Publication pour la présente candidature LAFPT

Titre	Routage Préfixé dans GRAPP&S Applications à un Système Hiérarchique de Gestion de Données
Auteurs	Thierno Ahmadou DIALLO; Olivier FLAUZAC; Luiz Angelo STEFFENEL; **Samba N'DIAYE**
Référence	12e Colloque africain sur la recherche en informatique et mathématiques appliquées (CARI 2014)
Editeur	CARI
Pages	239-246
Année	2014
DOI	**http://www.cari-info.org/actes_2014/p25.pdf**
URL	http://www.cari-info.org/senegal-2014/
Index	
ISBN	
Encadreur	Oui
Extrait d'une thèse	Oui

Accéder au rapport CARI-2014 / Access to CARI'2014 report

PROCEEDINGS

Routage Préfixé dans GRAPP&S

Applications à un Système Hiérarchique de Gestion de Données

Thierno Ahmadou DIALLO[*#] - Olivier FLAUZAC[*] -
Luiz Angelo STEFFENEL[*] - Samba N'DIAYE[#]

[*] Département de Mathématique et Informatique
CReSTIC, Université de Reims Champagne-Ardenne, REIMS Cedex, FRANCE
{olivier.flauzac, luiz-angelo.steffenel}@univ-reims.fr
[#] Département Informatique, LMI, Université Cheikh Anta DIOP
5005 Dakar-Fann, SÉNÉGAL
{thierno80.diallo, samba.ndiaye}@ucad.edu.sn

RÉSUMÉ. Nous avons proposé dans nos travaux précédents GRAPP&S (Grid APPlication & Services), une spécification d'un Framework pour le stockage et l'indexation 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 (web services, cloud etc.). Cet article propose un mécanisme de routage préfixé pour la recherche et l'accès aux données de GRAPP&S. En effet, si la recherche d'une information dans les middlewares Pair-à-Pair (P2P) varie selon l'architecture de ceux-ci, le transfert de données dépend généralement d'une connexion directe entre la source et l'intéressé. Le mécanisme de routage que nous proposons dans GRAPP&S permet d'effectuer 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. Ce mécanisme contribue donc à relier des ressources très isolées mais aussi permet d'accroître la sécurité des réseaux, en offrant la possibilité de contrôler l'accès aux ressources.

ABSTRACT. We proposed in our previous work GRAPP&S (Grid Application & Services) a specification of a distributed Framework detached middleware communication (P2P, or other) on which it is deployed. GRAPP&S provides storage and indexing of various data formats such as files, data streams, queries on databases as well as access to remote services (web services, cloud etc.). The data is managed transparently to the user through proxies specific to each type of data. This paper proposes a routing mechanism prefixed for research and access to data GRAPP&S. This access does not depend on a direct connection between the nodes, as in most P2P or other networks. In GRAPP&S, it is always possible to route the data transfer path used when looking at cases where a direct connection between the nodes is not possible.

MOTS-CLÉS : Systèmes Pair à Pair, Routage préfixés, Proxies, Adressage hiérarchique

KEYWORDS : Peer-to-Peer Systems, Prefix Routing, Proxies, Hierarchical Addressing

1. INTRODUCTION

Dans un système P2P, les ressources sont distribuées entre tous les pairs. Afin d'accéder à une ressource, les nœuds doivent d'abord chercher ces ressources, obtenant ainsi les coordonnées des nœuds qui les détiennent. Généralement, une clé identifie une ressource dans un système P2P et, selon la manière dont les systèmes P2P organisent ces clés, les mécanismes de recherche et accès peuvent varier.

Dans les P2P non-structurés, comme par exemple Gnutella [6], l'absence d'une organisation des clés (ou d'un nœud d'index) force l'utilisation d'une méthode par inondation lors de la découverte des ressources. Toutefois, la réponse à une requête de recherche peut emprunter un chemin direct vers le nœud demandeur, si cela est possible (i.e., si aucune contrainte telle qu'un pare-feu n'empêche un tel envoi). Un inconvénient de cette approche est que l'inondation des messages de recherche contribue à limiter l'échelle du réseau. Des alternatives telles que celles proposées par les systèmes comme KaZaA [11], Gnutella0.6 [8] reposent sur des *super-pairs*, des nœuds intermédiaires qui permettent la concentration des requêtes et l'élagage de l'arbre de diffusion. De manière générale, pour N nœuds dans un système P2P non-structuré on a une complexité de recherche de $O(N)$.

