REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'enseignement supérieur et de la recherche scientifique

Université des sciences et de la technologie Mohamed Boudiaf

Faculté des sciences

Département d'informatique

MEMOIRE

Présenté par

ARBAOUI Soraya

Pour obtenir

LE DIPLOME DE MAGISTER

Spécialité Informatique

Option : Systèmes Réseaux et Bases de Données

Intitulé

IMPLEMENTATION D'UN SERVICE D'INFORMATION PAIR A PAIR A TRAVERS LA PLATE-FORME JXTA « IISWJ »

Soutenu le 09/2011 dans la salle des soutenances

Devant les membres du jury :

NOM ET PRÉNOM	GRADE ET ÉTABLISSEMENT	EN QUALITÉ DE
A.Chouarfia	MCA, USTO	Président
R.Mekki	MCA, USTO	Rapporteur
K.Belkadi	MCA, USTO	Examinateur
S.A.Rahal	MCA, USTO	Examinateur

2010/2011

Dédicace

Je souhaite bien dédier ce modeste travail :

A ma nation musulmane,

A mes parents,

A mon époux Chikh Habib,

A mes enfants.

A mes frères et sœurs.

A toutes mes amies.

Remerciements

D’abord je tiens à exprimer mes plus vifs remerciements à mon encadreur, le docteur R.Mekki pour les nombreux conseils, orientations et encouragements qu’elle a su me prodiguer durant la réalisation de ce mémoire.
Je remercie les membres du jury d’avoir accepter de juger ce modeste travail.
Mes vifs remerciements s’adressent également au professeur A.Chouarfia pour l’intérêt qu’il a porté à ce mémoire ainsi qu’à ses différentes remarques et orientations.
Merci à tous ceux qui ont contribué à la réalisation de ce travail, de prés ou de loin.

Arbaoui Soraya

Sommaire

Introduction……1

Chapitre 1. Le pair-à-pair (peer to peer)

1.1 Introduction…… 3

1.2 Objectifs…… 3

1.3 Topologie du P2P…… 4

1.5 Caractéristiques des systèmes Peer to peer…… 12

1.5.1 Concepts de base…… 12

1.5.2 Fonctionnement …… 14

1.6 Conclusion…… 18

Chapitre 2. Les concepts de JXTA

2.1 Introduction……. 19

2.2 Le pair (peer)…… 19

2.3 Les Groupes de pairs (PeerGroups)…….. 20

2.4 Les modules dans JXTA…….. 23

2.5 Les messages…… 24

2.6 Les pipes…….. 24

2.7 La sécurité……. 29

2.8 Architecture de JXTA…….. 30

2.9 L'organisation du réseau…….. 32

2.9.1 SRDI (index distribute de partage de resources)……... 33

2.9.2 Firewalls et Nats…… 36

2.10 Conclusion…….. 37

Chapitre 3. Les protocoles et services JXTA

3.1 Introduction…… 38

3.1.1 Peer Discovery Protocol (PDP) …….. 40

3.1.2 Peer Information Protocol (PIP) …….. 40

3.1.3 Peer Resolver Protocol (PRP) …….. 41

3.1.4 Pipe Binding Protocol (PBP)…….. .. 42

3.1.5 Endpoint Routing Protocol (ERP)…… 43

3.1.6 Rendezvous Protocol (RVP)…… 44

3.2 Qu'est ce qu'un Module ?..... 45

3.3 Qu'est ce qu'un Service ?..... 45

3.4 Publication d'un Service JXTA…….. 46

3.5 LE service CMS…… 49

3.5.1 Caractéristiques du fichier partagé…… 49

3.5.2 Etapes de mise en place du service CMS…… 51

3.6 Conclusion…….. 51

Chapitre 4. La mise en place du service IISWJ

4.1 Introduction…… 52

4.2 Architecture de service…… 52

4.2.1 Ses caractéristiques…… 52

4.3 Configuration des pairs…… 53

4.3.1 Configuration avec la VirtualBox…….. 53

4.3.2 Configuration derrière Firewall/Nat…….. 53

4.3.3 Classe NetworkManager…….. 54

4.4 Assignation manuelle des identificateurs du service…….. 55

4.5 Création de classes…… 56

4.5.1 Classe Creating ModuleID()…….. 56

4.5.2 Classe RendezVous Feth, l'informateur…….. 57

4.5.3 Classe Edge Nasr, the Customer…….. 57

4.5.4 Classe Information-Service()…….... 58

4.5.5 Classe calculatrice()…….... 59

4.6 Exécution du service IISWJ…….... 62

4.7 Organisation du Service en vue d'un système de stockage distribué…… 69

4.8 Conclusion…….... 70

Conclusion générale ....

Table des figures.....

Bibliographie ...

Liste des figures

Chapitre 1. le pair à pair (P2P)

Figure 1.1- la topologie P2P …… 5

Figure 1.2- Architecture centralisée du peer to peer…… 8

Figure 1.3- Amélioration de l'architecture centralisée du peer to peer…… 9

Figure 1.4- De l'architecture centralisée du peer to peer vers l'architecture décentralisée…… 10

Figure 1.5- Le modèle hybride du Peer to Peer…… 11

Figure 1.6- Taxonomie des applications Peer to Peer……... 12

Chapitre 2. Les concepts de JXTA

Figure 2.1 Unicast et propagated pipes …….. 25

Figure 2.2 La connexion par les Pipes…… 26

Figure 2.3 Les différentes couches de l'architecture JXTA…… 32

Figure 2.4 propagation de requêtes via les peers Rendezvous…… 34

Figure 2.5 le fonctionnement de SRDI (index distribué de partage de ressources)… 36

Figure 2.6 L'acheminement d'un message à travers un pare-feu 37

Chapitre 3. Les protocoles et services JXTA

Figure 3.1- La pile de protocoles JXTA ……... 38

Figure 3.2- Définition d'un Service…….. 45

Figure 3.3- Etapes de publication d'un service JXTA…….. 47

Chapitre 4. L'architecture de JXTA

Figure 4.1- Le Service IISWJ…….. 52

Figure 4.2- Réseau physique utilisé pour notre Service……. 54

Figure 4.3- L'interface principale de IISWJ……. 62

Figure 4.4- La vérification de l'existence d'une ancienne configuration……. 63

Figure 4.5- Doit-on autoriser le multicast ?..... 63

Figure 4.6- Identification du RendezVous Feth par JXTA……... 63

Figure 4.7- Connexion au réseau JXTA et Chargement de IISWJ…….. 63

Figure 4.8- Exécuter le Peer Nasr et supprimer une configuration existante……... 64

Figure 4.9- Faire communiquer Feth-Nasr avec l'endpoint approprié…….. 64

Figure 4.10- Doit-on autoriser le multicast ?..... 64

Figure 4.11- Identification de l'edge Nasr par JXTA…… 64

Figure 4.12- Attendre la connexion de Nasr avec Feth..... 65

Figure 4.13- Localisation de l'implémentation du Service..... 65

Figure 4.14- Connexion de Nasr avec Feth…… 65

Figure 4.15- Service CMS (Partage de contenu)…… 66

Figure 4.16- Sélection de dossiers..... 66

Figure 4.17- IISWJ est prêt à renvoyer la calculatrice…….. 67

Figure 4.18- la calculatrice demandée par Nasr……. 67

Figure 4.19 - BiDipipe entre Feth et Nasr…….. 67

Figure 4.20- Arrêt du Service IISWJ……. 68

Figure 4.21- Arrêt du réseau JXTA…… 68

Introduction

Le peer-to-peer est dans l'esprit des développeurs récents car Internet est composé d'immenses ressources qui sont potentiellement utilisables mais souvent mal exploitées. Ces ressources sont de trois types: bande-passante, espace de stockage et capacité de traitement. C'est parce que le modèle client-serveur classique ne parvient pas à tirer le maximum de ces ressources que le modèle P2P est né.
Cependant beaucoup ont évité des solutions existantes puisqu'elles sont généralement à usage unique c'est-à-dire limitées au partage de fichiers ou liées à la messagerie instantanée.
Bien que le P2P peut libérer et distribuer un montant impressionant de puissance de calcul, de stockage et de bande passante, l'absence d'un cadre générique pour la construction de solutions interopérables de P2P a empêché la plupart des entreprises de tirer pleinement parti de cette technologie de potentiel.

Le projet JXTA est l'initiative lancée par Sun pour standardiser les applications P2P qui manquent de normes et de soutien industriels. Depuis Avril 2001, une première mouture de leur plate-forme de développement a été placée dans le domaine de l'OpenSource. Cette plate-forme a pour but de faciliter l'interopérabilité entre les différentes applications P2P , d'être indépendante du hardware et software utilisés et enfin de pouvoir être intégrée dans n'importe quel type d'équipement pouvant être relié à d'autres.
Par ailleurs, JXTA est un ensemble de protocoles avec des capacités révolutionnaires pour le développeur de l'application qui n'est plus confondu à la messagerie instantanée, au calcul sur grille ou même au partage de fichiers car JXTA est toute ces choses et plus encore Juxta est un ensemble de 6 protocoles d'échange de messages au format XML (ce format pouvant être considéré comme universel). Elle réalise également une abstraction des canaux de communication. Ainsi, elle ne tient compte que de la qualité de service offerte par le lien sans se préoccuper du protocole de transport réellement utilisé.

En effet l'architecture de Jxta repose sur trois couches : la couche application au sommet, la couche services et plus bas la couche noyau.
Dans la couche Service dont nous nous intéressons et qui n'est rien d'autre qu'une librairie qui étend les fonctionnalités offertes par Jxta Core et qui fournit des mécanismes pour rechercher, partager, indexer et mettre en cache du code et du contenu.
Il existe des services essentiels dans la plateforme jxta tels que :

- le protocole amélioré de découverte de ressources.
- indexation de ressources (hash table).
- objets distribués (jxta RMI).
- calcul distribué.

Parallèlement, il peut exister des services optionnels comme celui que nous avons mis en œuvre « IISWJ » .
Cependant le Service JXTA a recours à un pipe bidirectionnel qui est un canal pour communiquer avec les autres pairs dans un groupe de pairs définissant des objets de type net.jxta.endpoint.Message qui pourront être chargés sur ce canal.

Notre Service « IISWJ » utilise classiquement le tandem de classes : JxtaServerPipe/JxtaBidiPipe pour servir les requêtes vers celui-ci, une instance de type JxtaServerPipe est constamment à l'écoute de connexions et lorsqu'une connexion est acceptée, elle pourra fournir une instance de type JxtaBidiPipe grâce à laquelle il est alors possible de procurer un service à un pair du réseau JXTA. C'est au moment où le service détecte l'arrivée d'un message dans le pipe bidirectionnel que la méthode transmise au démarrage du service s'exécute.

Deux variables particulières sont liées dynamiquement au code, de sorte que celui-ci peut tout à fait les utiliser : ce sont les variables event et pipe ; la variable event qui est de type net.jxta.pipe.pipeMsgEvent permettra en particulier de récupérer le message jxta reçu dans le pipe et la variable pipe de type net.jxta.util.JxtaBidipipe pourra servir à envoyer des messages à l'autre bout du pipe.
En premier notre service permet le partage de contenu (Content Sharing and the Content Management Service (CMS)) permettant ainsi à un peer de partager des données comme des documents ou des fichiers graphiques.

Comme nous avons pu implanté une calculatrice qui peut être invoquée par un pair, d'autres méthodes peuvent être rajoutées aussi.
Actuellement, il existe d'autres travaux exploitant la notion de service jxta combinant l'agilité du langage Groovy qui permet d'écrire plus facilement des services P2P pour le framework jxta avec l'implémentation java (JXSE, version 2.5).

Dans ce mémoire, nous essayerons d'expliquer les différentes étapes qui nous ont permis de mettre en œuvre notre service peer to peer sous la plate-forme JXTA « IISWJ ».

Le mémoire est composé de quatre chapitres :

Chapitre 1: " Le pair-à-pair" est une introduction générale sur la technologie Peer to Peer, avec ses différentes architectures et applications.

Chapitre 2: " Les concepts de JXTA ce chapitre permet de comprendre les différentes ressources au sein de JXTA ainsi que son architecture.

Chapitre 3: " Les protocoles et Services JXTA" ce chapitre présente la notion détaillée des protocoles et Services dans JXTA.

Chapitre 4: " La mise en place du Service IISWJ" ce dernier chapitre présente les différentes étapes pour l'exécution du Service proposé.

Chapitre 1

Le Peer to Peer

- **Introduction**
- **Objectifs**
- **Topologie**
- **caractéristiques des systèmes peer to peer**
- **Les applications P2P**
- **La différence entre JXTA et les autres technologies P2P**
- **Le projet JXTA**
- **Conclusion**

Chapitre 1 Le Pair à Pair (peer to peer)

1.1. Introduction

L'architecture peer to peer (P2P), que l'on traduit généralement par réseau de "poste à poste", "pair à pair" ou encore "d'égal à égal" désigne une architecture de réseau où les postes connectés communiquent directement entre eux et partagent leurs ressources. Tous les postes ont un rôle équivalent, à la fois client et serveur par rapport à ces ressources (espace de stockage, puissance de calcul...), d'où leur appellation de "servent" (contraction de serveur et client). C'est la mise en commun de ces ressources qui fait la force et le succès de ces réseaux.

Exemples de ressources :

- Le contenu : les fichiers présents sur la machine.
- La bande passante : communication (messagerie/ téléphonie, streaming audio/vidéo).
- La puissance de calcul ou mémoire : calculs scientifiques.
- L'espace disque : sauvegarde croisée.

1.2. Objectifs du P2P

Différentes applications peuvent utiliser le P2P, les objectifs varient:

- Partage et réduction des coûts entre les différents peers.

- Fiabilité et passage à l'échelle, l'absence d'élément centralisé pour l'échange des données permet d'accroître la fiabilité en supprimant tout point central de panne et d'améliorer le passage à l'échelle en évitant les goulots d'étranglement.

- Agrégation des ressources et interopérabilité, en mettant en commun des ressources individuelles comme la puissance de calcul ou l'espace de stockage.

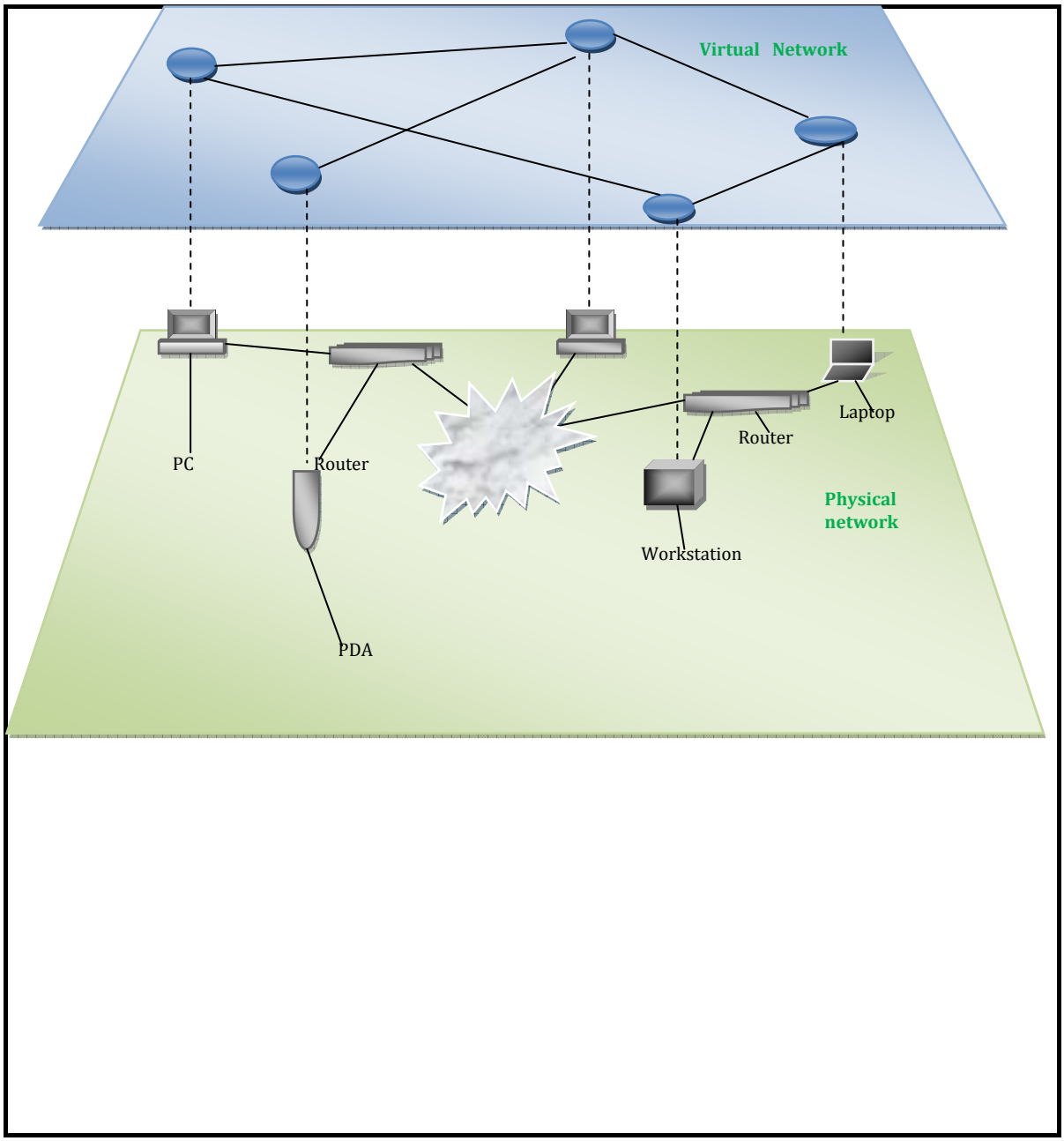
- Accroissement de l'autonomie en l'absence d'une autorité centrale, il est de la responsabilité de chacun de partager ou non des fichiers.

- Anonymat pouvant être assuré par certaines applications, en utilisant par exemple des algorithmes de routage qui rendent quasiment impossible le pistage d'une requête.

- Communication ad-hoc et collaborative.

1.3. La topologie du P2P

(Les réseaux peer to peer se greffent au réseau public qui est internet. Ils présentent donc une topologie virtuelle ou de l'anglicisme : « overlay network »).



La taxonomie ci-dessous permet de classer les systèmes peer to peer dans les différents systèmes informatiques existants.

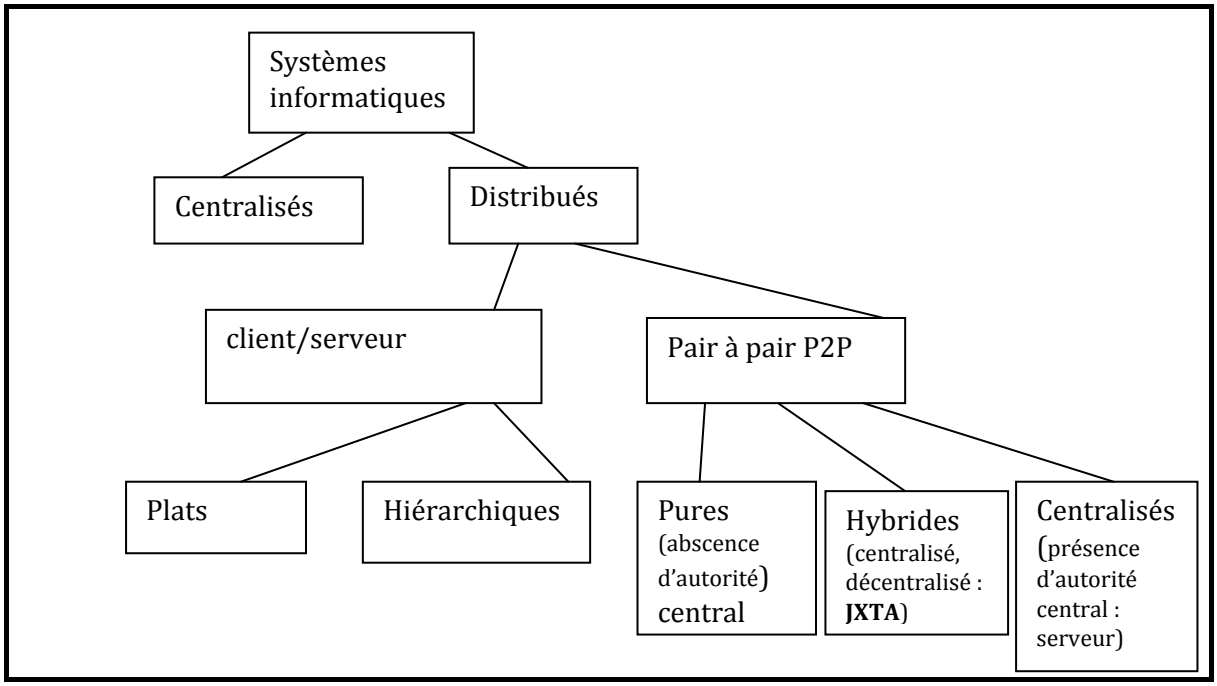


Figure 1.1 La topologie P2P.

1.4. Caractéristiques des systèmes P2P

- Un pair a une sensibilisation pour les autres pairs.
- Les pairs créent des réseaux virtuels.
- Chaque nœud peut être à la fois client et serveur.
- Un mode de communication standard (TCP, HTTP).
- A l'inverse d'un environnement Client-Serveur la performance d'une application P2P augmente lorsque plusieurs nœuds sont connectés : partage de ressources, disponibilité de l'information au cas où le serveur tombe en panne, réduction du trafic sur le serveur (réplication des données sur d'autres nœuds).

1.4.1 Concepts de base

1. Mode Point-to-Point

Permet de relier deux pairs. Ce n'est plus qu'ouvrir un port de communication entre les pairs. La communication Point-to-Point est utilisée pour l'envoi des messages adressés à un pair.

