les auteurs ont proposé un modèle hiérarchique de DHT (HDHT), où les pairs sont organisés en groupes. L'objectif des HDHTs est d'améliorer l'architecture plane DHT conventionnelle, par une exploitation des ressources hétérogènes des pairs, la mise en cache des infrastructures, la transparence et l'autonomie de différentes parties du système, afin de rendre la recherche de clés plus efficace tout en produisant moins de trafic.

Les auteurs de [13] proposent l'architecture hiérarchique CBT pour améliorer le protocole de téléchargement de fichier de Bittorrent dans un réseau de grande échelle. Il utilise trois types de pairs : les sources, les téléchargeurs et les super pair. Tous les supers pairs sont connectés au torrent tracker (gérant de l'index des ressources) pour former un réseau dédié.

La disponibilité des informations est fortement liée au torrent tracker, car si ce dernier tombe en panne tout le service disparait.

Les travaux effectués dans [8, 7, 13] reposent sur une architecture hiérarchique à deux niveau. Leur objectif est d'améliorer efficacement les performances tels que la latence lors d'une recherche et générer moins de trafic réseau. Cependant, ces architectures montrent leur limite dans les environnements dynamiques à cause de l'instabilité des nœuds. Une alternative est proposée dans [12], où les auteurs proposent HP2P, une architecture hiérarchique hybride à deux niveaux, combinant DHT et systèmes P2P non structuré. HP2P utilise dans son premier niveau Chord et au deuxième niveau Kazaa, et procède par inondation lors d'une recherche. Ceci fait de HP2P un système robuste car elle combine les avantages des DHT et des systèmes non structurés, mais reste aussi limité par leurs inconvénients, comme la dépendance aux ressources de même type (fichiers uniquement) et le nombre de messages exponentiel généré lors d'une recherche par inondation. Face à ces travaux, nous sommes motivés à proposer la spécification d'une architecture hiérarchique plus générique, qui permet le stockage de tous les formats de données tout en conservant les avantages vis-à-vis de la performance réseau.

Kessis et al. [9] proposent un intergiciel de gestion de ressources dans un contexte multi-échelle. Le but de ce système est de proposer une gestion flexible d'un ensemble de ressources réparties de façon multi-échelle. Dans cet article, le terme multi-échelle est employé pour décrire la répartition des ressources à travers des réseaux de tailles différentes : PAN (Personal Area Network), LAN (Local Area Network), WAN (Wide Area Network). La solution proposée consiste à regrouper les ressources en domaines et fédérations de domaines afin d'implémenter différents modèles de gestion. Il est également possible de passer d'un modèle de gestion à un autre au cours de l'exécution. Une représentation architecturale est extraite des ressources gérées afin de créer une couche d'abstraction homogène qui représente des ressources hétérogènes. Cette couche est composée d'éléments basiques qui peuvent être assemblés pour former des éléments composites. Les éléments basiques et composites sont tous considérés comme des éléments gérés de façon homogène du point de vue de l'application.

3. L'ARCHITECTURE GRAPP&S

3.1 Modèle et Définitions

3.1.1 Définitions

Pour la définition de l'architecture GRAPP&S nous considérons un modèle de communication représenté par un graphe non orienté et connexe $G = (V, E)$, où V désigne l'ensemble des nœuds du système et E désigne l'ensemble des liens de communications qui existent entre les nœuds. Le modèle utilisé pour notre système est étudié dans [4]. Deux nœuds u et v sont dits adjacents ou voisins si et seulement si u, v est un lien de communication de G. $u_i, v_j \in E$ est un canal bidirectionnel connecté au port i pour u et au port j pour v. Donc les nœuds u et v peuvent mutuellement envoyer ou recevoir des messages en mode asynchrone.

Un message m en transite est noté $m(id(u), m', id(v))$ où $id(u)$ est l'identifiant du nœud qui envoie le message, $id(v)$ est l'identifiant du nœud de réception, m' indique le contenu du message. Chaque nœud u du système a un identifiant unique id et dispose de deux primitives : `send(message)` et `receive(message)`. Par soucis de clarté, nous introduisons quelques définitions.

Definition 1. Un nœud est définit comme étant une capacité de calcul, de stockage, avec des moyens et des canaux de communications.