Les réseaux P2P structurés tels que Chord [12] et Pastry [10], par contre, sont basés sur les DHT (*Distributed Hash Table*). Dans ce cas, l'algorithme de recherche associe un pair p à une clé de ressource k, de manière à ce que ce pair p soit agisse comme l'index de cette ressource. Ainsi, lorsqu'un nœud q recherche une ressource, il envoie la requête au pair r (parmi la liste de pairs connus à q) avec l'ID le plus proche de k, selon une certaine métrique. Lorsqu'un nœud r reçoit une requête, soit il détient la clé de la ressource et peut renseigner l'adresse où cette ressource est disponible, soit il renvoie le message à un nœud avec un ID plus proche à celui de la clé recherchée. Avec ce mécanisme, la complexité de la recherche dans un réseau P2P structuré de N nœuds est $O(logN)$. Donc, l'avantage d'un P2P structuré est son efficacité dans la recherche, mais il est plus fragile à la volatilité des nœuds, car cela pourrait entraîner des erreurs de table de routage.

Une caractéristique commune à tous les réseaux P2P, structurés ou pas, est qu'ils ont tous un design horizontal, avec un routage à plat (non hiérarchique). Dans ces réseaux, tous les pairs sont identiques dans le sens où tous les pairs utilisent les mêmes règles pour déterminer le routage d'une requête. Cette approche est très différente de celle de l'Internet, où le routage hiérarchique est utilisé.

Cet article propose un mécanisme de routage hiérarchique par préfixes (routage préfixé) pour la localisation des données et services gérés par notre framework GRAPP&S [4, 2, 3]. Avec ce mécanisme, l'accès aux données ne dépend pas d'une connexion directe entre les nœuds, comme dans la plupart des réseaux P2P. Dans GRAPP&S, il est toujours possible d'effectuer 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 l'état de l'art concernant le routage dans les réseaux P2P et le routage par préfixes. La Section 3 décrit brièvement les éléments du framework GRAPP&S, alors que les Sections 4 et 5 détaillent respectivement les mécanismes de recherche et d'accès aux données de GRAPP&S. Finalement la Section 6 propose nos conclusions.

2. ETAT DE L'ART

Les auteurs de [7] proposent le routage hiérarchique pour réduire l'information de routage dans les réseaux de large échelle. Cette technique consiste à garder sur chaque nœud une information de routage complète pour un ensemble de nœuds "proches". Ce mécanisme est appliqué dans Gnutella 0.6 [8], où il est introduit un type spécial de pair, les *supers pairs*, qui connaissent les contenus stockés chez d'autres pairs et qui peuvent répondre aux requêtes à leur place. Les supers pairs sont interconnectés par un réseau de niveau supérieur, donc Gnutella0.6 peut être considéré comme un réseau P2P hiérarchique non-structuré. Un autre réseau P2P hiérarchique hybride est présenté dans [9]. Cette proposition combine des méthodes d'inondation et de DHT. Chord est utilisée pour la couche supérieure des super-nœuds stables et les inondations pour la couche inférieure de nœuds instables. En cela, la couche inférieure est robuste et moins d'informations sont maintenues au niveau du super-pair mais, en raison des inondations, le coût de la recherche de niveau inférieur est $O(n)$. Ce système ne s'adapte pas bien si il y'a de nombreux nœuds dans le niveau inférieur.

Les auteurs de [1] ont introduit le routage par préfixes pour minimiser l'information de routage dans les réseaux dynamiques. L'idée du routage par préfixes est d'attribuer à chaque nœud n une étiquette $\alpha(n)$ qui est un mot sur un alphabet Σ contenant au moins deux symboles. Chaque lien est aussi étiqueté par un mot de Σ^* qui est un préfixe de quelques étiquettes de nœuds. Un message destiné au nœud n est transmis sur le lien dont l'étiquette est le plus long préfixe de $\alpha(n)$.

