République Algérienne Démocratique et Populaire

Ministère de l'enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la technologie d'Oran

USTO

-Mohamed Boudiaf-

Faculté des Sciences

Département d'informatique

Mémoire pour l'obtention de magister en informatique

Option : Reconnaissance des formes et intelligence artificielle

Présenté par : BOUHOUYA Siham

Thème

Implémentation d'un agent mobile dans un environnement pair à pair

Soutenu publiquement le 26 Mai 2011 devant le jury :

Président	**Mr. A. BENYETTOU**	**Professeur**	**USTO -MB-**
Rapporteur	**Mme. R. MEKKI**	**Maitre de conférences**	**USTO -MB-**
Examinateurs	**Mme. F. BENDELLA**	**Maitre de conférences**	**USTO -MB-**
	Mme. H. FIZAZI	**Maitre de conférences**	**USTO -MB-**

Remerciement

Merci Dieu le tout puissant pour m'avoir donné la foi et le courage pour accomplir ce travail.

Je tiens à remercier particulièrement Dr. MEKKI Rachida pour la confiance qu'elle ma accordé, pour son aide et ses précieux conseils apportées durant la réalisation de ce modeste travail.

Mes remerciements vont également à Dr. BENYETTOU Abdelkader pour avoir accepter de présider ce jury.

Je tiens à remercier aussi Dr. FIZAZI Hadria et Dr. BENDELLA Fatima de m'avoir fait l'honneur d'accepter d'examiner ce travail.

Dédicace

Je dédie ce modeste travail

A ma très chère mère,

A mon père,

A mon époux,

A mes chères sœurs,

A mes petites nièces,

A mes neveux.

TABLE DES MATIERES

❖ **Chapitre III : La plateforme JXTA**

Liste des figures

Liste des tableaux

Résumé

Le pair à pair est la naissance d'un nouveau paradigme dans l'informatique distribué, bouleversant le modèle traditionnel client/serveur. La technologie des agents mobiles est un sujet de recherche depuis plusieurs années, datant du milieu des années 1990 [Che 02], mais l'utilisation des agents mobiles reste limitée. Le pair à pair comme une technologie émergente, exerce une énorme influence sur les scénarios de différentes applications. Pendant ce temps, les agents, en aidant les applications dans le paradigme traditionnel avec des moyens intelligents, doivent pouvoir s'intégrer dans cette nouvelle tendance. Dans ce mémoire nous discutons les agents mobiles dans une application de commerce électronique basée sur une plateforme pair à pair. Notre recherche montre que les agents mobiles avec leurs mobilités du code et le comportement asynchrone, implémentés sur des plateformes P2P comme JXTA, forment une approche très puissante dans les systèmes distribués, permettant à plusieurs applications de fonctionner plus efficacement. L'objectif global de ce mémoire est de montrer qu'une combinaison réussie entre les deux, agents mobiles et pair à pair, élimine beaucoup d'obstacles et apporte de grands avantages. Cette combinaison, en particulier sur la plateforme JXTA, est une collaboration idéale, profitant au maximum de ces deux technologies.

Mots clés : Agent**s** mobiles, Pair à pair, JXTA.

Introduction générale

Contexte de travail

Avec le développement constant d'Internet et la croissance exponentielle du réseau d'information avec World-Wide Web, la qualité d'accès à l'information perçue par l'utilisateur est devenue un réel problème. Le vieux modèle client/serveur a montré son déficit, vu les problèmes rencontrés par ces systèmes centralisés notamment la surcharge des serveurs, le passage à l'échelle et la panne des serveurs. Les applications créées sur ce modèle ne sont clairement plus adaptées aux volumes et aux contenus des données actuelles. Elles constituent des goulots d'étranglement évidents liés au nombre de connexions et au débit limité qu'elles proposent. En comparaison, les capacités des machines connectées à l'Internet forment une quantité de ressources disponibles qui est à la base des réseaux pair à pair. Les nœuds physiques qui auparavant ne faisaient qu'utiliser le service, sont désormais utilisés pour fournir le service.

Le réseau pair à pair représente l'une des tendances dans l'informatique distribué, un système qui ne dépendent plus de l'infrastructure centralisé d'une tierce partie. Chaque nœud (pair) peut agir simultanément en tant que serveur et client d'où l'appellation 'servent', les nœuds connectés sont capables de communiquer directement ente eux et partager leurs ressources sans avoir besoin d'un point central. Autrement dit le pair à pair offre un moyen de décentralisation des services et des ressources autrefois centralisés sur un ou plusieurs serveurs.

Les systèmes pair à pair sont aujourd'hui largement répandus, ils ont prouvé leur efficacité dans diverse domaines. Leur utilisation va du partage du temps CPU [Seti@home,1997], au partage de données [Cla 00, eMule, 2002], en passant par les communications [Bas 04], et le stockage de données [Rhe 03]. Et ça se justifie par les caractéristiques de ce type de réseaux et les avantages offerts par celui-ci.

Les réseaux pair à pair constituent un support auto –organisé, flexible, tolérant aux pannes et permettant le passage à l'échelle avec moindre coût. Notre projet est basé dans et autour de l'environnement de cette technologie.

Les agents mobiles sont des composants, des entités ou tout simplement du code applicatif ayant une propriété mobile leur permettant de se déplacer sur un réseau hétérogène d'un dispositif à un autre sous leur propre contrôle. Afin de fournir un service précis à leurs propriétaires. Lors de la migration l'agent transporte son code et son état d'exécution à la

prochaine destination. Les agents mobiles sont dotés aussi des capacités d'adaptabilité, d'autonomie, ils sont capables de se multiplier ou se cloner, ils peuvent également créer des nouveaux agents.

Les agents mobiles offrent plusieurs avantages améliorant la performance de plusieurs applications distribuées. L'amélioration peut être ressentie dans la réduction du trafic réseau, la répartition dynamique de charge, la commodité par rapport aux programmeurs ou simplement dans l'habilité de continuer l'interaction avec un utilisateur durant une déconnexion du réseau. Cette technologie est une approche très puissante aide les applications réseau avec des moyens intelligents. Une partie des applications ne nécessiteront pas cette technologie, mais certaines se révèleront plus efficaces en l'utilisant. Des applications comme le calcul parallèle, la collecte des informations avec filtrage, le commerce électronique, etc.

L'utilisation des agents mobiles présente plusieurs avantages. Néanmoins, ils posent quelques problèmes, dont les plus importants sont ceux relatifs à l'interopérabilité, la sécurité et la tolérance aux pannes des systèmes d'agents mobiles .

Le pair à pair et les agents mobiles représentent des objets de recherche actifs dans plusieurs projets, mais la combinaison entre eux vient d'être récemment prise dans le cadre de la recherche. Avec ces technologies ensemble, beaucoup d'applications utilisant le modèle traditionnel client / serveur pourrait être changées pour fonctionner beaucoup plus efficacement.

Une telle combinaison nécessite une plate forme qui profite au maximum des avantages de ces deux technologies. La technologie JXTA est un ensemble de six protocoles pair à pair permettent de gérer un certain nombre de fonctionnalités communes, comme la gestion des pairs, la découverte de ressources, l'invocation de services sur le réseau, la communication inter-pair, la sécurité, etc. Ces protocoles peuvent servir de briques de base pour la mise en œuvre de services pair-à-pair. Permettant à n'importe quel appareil connecté sur le réseau, allant des téléphones cellulaires et assistants numériques sans fil pour PC et serveurs, de communiquer et de collaborer d'une manière P2P

Problématique

Depuis la révolution du .com, et le succès instantané des applications pair à pair telles que Napster, Morpheus, LimeWire et Sun JXTA. Il est clair que dans de nombreux cas, le client est mieux servi avec les systèmes pair à pair d'un système client/serveur. La technologie des agents mobiles est un sujet de recherche depuis plusieurs années, datant du milieu des années 1990 [Che 02], mais l'utilisation des agents mobiles reste limitée. Ceci est dû à une adéquation avec le modèle client/serveur.

Le pair à pair comme une technologie émergente, exerce une énorme influence sur les scénarios de différentes applications. Pendant ce temps, les agents, en aidant les applications dans le paradigme traditionnel avec des moyens intelligents, doivent pouvoir s'intégrer dans cette nouvelle tendance.

Objectif

Dans le cadre de ce mémoire nous discutons les agents mobiles dans une application de commerce électronique basée sur une plateforme pair à pair. Notre recherche montre que les agents mobiles avec leurs mobilité du code et le comportement asynchrone, implémentés sur des plateformes P2P comme JXTA, forment une approche très puissante dans les systèmes distribués, permettant à plusieurs applications de fonctionner plus efficacement. L'objectif global de notre travail est de montrer qu'une combinaison réussie entre les deux, agents mobiles et pair à pair, élimine beaucoup d'obstacles et apporte de grands avantages en présentant une résilience qui manque dans l'adaptation des agents mobiles. Cette combinaison, en particulier sur la plateforme JXTA, est une collaboration idéale, profitant au maximum de ces deux technologies.