2. Mode diffusé (mode broadcast)

La diffusion de données en utilisant le P2P permettrait de toucher beaucoup plus de personnes avec des moyens bien inférieurs à ceux nécessaires aujourd'hui. Actuellement, si on veut emettre de la vidéo (par exemple), il faut non seulement une machine qui diffuse la vidéo

(logique), mais également une très bonne connexion : plus le nombre d'utilisateurs est important, plus la connexion est importante. En revanche, en utilisant le P2P, chaque personne devient à la fois réceptrice et emettrice. Ainsi, le diffuseur initial n'a plus que quelques clients connectés à lui, chacun de ces clients jouant également le rôle de diffuseurs pour d'autres clients.

3. Graphe de fonctions

Les graphes de fonctions décrivent les nœuds connectés, donc ils représentent les taches reliées à la topologie du réseau. En plus ils donnent les informations sur la fonctionnalité de chaque nœud du réseau, le type, la qualité et la capacité de la connexion entre les nœuds. Le graphe de fonction comprend le routage, le chargement, la recherche et d'autres fonctions.

4. Réduction de la bande passante

Le P2P permet de bénéficier d'une meilleure bande passante grâce à la réduction du trafic des requêtes.

On peut réduire la consommation de la bande passante de manière significative d'un serveur dans les applications traditionnelles Client-Serveur (les sites web, les fichiers de dépôt) à celui d'un seul pair. Un pair peut transmettre des fichiers à d'autres pairs qui vont également les transmettre à tous les pairs du réseau. Donc au lieu d'un serveur, tous les pairs partagent leurs bandes passantes pour distribuer les fichiers.

5. Barrières internet « Firewall, NAT, Serveurs proxy »

Un des problèmes majeurs de l'internet est que les barrières peuvent simplement bloquer des applications et la communication entre les pairs d'un réseau.

Un Firewall est la barrière la plus connue. Les Firewalls filtrent les ports de communication, limitent l'accès à certains ordinateurs et surveillent le trafic.

Les serveurs proxy et les périphériques NAT ne sont pas des barrières.

- **Serveur proxy :** Les réseaux locaux sont fréquemment reliés à Internet via des passerelles ou routeurs, ils utilisent le plus souvent le protocole TCP/IP. Le rôle initial du serveur proxy ou mandataire était de relayer les requêtes d'un réseau Local ou privé vers des réseaux distants. Aujourd'hui ce rôle de relayeur est devenu obsolète. Les serveurs proxy ou mandataire, sont dotés de nouvelles fonctions:
  - Fonction de cache,
  - Fonction de filtre,

- Fonction de sécurité (barrière entre internet et le réseau local ou privé),…

- **Le NAT (Network Adress Translation)** : permet de partager une connexion internet entre plusieurs ordinateurs sur un réseau local. En effet, les PC sur un réseau local ont des IP locales qui ne sont pas routables sur internet. De ce fait, le NAT est né afin de permettre à tous les PC d'un réseau de se connecter à internet.
  Le NAT contredit le principe fondamental d'IP d'une communication de bout en bout (les stations d'extrémité établissent et gèrent elles-mêmes leurs communications). Le NAT pose donc des problèmes dans l'établissement de communications utilisant certains protocoles de sécurité assurant une authentification et un cryptage, des applications peer-to-peer et autres tels que FTP.
  En matière de sécurité, ce n'est jamais qu'une option qui ne peut remplacer un filtrage IP pertinent.
  Par ailleurs, son utilisation répandue rend opaque l'étendue réelle de l'Internet. Rappelons que ce principe a été mis en place pour répondre de manière temporaire au manque d'adresses IP.

1.4.2 Fonctionnement du Pair-à-Pair

A. Architecture centralisée : le peer-to-peer assisté

Le réseau peer to peer le plus connu du grand public est sans doute Napster. Son originalité réside dans le fait qu'il utilise une architecture centralisée ce qui a contribué à son succès mais aussi à sa perte mais dans la réalité, ce type d'architecture exige un tel investissement en ressources que les services restent rarement de bonne qualité (lenteur, disponibilité...). Soit ils sont saturés, soit ils sont limités en termes d'utilisateurs simultanés autorisés. Concrètement, dans toute architecture centralisée, un dispositif exclusivement serveur se charge de mettre en relation directe tous les utilisateurs connectés. L'intérêt de cette technique réside dans l'indexation centralisée de tous les répertoires et intitulés de fichiers partagés par les abonnés sur le réseau. En général, la mise à jour de cette base s'effectue en temps réel, dès qu'un nouvel utilisateur se connecte ou quitte le service. Un client lançant une requête comme avec un moteur de recherche classique obtient une liste d'utilisateurs actuellement connectés au service et dont les fichiers partagés correspondent au fichier recherché. Dès lors, il suffit de cliquer sur l'un des intitulés du lien pour se connecter directement sur la machine correspondante et entamer le transfert. Dans ces conditions, à aucun moment les fichiers ne se retrouvent stockés sur le serveur central.

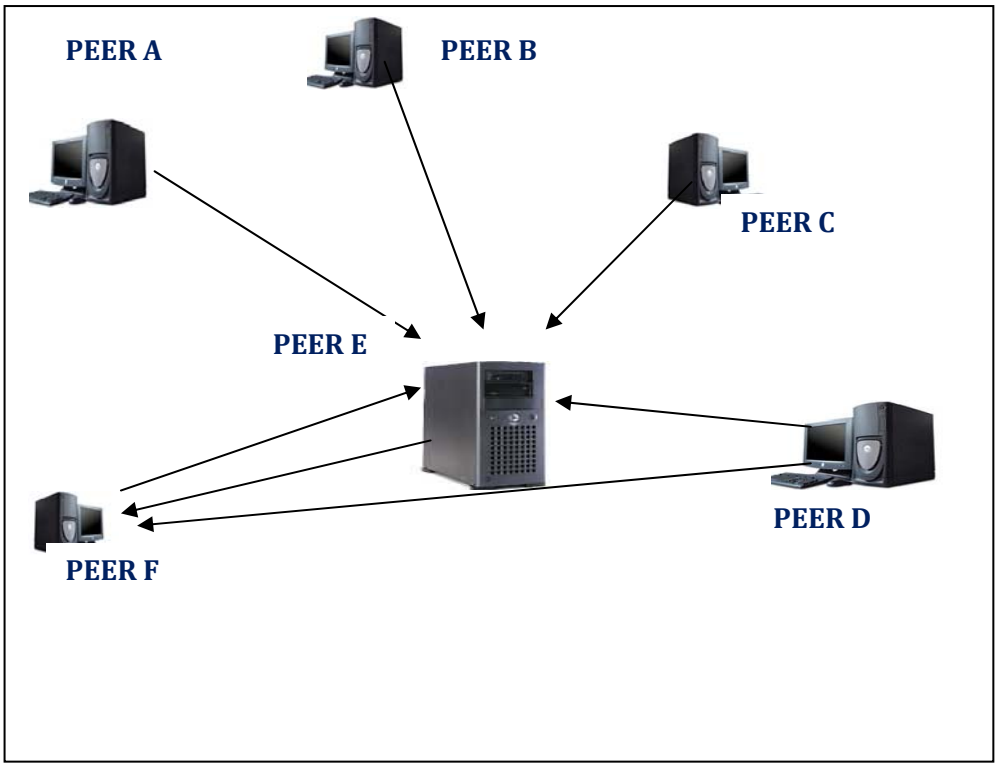


Figure 1.2 Architecture centralisée du peer to peer

• Un utilisateur recherche un fichier ressource en envoyant une requête au serveur central

• Le serveur central répond et transmet la liste des ordinateurs utilisateurs proposant le fichier demandé.

• L'utilisateur télécharge le fichier directement à partir d'un des ordinateurs renseignés par le serveur.

Au niveau de la sécurité, une architecture P2P centralisée s'avère particulièrement vulnérable. Elle ne propose qu'une seule porte d'entrée, son serveur centralisé, ce qui constitue le point vulnérable de tout le réseau. Il suffirait effectivement de bloquer ce serveur pour déconnecter tous les utilisateurs et stopper le fonctionnement de l'ensemble du réseau. Autre chose, le fait de passer à travers une architecture centralisée, où il faut s'enregistrer pour pouvoir y accéder, ne garantit bien évidemment aucun anonymat. Le service connaît l'adresse IP de votre machine et le type de fichiers que vous téléchargez. Il peut facilement élaborer des profils d'utilisateurs.

B. Une amélioration du Peer-to-Peer centralisé

Pour résoudre les problèmes de robustesse et améliorer la qualité de connexion avec le serveur, le serveur central de l'architecture centralisée est remplacée par un anneau de serveur. Ceci permet d'éviter la chute du réseau si une panne se produit sur un serveur, car il y a toujours un point de connexion valide aux serveurs. De plus l'utilisation de plusieurs serveurs

permet de mieux répartir les demandes de connexions et donc de limiter la chute de bande passante. Chaque serveur peut avoir accès aux informations des clients connectés sur les autres. L'accès aux données partagées est donc totalement transparent pour les utilisateurs.

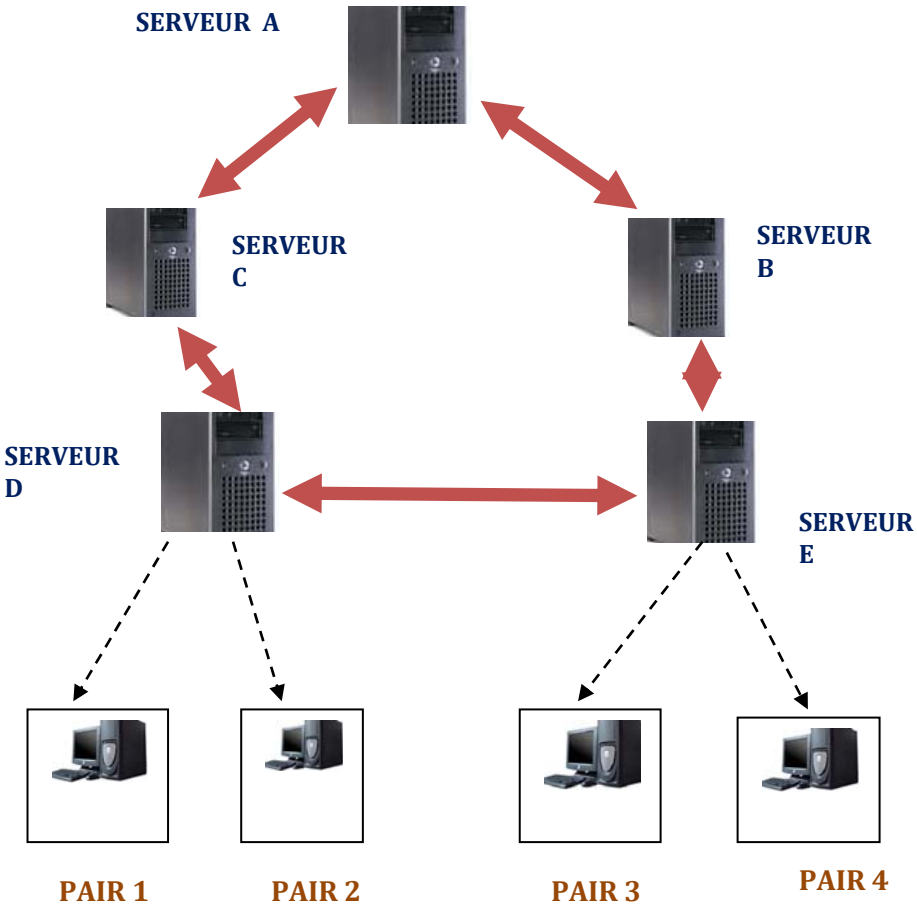


Figure 1.3 Amélioration de l'architecture centralisée du peer to peer

C. De l'architecture centralisée vers l'architecture décentralisée

Nous avons vu que l'architecture centralisée pose des problèmes de sécurité, robustesse et de limitation de la bande passante. Les problèmes sont directement issus de l'utilisation de serveurs dont le seul but est de posséder l'annuaire des clients. Si on désire supprimer les serveurs centraux il faut donc trouver le moyen de constituer un annuaire sur chaque client, puis de les faire communiquer. C'est sur ces mécanismes que sont basés les réseaux Peer to Peer décentralisés. Il n'y a donc plus de serveurs centraux, ce sont tous les éléments du réseau qui vont jouer ce rôle. Cette seconde génération de systèmes pair-à-pair est qualifiée de

réseaux purs. Un grand avantage de ce nouveau type de réseaux est en théorie le total anonymat qu'il procure en évitant de communiquer avec une machine centralisant les demandes et en évitant les problèmes de récupération des données utilisateurs.

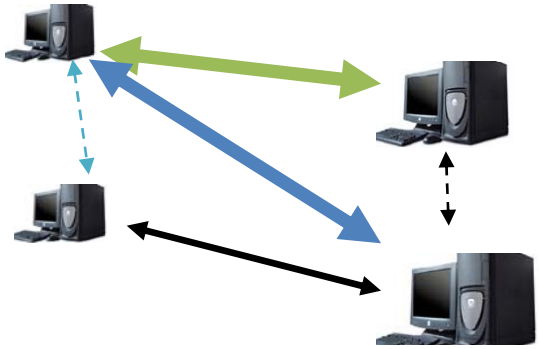


Figure 1.4 De l'architecture centralisée vers l'architecture décentralisée du P2P

D. Architecture décentralisée

Contrairement aux réseaux centralisés où il suffisait de se connecter au serveur pour avoir accès aux informations, pour avoir accès à une information en décentralisé aurait-il fallu:

- Apprendre la topologie du réseau sur lequel le client est connecté.
- Rechercher l'information sur tous les nœuds.
- Recevoir une réponse d'un nœud répondant aux critères.

La création de l'annuaire

- Le client se connecte sur le réseau ne connaissant pas la topologie du réseau.
- Pour connaître les autres membres du réseau, il va "broadcaster" une demande d'identification des nœuds du réseau.
- Les nœuds recevant cette demande vont à leur tour la répercuter sur tous les nœuds voisins et ainsi de suite.
- Lorsque que la trame est reçue et identifiée par un autre client, le nœud renvoi une trame d'identification au premier client demandeur.
- Ainsi le client initial va pouvoir identifier tous les nœuds du réseau et se créer un annuaire.

Les limites : Le principal inconvénient de cette méthode sont les séries de Broadcast qui sont diffusées encombrant le réseau qui a pour conséquence de ralenti les échanges de données entre les machines.

E. Un modèle hybride : les réseaux Super Nœuds

Le modèle super nœud a pour but d'utiliser les avantages des 2 types de réseaux (centralisé et décentralisé). En effet sa structure permet de diminuer le nombre de connexions sur chaque Serveur, et ainsi d'éviter les problèmes de bandes passantes.

D'autre part le réseau de serveurs utilise un mécanisme issu des réseaux décentralisés pour tenir à jour un annuaire client et un index des fichiers à partir des informations provenant des autres serveurs. Un serveur peut donc proposer à n'importe quel client toutes les informations contenues sur le réseau. Le réseau n'est plus pollué par les trames de broadcast. Mais la contrepartie est que l'anonymat n'est plus assuré.

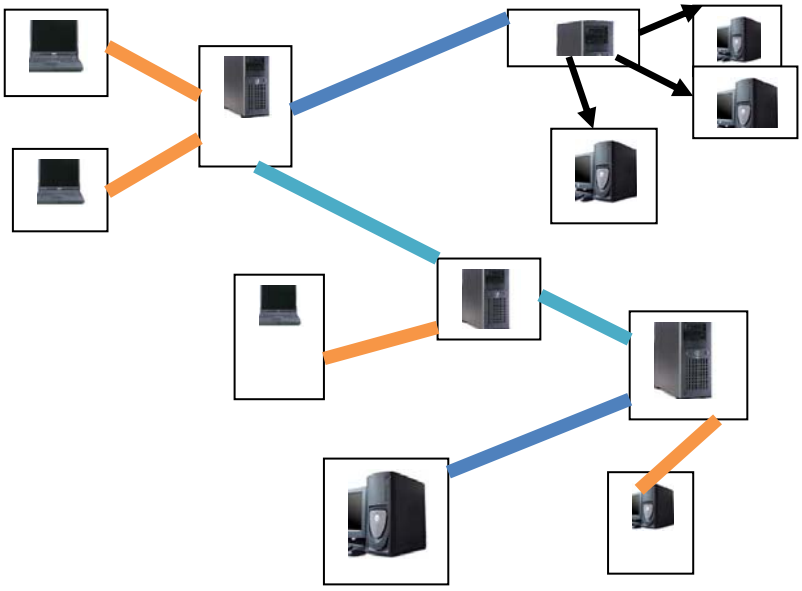


Figure 1.5 Le modèle hybride du P2P

Nous remarquons d'après les trois types de générations précédentes (centralisée, décentralisée et hybrides) que l'utilisation d'un système pair-à-pair nécessite pour chaque nœud l'emploi d'un logiciel particulier. Le modèle pair-à-pair va bien plus loin que les applications de partage de fichiers. Il permet en effet de décentraliser des services et de mettre à disposition des ressources dans un réseau. Tout nœud d'un réseau pair-à-pair peut alors proposer des objets et en obtenir sur le réseau. Les systèmes pair-à-pair permettent donc de faciliter le partage d'informations. Ils rendent aussi la censure ou les attaques légales ou pirates plus difficiles. Ces atouts font des systèmes pair-à-pair des outils de choix pour décentraliser des services qui doivent assurer une haute disponibilité tout en permettant de faibles coûts d'entretien. Toutefois, ces systèmes sont plus complexes à concevoir que les systèmes client-serveur.

1.5. Les applications P2P

1.5.1 Introduction

Les applications P2P disponibles sur Internet, offrent une grande base d'utilisateurs, des milliers de fichiers téléchargeables avec une utilisation efficace des ressources du réseau. Ces dernières années, les applications P2P sont devenues une base pour développer de nouvelles idées commerciales qui permettent aux utilisateurs de se rencontrer sur Internet et de partager des intérêts communs.

La figure suivante montre plusieurs types d'application P2P

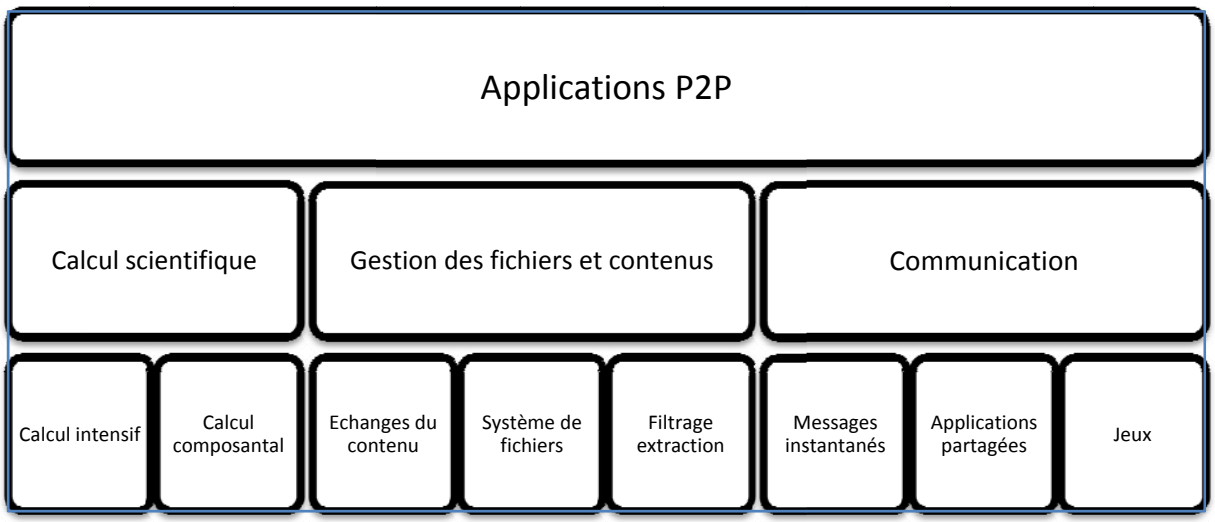


Figure 1.6- Taxonomie des applications P2P.

Il existe différentes types d'applications utilisant le concept du P2P :

A. Communication:

(P2P Chat) largement répandue grâce à la caractéristique qu'ont les pairs d'interagir directement entre eux. Les implémentations varient beaucoup.

Exemples de systèmes pair à pair dans la messagerie instantanée:

- AOL Instant Messenger,
- Yahoo! Messenger,
- MSN Messenger,
- ICQ,
- Skype, etc.

B. Applications partagées:

Ce type d'applications permet à des usagers de collaborer en temps réel sans utiliser de serveur central. Ces applications permettent de travailler de manière commune sur un projet distribué en permettant le partage des idées, ressources, documents et l'emploi du temps via une communication interactive.

Exemples :

- Groove,
- Magi,
- PowerPoint distribué,
- NextPage,
- Knari.

C. Les jeux en réseau.

C'est un domaine vaste pour le P2P, dont l'architecture est exempte de toute autorité centrale, ceux-ci font également partie des applications de collaboration P2P.

Exemple: Doom

Le projet JXTA a été utilisé dans plusieurs applications de jeux (le jeu d'échecs, tic-tac-toe) ; le joueur peut créer un nouveau groupe JXTA, inviter des joueurs du monde entier comme il peut limiter l'accès à son groupe de jeu.

D. Gestion des fichiers et contenus

Généralement utilisée pour échanger les fichiers multimédia (musique, vidéo, etc.).

Exemples :

- Napster,

- Gnutella,
- Emule,
- Kazaa,
- BtTorent, etc.

E. Les bases de données distribuées

L'idée est de sauvegarder des fichiers et des informations de manière distribuée sur le réseau (sans se limiter aux disques durs locaux).

Exemples : Mariposa, Chord, etc.

F. Le calcul scientifique: grilles de calcul

Le P2P permet de mettre en commun de nombreux ordinateurs, disposant chacun de ressources limitées. Mais en « additionnant » ces ressources, on obtient des performances théoriques considérables. Par exemple le calcul matriciel dont le but est de diviser ce gros calcul en petits calculs que l'on pourra répartir ente les pairs.