Les auteurs [1] ont montré que tout réseau admet un routage par préfixes et proposent une méthode de construction de fonctions de routage par préfixes. Ces algorithmes ont tous un coût optimal en temps de $O(1)$. L'inconvénient majeur des routages par préfixes est la taille des étiquettes de nœuds qui augmentent linéairement avec la taille du réseau. Pour certains réseaux, la taille de l'adresse de certains nœuds peut dépasser $O(Dlog\Delta)$, où D est le diamètre du réseau, Δ est le degré du réseau.

3. DESCRIPTION DU FRAMEWORK GRAPP&S

Afin de présenter GRAPP&S [4, 2, 3], nous introduisons quelques notations. Une communauté (C_i) est une entité autonome, qui regroupe des nœuds qui peuvent communiquer et qui partagent une propriété définie : même localisation, même autorité d'administration, ou même domaine d'application. Une communauté contient un seul processus *communicator (c)* et au moins un processus *ressource manager (RM)* et un processus *data manager (DM)*, et ces processus sont organisés de façon hiérarchique dans une communauté. La Figure 1 illustre le déploiement d'une communauté de GRAPP&S sur le middleware Pastry.

Le processus (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-feux. C'est le point d'entrée de la communauté et assure sa sécurité grâce à l'établissement de *Service-Level Agreements(SLA)* et à la gestion des tables de routage inter-communautés.

Le processus RM assure l'indexation et l'organisation des données dans la communauté. Les RM participent à la recherche de données dans la communauté.

Le processus DM est un proxy qui interagit avec les sources de données variées, tels que les bases de données, les services distants (cloud, serveur mail, FTP, Webdav). Un

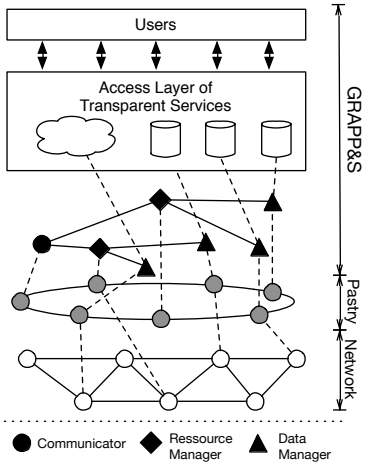

Figure 1. *Illustration de la distribution des nœuds de GRAPP&S sur l'overlay Pastry*

DM est un service qui dispose d'une interface (proxy) adaptée aux différentes sources de données, d'un gestionnaire de requêtes et d'un gestionnaire de communication.

3.1. Adressage hiérarchique

Dans GRAPP&S, chaque nœud 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 dans une même machine. Ainsi, la solution la plus intéressante est celle proposée par JXTA [13], 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 (ID_{c_i}) est équivalent à son ID_local, l' ID_{RM_i} est formé par ID_{c_i}-ID_{RM_i}, et l' ID_{DM_i} prend la forme ID_{c_i}-ID_{RM_i}-ID_{DM_i} .

L'avantage de l'adressage hiérarchique propre à GRAPP&S est que cela le rend indépendant du modèle d'adressage du réseau $overlay$ sur lequel GRAPP&S est implémenté. En plus l'adressage hiérarchique permet de remonter facilement la structure de GRAPP&S, en définissant les chemins par lesquels transitent les requêtes. Ce qui va empêcher l'inondation des liens de GRAPP&S.

3.2. Méthode de routage de base

Soit T un arbre et une numérotation des sommets de T suivant un DFS [5] dans l'arbre. Pour chaque sommet X d'identifiant Id_X l'adresse de X est constituée par la chaine binaire représentant Id_X concaténé par les chaines binaires des sommets parents de X dans T. L'adresse de chaque sommet X est donc constituée par une chaine binaire $PATH_X$ et d'un entier $lpath_X$ représentant la longueur de la chaine.

Pour tout sommet X de T : soit $Masque_F$ le masque constitué par une chaine binaire de $lpath_X$+8 bits dont les $lpath_X$ premiers bits sont tous à 0 et les 8 derniers bits sont à 1 (par exemple pour un sommet X tel que $lpath_X$= 16, son masque sera $Masque_X$= 00000000. 00000000. 11111111). Si Y est un descendant de X dans T, alors $PATH_X$ est une sous chaîne de $PATH_Y$ et en appliquant un ET logique entre $Masque_X$= 00000000.