Organisation du mémoire

Ce document est structuré de la manière suivante, un premier chapitre présente une vue d'ensemble du paradigme pair à pair, nous étudierons dans ce chapitre la notion de pair à pair, ses domaines d'application, les différentes architectures, la notion de vision des pairs sur le réseau et les avantages de ce type de réseau.

Le second chapitre porte sur la notion d'agents, nous abordons dans ce chapitre les définitions essentielles de cette thématique. Nous insistons sur la notion d'agent mobile qui est au cœur de notre travail.

Vu qu'on a utilisé la plateforme JXTA, il s'avère indispensable de faire une étude générale sur ce projet, c'est quoi JXTA ?, pourquoi le JXTA ?, voir également ses différents protocoles, composants et concepts de base dans chapitre trois.

Le chapitre cinq articule autour des technologies liée à notre problématique, dans ce chapitre on s'est intéressé beaucoup plus aux travaux antérieurs et récents concernant les agents mobiles.

Un dernier chapitre, consacré à l'étude conceptuelle et l'implémentation de notre système. Dans cette partie nous verrons une description détaillé du système.

Enfin, nous terminons par une conclusion générale, qui résume l'apport essentiel de notre travail et présentons nos perspectives de recherche.

Chapitre I
Les réseaux pair à pair

Introduction

L'architecture client/serveur était l'architecture la plus adoptée par les systèmes informatiques multi tiers. Une telle architecture est basée sur un point central du réseau appelé le serveur. Ce dernier s'occupe de répondre aux différentes demandes de ressources en provenance des autres points du réseau (les clients).

Mais à cause aux limites rencontrées dans le paradigme client/serveur, plus précisément à l'extrémité serveur (la pane de serveur paralyse le réseau complet), le besoin d'une architecture décentralisée est indispensable. Une architecture sans un point d'administration et contrôle unique.

Le pair à pair, en anglais peer to peer est un modèle de réseau de la catégorie distribuée décentralisée ou les nœuds jouent simultanément le rôle de client et le rôle de serveur. Chaque nœud du réseau fournit des services et également se bénéficier des offres des autres nœuds, par conséquence la charge des taches de réseau est répartie sur tout les nœuds. Autrement dit le peer to peer offre un moyen de décentraliser des services et des ressources autrefois centralisés sur un ou plusieurs serveurs. Par exemple, le calcul distribué, le partage de fichiers, messagerie instantanée et la collaboration sont les fruits du développement d'applications déployées sur des réseaux pair à pair visant à délocaliser les services.

Ce chapitre donne une vue globale sur les réseaux Peer-To-Peer : des définitions, les principes de fonctionnement, les différents architectures, domaines d'application et les apports de ce type de réseaux.

1. Définitions

Un passage rapide par la littérature révèle un grand nombre de définitions du «pair à pair». La définition la plus stricte est celle du pur pair à pair : « *pair à pair un système totalement distribué dans lequel tous les nœuds sont équivalents en terme de fonctionnalités et de tâches à résoudre* » [And 04]. Cette définition ne concerne pas les systèmes qui utilisent la notion de super nœuds.

On a aussi cette définition qui est largement acceptée: "*A distributed network architecture may be called a Peer-to-Peer (P-to-P, P2P) network, if the participants share a part of their own hardware resources (processing power, storage capacity, network link capacity, printers). These shared resources are necessary to provide the Service and content offered by the network (e.g. file sharing or shared workspaces for collaboration). They are*

accessible by other peers directly, without passing intermediary entities. The participants of such a network are thus resource (Service and content) providers as well as resource (Service and content) requestors (Servent-concept) "[Sch 01]

[kar 02] a définit le Peer-to-Peer: "*peer-to-peer refers to a class of systems and applications that employ distributed resources to perform a critical function in a decentralized manner*".

Les systèmes et les applications classés comme étant peer to peer ou non selon la façon dont ils sont aperçus de l'extérieur, et non pas par leur fonctionnement interne. Le peer to peer modélise donc l'échange direct des ressources et services entre ordinateurs. Chaque élément est client et serveur à la fois. La relation client/serveur n'est plus associée aux nœuds, mais aux transactions : chaque nœud peut être client dans une transaction et serveur dans une autre [Fel 02].

Chaque élément dans un réseau Peer-to-Peer est appelé nœud. Tous les nœuds ont un rôle équivalent et peuvent être simultanément le client et le serveur d'où l'appellation 'servent ' (concaténation de serveur et client). Un nœud peut partager ses ressources (données, mémoire, processeur,…) avec les autre nœuds du réseau, ce qui fait le point fort de ce genre de système.

2. Caractéristiques des réseaux pair à pair

Les réseaux P2P[1] sont distingués par leurs caractéristiques des autres architectures réseau. Parmi ces caractéristiques, on mentionne :

2.1La décentralisation

La décentralisation peut être sélectionnée autan que la caractéristique principal des réseaux P2P. Le plus souvent, le P2P est un système, complètement décentralisé, sauf pour le cas de l'architecture P2P centralisée dans la quelle seules les ressources sont décentralisées. La décentralisation des réseaux P2P leur donne les caractéristiques suivantes :

- Equilibrage de la charge.
- Le passage à l'échelle.
- La tolérance aux pannes.
- La réduction des couts.

[1] P2P : Peer to Peer

2.2Auto –organisation

Les réseaux P2P sont ad hoc, donc ils indispensable de définir un mécanisme d'auto – organisation qui gère l'arrivée et le départ des pairs.

2.3Interopérabilité

L'interopérabilité dans les réseaux P2P est garantie par l'utilisation des protocoles dédiés à des technologies multiplateformes comme JAVA.

Ces caractéristiques permettent d'apparaitre l'utilisation des réseaux pair à pair dans plusieurs domaines.

3. Les domaines d'application des réseaux pair à pair

On peut distinguer les plus principaux domaines d'applications :

3.1 Travail collaboratif

Les systèmes de collaboration consiste à relier plusieurs personnes géographiquement dispersées afin d'éditer un objet partagé ou accomplir une tache. Un réseau P2P permet la coopération entre eux sans utiliser de serveur central. Ce type d'application doit être tolérant aux fautes et respecter des contraintes de temps réal car la collaboration se fait généralement par des utilisateurs sensibles au moindre retard de réponse. Quelques exemples peuvent être cités : Jabber, Skype, etc.

3.2 Le calcul distribué

Le P2P permet de mettre en commun de nombreux ordinateurs disposant chacun de ressources limitées. Bien au contraire, l'agrégation de ces ressources fournit théoriquement une performance considérable. Confier un calcul d'une grande complexité à plusieurs nœuds permet de réduire considérablement le temps d'exécution nécessaire. Un exemple qui revient souvent est *SETI@Home* (Search for Extraterrestrial Intelligence).

3.3 Partage de fichiers

C'est la classe d'applications ayant fait connaitre le pair à pair au grand public. En effet ces programmes ont la possibilité d'avoir accès à une concentration très importante d'informations stockées sur des pairs à travers le monde. Ces données peuvent être téléchargées à partir de n'importe quel pair connecté au réseau. Les fichiers musicaux, films,

et livres divers qu'on y trouve peuvent être téléchargés librement. Dans cette catégorie nous pouvons citer : Gnutella, Kazaa, eMule, iMesh, LimeWire, Shareaza, Bearshare, Bittorent, ANts ou encore FreeNet. [Wai 07]

3.4 Moteurs de recherche

Le nombre immense de page web générés dynamiquement ainsi que les droits d'accès aux bases de données distantes s'inscrivent dans les problèmes rencontrés par les moteurs de recherche traditionnels. Le mécanisme de recherche dans les systèmes P2P, basé sur l'inondation, permet de surmonter ce type de limite. InfraSearch [2] représente l'exemple le plus courant de moteur de recherche P2P.

3.5Les plateformes de développement

Ce sont des environnements de développement d'application pair to pair qui, tout comme un système P2P classique, permette le partage de fichiers, la sécurité, la recherche de documents et la communication entre les pairs. Le principal but des plateformes est de rendre les applications de pair à pair indépendantes des systèmes exploitation afin de toucher plus grand public. Un deuxième but non négligeable est de faciliter le développement aux programmeurs. [Mil 02] On peut citer comme exemple la plateforme JXTA.

3.6 Le stockage distribué

L'espace de stockage fournit par chaque nœud du réseau peut être un support de stockage d'un grand volume. Un tel support peut être exploité avec le moindre coût.

[2] InfraSearch : un moteur de recherche distribué basé sur Gnutella.

