Structuration auto-adaptative des grilles pair-à-pair à large échelle

Bassirou Gueye and Ibrahima Niang
Département de Mathématiques et d'Informatique
Université Cheikh Anta Diop
Dakar, Senegal
Email: {bassirou.gueye, ibrahima1.niang}@ucad.edu.sn

Olivier Flauzac and Cyril Rabat
CReSTIC, UFR Sciences Exactes et Naturelles
Université de Reims Champagne Ardenne
Reims, France
Email: {olivier.flauzac, cyril.rabat}@univ-reims.fr

***Résumé*—Dans ce papier, nous proposons une extension et une implémentation de notre solution de structuration auto-adaptative dans un environnement de grilles P2P à large échelle. La spécification (nommée P2P4GS) que nous avons proposée est générique c'est à dire qu'elle n'est pas liée à une architecture pair-à-pair particulière ou à un protocole de gestion de services. De plus, elle est auto-adaptative et permet aussi bien le déploiement, la recherche et l'invocation de services tout en respectant le paradigme des réseaux P2P. Etant donné que la taille des systèmes à large échelle croît en termes de nombre de noeuds, de services et d'utilisateurs, nous proposons une approche d'auto-organisation du système en communautés en vue de garantir un passage à l'échelle. Des nœuds que nous appelons *PSI* (Proxys Système d'Information) seront responsables de la localisation des services des nœuds de leurs voisinages. Afin de permettre une recherche efficace dans le système, un arbre couvrant constitué par l'ensemble des noeuds PSI sera maintenu.**

***Keywords*-Réseaux pair-à-pair, Grilles de calcul, Services web, algorithmes distribués, arbres couvrants, Gia, Pastry, Kademlia**

I. Introduction

L'évolution des systèmes informatiques se caractérise par une tendance forte vers la décentralisation des services. En effet, la communication, le partage de données et des ressources de calcul et de stockage sont des besoins fortement exprimés par les nouvelles applications informatiques [1]. Pour faire face à ces exigences, des systèmes de partage à large échelle tels que les grilles et les systèmes pair-à-pair se sont imposés.

Les grilles de calcul (Grid Computing) [2] sont une technologie dont le but est d'offrir aux organisations virtuelles, ainsi qu'à la communauté scientifique des ressources informatiques virtuellement illimitées. Par le biais de ces grilles, les utilisateurs ont la possibilité d'accèder à des ressources distantes de calcul et de stockage, de lancer des applications qui demandent beaucoup de ressources non disponibles localement.

L'émergence des Services Web [3] a fourni un cadre qui a initié son alignement avec les technologies de grilles de calcul ainsi que celles de cloud computing [4]. En effet, les grilles de services représentent le résultat de recherche établi par le OGF (*Open Grid Forum*) et ayant abouti à l'OGSA (*Open Grid Service Architecture*) [5]. Un effort particulier de normalisation a été fourni afin d'apporter, comme dans le cas des Services Web, toutes les ressources et les moyens nécessaires au développement d'applications dans ces grilles de services. Cependant, ces grilles sont généralement basées sur des architectures hiérarchiques présentant un fort degré de centralisation [2][9]. Cette centralisation implique une gestion unifiée des ressources, mais aussi des difficultés à réagir vis-à-vis des pannes et des fautes qui impactent la communauté.

Pour remédier à ces inconvénients, des solutions de convergence des grilles informatiques et des systèmes pair-à-pair ont été proposées [10][11][12][13].

Un système pair-à-pair (*P2P*) [14] connu également sous le nom de réseau d'égal à égal ou poste à poste est un système où chaque pair peut jouer à la fois le rôle de client et de serveur. Contrairement au modèle client-serveur, le modèle P2P présente une infrastructure décentralisée. Il permet à des pairs du réseau de communiquer, de collaborer et de partager directement leurs services sans avoir besoin de faire appel à un serveur central.