00000000. 11111111 et $PATH_Y$ on a le numéro de Y. Si Y n'est pas descendant de X dans T, alors il faut passer par le père de X pour aller en Y. Soit est_parent**(X,Y)** la fonction qui pour tout couple de sommet retourne VRAI ou FAUX suivant que X est parent de Y dans l'arbre T ou pas. L' architecture de GRAPP&S est constituée de trois niveaux, ce qui limite la taille des adresses même dans un réseau de grande taille.

La méthode d'expédition de message dans notre système hiérarchique est donc inspiré de celui proposé par [1]. L'algorithme de routage de base (Algorithme 3), permet la transmission de message entre deux sommets, en connaissant l'identifiant du nœud source et celui du nœud destinataire.

4. RECHERCHE DE DONNÉES

Parmi les fonctionnalités principales de GRAPP&S, il y a la recherche et l'accès aux données. Ce mécanisme de recherche est propre à GRAPP&S et indépendant des mécanismes des overlays utilisés par l'implémentation. Cette section décrit notre algorithme de recherche de données dans une communauté de GRAPP&S. Les différents nœuds qui interviennent dans cette recherche disposent de quelques primitives :

– $Send$ qui permet à un nœud source d'envoyer un message à un nœud destination. Cependant, cette primitive nécessite une connexion directe entre les nœuds, raison pour laquelle elle est associée à la primitive $Route$ (Algorithme 3).

– $Route$ permet a un nœud source de transmettre un message à un nœud quelconque, soit en l'envoyant directement grâce à $Send$, soit en relayant l'information à un nœud avec le mécanisme de routage préfixé. Afin d'obtenir une adresse dans le mécanisme de routage préfixé, on utilise une méthode $getNextHop()$ qui effectue la comparaison des adresses et applique les masques de sous-réseau afin d'obtenir l'adresse du prochain nœud en direction de la destination.

– $Receive$ qui permet de traiter les messages reçus, selon leurs types et selon le rôle joué par le nœud.

– $Recherche\_locale$ qui permet à un DM de répondre à une recherche de ressource.

Lors de la recherche, les messages peuvent assumer deux formes :

– message de recherche - ce message a la forme $("req", contenu, source)$

– message de réponse - ce message a la forme $("reply", contenu, source)$

Quand un client cherche une donnée sur GRAPP&S, il entre en contact avec un proxy DM_i, qui envoie une requête Y contenant les caractéristiques de la donnée et le type "**req**". Comme le message n'a pas un destinataire précis, la recherche dans une communauté de GRAPP&S se fait par paliers, de manière à respecter l'organisation hiérarchique du réseau. La procédure de recherche est comme suit :

1) $DM_i \in C_i$ envoie la requête Y a son nœud $RM_i \in C_i$ grâce à $Route()$.

2) RM_i vérifie grâce à la méthode $Recherche\_locale$(m) si dans son indexe il y a un voisin DM qui contient la donnée recherchée,

a) Si oui, alors le nœud RM_i retourne au nœud DM_i une liste de nœuds DM qui contiennent l'information recherchée, sous la forme d'un message avec le type "**reply**".

b) Sinon, le nœud RM_i fait suivre la requête à son nœud $c_i \in C_i$ pour une diffusion aux autres nœuds $RM_k \in C_i$ tel que les $RM_k \neq RM_i$ (Algorithme 5).

- Lorsqu'un un nœud $RM_k \in C_i$ trouve une correspondance, une réponse estampillée "**reply**" sera retournée au nœud DM_i émetteur grâce à $Route()$.