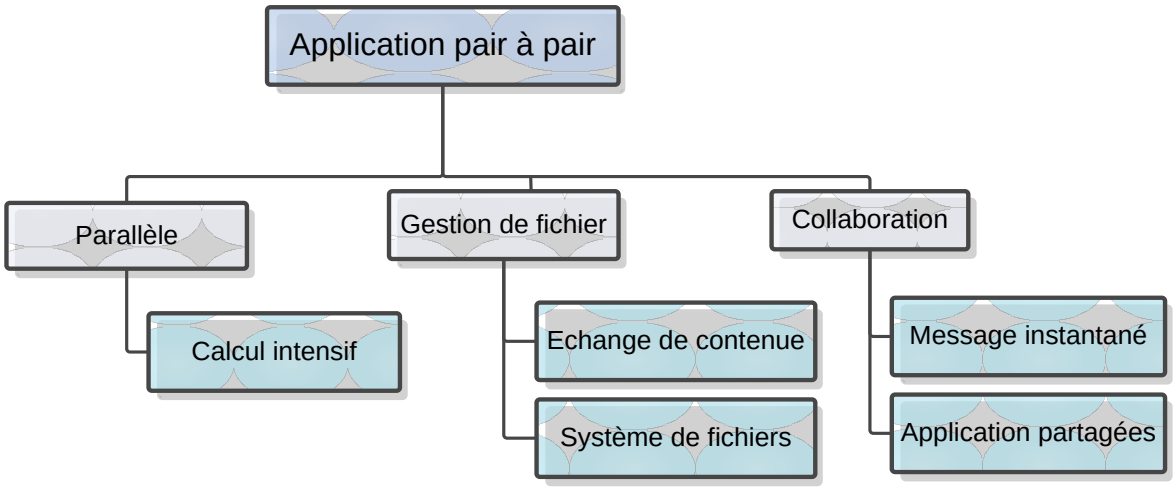


Figure 1.1. Domaines d'applications des réseaux P2P

4. Les architectures des systèmes informatiques

Les systèmes informatiques peuvent être classés en deux grandes catégories, les systèmes centralisés qui reposent sur les mainframes et les systèmes distribués. Dans les systèmes centralisé tout est localisé sur la même machine et accessible de l'extérieur si cela est nécessaire.

Les systèmes distribués sont définis comme étant un ensemble de machines autonomes indépendantes interconnectées à travers un réseau. Ces derniers peuvent être divisés à leur tour en deux modèles : le modèle client/serveur (modèle hiérarchique) et le modèle pair à pair qui peut être pur ou hybride. La figure suivante représente une simple classification des systèmes informatiques.

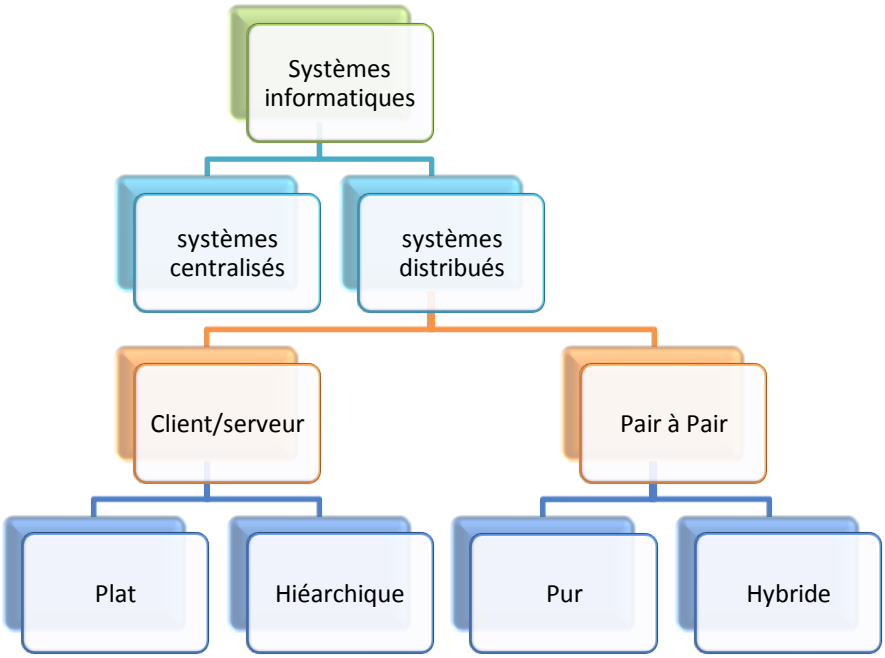


Figure 1.2. Classification des systèmes informatiques

Dans ce qui suit on s'intéressera aux systèmes pair à pair en donnant des exemples d'applications pour chaque type d'architecture.

5. Les types d'architectures pair à pair

5.1 Architecture purement décentralisée

Une architecture est dite purement décentralisée si elle fait partie des réseaux P2P (définit dans la section précédente) et si l'on enlève un nœud choisi aléatoirement le système continue à fonctionner correctement. [Sch 01]

Dans ce type d'architecture, l'ensemble des nœuds sont égaux et jouent le même rôle. Chaque pair gère donc les recherches et le partage, sans passer par un serveur central ou des super-peer.

Pour acquérir une ressource, le pair transmet une requête à ses voisins qui font de même et ainsi de suite (inondation).Pour empêcher la génération d'un nombre exponentiel de requêtes, la propagation est contrôlée par un paramètre qui indique la durée de vie de la requête : "Time To Live" (TTL), ce dernier est un entier qui est décrémenté à chaque fois que la requête arrive sur un pair. une fois la ressource est trouvée , avant l'expiration du délai (TTL=0), une connexion directe est établie entre les deux pairs.

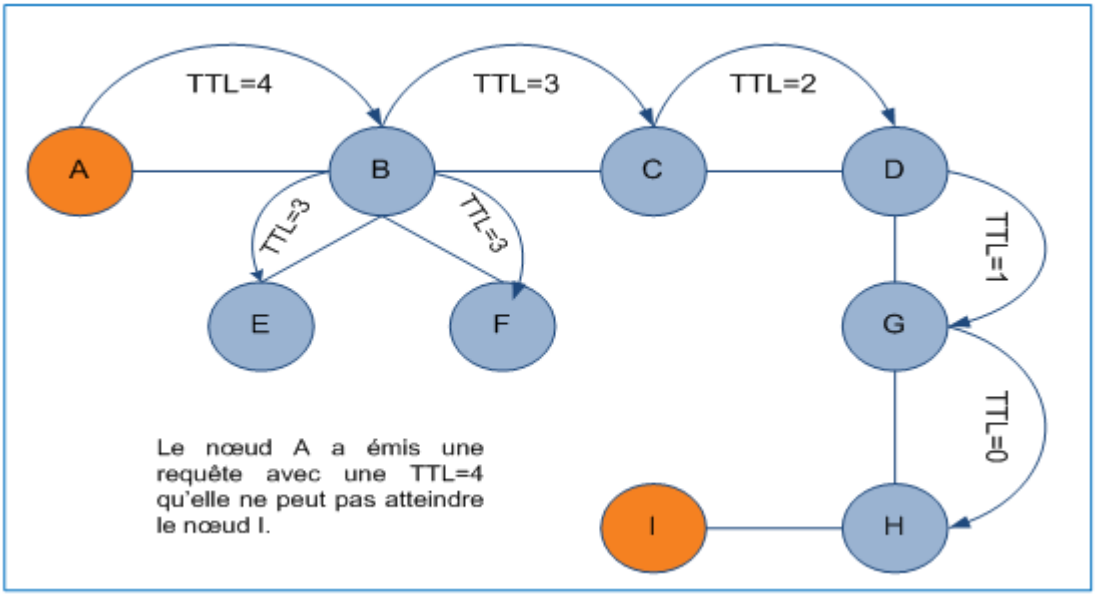


Figure 1.3 : le principe de TTL

Exemple d'architecture purement décentralisée : Gnutella, Freenet.

Gnutella :

Est un Protocole de partage de fichiers, version 0.4 développé en mars 2000, par Justin Frankel et Tom Pepper. Gnutella se base sur un réseau complètement décentralisé, modèle pur.

Chaque client joue à la fois le rôle de client et de serveur « servent ». Communication par message. Fonctionne au dessus de TCP/IP.

Connexion d'un pair Gnutella :

1. Un nouveau pair s'identifie sur le réseau en envoyant un '*ping*' Gnutella.
2. Le Ping est propagé à ses voisins, qui envoient eux-mêmes des Pings.
3. Tous les voisins répondent par un Pong.

