الجمهورية الجزائرية الديمقراطية الشعبية

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليم العالي والبحث العلمي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جامعة أبي بكر بلقايد – تلمسان –

Université Aboubakr Belkaïd – Tlemcen –

Faculté de SCIENCE

Mémoire de fin d'études

Présenté pour l'obtention du **diplôme** de **MASTER**

En : Informatique

Spécialité : réseaux et systèmes distribués

Par :

KHERBOUCHE Abderrahim

HAMEL Mohammed Youcef

Thème

Partage de fichiers et messagerie instantanée dans un réseau pair-à-pair sur JXTA

Soutenu publiquement, le 03/ 07/2017, devant le jury composé de :

• Mme Didi Fedoua	Président
• Mme Iles Nawal	Examinateur
• Mr Belhoucine Amine	Encadrant

Remerciement

Pour commencer, on remercie le bon Dieu, par sa grâce nous sommes parvenus à élaborer notre modeste travail.

C’est avec une profonde reconnaissance et considération particulière que nous remercions notre encadreur Mr, Belhocine Amin pour son soutien, ses conseils judicieux et sa grande bienveillance durant l’élaboration de ce projet.

Nous saisissons également cette opportunité pour remercier l’ensemble des enseignants qui ont contribué à notre formation.

Ainsi nous exprimons notre reconnaissance à tous les membres du jury d’avoir répondu présent afin d’évaluer cet humble travail et d’apporter les critiques nécessaires à la mise en forme de cet ouvrage.

On tient aussi à remercier nos proches, amis et connaissances de nous avoir portés aide, ne serait-ce que par un mot d’encouragement ou un soutien moral.

Dédicace

On dédie cet humble travail :

- A nos chers parents qui se sont donné corps et âmes pour notre réussite.
- A nos frères et sœurs.
- A nos familles.
- A tous ceux qui nous ont aidés que ce soit de près ou de loin.

HAMEL & KHERBOUCHE

Table des matières

Table des figures

Introduction générale

Actuellement le pair-à-pair gagne du terrain sur les esprits des développeurs récents grâce aux maintes ressources aux quelles internet dispose et qui sont souvent mal exploitées. On distingue trois types de ressources : bande passante, espace de stockage et capacité de traitement. Le modèle client-serveur classique ne parvient pas à tirer pleinement profit de ces ressources la raison pour laquelle le modèle pair-à-pair est né.

Le projet JXTA, l'œuvre de Sun Microsystems, avait pour objectif de standardiser les applications pair-à-pair qui manquent de normes et de soutien industriels. Lancé en avril 2001, une première version de leur plateforme de développement a vu le jour dans l'Open Source. Elle facilite l'interopérabilité entre les différentes applications pair-à-pair. En plus, elle est indépendante du matériel ou du système utilisés comme elle peut être intégrée dans n'importe quel type d'équipement pouvant être relié à d'autre.

Dans le cadre du projet de fin d'études, nous avons développé une application pair-à-pair qui permet le partage de fichiers, la découverte des pairs connectés et le téléchargement des ressources une fois la liaison établit, plus une messagerie instantanée.

Dans ce mémoire nous allons essayer d'expliquer les étapes empruntés pour mettre en œuvre nos services sous la plateforme JXTA.

Le mémoire est composé de quatre chapitres :

Le premier chapitre explique brièvement les deux architectures à savoir le client-serveur et le pair-à-pair.

Le second chapitre explique plus en détail l'architecture pair-à-pair, ses modèles (structurés et non-structurés) notamment leurs approches, avec des exemples concrets.

Le troisième chapitre présente la plateforme JXTA, ses notions, protocoles et services. Il éclaircit les objectifs de cette plateforme et les plus qu'il a apporté par-rapport aux autres systèmes pair-a-pair.

Le quatrième chapitre celui de la mise en œuvre de l'application, il montre à la fois sa réalisation et son fonctionnement avec une sorte de mode d'emploi.

I. CHAPITRE

Généralités

1. Introduction

A l'aube de la micro-informatique, tout juste après la guerre mondiale, le matériel informatique était couteux, Seule les entreprises pouvaient s'en en procurer. La disquette était le seul moyen d'échanger les données de poste-à-poste. Cela ne causait aucun soucie dans un même département, par contre lorsqu'il s'agit d'un échange distant ça devenait plus compliqué.

Au fil du temps les besoins des entreprises s'élargissent à en falloir envisager une autre alternative au mode d'échange des données.

En 1960, les ingénieures militaire et industriel ont collaboré afin de résoudre ce problème. La carte réseau a vu le jour suite aux recherches du consortium ''D.I.X'' (Digital, Intel, Xerox) dont la fonction est la communication explicite entre poste.

2. Le modèle client-serveur

2.1 Introduction à la notion client/serveur

Dans les entreprises, le nombre élevé des machines peut devenir parfois problématique, même le réseau internet est basé sur ce type d'architecture appelée centralisée, par conséquent la maintenance et la gestion deviennent des enjeux capitaux, les répercussions peuvent être catastrophique.

Comme son nom l'indique cette architecture est composée d'un client et d'un serveur qui vont collaborer afin d'effectuer une tâche particulière.

2.2 Définition

Le modèle client-serveur se traduit autour d'un réseau dont deux types d'ordinateurs y sont connectés, « client et serveur ». La communication client-serveur est rétablie par des protocoles. Les données et les applications sont réparties entre client et serveur de tel sorte à réduire le cout (latence, fraies matérielles, etc.). Le client-serveur fait référence au dialogue entre processus via des messages. Le processus serveur réalise ce dont le processus client sous-traite comme service. Généralement les processus s'exécutent dans des machines, des OS et des réseaux hétérogènes. Et c'est grâce au

client-serveur que les environnements hétérogènes permettent de tirer profit du meilleur des deux monde (interface des Pc, puissance et sécurité des serveurs).

La fibre SQL ingénierie (un prestataire de service spécialisé dans la mise en place de solutions de type ‘’tout internet’’ créé en 1990, il devient le groupe SQLI) propose la définition suivante : « Une application qui fait appel à des services distants au travers d'un échange de messages plutôt que par un échange de fichiers est conforme au modèle client-serveur. » cette définition large recouvre une multitude d'approches. [1]

2.3 Notion client / serveur

Un environnement client-serveur est un mode de communication entre plusieurs programmes ou logiciels, celui qui envoie des requêtes dit Client, l’autre en revanche attend les requêtes des clients et y répond, dit serveur, autrement dit dans des ordinateurs Client le logiciel client est exécuté, ainsi dans des ordinateurs serveurs le logiciel serveur s’y exécute.

Les serveurs sont généralement des ordinateurs voués à abriter des logiciels serveurs, leurs aptitudes sont bien supérieures à celles des ordinateurs personnels de par leur puissance de calcul, les entrées-sorties et connexions réseau. Ils se distinguent par leur capacité de répondre aux requêtes d’un grand nombre de Clients. Quant aux clients, habituellement se sont des ordinateurs personnels ou des appareils individuels tels que le téléphone, la tablette... etc.

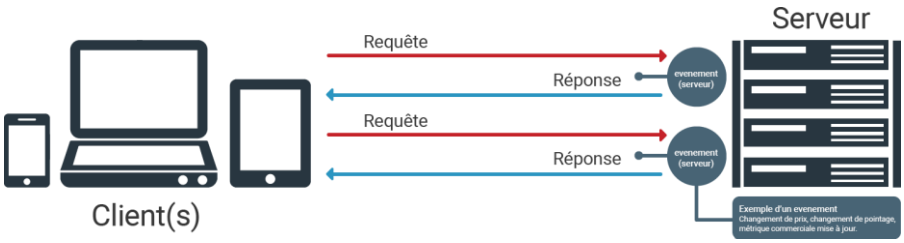


Figure 1. Client-serveur

2.3.1 Le client

Le logiciel Client permet à son utilisateur d’exploiter l’ensemble des services présents sur un réseau donné, d’ailleurs l’appellation reste la même pour les ordinateurs

connectés à un serveur. La couche applicative du modèle OSI dicte la conduite de la communication de ce dernier. [2]

Caractéristique client [3]

- Actif (maitre).
- Envoie des requêtes au serveur.
- Il attend et reçoit les réponses du serveur.
- Il existe de sorte de client-serveur, il est dit plat si les clients communiquent qu'avec un seul serveur a l'inverse de l'hiérarchique ou les clients contactent seulement les serveurs en dessus d'eux, par exemple les serveurs DNS.

2.3.2 Le serveur

Les machines dotées d'un système d'exploitation adapté à la gestion du réseau sont appelé Serveur, ces stations de travail ayant ce rôle sont généralement de puissants ordinateurs. A son tour le serveur communique comme prévu avec le client par la couche application du modèle OSI.

Parmi les services offerts par le serveur on cite :

La gestion des données, elle consiste à conserver les données enregistrées par les utilisateurs en leur attribuant un espace disque suffisant.

Le partage d'information, est le fait de pouvoir stocker un ensemble de fichiers et de logiciels mise à disposition des clients.

L'administration du réseau, permet la communication entre les clients.

La sécurité, assure l'intégrité et la confidentialité des informations, ou encore l'intrusion d'un virus dans le système. Les parades contre les pirates sont multiples, par exemple les pare-feu, l'antivirus… etc. [2]

Caractéristique serveur [3]

- Le serveur sert à héberger des services, il est éventuellement possible qu'il soit spécialisé en serveur d'applications, de fichier, de terminaux ou de messagerie électronique.
- Si il est en attente d'une requête, il est passif (esclave).

- Il est disposé de répondre aux requêtes des clients lorsqu'il est à l'écoute.
- A la réception d'une requête il se met au traitement pour y répondre.
- Il peut répondre à plusieurs requêtes de divers clients simultanément.
- Il est contrôleur d'accès et responsable à l'intégrité globale.
- Aucun impact sur les clients lors de la mise à niveau des serveurs du moment où l'interface des messages reste la même.

2.4 Avantage et inconvénients

L'architecture centralisée présente avant tout un avantage sur deux plans simplicité et puissance de calcul, c'est-à-dire les interactions entre l'utilisateur et le serveur sont assurés par le client, il a pour tâche de s'en charger du traitement inhérent à ces interactions et envoie les requêtes au serveur. Le serveur, quant à lui, exécute les tâches correspondantes après avoir traité les requêtes de tous les clients. [2]

Ses inconvénients se résument à son non-Scalabilité, le besoin du serveur à avoir assez de ressources peut être problématique notamment en terme de coût afin d'être capable de supporter un grand nombre de pairs. Sans négliger qu'il présente un point unique de défaillance, une panne du serveur et c'est tout le système qui se trouve en panne ce qui fait sa vulnérabilité face aux attaques tel que le déni de service. [4]

2.5 Caractéristiques Client / Serveur

Les éléments qui caractérisent une architecture client-serveur sont [5]:

Service, la relation entre l'ensemble des processus qui tourne sur de multiple machine sont dites modèle client-serveur, le client consomme ce que le serveur fournie comme services.

Partage de ressources, Un serveur assure la gestion et le contrôle d'accès aux ressources de plusieurs clients.

Protocole asymétrique, l'asymétrie du protocole de communication est due au partage de ressources, décrite par le scénario à sens unique qui réside entre le client et le serveur, le client débute l'interaction, le serveur attend les requêtes des clients.

Transparence de la localisation, peu importe que le service soit sur la même machine ou accessible par le réseau l'architecture client-serveur doit dissimuler au client la localisation du serveur.

Message, les échanges entre client et serveur se font par messages.

Evolution, est le fait que l'architecture client-serveur ait le pourvoir de garantir une éventuelle évolution aussi bien qu'horizontal (lorsque le nombre des clients augmente) que vertical (l'évolution d'à la fois le nombre et les caractéristiques des serveurs).

3. Le modèle pair-à-pair

3.1 Introduction à la notion pair-à-pair (Peer-to-Peer ''P2P'')

Le concept de l'ordinateur central puissant était le premier abord du développement informatique, il est le centre de toute communication entre terminaux. Un concept quasi-identique a vu le jour par l'évolution de ce dernier, le modèle client-serveur, un logiciel propriétaire assure la liaison des clients au serveur dans le réseau.

Ces dernières années, internet a vécu une croissance intéressante, les statistiques du trafic ont montré que le trafic pairs à pair a dominé internet avec un pourcentage 72% du trafic total.

L'utilisation des réseaux pair-à-pair est aujourd'hui en pleine croissance grâce à leurs avantages, caractéristiques et leur propagation dans tous les domaines. Pour cela plusieurs logiciels ont été développés. [6]

A l'inverse de ce que croient les gens le terme pair-à-pair ne se résume pas qu'au partage des fichiers, mais aussi dans l'intercommunication entre l'ensemble des machines de manière décentralisée pour acheminer le trafic des données et des services. [4]

3.2 Définition

Au cours de notre étude, nous avons trouvé plusieurs définitions pour les systèmes pair-à-pair, nous citons ici quelques-uns :

- Le pair-à-pair (P2P : Peer-to-Peer) est un réseau informatique dans lequel chaque nœud du réseau est un serveur et un client à la fois. L'information est répartie sur l'ensemble des machines qui composent le réseau pair à pair. [7]
- Les systèmes pair-à-pair permettent à plusieurs machines de communiquer via un réseau, comme il permet aussi à tous les pairs de jouer les deux rôles client et serveur à la fois, de plus chacun de ces nœuds permet de télécharger des services ou ressources fourni par un autre nœud dans le réseau. [4]
- Les réseaux pair-à-pair (égal à égal) offrent l'opportunité d'avoir une communication et une collaboration des nœuds de manière explicite, ces derniers permettent aussi la décentralisation du réseau, le partage de l'ensemble des ressources du réseau pair-à-pair sans transiter par un serveur ou un pair central. [6]

3.3 Principe et fonctionnement

Le système pair-à-pair est composer par un ensemble de machine, ces machine ont besoin d'un logiciel particulier car ces derniers sont très importants pour sa mise en œuvre, ces logiciels sont capable de fonctionner à la fois en mode client et en mode serveur. Cette particularité est l'une des avantages de ce system, à partir de là, chaque pair est capable d'envoyer et récupérer des données. De plus ce system comporte un nombre important de nœuds qui collaborent l'un avec l'autre, par conséquent plus le nombre de services disponibles dans ce système est grand plus la probabilité de trouver un service est élevé, ce system nous permet aussi de partager l'information facilement et empêcher plus au moins les attaques légales ou pirates. Le modèle pair-à-pair permet la décentralisation des services et mettre à disposition des ressources dans le réseau, appelées objets. Dont chaque nœud du réseau a la possibilité d'en publier et d'en obtenir sur le réseau. [8]

Ce modèle a deux méthodes pour réaliser la tâche principale du réseau, qui est le partage et l'exploitation des ressources du réseau.

- La méthode centralisée est basée sur un serveur qui possède la liste des fichiers partagés et qui oriente les utilisateurs vers un utilisateur possédant le fichier souhaité.