Definition 2. Une donnée brute est un flux d'octets qui peut être sous différentes formes : une base de données objet ou relationnelle, un fichier (texte et hypertexte, XML), un flux (vidéo, audio, VoIP), des fichiers P2P, des requêtes de base de données ou des résultats issus d'un calcul/service.

3.1.2 Communication et les réseaux overlay

Le modèle de communication présenté en Section 3.1.1 est suffisamment générique pour ne pas inférer sur la manière dont les messages sont effectivement livrés, se limitant uniquement à la définition des propriétés de communication bidirectionnelles entre deux vertex. Pour cette raison, l'architecture de GRAPP&S peut s'appuyer sur n(importe quel un réseau de communication *overlay* qui garantit une communication bidirectionnelle fiable entre deux vertex, e qui permet d'explorer différents chemins de communication pour chaque arête dirigée (réseau routé). Ceci donne une plus grande liberté d'implémentation et d'adaptation à l'environnement d'exécution, vu que les opérations send/receive peuvent être implémentées de différentes façons, selon les capacités de communication des nœuds. Dans ce cas, trois scénarios principaux peuvent être considérés :

- `Push`, où l'émetteur est capable d'envoyer un message directement au destinataire,
- `Pull`, où le récepteur cherche régulièrement des messages en attente (ce modèle est fréquemment utilisé dans le cas des réseaux derrière un NAT/pare-feu), et
- `Proxy`, où les voisins doivent passer par un nœud intermédiaire afin d'échanger des messages (par exemple, grâce à un middleware *publisher/subscriber*).

Dans ces trois scénarios, il est toujours possible d'établir un voisinage direct ou partiel entre les processus, ce qui est compatible avec le modèle par graphes connectes dirigés et qui répond donc aux besoins de l'architecture GRAPP&S.

3.2 Éléments de l'Architecture GRAPP&S

Afin de présenter notre architecture, nous introduisons dans un premier temps quelques notations. Une communauté (C_i) est une entité autonome, qui regroupe des nœuds qui peuvent se communiquer et qui partagent une propriété définie : même localisation, même autorité d'administration (des serveurs distants appartenant à la même entreprise, par exemple) ou même domaine d'application (base de données métier, par exemple). Une communauté contient un seul processus *Communicator* - (c) et au moins un processus *Ressource Manager* - (RM) et un *Data Manager* - (DM) et ces processus sont organisés de façon hiérarchique dans une communauté. L'interconnexion entre différentes communautés C se fait grâce à des liens de voisinage point-à-point entre les processus Communicateur.

3.2.1 Communicator (c)

Le nœud *Communicator* (c) joue un rôle essentiellement lié au transport d'informations et à l'interconnexion entre différentes communautés, comme par exemple lors du passage de messages à travers des pare-feu. C'est le point d'entrée de la communauté, et il assure sa sécurité vis-à-vis de l'extérieur, grâce à l'établissement de *Service-Level Agreements* (SLAs) avec les autres communautés. De même, le communicateur coordonne la sécurité intérieure de la communauté, et peut modifier ses politiques d'accès grâce à des décisions prises au sein de la communauté [6]. Un nœud c dispose d'un identifiant unique (ID) à partir duquel on construit les identités des autres nœuds de la communauté. Ce nœud ne stocke pas de données et ne fait pas d'indexation.

3.2.2 Ressource manager (RM)

Les processus *Ressource Manager (RM)* assure l'indexation et l'organisation des données et services dans la communauté. Ils reçoivent les requêtes des utilisateurs et assurent leur prétraitement. Les nœuds RM participent à la recherche de données dans la communauté. Pour des fins de tolérance aux fautes et performance, les informations indexées par les RM peuvent être redondantes et/ou partiellement distribuées (DHTs, par exemple). Afin de rendre plus performante la coordination des RMs, nous préconisons l'élection d'un RM designé (voir section 3.4.3).

3.2.3 Data manager (DM)

Les processus *Data Manager (DM)* interagissent avec les sources de données, qui peuvent être dans différents supports tels que les bases de données (objet ou relationnelle), les documents (texte/XML/multimédia), des flux (vidéo, audio, VoIP), des données issus de capteurs ou encore une service cloud. Un nœud DM est un service qui dispose des composants suivants :

- *(i)* une interface (proxy) adaptée aux différentes sources de données (disque dur, serveur WebDAV, FTP, base de données, stockage sur cloud type Dropbox, etc.) et reliée à ceux-ci par un protocole de connexion spécifique au type de donnée, par exemple JDBC, ODBC, FTP, etc.
- *(ii)* un gestionnaire de requêtes qui permet d'exprimer des requêtes locales ou globales et
- *(iii)* un gestionnaire de communication qui permet au nœud DM de communiquer avec le nœud RM auquel il est connecté.