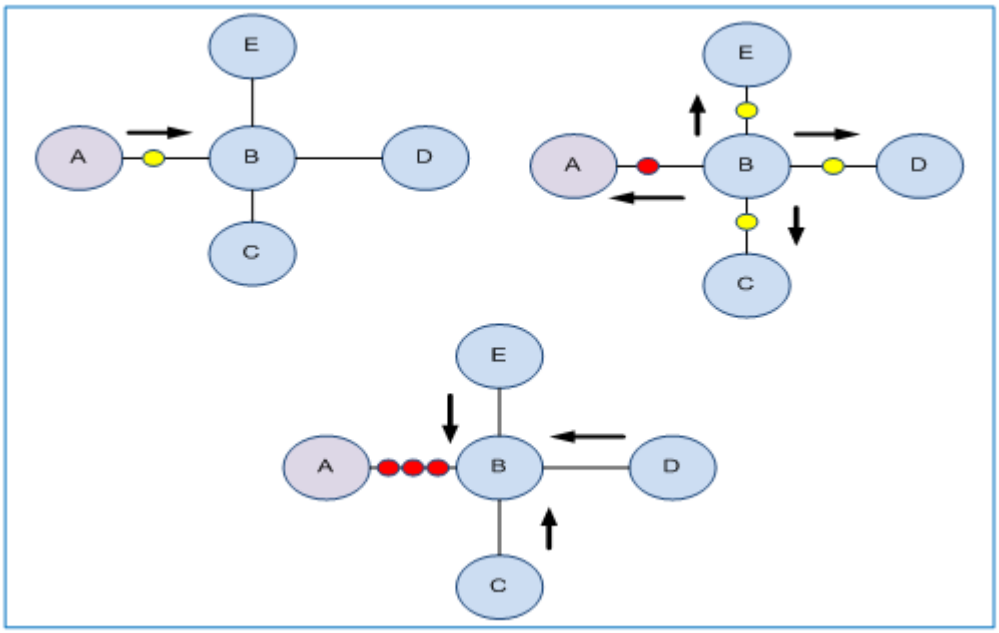


Figure 1.4. Routage des messages au moment de connexion d'un nouveau nœud.

La recherche Gnutella :

Chaque pair possède un index de ses propres fichiers, la recherche se fait comme suit :

1. Le pair envoie un message Query au(x) premier(s) nœud(s) qu'il connait.
2. A la réception d'une requête Query, le pair parcourt son index afin de trouver le fichier recherché et établir la connexion avec le pair source.
3. Sinon il propage la requête au(x) nœud(s) qu'il connaît. Ces derniers reprennent le processus à partir de l'étape 2, jusqu'à trouver le nœud qui possède la ressource. Ce dernier envoie un message '*Pong*' au peer source qui répond par une requête '*Get*' pour commencer le transfert de fichier

Les avantages [Fel 02]

- Administration simple, car mutualisé.
- Topologie évolutive.
- Disponibilité du réseau, on ne peut arrêter.

Les limites [Fel 02]

- Réseau rapidement inondé par des ping-pong.
- Supporte mal la montée en charge du nombre d'utilisateurs.
- Manque de fiabilité dans ses requêtes.

5.2 Architecture hybride

La différence entre l'architecture hybride et celle purement décentralisée est que la première maintient toujours l'utilisation d'un élément central. Elle se distingue également de l'architecture client/serveur par le fait que le partage des ressources est maintenu par les pairs. D'où la définition : un réseau pair à pair est classé dans les réseaux pair à pair hybride si l'existence d'une entité centrale est nécessaire pour fournir une partie de service offert par le système. [Sch 01]

a. Architecture hybride avec un seul serveur

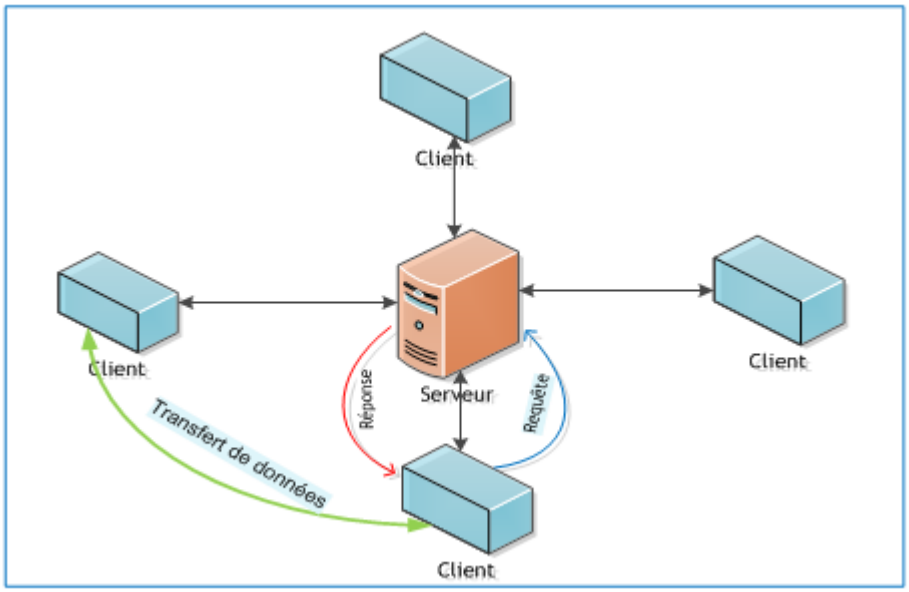


Figure 1.5. Schéma d'une architecture P2P hybride.

- Un serveur est connecté pour obtenir des méta-informations (identité du pair sur lequel sont stockées les informations, vérifier les crédentials de sécurité).
- Echange réalisée en pair à pair.

Exemple d'utilisation de l'architecture hybride : Napster[3], Goove, Magi.

Napster :

Fondé en mai 1999 par Shawn Fanninget SeanParker, est considéré comme le premier P2P, même s'il possédait une architecture centralisée. C'est en effet le premier logiciel d'échange à avoir connu un succès mondial. Sa technologie a permis aux gens d'échanger facilement des chansons au format MP3.

Au moment de connexion :

Un pair se connecte directement au serveur avec des primitives de service: login, request, login ok, login failed… Le pair communique au serveur central la liste des fichiers qu'il partage ainsi que le numéro de port utilisé pour le téléchargement et l'échange des données en utilisant une connexion ad hoc. [Fel 02]

Au moment de la recherche :

1. un pair envoie une requête au serveur central.
2. Le serveur lui répond par une liste des pairs qui hébergent le fichier recherché.
3. Le pair sélectionne le pair cible qui répond le mieux à certains critères comme sa bande passante, sa disponibilité,...etc.
4. Il contacte directement le pair cible pour commencer le téléchargement.

Les avantages :

- Avantages habituels d'un serveur central :
  - ✓ Facile à administrer.
  - ✓ Facile à contrôler, à fermer.
- Evite les recherches couteuses sur le réseau :
  - ✓ Pas de routage.
  - ✓ Planification de la gestion des utilisateurs.
- Tolérance aux fautes :
  - ✓ Par un sondage régulier des pairs connectés, état cohérent.
- Service novateur :
  - ✓ Généralise le pair to pair.
  - ✓ Généralise les procès contre le non -respect des droits d'auteur. [Fel 02]
  - ✓

[3] http://www.napaster.com.

Les limites :

- Pas d'anonymat par tout.
- Limites habituelles d'un serveur central :
  - Disponibilité.
  - Passage à échelle :
    - Saturation de la bande passante.
    - Saturation du nombre de processus.
  - Facile à fermer.
- Mauvaises informations du débit des pairs pour ne pas être sollicités. [Fel 02]

Afin de pallier les problèmes rencontrés dans les architectures pair à pair centralisées (limites du serveur) des améliorations ont été proposées dans le but de promouvoir la tolérance aux pannes ainsi que la disponibilité du système.

b. Architecture hybride avec plusieurs serveurs

Le principe consiste à remplacer le serveur par un réseau de serveurs. Ceci permet d'éviter l'arrêt de système si un des serveurs tombe en panne. Un serveur doit être en mesure de connaitre tous les serveurs sur le réseau afin de permettre à ses clients de faire des recherches sur les groupes de pair distants. Au moment de la connexion, le client doit choisir un des serveurs. La liste de ces derniers est soit disponibles sur internet ou bien fournie par l'application.

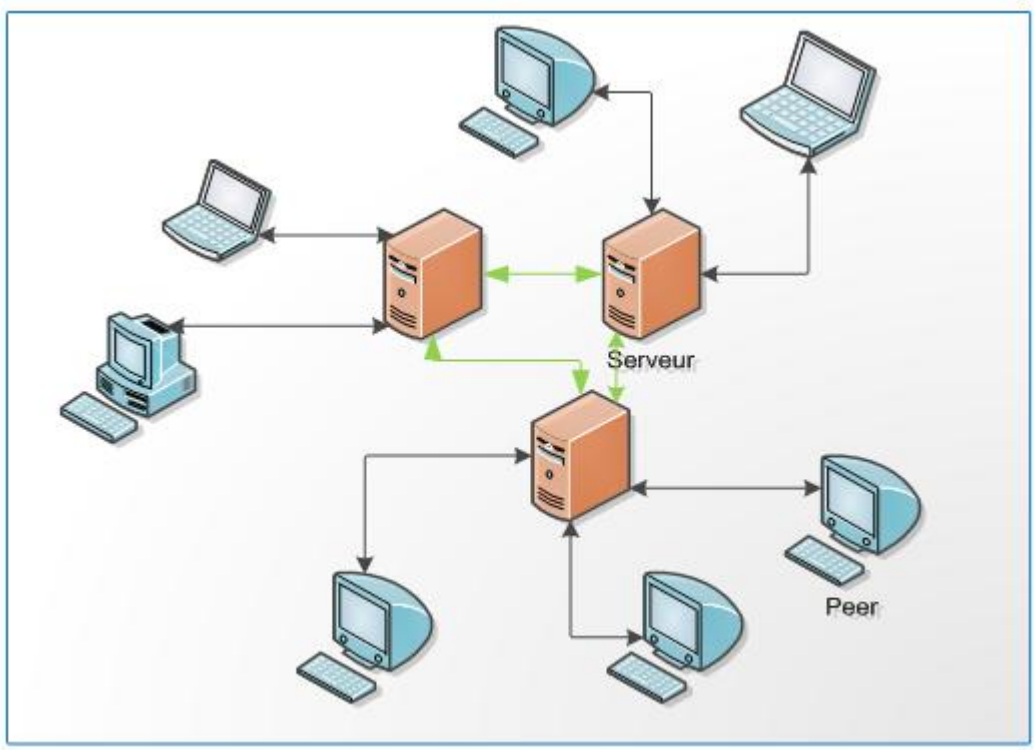


Figure 1.6. Architecture centralisée à plusieurs serveurs.