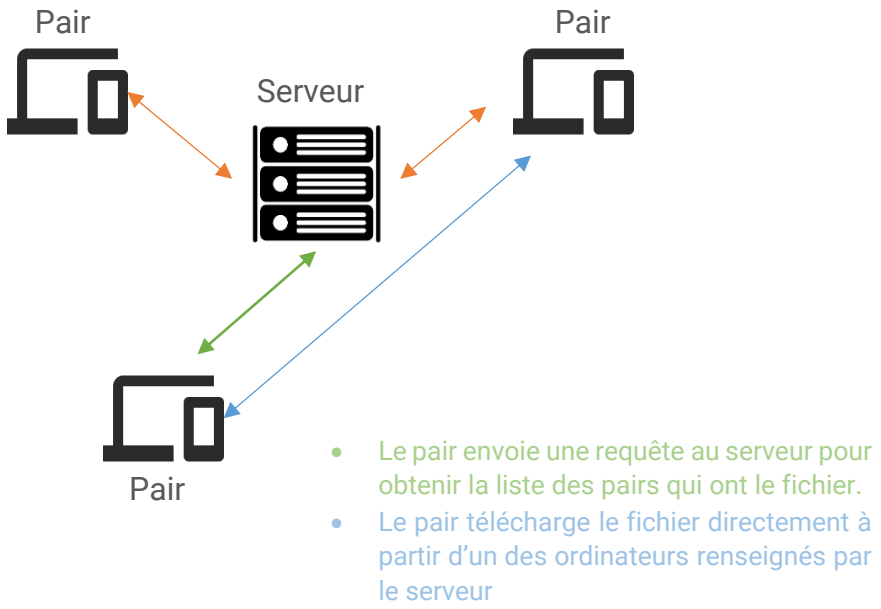


Figure 2. pair-a-pair approche centralisée

- La méthode décentralisée utilise chaque utilisateur comme un mini-serveur et ne possède aucun serveur fixe. Cette méthode a l'avantage de répartir les responsabilités. [5]

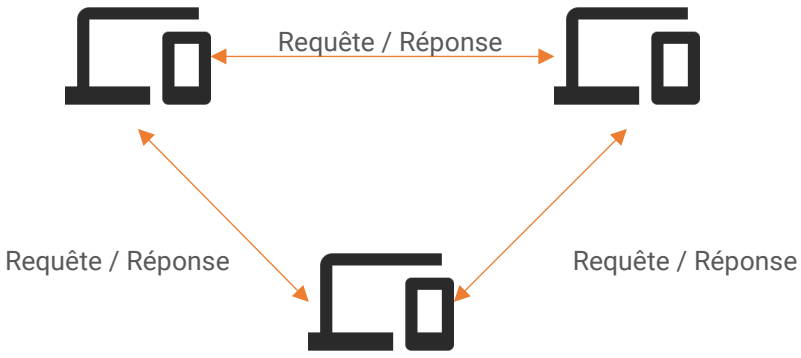


Figure 3. Pair-à-pair

3.4 Avantages du P2P :

- Décentralisation des ressources
- Les communications entre les paires sont directes
- Passage à l'échelle : c'est-à-dire servir un nombre important de nœuds (jusqu'à des milliers ou des millions) pour partager leurs ressources tout en maintenant une bonne performance du système.
- Tolérance aux fautes
- Possibilité de créer des groupes
- Si une machine tombe en panne, cela ne remet pas en cause l'ensemble du système.
- Le réseau est faiblement couplé.
- Connectivité occasionnelle.
- La réplication, redondance des données. [9]

3.5 Inconvénient du p2p :

- Pas de Qualité du Service (QoS)
- Problèmes de sécurité
- Les temps de localisation sont plus longs
- Les nœuds et connexions sont non fiables. [9]

3.6 Caractéristique du réseau P2P

Parmi les principales caractéristiques du modèle pair-a-pair on cite [8] :

Décentralisation, c'est-à-dire, chaque nœud gouverne ses propres ressources ce qui permet la décentralisation du contrôle, par la suite le système est capable de fonctionner sans la nécessité d'une administration centrale ce qui empêche les goulets d'étranglements et d'augmenter la tolérance du système aux pannes et aux défaillances.

Passage à l'échelle, elle consiste à la contribution de nombreux nœuds (jusqu'à des milliers voire des millions) afin de partager leurs ressources en garantissant la performance du système. De ce fait le système doit offrir des méthodes adéquates avec l'environnement où les données à partager sont volumineux, résultat d'un

échange important de messages suite au partage des ressources d'un grand nombre de nœuds dans un réseau largement distribué.

L'auto-organisation, pour cela il suffit de rejoindre un point d'accès via un nœud déjà connecté afin de se connecter en retour dans le système, désormais avec le déploiement des systèmes P2P sur internet, cette démarche est plus facile par rapport au cout qui ne demande pas une infrastructure couteuse. Il se doit au système pair-à-pair d'être ouvert, autrement dit, il est primordiale qu'un utilisateur sur un nœud ait la capacité de connecter son nœud sans avoir à faire à l'intermédiaire d'une seconde personne ni par une autorité centrale.

Autonomie des nœuds, la gestion des ressources propres aux nœuds est autonome, Libre au nœud de décider quelle partie de ses données veut partager d'autant qu'il est libre de se connecter et se déconnecter par sa volonté. D'ailleurs son autonomie ne se limite pas qu'à ça mais il est également de sa portée de gérer sa puissance de calcul et sa capacité de stockage.

Hétérogénéité, le système pair-à-pair dispose de techniques adaptées afin de contrer les problèmes liés à l'hétérogénéité des ressources engendrés par l'autonomie des nœuds ayant des architectures aussi bien matérielles que logicielles hétérogènes.

Dynamisme, encore une fois l'autonomie pose problème, le fait que chaque nœud peut quitter le système à sa guise, ses ressources du système disparaissent avec lui, et vice versa lorsqu'un nouveau nœud se connecte, ses ressources s'ajoutent au système, donc l'instabilité des nœuds a fait que le système pair-à-pair soit dans l'obligation de pouvoir gérer cette forte variation du nombre de ressources. Comme elle se doit également de tolérer ou minimiser l'impact d'une éventuelle panne d'un nœud sur la performance de tout le système.

4. Conclusion

Habituellement, la technique client-serveur appuie l'échange de service entre ordinateurs, selon elle, il n'y a de place que pour une seule entité centrale très puissante, c'est le serveur, quant aux restes des entités, souvent de puissances inférieurs, c'est des clients. Seul le serveur fournit des services aux clients, le client n'a qu'à consommer les services exécutés par le serveur sans avoir à partager ses propres ressources. Tandis que l'architecture pair-à-pair advienne comme une solution de rechange de par ses multiples avantages par rapport au modèle client-serveur. Le tableau ci-dessous montre la comparaison des deux modèles.

critère	**Modèle client-serveur**	**Modèle pair-à-pair**
Gestion	Supervisé	Auto-organisé
Présence	Permanente	Ad Hoc
Accès aux ressources	Recherche	Découverte
Organisation	Hiérarchique	Distribuée
Mobilité	Statique	Mobile
Disponibilité	Dépendante du serveur	Indépendante des pairs
Nommage	DNS	Indépendant
Modèle de programmation	RPC	Asynchrone

Tableau 1. Comparaison des infrastructures client/serveur et P2P

II. CHAPITRE

Les réseaux pair-à-pair

1. Introduction

Ces dernières années les réseaux pair-à-pair ont constaté une évolution perceptible au point que les nouvelles versions du système pair-à-pair mettent à disposition à leurs utilisateurs des applications de « télévision sur internet » et « vidéo-à-la-demande ». De nos jours avoir recours à l'utilisation des réseaux pair-à-pair est devenu une nécessité, que ce soit pour un usage public ou bien pour des applications scientifiques.

Les échanges de messages entre pairs dans un réseau pair-à-pair sont assurés par son protocole englobant toutes les règles. Ainsi toutes activités est encadré : recherche des adresses IP, gestion et contrôle des fragments de fichiers échangés, etc. toutefois l'installation de ces protocoles génère un trafic très important d'information sur ces réseaux, d'où l'apparition de deux types de réseaux pair-à-pair. Les réseaux non-structuré dont le choix des liens entre pairs se fait de manière arbitraire, les réseaux structurés, eux ont pour but d'optimiser les protocoles selon divers critères (quantité de données échangées, complexité de recherche) en s'appuyant sur des topologies ayant des propriétés mathématiques favorables. [10]

2. Classification des architectures pair-à-pair

Les systèmes informatiques peuvent être classé en deux catégories différentes systèmes distribués et systèmes centralisés, ces derniers à leur tour sont divisés en deux modèles, le modèle client/serveur et le modèle pair à pair, le modèle pair à pair est aussi divisé en deux réseaux, les réseaux structurés et non structurés.

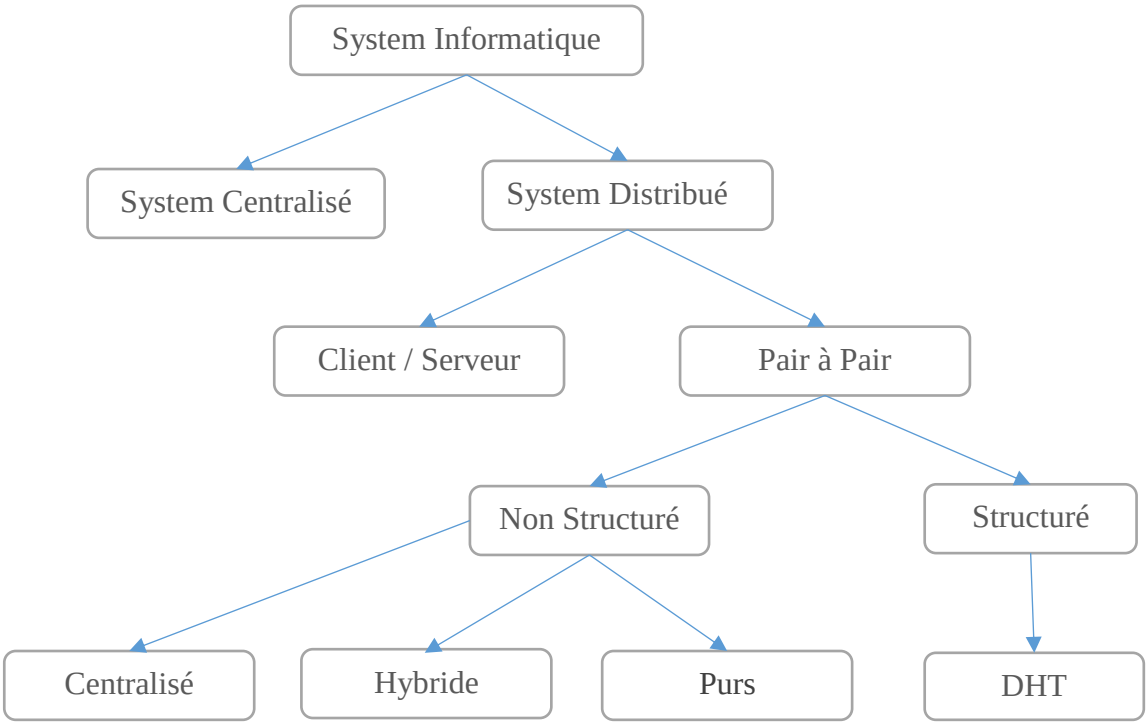


Figure 4 : Classification de l'architecture P2P

2.1 Réseaux non structures :

2.1.1 Définition

Dans un réseau pair à pair non structuré les liens entre les pairs sont basés sur des liens physiques, ce dernier a la possibilité de supporter des requêtes plus complexes avec des métas informations. La connexion d'un nouveau pair se fait par un pair (intermédiaire) déjà connecté dans le réseau, malgré cela ce system n'arrive pas à localiser les fichiers impopulaires. [4]

2.1.2 Approche centralisée

La première génération du système pair à pair a gardé le concept de centralisation. La particularité de ce modèle par rapport au modèle client/serveur est dans la récupération d'objets qui est décentralisé et qui s'effectue directement entre les pairs sans passer par le serveur central. Ce dernier contient aucune ressources il possède juste un annuaire qui contient tous les informations concernant la description des ressources partagées (nom, taille, chemin…), ainsi que des informations sur les pairs qui les hébergent (nom du pair,

adresse IP, nombre de fichiers partagés), l'exemple le plus courant reposant sur ce modèle est Napster (system de partage de fichiers MP3). [8]

Napster

En 1999, Shawn Fanning un étudiant à l'université de boston aidé par Ritter et Sean Parker, créent et développent une application de partage de fichiers musicaux mp3 sur le réseau internet, Napster est le premier réseau pair-à-pair grand public qui adopte le modèle pair à pair centralisé, cela signifie que ça façon de fonctionner est la même que l'architecture pair-à-pair centralisée.[4], [8]

Comment fonctionne Napster ?

Tout d'abord tous les utilisateurs doivent avoir à la fois le logiciel et une connexion internet pour exécuter ce dernier. Après avoir une connexion entre les clients et le serveur, les clients doivent annoncer au serveur leur ressources (fichiers, adresse IP, port) pour que les autres clients puissent les contacter, Par la suite le serveur maintient un annuaire qui contient la liste des clients connectés et aussi des informations sur ces clients en particulier les fichiers partagées, les clients doivent contacter le serveur napster en envoyant une requête. S'ils veulent chercher un fichier dans le réseau, quand le serveur reçoit une requête il renvoie aux clients une liste des réponses qui contient adresse IP, nom du client ,la taille du fichier.., après le client demandeur établit une connexion direct avec le client qui lui convient le mieux parmi la liste des clients possédant le fichier désiré, il envoie par la suite au pair choisi son adresse IP et le nom du fichier voulu, l'échange du fichier se fait directement entre eux. [11]

Avantages d'un système centralisé [4], [6] :

- Avantages habituels d'un serveur central : facile à administrer et à contrôler et aussi à fermer.
- Evite les recherches coûteuses sur le réseau : pas de routage ni de planification de la gestion des utilisateurs.
- Tolérance aux fautes : par un sondage régulier des pairs connectés, état cohérent.

Inconvénient d'un système centralisé [4], [6]:

- Les limites habituelles d'un serveur central : l'attaque ou la panne du serveur qui rend le serveur non disponible dans le réseau.
- Problème de passage à l'échèle : saturation de la bande passante et du nombre de processus.
- Pas d'anonymat car chaque pair est connu du serveur et des pairs sur lesquels il télécharge.