Toutefois, soulignons qu'il n'existe pas dans la littérature une sémantique de la chaîne d'exécution des services depuis le déploiement selon différentes stratégies jusqu'à l'exécution, tout ceci en passant par la localisation et l'invocation. En effet, la plupart des solutions de grilles pair-à-pair se limitent sur la découverte de ressources et se distinguent dans le type d'algorithme de recherche utilisé [15][11][12][13][16].

Dans cet article, nous proposons une extension et une implémentation de notre solution nommée P2P4GS (*Peer-To-Peer For Grid Services*) et décrite dans [17][18]. En fait, P2P4GS est une spécification pour la gestion de services dans un environnement de grilles basé sur une architecture pair-à-pair. La spécification est générique c'est à dire non liée à une architecture pair-à-pair particulière ou à un protocole de gestion de service. En outre, la solution proposée est auto-adaptative, permettant aussi bien le déploiement, la recherche, l'invocation et l'exécution de services tout en respectant le paradigme des réseaux P2P.

Le reste du document est organisé comme suit. Dans la section II nous exposons les travaux connexes. Une vue d'ensemble de notre spécification est décrite dans la section III. La section IV présente notre contribution. Dans cette section, nous décrivons notre approche de structuration d'un système de grilles P2P à large échelle. Nos résultats expérimentaux sont présentés dans la section V. Une conclusion ainsi que des perspectives futures sont données dans la section VI.

II. Travaux connexes

L'exécution de code distant a donné lieu au schéma général des RPC (*Remote Procedure Call*). Cette spécification exprime les échanges entre le consommateur de services (le client) et la ressource (le serveur). Ce concept a été transféré dans les grilles, soit à partir des bibliothèques permettant la mise en place de solutions techniques, comme Globus [19], soit à partir des solutions directement conçues pour assurer le GRID RPC [20][9]. Néanmoins, les solutions proposées sont hiérarchiques et nécessitent des points de centralisation de la connaissance. Comme dans le cas des exécutions d'applications sur le Web, des solutions d'implantation [6] et de découverte de ressources [21][22] dans de tels environnements ont été proposées. Cependant, l'ensemble de ces travaux est basé sur des solutions d'architectures hiérarchiques qui présentent un degré de dynamicité très réduit et dont les services d'infrastructure, comme le déploiement et la localisation de services, sont eux même centralisés. En effet, les approches basées sur les systèmes centralisés ou hiérarchisés souffrent d'un point de défaillance unique, de goulots d'étranglement dans les systèmes hautement dynamiques, et de manque d'évolutivité dans les environnements hautement distribués [23].

Heureusement, les grilles informatiques et les systèmes pair-à-pair partagent plusieurs caractéristiques et peuvent ainsi être utilement intégrés. En effet, Ian Fooster et al. precisent dans [10] que les grilles et les systèmes pair-à-pair ont tendance à converger vers le même objectif, tout en partant d'un point de départ différent. L'objectif de l'intégration de ces deux paradigmes est de pallier les inconvénients des systèmes traditionnels de grilles. Marin Pérez et al. affirme ainsi dans [24] que la fusion des avantages de ces deux systèmes (P2P et grilles) permet d'obtenir une grande flexibilité ainsi qu'un gain énorme en termes de temps de recherche. Par ailleurs, ils soulignent que l'objectif ultime des grilles informatiques est d'intégrer les systèmes P2P et les services Web.

Signalons toutefois que la plupart des solutions de grilles pair-à-pair [11][12][25][13][26] se limitent à la découverte de ressources et se distinguent généralement dans le type d'algorithme de recherche utilisé. Ce constat motive les auteurs du *survey* [27] à soutenir que la découverte de ressources est un des défis essentiels dans un environnement de grille.