Exemple d'architecture à plusieurs serveurs : edonkey

edonkey :

A partir de 2003, le réseau eDonkey devient très populaire auprès du grand public souhaitant partager des fichiers. Il est aujourd'hui le Peer-to-Peer le plus utilisé.

Edonkey utilise des serveurs spécialisés, chacun comporte des index contenant les peers qui lui sont associés ainsi que les fichiers partagés par ceux-ci. Lorsqu'un client se connecte à un serveur, il partage la liste des ressources à sa possession. Le serveur à son tour met à jour son index des peers et des ressources. [Soy 05]

c. Architecture hybride partiellement centralisé (modèle super Peer)

Dans cette architecture il existe des pairs particuliers appelés "super peer" ou serveurs locaux qui gèrent un groupe de pairs. [Dij 06]

Ces super peers se caractérisent par :

- Une bonne bande passante.

- Puissance de calcul.
- Ils disposent d'un index des ressources de leur cluster.
- Une haute disponibilité.

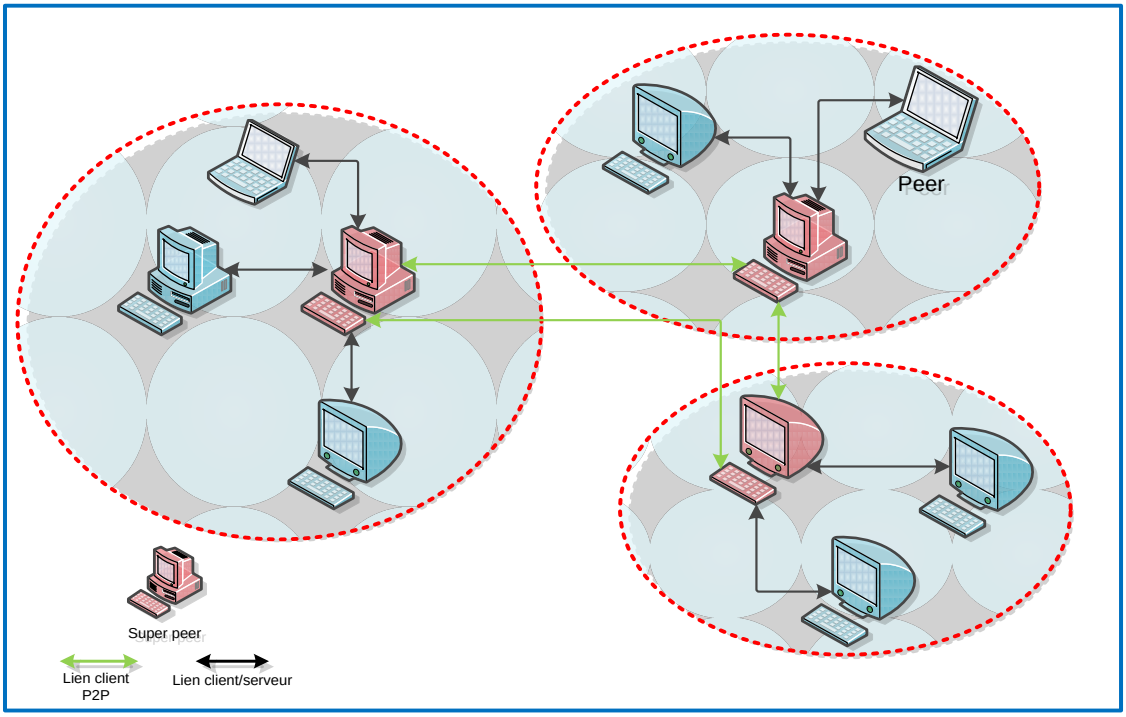


Figure 1.7. Schéma du modèle super peer.

Lorsqu'un client se connecte au système il est affecté à un des groupes. Au moment de la recherche, le client lance une requête classique (client/serveur) au super nœud, ce dernier parcourt son index afin de localiser le fichier. Si ce dernier n'existe pas au niveau de son groupe, alors deux cas se présentent :

- Soit il transmet la requête aux autres super-peer.
- Soit il transmet au client l'adresse d'un autre super-peer afin de pouvoir l'interroger.

Contrairement à l'architecture avec un réseau de serveurs, les super nœuds ne constituent pas un point de faille puisque leurs rôles sont assignés dynamiquement, et si un d'eux tombe en panne, le système entreprend automatiquement des mesures de remplacement.

Avantage :

- Indexation centralisée des ressources par les supers-peer.
- Offre une meilleure bande passante.

Inconvénients :

- Complexité dans la mise en œuvre.
- La fonction de super-peer consomme environs 10% des ressources du poste sur lequel il est opérant.

Exemple d'utilisation du concept super-Peer: Skype[4]

Skype :

Skype est une application Peer-to-Peer de voix sur IP qui se base sur le modèle super-Peer. [skyp@].

Au moment de l'installation le client dispose d'une liste contenant les super-nœuds reconnus. Au moment de la connexion l'un des super-nœuds doit être accessible pour que le client puisse se connecter. Une fois la connexion est établie le client constitue une liste des super-peer disponibles sur le réseau, au maximum 200, pour faire face au problème de déconnexion du super-peer sur lequel il est connecté.

Au moment de la recherche d'un contact, le super-peer communique à son client l'adresse (IP/Port) de quatre super-peers à interroger. Si le contact n'y est pas trouvé le client demande au super-peer huit autres adresses de super-peer pour les interroger et ainsi de suite jusqu'à trouver le contact. Pendant la recherche les résultats intermédiaires sont enregistrés par le super-peer afin d'optimiser les futures recherches.

6. Les catégories des réseaux pair à pair

Les réseaux pair à pair sont classifiés en deux catégories : les structurés et non structurés.

[4] http://www.skype.com.

6.1Pair à pair structurés

Les P2P structurés sont basés sur l'établissement d'une DHT (Distributed Hash Table) permettant de "placer" les nouveaux nœuds aux seins du réseau. Chaque nœud reçoit une liste de voisins avec lesquels il pourra communiquer. Il s'agit ici de voisins "logiques", ceux-ci pouvant se trouver à l'autre bout du monde. [Mar 06]

De plus, chaque pair est responsable d'une partie spécifique du contenu du réseau. Ils sont en général identifiés par un couple clé/valeur. Elle permet de réaliser des recherches avec un nombre de messages croissant de façon logarithmique, contrairement à certain algorithme proposant des techniques de "flood". Parmi ces P2P, on peut citer : Freenet.

6.2 Pair à pair non structurés

Un p2p est non structuré quand les liens entre les nœuds sont établis de façon arbitraire. La communication entre les nœuds est donc plus difficile à gérer. Parmi ces P2P, on peut citer: Gnutella, eDonkey.

7. Avantages des réseaux pair à pair

Dès leur apparition, les P2P se sont distingués des architectures traditionnelles grâce à leurs différents avantages :

- La répartition de la charge : les échanges sont gérés directement par les paires, ce qui élimine un des principaux problèmes des architectures clients/serveurs : la répartition de la charge. Les P2P ne posent pas le problème de congestion des réseaux qui peuvent se produire autour d'un serveur central devant répondre à de très nombreuses demandes. Chaque pair gère ses propres données et répond aux requêtes des autres pairs. [Mar 06]
- La capacité de stockage : chaque nœud ne possède qu'une infime partie des données du réseau, qu'il partage avec les autres ordinateurs. La capacité de stockage est ainsi infiniment supérieure à celle d'un serveur traditionnel, qui ne pourrait supporter une telle quantité d'information (que ce soit au niveau technique ou au niveau du coût de mise en place)
- Outils de travail collaboratif : la dynamique qui caractérise les systèmes de travail collaboratif coïncide considérablement avec les caractéristiques des

réseaux P2P. En effet, le pair à pair constitue un bon support à plusieurs approches et techniques utilisées pour assurer la cohérence des objets partagés dans tel systèmes

- Extensibilité : auto-configuration des pairs. Il est très facile de rajouter de nouveaux pairs. Ceux-ci sont gérés dynamiquement.
- La mobilité : les utilisateurs ne sont pas obligés à déplacer leurs ressources (documents texte, image, vidéo), ils ont la possibilité d'accéder à distance à leurs propres ressources et d'une façon uniforme via le réseau.
- Résistance aux pannes (sauvegardes croisées) : les données étant présentes sur de nombreux postes différents.
- La puissance de calcul : une application répartie, s'activant en général avec l'écran de veille des ordinateurs, basée sur la technologie Peer-to-Peer peut donc permettre de mutualiser la puissance de calcul non utilisée pour des recherches demandant des capacités considérable, qu'un serveur isolé ne peut posséder.