2.1.3 Approche décentralisée (Purs)

Dans cette architecture tous les nœuds sont égaux et jouent le même rôle et communiquent entre eux, c'est-à-dire, ils s'en passent du serveur, chaque élément du réseau est appelé servent qui est le rassemblement de deux mots client et serveur, la question qui se pose dans cette approche est comment un servent peut découvrir et accéder à une ressource dans ce type d'architecture ? La technique d'inondation en est la solution. [8]

Recherche par inondation

Cette solution a pour but de découvrir une ressource dans le réseau par la transmission d'une requête d'un servent à autre jusqu'à arriver au servent qui dispose de l'objet voulu. Pour éviter l'inondation du réseau durant un temps trop long, la requête est associé par un temporisateur TTL "Time To Live". Généralement la valeur du TTL est allouée à 7, si la valeur de ce dernier arrive à zéro la requête n'est plus renvoyée, mais l'inconvénient majeur de cette solution est lorsque le TTL expire c'est-à-dire il arrive à la valeur zéro sans être sûr de traverser tous les éléments du réseau, ce qui peut conclure une recherche par un échec malgré la disponibilité de l'objet dans le réseau. Cette technique est utilisée par le Protocol Gnutella. [8]

Gnutella

Gnutella est un protocole de partage de fichiers qui repose sur l'architecture pair à pair non structurée, succédant à Napster, crée en mars 2000 par Justin Fränkel et Tom Pepper, le réseau est composé par un ensemble de pairs connectant le réseau dont la transmission des informations entre les servents (ou nœuds) du réseau est basé en moyenne sur 5 descripteur (voir le tableau). Si un client veut rejoindre le réseau Gnutella,

premièrement il envoie une trame d'identification (PING) a ses voisins qui eux-mêmes la transmettront à leurs tours et ainsi de suite. Lorsque le TTL devient 0 la retransmission s'arrête, avec cette technique d'inondation des messages, on peut vite tomber dans le problème des boucles, pour éviter ce soucie, la trame va être stocké pendant un court instant lorsque le nœud reçoit la trame Ping. Si un nœud reçoit une trame identique pendant ce laps de temps il la rejette car elle est déjà reçue, par la suite chaque client qui reçoit une requête Ping répond avec une requête Pong qui contient l'adresse IP, le numéro de port, le nombre et la taille des fichiers partagés. Avec tout ça le client peut identifier les nœuds connectés dans le réseau. [8]

Tableau 2 descripteurs (requêtes) principaux utilisés dans le protocole Gnutella

Type	**Description**	**Information**
Ping	Annonce la disponibilité, et lance une recherche de pair	Vide
Pong	Réponse à une requête Ping	Adresse IP, N° port, Nombre et Taille de fichiers partagés
Query	Requête visant à trouver un ou plusieurs fichiers	Bande passante, adresse IP et numéro du port, nom du fichier
QueryHit	Une réponse à un Query si on possède le ressource	Adresse IP, numéro du port, bande passante, taille de fichier
Push	Demande de téléchargement pour les pairs derrière un firewall	IP, Index du fichier demandé numéro du port…

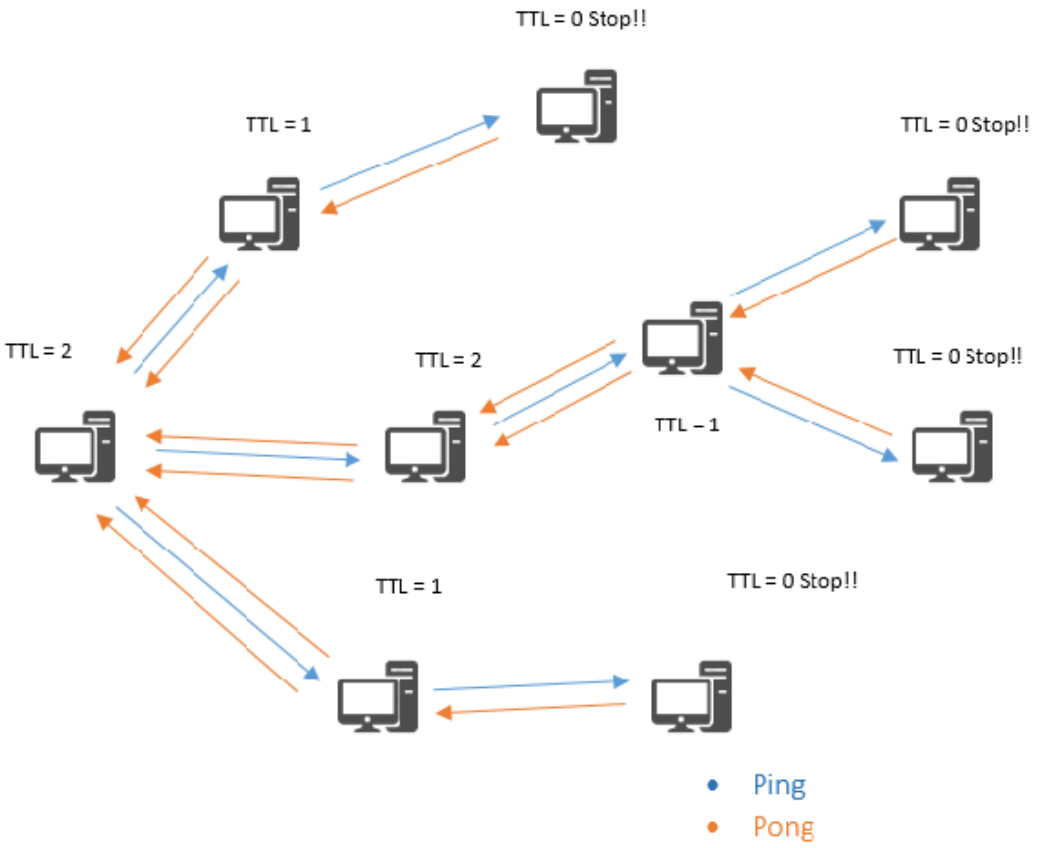


Figure 5 : la découverte des pairs avec Ping et Pong

La recherche

La recherche des ressources dans le réseau Gnutella se lance quand l'un des servents envoie un descripteur Query en spécifiant la vitesse de transfert minimum et les critères de recherche. Si un servent reçoit une trame de type Query et s'il dispose de la ressource, il renvoie une requête de réponse QueryHit au voisin qui lui a retransmis la requête, spécifiant son adresse IP et son numéro de port TCP ou l'objet peut être téléchargé, la réponse va remonter jusqu'au servent qui a demandé la ressource. ce dernier va sélectionner les fichiers à télécharger et envoyer une requête de téléchargement au servent qui possède la ressource, Il existe un autre descripteur Push est utilisé si les ressources sont derrière un pare feu. [8]

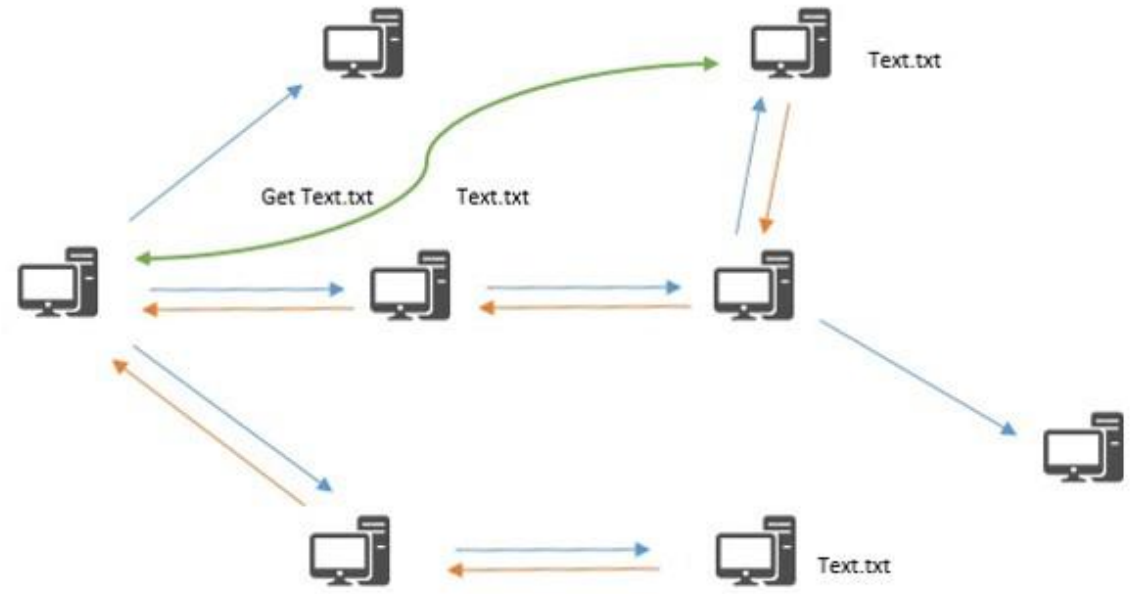


Figure 6 : recherche des fichiers sur un réseau Gnutella

Avantages d'un système décentralisé

- Administration simple.
- Topologie évolutive.
- Disponibilité du réseau, on peut l'arrêter.
- Simplicité du mécanisme de routage et de recherche.
- Réseau tolérant aux fautes.
- S'adapte bien à la dynamique du réseau.

Inconvénient d'un système décentralisé

- Pas de sécurité.
- Problème du free-riding (personnes ne partageant pas de fichiers).
- La recherche non déterministe à cause de la limite du TTL.
- Gros consommateur de Bande passante.
- Ne supporte pas le passage à l'échelle parce que le réseau est rapidement inondé par des Ping et des Pong.

2.1.4 Approche Hybride

Ce modèle est une combinaison entre le modèle centralisé et le modèle décentralisé, autrement dit ce dernier est organisé d'une manière hiérarchique. On distingue deux types de nœuds, le nœud standard et le super nœud, la liaison entre eux se fait de manière centralisée, chaque nœud standard se connecte à un super nœud. L'interconnexion entre les supers nœuds est située en haut niveau de la hiérarchie selon le modèle décentralisé, FastTrack est un exemple typique de protocole P2P partiellement centralisé. Dans FastTrack les super nœuds ont un rôle particulier et sont caractérisés par une forte capacité de calcul et une large bande passante etc. Ces super-nœuds ont la possibilité de gérer des groupe de nœuds (nœuds standard moins puissants) ou ils servent de serveur local dont les objets partagés par les nœuds standards sont enregistrés sur ces derniers (voir la figure ci-dessous), de nombreuses applications utilisent ce modèle, par exemple Kazaa, Bit Torrent. [11], [12]

Remarque : ce modèle permet de diminuer le nombre de messages reçus par une grande partie des nœuds ordinaires en augmentant la charge des super nœuds, cela permet de réaliser la recherche entre eux en un temps plus court sachant que le nombre de super nœuds est beaucoup plus petit que le nombre total de pairs.

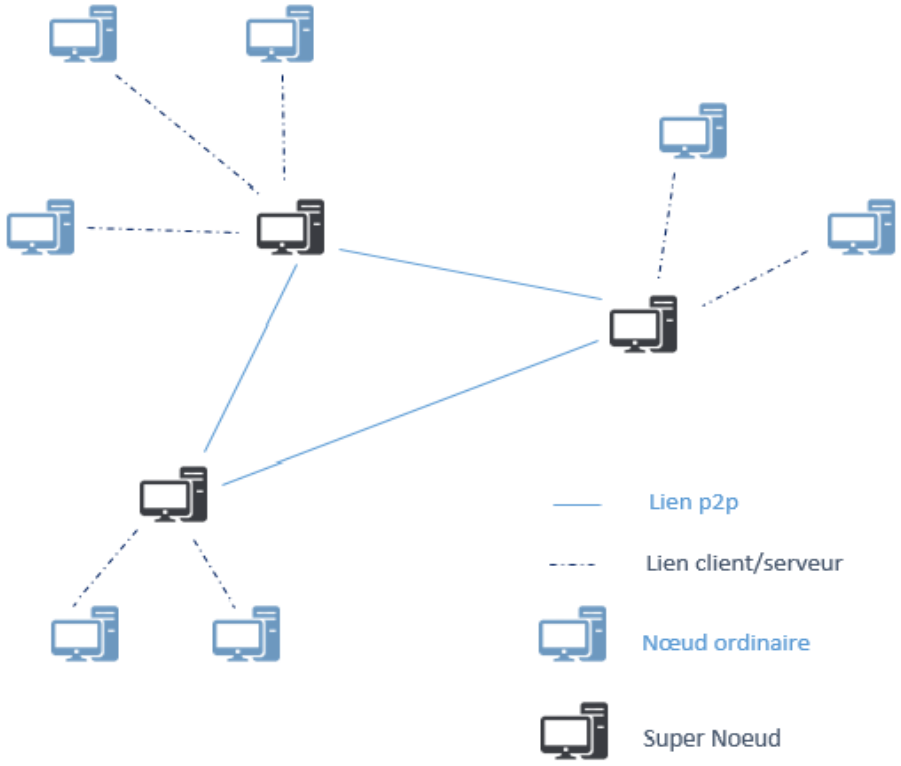


Figure 7 : architecture hybride

Kazaa

Comme on a dit précédemment le réseau est basé sur deux entités les pairs ordinaires et les super-pairs qui sont important dans le fonctionnement du réseau car ces derniers en plus de leur rôle spécifique à savoir l'hébergement de la liste des fichiers partagés par les clients peuvent aussi partager et télécharger des fichiers comme les autres pairs ordinaires.

La connexion des pairs est faite par le contacte du pair au super pairs actifs dans le réseau, il lui envoie la liste des fichiers qu'il veut partager afin qu'ils soient indexés, ainsi il lui envoie des requêtes pour but d'obtenir les fichiers souhaités. Dès que le contacte est établit entre les super-pairs et le pair ordinaire, le pair va obtenir l'adresse IP du pair qui dispose du fichier désiré. Les adresses IP fournie par le super pair sont cherchés en deux étapes, en première, le super-pair doit chercher dans sa propre indexation local(la liste des fichiers hébergés) cette étape dans le cas centralisé, la deuxième, il envoie des requêtes aux autres super-pairs dont la communication entre ces

derniers repose sur l'inondation des messages, les données restent toujours distribuées sur les pairs et aussi les échanges entre les pairs reposent sur le protocole http. [8]

Avantages d'un système hybride [13]

- Présence d'un serveur central : facile à administrer, et donc facile à contrôler
- Evite les recherches couteuses sur le réseau : pas de routage et planification de la gestion des utilisateurs
- Tolérance aux fautes en sondant régulièrement les pairs connectes et en maintenant un état cohérent
- Service novateur qui généralise le pair à pair

Inconvénient d'un système hybride [13]

- Pas d'anonymat partout car chaque pair est connu du serveur et des pairs sur lesquels il télécharge.
- Limites habituelles d'un serveur central : problème de disponibilité, de passage à l´échelle (saturation de la bande passante et du nombre de processus). Le cas de Napster : facile à fermer.
- Certains pairs peuvent mentir sur leur débit pour ne pas être sollicités.

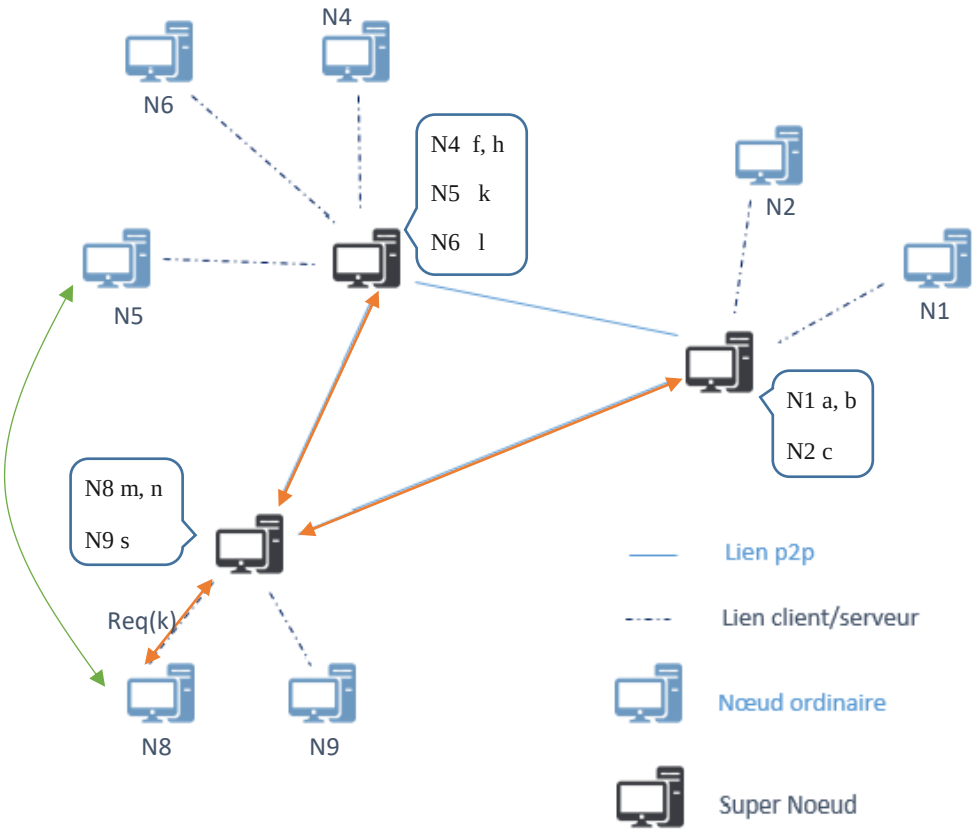


Figure 8 : la recherche dans kazaa

2.2 Protocoles réseau structuré

2.2.1 Définition

Un réseau structuré repose sur la structure logique du réseau, les nœuds sont reliés par un réseau recouvrant construit sous certaines contraintes, répondant à plusieurs propriétés et connectant les pairs selon une structure particulière donnée. La topologie virtuelle est assurée par des méthodes de routage et de localisation, chaque topologie a ses propres méthodes de routage, de nommage et de recherche