3.3 Gestion de la Communauté

GRAPP&S peut etre déployé dans plusieurs types d'architecture selon le placement des nœuds. Dans le modèle de placement (i), les nœuds peuvent être regroupés dans une seule machine physique (voir Figure 1a). C'est l'exemple typique d'une machine d'un particulier, qui souhaite héberger une communauté de l'architecture. Le placement des nœuds sous cette forme peut être justifié par sa simplicité à mettre en œuvre lors de sa phase d'implémentation, en utilisant les concepts d'héritage et de polymorphisme. Les nœuds sont interconnectés par des sockets, des solutions RPC pour qu'ils puissent communiquer par message dans les deux sens entre deux nœuds.

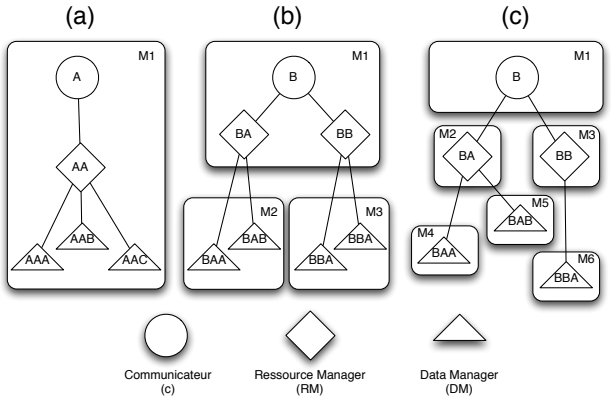

Figure 1: Organisation des nœuds (a) dans une machine, (b) dans un cluster et (c) dans un réseau

Dans (*ii*) les nœuds sont organisés dans une ferme de serveurs telle qu'un cluster, ce qui est caractéristique des réseaux HPC (Figure 1b). Finalement, (*iii*) les nœuds peuvent être regroupés s'ils partagent une même propriété de localisation ou d'administration (voir Figure 1c). Ceci est l'exemple d'un réseau formé par les nœuds d'une entreprise ou d'un laboratoire de recherche.

Chaque nœud de GRAPP&S a un identifiant (ID) unique. Les adresses IP ou MAC ne sont pas des identifiants suffisamment précis car ils ne permettent pas d'identifier de manière unique les différents nœuds qui peuvent résider sur une même machine (par exemple, un RM et plusieurs DM). De plus, l'utilisation des adresses IP ou MAC ne garantit pas une identité unique, vu que les adresses IP privés peuvent être réutilisées tout autant que les adresses MAC. En effet, certains fabricants peu scrupuleux réutilisent les adresses MAC qui les sont attribués, et cela occasionne de nombreux problèmes dans les réseaux locaux, tout comme sur le déploiement des réseaux IPv6 [1]. Ainsi, la solution la plus intéressante est celle proposée par JXTA [8] et qui utilise une chaine de 128 bits. Chaque nœud dispose ainsi d'une chaîne unique ID_{local}, sous la forme "*urn :nom_communaute :uuid : chaine-de-bit*". L'expression de l'adressage hiérarchique se fait par la concaténation des IDs sous forme de préfixe, i.e., l'ID du nœud c_i est équivalent à son ID_{local}, l'ID du nœud RM_i est formé par ID_{c_i}/ID_{RM_i}, et l'ID du nœud DM_i présente la forme $ID_{c_i}/ID_{RM_i}/ID_{DM_i}$.

Un avantage de l'utilisation d'un modèle d'adressage propre à GRAPP&S est que cela le rend indépendant du modèle d'adressage du réseau *overlay* sur lequel GRAPP&S est implémenté. Ainsi, deux communautés GRAPP&S implémentées sur des middlewares différents (FreePastry [2] et Phex [3], par exemple) seront toujours compatibles, une fois la connexion établie entre leurs *communicator*.