III. Background : la spécification P2P4GS

La spécification P2P4GS [17][18] que nous avons proposée présente l'originalité de ne pas lier l'infrastructure pair-à-pair à la plate-forme d'exécution de services. En fait, la couche de gestion de la grille P2P est séparée de celle de localisation et d'invocation de services. De plus, la spécification est générique et tend donc à être applicable sur toute architecture pair-à-pair. En outre, P2P4GS est auto-adaptative et permet aussi bien le déploiement, la localisation et l'invocation de services tout en respectant le paradigme des réseaux P2P.

Etant donné que la taille des systèmes distribués croît en termes de nombre de nœuds, de services et d'utilisateurs, nous avons proposé de structurer notre système en communautés (Figure 1) en vue de garantir un passage à l'échelle. Ainsi, nous limitons la connaissance sur la localisation des services sur des nœuds appelés *Proxys Système d'Information* (PSI). En vue de ne pas surcharger ces PSI, nous déléguons les tâches d'invocation et d'exécution de services à des nœuds dits *Proxys Invoquant* (PI). Etant donné que les services n'ont pas les mêmes contraintes d'exécution en termes de CPU, de mémoire, de bande passante, de plateforme d'exécution, etc., un nœud est dit proxy invoquant pour un service S_i donné si et seulement si :

i) il connaît sa localisation ;
ii) il respecte ses contraintes d'exécution, c'est-à-dire s'il possède la partie cliente (*stub*) du service.

Soulignons qu'un nœud respectant seulement la condition *i)* sera dit *Proxy Localisant* (PL) pour le service S_i.

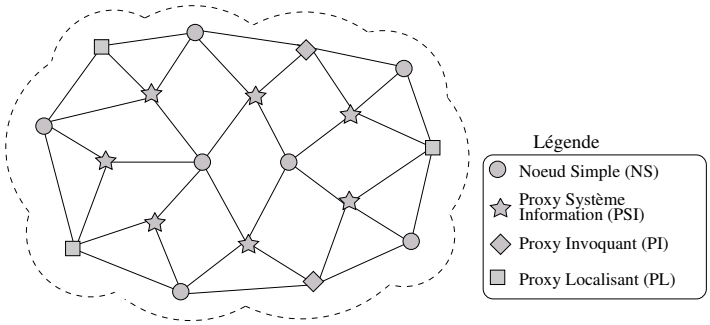

Figure 1: Structuration de notre système

Remarque : afin d'éviter une surcharge en mémoire des nœuds PI et PL, nous proposons de supprimer la connaissance sur la localisation d'un service à la fin d'un "*timer*" (T_{Live}). De ce fait, un service qui est sollicité assez rarement ne va pas être mémorisé de façon indéfinie. Par contre, un nœud restera PI ou PL pour un service assez fréquemment sollicité du moment où son T_{Live} sera réinitialisé après chaque appel.

IV. Approche de structuration auto-adaptative dans P2P4GS

Dans cette section, nous présentons dans un premier temps les résultats de notre approche de structuration du système décrite dans [17][18]. Ensuite, nous proposons une extension de cette approche en vue d'optimiser les coûts de gestion (en messages), de maintenance ainsi que de déploiement et de recherche dans un tel environnement.

Comme nous l'avons décrit plus haut, la croissance de la taille des systèmes à large échelle en termes de nombre de noeuds, de services et d'utilisateurs, pose la question de la scalabilité. Pour garantir ce passage à l'échelle, nous proposons de structurer le sytème en communautés de grilles P2P. L'originalité de notre approche d'organisation repose sur le fait d'être réaliste contrairement à la plupart des techniques de structuration qui considèrent que le nombre de nœud du système est connu au départ. Hors, dans un environnement à grande échelle où les nœuds sont géographiquement dispersés, le réseau ne peut pas se former de manière spontanée. Par

conséquent, nous proposons d'élire les PSI au fur et à mesure de la connexion des nœuds en se basant sur leurs voisinages.