2.2.2 CAN

CAN est l'un des protocoles structurés du réseau pair-à-pair, il se sert d'une topologie à N dimensions. Les ressources sont réparties dans une zone multidimensionnelle, Chaque pair couvre une zone bornée par des coordonnées. Une zone contient les ressources partagées par le pair. Si on prend l'exemple d'un repère à deux dimensions, chaque pair prend en charge une zone définie par deux coordonnées,

minimales et maximales, le pair se doit de maintenir à jour sa liste des voisins (nord, est, sud, ouest). [10]

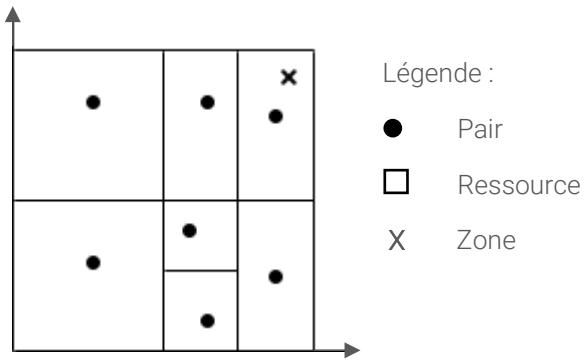


Figure 9.CAN

Lookup

Cet algorithme a pour tâche de trouver un pair sur le réseau, cela dit pour trouver celui qui gère les coordonnées cherchées, il faut d'abord pouvoir router les messages dans le réseau. Pour ce faire, le routage passe par trois étapes :

1. Si les coordonnées cherchées se situent dans le pair en question, le message de réponse est donc directement envoyer.
2. Sinon, les différentes dimensions sont parcourues en commençant par la première, dans le cas où la cordonnée cherchée est contenue par le pair, il passe à la seconde dimension, autrement il envoie le message au voisin le plus proche de la coordonnée cherchée sur l'axe courant.

3. Si l'on en vient à avoir de nombreux voisins pour une dimension et une direction, alors le message est envoyé au voisin le proche de la coordonnée recherchée sur la dimension d'après. [10]

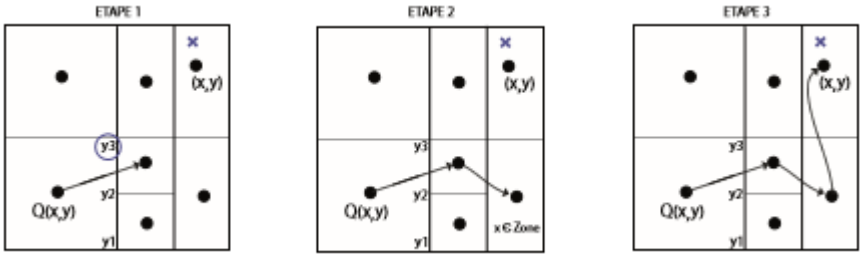


Figure 10 Exemple de routage d'un message

Join

Afin de permettre au pair de joindre le réseau, la fonctionnalité d'adhésion s'appuie sur l'algorithme de Split, lorsqu'un nouveau pair arrive sur le réseau il va falloir scinder l'espace géré. L'algorithme défini dans CAN passe par plusieurs étapes. Pour que le nouveau pair puisse se positionner dans le réseau, il faut :

- Choisir des coordonnées de manière aléatoire.
- Trouver le pair ayant la gestion de ces coordonnées sur le réseau en se servant d'un mécanisme de routage.
- Diviser la zone du pair trouvé et distribuer les ressources.
- Mettre à jour les voisins du pair trouvé de l'ajout d'un nouveau pair. [10]

Leave

C'est un algorithme qui assure le départ d'un pair du réseau de façon propre, sans aucune répercussion sur les informations, à cet instant le pair se doit de distribuer ses ressources à ses pairs voisins. D'abord il informe ses voisins de son départ pour que le pair dont il est issu lors de son arrivé puisse recouvrir sa zone, donc fusionner avec sa zone. [10]

Figure 11 Algorithme protocole CAN : Leave

2.2.3 Chord

Contrairement à CAN, Chord utilise une topologie en anneau. Il dispose notamment d'un algorithme d'une complexité d'au plus *O (log N)* requêtes, il utilise le principe des tables de hachage distribuées afin de trouver une information dans un anneau de *N* éléments. Cette complexité est due à la liste des pairs successeurs en puissance de deux dont chaque pair maintient. [10]

L'espace d'adressage

Le hachage de Chord consistent à attribuer à chaque nœud et chaque donnée un identifiant ID à *m* bits, où *m* est un paramètre prédéfinie de système. Les ID se situent dans un intervalle de 0 à 2^m -1. Les nœuds sont ordonnés dans un cercle d'identifiant modulo 2^m, l'identifiant d'une donnée est caché dans le nœud qui la succède, c'est le prochain nœud dans le cercle dans le sens des aiguilles d'une montre. [4]

Mécanisme de routage

Si un nœud cherche une clé, il utilise l'algorithme suivant :

- Il cherche parmi ses fingers le nœud dont l'identifiant est le plus grand tout étant inférieur à la clé et lui transmet le message.
- Lorsque le nœud reçoit le message, il use du même algorithme.
- Dans la recherche la distance entre le nœud destination et le nœud courant est divisé par un facteur, au moins 2 à chaque étape afin de d'avoir une complexité de sauts *O* (log *N*). [4]

Arrivé d'un nœud

À l'arrivé d'un nœud, Chord utilise d'abord le hachage pour but de générer son ID, ensuite il contacte le nœud du Bootstrap (qui existe déjà dans le réseau) pour chercher le successeur de son ID. Par conséquent le nœud successeur devient le successeur du nouveau nœud dans le réseau, la table de routage du nouveau nœud est remplie de manière correcte grâce au protocole de stabilisation.

Le Boostraping comprend les démarches suivantes :

- Le nœud arrivant doit utiliser le hachage pour générer son id
- Il doit contacter un nœud déjà existant dans les réseaux pour chercher son successeur. Le nouveau nœud utilise le protocole de la stabilisation pour corriger les tables des routages ce protocole est utilisé périodiquement en arrière-plan
- Construction de sa table de routage.

Le protocole de stabilisation inclue deux fonctions principales :

Stabilize (): permet aux nœuds de connaitre les nouveaux nœuds et de mettre à jour leur successeur et prédécesseur. Un nœud réalise ceci en demandant à son successeur Ns son prédécesseur Nj. Si Nj=Ni, alors Ni prend Nj comme étant son successeur. Ceci arrive si Nj est un nouveau nœud dans le réseau. Supposant que $i < j < s$ et Ni et Ns sont des nœuds existant dans le réseau. Quand Nj rejoint le réseau pour la première fois, il cherche j et trouve l'adresse de Ns. Il prend alors Ns comme étant son successeur. Et finalement, Ns transfert les données (leurs Ids) qui sont dans l'intervalle]i, j] vers Nj.

Fixfingers (): assure que les tables de routage (finger) soient mises à jour correctement. [4]

La recherche des données :

L'algorithme de recherche envoie la requête nœud par nœud en servant des liens de successeurs jusqu'à ce que le successeur de la donnée soit atteint. La longueur du chemin de recherche est sensée être O(n) sauts mais avec la table de routage (finger) réduit considérablement la longueur en O(long N) sauts.Une requête de recherche est routée avec une table de routage tout au long de l'espace d'adressage. Une fois la requête reçu, en premier le nœud vérifie si l'ID de la donnée recherchée est entre l'ID du nœud

actuel et l'ID de son successeur. Si c'est le cas, alors il retourne son successeur comme destination et termine la recherche. Dans le cas contraire, le nœud actuel relais la requête au nœud de sa table de routage dont le Id est le plus proche, mais précédant le Id de la donnée recherchée. Le processus de relai procède récursivement (ou itérativement) jusqu'à ce que le nœud destination soit trouvé. [4]

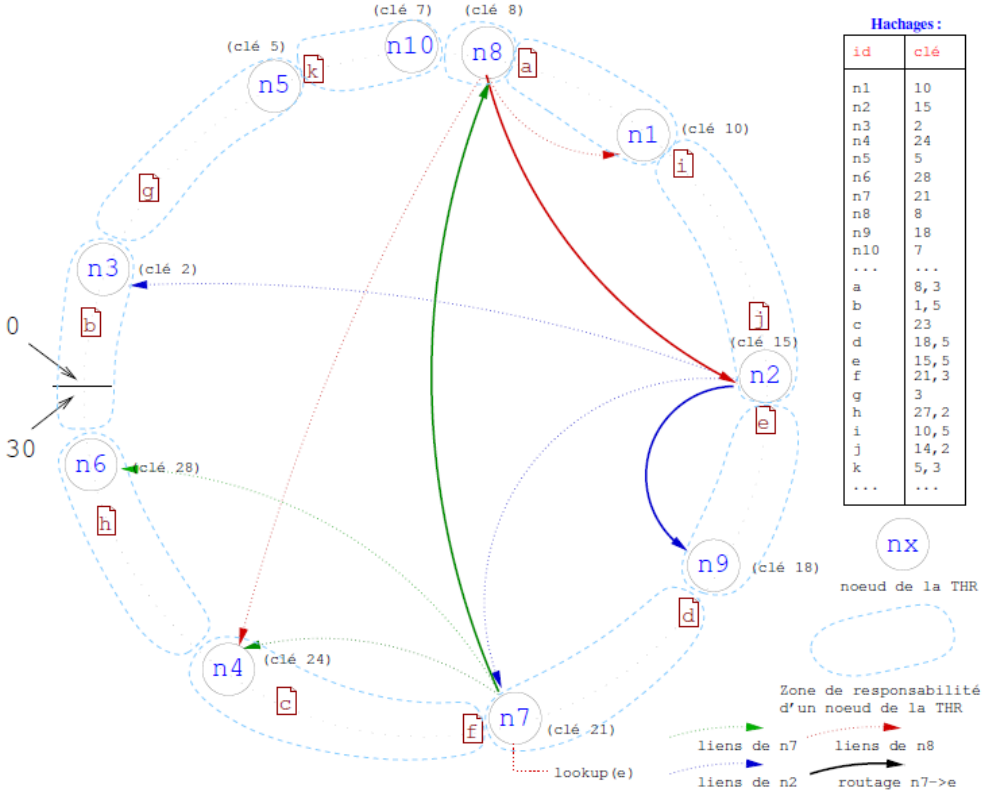


Figure 12 : algorithme de recherche Chord

2.2.4 Pastry

Dans le protocole Pasty la table de hachage distribué (DHT) mise en œuvre est similaire à Chord sur un réseau pair-à-pair en anneau virtuel. Chaque nœud assure la gestion des informations dont les clés sont numériquement proches de son propre identifiant, dans ce protocole l'espace d'identifiant des nœuds et des clés sont confondus c.à.d. dans le même espace.

Dans la mesure où les nœuds rejoignent ou quittent l'anneau à n'importe quel instant, le protocole se doit de poursuivre les requêtes peu importe de quel nœud proviennent avec peu ou pas de risque de perte de données, et répondre par l'information recherchée au nœud qui est à l'origine de cette requête. Il est également à sa portée d'utiliser une métrique de routage fournie par un programme externe, tel que *Ping* ou *traceroute*, pour qu'il puisse déterminer les meilleurs itinéraires à stocker dans sa table de routage, ce protocole réalise un système complètement décentralisé, scalable et fiable.

Fonctionnement

A l'évidence Pastry se distingue non pas par sa fonctionnalité de table de hachage qui est pratiquement ordinaire, identique à celle des autres DHT mais par le réseau de routage de recouvrement construit sur ce concept de DHT, ce qui présente à Pastry un atout d'évolution et de tolérance aux pannes devançant ainsi les autres réseaux, tout en minimisant le cout global d'acheminement de paquet d'un nœud à autre, en contournant la notion d'inondation. Ceci est assuré par une métrique de routage fournie par un programme externe basée sur l'adresse IP du nœud cible, et peut être facilement changée a un nombre de sauts plus petit, une latence plus faible, une bande passante plus grande ou encore à une combinaison générale de métriques.

L'espace des clés de la table de hachage est circulaire semblable à celui de Chord, les identifiants sont des entiers non-signés à 128 bits qui représentent leur position dans cet espace. Les ID de nœuds sont choisis de manière aléatoire et uniforme afin que les ID de pairs adjacents puissent être géographiquement éloignés. Chaque pair contribue à établir en dessus de la table de hachage un réseau de recouvrement de routage en découvrant et échangeant des informations d'état comprenant une liste de nœuds feuille, une liste de voisin et une table de routage.

Liste des nœuds, c'est les L/2 pairs les plus proches à l'ID du nœud dans chaque direction autour du cercle.

Liste de voisinage, représente les M pairs les plus proches en termes de métrique de routage, bien qu'elle ne soit pas utilisée directement dans l'algorithme de routage, elle consiste à maintenir les directeurs de la localité dans la table de routage.

La table de routage, elle contient une entrée pour chaque bloc d'adresse qui lui est attribué, pour former les blocs d'adresses, la clé de 128 bits est divisée en chiffres de b bits de long, ce qui donne un système de numérotation a base 2^b. Cela partitionne les adresses en niveaux distincts du point de vue du client, le niveau zéro représente un préfixe commun à zéro chiffre entre deux adresses, niveau 1 un préfixe commun à un chiffre et ainsi de suite. La table de routage contient l'adresse du pair le plus proche connu pour chaque chiffre possible à chaque niveau de l'adresse, sauf pour le chiffre qui appartient au pair lui-même à ce niveau particulier. Cela se traduit par le stockage de $\boldsymbol{2b}$ - 1 contacts par niveau, avec le nombre de niveaux à l'échelle de ($\boldsymbol{log2}$ N)/b. Les valeurs de b ≈ 4, L ≈$\boldsymbol{2b}$ et M ≈$\boldsymbol{2b}$ représentent des valeurs opérationnelles sur un réseau typique. [14]

Routage

Le routage des paquets dans l'espace d'adressage dépend de l'attribution antérieur de l'ID au pair, le paquet prend sa place dans l'anneau circulaire au près du nœud ayant un ID plus proche que le sien, au cas où un poste reçoit ou veut envoyer un paquet, d'abord il vérifie l'ensemble de feuilles et l'envoie directement au nœud destinataire. Dans le cas contraire lors d'un échec, le prochain pair consulte sa table de routage afin de pouvoir trouver une adresse de nœud partageant un préfixe plus long avec l'adresse destinataire que le pair en question. Si l'homologue ne trouve pas de contacts ayant un préfixe plus long ou le contact est mort, il choisit un autre pair de sa liste de contact avec la même longueur du préfixe dont l'ID est numériquement plus proche que la destination et il lui envoie le paquet, étant donné que le chiffre correspondant à l'adresse est constant ou augmente, dans la mesure où il est constant la distance entre le paquet et sa destination diminue, par déduction le protocole de routage converge. [15]

Arrivée et départ d'un nœud

Habituellement, le premier nœud que contacte le nouveau nœud est supposé être près géographiquement, ce dernier prend comme table *neighborhood set* celle du premier nœud contacté. Le nœud qui envoie la requête insère dans la table de routage du nouveau nœud la ligne actuelle de sa propre table de routage (premier nœud insère la ligne n°0, deuxième nœud insère la ligne n°1,...), de même pour les autres qui se situe dans l'itinéraire de la requête, le dernier nœud insère sa table *leaf set* dans le nouveau nœud. Pour conclure le nouveau nœud met a jour chaque nœud qui se trouve dans sa table de routage ou *neighborhood set* ou *leaf set*.