1. `http://tools.ietf.org/html/draft-gont-v6ops-slaac-issues-with-duplicate-macs-00`
2. `http://www.freepastry.org`
3. `http://www.phex.org`

3.4 Gestion des Nœuds

La topologie du réseau change fréquemment à cause de la mobilité des nœuds. Nous travaillons dans l'hypothèse où les tout nœud qui arrive dans le réseau est initialement un nœud DM. Selon les conditions de l'environnement où ce nœud se trouve, il peut se voir attribuer des rôles supplémentaires et "monter" dans l'hiérarchie.

3.4.1 Connexion d'un nœud

Quand un nœud DM arrive dans le réseau il dispose de deux moyens pour trouver un nœud RM sur lequel il peut se connecter.

- Si le nœud DM_i connait un ou plusieurs nœuds RM, il envoie un message de diffusion `Hello()` et collecte toutes les identités des nœuds RM, qu'il garde dans un tableau ordonné par l'identifiant. Il peut ainsi se connecter au nœud RM qui a l'identifiant le plus grand. Si ce dernier se déconnecte, alors le DM_i le supprime du tableau et se connecte au nœud RM suivant ;
- Si par contre le nœud DM ne connait aucun nœud RM, il doit effectuer une découverte sur le réseau local (par exemple, grâce à un multicast) ou contacter un service d'annuaire qui peut indiquer l'identifiant d'un nœud RM_i. Comme la manière de trouver le nœud RM_i dépend de l'implémentation, elle n'est pas précisée dans notre architecture.

Finalement, si aucune tentative de connexion à un nœud RM (et par extension, un nœud c) ne réussit, le nœud DM a la possibilité de former sa propre communauté. Il assume ainsi les trois rôles c, RM et DM, jusqu'à ce que d'autres nœuds le rejoignent. À ce moment, une élection pourra avoir lieu afin de redistribuer les rôles entre les nœuds.

3.4.2 Déconnexion d'un nœud

Les nœuds peuvent subir des déconnexions volontaires ou des involontaires (pannes). Comme le cas des déconnexions volontaires est trivial, nous nous concentrons ici sur les déconnexions involontaires.

Entre deux niveaux hiérarchiques, les pannes peuvent être détectées soit par des messages périodiques de type *Pull* (aussi connu comme *heartbeat*), à la demande par des messages *Push* (*ping-pong*) [5] ou encore en s'appuyant sur un mécanisme propre au middleware *overlay*. Pour les nœuds appartenant à un même niveau hiérarchique, la surveillance peut aussi se faire grâce à un mécanisme de passage de jetons "de service". Cela permet non seulement l'allégement du mécanisme de détection (il suffit de surveiller son prédécesseur et son successeur) comme permet la diffusion rapide des informations à l'ensemble des nœuds.

Pour la mise en place d'un mécanisme générique de détection de défaillances, nous préconisons une procédure en deux étapes. Tout d'abord, chaque nœud dispose d'une liste de voisins $\{N_1, \ldots, N_n\}$ composée des nœuds en contact direct (par exemple, un RM_i est en contact avec son c, ses DMs et ses voisins RM_{i-1} et RM_{i+1}. À cette liste de voisins est associé une liste de temporisateurs d'attente $\{ta_1, \ldots, ta_n\}$.

Lorsque aucun message du nœud N_k n'est reçu jusqu'à l'expiration du temps ta_k, une suspicion de défaillance est levée et doit être vérifiée auprès d'un deuxième nœud qui est aussi en contact direct avec le nœud suspect. Ainsi, si la suspicion concerne le nœud c, un nœud RM_i interroge son voisin direct RM_{i+1} avec un message jeton initialisé à `faux`. Si RM_{i+1} a reçu un message du nœud c avant l'expiration de son temps d'attente ta_c, RM_{i+1} modifie la valeur du jeton à `vrai` et retourne le message jeton à son émetteur RM_i. Ceci signifie (indirectement) que le nœud c n'est pas déconnecté et le nœud RM_i peut envoyer à nouveau un message au nœud c. Si par contre RM_{i+1} n'a pas été contacté récemment par c, il fera suivre un jeton la valeur `faux` qui, grâce au passage du jeton, alertera tous les nœuds RM $\{RM_1, \ldots, RM_n\}$ de la défaillance de c.

De manière similaire, si un nœud c suspecte un nœud RM_k, il peut demander confirmation à RM_{k+1}. Évidemment, cette procédure générique peut s'adapter aux différentes situations telles qu'un nœud qui contient un RM et plusieurs DM. Dans ce cas, le mécanisme de détection peut être allégé pour mieux répondre aux caractéristiques du nœud.