c) Si la communauté C_i est connectée à d'autres communautés, alors le c_i fait suivre la requête selon les SLAs mises en place.

```
m : message ;
role : rôle du nœud (DM, RM, c) ;
m.type : type de message (req | reply | get | data) ;
Procédure Receive(m)
    Si (role=DM) Alors
        Si (m.type=reply) Alors
            m' ← ("get", m.data, local_address)
            Route(m', m.source) ;
        Fin Si
        Si (m.type = get) Alors
            m' ← ("data", data, local_address)
            Route(m', m.source) ;
        Fin Si
    Fin Si
    Si (role=RM) Alors
        Si (m.type=req) Alors
            Si (recherche_locale(m)=FAUX) Alors
                Si (origin=child) Alors
                    Route(m, father) ;
                Fin Si
            Fin Si
        Sinon
            Route(m, m.destination) ;
        Fin Si
    Fin Si
    Si (role=c) Alors
        Si (m.type=req) Alors
            Diffusion(m) ;
        Sinon
            Route(m, m.destination) ;
        Fin Si
    Fin Si
Fin
```

Algorithme 1: Traitement des messages selon les rôles des nœuds

```
m : message de recherche ;
m.source : adresse du DM qui a émis le message ;
m' : message de réponse ;
local_address : adresse du nœud local ;
Fonction Recherche_locale(m) : booléen
    listeDM ← RM.Verif_of_RM(m.data)
    Si (listeDM ≠ ∅) Alors
        Pour (chaque DM_k de RM tel que DM_k ∈ listeDM) faire
            m'← ("reply", m.data, DM_k)
            Route(m', m.source) ;
        Fin Pour
        Retourner VRAI
    Sinon
        Retourner FAUX
    Fin Si
Fin
```

Algorithme 2: Recherche locale

```
m : message a envoyer à la destination ;
local_address : adresse du nœud local ;
destination : nœud destination ;
add : Adresse d'un nœud ;
Procédure Route(m[, destination])
    Si (destination ≠ ∅) Alors
        Si (canReach (destination)) Alors
            add ← getAdd (destination) ;
            Send(m, add, local_address) ;
        Sinon
            add ← getNextHop (local_address, destination) ;
            Send(m, add, local_address) ;
        Fin Si
    Sinon
        add ← father ;
        Send (m, add, local_address) ;
    Fin Si
Fin
```

Algorithme 3: Routage de base

```
Index_of_RM[0...n] : index du nœud RM ;
X ←{} : liste de DM ;
TypeMime : Type MIME de l'information ;
TypeMimeRec : Critère de recherche ;
Fonction Verif_of_RM(TypeMimeRec) : X ←{}
    Pour (i allant de 0 à n) faire
        Si (Index_of_RM[i].TypeMime=TypeMimeRec)
        Alors
            X ← DM_i ;
        Fin Si
    Fin Pour
    Retourner X ;
Fin
```

Algorithme 4: Vérification de l'index RM

```
m : message a diffuser ;
origin : adresse du RM qui a transmis le message ;
local_address : adresse du nœud local ;

Procédure Diffusion(m, origin)
    Pour (chaque fils RM_k de c_i tel que RM_k ≠ origin) faire
        Route(m, RM_k)
    Fin Pour
Fin
```

Algorithme 5: Diffusion de message aux RMs

5. TRANSFERT DE DONNÉES DANS GRAPP&S

En cas de réussite lors de la recherche, le client (DM_x) obtient l'identifiant du nœud DM_y qui détient la donnée. Pendant l'étape de transfert de données, les messages peuvent assumer deux formes :

– message de demande - ce message a la forme $(\text{"}get\text{"}, contenu, source)$

– message de transfert - ce message a la forme $(\text{"}data\text{"}, contenu, source)$

Dans ce cas, il a deux possibilités pour entrer en contact avec DM_y et récupérer la donnée, soit par une connexion directe, soit par une connexion routée. Dans les deux cas, on fera appel à la primitive $Route()$, qui sera responsable par l'envoi d'un message de type "get". Si la connexion directe est possible, l'envoi se fera par $Send()$, dans le cas contraire il se fera par le mécanisme de routage hiérarchique.

De même manière, lorsqu'il reçoit un message de type "**get**", le nœud qui détient la donnée peut la transmettre avec la primitive $Route()$, grâce à un message de type "**data**".

6. CONCLUSIONS

Dans cet article nous présentons l'une des principales fonctionnalités de GRAPP&S, c'est-à-dire la recherche et l'accès aux données. L'organisation hiérarchique de GRAPP&S et son système d'adressage hiérarchique définissent les chemins par lesquels transitent les requêtes de recherche, ce qui empêche l'inondation des liens du réseau. Comme l'accès aux données de GRAPP&S ne dépend pas d'une connexion directe entre les nœuds, il est toujours possible par routage de rapatrier les données par le chemin utilisé lors de la recherche, au cas où une connexion directe entre les nœuds n'est pas possible.