Une défaillance d'un nœud est détectable grâce aux messages périodiques de vérification ou la non-réponse du nœud lors du routage d'une requête. Pour remplacer un nœud défaillant R_{xy} dans la table de routage, le nœud demande a un voisin de la même ligne Rxi la route vers Rxy ($i \neq y$). Si le problème n'est pas résolu, tester l'entrée de R_{zy} ($z = x+1$) dans la ligne suivante de la table de routage. Pour le *leaf set*, si un nœud du *leaf set* est défaillant le nœud actuel envoie une requête au nœud ayant l'ID le plus grand dans sa *leaf set* du côté du nœud défaillant. Cette requête lui demande d'envoyer sa propre table *leaf set* et à partir de cette table le nœud actuel choisi un nœud pour remplacer le nœud perdu. Pour la table de voisinage ou *neighborhood set*, la défaillance d'un nœud est détectée grâce aux messages périodiques. Une fois que la défaillance est détectée, le nœud actuel demande aux nœuds voisins de lui envoyer leurs propres tables de voisinage et choisi parmi les nœuds découverts celui qui est à la distance la plus petite. [15]

3. Conclusion

Ce chapitre aborde les différentes architectures des réseaux pair-à-pair, il en compte deux et présente trois types de réseau de chaque, le premier, le non-structuré comporte Napster, Gnutella et Kazaa sachant que chacun présente une approche à part, le second, le structuré, quant à lui, comporte Can, Chord et Pastry. Tout cela en développant sur leur fonctionnement, mécanisme de recherche, etc.

III. CHAPITRE

La plateforme JXTA

1. Introduction

L'origine du mot Jxta est juxtapose, désignant pour placer deux entités côte à côte ou à proximité. L'équipe de développement du Sun a choisi ce nom afin de reconnaitre que les solutions pair-à-pair existeraient en outre des solutions client-serveur courantes plutôt que de les remplacer complétement.

En avril 2001, Bill Joy a placé le projet JXTA sous responsabilité de la communauté de développement de pair-à-pair en adoptant une licence basé sur Apache Software License Version 1.1. En plus de maintenir les spécifications des protocoles JXTA v1.0. Désormais, le projet Jxta a une exécution de référence en java, avec des réalisations en C, Ruby et PERL. Le projet Jxta abrite de nombreux projet de la communauté Jxta dont ils touchent divers domaines tel que l'intelligence artificielle.

2. Les protocoles

Les spécifications de protocoles de JXTA v1.0 définissent les blocs fonctionnels de base et les protocoles de réseau pairà-pair [16] :

- **Peer Discovery Protocol** : Ce protocole s'appuie sur le protocole de résolution pour l'envoie de ses requêtes et des réponses associées. Il offre aux pairs de mettre à disposition leurs ressources et les découvrir sur un réseau Jxta, ces ressources sont décrites par une annonce. Chaque type de ce dernier possède des attributs qui sont utilisés lors de la publication ou la recherche de ces documents attribués aux ressources.
- **Peer Resolver Protocol** : il permet l'implémentation de services de plus haut niveau, exemple la couche applicatif. C'est à ce niveau de la spécification des protocoles qu'est introduite la capacité aux pairs de participer au mécanisme d'index distribué des ressources partagées. Sur un réseau de pairs rendez un tel index est réparti au sein d'un groupe permettant un meilleur routage des requêtes dans leur diffusion. Pour l'envoie et la réception de message le protocole de résolution s'appuie (implémente) sur le protocole de rendez-vous. Ce protocole est le second protocole noyau.
- **Rendez-vous Protocol** : ce protocole permet aux pairs ordinaires de s'abonner à un service de réception de messages propagés dans un groupe Jxta. De même il

est capable de diffuser des messages aux pairs membres d'un groupe. Aussi il permet d'organiser la structure du réseau logique des pairs rendez-vous dans le contexte d'un groupe, il repose sur le protocole de routage pour l'envoie et la réception des messages.

- **Peer Information Protocol** : il permet aux pairs d'obtenir des informations (la charge réseau, la durée de vie du pair, ses capacités, etc) à propos des autres pairs d'un réseau Jxta, il repose également sur le protocole de résolution.
- **Pipe Binding Protocol** : Ce protocole permet de crée une communication avec un ou plusieurs pairs via un canal de communication virtuel c'est-à-dire il est utilisé pour mettre en relation les extrémités d'un canal de communication par leur(s) résolutions en adresse réseau. Ce mécanisme est le cœur du fonctionnement des communications entre pairs dans la spécification Jxta. Il utilise le protocole de résolution pour déterminer le(s) pair(s) en écoute sur l'extrémité d'un canal.
- **Endpoint Routing Protocol** : Ce protocole permet aux pairs de découvrir la route à suivre pour envoyer un message à un autre pair. Ce mécanisme est utilisé lorsqu'il n'existe pas de route directe entre deux pairs souhaitant communiquer. Il va alors permettre de trouver le(s) pair(s) par lequel le message doit transiter. Ce protocole est l'un des deux protocoles noyau.

3. Architecture de Jxta

L'architecture de JXTA est compose de trois couches [17] sont représentées sur la figure :

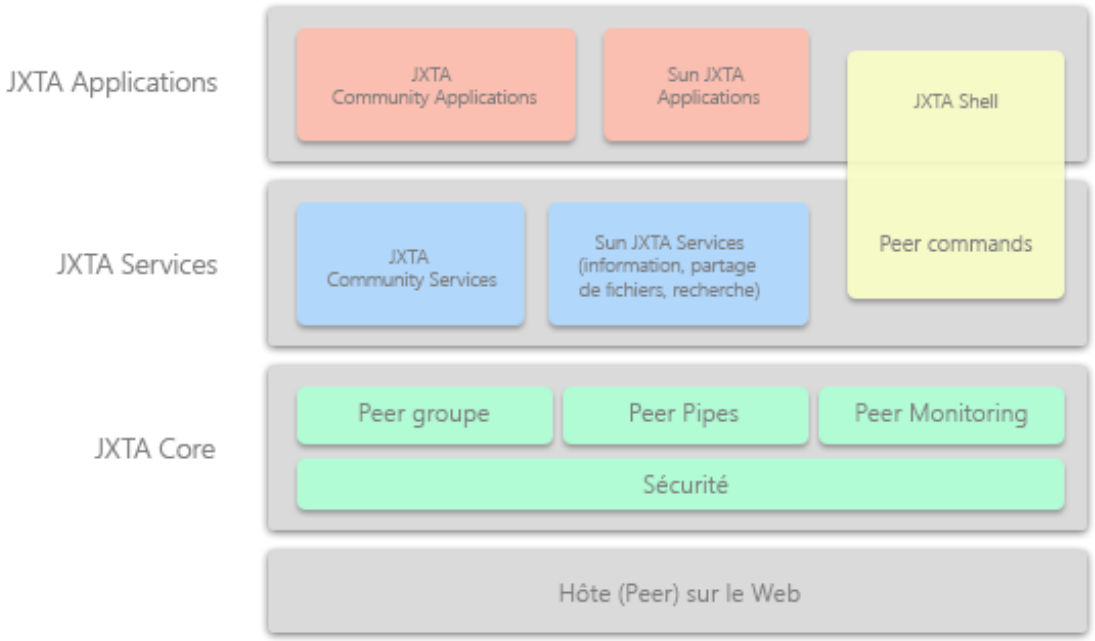


Figure 13 : Architecture de la plateforme JXTA

Noyau de JXTA (JXTA Core) : les primitifs minimaux et essentiels tel que le pair, groupe de pair qui sont communs à des réseaux pair-à-pair sont encapsulé par le noyau de JXTA ,de même ce dernier inclut des modules de fonctionnement des principaux mécanismes pour des applications pair-à-pair y compris la découverte, les transports de communication (y compris Firewall et Nat), la création et la gestion des pairs et des groupes de pair, et aussi les primitifs de sécurités.

Couche de services : Cette couche est au-dessus de la couche noyau, elle inclut les services de réseau qui ne peuvent pas être absolument nécessaires pour le fonctionnement de réseau pair-à-pair, mais être commune ou souhaitable dans un environnement pair-à-pair. Les exemples des services de réseau y compris la recherche et l'indexation, les systèmes de stockage, les systèmes de distribution de fichiers, authentification.

Couche application : la couche application inclut l'exécution des applications intégrées, telles que la communication de messages instantanée, partage de données la gestion et la livraison de données…etc.

4. Objectifs

Parmi les objectifs du JXTA, fournir les mécanismes de base pour la construction du système pair-à-pair, exemple, la communication et la découverte des ressources entre pairs, etc. il a pour objectif de devenir une plate-forme générique pour la conception du système pair-à-pair, indépendante de toutes architectures ou langages.

Il suffit au JXTA de définir uniquement le format des messages émis par chaque protocole spécifié sans avoir à utiliser des algorithmes d'implémentation des protocoles ni même les interfaces de programmation associées. Par conséquent, libre aux implémentations de la spécification JXTA d'utiliser une technique particulière, telle que la découverte de ressources sur les réseaux virtuel ou alors la propagation des messages. La force de JXTA réside donc dans sa plate-forme générique qui sert à cadrer l'évaluation de nouveaux algorithmes pair-à-pair.

Il offre également l'interopérabilité aux applications pair-a-pair et permet de réduire le coût de développement de tels systèmes. Actuellement les principaux systèmes pair-à-pair en particulier les systèmes de partage de fichiers, usent des protocoles spécifiques et généralement propriétaires. Or, le temps de développement de tels protocoles est relativement long et donc couteux, par contre dans JXTA la collaboration entre les différents systèmes est nettement simple puisque ils reposent sur la même plate-forme. En outre, il permet de construire une application P2P sur des briques de base déjà existantes.

La mise en œuvre des implémentations de JXTA sur n'importe quel équipement électronique est assurée son par indépendance vis-à-vis des architectures matérielles et des langages. [16]

5. Concepts fondamentaux

Les principaux concepts auxquels les protocoles définis par JXTA font référence [16] :

5.1 Pair

L'entité de base d'un réseau JXTA, identifiée par une adresse logique de façon unique, appelée pair ID, indépendante de sa localisation dans le réseau physique, un pair correspond à un processus qui implémente les protocoles noyau de JXTA. On distingue trois types de pair :

Pair standard : il peut émettre et recevoir les requêtes. Il dispose aussi d'un cache local sous la forme d'une base de données. Ce cache sert à stocker des informations sur les ressources découvertes dans le réseau pair-à-pair.

Pair rendez-vous : typique aux caractéristiques du pair standard. En plus, il est responsable de l'indexation des ressources existantes sur le système pair-à-pair afin de les trouver efficacement, il a pour mission de propager les messages dans le système, il est donc assimilé à un super-pair.

Pair relai : il dispose également des mêmes caractéristiques que les pairs standards. En outre, il est capable de stocker des messages temporairement pour d'autres pairs. Aussi, il permet de franchir les pare-feu et d'autres systèmes de translation d'adresse (NAT). En effet, un pair derrière un pare-feu n'est pas accessible depuis l'extérieur, à moins qu'il initie la connexion, notamment vers un pair relai dont il lui transmet ses messages en attente. [16]

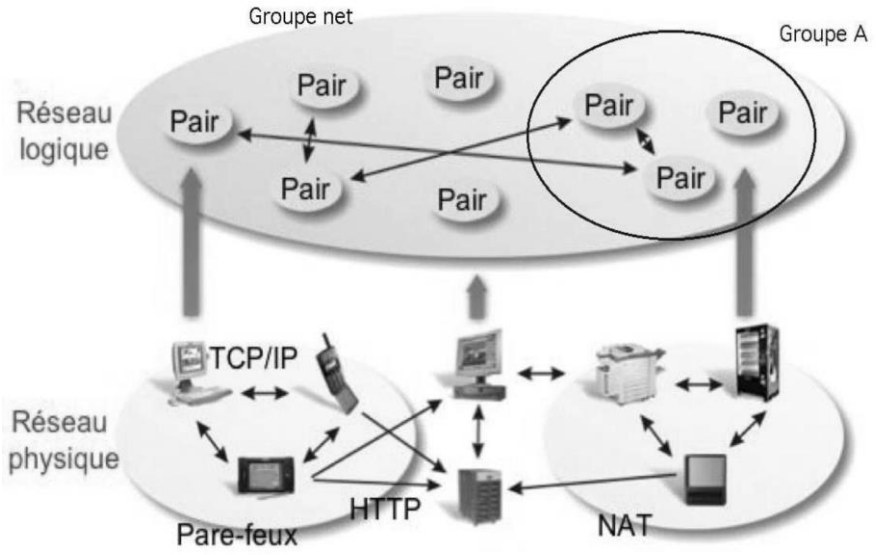


Figure 14 : Réseau virtuel JXTA au-dessus d'un réseau physique.

5.2 Groupe de pairs

C'est un groupe de pairs ayant des intérêts communs, identifié par un ID groupe unique. Un groupe définit un domaine, éventuellement sécurisé, ou les pairs peuvent s'échanger des messages. D'autre groupe peuvent se former dans un domaine. Une hiérarchie de groupe peut être instaurée pour raffiner un regroupement de pairs. Un pair peut rejoindre plusieurs groupes simultanément. Il est capable de prendre différents rôles (standard, rendez-vous, etc.) dans chacun des groupes dont il est membre. Tous les groupes JXTA sont créés dans le groupe *net*, tous les pairs JXTA peuvent donc s'échanger des messages, si l'identifiant d'un groupe est modifié, un groupe de pairs *net* privé et propre à eux est ainsi créé, utilisant cet identifiant. Voir la figure ci-dessus.

5.3 Annonce

Dans un réseau de JXTA les entités tels que (pair, groupes, services, pipe) sont représentés par les annonces qui sont éditées par les pairs sur le réseau. Ces annonces sont des documents XML sous la forme de métadonnées, qui possèdent des éléments avec l'information à propos de la ressource étant annoncée.

Il existe plusieurs types d'annonces prédéfinies par JXTA (les annonces de pairs, de groupes, de canaux etc...). Ces annonces sont utilisées par les protocoles JXTA dans les échanges de messages entre les pairs. Chaque annonce est publiée via le protocole de découverte avec une durée de vie précise (qui peut être étendue). Cette publication se fait toujours dans le contexte d'un groupe, et aussi peut être publiée dans un ou plusieurs groupes, citons un exemple d'annonce.

```
<? XML version="1.0"?>
<!DOCTYPE jxta:PGA>
<Jxta:PGA xmlns:jxta="http://jxta.org">
<GID>
urn:jxta:uuid-
AAA122616461AAAAAAA124615032503302
</GID>
<MSID>
urn: jxta:uuid-
DEADBEEFDEAFBABAFEEDBABE000000010306
</MSID>
<Name>
JPDA_ROOT_GROUP
</Name>
<Desc>
47
Root application group
</Desc>
</Jxta:PGA>
```

Figure 15 : annonce d'un groupe de pair

5.4 Canal virtuel

Ce canal est un moyen de communication entre les pairs, dont l'identification de ce dernier d'une manière unique par un *Pipe ID et* indépendamment de la localisation des pairs d'un groupe JXTA l'utilisant.