D'autres exemples:

- Chord
- Cx : Transformer votre PC en ressources globales de calcul
- Farsite : Stockage de données dans un environnement non sécurisé
- Globe : Gestion d'objets distribués
- OceanStore : Stockage massif de données
- Pastry : Une sous-couche pour les applications P2P
- XtremWeb : Calcul scientifique à travers le Web

1.5.2 La description de quelques réseaux P2P actuels

A. e-Donkey et e-Mule

Le réseau e-Donkey est né en septembre 2000 mais c'est au cours de l'année 2001 qu'il a connu son véritable essor. Par rapport à ses prédécesseurs, e-Donkey fonctionne dans un mode décentralisé avec une multitude de serveurs : l'architecture est proche du super noeud. e-Donkey permet le transfert de tous types de fichiers et ce, quelque soit la plateforme puisqu'il dispose de client Windows et Unix. Chaque utilisateur peut ouvrir son propre serveur et tous les serveurs peuvent être reliés entre eux. Lorsqu'un client se connecte sur un serveur il lui fournit la liste des fichiers partagés. Une fois connecté, il peut

rechercher un fichier. Soit il le recherche sur le serveur sur lequel il est connecté, soit sur tous les serveurs connus par l'intermédiaire des autres peers (fonctionnalité réservée à e-Mule).

B. FastTrack: Kazaa, Grokster, Morpheus, Imesh

Le réseau FastTrack, dont le client le plus connu du grand public est Kazaa peut être comparé à e-Donkey sur plusieurs points. Tout comme lui, il permet le partage de fichiers de tous types et emploie le protocole MFTP. Il connaît un énorme succès dans le monde, sauf en France. Plusieurs raisons à cela :

• la présence d'un spyware dans la version grand public du client Kazaa,

• le peu de fichiers disponibles en version française (la tendance s'inverse peu à peu).

Sa force réside dans la facilité d'utilisation puisque son interface est très intuitive et comporte un lecteur multimédia et une bibliothèque pour organiser ses fichiers.

C. Gnutella

Gnutella se distingue en proposant un réseau entièrement libre. Pour écrire un client pour le

réseau Fasttrack, il faut avoir une licence. Gnutella (comme son nom l'indique) et sous licence Gnu et donc libre de droit. Il s'agit d'un peer-to-peer décentralisé qui a évolué au fil du temps. Son architecture était comparable à e-Donkey avec des fonctionnalités en moins (pas de recherches sur les serveurs voisins etc.). Suite à l'explosion du nombre de clients (de 75.000 à 300.000), le réseau a évolué vers le pur peer-to-peer. Pour se connecter l'utilisateur fait une sorte de "broadcast" vers l'Internet afin de trouver les utilisateurs les plus proches.

D.Groove

Groove (Groove Network) est un outil de partage destiné principalement aux entreprises. Il offre une sécurisation des échanges et une certaine pérennité dans son développement. Il s'agit d'un peer-to-peer centralisé pour l'authentification des acteurs du réseau. Orienté échange professionnel, il propose le partage de fichiers, des plateformes d'échanges, des outils de communication et un fonctionnement avec Microsoft Office.

J. ICQ

ICQ, le célèbre outil de communication utilise un réseau peer-to-peer pour fonctionner. Son architecture est centralisée mais la communication entre clients se fait indépendamment du serveur central. C'est grâce à cette architecture qu'ICQ peut aujourd'hui supporter un nombre si important de personnes sur son réseau.

1.6. La différence entre JXTA et les autres technologies Peer to Peer

Comme JXTA, JINI est une technologie développée par Sun, elle permet de fournir des services et d'interconnecter n'importe quel composant (PC, laptop, PDA, etc.). Bien qu'elles ont des buts similaires, JXTA supporte plusieurs langage (C, JAVA) tandis que JINI ne repose que sur la plateforme JAVA. Une autre différence, JINI est principalement conçu pour les réseaux locaux alors que JXTA fonctionne dans les réseaux locaux et dans l'internet. Aussi, JINI utilise un serveur centralisé pour localiser les services dans le réseau et il repose sur la méthode Remote Method Invocation (RMI) et la sérialisation des objets pour la communication entre les objets distants. JXTA repose sur le langage XML plutôt que sur la sérialisation des objets, pour l'échange des donnés structurés et la découverte des services entre les peers dans un réseau Peer to Peer Napster est une architecture Peer to Peer hybride, conçue pour la recherche et le téléchargement des fichiers MP3, donc c'est une application de collaboration plutôt qu'une plateforme de création des applications Peer to Peer, elle est constituée d'un serveur centralisé pour améliorer les fonctions de recherche et qui peut être modélisée comme un peer Rendez-vous et de multiple peers simples. Tout en utilisant le protocole de transport TCP.

Le .Net est basé sur l'architecture Client/Serveur pour fournir des services disponibles via l'internet tandis que la création des applications Peer to Peer exige un travail lourd aux développeurs, car elle exige une spécification complète de toutes les interactions de noyau Peer to peer (exemple: Peer Discovery), donc l'utilisation .Net pour les applications Peer to Peer est coûteuse à cause de la reconstruction des mécanismes de bases qui sont déjà définis par les protocoles de JXTA.

Une différence majeure entre JXTA et les autres méthodes tels que RMI, COBRA, etc. est que les données ne soient plus bloqués par les firewalls et NAT.

1.7. Le projet JXTA

JXTA a été développée pour une plateforme Peer to Peer. Le nom JXTA, se prononce "juxta" est dérivé du mot juxtapose ayant le sens de placer deux entités l'une à coté de l'autre ou à proximité.

- L'équipe de développement chez SUN sous la direction de Bill Joy et Mike Clary ayant reconnu que les solutions Peer to Peer vont toujours exister à coté des solutions Client/Serveur, plutôt que de les remplacer complètement ils ont conçu une solution qui standardise toutes les applications Peer to Peer.
- JXTA est donc une plateforme réseau Peer to Peer développée par Sun qui permet la communication avec d'autres plateformes JXTA dans le même réseau.
- Le noyau de JXTA est tout simplement formé d'un ensemble de protocoles, qui le rend puissant.
- Dans JXTA un peer peut être un relai, un Rendez-vous fournissant un ensemble de services pour le réseau. En plus, les peers JXTA utilisent plusieurs protocoles tels que HTTP, TCP, IP,… alors contrairement au Naphte, JXTA fournit une solution complète d'acheminement des messages à travers plusieurs pare-feu. Les services Web inclut des technologies telles que le XML, SAVON, et UDDI. Ces services .NET et JXTA ont des objectifs différents.

- Objectifs de JXTA

Les objectifs de la plateforme JXTA sont :

- L'interopérabilité entre les applications Peer to Peer.
- L'interopérabilité entre les terminaux de réseaux Peer to Peer.
- L'indépendance vis-à-vis du matérielles et systèmes d'exploitation.
- supporte plusieurs langages (JAVA, C++, C, etc.)
- ubiquité c'est à dire fournir des services et une infrastructure pour les applications Peer to Peer qui fonctionne sur n'importe quel composant (PC, PDA, etc.).

1.8. Conclusion

La technologie pair- à- pair offre des systèmes pair-à-pair qui ont été les derniers invités à arriver sur Internet, ils représentent un nouveau moyen d'offrir des services au bout de l'internet pour et par les usagers.

Les systèmes pair-à-pair ne sont rien d'autres que des applications distribuées et massivement déployées, supportées par les ressources fournies par des usagers interconnectés par Internet, ces systèmes sont donc composés par un grand nombre d'hôtes, leurs propriétés ont permis le passage à l'échelle.

Cependant, la plupart des applications P2P actuellement existantes ne peuvent s'exécuter que sur un seule type d'architecture et ne permettent pas de partager directement leurs ressources avec les autres. C'est dans l'optique de cette problématique que SUN Microsystem a créé le projet JXTA que nous détaillerons dans le chapitre qui suit.

Chapitre 2

Les concepts de JXTA

- **Introduction**
- **Le pair (Peer)**
- **Le groupe de pairs (PeerGroup)**
- **Les modules**
- **Les messages**
- **Les pipes**
- **La sécurité**
- **Architecture de JXTA**
- **L'organisation du réseau**
- **Conclusion**

Chapitre 2. Les concepts de JXTA

2.1. Introduction

Le but du projet JXTA est de créer une plateforme P2P qui permet de construire simplement, facilement et efficacement un ensemble de services d'applications distribuées pour tout dispositif qui pourrait être un pair JXTA.

Comme toute architecture réseau, l'architecture Peer to Peer exige un lexique qu'il faut comprendre. Pour cela on va définir les concepts de base critiques afin de comprendre la plateforme JXTA.

2.2. Le Peer (Pair)

Un pair est une entité réseau, qui implémente un ou plusieurs protocoles JXTA. Les pairs peuvent résider sur les capteurs, les téléphones portables, les PDAs, les PC, les serveurs, les supercalculateurs…Chaque pair a un identifiant unique le Peer ID et fonctionne d'une manière indépendante et asynchrone des autres pairs.
Les pairs publient une ou plusieurs adresses réseau utilisées au niveau des protocoles JXTA. Chaque adresse est annoncée par un « Peer endpoint », qui identifie l'adresse réseau. Les peer endpoint sont utilisés pour établir une connexion directe « point à point » entre deux pairs.
Un pair JXTA nécessite d'effectuer une série d'opérations pour joindre et participer à un réseau JXTA.

- Bootstrap dans la plateforme JXTA : charger les classes de la plateforme JXTA, rejoindre le groupe de pairs par défaut, ouvrir le listener de sockets, etc…
- Joindre un groupe de pairs.
- Publier ses annonces.
- ouvrir un pipe d'entrée (input pipe).
- Renseigner les autres pairs.
- Obtenir des annonces de pipes.
- Ouvrir un pipe de sortie (output pipe).

Les pairs JXTA sont divisés en trois grandes catégories :

- **Minimal-Edge-Peer** : les pairs qui implémentent les fonctions de base JXTA.
  Exemples typique : les capteurs, les machines domotiques.

- **Full-Edge-Peer** : les pairs qui implémentent toutes les fonctions de base et les services standards JXTA.
  Exemples typique : les téléphones portables, les PCs, les serveurs, etc.

- **Super-Peer** : ce sont des pairs qui implémentent et fournissent des ressources pour supporter le déploiement et l'exploitation d'un réseau JXTA, un Super-Peer peut implémenter l'une ou l'ensemble des fonctions qui suivent:

  - **Relay** : utilisé pour sauvegarder et transférer les messages entre les pairs qui ne sont pas connectés directement à cause des firewalls ou NAT. Donc uniquement les pairs qui ne sont pas capables de recevoir des connexions des autres pairs exigent un Relay.

  - **Rendezvous** : maintient les indexes des annonces globales et aide les edges et proxies peers pour la recherche des annonces et gère la diffusion des messages.

  - **Proxy** : utilisé par les Minimal-Edge-Peers pour accéder à toutes les fonctionnalités du réseau JXTA. Le peer proxy traduit et résume les requêtes, répond aux questions et fournit un support fonctionnel pour les Minimal-Edge-Peers.

2.3. Les Peer Groups

Un peer group est un ensemble de peers qui ont des services et des intérêts communs, les pairs s'organisent en des peer groups, chaque groupe est identifié par un seul identifiant « Peer group ID ». Un peer appartient à un ou plusieurs groupes, par défaut, le premier groupe instancié est le Network Peer Group. Tous les pairs appartiennent à ce groupe comme ils peuvent joindre un autre Peer group à n'importe quel moment.
Les protocoles JXTA décrivent comment les peers s'annoncent, se reconnaissent, se joignent, et surveillent les peer groups, mais ils ne dictent pas quand et pourquoi les peer groups sont crées.

Le but de la création d'un groupe peut être :

- **création d'un environnement sur (Secure environement) :** les groupes créent un domaine local de contrôle dont lequel une politique spécifique de sécurité est exécutée.

La politique de sécurité peut être une simple politique (username/password) ou sophistiqué (clé publique de cryptographie).

Les frontières d'un peer group permettent aux peers membres d'accéder et de publier des contenus protégés. Les peer groups forment des régions logiques dont les frontières limitent l'accès aux ressources d'un peer group.

- **création d'un environnement délimité (Scooping environement) :** les groups permettent la création d'un domaine local de spécialisation. Par exemple, les peers peuvent se regrouper pour mettre en place un réseau de partage de document ou d'un réseau de partage de CPU. Les peer groups servent à subdiviser le réseau en des régions qui fournissent un mécanisme de délimitation. Les frontières d'un peer group définissent la zone de recherche quand on recherche un contenu d'un groupe.

- **création d'un environnement de contrôle (Monitoring environement) :** les peer groups permettent à des peers de contrôler un ensemble de peers pour tout usage spécial (cardiaque, trafic, introspection, la réeddition de comptes).

Les groupes peuvent également constituer une hiérarchie parent-enfant, dans laquelle chaque groupe a un seul parent. Les demandes de recherche sont propagées à l'intérieur du groupe. L'annonce du groupe est publiée dans le groupe père en plus dans le groupe lui-même.

✓ L'EndPoint

C'est la méthode de base d'adressage utilisée par les applications JXTA pour communiquer les uns avec les autres. Un endpoint est une destination logique qui met en œuvre un protocole spécifique de communication. Un peer peut avoir de multiples endpoints donc il peut communiquer via différents protocoles.

✓ Network Services

Les Peers coopèrent et communiquent afin de publier, découvrir, et d'invoquer des services réseau. Les peers peuvent publier plusieurs services qui à leur tour sont découverts via le Peer Discovery Protocol.

Les protocoles JXTA reconnaissent deux niveaux de services réseau:

- **Peer Services :** un peer service n'est accessible que sur le pair qui l'a publié. Si ce pair devait échouer, le service échoue également. Plusieurs instances du service peuvent être utilisées sur différents pairs mais chaque instance publie sa propre annonce.

- **Peer Group Services :** un peer group service se compose d'une collection d'instances du service qui peuvent potentiellement coopérer entre eux en cours d'exécution sur plusieurs membres de l'hébergement de peer group, si l'un des pairs échoue, le peer group service collectif n'est pas affecté tant que le service est toujours disponible à partir d'au moins un membre peer. Les peer group services sont publiés dans le cadre du peer group advertisement.

Les services peuvent être soit préinstallés dans un pair ou chargés à partir du réseau. Afin de réellement faire tourner un service, un pair doit trouver une mise en œuvre appropriée pour son environnement d'exécution. Le processus de recherche, de téléchargement et d'installation d'un service du réseau est analogue à effectuer une recherche sur Internet pour une page Web, la récupération de la page, puis d'un plug-in qui est requis par la page.

✓ Peer Group Services

Un peer group fournit un ensemble de services connus collectivement comme le peer group services. JXTA définit un ensemble de services de peer group services. Ce peer group services constitue le service de signature du peer group, tout peer qui rejoint le groupe doit mettre en œuvre ces services. Des services supplémentaires peuvent être développés pour fournir des services spécifiques. Pour que deux pairs puissent interagir via un service, ils doivent tous les deux être membres du même peer group. Les peer group services, qui doivent être implémentées par tous les pairs afin de participer au réseau JXTA sont:

- **Endpoint Service :** l' Endpoint service est utilisé pour envoyer et recevoir des messages entre les pairs. Le Endpoint Service met en œuvre le Endpoint Routing Protocol avec de simples implémentations qui ne fournissent qu'une petite partie de l'ensemble du protocole.

- **Resolver Service :** Le resolver service est utilisé pour envoyer des requêtes de demande générique à d'autres pairs. Les pairs peuvent définir et échanger des requêtes pour trouver toutes les informations qui peuvent être nécessaires (par exemple, recherche des annonces, de déterminer le statut d'un service ou la disponibilité d'un pipe endpoint).

  En plus des peer group services du noyau (nécessaires pour chaque peer), plusieurs autres services sont définis pour chaque peer group. Les services standards mis en œuvre par chaque peer group spécifient les services dont ils ont besoin ». Un groupe de pairs ne mentionne que les services qu'il juge utile. Les standards peer group services qui se trouvent dans la plupart des peer groups sont les suivants:

- **Discovery Service :** est utilisé par les peer membres pour trouver les peer group resources, tels que les peers, des peer groups, les pipes et les services.

- **Membership Service :** est utilisé par les peer membres pour établir solidement les identités et la confiance au sein d'un peer group. Identités permettent aux applications et services de déterminer qui a fait la demande d'une opération et de décider si cette opération doit être autorisée. Les applications peuvent faire leur propre contrôle d'accès ou d'utiliser le Service d'accès (Access Service).

- **Access Service :** est utilisé pour valider les requêtes faites par l'un des peers à un autre. Le peer qui a reçu la requête identifie ce peer, et des informations sur la requête seront faites afin de déterminer si l'accès est autorisé.

- **Pipe Service :** est utilisé pour créer et gérer des pipes entre les membres du peer group.

- **Monitoring Service :** est utilisé pour permettre à un peer de surveiller d'autres membres du même peer group.

2.4. Les modules

Les modules sont des définitions de services et d'applications disponibles sur les peers ou dans un peer group. Les modules sont utilisés pour définir le code à exécuter. Pour s'assurer que les différents peers écrits avec des différents languages, versions, systèmes d'exploitation, et modules sont définis avec trois annonces XML:

- **Module Class** : Définit la version spécifique d'un module.

- **Module Specification :** Définit le module avec une plate-forme de définition qui inclut le comportement.

- **Module Implementation** : instance spécifique d'un module sur une plate-forme. Chaque annonce est utilisée pour réduire l'implémentation finale utilisée par un peer. Toutefois, seuls les identifiants respectifs peuvent être disponibles et il n'existe pas de véritable publicité disponible sur le réseau global. La raison est que certains modules ne sont pas publiques ou qu'ils n'ont pas des versions et des implémentations différentes. En revanche, les annonces sont à l'intérieur d'un peer et uniquement référencées par leur identifiants.

2.5. Les messages

Les messages JXTA sont utilisés pour assurer la communication entre les services et les applications JXTA. Ces messages sont l'unité de base d'échange de données entre les peers.

Chaque protocole JXTA est défini comme un ensemble de messages qui sont échangés par les pairs participants. Les messages sont envoyés entre les peers à l'aide de l'Endpoint Service, Pipe Service ainsi que JxtaSocket et d'autres approches. La plupart des applications n'ont pas besoin d'utiliser les pipes unidirectionnelles ou le JXTA Endpoint Service directement. Au lieu de cela, les applications et les services utilisent les Sockets JXTA et JxtaBiDiPipe (canaux de communications) pour envoyer et recevoir des messages.

La messagerie en JXTA est faite de deux manières différentes, le premier est la façon standard que l'on pourrait s'attendre de XML. Les messages sont des paquets qui contiennent une charge utile de données au format XML pour suivre les normes.

Le deuxième type de message est un message binaire très économique. Malgré le désir de l'utilisation de XML dans tous les messages JXTA, la réalité est qu'il ya beaucoup de messages envoyés et reçus. L'essentiel des messages XML est l'envoi de gros volumes de données qui semble inefficace. Aussi, parce que les messages sont envoyés d'une application à une autre, il est simple de normaliser le contenu du message. Les restes des protocoles sont encore en XML.

L'utilisation de messages binaires dans un protocole XML peut sembler contraire. par contre son avantage réside dans la compacité des données. Premièrement, les données peuvent être compressées en utilisant des techniques standards.

La compression de données telles que le texte peut créer une énorme économie de temps et de transmission. Une autre raison pour laquelle les données sont en binaire est que beaucoup de messages sont déjà en binaire. Lors du transfert de messages via XML, les données devraient être converties en XML.

Une autre raison réside dans le cryptage des données qui doivent être converties en binaire, le déplacement à partir du binaire au lieu du XML donne un bon sens.

2.6. Les Pipes

Un pipe est une connexion virtuelle entre les peers. Normalement, nous pensons que les communications peer to peer se font selon une connexion directe, mais ce n'est pas toujours possible. Le problème est que de nombreux peers ne peuvent pas se connecter directement à cause des pare-feux ou d'autres obstacles. Les Pipes sont conçus comme une couche sur de multiples protocoles de communication et pour soutenir les communications relayées via les peers passerelles.

Le Pipe est une fonction importante de JXTA. Il crée un paradigme très utile qui permet à des peers de communiquer en réseau, malgré les pare-feux ou d'autres obstacles.

Les pipes offrent deux modes de communications:

- ✓ **Point-to-point- pipes (Unicast pipes):** pour connecter deux peers.
  - ✓ **Propagate pipes (wire pipes) :** pour connecter un peer à de multiples destinations. Propagate pipes peuvent aussi avoir des peers multiples impliqués dans la connexion, y compris ceux qui sont des endpoints dans la communication.

Il existe une version sécurisée qui fournit un canal de communication fiable et unicast, Secure Unicat Pipes.

Unidirectional pipes sont une abstraction de communication JXTA de très bas niveau. Il est recommandé que la plupart des développeurs utilisent la communication de haut niveau fournie par le JxtaSocket et JxtaBiDipipe services (se place au dessus des sockets) décrits dans la section suivante.