L'approche proposée dans [17][18] est simple. En effet, lorsqu'un nœud se connecte au système et établit son voisinage initial (qui dépend du protocole P2P sous-jacent), il consulte le statut de ces derniers. Ainsi :

– s'il a au moins un voisin PSI, il se déclare alors nœud simple (SN) et envoie la liste de ses services hébergés à son (ses) PSI ;
– sinon, il se déclare PSI et informe ses voisins

La Figure 2 montre le pourcentage de PSI formés en fonction du protocole P2P et de la taille du système.

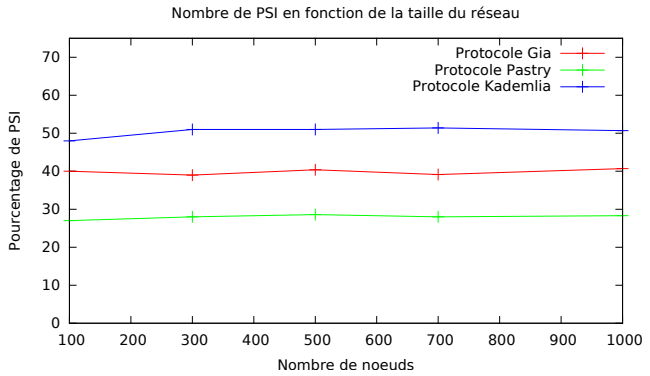

FIGURE 2: Pourcentage de PSI en fonction du protocole P2P et de la taille du réseau

Comme nous pouvons le constater, cette approche passe à l'échelle puisque le pourcentage de PSI formés reste relativement constant lorsque la taille du réseau augmente. Toutefois, ce pourcentage est suppérieur à 20% pour les différents protocoles P2P. Hors, comme le précise plusieurs travaux dans la littérature, nous préconisons un pourcentage de PSI compris entre 5% et 20% pour un meilleur dimensionnement du système.

Nous proposons ainsi d'optimiser cette approche en vue d'une part de diminuer le pourcentage de PSI formés pour élire les nœuds plus distingués (plus réputés) et d'autre part de diminuer les nombre de messages de gestion de l'overlay afin d'éviter une surcharge du réseau. Pour ce faire, nous rajoutons le critère degré de connexion dans l'approche initiale.

Ainsi, une fois qu'un nœud se connecte et établit son voisinage initial, il doit attendre d'avoir un nombre de voisins minimal ($\Delta_{RequiredMinDegree}$) pour être candidat potentiel à l'élection de PSI. De ce fait, ceux sont les nœuds plus réputés et plus stables qui ont plus de chance d'être PSI.

Rappelons que la découverte de ressources constitue un des défis essentiels dans un environnement de grille. Pour répondre à cette problèmatique, nous proposons de construire un arbre couvrant constitué uniquement des PSI ainsi formés. La particularité de cette approche est que la construction de l'arbre couvrant se fait au moment même de la structuration du système et ne nécessite pas par conséquent un coût supplémentaire en messages.

Le principe d'exécution se déroule comme suit :

Lorsqu'un nœud se connecte au système, il établit son voisinage (qui évolue au cours du temps) et demande à ses voisins leur statut. A la réception de réponse, le nœud exécute l'algorithme 1. Initialement, le nœud a le statut *UNDEF* et s'il a au moins un voisin PSI, il se déclare alors nœud simple (NS). Si par contre la taille de son voisinage atteint le degré minimal requis et qu'il n'a de voisin PSI, il se déclare alors PSI et informe ses voisins. Ces derniers vont envoyer la liste de leurs services hébergés ainsi que les adresses des autres PSI auquels ils sont voisins.