Conclusion

Les systèmes Pair à apir représentent l'architecture décentralisée la plus populaire. Grace à ses applications en particulier celle du partage et de communication, le pair à pair a connu un large succès auprès du grand public, le nombre d'utilisateurs des applications P2P est en croissance incessante. D'après [Dij 06], le trafic P2P représente 60% du trafic des réseaux hauts débit le jour, et 90% la nuit. Et avec le temps le domaine application pair à pair est de devenu de plus en plus élargi.

Dans ce chapitre on a présenté le pair à pair qui est un aspect fatal de l'informatique distribué. Dans le prochain chapitre on va discuter la technologie d'agents mobiles, qui est un autre aspect non moins important du P2P dans l'informatique distribuée en particulier dans IAD[5].

[5] Intelligence Artificielle Distribuée.

Chapitre II

Technologie

Agents mobiles

Introduction

Le terme d'agent mobile est issu principalement de deux domaines distincts : l'intelligence artificielle et les systèmes distribués avec la migration de processus. La technologie à base d'agent mobile aide à améliorer la performance de plusieurs applications distribuées avec des moyens intelligents. Amélioration peut être ressentie dans la réduction du trafic réseau, la répartition dynamique de charge, la commodité par rapport aux programmeurs ou simplement dans l'habilité de continuer l'interaction avec un utilisateur durant une déconnexion du réseau. L'utilisation des agents mobiles apport plusieurs avantages. Néanmoins, ils posent quelques problèmes, dont les plus importants sont ceux relatifs à l'interopérabilité, la sécurité et la tolérance aux pannes des systèmes d'agents mobiles.

Ce chapitre introduit le concept d'agent mobile, en mettant l'accent tout d'abord sur la notion d'agent (définition, caractéristiques, classification et architecture). Nous donnons ensuite des définitions de l'agent mobile. Nous donnons un aperçu sur les concepts fondamentaux des agents mobiles. Et enfin nous terminons par les travaux antérieurs et récents concernant les agents mobiles.

1. Les agents logiciels

Il existe à l'heure actuelle, une pléthore de définition de l'agent, mais tout le monde s'accorde à dire que la notion d'autonomie est au centre de la problématique des agents. Nous avons retenu les définitions suivantes :

1.1 Définitions

Définition 1 : « *Agent n.m. 1. Toute personne ou tout phénomène physique qui a une action déterminante. 2. Celui qui est chargé d'une mission par une société, un gouvernement, un particulier.* » [Dictionnaire Larousse].

Définition 2 : « *un agent est un système informatique situé dans un environnement, et qui agit d'une façon autonome et flexible pour atteindre les objectifs pour lesquels il a été conçu* ». [Jen 98].

Définition 3 : « *On appelle agent une entité réelle ou abstraite qui est capable d'agir sur elle-même et sur son environnement, qui dispose d'une représentation partielle de cet environnement, qui, dans un univers multi-agents, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de sa connaissance et des interactions avec les autres agents.* » [Fer 95].

Parmi ces définitions, nous avons choisi celle de Ferber, un agent est :

- Une entité autonome qui peut offrir des services.
- Une entité dont le comportement est la conséquence de ses objectifs, de sa perception, de ses compétences et des communications qu'elle peut avoir les autres agents.
- Une entité qui possède des ressources.
- Une entité qui est apte à agir sur l'environnement du système auquel il appartient,
- Une entité qui peut communiquer avec les autres agents.
- Une entité qui est capable de se reproduire.

1.2 La structure interne des agents

Dans un agent on distingue essentiellement : le savoir-faire, les croyances, la connaissance de contrôle, l'expertise de l'agent et la connaissance de la communication.

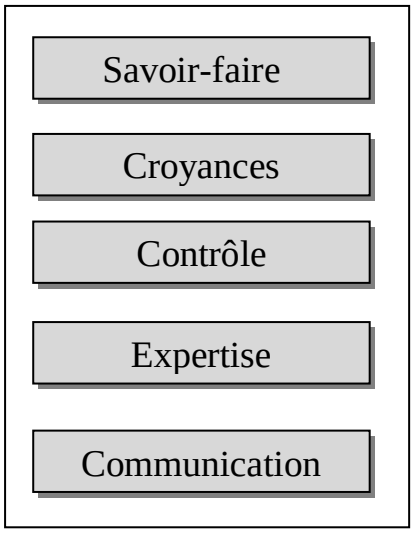


Figure2.1 : structure interne d'un agent [Gue 01]

- Savoir-faire : est une interface permettant la déclaration des connaissances et des compétences de l'agent. Il permet la sélection des agents à solliciter pour une tache donnée.
- Croyances : dans un univers multi-agent, chaque agent possède des connaissances sur lui-même et sur les autres. Ces connaissances ne sont pas nécessairement objectives.
- Contrôle : la connaissance de contrôle dans un agent est représentée par les buts, les intentions, les plans et les taches qu'il possède.
- Expertise : c'est la connaissance sur la résolution du problème.
- Communication : l'agent doit posséder un protocole de communication lui permettant d'interagir avec les autres agents pour une bonne coopération et une bonne coordination.

1.3 Les propriétés des agents

Parmi les nombreuses propriétés pouvant qualifier un agent, on opte les suivantes :

- **L'autonomie**

Permet à l'agent de produire des actions à partir de perception, et éventuellement d'états internes, ou d'une mémoire. Aucune intervention extérieure. Cela signifie que l'agent doit être capable d'accomplir sa mission son supervision ou intervention de tiers (humain ou agent) [Cai 07]

- **La réactivité**

C'est la capacité qu'un agent puisse percevoir les changements environnementaux et modifier son comportement en élaborant des réponses dans les temps requis. [Cai 07]. Un agent réactif est un agent qui est capable de prévoir son environnement (qui pourra être utilisateur à travers une interface graphique…) et cela en maintenant un lien constant avec celui-ci afin de répondre aux changements qui y survienne.

- **Sociabilité**

Permet à l'agent de communiquer avec les autres agents, ou avec l'utilisateur par le biais un langage symbolique [Hon 07] [Pan 08] dont le but de résoudre son problème et d'aider les autres dans leurs activités.

- **Situation**

Un agent est un système informatique situé dans un environnement, cela signifié que l'agent peut recevoir des entrées sensorielles provenant de son environnement et qu'il peut effectuer des actions qui sont susceptibles de changer cet environnement.

- **L'adaptabilité**

C'est la capacité de l'agent à s'adapter aux changements de l'environnement. Pour cela l'agent adaptatif possède des mécanismes d'apprentissage et/ou d'évolution, qui s'appliquent durant la vie de l'agent dans l'environnement.

1.4 Taxonomie des agents

1.4.1 Les agents réactifs

Ils possèdent quelques compétences leur permettant d'accomplir une ou plusieurs actions, ces compétences sont statiques et n'évoluent pas a travers le temps, et ceux, a cause du fait qu'ils ne peuvent pas tenir compte des actions passées parce qu'ils n'ont pas d'histoire c'est-à-dire qu'ils ne possèdent pas de mémoire. Leur comportement est de type « stimuli réponse ».

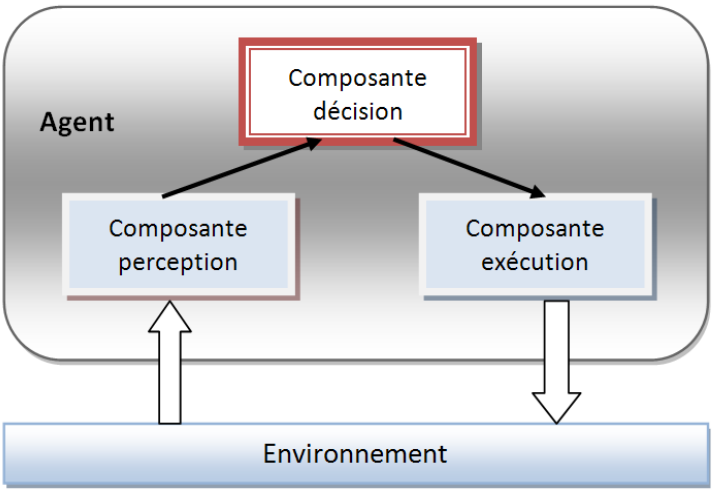


Figure 2.2 : Agent réactif [Flo 02].

1.4.2 Les agents délibératifs

Les agents délibératifs, également appelés cognitifs [Pan 08], sont issus des premiers travaux de IA[1]. Et d'après [Gue 01] sont les agents qui disposent d'une capacité de raisonnement, d'une aptitude à traiter des informations diverses liées au domaine d'application, et des informations liées à la gestion des interactions avec les autres agents et l'environnement.