Un canal virtuel permet aux pairs souhaitant communiquer de s'abstraire de leur localisation physique, pareil pour la topologie réseau. Aussi, ils disposant d'une communication logique directe, même si de manière sous-jacente de multiples sauts peuvent être fait, des pare-feu traversés et différents protocoles de transports réseau. Parmi les protocoles définis de JXTA, le protocole de liaison

dont les deux extrémités du canal virtuel sont résolu par lui pour les associer physiquement à des pairs. Sur le pair émetteur un canal virtuel en sortie est créé pour envoyer les messages, l'autre côté le pair récepteur, un canal virtuel en entrée est créé pour but de recevoir les messages envoyés. Citons un exemple d'une annonce d'un canal virtuel.

```
<? xml version="1.0"?>
<!DOCTYPE jxta:PipeAdvertisement>
<jxta: Pipe Advertisement xmlns:jxta="http://jxta.org">
<Id>urn: jxta: uuid-5961629B8021441095C3AB1101C76DCC ...
04</Id>
<Type>JxtaUnicast</Type>
<Name>Pipe Test</Name>
</Jxta:PipeAdvertisement>
```

Figure 16 : Annonce d'un canal virtuel de type unidirectionnel ayant pour nom Pipe Test.

5.5 Service réseau

Un service réseau est une interface de programmation qui offre une ou plusieurs fonctionnalités particulières dans un groupe d'un réseau JXTA. La notion de service réseau n'est pas spécifiée de manière précise par JXTA, en particulier la manière de les invoquer. L'objectif est de rendre les standards actuels et futurs des services web compatibles et interopérables tel que WSDL (Web Services Description Langage).

Le WSDL sert à décrire :

- le protocole de communication (SOAP RPC ou SOAP orienté message)
- le format de messages requis pour communiquer avec ce service
- les méthodes que le client peut invoquer
- la localisation du service.

La description d'un service réseau dit JXTA se fait par une annonce de type *spécification de module*. Cette annonce contient un identifiant du canal virtuel pour que le service soit fonctionnel.

Il existe deux types de services réseau JXTA : le service de pair et le service de groupe.

Service de pair : seul le pair ayant publié un service a accès à ce service et ça dans les groupes à partir desquels la publication s'est faite. L'utilisation du service dépend de la disponibilité du pair.

Service de groupe : un service de groupe s'exécute sur plusieurs pairs, il se compose d'une série d'instances de ce service, il s'exécute sur plusieurs pairs qui coopèrent éventuellement pour fournir le service. Les services de groupes doivent tolérer la défaillance d'un des pairs du groupe.

Sachant que la notion de groupe dans JXTA est centrale, l'utilisation des services est bornée et une frontière est créée avec l'extérieur. L'ensemble des services dont chaque pair membre du groupe doit fournir est défini à l'intérieur de ces frontières.

6. Avantages et inconvénients de la plateforme JXTA

- ✓ Elle offre un ensemble de service de haut niveau permettant la création et l'utilisation des applications P2P
- ✓ Une grande variété de services, périphériques, langages de programmation, systèmes d'exploitation, technologies réseaux
- ✓ L'utilisation du format XML fournit un haut niveau d'abstraction
- ✓ L'indépendance de langage de programmation
- ✓ Plusieurs protocoles de transports sont utilisés
- Le haut niveau d'abstraction des spécifications de JXTA, les rend plus difficile à comprendre
- Le parsing multiple de message XML par les protocoles, services et application peut éventuellement nuire la performance globale d'un pair.
- Non fiabilité quant à l'émission et à la réception des messages de même que le principe généralisé du «best effort » [17]

7. Conclusion

JXTA est un projet Open source initié par Sun Microsoft, la spécification JXTA a pour but de fournir un environnement pair-à-pair générique, selon les besoins des applications, indépendamment des langages et des réseaux, JXTA permet de crée facilement des applications pair-à-pair interopérables. JXTA comporte un ensemble de protocoles qui sont à la disposition des développeurs des systèmes pair-à-pair, ces protocoles offrent des mécanismes utiles pour but de réaliser la découverte, l'organisation, la surveillance et les communications entre les pairs. Les développeurs ont la liberté d'implémenter un ou plusieurs protocoles pour réaliser des applications pair-à-pair. JXTA n'est pas restreint à la réalisation des systèmes pair-à-pair mais aussi des systèmes client/serveur, web service, RPC, RMI…etc.

JXTA est en évolution permanente, il existe des services en cours d'amélioration comme le Shell et la sécurité, toutefois JXTA à besoin des meilleurs performances pour faire ses preuves à grande échelle.

IV. CHAPITRE

Mise en œuvre de l'application

1. Introduction

L'application a pour objectif de résoudre un problème existant avec un protocole particulier relativement récent, le problème autrement dit 'besoin' se résume au partage de fichiers, et qui dit partage, dit recherche, plus une messagerie instantanée. Le tout en se servant du protocole JXTA, ce protocole qui a manifestement a fait ses preuves dans de nombreuses situations.

2. Réflexion

Etant donné que JXTA est open source, il a suffi de télécharger l'ensemble des librairies ayant un rôle dans la mise œuvre de l'application, on en compte deux essentielles, celle de JXTA et celle du CMS (Content Management Service). Le premier prend en charge la partie des créations des pairs, end-point, rendez-vous y compris la partie messagerie, le second prend en charge le partage et la recherche

3. Caractéristiques de l'application

- L'application cible le grand public, un utilisateur ordinaire disposant d'un ordinateur peut s'en procurer et tirer profit de ses services.
- L'application est développée en java, ceci élargie considérablement son potentiel d'abord en terme de fiabilité (la puissance du langage) mais bien plus encore en terme de perspective.
- L'application utilise une technologie open-source, la JXTA comme protocole de communication, la CMS, une librairie de partage / recherche de fichiers.
- L'application fonctionne dans un réseau local seulement.
- Parmi les services offerts par l'application, on cite le partage de fichier, bien évidemment seuls ceux ayant l'application et qui sont connectés au réseau sont capable d'y accédés, il y'a aussi la recherche des fichiers, pour ce faire il suffit de connaitre le titre approximative de ce qu'on souhaite trouver, pour terminer il y'a la messagerie instantanée, tout ce qu'il y'a de plus ordinaire, on précise que tout se déroule dans un réseau local.

- L'application dispose d'une interface graphique simple, intuitive ce qui rend son maniement instinctif.
- L'application repose sur un modèle pair-à-pair puisque par définition c'est un ensemble de protocoles pair-à-pair, par conséquent il hérite de ses avantages.

4. Démarche algorithmique

Dans un premier temps le programme ne fait qu'initialiser et créer les diverses composantes dont il a besoin afin de pouvoir à la fois communiquer ses informations et en recevoir, on va l'appeler la phase de création, les plus essentiels sont :

- Création du groupe (NetPearGroup).
- Création du module de la publicité (ModuleAdvertisement).
- Création du Pipe de publicité (PipeAdvertisement).
- Création d'un Pipe de communication.
- Création d'un service de découverte (getDiscoveryService, getPipeService).

Une fois terminé, le programme se met à l'écoute, si un message arrive, un évènement de déchiffrage se déclenche, dans le cas contraire, si on souhaite chercher un service, à partir du service de découverte se lance une recherche, d'abord local ensuite distante, si la recherche s'avère positive un lien est établie à travers la **Création du OutputPipe**, les messages peuvent donc être transmis (un éventuel échec de création n'est pas improbable).

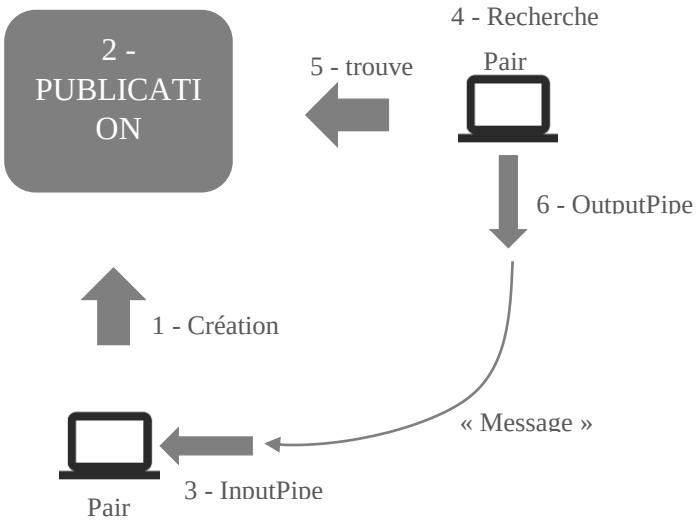


Figure 17 : étape de publication d'un service JXTA

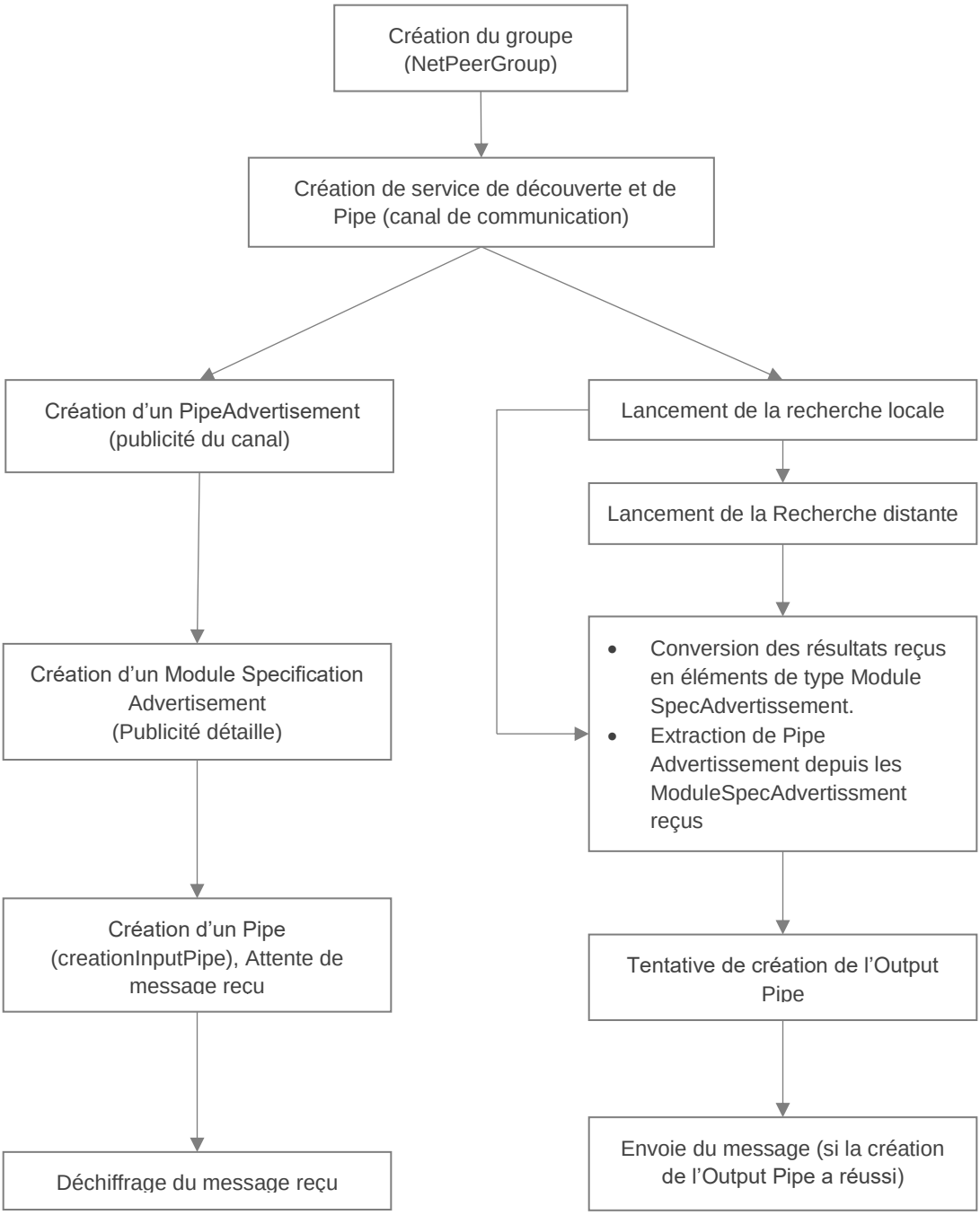


Figure 18 étape d'envoie / réception de message

Quant au partage de fichier est assuré par le service CMS, le CMS a pour objectif de garder un registre précis de tous les fichiers stockés sur un pair local qui peuvent être partagés. Pour que le CMS puisse optimiser l'utilisation des ressources du système, il nécessite que le contenu partagé reste sur le système du fichier. Il ne met pas en cache et ne fait pas de copies sur ses fichiers. Par contre, il crée une sorte de magasin ou toutes les informations du contenu partagé est inclut dedans, telle que les annonces du contenu. Le magasin est utilisé pour avoir un accès rapide à la liste du contenu partagé sur le pair. Sachant que chaque fichier partagé doit avoir une annonce unique qui décrit son contenu.

Le format de la publicité est représenté dans la figure ci-dessous,

```
<?xml version="1.0">
<!doctype jxta:contentAdvertisement>
<Jxta:contentAdvertisement>
<name>ship.html</name>
<cid>
 md5:2b9cbd6ab82c8fee8fe2a2b9e7eab7a85
</cid>
<Type>text/html</type>
<Length>1234</length>
<description>Page for Displaying Model
Ships
```

Figure 19 : un exemple d'une annonce d'un contenu CMS

Les éléments de la publicité :

- Name file : le nom du fichier qui va être partagé.
- Cid : l'ID représentant le fichier à partager, cet ID est unique.
- Type : Un type MIME pour le fichier partagé : JPG, PNG, TXT…etc.
- Length : la taille du fichier à partager (en bits.).
- Description : une description optionnelle du fichier à partager.

En ce qui concerne la récupération du contenu (annonce) d'un pair distant, le CMS (Content Management Service) utilise un canal JXTA pour la réception des requêtes et contenus. Un seul tube initial est ouvert pour chaque instanciation CMS. Ce tube unique à la faculté de permettre au système de recevoir d'autres publicités de canal afin d'ouvrir

des connexions supplémentaires en cas de besoin. Toujours au moment de l'instanciation, la méthode *init()* est invoqué, celle-là possède trois paramètre, le groupe de pair (Peer group), identifiant du pair (Peer Id) et l'annonce du pair (Advertissement).