Algorithme 1: Lorsque le nœud i reçoit une réponse de demande de statut de son voisin j

```
1  if statut_i = UNDEF ∧ numberOfResponses = Δ_RequiredMinDegree then
2      if ∃ k ∈ neigh_i / statut_k = PSI then
3          statut_i ⟵ SN;
4          forall the k ∈ neigh / statut_k = PSI do
5              send(k, clusterManagement(ServiceList_i, IspList_i));
6          end
7      else
8          statut_i ⟵ PSI;
9      end
10     forall the k ∈ neigh do
11         send(k, (status_i)) ; /* Envoyer à k mon statut */
12     end
13 else
14     updateState(id_j, status_j) ; /* Mise à jour du statut de j */
15     if statut_i = SN ∧ statut_j = PSI then
16         send(j, clusterManagement(ServiceList_i, IspList_i));
17     end
18 end
```

A la réception d'un message *clusterManagement* (algorithme 2), le PSI mémorise la liste de services dans son registre de services et la liste des ardresses de PSI dans sa table de routage. Ensuite, il choisit dans cette table le PSI qui a numériquement le plus petit identifiant pour alimenter sa table de passerelles (*gatewayTable*). Le nœud choisi sera informé et ce dernier va aussi ajouter l'adresse du nœud source dans sa table de passerelles.

Algorithme 2: Lorsqu'un nœud PSI i reçoit un message $clusterManagement(serviceList_j, psiList)$ de son voisin j

```
1  forall the S ∈ serviceList_j do
2      serviceRegistry ⟵ serviceRegistry ∪ {(S, id_j)};
3  end
4  forall the q ∈ psiList do
5      if ∄ k = (id_k) ∈ routingTable / id_k = id_q then
6          routingTable ⟵ routingTable ∪ {id_q};
7      end
8  end
9  if |gatewayTable| = 0 ∧ |routingTable| > 0 then
10     gatewayTable ⟵ gatewayTable ∪ {id_k / id_k =
       min(routingTable)};
11     Informer à k qu'il sera ta passerelle lors du routage;
12 end
```

V. Étude expérimentale

Dans cette section, nous donnons une implémentation et une évaluation de notre solution par simulations. Après une étude poussée des simulateurs de réseaux P2P, nous avons opté pour le Framework Oversim [28] d'OMNeT++ [1] qui est un simulateur open source à événement discret et hautement modulaire. En fait, plusieurs protocoles P2P (structurés comme non structurés) sont implémentés dans OverSim.

Ainsi, afin d'atteindre nos objectifs, nous avons implémenté notre solution sur des protocoles P2P fonctionnant de manière totalement différente, à savoir Gia [29] qui est un overlay non structuré fortement basé sur le protocole Gnutella [2], Pastry [30] qui est un overlay structuré en anneau et Kademlia [31] qui est lui un overlay structuré en hypercube bien que modélisé souvent sous la forme d'un arbre. Soulignons que Kademlia est le premier protocole déployé réellement à une grande échelle, avec le logiciel eMule, via l'implémentation Kad et plus récemment dans la version Exeem [3] avec Bittorent.

Nos simulations sont effectuées sous Grid'5000 [4]. Nous avons utilisé les nœuds de Saint Rémi du Centre de Calcul de Champagne-Ardenne ROMEO [5] et les nœuds de Sophia [6].

Pour évaluer le pourcentage de PSI en fonction du degré minimal requis, nous considérons des réseaux avec un nombre de nœuds variant entre 500 et 5000. Dans cette étude, nous considérons trois valeurs du degré minimal requis à savoir 5, 10 et 15. Les Figures 3, 4 et 5 représentent le pourcentage de PSI formés en fonction du degré minimal requis, de la taille du réseau et du protocole P2P sous-jacent.