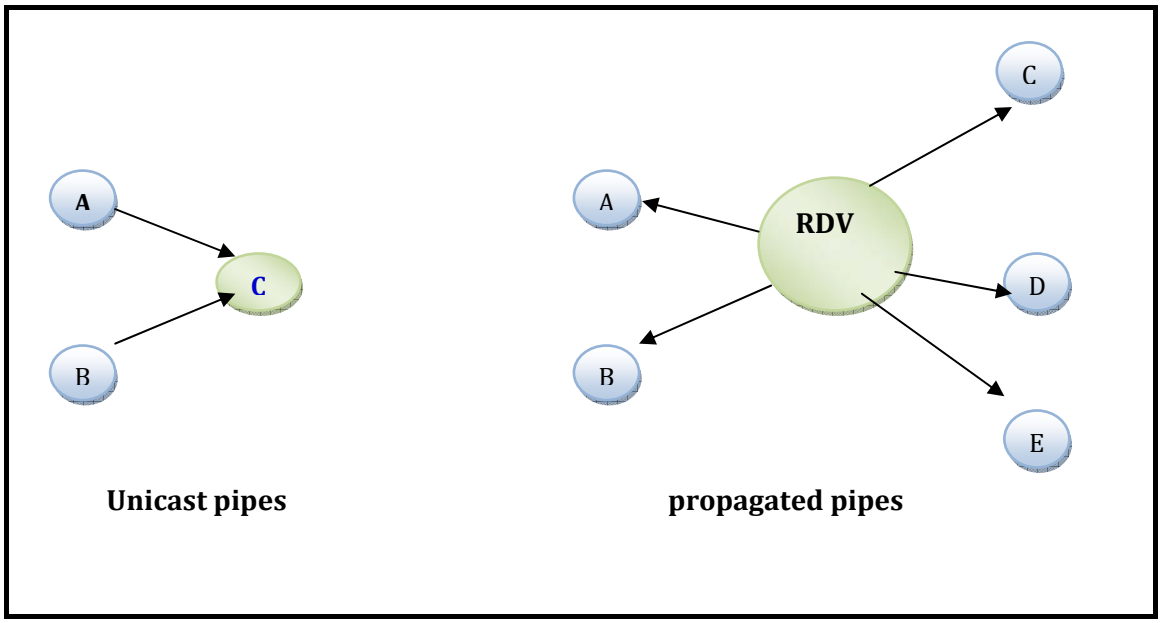


Figure 2.1- Unicast et Propagated pipes.

- JxtaSocket et JxtaBiDipipe

Les pipes JXTA fournissent des canaux de communication unidirectionnels non fiables. Afin de rendre les pipes plus utiles pour les services et les applications, il est nécessaire de mettre en œuvre des canaux de communication fiables et bidirectionnels aux dessus des primitives pipe.

JXSE fournit des fonctionnalités pour atteindre le niveau de qualité de service requis par la plupart des applications:

- Fiabilité
- Séquencement des messages
- La livraison
- Affichage des messages et des interfaces des flux
- Sécurité.

o JxtaSocket et JxtaServerSocket

- Sous-classe java.net.Socket et java.net.ServerSocket respectives ,
- Bâties sur les pipes, endpoint des messages et la bibliothèque de fiabilité,
- Fournissent des canaux de communications bidirectionnels, fiables et sûres
- Exposent une interface de flux,
- Fournissent une mémoire tampon interne configurable et l'extraction de messages.

o JxtaBiDiPipe et JxtaServerPipe

- Bâtis sur les pipes, endpoint des messages, et une bibliothèque fiable,
- Fournissent des canaux de communications bidirectionnels, fiables et sûres,
- Exposent une interface de message,
- Pas d'extraction de messages, les applications doivent s'assurer que la taille d'un message ne dépasse pas 64K.

JxtaServerSocket et JxtaServerPipe fournissent un pipe d'entrée pour traiter les demandes de connexion et pour négocier les paramètres de communication. D' autre part, JxtaSocket et JxtaBiDiPipe lient les pipes privés dédiés indépendants de la demande de connexion pipe.

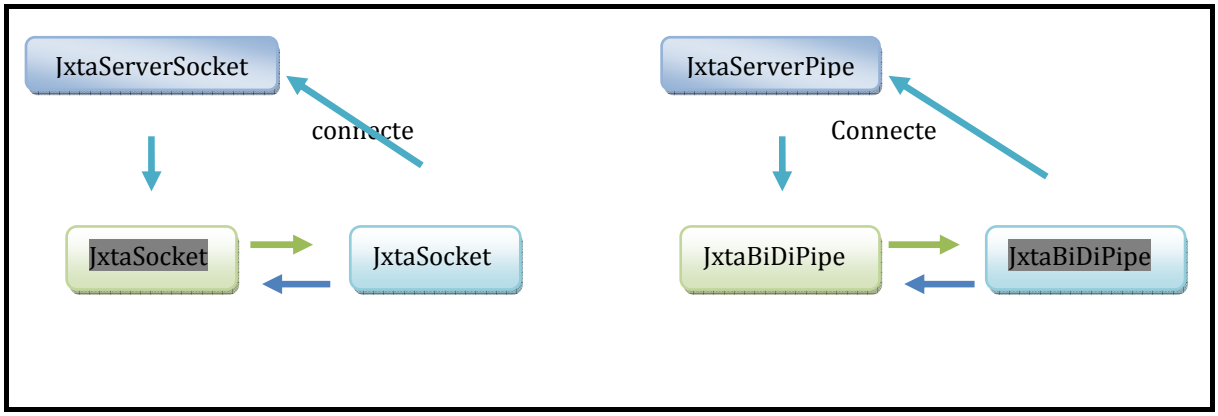


Figure 2.2- La connexion par les Pipes.

- **Annonces (Advertisements)**

Une annonce (Advertisement) est un document XML qui décrit, nomme et publie un peer, peergroup ou un service du réseau JXTA.
Les annonces doivent respecter les normes de codage, les étiquettes et le contenu.
Elles sont utilisées pour l'échange d'informations sur ce qui est disponible dans le réseau JXTA.

Les protocoles JXTA définissent les types des annonces suivantes:

- **Peer Advertisement** - décrit les ressources des peers. La principale utilisation de cette annonce est de détenir des informations spécifiques sur les pairs, comme son nom, son ID, les Endpoints disponibles, et tous les attributs run-time dont l'individuel group services veut publier (comme le fait d'être un peer rendezvous pour le peer group).

- **Peer Group Advertisement** - décrit les ressources spécifiques des peer group, telles que le nom, l'ID du peer group, la description, la spécification et les paramètres de service.

- **Pipe Advertisement** - décrit un canal de communication (pipe), et est utilisé par le pipe service pour créer les canaux d'écoute et d'émission (input pipe, output pipe) Endpoints. Chaque Pipe Advertisement contient une symbolique ID, un pipe type (point-to-point, propagate, secure, etc.) et un pipe ID unique.

- **Module Class Advertisement** - décrit un module de classe. Son but principal est de documenter officiellement l'existence d'un module de classe. Il comprend un nom, une description et une ID unique (ModuleClassID).

- **ModuleSpecAdvertisement** - définit la spécification d'un module. Son objectif principal est de fournir des références à la documentation nécessaire afin de créer des implémentations conformes de cette spécification. Un deuxième objectif est, éventuellement de permettre les instances en exécution, utilisables à distance, par la publication des informations telles que Pipe Advertisement. Il comprend le nom, la description, l'ID unique (ModuleSpecID), pipe advertisement, et champ de paramètre contenant des paramètres arbitraires interprétés par chaque application.

- **ModuleImplAdvertisement** - définit l'implémentation d'un module de spécification. Il comprend un nom, ModuleSpecID associés, ainsi que du code, package, et champs de

paramètres qui permettent à un peer de récupérer les données nécessaires pour exécuter l'implémentation.

- **Rendezvous Advertisement** - décrit le peer qui agit comme un peer rendezvous pour un peer group donné.
- **Peer Info Advertisement** - décrit les infos sur un peer. La principale utilisation de cette annonce est de détenir des informations précises sur l'état actuel d'un peer, comme le temps de fonctionnement, nombres de message entrants et sortants, l'heure du dernier message reçu, et le temps du dernier message envoyé.

Chaque annonce est représentée par un document XML. L'annonce est composée d'une série d'éléments hiérarchiquement organisés. Chaque élément peut contenir des données ou des éléments supplémentaires. Un élément peut aussi avoir des attributs qui sont composés de (nom-valeur string pairs). Un attribut est utilisé pour stocker les méta-données, ce qui permet de décrire les données de l'élément.

Un exemple de Pipe Advertisment :

```
<?xml version="1.0"?>
<!DOCTYPE jxta:PipeAdvertisement>
<jxta:PipeAdvertisement xmlns:jxta="http://jxta.org">
<Id>
urn:jxta:uuid-
59616261646162614E504720503250338E3E786229EA460DADC1A176B69B731504
</Id>
<Type>
JxtaUnicast
</Type>
<Name>
TestPipe
</Name>
</jxta:PipeAdvertisement>
```

2.7. La sécurité

Les réseaux dynamiques P2P, tels que le réseau JXTA, nécessite de soutenir différents niveaux d'accès aux ressources.

Les peer JXTA donnent un rôle de base de modèle de confiance, dans lequel un peer unique agit sous l'autorité qui lui a été accordée par un autre peer de confiance pour effectuer une tâche particulière.

Cinq exigences de sécurité doivent être fournies:

- **Confidentialité -** garantit que le contenu d'un message ne soit pas divulgué aux personnes non autorisées.
- **Authentification -** garantit que l'expéditeur est celui qu'il prétend être.
- **Autorisation -** garantit que l'expéditeur est autorisé à envoyer un message.
- **L'intégrité des données -** garantit que le message n'a pas été modifié accidentellement ou délibérément en transit.
- **Réfutation –** le message est envoyé par un expéditeur identifié, ce n'est pas une copie d'une réponse transférée précédemment.

Les messages XML offrent la possibilité d'ajouter des méta-données comme les credentials, les certificats, les digests et les clés publiques aux messages JXTA afin de satisfaire le respect des exigences de sécurité de base.

L'intention des protocoles JXTA est d'être compatible avec une large couche de transport, accepter des mécanismes de sécurité pour les architectures à base de message, comme SSL (Secure Sockets Layer) et de l'Internet Protocol Security (IPSec).

▪ Identificateurs (IDs, Identifiants)

JXTA offre une large sélection des identifiants pour une identification unique de chaque ressource du réseau. Actuellement, il existe six types d'entités JXTA qui ont des types ID définis: les peers, des peer groups, les pipes, le contenu, le module des classes, et le module de spécifications.

Les URN (uniform resource locator) sont utilisés pour exprimer les identificateurs de JXTA (JXTA IDs). Une URN est une chaîne de caractères utilisée pour identifier une ressource dans le réseau.

JXTA utilise un identificateur (URN) de 128 bit pour référencer n'importe quelle entité dans le réseau.

Les identificateurs sont utilisés comme des pointeurs de références. Dans la plate-forme de référence, les identificateurs sont utilisés pour l'indexation des noms de fichiers et pour la recherche.

Un exemple du peer ID : urn:jxta: uuid- 59616261646162614A78746150325033F3BC76FF13C2414CBC0AB663666DA53903
Un exemple du pipe ID : urn:jxta:uuid- 59616261646162614E504720503250338E3E786229EA460DADC1A176B69B731504

2.8. Architecture de JXTA

Jxta est compose de trois couches : couche Noyau, couche Services et couche Application.

- **La couche noyau :** Elle fournit aux développeurs :

  - ✓ Les mécanismes essentiels tels que les peer groups, peer IDs, peer monitoring, etc.
  - ✓ Les mécanismes de communication tels que les pipes, les annonces.
  - ✓ Les mécanismes de sécurité
  - ✓ Les mécanismes de gestion des peers.

  Ces mécanismes permettent de bâtir des services et des applications dans un environnement d'exécution sûr et multi-plateformes.

- **La couche services :** Elle permet de faciliter le développement des applications. C'est une librairie qui étend les fonctionnalités offertes par la couche noyau.

  Les principaux services proposés dans cette couche :

  - ✓ mécanismes de recherche et d'indexation (JXTA Search)
  - ✓ partage de ressources (JXTA cms et JXTA gisp)
  - ✓ stockage (jxtaspaces)
  - ✓ traduction de protocoles (JXTAxmlrpc, jxta-rmi, jxtasoap)
  - ✓ calcul distribué, etc.

- **La couche applications :** Au sein de la Couche Applications, les applications seront développées sur base des services disponibles dans la Couche Services accédant aux mécanismes de base fournis dans la Couche Noyau.

Il est toutefois important de remarquer que le but du projet JXTA n'est pas le développement d'applications mais la fourniture des mécanismes fondamentaux indispensables au développement d'applications reposant sur la technologie P2P.

La conception des applications est laissée au soin de la communauté d'utilisateurs et développeurs JXTA.

Afin de mettre en place un Service JXTA, il faut respecter l'architecture suivante :

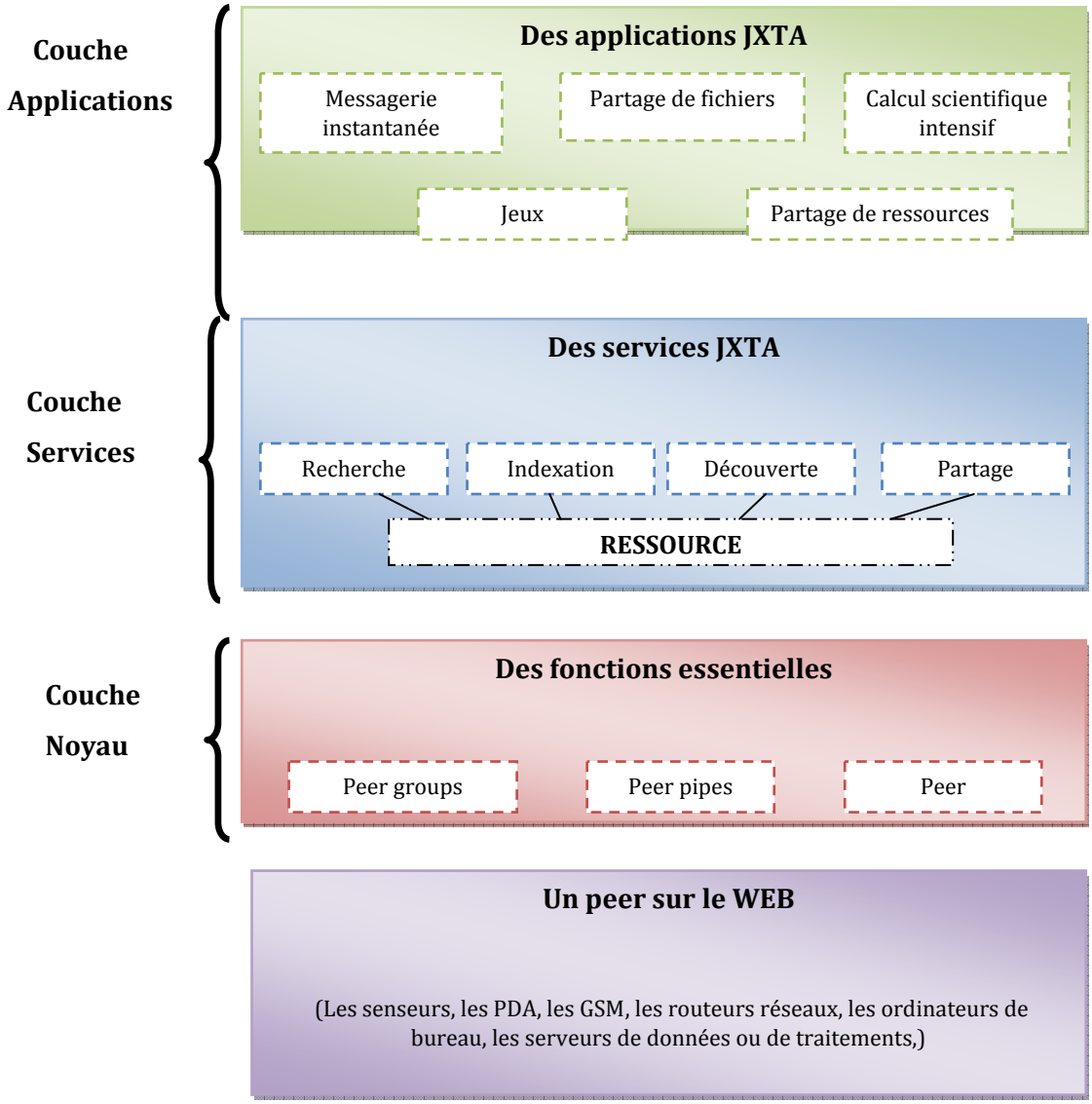


Figure 2.3- Les différentes couches de l'architecture JXTA

2.9. L'organisation du réseau

JXTA est un réseau Ad-hoc, multi-hop, et adaptatif composé de pairs connectés. Les connexions dans le réseau peuvent être transitoires donc le message de routage entre les pairs est non-déterministe. Les Peers peuvent rejoindre ou quitter le réseau à tout moment, ce qui aboutit à des informations de routage dynamique.

Le point commun entre les diverses applications JXTA est qu'ils communiquent tous en utilisant les protocoles JXTA.
L'organisation du réseau n'est pas mandatée par l'interface JXTA, mais en pratique il existe quatre sortes de peers :

- **Minimal edge peer :** c'est un pair qui peut envoyer et recevoir des messages, mais ne cache pas les annonces, ni renvoie les messages à d'autres pairs. Les peers sur les dispositifs avec des ressources limitées (par exemple, un PDA ou un téléphone cellulaire) seraient des Minimal edge peer.

- **Full-featured edge peer** : est un pair qui peut envoyer et recevoir des messages et cache les annonces. Un simple pair répond à une requête discovery par les informations dans son cache des annonces, mais il ne transmet pas les requêtes discovery. C'est le type des pairs le plus répandu.

- **Rendez-vous peer :** c'est une infrastructure, il aide les pairs par la propagation des messages, la découverte des annonces et des routes, et surtout il tient à jour une topologie de l'infrastructure d'autres pairs, qui a alors utilisé pour contrôler la propagation et la maintenance de la table de hachage distribuée. Chaque peer group possède son propre ensemble de rendez-vous peer et peut avoir autant de rendez-vous peer, selon les besoins. Edge peers envoient des requêtes de découvertes et de recherches au rendez-vous peer qui à leur tour, peuvent transmettre les requête qu'ils ne peuvent pas traiter à d'autres rendez-vous peer utilisant la topologie mappée dans la table de hachage distribuée.

- **Relay peer :** est une infrastructure qui aide les peers non adressables par (pare-feu / NAT) par le relais de message. un peer peut demander une boîte de message dans la mémoire du relay peer afin de faciliter le relais de message chaque fois que nécessaire.

2.9.1 Index distribué de partage de ressources (SRDI)

Le SRDI (Shared Ressources Distributed Index) est un service puissant qui fournit un mécanisme efficace de recherche de données basé sur la propagation des requêtes au sein du réseau JXTA.

Les Rendez-vous peer indexent les ressources publiées par un Edge peer, les Edge peers utilisent le service SRDI pour publier les annonces aux Rendez-vous peers, utilisant l'hiérarchie des Rendez-vous peer. Les requêtes sont propagées entre les Rendez-vous peers pour réduire le nombre de peers impliqués dans la recherche des annonces.

Chaque Rendez-vous peer maintient sa propre liste de Rendez-vous peer dans le peer groupe. Quand un peer publie une nouvelle annonce, l'annonce est indexée par le service SRDI utilisant des clés telles que le nom ou l'ID de l'annonce. Seuls les indexs des annonces sont publiés pour les Rendez-vous peers par le SRDI, ce qui réduit la quantité des données qui doivent être sauvegardées au niveau des Rendez-vous peers. Le Rendez-vous peer propage également l'index aux autres Rendez-vous peers. Exemple:

A est un edge peer qui est configuré pour utiliser le Rendez-vous peer R1.

Quand ce peer initialise une requête de recherche ou de découverte, elle est initialement envoyée à R1 et autres peers dans son réseau local via le multicast.

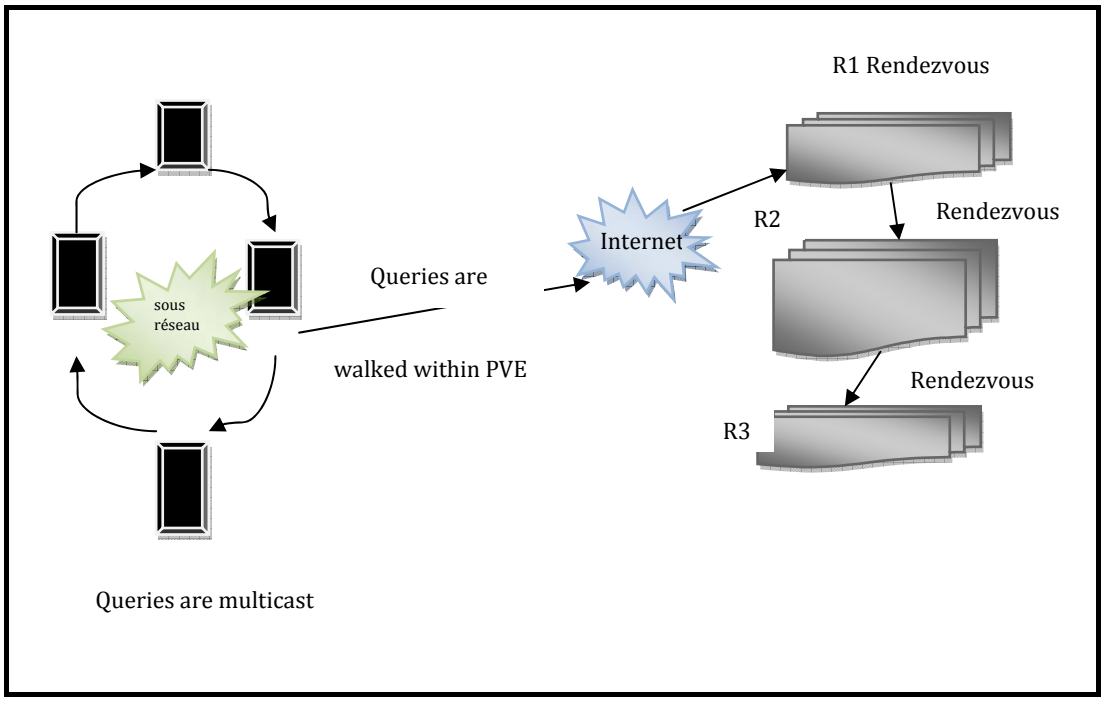


Figure 2.4- propagation de requêtes via les peers Rendezvous.