À la suite de la confirmation d'une défaillance, les nœuds concernés doivent *(i)* mettre à jour leurs informations (liste de voisin, tables d'index, etc.) et éventuellement (*ii*) procéder à l'élection d'un nouveau RM (respectivement c) qui prendra en charge les éventuels DM (ou RM) orphelins.

3.4.3 Algorithmes d'élection

Vu le caractère dynamique et volatile des réseaux informatiques, il est important de choisi un algorithme d'élection qui soit le plus léger et réactif possible. L'élection d'un nœud peut être nécessaire en deux situations : soit pour remplacer un nœud défaillant et garantir la continuité du service (par exemple, lors de la panne d'un nœud c), mais aussi pour simplifier la coordination entre les nœuds de même type, avec par exemple l'élection d'un RM qui agirait comme "supernode" pour l'indexation de données et services.

Il faut noter que la connaissance préalable des nœuds du niveau supérieur n'est pas obligatoire, vu que différentes techniques permettent d'obtenir les identifiants des autres nœuds. La méthode la plus simple consiste à utiliser directement le mécanisme d'adressage GRAPP&S : étant indépendant du middleware de communications, ce système d'adressage permet facilement de remonter la hiérarchie GRAPP&S et de contacter d'autres nœuds (grâce au routage du réseau *overlay*). Il suffit donc de remonter les niveaux de son propre identifiant ou de contacter d'autres nœuds dont on récolté les identifiants (ceux dont on a reçu des requêtes récemment, par exemple). Cette technique permet aussi de contacter d'autres *communicators* c_j et de réintégrer un réseau de communautés auprès la déconnexion involontaire de son *communicator* c_i. En dernier recours, GRAPP&S peut s'appuyer sur les éventuels mécanismes de découverte de topologie (broadcast/multicast) offerts par le propre *overlay*.

Vu que le problème de la reconnexion au reste de la communauté peut être traité de manière plus ou moins simple au sein de la propre architecture GRAPP&S, il est intéressant de se pencher sur les algorithmes d'élection eux-mêmes. Dans GRAPP&S, nous préconisons un algorithme d'élection distribué inspiré des protocoles de routage OSPF et IS-IS [11, 10, 2]. En effet, les nœuds GRAPP&S disposent d'un identifiant unique qui peut être utilisé de manière systéma-

tique par ces algorithmes d'élection.

Le choix entre les algorithmes de IS-IS ou d'OSPF est plus lié aux préférences d'implémentation et à l'hétérogénéité des nœuds. En effet, l'algorithme d'élection de IS-IS est de type déterministe, où l'élu est toujours le nœud avec le plus grand identifiant (appelé *DIS - Designated IS*). Ce mécanisme est simple à implémenter et ne requiert pratiquement aucun échange d'informations car les nœuds disposent déjà d'une liste avec les identifiants de leurs voisins, il ne resterait que le coût associé à la prise de fonctions d'un nœud elu à un rôle différent de celui qu'il occupait précédemment. L'inconvénient de cette technique est qu'un réseau avec un fort taux de volatilité peut occasionner des élections à répétition, soit lors de la déconnexion du leader, soit lors de la connexion d'un nœud avec un identifiant prioritaire.

Dans les cas où la volatilité risque d'impacter la performance du réseau, il est possible d'utiliser un mécanisme non-déterministe comme celui d'OSPF. Dans ce type d'algorithme, plus conservateur, le choix d'un leader (*DR - Designated Router*) n'est nécessaire que si le leader actuellement en place disparaît. Ainsi, l'entrée de nouveaux nœuds dans la communauté a un impact moins important sur le fonctionnement du réseau.

4. OPÉRATIONS DANS GRAPP&S

4.1 Stockage et Indexation

Le stockage de donnée dans le réseau GrAPP&S fait intervenir les nœuds Data managers DM, alors que les nœuds Ressource Manager RM permettent d'indexer les données et les services. À la fin, chaque donnée est identifiée de manière unique grâce à l'identifiant du nœud DM, auquel s'ajoute une extension contenant des informations et le type MIME des données. Ceci permet de franchir la barrière du simple "nom de fichier", et peut donc faire cohabiter des données statiques (fichiers), des données dynamiques (requêtes sur une base de données, résultats d'un calcul) et des données à caractère temporaire (flux voix ou vidéo, état d'un capteur, etc.).