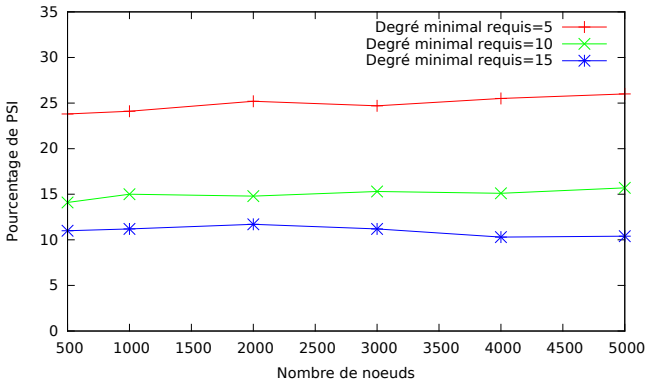

FIGURE 3: Protocole Gia : Pourcentage de PSI en fonction du degré minimal requis et de la taille du réseau

D'une manière générale, nous constatons que quelque soit le protocole P2P sous-jacent et quelque soit la valeur du degré minimal requis, le pourcentage de PSI formés reste relativement constant lorsque la taille du réseau augmente. Ce qui confirme le passage à l'échelle de notre solution.

1. http ://www.omnetpp.org/
2. http ://rfc-gnutella.sourceforge.net/
3. http ://www.exeem.com/
4. https ://www.grid5000.fr/
5. https ://romeo.univ-reims.fr
6. http ://www-sop.inria.fr/grid5000/

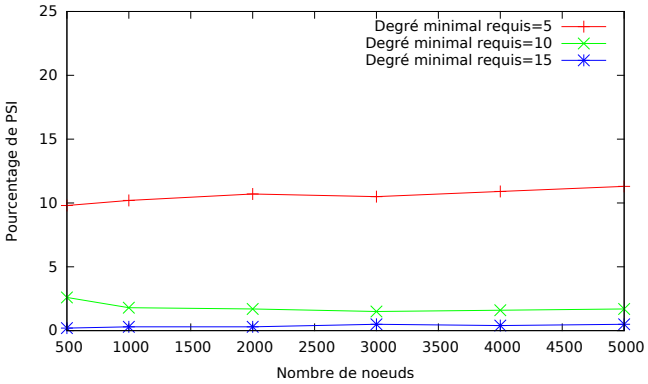

FIGURE 4: Protocole Pastry : Pourcentage de PSI en fonction du degré minimal requis et de la taille du réseau

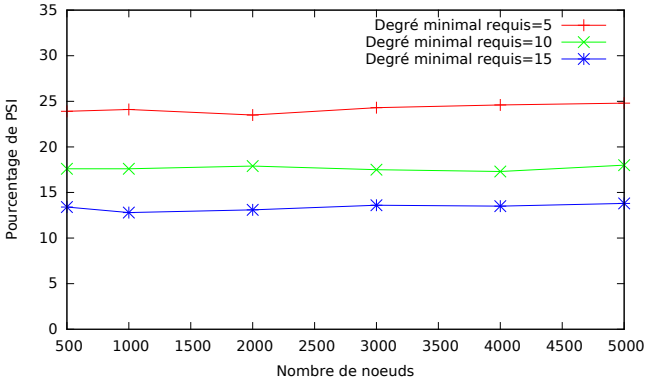

FIGURE 5: Protocole Kademlia : Pourcentage de PSI en fonction du degré minimal requis et de la taille du réseau

Rappelons que nous avons introduit le critère du degré minimal requis comme une contrainte dans le but de réduire le pourcentage de PSI formés dans la première approche et ainsi éviter d'avoir des clusters singletons ou contenant un nombre insignifiant de nœuds. Ainsi, comme nous pouvions l'imaginer, les résultats représentés dans les Figures 3, 4 et 5 montrent que plus le degré minimal requis augmente, plus le pourcentage de PSI formés diminue.

D'autre part, nous évitons aussi de construire un faible nombre de PSI très denses en vue de ne pas les surcharger et aussi ne pas favoriser des goulots d'étranglement dans le réseau. De ce fait, comme le précise plusieurs travaux dans la littérature, nous préconisons un pourcentage de PSI compris entre 5% et 20% pour un meilleur dimensionnement du système.