Après l'instanciation du CMS, le pair peut donc partager ou chercher des fichiers dans le réseau. Deux opérations sont nécessaires pour partager un contenu, la première est d'obtenir le content Manager du service CMS. La seconde c'est d'appeler la méthode *share()*, elle comprend quatre paramètres :

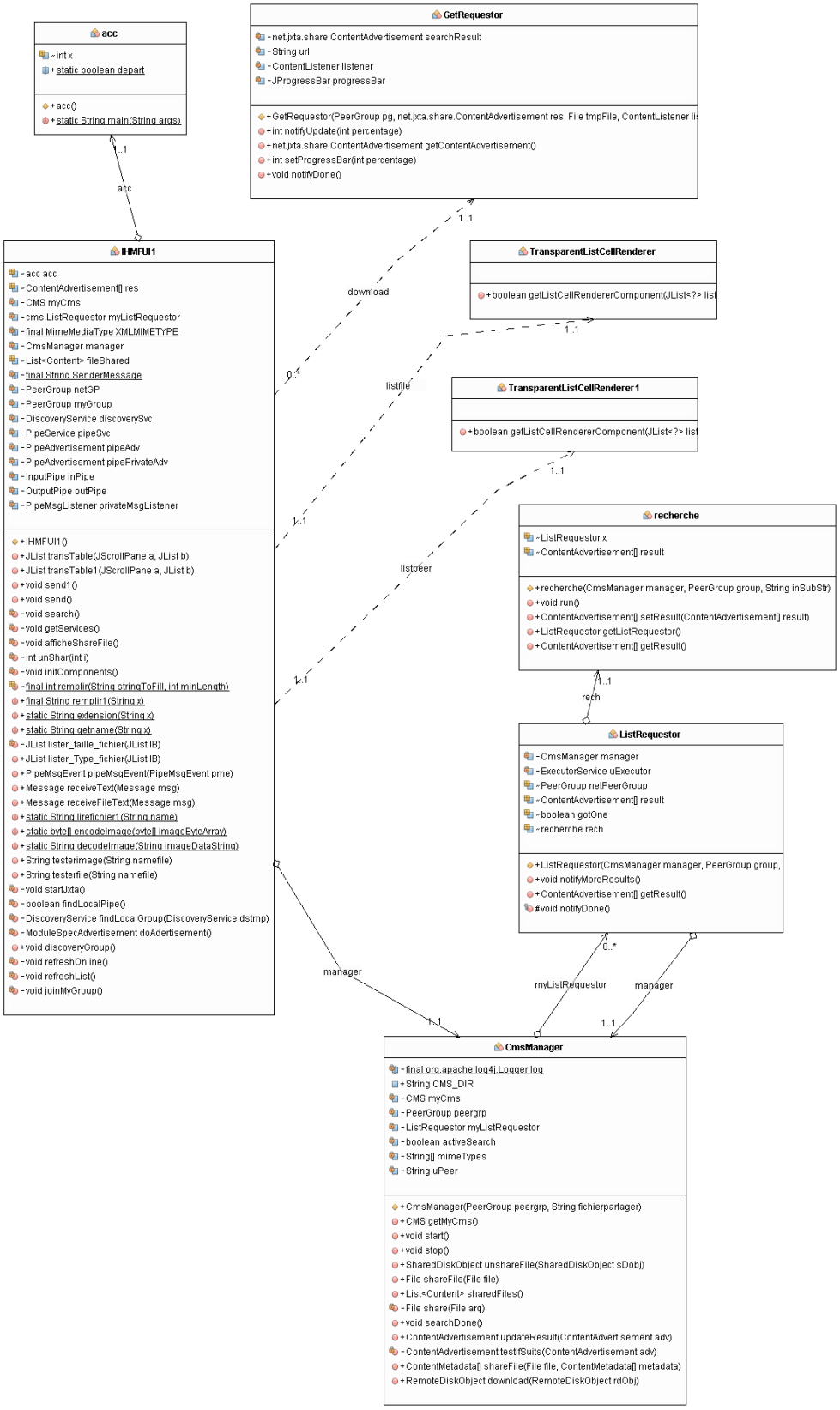
- **File Object** : le fichier à partager.
- **Name** : le nom du fichier (optionnel).
- **type** : le type MIME (exemple : jpg, txt), optionnel.
- **desc** : description du fichier (optionnelle).

Prototype de la méthode *share()* :

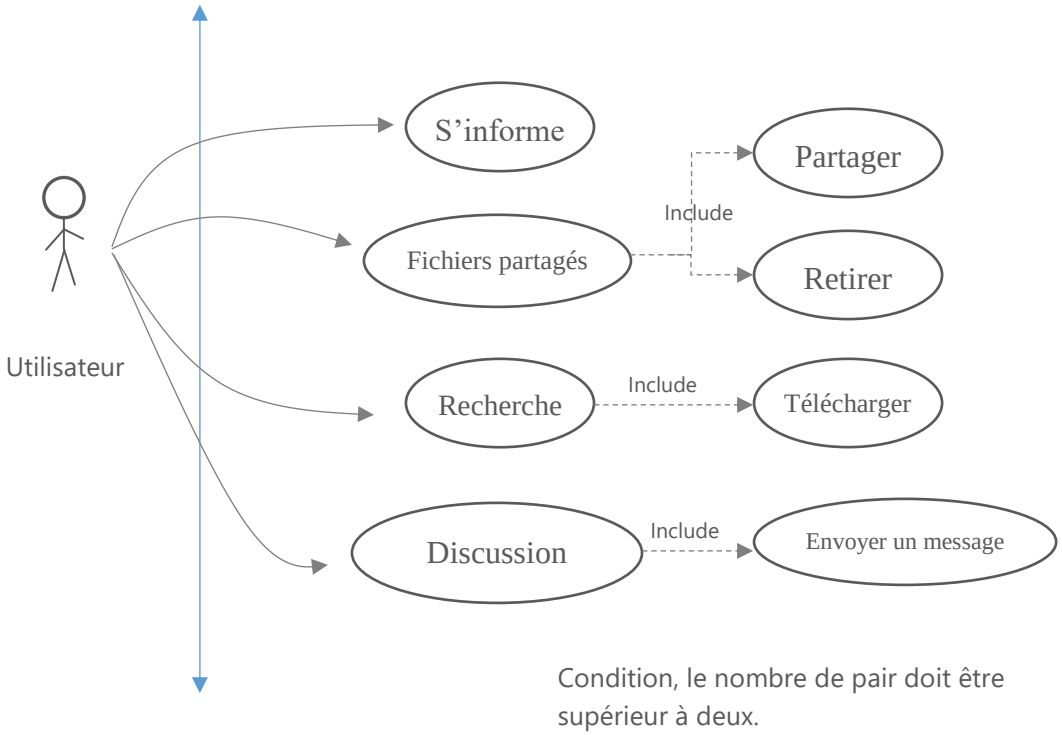
share(java.io.File, java.lang.String, java.lang.String, java.lang.String)

Enfin il y'a la recherche qui est réalisée au sommet des services JXTA traditionnels, y compris les tuyaux et les messages. Lorsqu'une requête est requise, un message LIST_REQ (Requête) est envoyé aux autres pairs pour obtenir en retour une liste des fichiers actuellement partagés. Tous les pairs distants répondent avec un message LIST_RES (Résultat) contenant une ou plusieurs publicités du contenu. Lorsque le pair en question sélectionne l'un des fichiers à télécharger, un message GET_REQ (Requête) spécifique est envoyé au pair distant, selon les informations des contenues dans sa publicité du fichier. La communication est réalisée en utilisant les protocoles JXTA.

4.1 Diagramme de classes



4.2 Digramme de cas d'utilisation



5. Interfaces et fonctionnement

L'application comporte une interface secondaire et une autre principale, par contre cette dernière en compte quatre volets, Chaque volet à ses propres taches. L'interface secondaire sert à présenter l'application

Figure 20 : fenêtre d'accueil

Contrairement à l'interface secondaire, l'interface principale comporte l'ensemble des services. Un menu est à disposition de l'utilisateur afin de pouvoir naviguer entre les volets, voir la figure ci-dessous :

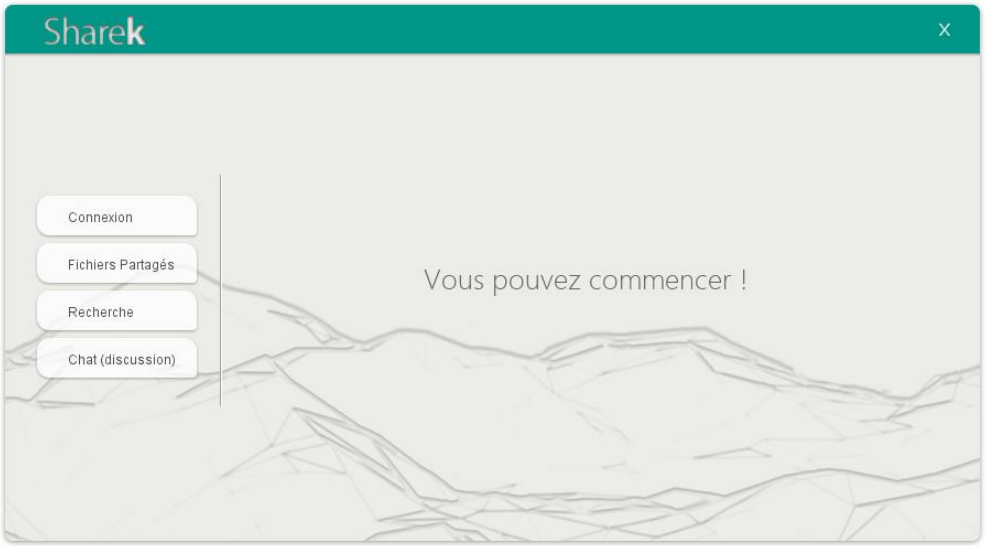


Figure 21 : menu de navigation.

Dans le menu on y trouve quatre boutons, le premier sert à renseigner l'utilisateur sur sa connexion (un utilisateur lambda n'a pas vraiment besoin de ce service). Le second sert à partager des fichiers ou alors en retirer. Le bouton chercher comme son nom l'indique sert à chercher toute sorte de fichier et en télécharger, quant au tout dernier bouton il sert à ouvrir une fenêtre de chat, avec une liste des clients connectés, il permet d'établir une connexion avec le client souhaité.

5.1 Le volet connexion

Dans le volet connexion, des informations sur la connexion JXTA sont affichées, vous pouvez ainsi vérifier si tout se déroule comme prévu. Par exemple si le pair a bien été créé, s'il est toujours connecté, si le service de découverte a bien été instancié… etc.

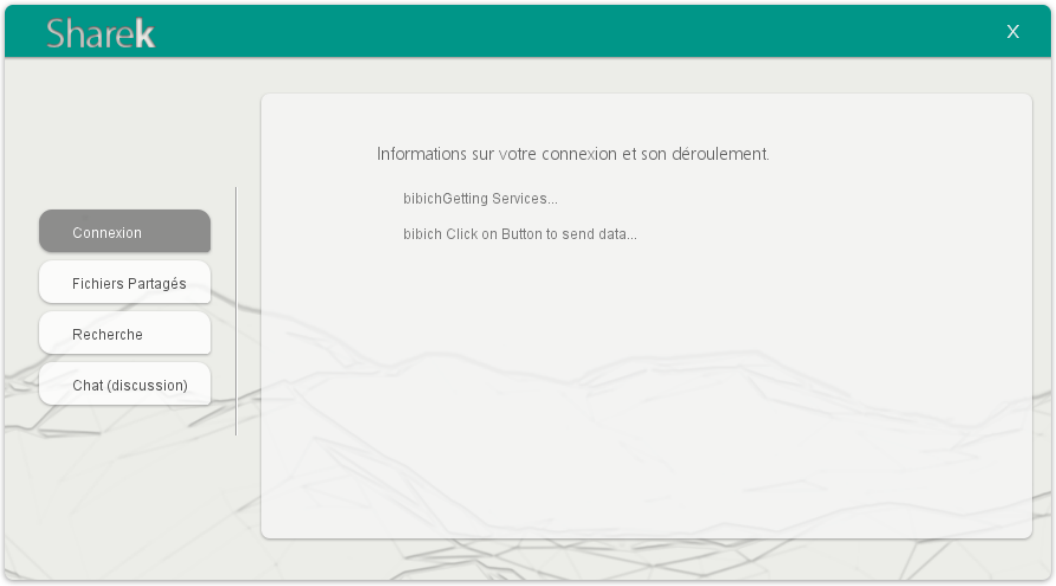


Figure 22 : le volet connexion.

5.2 Le volet fichiers partagés

Le volet ''fichiers partagés'' permet d'afficher l'ensemble des fichiers partagés avec certains détails comme le type du fichier (PDF, doc, mp3… etc.), en partager d'avantage et en en retirer au cas de besoin, Deux façons sont disponible pour partager les fichiers. On souligne que tout type de fichier peut être partagé et téléchargé.

La première, consiste à remplir un dossier, ce dossier est particulier, tout fichier appartenant à ce dossier est automatiquement partagé, le dossier en question s'intitule « client » ce dossier est créé lors de l'installation de l'application. À l'ouverture de l'application, le programme reprend le dossier et partage chaque document existant, on parle ici du boulot du CMS bien évidement.

La deuxième, est tout aussi simple que la première a une différence près, celle-là fonctionne lorsque l'application est ouverte. Il existe un bouton partager, ce dernier ouvre une fenêtre de vos dossiers afin de pouvoir choisir quel fichier désiriez-vous partager sans avoir à déplacer ou faire une copie des fichiers à partager, sachant que CMS non plus ne va ni déplacer ni faire une copie, il va reprendre le chemin du dossier et le mettre dans une annonce dans le fichier *shares.ser*. Au cas d'éventuel téléchargement, le client peut retrouver le chemin dans l'annonce et ainsi en télécharger le document souhaité.

Un bouton ''retirer'' assure le retrait des fichiers partagés. Au moment du clic, l'annonce (publicité) est retirée, le fichier disparait du réseau. A moins qu'il y'ait une autre copie partagée par un autre utilisateur disposant de l'application, ce qu'on appelle une réplique, plus le fichier est partagé, plus il est populaire plus il est disponible dans le réseau.

La figure ci-dessous montre l'ensemble de fonctionnalités citées.