Les deux caractéristiques principales de l'agent cognitif sont :

1. Une forte intentionnalité. aspire continuellement à la réalisation d'un certain nombre de buts explicitement définis par le propriétaire [Pan 08].
2. L'importance centrale du processus de décision. Synthétiquement, l'agent est capable d'évaluer toutes les situations avant de choisir la meilleure action, en adéquation avec les buts à accomplir [Pan 08].

1.4.3 Les agents hybrides

C'est une architecture composée d'un ensemble de modules organisés dans une hiérarchie, chaque module étant soit une composante cognitive, soit une réactive. De cette manière, on combine le comportement proactif de l'agent, dirigé par les buts, avec un comportement réactif aux changements de l'environnement. En plus, on espère obtenir simultanément les avantages des architectures cognitives et réactives, tout en éliminant leurs limitations [Flo 02].

1.5 Quelques exemples d'architectures

L'architecture d'un agent est la description de son organisation interne, les données et les connaissances de l'agent, les opérations qui peuvent être effectuées sur ses composantes, et le flux de contrôle des opérations.

Le choix d'une architecture ou d'une autre est lié à la structure conceptuelle de l'agent et représente la décision du concepteur sur la façon de bâtir l'agent artificiel.

Agent cognitifs--------->architecture BDI

Agent réactifs--------->architecture de subsomption.

Agent hybrides--------->architecture InterRap.

[1] IA Intelligence Artificielle.

1.5.1 Architecture BDI (**B**lief ***D**esire **I**ntention*)

Les architectures BDI sont l'exemple le plus représentatif d'architecture permettant de bâtir un agent cognitif.

Blief ou croyances ce sont les informations que l'agent possède sur son environnement et sur les autres agents. Les croyances peuvent être incomplètes, incertaines, ou incorrectes, contrairement aux connaissances de l'agent qui sont toujours vraies.

Desire ou désirs représentent les états de l'environnement parfois de lui-même que l'agent aimerait voir réalisés.

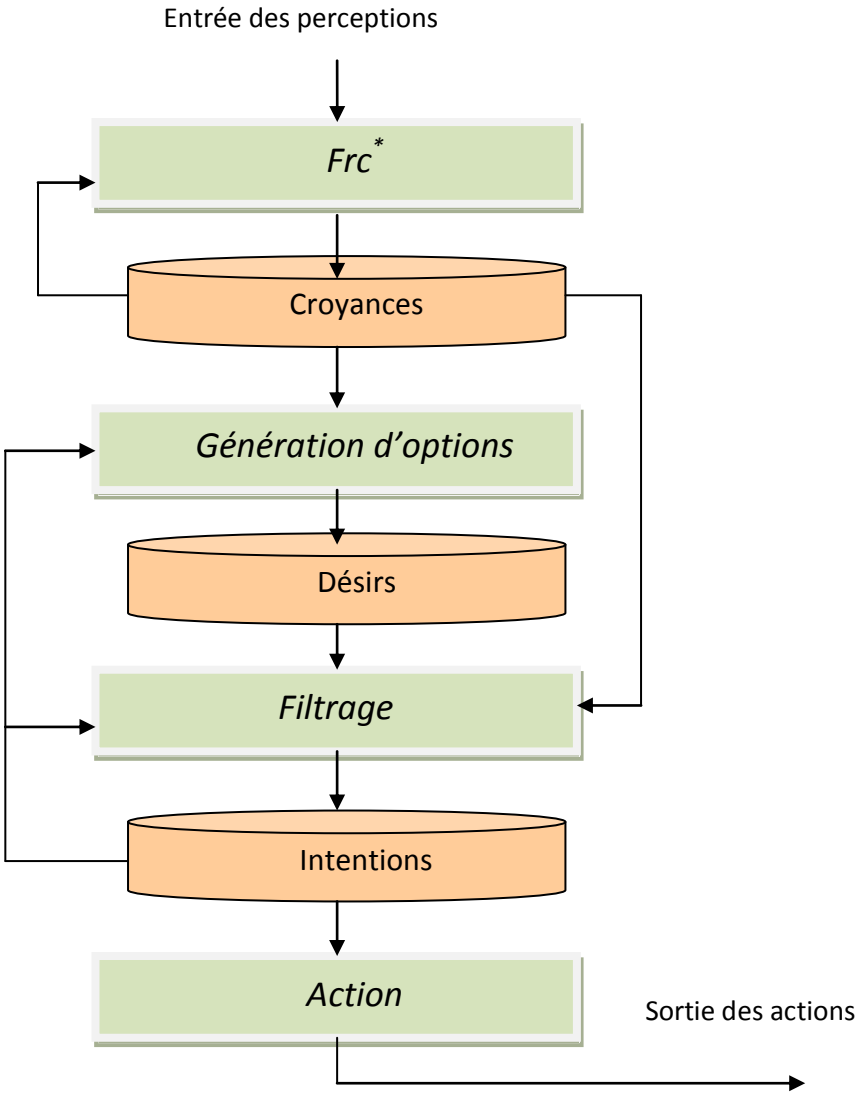


Figure 2.3 : Architecture d'agent BDI [Gue 01]

*Frc : fonction de révision des croyances.

Intention c'est les actions que l'agent a décidé de faire pour accomplir ses désirs.

L'architecture BDI est composée de :

Structure de données représentant les croyances, désirs et intentions de l'agent.

Les fonctions qui représentent les délibérations des croyances, des désirs et des intentions.

Le raisonnement : décidé comment faire.

Cette architecture basée sur le raisonnement pratique. Les intentions jouent un rôle central dans ce modèle [Gue 01]

1.5.2 Architecture à subsomption

L'architecture à subsomption [Bro 86] en anglais « subsumption architecture », qui représente l'architecture réactive la plus influente [Flo 02], proposé par Rodney Brooks.

Principe de l'architecture à subsomption :

- L'agent raisonne en utilisant un ensemble de modules ou chaque module étant responsable de la réalisation d'une tache simple.
- On a un ordre complet sur ces modules.

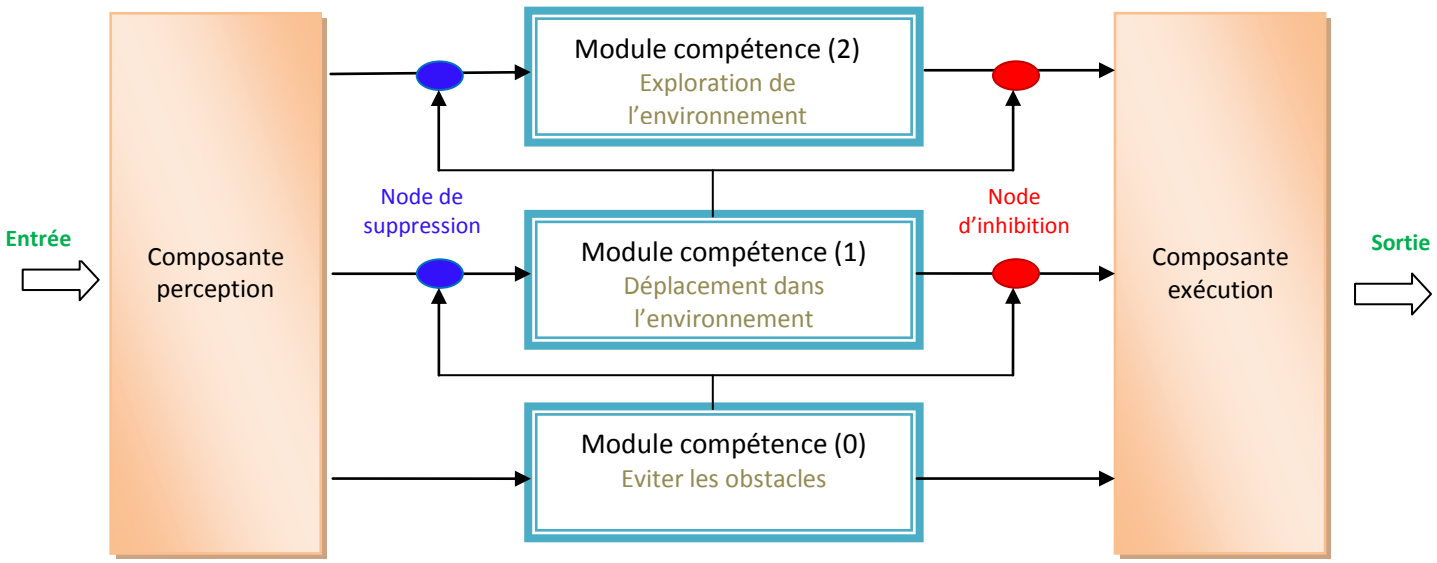


Figure 2.4 Architecture réactive à subsomption [Flo 02].

- Chaque module peut :

  - Restreindre les possibilités du module suivant (action inhibition).
  - Modifier les choix du module suivant (action de correction).

Les modules sont organisés en couches hiérarchisées, les couches supérieures ayant une priorité plus petite que les couches inférieures.