Pour le cas de Gia (Figure 3), nous remarquons que pour les valeurs respectives du degré minimal requis (5, 10, 15), nous obtenons respectivement des pourcentages de PSI compris entre 23,8% et 26%, 14,1% et 15,6%, 10,3% et 11,7%. Nous pouvons ainsi déduire que les valeurs de degré minimal requis 10 et 15 donnent de meilleurs dimensionnements.

Pour ce qui concerne le protocole Pastry (Figure 4), c'est la

valeur 5 de degré minimal requis fournissant un pourcentage de PSI entre $9,8\%$ et $11,3\%$ qui donne de meilleures proportions. Par contre les valeurs de degré minimal requis 10 et 15 fournissant des pourcentages inférieurs à 3%.

Enfin, dans le cas de Kademlia (Figure 5), nous obtenons des pourcentages de PSI compris entre 23,9% et 24,8% pour le degré minimal requis 5, entre 17,3% et 18% pour la valeur 10, entre 12,8% et 13,8% pour le degré minimal requis 15. Ainsi, les valeurs de degré minimal requis 10 et 15 fournissent un meilleur dimensionnement du réseau.

VI. Conclusion

Dans ce article, nous avons proposée une extension et une implémentation de notre solution de structuration auto-adaptative dans un environnement de grilles P2P à large échelle. La spécification P2P4GS que nous avons proposé est générique c'est à dire non liée à une architecture pair-à-pair particulière ou à un protocole de gestion de service. Afin de permettre une recherche efficace dans le système, nous construisons un arbre couvrant constitué uniquement des nœuds PSI.

Dans nos futurs travaux, nous détaillerons nos différentes stratégies de déploiement ainsi que notre solution de recherche efficace de services dans un environnement de grilles P2P à large échelle.

Références

[1] D. Bernholdt, S. Bharathi, D. Brown, K. Chanchio, M. Chen, A. Chervenak, L. Cinquini, B. Drach, I. Foster, P. Fox *et al.*, "The earth system grid : Supporting the next generation of climate modeling research," *Proceedings of the IEEE*, vol. 93, no. 3, pp. 485–495, 2005.

[2] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid : Enabling scalable virtual organizations," *International journal of high performance computing applications*, vol. 15, no. 3, pp. 200–222, 2001.

[3] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, *Web services - Concepts, Architectures and Applications*. Springer Verlag, ISBN 3-540-44008-9, chapter 5, 2004.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[5] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam *et al.*, "The open grid services architecture (ogsa), version 1.5." OGF Specification GFD-I. 080, July 2006.

[6] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling, and P. Vanderbilt, "Open grid services infrastructure (ogsi), version 1.0," June 2003.

[7] K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, and N. J. Wright, "Performance analysis of high performance computing applications on the amazon web services cloud," in *IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom)*, 2010, pp. 159–168.

[8] S. Bhardwaj, L. Jain, and S. Jain, "Cloud computing : A study of infrastructure as a service (iaas)," *International Journal of engineering and information Technology*, vol. 2, no. 1, pp. 60–63, 2010.

[9] E. Caron and F. Desprez, "Diet : A scalable toolbox to build network enabled servers on the grid," *International Journal of High Performance Computing Applications*, vol. 20, no. 3, pp. 335–352, 2006.

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

[11] P. Trunfio, D. Talia, H. Papadakis, P. Fragopoulou, M. Mordacchini, M. Pennanen, K. Popov, V. Vlassov, and S. Haridi, "Peer-to-peer resource discovery in grids : Models and systems," *Future Generation Computer Systems*, vol. 23, no. 7, pp. 864–878, 2007.

[12] T. Kocak and D. Lacks, "Design and analysis of a distributed grid resource discovery protocol," *Cluster Computing*, vol. 15, no. 1, pp. 37–52, 2012.

[13] J. A. Torkestani, "A multi-attribute resource discovery algorithm for peer-to-peer grids," *Applied Artificial Intelligence*, vol. 27, no. 7, pp. 575–598, 2013.