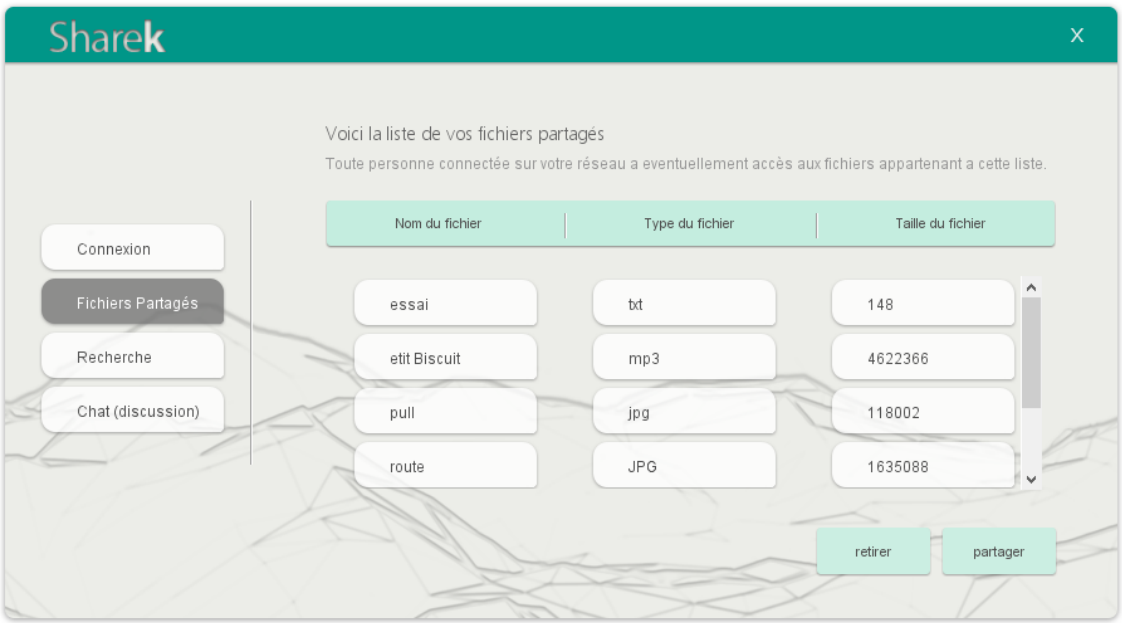


Figure 23 : le volet des fichiers partagès

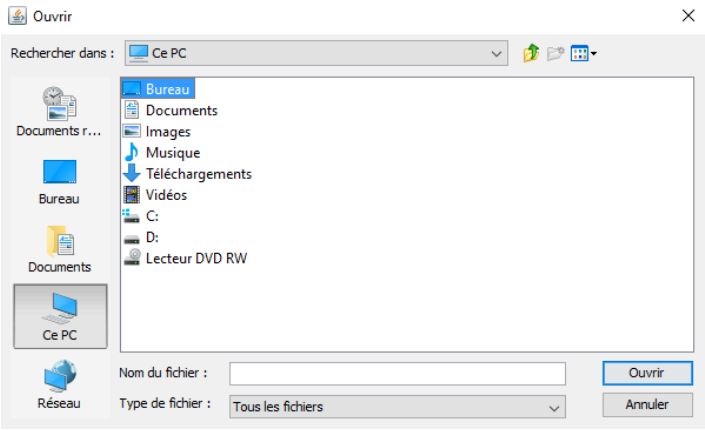


Figure 24 : une fenêtre pour parcourir les fichiers et sélectionner le fichier à partager

5.3 Le volet recherche

Le volet ‘’recherche’’ à son tour à un espace d’affiche pour lister le résultat d’une recherche (un bouton de recherche a pour tâche de lancer la recherche), il a aussi une entré (texte) pour éditer la recherche, pour ensuite télécharger le fichier souhaité si le résultat s’avère positif (un bouton télécharger a pour tâche de lancer le téléchargement après avoir indiqué le fichier désiré dans la liste du résultat de la recherche). Une barre de téléchargement se trouve juste en dessous de la barre d’entête de la liste des résultats, elle permet de suivre l’avancement du transfert. La figure suivante illustre l’explication

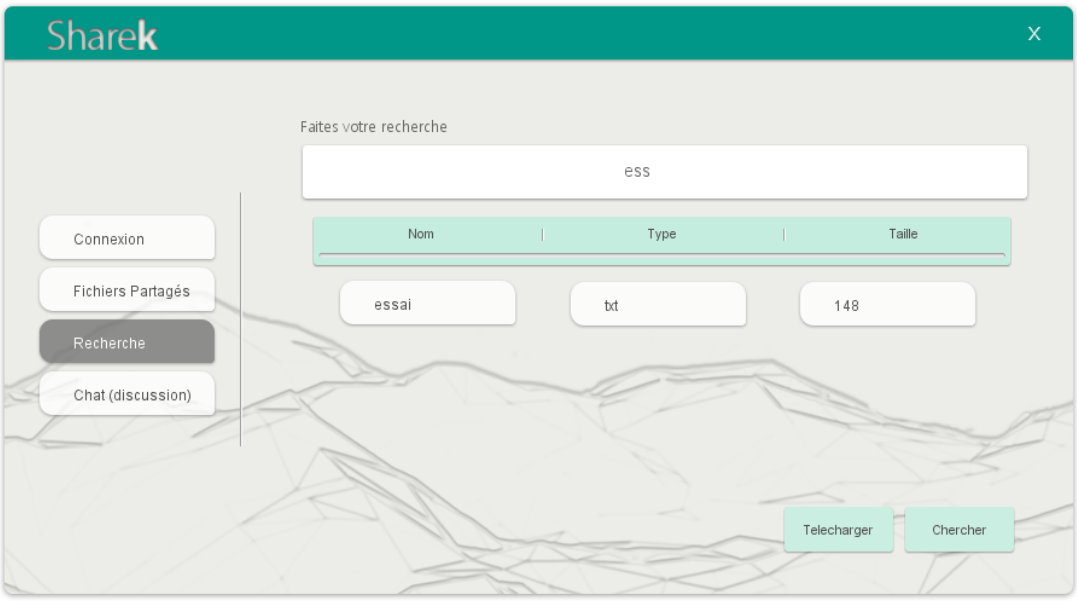


Figure 25 : le volet de recherche

5.4 Le volet chat

Le volet ‘’chat’’ contient une liste des pairs connectés dans le réseau, elle a la faculté de s’auto-actualiser, un espace pour écrire les messages à envoyer et un autre pour afficher celles qui sont reçu avec pour option auto-défilement. Pour commencer une discussion il suffit de choisir le pair avec qui on souhaite entretenir un dialogue, un message apparaitra indiquant que vous pourriez commencer votre discussion, Voir la figure 26.

Comme il est possible aussi d’envoyer des documents au cours de la discussion lorsque vous cliquez sur le bouton joint (celui avec l’icône d’un trombone) une fenêtre apparaitra afin de pouvoir choisir quel fichier souhaitez-vous envoyer, voir la figure 27.

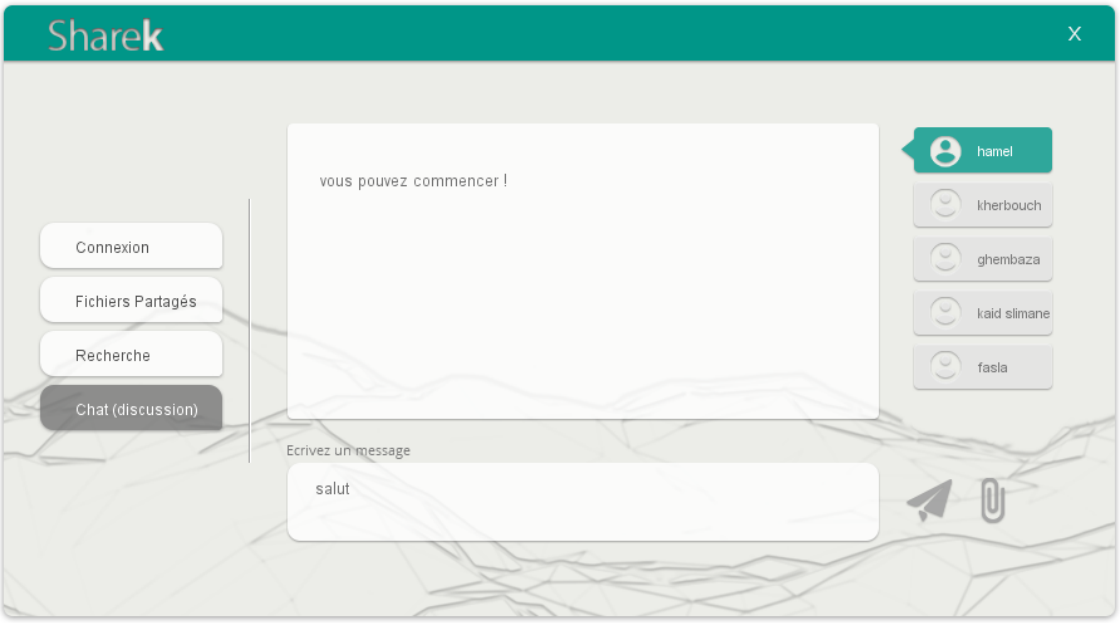


Figure 26 : le volet chat

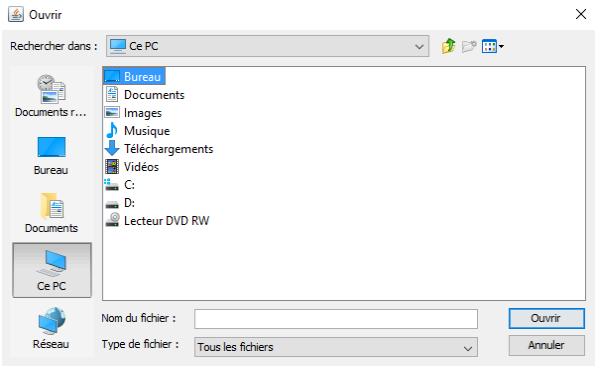


Figure 27 : fenêtre permettant de choisir un fichier à transmettre

6. Conclusion

Ce chapitre explique la mise en œuvre de l'application, il présente les caractéristiques, les différents outils utilisés pour la réalisation et le mode d'emploi de l'application, le chapitre présente à la fois l'aspect graphique et algorithmique de l'application. Les captures d'écran font l'objet d'illustration. Pour résumer, l'application est développée en java, elle s'appuie sur les protocoles JXTA, par conséquent elle se sert des protocoles pair-à-pair et implémente une librairie CMS pour faciliter le partage de fichiers. vis-à-vis d'un utilisateur lambda, l'apparence est plutôt agréable ce qui rend sa manipulation instinctive.

Conclusion générale

Avec l'apparition de l'internet et son impact dans l'économie et la société qui ne cesse de s'accroitre, l'informatique réseau a vu de nombreuse innovation pour exploiter les ressources d'un réseau de cette ampleur.

Actuellement, les systèmes pair-à-pair sont pris au sérieux, ils sont capables de façonner radicalement certaines approches de l'informatique réseau, n'empêche que ces systèmes (pair-à-pair) affiche certains handicapes à cause de leurs diversifications et hétérogénéités car chacun d'eux à sa propre méthode de recherche par conséquent la communication entre eux est quasi impossible.

La solution était donc l'initiative de grandes entreprises ainsi que la communauté des chercheurs qui ont bâti une plateforme de programmation réseau, pour ambition de devenir un standard du pair-à-pair qui pourra ainsi supporter pratiquement tous les services.

Ce travail consiste à définir notre approche pour la conception d'un service pair-à-pair permettant le partage de fichiers, le téléchargement et le chat dans un réseau local, cette approche s'inspire à la fois des systèmes pair-à-pair et de la plateforme JXTA.

Au cours de la réalisation de l'application, on s'est confronté à de nombreux problèmes, l'un des obstacles, le manque de documentation, le site officiel n'est plus actif, les tutoriels manquent de détails (exemple la version des librairies utilisées) ou ne fonctionne pas. Autre problème, les librairies manquent de documentation (le rôle des méthodes était parfois superflu). Malgré ces contraintes nous avons parvenu à terminer notre projet et arriver à termes de nos objectifs préalablement fixés.

Perspectives

- Améliorer la vitesse du téléchargement
- Ajouter de nouvelles fonctionnalités, exemple appelle audio ou vidéo.
- Le téléchargement parallèle.
- Une version Android de l'application (étant donné qu'elle est développé en java).
- Sécuriser les échanges de données.

Références

[1] Kitambala OMARI, “La réalisation d’une application de contrôle total des processus d’un ordinateur distant,” 2010. [Online]. Available: http://www.memoireonline.com/05/12/5812/m_La-realisation-dune-application-de-contrle-total-des-processus-dun-ordinateur-distant0.html. [Accessed: 10-Apr-2017].

[2] T. Anthony SOUSA LOPES Eric, “L’échange de fichiers dans les réseaux Pair-à-Pair,” 2004.

[3] Chouchane Sahraoui Yacine Med Redha, “Conception et réalisation d’un système multi-agents pour les enchères en ligne,” *memoireonline.com*, 2009. [Online]. Available: http://www.memoireonline.com/04/12/5734/m_Conception-et-realisation-dun-systeme-multi-agents-pour-les-encheres-en-ligne6.html. [Accessed: 07-Apr-2017].

[4] B. Amine, “Transfert de données dans les architectures P2P,” 2016.

[5] B. Souheyla, “Etude et Administration des Systèmes de Supervision dans un Réseau Local.”

[6] A. J. Muhamadu, “Thème Partage de fichiers de pair à pair sur JXTA,” p. 72, 2016.

[7] R. Steinmetz and K. Wehrle, *Peer-to-peer systems and applications*. éd. Spinger.

[8] H. HAFI, “protocole pour la sécurité des réseaux sans fil peer to peer,” .

[9] G. Hameau and Y. Miclard, “Les protocoles Peer-to-Peer.” [Online]. Available: https://wapiti.telecom-lille.fr/commun/ens/peda/options/ST/RIO/pub/exposes/exposesrio2005ttnfa2006/miclard-hameau/#_Toc125902610. [Accessed: 12-Apr-2017].

[10] F. Kilanga Nyembo, “Réseaux pair-à-pair structurés en ProActive,” 2009.

[11] S. KTARI, “Interconnexion et routage dans les systemes pair a pair,” 2010.

[12] E. Riviere and G. Philippe, “Rechercher parmi ses pairs ou quand le hasard ne fait pas si bien les choses.”

[13] A. Benoit, “Notes de cours (ENS Lyon, M1) Chapitre 2 : Reseaux Pair à Pair,” 2006.

[14] K. Joseph, “mise au point d’une application de téléchargement peer to peer,” pp. 2014–2015, 2015.

[15] B. Amine, “Chap 3 : Réseaux structurés Introduction.”

[16] M. Jan, “JUXMEM : un service de partage transparent de données pour grilles de calcul fondé sur une approche pair-à-pair SOUTENUE,” 2011.

[17] B. LAHOUARIA, “Evaluation de performance des couches de communication de JXTA,” 2012.

[18] J. Gradecki and J. Mastering, *Building Java Peer-to-Peer Applications*. 2002.

[19] B. J. Wilson, *Jxta*. 2002.

[20] D. Brookshier, D. Govoni, N. Krishnan, and J. Soto, *Jxta: Java P2P Programming*. 2002.

Résumé

Dans ce mémoire nous nous concentrons sur la couche services de la plateforme JXTA, ceci dit nous avons introduit les notions de base du pair-à-pair, les concepts de la plateforme JXTA et ses protocoles, ce qui nous a permis de mettre en place nos services. L'objectif des services est d'abord d'offrir la possibilité de partager du contenu, un pair peut donc partager des données telle que des fichiers textes ou des images avec des pairs voisin (dans un réseau local). Le CMS conçoit des annonces (Advertisements) pour fournir ses informations et s'appuie sur le concept du Pipe JXTA pour le transfert de fichier. Ensuite y'a la messagerie instantanée qui permet aux pairs d'entretenir une discussion, il s'appuie entièrement que sur du JXTA.

Dans ce modeste travail, de par notre application, on a offert à un client lambda une sorte de solution alternative de ce qui peut exister dans le marché.

Mots-clés : P2P, JXTA, Service, Annonces, messagerie, partage, CMS.

Abstract

In our Master thesis, this lead us to introduce the basic notions of peer-to-peer, the concepts of the JXTA platform and its protocols, which enabled us to set up our services. The objective of services is first to offer the possibility of sharing content, so a peer can share data such as text files or images with neighboring peers (in a local network). The CMS conceive advertisements to provide its information and is based on the concept of the JXTA Pipe for file transfer. Then there is instant messaging that allows peers to entertain a discussion, it relies entirely on JXTA.

In this modest work, by our application, we offered to a lambda client customer a kind of alternative solution to what may exist in the market.

ملخص

في هذه المذكرة لقد ركزنا على طبقة الخدمات، وهذا يؤدي بنا إلى تقديم المفاهيم الأساسية للنظير إلى نظير والمفاهيم الأساسية ل(JXTA) وبروتوكولاتها، التي سمحت لنا لإقامة خدماتنا .الهدف الأول من الخدمات هو توفير القدرة على مشاركة المحتوى، وبالتالي النظير يمكنه تبادل البيانات مثل الملفات أو الصور مع أقران مجاورين له (في الشبكة المحلية)، يصمم (CMS) الإعلانات لتقديم معلوماته ويستنفد على مفهوم أنابيب (JXTA) لنقل الملفات، ثم هناك الرسائل الفورية التي تسمح لأقرانه على المناقشة، لأنه يعتمد كليا على (JXTA) .

في هذا العمل المتواضع، من خلال تطبيقنا، لقد عرضنا على العميل "لامدا" نوعا من الحل البديل لما يوجد في السوق .