- Les requêtes dans un réseau local sont propagées aux peers du sous-réseau, utilisant le broadcast ou le multicast. Les peers qui les reçoivent répondent directement à celui qui les a envoyées (si l'information existe dans leurs caches).

- les requêtes au-delà du réseau local sont envoyées au Rendez-vous peer, ce dernier essaye de répondre à la requête en consultant son cache local. Si le cache contient l'information il l'envoie directement au peer. Si son cache SRDI contient l'index de la ressource, il informe le peer qui l'a publié pour qu'il réponde au peer ayant envoyé la requête.

- Si l'information n'existe pas dans son cache, il va utiliser un algorithme (limited-range walker) pour la mise en marche de la recherche de Rendez-vous peer qui l'index. Une fois que la requête atteind le peer, ce dernier va répondre directement à celui qui a généré la requête.

Le service SRDI utilise un schéma d'adressage SHA1 où les 160 bits espace d'adressage de hachage sont divisés entre une liste ordonnée de nœuds de rendez-vous. Quand les indices sont reçus, ils sont hachés, pour déterminer leur adresses de réplication, ces dernières seraient mappées sur leurs lieus de destination :(réplique de rendez-vous).

La figure ci-dessous montre une vue logique de la façon dont le service SRDI fonctionne. Une fois un peer A publie une série d'annonces, un ensemble d'indexes sous la forme d'un message SRDI est envoyé à son rendez-vous, RDV1 dont ses indexes sont stockés puis reproduits (en fonction de leur hash mapping) sur les rendez-vous 2, 3, et 4. Le peer C émet alors une requête pour une annonce A, qui est parcourue dans rendez-vous 2, puis mappée à rendez-vous 3, puis finalement transmise au peer A.

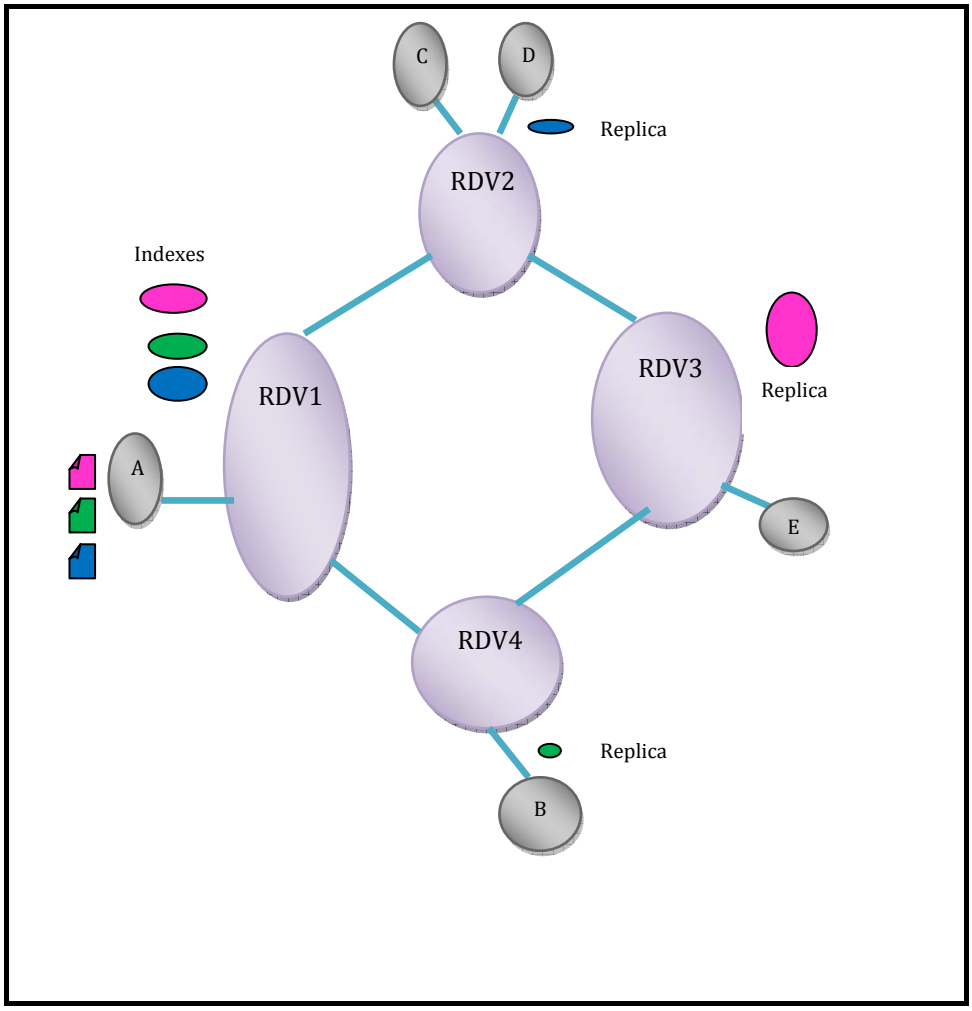


Figure 2.5- le fonctionnement du SRDI.

2.9.2 Firewalls et NAT

Un peer derrière un pare-feu peut envoyer un message directement à un des peers en dehors d'un pare-feu mais les peers à l'extérieur du pare-feu ne peuvent pas établir une connexion directe avec un peer derrière le pare-feu. C'est identique pour les peers qui se trouvent derrière un périphérique NAT.

Pour que les peers JXTA puissent communiquer à travers un pare-feu, les conditions suivantes doivent être respectées.

- Au moins un des peer d'un peer group à l'intérieur du pare-feu doit être au courant d'au moins un des peers de l'extérieur du pare-feu.
- Le peer à l'intérieur et le peer à l'extérieur du pare-feu doivent être au courant les uns des autres et doivent supporter un protocole de transport commun (HTTP ou TCP).

- Le pare-feu doit au moins permettre des connexions sortantes HTTP ou TCP.

La figure ci-dessous représente un scénario d'acheminement d'un message à travers un pare-feu. Dans ce scénario, JXTA Peers A et B veulent passer un message, mais le pare-feu les empêche de communiquer directement. D'abord JXTA Peer A établit une connexion avec des pairs C utilisant un protocole tel que HTTP qui peut pénétrer dans le pare-feu. Ensuite Peer C établit une connexion avec les peers B utilisant un protocole tel que TCP / IP. Donc une connexion virtuelle est établie entre les peers A et B.

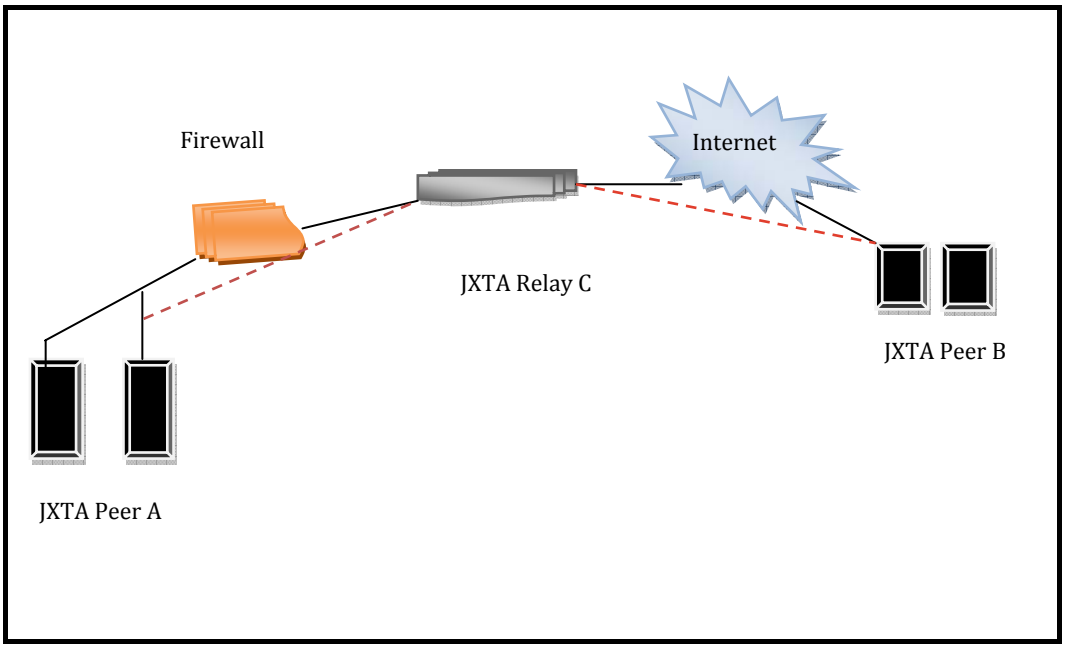


Figure 2.6- L'acheminement d'un message à travers un pare-feu.

2.10 Conclusion

L'architecture de la plateforme JXTA est très complexe, elle s'appuie sur des standards ouverts tels que XML, JAVA et sur des concepts qui font d'UNIX un puissant et flexible système d'exploitation : SHELL avec commandes interopérables, ainsi que l'utilisation de pipes pour accomplir des tâches complexes (communication entre peers).

JXTA est une plateforme de programmation réseau, définie comme étant un ensemble de six protocoles que nous détaillerons dans le chapitre qui suit.

Chapitre 3

Les concepts de JXTA

- **Introduction**
- **Qu'est ce qu'un module en JXTA ?**
- **Qu'est ce qu'un Service en JXTA ?**
- **Publication d'un Service JXTA**
- **Conclusion**

Chapitre 3. Les protocoles et Services JXTA

3.1. Introduction

JXTA définit un ensemble de protocoles [ANT 02] [DAL 03], ces derniers sont des formats de messages XML qui permettent la communication entre les peers.

Il y a six protocoles qui sont utilisés par les peers pour se découvrir, publier et découvrir des ressources dans le réseau, et pour communiquer et acheminer les messages.

La figure suivante représente l'architecture de ces six protocoles:

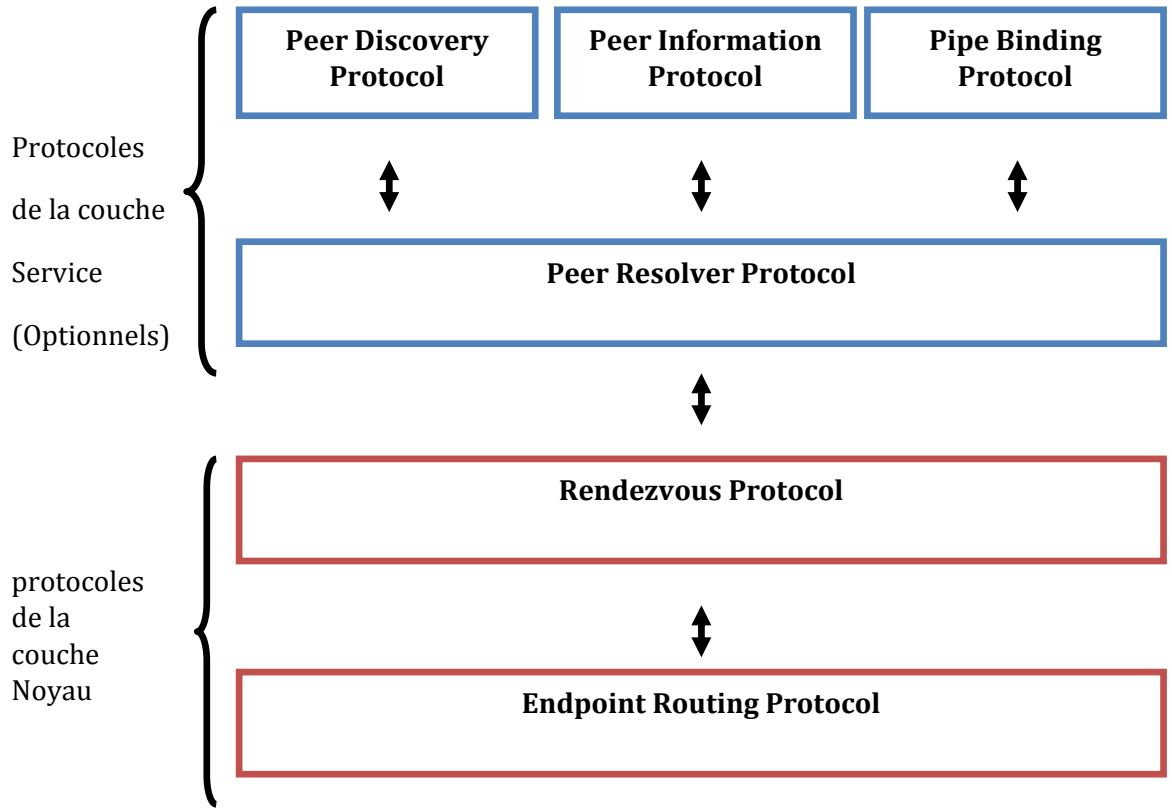


Figure 3.1- La pile de protocole JXTA

Les six protocoles sont :

- **Peer Discovery Protocol (PDP) :** permet à un peer d'annoncer ses propres ressources (peer, peer group, pipes, services) et de découvrir les ressources d'autres peers. Chaque ressource est décrite et publiée par une annonce.

- **Peer Information Protocole (PIP) :** permet aux peers d'obtenir des informations sur d'autres peers.

- **Peer Resolver Protocol (PRP) :** permet aux peers d'envoyer des requetes génériques à un ou plusieurs peers, et de recevoir une ou plusieurs réponses. Les requêtes peuvent être propagées à tout les peers d'un peer groupe ou à un peer spécifique.
- **Pipe Binding Protocol (PBP) :** permet d'établir un canal de communication virtuel (pipe) vers un ou plusieurs peers.

- **Endpoint Routing Protocol (ERP) :** permet d'obtenir les informations sur les routes disponibles pour envoyer un message à un autre peer.

- **Rendezvous Protocol (RVP) :** permet de propager les messages dans le réseau.

Tous les protocoles JXTA sont asynchrones et se basent sur le modèle requete/réponse. Un peer dans le réseau JXTA utilise un des protocoles pour envoyer une requête à un ou plusieurs peers de son peer group, en contrepartie il peut ne recevoir aucune, une seule ou plusieurs réponses à sa requête.

Par exemple, un peer utilise le PDP pour envoyer une requête de découverte de tous les peers dans le Net peer group. Dans ce cas, plusieurs peers peuvent répondre avec des réponses de découverte.

Un deuxième exemple, un peer peut envoyer une requête de découverte d'un pipe nommé "aardvark". Si le pipe n'existe pas, le peer ne reçoit pas de réponse à sa requête de découverte.

Les peers ne doivent pas implémenter tous les six protocoles, ils ne doivent implémenter que les protocoles dont ils ont à besoin.

3.1.1 Peer Discovery Protocol (PDP)

Le Peer Discovery Protocol est utilisé par un peer pour découvrir les ressources des autres peers, les ressources sont représentées par les annonces. Une ressource peut être un peer, un peer group, un pipe, un service ou une autre ressource qui a une annonce [SUN 07].

Le PDP permet à un peer de trouver des annonces dans le réseau. Le PDP est un protocole de découverte défini par défaut par l'utilisateur pour tous les peer groups et le Net peer group, si un peer group n'a pas définit un service de découverte.

Il existe plusieurs façons pour découvrir l'information distribuée dans le réseau. L'implémentation actuelle du JXSE utilise la combinaison de deux méthodes, le multicast sur IP dans un LAN, les rendezvous et les relais dans un WAN.

Les nœuds rendezvous fournissent un mécanisme pour diriger les requêtes dans le réseau (vers tous les pairs connus) pour la découverte de l'information d'une façon dynamique. Parmi les peers qui reçoivent les requêtes, les peers simples peuvent éventuellement répondre aux requêtes, alors que les peers rendezvous permettent en plus de faire suivre la requête.

PDP utilise le Peer Resolver Protocol pour envoyer ses requêtes et recevoir les réponses.

3.1.2 Peer Information Protocol (PIP)

Le Peer Information [SUN 07][WIL 02]Protocol fournit un ensemble de message pour obtenir des informations sur les peers (état, trafic, etc.) identifiés à l'aide du Peer Discovery Protocol. Ces informations peuvent être utilisées pour le déploiement commercial ou le déploiement interne des applications JXTA.
Par exemple, dans le déploiement interne, l'information peut être utilisée par le département IT afin de contrôler le comportement d'un nœud et pour rediriger le trafic du réseau afin d'améliorer la performance globale.
L'information se fait par : l'envoi d'un message Ping afin de vérifier l'existence d'un peer et pour obtenir les informations sur ce dernier. Le message Ping précise le contenu de la réponse si elle est complète (peer advertisement) ou une simple réponse (Alive et uptime).

L'envoi du message Peerinfo afin de répondre à un message Ping. Ce message contient les informations : credential of the sender, the source peer ID, target peer ID, uptime, et le peer advertisement.

3.1.3 Peer Resolver Protocol (PRP)

Le Peer Resolver Protocole **[SUN 07] [WIL 02]** permet l'envoi des requêtes génériques à un ou plusieurs peers et l'identification des réponses correspondante. Une requête peut être adressée à un peer spécifique utilisant l'unicast message, ou bien peut être propagée à tous les membres d'un peer groupe via le peer rendezvous.

Le PRP est un protocole de base, qui est utilisé par d'autres protocoles. Par exemple, le protocole PDP l'utilise pour découvrir les ressources d'un peer (le contenu de l'échange est constitué par des annonces), alors que le protocole PIP l'utilise pour communiquer les informations d'état.

Chaque requête est adressée à un nom de contact spécifique. Ce nom de contact définit la sémantique particulière de la requête et de sa réponse, mais n'est pas associée à un Peer spécifique. Une requête peut être reçue par n'importe quel nombre de Peers d'un PeerGroup, et sera traitée par un Peer en fonction du nom du contact s'il est défini sur ce Peer. Ce nom de contact permet de savoir à quel service de haut niveau est destiné le message pour être traité.

Par exemple, les protocoles Peer Discovery Protocol, Peer Information Protocol et Pipe Binding Protocol s'inscrivent comme service auprès de PRP au moyen d'un nom de contact en vue de recevoir les messages qui leur sont destinés.
Dans les deux implémentations de référence (Java et C) de la plateforme, les instances de services génèrent un nom de contact unique sur le Peer mais identique pour les instances situées sur d'autres Peers. Par convention, on concatène le nom du service et l'identifiant du groupe pour rendre ce nom de contact commun au groupe. A cela, on ajoute des paramètres additionnels pour assurer l'unicité au sein d'un Peer.
Les peers peuvent également participer à un « index distribué des ressources partagées » (Shared Resource Distributed Index (SRDI)). Cet index fournit un mécanisme générique que les services JXTA peuvent utiliser soit pour diriger les requêtes dans une direction où elles ont plus de chance d'obtenir une réponse, ou pour propager des messages vers des Peers susceptibles d'être intéressés par le type de messages à propager.

Cependant, PRP n'offre aucune garantie quant à la réception des requêtes ou des réponses correspondantes. Il utilise comme PDP le principe du "best effort" pour envoyer les requêtes ; ce principe maximise la chance d'obtenir une réponse si elle peut être obtenue. De même, le protocole n'est pas tenu de répondre à une requête. Il n'y a pas non plus de garantie qu'un message SRDI soit honoré.
PRP ne suppose pas non plus la présence d'un transport fiable des messages.

La propagation des requêtes à l'ensemble suivant de Peers est laissée à RVP.
RVP est responsable de déterminer l'ensemble de Peers qui doivent recevoir le message à propager. Cependant, il ne propage pas automatiquement un message, il laisse le service (le contact de la requête) se charger de déterminer si le message doit être propagé ou non.
La politique de PRP est la suivante : si le contact de la requête ne donne pas pour instruction de "jeter" la requête et si son Peer est un Rendezvous, alors la requête est repropagée (en fonction de la gestion du TTL et de la détection de boucle de RVP). Si le contact de la requête le désire, une requête identique peut être réémise avec le Peer comme émetteur.

3.1.4 Pipe Binding Protocol (PBP)

Le Pipe Binding Protocol [SUN 07] [WIL 02] permet de relier deux ou plusieurs peers endpoint au moyen des pipes et en fonction des annonces du pipe reçu ou envoyé.

Un pipe est une file abstraite de messages qui supporte des opérations de création, de destruction, de connexion et de déconnexion à un ou plusieurs peers, d'envoi et de réception de messages.

La requête de connexion à un pipe se fait selon :

- L'envoi d'un message PBP query pour trouver les peers utilisant le même pipe advertisement.

- L'envoi d'un message PBP answer c'est la réponse au message précédent envoyée par tous les peers qui écoutent le pipe.

Le PBP utilise PRP qui est toujours nécessaire pour lier les peers endpoint entre eux quel que soit le type du protocole de transport du peer endpoint (TCP, http, UDP, etc.)

Le PBP n'impose pas la réponse à une requête de résolution.

3.1.5 Endpoint Routing Protocol (ERP)

Le Endpoint Routing Protocol [SUN 07] [DAL 03] permet à un Peer de trouver les routes disponibles pour l'envoi d'un message à un autre Peer. Pour obtenir ces routes, le Peer émetteur interroge un Peer Router et ce dernier lui fournit une liste de Peer Routers menant jusqu'au Peer de destination. Cependant, il est fréquent que, lorsque deux Peers communiquent ensemble, ils ne soient pas directement connectés par le même protocole de transport ou qu'ils soient connectés de manière "transient" (dial-up) ou soient séparés par un NAT ou un Firewall. Dans ces situations, un Peer Router jouera le rôle d'intermédiaire pour le Peer inaccessible de l'extérieur de son réseau interne.

Un Peer a le choix de devenir un Peer Router en choisissant d'implémenter le protocole ERP.