[14] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," *ACM Computing Surveys (CSUR)*, vol. 36, no. 4, pp. 335–371, 2004.

[15] A. Iamnitchi and I. Foster, "A peer-to-peer approach to resource location in grid environments," in *Grid Resource Management*. Springer, 2004, pp. 413–429.

[16] J.-S. Kim, B. Nam, and A. Sussman, "Scalable and effective peer-to-peer desktop grid system," *Cluster Computing*, vol. 17, no. 4, pp. 1185–1201, 2014.

[17] B. Gueye, I. Niang, O. Flauzac, and C. Rabat, "P2P4GS : A Specification for Services management in Peer-to-Peer Grids," in *The Fourth International Conference on Advanced Communications and Computation (INFOCOMP'2014)*. IARIA XPS, Paris, July 2014, pp. 41–46.

[18] B. Gueye, O. Flauzac, C. Rabat, and I. Niang, "A self-adaptive structuring for P2P-based Grids," in *14th IEEE International Conference on Innovations for Community Services (I4CS'2014)*, Reims, June 2014, pp. 121–128.

[19] I. Foster, "Globus toolkit version 4 : Software for service-oriented systems," *Journal of computer science and technology*, vol. 21, no. 4, pp. 513–520, 2006.

[20] K. Seymour, H. Nakada, S. Matsuoka, J. Dongarra, C. Lee, and H. Casanova, "Overview of gridrpc : A remote procedure call api for grid computing," in *Grid Computing—GRID 2002*. Springer, 2002, pp. 274–278.

[21] T. Gomes Ramos and A. C. M. A. de Melo, "An extensible resource discovery mechanism for grid computing environments," in *Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium on*, vol. 1. IEEE, 2006, pp. 115–122.

[22] F. Butt, S. S. Bokhari, A. Abhari, and A. Ferworn, "Scalable grid resource discovery through distributed search," *International Journal of Distributed and Parallel Systems (IJDPS)*, vol. 2, no. 5, pp. 1–19, 2011.

[23] Y. Deng, F. Wang, and A. Ciura, "Ant colony optimization inspired resource discovery in p2p grid systems," *The Journal of Supercomputing*, vol. 49, no. 1, pp. 4–21, 2009.

[24] J. M. Marin Perez, J. B. Bernabe, J. M. Alcaraz Calero, F. J. Garcia Clemente, G. M. Perez, and A. F. Gomez Skarmeta, "Semantic-based authorization architecture for grid," *Future Generation Computer Systems*, vol. 27, pp. 40–55, 2011.

[25] D. Chen, G. Chang, X. Zheng, D. Sun, J. Li, and X. Wang, "A novel p2p based grid resource discovery model," *Journal of Networks*, vol. 6, no. 10, pp. 1390–1397, 2011.

[26] J. A. Torkestani, "A distributed resource discovery algorithm for p2p grids," *Journal of Network and Computer Applications*, vol. 35, no. 6, pp. 2028 – 2036, 2012.

[27] N. J. Navimipour, A. M. Rahmani, A. H. Navin, and M. Hosseinzadeh, "Resource discovery mechanisms in grid systems : A survey," *Journal of Network and Computer Applications*, vol. 41, pp. 389–410, 2014.

[28] I. Baumgart, B. Heep, and S. Krause, "Oversim : A flexible overlay network simulation framework," in *IEEE Global Internet Symposium, 2007*. IEEE, 2007, pp. 79–84.

[29] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making gnutella-like p2p systems scalable," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM, 2003, pp. 407–418.

[30] A. Rowstron and P. Druschel, "Pastry : Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Middleware 2001*. Springer, 2001, pp. 329–350.

[31] P. Maymounkov and D. Mazieres, "Kademlia : A peer-to-peer information system based on the xor metric," in *Peer-to-Peer Systems*. Springer, 2002, pp. 53–65.