L'ajout d'une nouvelle donnée dans le réseau se fait ainsi : Quand un nœud DM_i arrive dans le réseau, il se connecte à un nœud RM et publie les caractéristiques de ses données pour être indexées. Toute modification des données sur un DM sont propagées au RM auquel il est connecté, qui par la suite peut mettre à jour ces informations et les partager avec les autres RM.

Cette propagation des informations peut prendre différentes formes selon les politiques utilisées lors de l'implémentation du réseau des RM. Une implémentation qui veut garder la simplicité pourra simplement garder un index local sur chaque RM, qui sera consulté lors d'une recherche. Au contraire, une implémentation souhaitant minimiser les échanges lors d'une recherche de données penchera sur l'utilisation d'un super-noeud au sein des RMs ou d'un mécanisme de DHT. Il est aussi possible de favoriser la réplication des index et (voir des données), ce qui exige une coordination entre les RM afin de garder la cohérence des copies. Dans tous les cas, la surcharge des fonctionnalités d'un nœud ("supernode") n'est pas une obligation dans notre structure mais simplement une spécificité pouvant être présente dans une implémentation donnée.

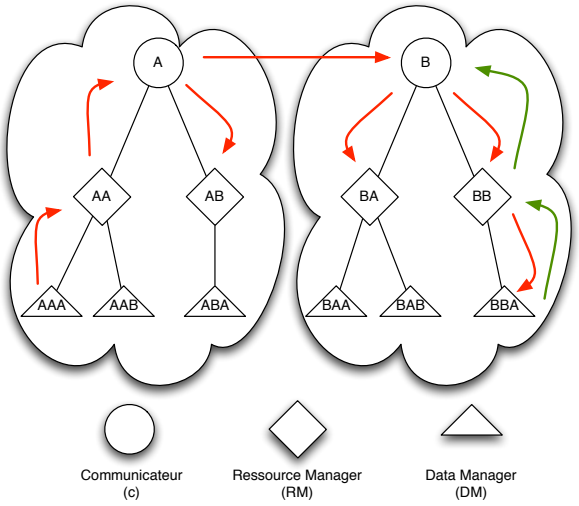

Figure 2: Recherche d'une information dans GRAPP&S et mécanisme de routage préfixé

4.2 Recherche

Quand un client cherche une donnée sur GRAPP&S, il entre en contact avec un proxy DM_i, qui envoie une requête Y contenant des informations qui identifient la donnée ou le service. Cette recherche dans une communauté de GrAPP&S se fait par paliers, de manière à respecter l'organisation hiérarchique du réseau. La Figure 2 illustre une partie de cette procédure de recherche :

1. Un nœud $DM_i \in C_i$ envoie la requête a son nœud RM_i E (C_i)
2. RM_i vérifie dans son indexe s'il y'a parmi ses voisins un DM qui contient la donnée recherchée
3. Si oui, alors le nœud RM_i retourne au nœud DM_i une liste de nœuds DM qui contiennent l'information recherchée
4. Sinon, le nœud RM_i fait suivre la requête soit directement aux voisins $RM_k \in C_i$ (si le mécanisme de communication le permet), soit à son nœud $c_i \in C_i$ pour retransmission aux autres $RM_k \in C_i$
5. quand un nœud $RM_k \in CM_i$) trouve la bonne réponse, alors la requête sera retournée au nœud DM_i émetteur en suivant le chemin inverse
6. Si la donnée recherchée n'est pas dans la communauté CM_i, alors le c_i fait suivre la requête vers d'autres communautés CM_j

En cas de réussie, le client obtient l'identifiant du nœud DM_x responsable par la donnée. Dans ce cas, le client a deux possibilités pour accéder à la donnée, soit par connexion directe, soit par une connexion routée. Dans le cas de la connexion directe, le client fait une requête directe au nœud DM_x afin d'accéder à la donnée. Comme le client peut se trouver dans un autre réseau qui ne permet pas l'accès directe à DM_x, la connexion peut se faire par routage interne dans GrAPP&S. Cela se fait de la manière suivante :