ERP définit un ensemble de messages de requêtes et leur traitement permettant aux Peers de router les messages jusqu'à destination. Lorsqu'un Peer doit envoyer un message jusqu'à un Peer EndPoint déterminé, il vérifie tout d'abord dans sa cache locale s'il connait déjà une route menant `a ce Peer EndPoint. Si ce n'est pas le cas, il envoie un message contenant une requête de résolution de route aux Peer Routers qu'il connait (Les Peer Routers peuvent être connus d'un Peer par pré-configuration ou découverts dynamiquement au moyen d'une recherche de Peer).
Suite à la réception d'une requête de résolution de route, si un Peer Router connait une route menant à la destination, il répond à la requête en retournant l'information de route (un liste ordonnée du Peer Router `a la destination finale) qui sera utilisée par l'´emetteur du message et les Peers intermédiaires. S'il ne connait pas de route, il effectue une recherche auprès des Peer Routers auxquels il est connecté. Ce mécanisme est récursif car à chaque intermédiaire, si la route d'un message est obsolète elle nécessitera une nouvelle recherche de route.

Pour éviter les boucles, chaque Peer EndPoint intermédiaire ajoutera une information aux messages envoyés par un Peer. Cette trace sera utilisée pour ne plus propager un message qui boucle, un message dupliqué ou pour enregistrer une nouvelle information de route par un Peer Router.

Comme pour les autres protocoles JXTA, ERP fournit les mécanismes essentiels au routage des messages (format des routes et des messages pour les échanger) et laisse la possibilité aux développeurs d'implémenter des services de routage plus efficaces et plus intelligents optimisant les routes choisies, connues et publiées.

3.1.6 Rendezvous Protocol (RVP)

Le Rendezvous Protocole [SUN 07] [DAL 03] est utilisé pour la propagation des messages et son contrôle (TTL, détection de boucle,…) au sein d'un peer group. Il permet aux peers de se connecter aux services (donc être capable de communiquer sur le réseau) et il contrôle la propagation (gestion de la durée de vie du message, détection des boucles, etc.).

Comme ERP introduit ses Peers Routers, RVP quant à lui introduit des Peers jouant un rôle spécial appelé Rendezvous Peers. Ces Rendezvous Peers re-propagent les messages qu'ils ont reçus. Un Peer peut devenir dynamiquement un Rendezvous Peer et/ou peut négocier dynamiquement un bail avec un Rendezvous Peer via l'EndPoint Routing Protocol.
Avant l'utilisation d'un Rendezvous Peer pour la propagation des messages dans le réseau, il faut d'abord obtenir un bail (lease) qui spécifie la période d'avoir une connexion alloué à un Rendezvous Peer. Pour cela le RVP définit 3 types de messages.

- ✓ **Lease Request message** – utilisé par les peers pour un bail de connexion à un Rendezvous Peer.

- ✓ **Lease Granted message** – est une réponse aux Lease Request message, envoyé par le Rendezvous Peer aux peers qui ont demandé un bail de connexion.

- ✓ **Lease Cancel message** – utilisé par un peer pour se déconnecter d'un Rendezvous Peer.

Les protocoles PRP et PBP reposent sur ce protocole.

3.2. Qu'est ce qu'un MODULE?

Un Module dans JXTA est un ensemble de fonctionalités qu'un peer peut utiliser, offrir ou obtenir d'un autre peer.
Un Module est une abstraction de services et d'applications dans le modèle de couches JXTA.

Un Module peut être initialisé, lancé et arrêté.

Dans JXTA , une application est un module qui peut être chargé et exécuté sur un périphérique.

3.3. Qu'est ce qu'un Service ?

Un service dans JXTA a les propriétés du Module tout comme une application JXTA

C'est aussi une ressource qui sert dans l'implémentation des protocoles JXTA, il peut être rattaché au niveau du pair (level peer) ou du peer group (peer group level).

Il ya plusieurs types de services décrits dans JXTA:

- Access Service
- Discovery Service
- Endpoint Service
- Membership Service
- Peer Info Service
- Pipe Service
- Resolver Service

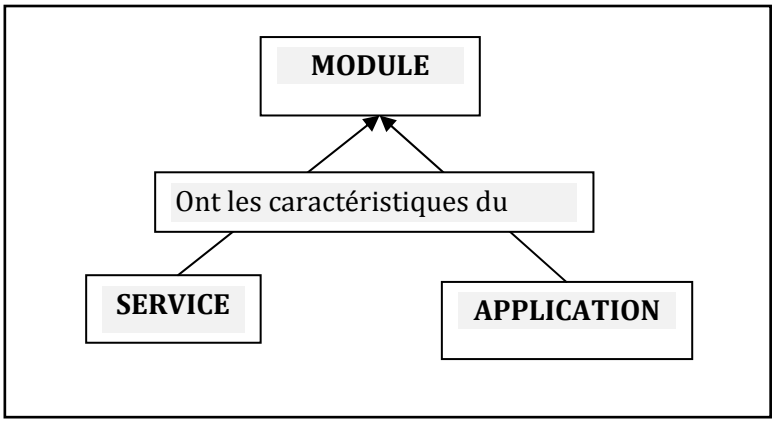


Figure 3.2 Définition d'un service

Il ya trois types d'advertisements associés au Module :

- **Module Class Advertisement** : Cet Advertisement déclare simplement l'existence d'un Module mais ne fournit aucune information sur son implémentation ou sur son utilisation, il contient le Module Class ID, le nom du Module et sa description.
- **Module Specification Advertisement :** Ce Module inclut beaucoup de précision sur le Module en fournissant des caractéristiques d'interfaçage, il contient un unique Module Specification ID, son nom, le nom de son créateur, une URI pour la spécification de la

documentation, la spécification du nombre de version et les paramètres de la spécification. Il peut aussi contenir un advertisement pour le pipe pour communiquer avec les autres instances des Modules de cette spécification ou bien le Module Specification ID du module proxy qui peut être employée pour communiquer avec les autres instances des Modules de cette spécification. Récemment, il peut aussi contenir un Module Specification ID d'un module d'authentification qui peut être requise pour communiquer avec les autres instances des Modules de cette spécification.

- **Module Implementation Advertisement :** Cet Advertisement fournit des informations concrètes implémentation sur le Module Specification. Il contient le Module Specification ID, des informations de compatibilité, le code sur l'implémentation du Module, des informations sur le code de l'implémentation, une URI du package contenant le code du Module Implementation, le nom de l'entité fournissant le Module Implementation ou des paramètres sur le Module Implementation.

3.4. Publication d'un Service JXTA

JXTA a mis à disposition des Sevices en les publiant grâce au ModuleSpecAdvertisement qui n'est rien d'autre qu'une publication de pipe qui peut être utilisé par d'autres peers pour créer des output pipes pour invoquer un Service. Donc comme les Services JXTA ont les caractéristiques des Modules chacun d'eux est identifié de manière unique pat ModuleSpecID.

Nous résumons donc brièvement : les trois types de publications possibles:

(**MCA) ModuleClassAdvertisement :** définit la classe du service et informe sur l'existance de la classe. L'identification est unique grâce à un ModuleClassID.

(MSA) ModuleSpecAdvertisement : définit une spécification de Service. l'identification est unique grâce à un ModuleSpecID. Son rôle est de fournir des références à la documentation nécessaire pour la création d'implémentations conformes à la spécification.

(**MIA) ModuleImplAdvertisement** : définit une implémentation pour une spécification du service donné.

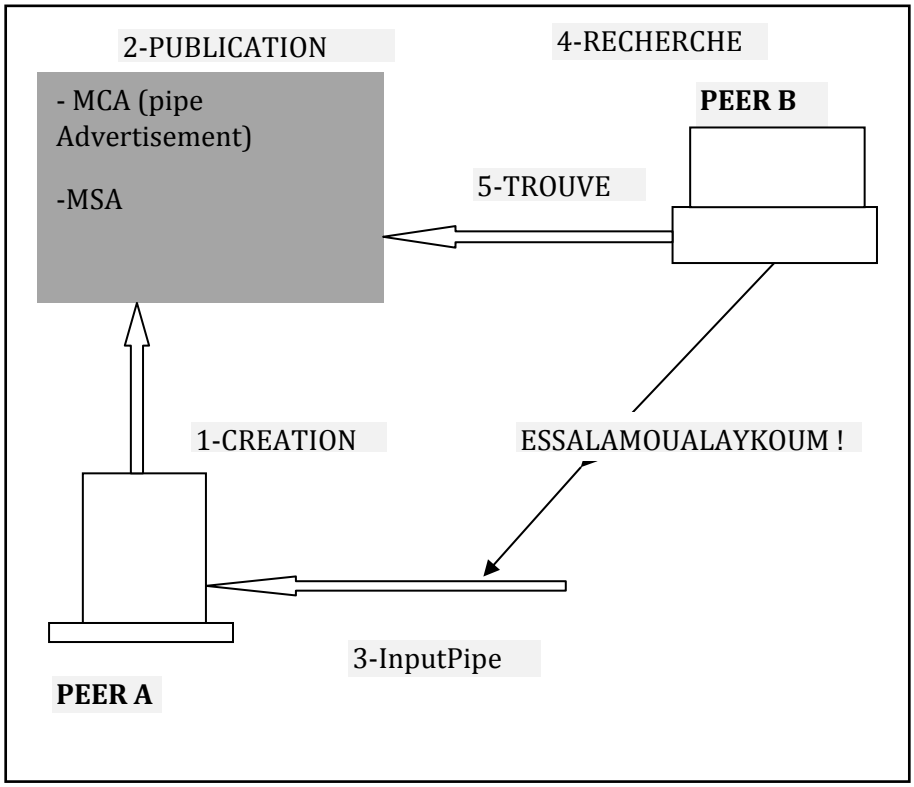


Figure 3.3 Etapes de publication d'un Service JXTA

```
PEER A

Starting Service Peer ....
Start the Server daemon
Reading in file pipeserver.adv
Created service advertisement:
jxta:MSA :

MSID : urn:jxta:uuid-
B6F8546BC21D4A8FB47AA68579C9D89EF3670BB315A
C424FA7D1B74079964EA206
Name : JXTASPEC:JXTA-EX1
Crtr : sun.com
SURI : http://www.jxta.org/Ex1
Vers : Version 1.0

jxta:PipeAdvertisement :
Id : urn:jxta:uuid-9CCCDF5AD8154D3D87A391210404E59BE4B888
209A2241A4A162A10916074A9504

Type : JxtaUnicast

Name : JXTA-EX1

Waiting for PEER B messages to arrive
Server: received message: ESSALAMOUALAYKOUM!
Waiting for PEER B messages to arrive
```

```
PEE B
Starting PEER B ....
Start the PEER B
searching for the JXTASPEC:JXTA-EX1 Service advertisement
We found the service advertisement:
jxta:MSA :
MSID : urn:jxta:uuid-
FDDF532F4AB543C1A1FCBAEE6BC39EFDFE0336E05D31465CBE9
48722030ECAA306
Name : JXTASPEC:JXTA-EX1
Crtr : sun.com
SURI : http://www.jxta.org/Ex1
Vers : Version 1.0
jxta:PipeAdvertisement :
Id : urn:jxta:uuid-
9CCCDF5AD8154D3D87A391210404E59BE4B888209A224
1A4A162A10916074A9504
Type : JxtaUnicast
Name : JXTA-EX1
message "ESSALAMOUALAYKOUM!" sent to the Server
WA ALAYKOUMOUESSALAM ....
```

3 .5. Le Service CMS (Content Management Service)

Le but du CMS (Service de Gestion de Contenu) est d'enregistrer tous les fichiers (text, media,..) stockés sur un peer local (PC) et susceptibles d'être partagés. Le pair désirant partager son contenu avec d'autres pairs doit les créer à travers des annonces (Advertisements).

3.5.1 Caractéristiques du fichier partagé

Chaque fichier partagé doit posséder une unique annonce (Advertisement) décrivant son contenu.

Le format de l'annonce (Advertisement) contient les éléments suivants :

- Name : le nom du fichier qui va être partagé.
- cid : l'ID représentant le fichier à partager, cet ID est unique.
- Length : la taille du fichier à partager en bits.
- Description : une description optionnelle du fichier à partager.

Voici un exemple d'une annonce d'un contenu CMS

```
< ?xml version= 1.0>

< !doctype jxta :contentAdvertisement>

<jxta :contentAdvertisement>

<name>ship.html</name>

<cid> md5:2b9cbd6ab82c8fee8fe2a2b9e7eab78a9</cid>

<type>text/html</length>

<description>page for displaying Model Ships</description>

</jxta:contentAdvertisement>
```

3.5.2 Etapes de mise en place du sous-service CMS

A. Initialisation

la première étape consiste en l'initialisation du service CMS. Ceci nécessite l'instanciation d'un nouveau objet CMS qui contient le service lui-même en appelant la méthode init() qui accepte trois paramètres : le groupe de pairs (PeerGroup) auquel le peer veut accéder contenant le système de gestion de contenu désiré en spécifiant un peer ID et une annonce (Advertisement) pour ce même peer.

B. partage de contenu (sharing content)

Après que le CMS soit initialisé, deux opérations sont possibles : sharing content et querying.
les deux opérations sont nécessaires pour le partage de contenu, la première permet d'obtenir le Content Manager du Service CMS tandis que pour le partage de fichier la méthode share() est utilisée qui contient quatre paramètres :
File object : le fichier à partager.
name : optionnel.
type : le type MIME (exemple :jpg) ; c'est un paramètre optionnel.
desc : description optionnelle du fichier.
Le prototype de la méthode de partage est le suivant :
share(java.io.File , java.lang.String , java.lang.String , java.lang.String)

C. Affichage de la liste du contenu à partager

Le CMS a la possibilité de fournir la liste de tous les fichiers à partager du peer courant, le content object contient le ContentAdvertisement qui possède l'information du fichier à partager, il publie aussi le PipeAdvertisement qui peut être utilisé par le remote peer

3.6 Conclusion

La plateforme JXTA est composée de six protocoles indépendants du langage de programmation , standardisant la manière pour laquelle chaque peer peut découvrir les autres peers , s'auto-organisent dans les groupes de pairs (peergroup), découvrent les services dans le réseau, communiquent avec les autres pairs et surveillent chacun d'eux.
Nous aborderons dans le chapitre qui suit les différentes classes qui ont permis l'implémentation de notre service.

Chapitre 4

Mise en place du Service IISWJ

- **Introduction**
- **Architecture du Service**
- **Configuration des Pairs**
- **Assignation manuelle des IDs**
- **Création des Classes**
- **Exécution du Service**
- **Organisation du Service en vue d'un système de stockage distribué**
- **Conclusion**

Chapitre 4. Mise en place du service IISWJ

4.1. Introduction

Dans ce chapitre, nous allons décrire comment implémenter le Service d'information (IISWJ), qui est composé de deux sous-services : le CMS et l'outil Calculatrice.

4.2. Architecture du Service

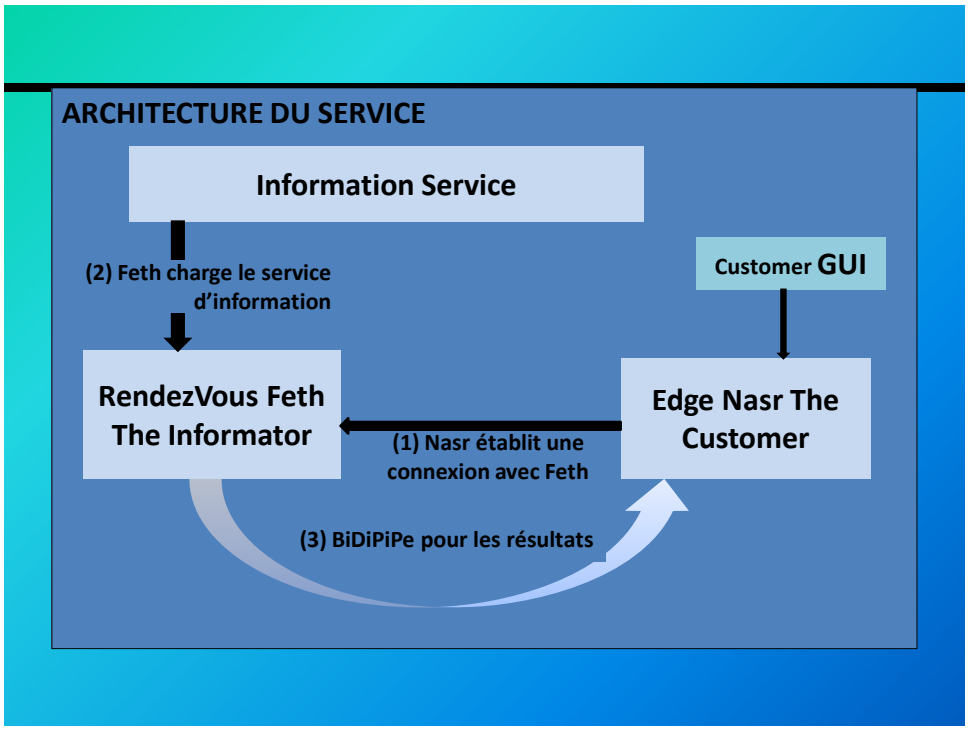


Figure4 .1 Le Service IISWJ

4.2.1 Ses caractéristiques

- ✓ Le service permet le partage de contenu.
- ✓ Le service peut charger et exécuter un service sur un Peer pour d'autres peers.
- ✓ Le Service peut être implémenté de diverses méthodes .
- ✓ Beaucoup d'autres fonctionnalités peuvent être ajoutées dans le futur telles que par exemple : la redirection des clients vers d'autres sous-services.

Nous concluons que le service est extensible.

4.3 Configuration des Peers :

Pour évaluer notre Service, nous l'avons implémenté de diverses méthodes, l'une d'entre elles était la demande de l'utilisation d'une calculatrice standard mais la plus grande difficulté rencontrée était la communication des peers en leur assignant des IP dynamiques car JXSE ne reconnaît pas cette fonction.

Donc pour faire communiquer nos Pairs nous avons eu recours à deux configurations qui ont abouti :

4.3.1 Configuration avec la Virtual box

Virtual Box est un logiciel de virtualisation créé par InnoTek. Il est disponible en tant qu'hôte sur les systèmes d'exploitation : Windows, Linux 32 et 64 bits, FreeBSD 32 et 64 bits et Mac OS X. Il supporte en tant qu'invité : Windows (dont Vista et 7), Linux 2.x, OS/2 Warp, OpenBSD et FreeBSD entre autres1.

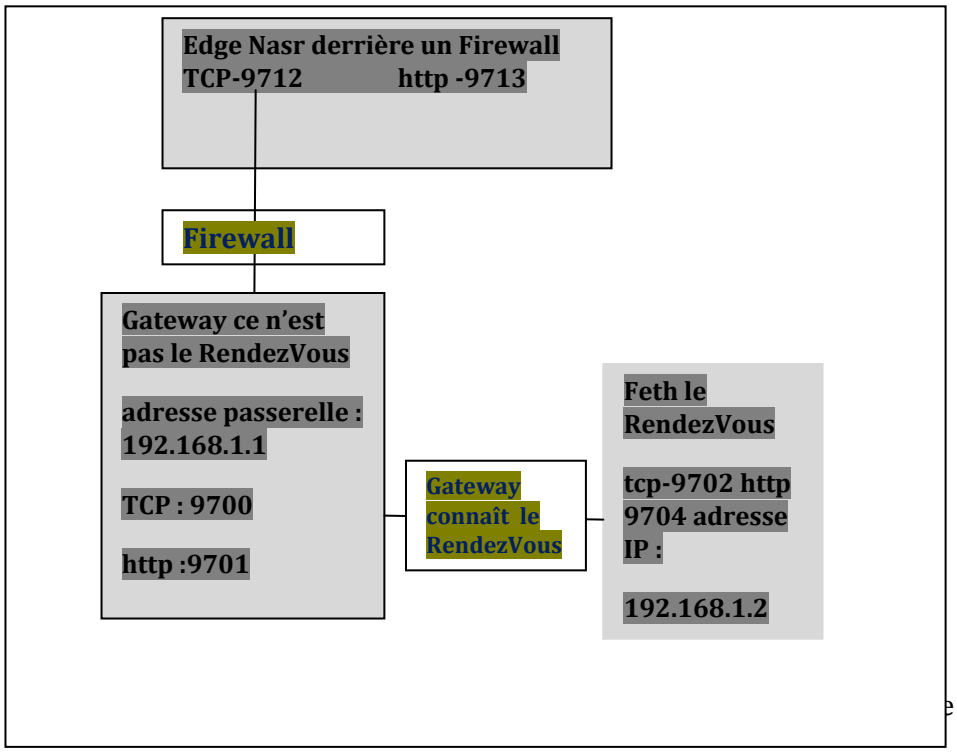
Donc, nous avons téléchargé la Virtualbox dans notre PC à laquelle on assigne une IP statique qui permet aux pairs du réseau JXTA de communiquer.

4.3.2 Configuration derrière Firewall/NAT

La communication des pairs est exécutée sur une machine localisée sur internet, les différents pairs peuvent exister indépendamment ou comme part d'un groupe de pairs. Chaque pair a théoriquement et pratiquement un accès direct aux différents pairs dans le réseau Jxta. Cependant, il ya deux situations :

La première est celle où ce peer résidant derrière un firewall, ce dernier étant désigné pour limiter les communications entrantes et sortantes, ceci dit, les ports utilisés par Jxta seront incapables d'acheminer ces informations.

La seconde qui doit être prise en considération est la NAT (Network Address Translation), si un réseau utilise la NAT, toutes les adresses IP internes ne sont pas visibles au monde extérieur. Notre topologie est la suivante :



4.3.3 Cla sse NetworkConfigurator

NetworkConfigurator fournit une interface simple de programmation de la configuration JXTA (PlatformConfig). C'est une Classe qui fournit des méthodes pour créer une instance de la plate-forme existante, elle permet de créer un Rendez-vous pour la configuration des pairs AD_HOC à partir de zéro.

4.3.3.1.EdgePeer

Un pair qui peut ou peut ne pas être derrière un pare-feu ou NAT (c'est à dire directement adressable ou non). Il est recommandé que cette classe de pairs soit configurée avec TCP / IP (entrants / sortants) et HTTP(sortant seulement), il faut aussi utiliser un relais Peer, et un rendez-vous Peer.