La prochaine étape de notre travail consiste à développer un prototype de GRAPP&S en s'appuyant sur le overlay Pastry afin de valider les algorithmes et la performance du système.

GRAPP&S servira à connecter les instituts d'enseignement, intéressés par le partage de ressources à travers une mise en commun des données éducatives de chaque institut sous forme de "communauté". Face aux problématiques des solutions de partage basées sur le cloud où le stockage des données est distant et la vitesse d'accès lente, GRAPP&S

devient une solution décentralisée qui permettra un accès rapide aux ressources du fait de la proximité des données et des consommateurs.

7. Bibliographie

[1] ERWIN M BAKKER, JAN VAN LEEUWEN, AND RICHARD B TAN, « Prefix routing schemes in dynamic networks », *Computer Networks and ISDN Systems,26(4) :403-421,1993.*

[2] T.DIALLO, O.FLAUZAC, L.STEFFENEL, AND S.N'DIAYE, « GrAPP&S, une approche de type Grid et système Pair à Pair », *In Actes du 4e CNRIA.URED, 2012.*

[3] T.DIALLO, O.FLAUZAC, L.STEFFENEL, AND S.N'DIAYE, « GrAPP&S, une Architecture Multi-échelle pour les Données et le Services », *In UBIMOB'13-9éme Journées Francophones Mobilité et Ubiquité,2013*

[4] T.DIALLO, O.FLAUZAC, L.STEFFENEL, S.N'DIAYE, AND Y.DIENG, « GrAPP&S, a Distributed Framework for E-learning resource Sharing », *In Fifth International IEEE EAI Conference on e- Infrastructure and e- Services for Developing Countries(AFRICOMM), Springer, 2014.*

[5] PIERRE FRAIGNIAUD, AND CYRIL GAVOILLE, « Routing in trees », *volume 2076 of Lecture Notes in Computer Science, page 757-772. Springer Berlin Heidelberg, 2001.*

[6] JUSTIN FRANKEL, AND T.PEPPER, « The Gnutella protocol specification v0.4 », *2003.*

[7] LEONARD KLEINROCK, AND FAROUK KAMOUN, « Hierarchical routing for Large Networks », *Computer Networks, 1 :155-174, 1977.*

[8] TOR KLINGBERG, AND RAPHAEL MANFREDI, « The Gnutella protocol specification v0.6 », *Technical specification of the protocol, 2002.*

[9] ZHUO PENG, ZHENHUA DUAN, JIAN-JUN QI, YANG CAO, AND ERTAO LV., « HP2P : A Hybrid Hierarchical P2P Network », *In Digital Society, 2007, ICDS'07,pages 18-, Washington, DC, USA, 2007. IEEE.*

[10] ANTONY ROWSTRON, AND PETER DRUSCHEL, « Pastry : Scalable, Decentralized Object location, and routing for large-scale peer-to-peer Systems. », *In Middleware 2001, pages 329-350. Springer, 2001.*

[11] SEUNGWON SHIN, JAEYEON JUNG, AND HARI BALAKRISHNAN, « Malware Prevalence in the Kazaa file-sharing network », *In Proceedings of the 6th ACM SIGCOMM conference on Internet measurement,pages 333-338. ACM, 2006.*

[12] ION STOICA, ROBERT MORRIS, DAVID KARGER, M FRANS KAASHOEK , AND HARI BALAKRISHNAN, « Chord :A scalable peet-to-peer lookup service for internet applications », *In ACM SIGCOMM Computer Communication Review, volume 31, pages 149-160.ACM,2001.*

[13] BERNARD TRAVERSAT, AHKIL ARORA, MOHAMED ABDELAZIZ, MIKE DUIGOU, CARL HAYWOOD, JEAN CHRISTOPHE HUGLY, ERIC POUYOUL, AND BILL YEAGER, « Project jxta 2.0 Super-peer virtual network », *Sun Microsystem White Paper.Available at www.jxta.org/project/www/docs,92,2003.*