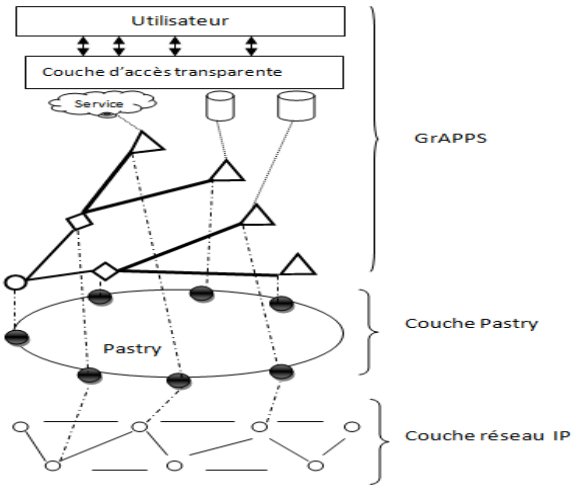

Figure 3: Illustration de la distribution des nœuds GRAPP&S sur un overlay Pastry

- Le client, par intermède d'un nœud DM_i, envoie une requête `Req`$(id(DM_x, Y, id(c_i))$ au nœud communicateur c_i.
- le nœud c_i, par routage préfixé, envoie la requête `Req` vers le nœud RM qui a indexé la donnée.
- Ce dernier fait suivre la requête vers le nœud DM_x responsable de la donnée.
- Une fois la donnée trouvée, le nœud DM_x retourne la bonne réponse au client en suivant le chemin inverse.

Ce mécanisme de recherche hiérarchique empêche l'inondation des liens du réseau. La hiérarchisation permet de définir des chemins par lesquels transitent les requêtes et comme la connectivité logique de notre architecture est par définition de $(n-1)$, il suffit d'appliquer un algorithme de type PIF pour agréger les requêtes et réduire le nombre de messages d'une recherche.

5. TRAVAUX FUTURS

5.1 Développement d'un Prototype sur Pastry

Un prototype de l'architecture GRAPP&S est en cours de développement. Ce prototype utilise le réseau overlay P2P FreePastry afin d'interconnecter les différents nœuds de l'architecture GRAPP&S, comme illustré en La Figure 3.

L'utilisation de FreePastry est purement pratique car l'overlay Pastry [15] se charge de toute interconnexion de bas niveau (ouverture de sockets, détection de défaillances, etc.), permettant ainsi aux développeur de se concentrer sur la coordination des nœuds GRAPP&S (élection, indexation des données et services, traitement de requêtes, recherche, etc.). L'utilisation d'un *design pattern* comme *Facade* permet un développement modulaire où la couche *overlay* peut être facilement remplacée. En effet, les contraintes de performance liées à la gestion des nœuds et au transfert des données peuvent être trop imposantes pour FreePastry, si la communauté est censée comporter un nombre élevé de nœuds ou si le réseau présente un taux de volatilité important.

En plus de démontrer l'application de l'architecture GRA PP&S et la mise en place de tests de scalabilité, ce prototype permettra aussi l'analyse de l'impact de la localisation des nœuds par rapport à la gestion de la hiérarchie. En effet, le fonctionnement de GRAPP&S repose sur une structuration hiérarchique pour le routage, la recherche et l'interconnexion de nœud distants. Si le placement des nœuds a un impact sur la performance de l'architecture, il devra être pris en compte aussi lors de l'élection des RMs et c.

5.2 Redistribution de ressources avec GAÏA

Si dans un premier moment GRAPP&S agit comme un médiateur pour la localisation de ressources stockées dans les différents DMs, il dispose aussi d'un potentiel d'évolution grâce à la redistribution de ressources à travers l'ordonnanceur GAÏA. Comme l'organisme homonyme de l'œuvre d'Isaac Asimov [1], GAÏA a pour but l'intégration et l'éventuelle réorganisation des ressources entre les différents DMs, selon des critères tels que la fréquence d'utilisation, la proximité vis-à-vis des clients, le besoin de réplication, le coût de stockage et, bien sûr, la faisabilité d'une telle distribution.

En effet, le projet GAÏA concerne le développement d'un ordonnanceur qui pourra gérer le stockage de certains types de données afin d'augmenter l'efficacité d'accès aux éléments les plus utilisés, tout en réduisant les coûts de stockage des données moins critiques. Grâce à GAÏA, GRAPP&S pourra mieux gérer l'intégration de services de stockage "lents" comme Amazon Glacier [4], qui présentent un coût réduit mais aussi des politiques d'accès plus restrictives que les mécanismes de stockage traditionnels.