4.3.3.2.PeerRelay,Resndez-vous

Cette Classe de pairs est prévue pour fournir des services d'infrastructure et est généralement directement accessible sur Internet. Il est recommandé que cette classe de pairs soit configurée avec TCP / IP (entrants / sortants) et HTTP (activé entrant seulement), elle agit comme un relais et rendez-vous.

Par défaut le NetworkConfigurator définit un Range/port allant de 9701 à 9799 avec un multicast établi avec ''224.0.1.85'' et le port 1234.

Par contre on a souligné les inconvénients suivants :

- C'est une facilité pour initialiser et charger par la suite une configuration existante mais ceci procure une gestion limitée du cycle de vie de la plate-forme.
- le deuxième inconvénient est dans le sous-réseau physique où l'on peut utiliser uniquement une tranche d'adresses 192.168.0.[1…10].

4.4 Assignation manuelle des identificateurs du service

Dans le système peer to peer, les ressources sont référencées d'une manière à assurer l'unicité de l'identifiant. On dénote six identifiants pour les ressources Jxta : Peer, PeerGroup, Pipes, Content, Module classes, Module specifications.

Format String

L'ID JXTA est constitué de trois parties :

- Namespace identifier – jxta
- Format specifier – urn
- ID – unique value

L'écriture de l'ID entier peut être interprétée par la forme Backus-Naur :

<JXTAURN> ::= ''urn :'' <JXTANS> '' :'' <JXTAIDVAL>

<JXTANS> ::= "jxta"

<JXTAIDVAL> ::=<JXTAFMT> "-" <JXTAIDUNIQ>

<JXTAFMT> ::= 1 * <URN chars>

<URN chars> ::= <trans> | "%" <hex> <hex>

<trans> ::= <upper> |<lower> | <number> |<other> |<reserved>

<upper> ::="A"|"B"|"C"|"D"|"E"|"F"|"G"|"H"|"I"|"J"|"K"|"L"|"M"|"N"|

"O"|"P"|"Q"|"R"|"S"|"T"|"U"|"V"|"W"|"X"|"Y"|"Z"

<lower> ::= ”a”|”b”|”c”|”d”|”e”|”f”|”g”|”h”|”i”|”j”|”k”|”l”|”m”|”n”|

“o”|”p”|”q”|”r”|”s”|”t”|”u”|”v”|”w”|”x”|”y”|”z”

<hex> ::=<number> |”A”|”B”|”C”|”D”|”E”|”F”| ”a”|”b”|”c”|”d”|”e”|”f”

<number> ::= “0”|”1”|”2”|”3”|”4”|”5”|”6”|”7”|”8”|”9”

<other> ::=”(“| “)”|”+”|” ‘ “| “ – “|” . “|” : “|”= “|” @”|” ;”|” §”|”_ “|” ! “|”* “|” , “

<reserved> ::= “ %”| ”/” | “? “| “#”

4.5 Création de classes

4.5.1 Classe CreatingModuleID()

L’assignation des IDs concernant notre Service IISWJ a été attribuée manuellement en l’abscence d’algorithme assurant cette tâche. Les IDs sont relatifs aux ModuleClassID et le ModuleSpecID, ces derniers ont été crées grâce au format présenté ci-dessus

La classe ci-dessous décrit la création des IDs du module.

```
import net.jxta.id .IDFactory ;

import net.jxta.impl.id.UUID.ModuleClassID;

import net.jxta.platform.ModuleSpecID;

public class CreatingModuleID() {

public static final String Name = "InfoService";

   public static void main(String[] args) {

      // Creating a new random module class ID

      ModuleClassID MyModuleClassID = (ModuleClassID)
IDFactory.newModuleClassID();

      // Creating a new random module specification ID

      ModuleSpecID MyModuleSpecID =
IDFactory.newModuleSpecID(MyModuleClassID);
```

Les IDs assignés manuellement sont :

"urn:jxta:uuid-F7A712D25D3047B88656FD706AEDE8DB05";

"urn:jxta:uuid-
F7A712D25D3047B88656FD706AEDE8DBC6A510B2026F4FD59A7DFA4F6712142506";

4.5.2Classe RendezVous Feth, l'Informateur

On initie Le Peer Feth comme RendezVous et on attend que l'utilisateur charge le Service d'information demandé grâce à la méthode loadModule(...).

```
// Creation of network manager NetworkManager.
    MyNetworkManager=new
    NetworkManager(NetworkManager.ConfigMode.RENDEZVOUS,
    Name, ConfigurationFile.toURI());
// Retrieving the network configurator
NetworkConfigurator MyNetworkConfigurator
    =MyNetworkManager.getConfigurator();
// Setting more configuration
MyNetworkConfigurator.setTcpPort(TcpPort);
MyNetworkConfigurator.setHttpPort(HttpPort);
MyNetworkConfigurator.setTcpEnabled(true);
MyNetworkConfigurator.setTcpIncoming(true);
MyNetworkConfigurator.setTcpOutgoing(true);
MyNetworkConfigurator.setHttpEnabled(true);
MyNetworkConfigurator.setHttpIncoming(true);
MyNetworkConfigurator.setHttpOutgoing(false);
```

4.5.3 Classe Edge Nasr (The Customer)

Le code qui suit permet au peer Edge Nasr de consulter Feth.

Nasr implémente l'interface PipeMsgListener qui requit la méthode PipeMsgEvent pour surveiller les messages reçus de Feth.

On initie Nasr comme Edge Peer et on attend jusqu'à ce qu'il se connecte au RendezVous Feth pendant au moins deux minutes.

On retrouve L'Advertisement du Module Specification correspondant au Service d'information, tout en créant le bidirectional pipe en extrayant l'Advertisement correspondant au pipe.

On attend alors une connexion au maximum 30 secondes pour une connexion avec Feth. Si une connexion est établie, on envoie une requête pour demander une information à Feth tout en établissant un Listener pour les réponses.

```
// Preparing the listener and Creating the BiDiPipe
PipeMsgListener MyListener = new
    Edge_Nasr_The_Customer();

// Retrieving the pipe advertisement from the module
    implementation advertisement

ModuleSpecAdvertisement TheModuleSpecAdvertisement =
    Information_Service.GetModuleSpecificationAdvertisemen
    t();
        PipeAdvertisement ThePipeAdvertisement =
    TheModuleSpecAdvertisement.getPipeAdvertisement();
```

4.5.4 La Classe Information_Service()

```
public Information_Service() {

  }

  public static PipeAdvertisement GetPipeAdvertisement() {

     // Creating a Pipe Advertisement

     PipeAdvertisement MyPipeAdvertisement = (PipeAdvertisement)
AdvertisementFactory.newAdvertisement(PipeAdvertisement.getAdver
tisementType());

     PipeID MyPipeID =
IDFactory.newPipeID(PeerGroupID.defaultNetPeerGroupID,
Name.getBytes());

     MyPipeAdvertisement.setPipeID(MyPipeID);

     MyPipeAdvertisement.setType(PipeService.UnicastType);

     MyPipeAdvertisement.setName("Information Service Pipe");

     MyPipeAdvertisement.setDescription("Crée par " + Name);
```

- **Les attributs statiques et non statiques**

Les attributs statiques et non statiques de IISWJ n'incluent pas uniquement son nom mais aussi les éléments textes qui permettent aux peers Feth et Nasr de communiquer à travers les messages. Les attributs non-statiques qui sont instanciés au début incluent un couple d'attributs dont le plus important reste le JxtaServerPipe .

- **Les méthodes statiques**

Les méthodes statiques qui permettent de retrouver les annonces (advertisements) sont :

- **GetPipeAdvertisement()**
- **GetModuleSpecificationAdvertisement()**

La méthode PipeAdvertisement() sera utilisée par les peers par le pipe de communication bidirectionnel.

- **Implémentation des interfaces**

IISWJ implémente les interfaces Service et Runnable, la première utilise un couple de méthode pour le lancement et l'arrêt du Service tandis que la deuxième est utile pour exécuter le JxtaServerPipe dans un thread séparé pour accepter des connections de la part des customers.

4.5.5 La Classe Calculatrice

La méthode intégrée au sein du service IISWJ est une classe permettant au chargement du service l'exécution de la calculatrice qui peut être demandée par le Edge Peer

```
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
```

Pour pouvoir créer l'advertisement du Module Implementation, l'indication du fichier jar est requise.

```
public calculatrice(){
        this.setSize(240, 260);
        this.setTitle("آلة حاسبة");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setLocationRelativeTo(null);
        this.setResizable(false);
        initComposant();
        this.setContentPane(container);
        this.setVisible(true);
    }
        private void initComposant(){
        Font police = new Font("Arial", Font.BOLD, 20);
```

4.6 Exécution du Service

Notre service se subdivise en deux sous-services :

A- Le premier assurant la notion de partage de contenu qui peut être étendue ave d'autres fonctionnalités : **- Affichage du Shared Content List (SCL)**

```
Content[] content = cms.getContentManager().getContent ;

fileList.removeAll() ;

for (int i= 0 ; i<content.length ; i++ {

system.out.println(content[i].getContentAdvertisement.getName());}
```

-Recherche pour affichage d'un contenu particulier

après que le conteu aie été chargé, le peer user pourra fournir une requête pour charger n'importe quel contenu, un peu de code supplémentaire est requis :

```
public GetReqestor ( PeerGroup pg, SearchResult
result, File tempFile, ContentListener listener) throws

InvocationTargetException

url = tmpFile.getAbsolutePath();

this.listener = listener;

class ListRequestor extends
CacheListContentRequest{ Boolean gotOne = false…
```

B- Le deuxième est une méthode quelconque, nous avons choisi un outil qui est une calculatrice standard qui peut être invoquée par un remote peer.

C- Exécution

Tout le code du logiciel a été implémenté en utilisant JXTA-JXSE version 2.5

Le logiciel IISWJ 0.1 se compose de trois packages :

Noyau	Interfaces	Presence
Edge_Nasr_TheCustomer.java	IIswjnterface.java	AdvertisementUtilities.java
RendezVous_Feth_TheInformator.java	IIswjInterfaceEventListener.java	StatusService.java
Information_Service.java	MainInterface.java	StatusClient.java
UserInfoRecord.java	FontIIswj.java	
ModuleIDs.java	MainInterfaceEventListener.java	
ContactHandler.java		
ContactFinder.java		

Figure 4.3- L'Interface principale de IISWJ

1. Initier Feth le peer RendezVous, en effaçant toute configuration existante et en autorisant le multicast.
2. Initier Nasr et attendre sa connexion avec Feth tout en effaçant des configurations existantes, en acceptant le multicast et en permettant à Feth d'être le peer RendezVous.
3. On clique OK pour établir un pipe de connexion avec Feth.
4. On clique OK pour permettre à Nasr d'envoyer sa requête, notre méthode choisie pour implémenter ce Service est une calculatrice en vue d'être utilisée par les peers clients .

5. Le Service d'Information chargé par Feth retourne automatiquement une réponse à Nasr qui à son tour va l'afficher.

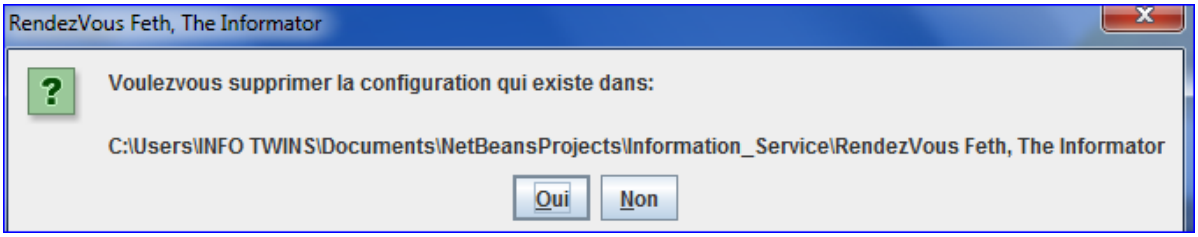


Figure 4.4- La vérification de l'existence d'une ancienne configuration.

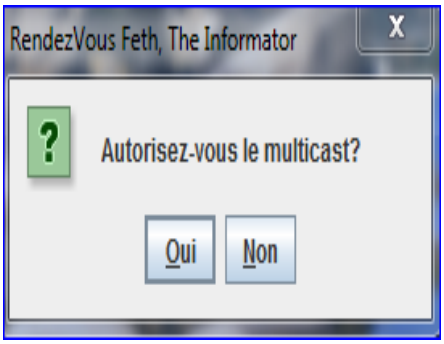


Figure 4.5- Doit-on autoriser le multicast ?

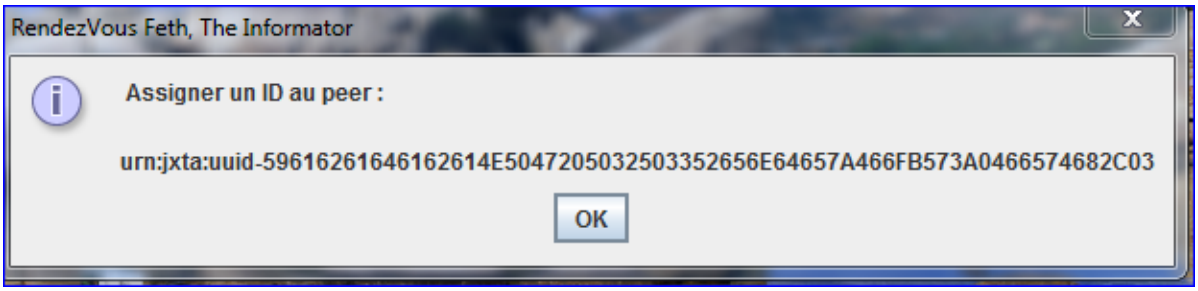
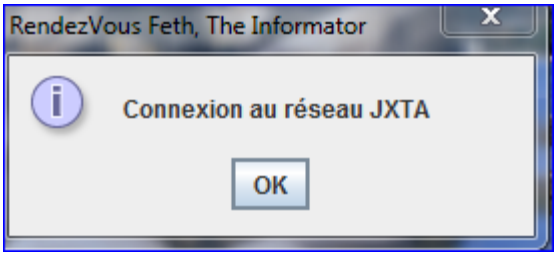


Figure 4.6- Identification du RendezVous Feth par JXTA



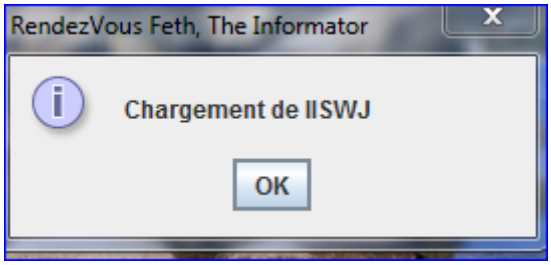


Figure 4.7- Connexion au réseau JXTA et Chargement de IISWJ

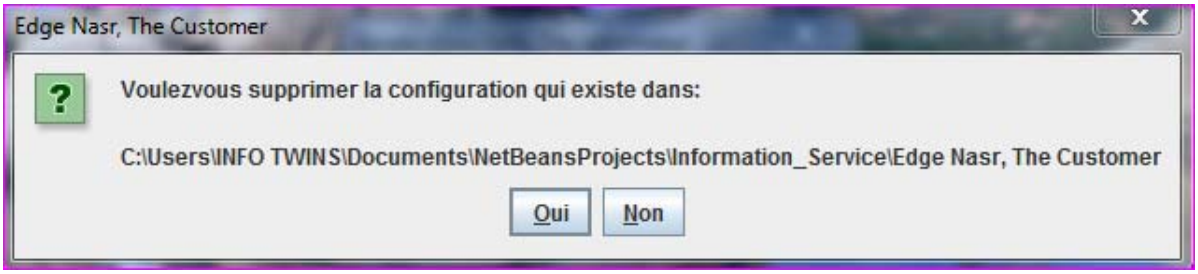


Figure 4.8- Exécuter le Peer Nasr et supprimer une configuration existante

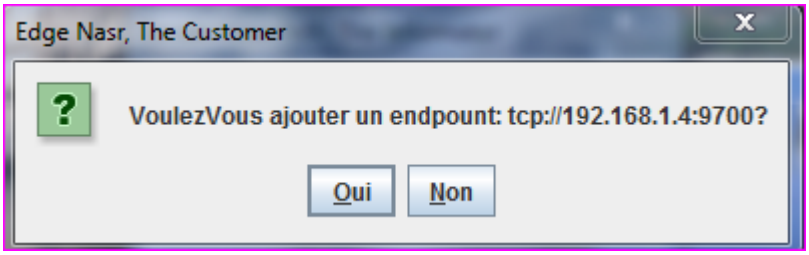


Figure 4.9- Faire communiquer Feth-Nasr avec l'endpoint approprié

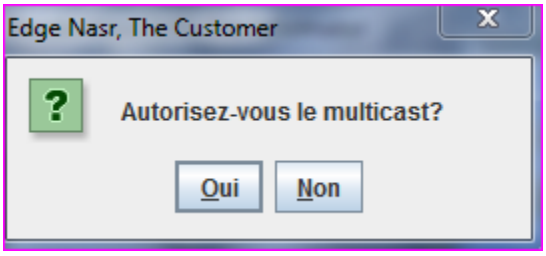


Figure 4.10- Doit-on autoriser le multicast ?

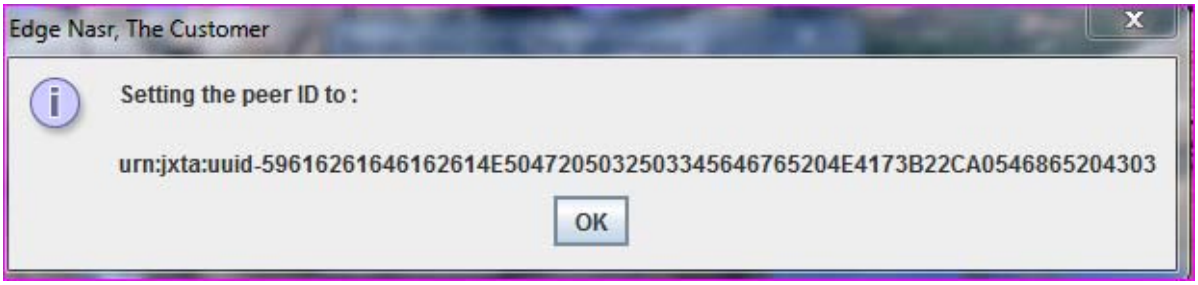


Figure 4.11- Identification de l'edge Nasr par JXTA

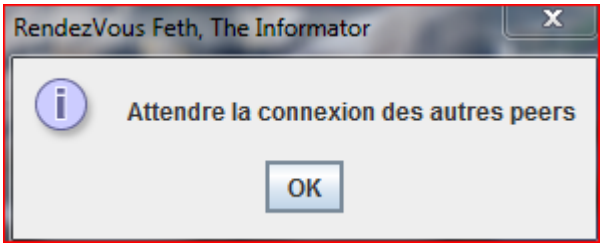


Figure 4.12- Attendre la connexion de Nasr avec Feth

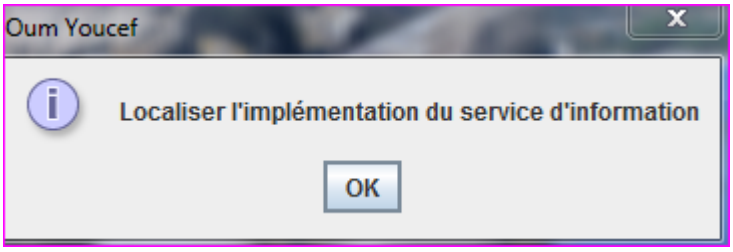


Figure 4.13- Localisation de l'implémentation du Service

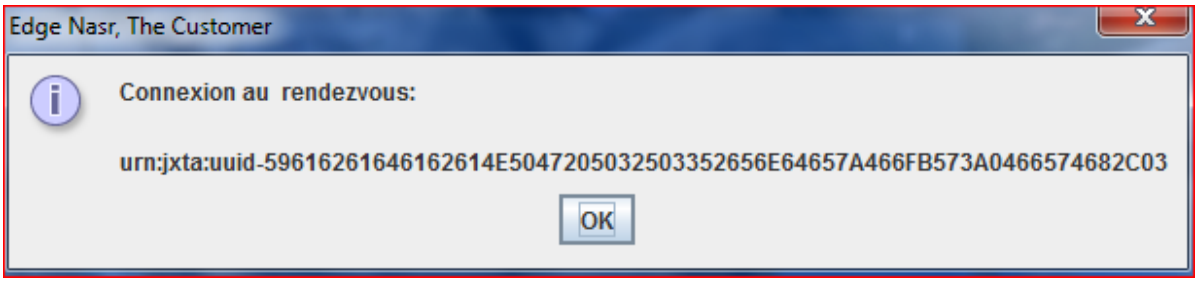


Figure 4.14- Connexion de Nasr avec Feth

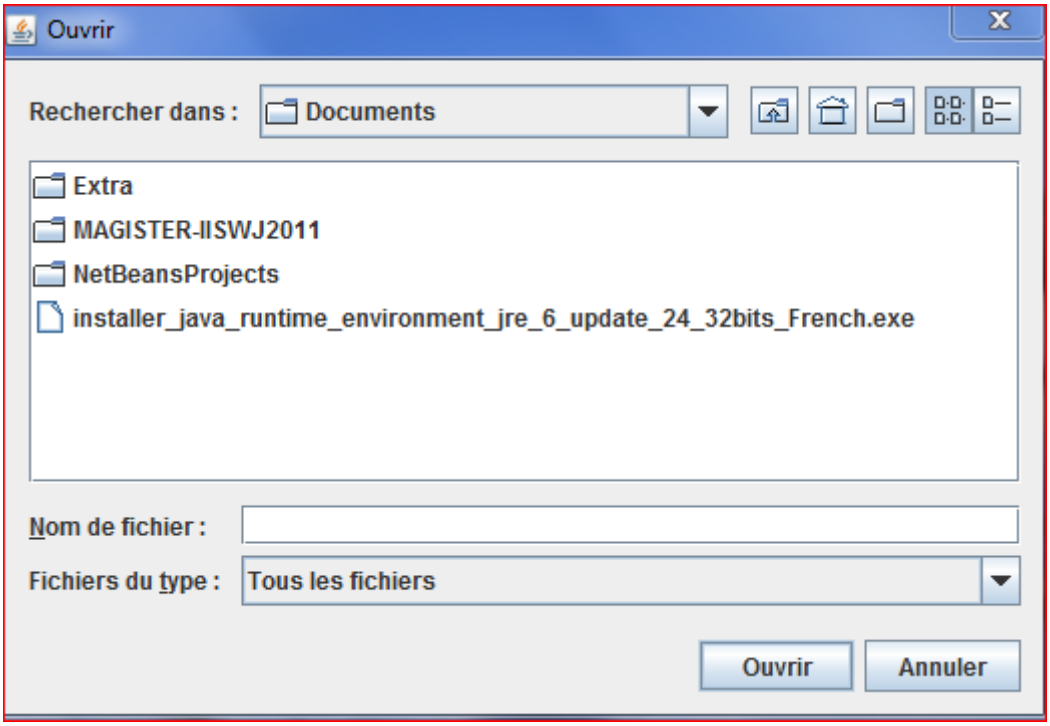


Figure 4.15- Service CMS (Partage de contenu)