6. CONCLUSIONS

Dans cet article nous présentons notre travail autour de GRAPP&S (GRid Applications and Services), une architecture multi-échelle pour l'agrégation de données et services. Nous présentons les principales fonctionnalités de GRAPP&S, comme la spécification de ses composants, des mécanismes de connexion, déconnexion, les opérations d'indexation, de recherche et d'accès aux données, tout comme les principes préconisés pour la coordination des nœuds.

En se basant sur les principes des systèmes multi-échelle, GRAPP&S a été conçu sous la forme d'un réseau hiérarchique basé autour du concept de "communautés", ce que permet l'intégration de sources de d'information disposant de protocoles d'accès hétérogènes et des règles de sécurité variés. De plus, l'utilisation des Data Managers, des proxies spécialisés pour l'adaptation et le traitement de différents types de données, il est possible d'intégrer de manière transparente aussi bien les données de type fichier que les bases de données, les flux (audio, vidéo), les services Web et le calcul distribué.

GRAPP&S est un travail en cours, dont un prototype en cours de développement servira à des tests de passage à l'échelle et comme plate-forme pour des services avancés telles que les grilles de sécurité [6] et l'optimisation du stockage avec l'ordonnanceur GAÏA, introduit dans cet article.

7. REFERENCES

[1] I. Asimov. *Foundation's Edge (Fondation Foudroyée)*. Bantam/Spectra, 1982.

4. `http://aws.amazon.com/fr/glacier/`

[2] M. Bhatia, V. Manral, and Y. Ohara. IS-IS and OSPF Difference Discussion. IETF Internet Draft, Jan. 2006.

[3] G. Blair and P. Grace. Emergent middleware : Tackling the interoperability problem. *Internet Computing*, 16(1) :78–82, Jan. 2012.

[4] J. Chalopin, E. Godard, Y. Métivier, and R. Ossamy. Mobile agent algorithms versus message passing algorithms. In *Principle of distributed systems*, volume 4305 of *LNCS*, pages 185–199, France, 2006. Springer.

[5] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2) :225–267, 1996.

[6] O. Flauzac, F. Nolot, C. Rabat, and L. Steffenel. Grid of security : a decentralized enforcement of the network security. In IGI. Global, editor, *Threats, Countermeasures and Advances in Applied Information Security*, pages 426–443. Manish Gupta, John Walp, and Raj Sharman (Eds.), April 2012.

[7] L. Garcés-Erice, P. Felber, E. W. Biersack, G. Urvoy-Keller, and K. W. Ross. Data indexing in peer-to-peer dht networks. In *ICDCS 2004*, pages 200–208, 2004.

[8] M. Ji. Hierarchical bidirectional chord. In *2010 International Conference on Educational and Information Technology (ICEIT 2010)*, pages 486–489. IEEE Xplore, 2010.

[9] M. Kessis, C. Roncancio, and A. Lefebvre. Dasima : A flexible management middleware in multi-scale contexts. In *Sixth International Conference on Information Technology : New Generations, (ITNG'09)*, page 1390–1396, Apr. 2009.

[10] J. Moy. OSPF Version 2. RFC 2328 (INTERNET STANDARD), Apr. 1998. Updated by RFCs 5709, 6549, 6845, 6860.

[11] D. Oran. OSI IS-IS Intra-domain Routing Protocol. RFC 1142 (Informational), Feb. 1990.

[12] Z. Peng, Z. Duan, J.-J. Qi, Y. Cao, and E. Lv. Hp2p : A hybrid hierarchical p2p network. ICDS '07, pages 18–, Washington, DC, USA, 2007. IEEE.

[13] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. *ACM SIGCOMM Computer Communication Review*, 31(4) :161–172, 2001.

[14] S. Rottenberg, S. Leriche, C. Lecocq, and C. Taconet. Vers une définition d'un système réparti multi-échelle. In *UBIMOB'12 - 8èmes Journées Francophones Mobilité et Ubiquité*, pages 178–183, 2012.

[15] A. Rowstron and P. Druschel. Pastry : Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, Nov. 2001.

[16] M. Satyanarayanan. Mobile computing : the next decade. *SIGMOBILE Mobile Computing and Communications Review*, (15) :2–10, Aug. 2011.

[17] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The case for vm-based cloudlets in mobile computing. *EEE Pervasive Computing*, (8) :14–23, Dec. 2009.