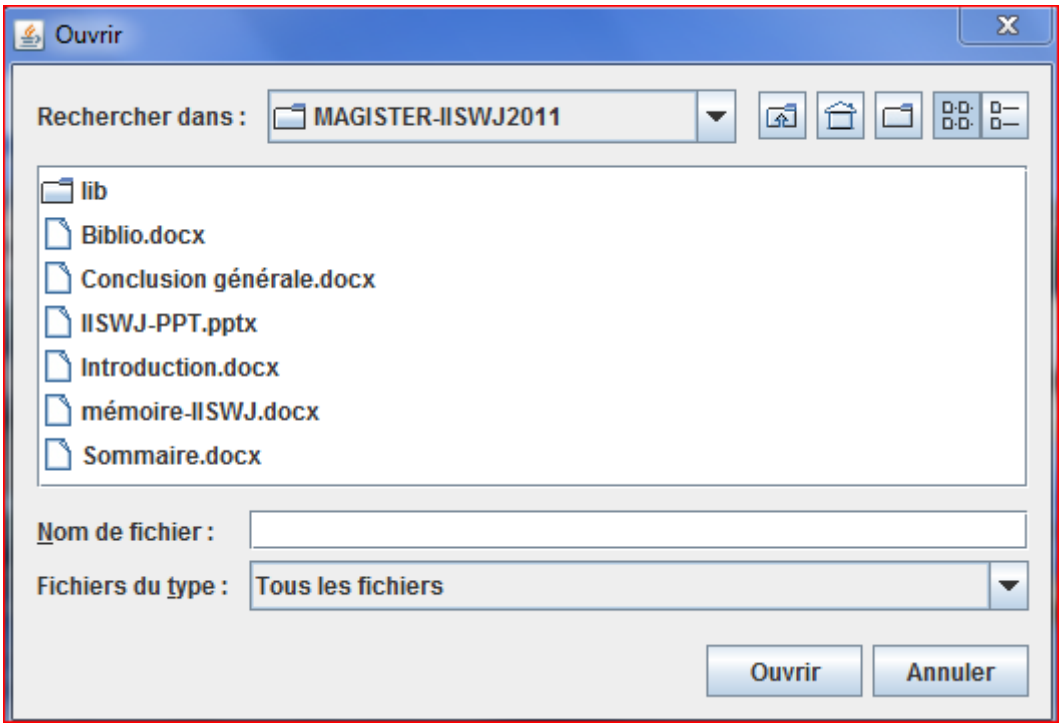


Figure 4.16- Sélection de dossiers

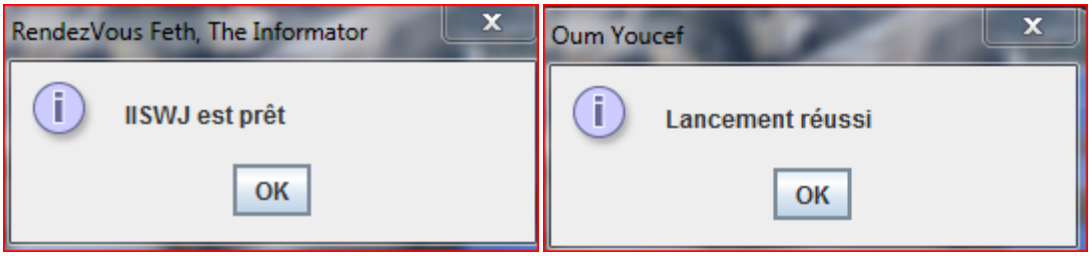


Figure 4.17- IISWJ est prêt à renvoyer la calculatrice

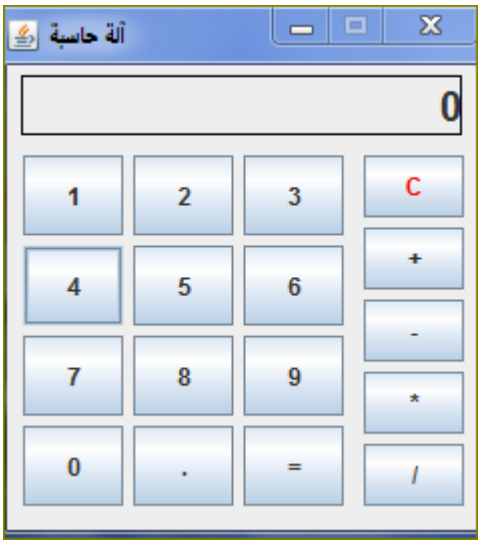


Figure 4.18- la calculatrice demandée par Nasr

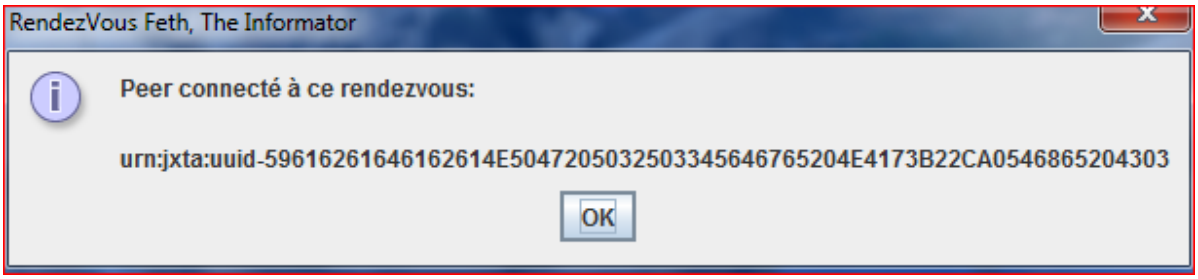


Figure 4.19- BiDipipe entre Feth et Nasr

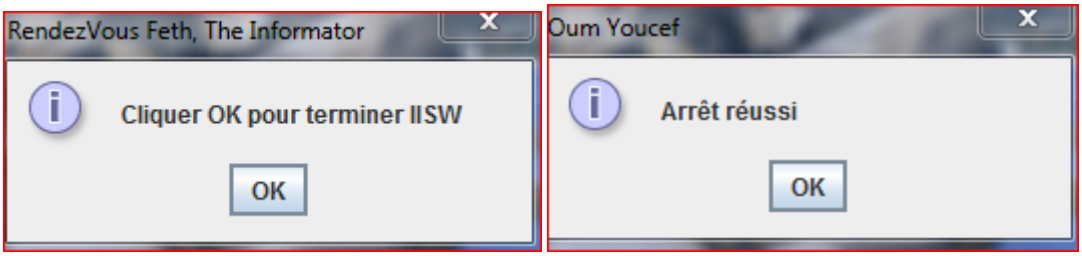


Figure 4.20 Arrêt du Service IISWJ

Figure 4.21 Arrêt du réseau JXTA

4.7 Organisation du Service en vue d'un système de stockage distribué

Nous souhaitons dans le futur concevoir un système de stockage distribué exploitant les résultats obtenus avec notre service IISWJ, mais il reste de résoudre le problème du Peer Relay pour pouvoir contourner les Firewall et les Nats. Le système de stockage envisagé sera composé de quatre services :

- **Service Database_ Peer**
- **Service Buffer_ Peer**
- **Service Collecting_Data Peer**
- **Service Client_ Peer**

Le problème réside dans le Service Collecting _Data Peer qui sera résolu en résolvant le problème du Relay Peer qui accédera alors facilement à internet pour la collecte de données.

4.8 Conclusion

Dans ce dernier chapitre, on a abordé l'implémentation de notre Service « **IISWJ »**, son architecture ainsi que les différentes classes crées pour sa mise en œuvre.

Ce Service semble extensible pour d'autres fonctionnalités telles que l'ajout d'autres Classes ou la redirection vers d'autres sous-Services.

CONCLUSION GENERALE

Avec l'arrivée de l'internet et son rôle actif croissant dans l'économie et la société, l'informatique réseau n'a eu de cesse de trouver des innovations pour exploiter les ressources qu'un réseau de cette ampleur contient.
Aujourd'hui, on parle sérieusement des systèmes PEER TO PEER capables de changer radicalement certaines approches de l'informatique réseau.
Cependant la diversification et l'hétérogénéité de ces systèmes a donné naissance à plusieurs problèmes car chacun d'eux a sa propre méthode de recherche et par conséquent la communication entre eux est pratiquement impossible.
Pour résoudre ces problèmes, les grandes entreprises ainsi que la communauté des chercheurs ont construit une plateforme de programmation réseau qui sera un standard du peer to peer et qui pourra supporter pratiquement tous les services.

Ce travail vise plus particulièrement à définir notre approche pour la conception d'un service pair à pair permettant le partage de données et l'invocation de méthodes particulières telles qu'une calculatrice intégrée au sein de ce service.
Notre approche du concept de ce service (IISWJ : Implementing an Information Service with Peer to peer Technology through Jxta) s'inspire à la fois des systèmes pair à pair (P2P) et de la plateforme JXTA.
Il s'agit pour cette approche hybride d'utiliser JXTA comme un standard pour le P2P et de faire passer à l'échelle la plateforme grâce à une topologie fondée sur un sous-réseau où la configuration de la passerelle est un routeur par lui-même.
Nous avons pu intégrer pour la première fois et parallèlement les protocoles http et tcp grâce à la classe NetworkManager du package Java.net.
C'est une contribution à petite échelle, le passage à grande échelle reste une inconnue mais sera possible dans le futur inchallah grâce aux Relay Peers figurant dans la conception de la plateforme et qui pourront nous conduire vers des systèmes de stockage pair à pair à grande échelle sous JXTA.
La proposition d'architecture utilise pour sa mise en œuvre la bibliothèque P2P JXTA (JXSE : JXTA sous le langage JAVA Version 2.5).
Le projet a aboutit au but fixé initialement mais il reste d'intégrer les résultats obtenus dans une interface graphique et d'ajouter d'autres sous services afin de déployer cette infrastructure au sein de notre université.

PERSPECTIVES

Les industriels sont fortement intéressés et demandeurs des nouvelles fonctionnalités que peut apporter le modèle P2P comme équilibre du trafic et de la charge du service exécuté, la tolérance aux fautes mais aussi le partage des coûts mis en œuvre. Comme ils ne peuvent décidément pas se passer de la gestion sur leurs réseaux afin de garantir la qualité de service attendue.
JXTA est un standard du Peer to Peer, c'est une technologie puissante se transformant d'elle-même, s'adaptant aux besoins des utilisateurs et aux facilités offertes par le langage Java, elle est dynamique attirant beaucoup de développeurs dont nous ressonçons actuellement plus de 30000.

Comme toutes les technologies, le futur de JXTA dépendra de l'évolution de plusieurs facteurs :

Technologiques : simplification de la complexité et universalité, c'est-à-dire faciliter l'interopérabilité et la communication sur différentes plateformes. Jxta peut aider les développeurs et les ingénieurs dans le développement d'applications sans se soucier des différents protocoles , de la connexion internet, elle peut aussi faciliter la communication et l'échange entre des services hétérogènes, c'est donc par excellence la meilleure perspective à envisager.

Sécurité : cette dernière semble préoccuper toutes les compagnies et les utilisateurs d'applications et de logiciels informatiques ce qui semble d'ailleurs une vitalité économique. Cependant Jxta

procure à ses développeurs l'emploi de TLS 1.0 ou d'implémenter leur propre sécurité dans la couche transport offerte par elle.

Educatif : Une technologie bien adoptée doit être bien documentée. Les développeurs et les architectes ont besoin de ces documents pour apprendre cette technologie et comment la déployée.
Comme pour la plupart des technologies open source l'accès aux livres et tutoriaux a été l'une des faiblesses de Jxta. Beaucoup de livres dont j'ai eu l'occasion de lire en entier et dont j'ai passé beaucoup de temps et qui dataient de 2002, leurs codes sont devenus obsolètes de nos jours.

Economique : Les implications économiques et financières d'une nouvelle technologie ont un grand impact sur son adoption de la part du secteur privé, si elle économise de l'argent sur un grand terme elle intéressera les industriels, alors et contrairement à beaucoup de technologies, Jxta ne requit pas beaucoup de ressources comme les serveurs, la bande passante ou des techniciens pour opérer elle repose essentiellement sur les ressources de chaque périphérique pour opérer sur le réseau ; sa conception l'a rendu plus économique que la classique technologie Client/Serveur.
C'est une technologie compétitive sur le marché grâce à son coût réduit.

Vers de nouvelles applications : Un ensemble d'applications basées sur Jxta et le pair à pair n'ont pas encore atteint leur but pourtant le P2P a déjà démontré quelques applications surprenantes telles que SKYPE (téléphonie sur Internet) donc au tour de Jxta qui dans son intention est universelle.
Dans notre Service IISWJ et grâce au JXSE , le but a été atteint mais nous souhaitons un service à grande échelle avec cette technologie qui a toutes les chances d'être le leader du P2P.

Bibliographie

[ALE 06] Dijoux Alexandre Emma Samuel, Peer-To-Peer, Université Claude Bernard LYON 1, 2006, Disponible sur http://master-info.univ-lyon1.fr/M2SIR/2006/peer2peer.pdf, la date de consultation novembre 2008.

[ANT 02] Gabriel Antoniu, Luc Bougé, Thierry Priol, Partage de mémoire à très grande échelle sur des réseaux pair-à-pair, Editeur INRIA - Domaine de Voluceau - (France), **N°** 4410, Mars 2002, Disponible sur ftp://ftp.inria.fr/INRIA/publication/dienst/RR-4410.pdf, la date de consultation novembre 2008.

[BRO 02] Daniel Brookshier, Darren Govoni, Navaneeth Krishnan, Juan Carlos Soto, JXTA: Java™ P2P Programming, Sams Publishing, 1st edition (April 1, 2002), Disponible sur http://gigapedia.com/items/19399/jxta--java-p2p-programming, la date de consultation novembre 2008.

[CMS 02] Microsystems, s:project CMS, the JXTA Content Manager Service.

[DAL 03] Quentin Dallons, JXTA : Un Framework Peer - to- Peer Open Source, Institut d'informatique FUNDP Namur-Belgique, 2003, Disponible sur http://www.info.fundp.ac.be/~ven/CISma/FILES/2003-quentin-DALLONS.pdf, la date de consultation novembre 2008.

[DOY 05] Doyen.G, Nataf.E, Festor.O, Une architecture hiérarchique pour une gestion distribuée des réseaux et services pair à pair, Benzekri .A, Sibilla.M, Eds, 6ième colloque Francophone sur la Gestion des Réseaux et Services -GRES 05, 2005.

[FEL 03] Gabrielle Feltin, Guillaume Doyen, Olivier Festor, Les protocoles Peer-to-Peer leur utilisation et leur détection, LORIA - Campus Scientifique, 7 Octobre 2003, Disponible sur http://2003.jres.org/actes/paper.70.pdf, la date de consultation novembre 2008.

[FES 03] Festor. O, Ben Youssef. N, Standards pour la gestion des réseaux et des services, chapitre Le modèle CIM, p. 113-157, Hermes, 2003.

[GRA 02] Joseph D. Gradecki, Joe Gradecki, Mastering JXTA Building Java Peer-to-Peer Applications, Wiley, 1 st edition (Aug 19 2002), Disponible sur http://gigapedia.com/items/ 9935/ mastering-jxta--building-java-peer-to-peer-applications, la date de consultation novembre 2008.

[JXT 02] jxta web site, http://www.jxta.org.

[KAZ 02] Kazaa, http://www.kazaa.com.

[KAN 01] Kan G., Gnutella, GoneSilent.com, _ Peer-to-peer : Harnessing the Power of Disruptive Technologies _, chapitre Gnutella, p. 94-122, O'Reilly & Associates, Inc., 2001.

[LIJ 02] Sing li, jxta peer to peer computing with java, wrox, janvier 2002.

[MAT 05] Vincent Matossian, [ECE 451/566] - Introduction to Parallel and Distributed Computing, Peer-to-Peer Networks + an Introduction to JXTA, December 12th 2005, Disponible sur http://www.ece.rutgers.edu/~parashar/Classes/ece451-566/slides/lecture-p2p.pdf, la date de consultation novembre 2008.

[MEK 09] R.MEKKI, F.Mourlin, Interoperable mobile agent software observation, Editeur, International Review on Computers and Software, Vol 3, N° 3 :276-282, mai 2008.

[MEK 09] R.MEKKI, R.Fezza, A simple chat Application based on JXTA, Editeur, Journal of Applied Sciences, **N°** 9(21) :3912-3916, octobre 2009.

[MOH 04] Mohanti.R, Instrumentation of the Jxta peer-to-peer framework, Master's thesis, Indian Institute of Technology Kharagpur, Novembre 2004.

[NAP 02] Napster: http: //www.napster.com

[OAK 02] Oaks S., Traversat B., Gong L., Jxta in a nutshell, O'Reilly, 2002.

[SHI 01] Shirky. C, Peer-to-peer : Harnessing the Power of Disruptive Techno-logies , chapitre Listening to Napster, p. 21-37, O'Reilly & Associates, Inc,2001.

[STO 01] Stoica. I, Morris .R, Karger. D, Kaashoek .M. F, Balakrishnan. H, Chord : A scalable peer-to-peer lookup service for internet applications, Proceedings of the ACM Conference on Applications, Technologies, Architectures and Protocols for Computer Communication - SIGCOMM'01, ACM Press, 2001, p. 149-160.

[SUN 05] Sun Microsystems, JXTA v2.3.x : Java Programmer Guide , April 2005.

[SUN 07] Sun Microsystems, JXTA Java™Standard Edition v2.5: Programmers GuideSeptember 10th, 2007, Disponible sur https://jxta-guide.dev.java.net/source/browse/ *checkout*/jxta-guide/trunk/src/guide_v2.5/ JXSE_ProgGuide_v2.5.pdf, la date de consultation novembre 2008.

[WIL 02] Brendon J. Wilson, JXTA, by New Riders Publishing, 1st edition June, 2002, Disponible sur www.brendonwilson.com/projects/jxta, la date de consultation novembre 2008.

[WRI 03] William Wright, Dana Moore, Jabber Developer's Handbook, Sams Publishing, 1 st edition (Jul 8 2003), Disponible sur http://gigapedia.com/items/131980/jabber-developer--039-s-handbook--developer--039-s-library-, la date de consultation novembre 2008.

IMPLEMENTING AN INFORMATION SERVICE WITH PEER TO PEER THROUGH JXTA TECHNOLOGY "IISWJ"

Abstract

JXTA is an open-source effort; it reflects the operations by which peers establish temporary associations to form a P2P network. Conceptually JXTA is made of three logical layers: Platform, Services and Applications.
In this article we are going to focus in the Services layer, therefore we are going to introduce the Peer-to-Peer architecture, the JXTA concepts, protocols and structure so finally we can be able to implement our Service "IISWJ''.
The purpose of (IISWJ) in first is to enable a peer to share data such as text documents with remote peers, to optimize the use of system resources, the CMS requires that the shared content remain on the file system stored on the local peer and to maintain consistency with the specification, the CMS relies on advertisements to provide main information and relies on JXTA pipes to transfer the content.
Secondly, with IISWJ a Customer Peer should be able to ask the use of the calculate built in this Service. So in this work, we showed you that a costumer peer has the choice to use the CMS into the JXTA system or invoque the calculator available in it.

Keywords: P2P, JXTA, Service, Information Service, Module, IISWJ.

IMPLEMENTATION D'UN SERVICE D'INFORMATION AVEC LA TECHNOLOGIE PEER TO PEER SOUS LA PLATE-FORME JXTA - (IISWJ)-

Résumé

JXTA est un effort open-source qui reflète les opérations à travers lesquelles des pairs peuvent établir des associations temporaires pour former un réseau P2P.
Conceptuellement, JXTA est constitué de trios couches logiques: Platform, Services et Applications.
Dans ce mémoire, nous focalisons sur la couche Services, nous avons introduit les notions de base du Peer-to-Peer, les concepts de la plate-forme JXTA, ses protocoles ce qui nous a permis d'implémenter notre Service (IISWJ).

Le but du Service (IISWJ) est de permettre premièrement le partage de contenu, donc un pair peut partager les données comme des fichiers textes avec des pairs voisins. Afin d'optimiser l'emploi des ressources systèmes, le CMS requit uniquement les fichiers stockés dans le système, la consistance du système est maintenue grâce aux spécifications du Module, le CMS conçoit des annonces (Advertisements) pour fournir des informations et repose sur le concept de pipes JXTA pour le transfert du contenu.
Secondement, IISWJ permet à un Peer Client d'invoquer l'utilisation de la calculatrice se trouvant au sein du Service.
Dans ce modeste travail, on a montré que le Client a le choix entre l'utilisation du CMS ou l'invocation d'une calculatrice standard.

Mots-clés: P2P, JXTA, Service, Service d'information, Module, IISWJ.