1.5.3 Architecture InterRap

Plusieurs architectures hybrides ont été conçues et utilisées. Une des architectures hybrides les plus connues est celle du système InteRRaP (‘Integration of Reactive Behavior and Rational Planning’ = ‘Intégration du comportement réactif et planification rationnelle’) [Flo 02]. C’est une architecture en couches avec des couches verticales où les données d'entrée, notamment les perceptions, passent d'une couche à l'autre, comme on le voit sur la figure 3.5. En cela, elle est un peu différente de l'architecture de subsomption qui est une architecture en couches horizontales, où les perceptions sont transmises directement à toutes les couches à la fois [Flo 02].

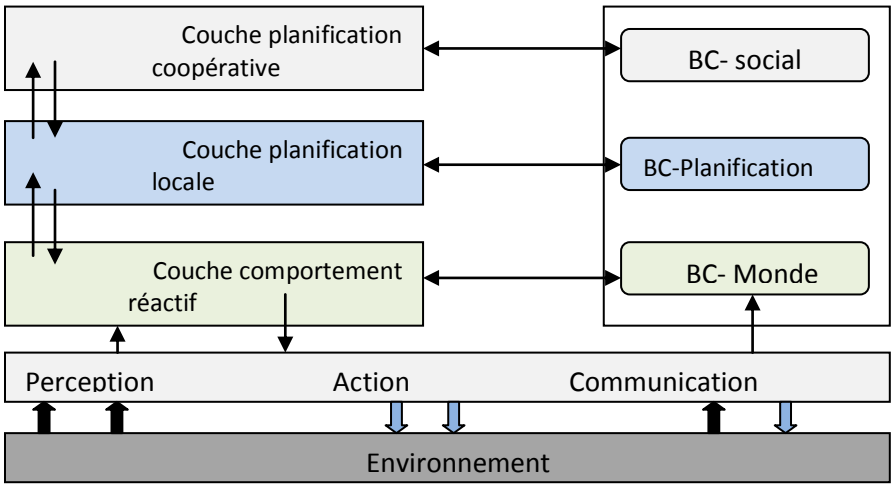


Figure 2.5 : Architecture InteRRap [Flo 02].

2. Les Agents mobiles

La mobilité est considérée dans la majorité des cas comme une propriété orthogonale des agents, qui ne sont pas tous forcement mobiles. Il existe deux sorte d’agents, ceux qui s'exécutent uniquement sur le système sur lequel ils sont créés, on appelle se genre les agents

stationnaires, et ceux qui ont la capacité de se migrer d'un site à un autre, sont les agents mobiles. Les agents stationnaires ne peuvent pas se déplacer sur le réseau, s'ils ont besoins d'informations ou d'interagir avec un agent sur un autre système, ils utilisent généralement les mécanismes de communication classiques tel que l'appel de procédure à distance (RPC[2]). En revanche, l'agent mobile n'est pas fixé au système sur lequel il commence l'exécution [Jen 98], il est libre de voyager à travers un réseau hétérogène, d'un hôte à un autre au cours de son exécution accéder à des données ou à des ressources nécessaires pour l'accomplissement de sa mission. Il se déplace avec son code, ses données propres et aussi avec son état d'exécution.

Le paradigme des agents mobiles propose d'utilisé la migration du code afin de supprimer la contrainte de connexion constante. Les deux parties doivent se connecter seulement durant la phase de migration. La figure suivante illustre ce paradigme, un client donne une mission à un agent, qui pour la réaliser se déplace dans le réseau de machines accédant localement aux services offerts par ces machines. On distingue trois phases :

- L'activation de l'agent mobile avec la description de sa mission.
- L'exécution de la mission par l'agent qui se déplace pour accéder aux services.
- La récupération éventuelle des résultats de l'agent mobile.

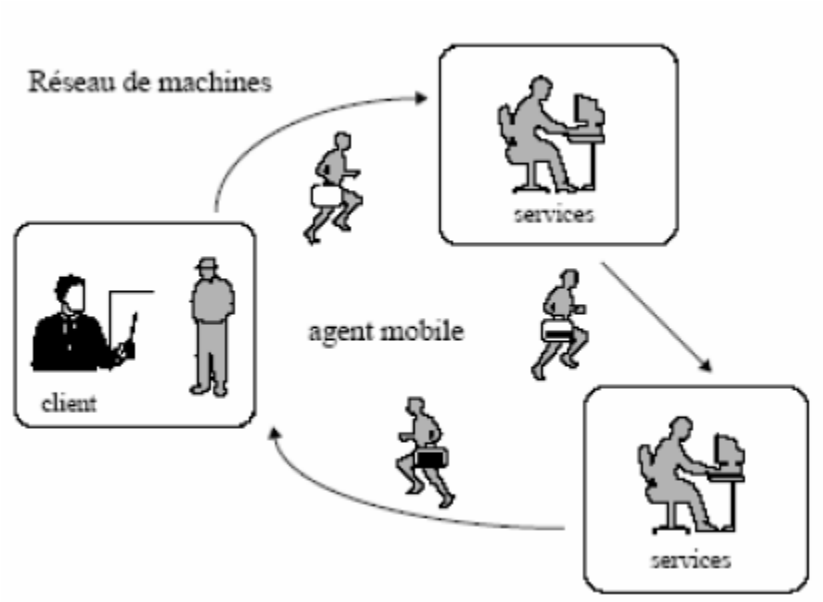


Figure 2.6 : le paradigme des agents mobiles. [Per 97]

[2] RPC : Remote Procedure Call

2.1 Quelques Définitions

Les agents mobiles sont définis comme étant :

Définition 1 :

J. Ferber a définit l'agent mobile comme *« Un agent logiciel qui peut se déplacer d'un site à un autre en cours d'exécution pour se rapprocher de données ou de ressources. Il se déplace avec son code et ses données propres, mais aussi avec son état d'exécution. L'agent décide lui même de manière autonome de ses mouvements. En pratique, la mobilité ne se substitue pas aux capacités de communication des agents mais les complète (la communication distante, moins coûteuse dans certains cas, reste possible). Afin de satisfaire aux contraintes des réseaux de grande taille ou sans fil (latence, non permanence des liens de communication), les agents communiquent par messages asynchrones. »* [Fer 95].

Définition 2 :

« Un programme qui migre à sa propre initiative d'une machine à une autre dans un réseau hétérogène. Dans chaque machine l'agent interagi avec les services des agents stationnaires et d'autres ressources afin d'accomplir une tache particulière, définie par son propriétaire. » [Sec 06]

Les agents mobiles sont des programmes dotés des capacités d'adaptabilité, d'autonomes qui peuvent se déplacer à travers un réseau hétérogène d'un hôte vers un autre sous leur propre contrôle. Dans le but est de fournir un service précis à son propriétaire

2.2 Attributs d'un agent mobile

Un agent mobile est une entité qui possède cinq composants : son état, son implémentation, son interface, son identifiant et son autorité. [Bar 03] L'agent transporte ses attributs lors son déplacement à travers le réseau.

a. L'état :

L'état d'un agent peut être considéré comme une photo instantanée de son exécution, dans la plupart des langages de programmation. On peut partitionner cet état en état d'exécution (qui inclut ses programmes et ses interfaces) et un état d'objet (qui contient les valeurs des variables d'instance dans un objet).

b. Le code:

C'est la logique de l'agent. Il contient les différentes fonctionnalités de l'agent. Les agents de même type ont le même code. Ce code doit être identifiable, lisible et surtout facilement séparable de sa plateforme locale pour pouvoir être éventuellement transférer à une plateforme distante. Pour cela il est généralement sous forme de code interprété. [Bra 05]

c. L'interface :

Un agent fournit une interface qui permet aux autres agents et systèmes d'interagir avec lui. Cette interface peut être un ensemble de méthodes qui permet aux agents et applications d'accéder aux méthodes de l'agent par un système de messagerie.

d. L'identifiant :

Chaque agent possède un identifiant unique durant son cycle de vie, qui lui permet d'être identifié et localisé. Puisque l'identifiant est unique, il peut être utilisé comme clé dans les opérations qui exigent un moyen pour référencer une instance particulière d'agents.

e. L'autorité :

Une autorité est une entité dont l'identité peut être identifiée par n'importe quel système auquel elle essaye d'accéder. Pour les agents il existe principalement deux types d'autorité :

- Le fabriquant (manufacturer) qui est le fournisseur du code d'implémentation de l'agent.
- Le propriétaire (owner) qui a la responsabilité du comportement de l'agent.

2.3 Caractéristiques des agents mobiles

Les principales caractéristiques d'un agent mobile sont :

1. **La mobilité :** c'est le mécanisme utilisé pour transporter l'agent entre les différents sites. Le déplacement d'un agent est contrôlé par la fonction de migration faible (transport du code et de données) ou forte (transport du code, de données et de l'état de l'exécution). [Men 04]

2. **La communication :** les agents mobiles peuvent communiquer sur le même site (communication locale) ou sur des sites différents (communication à distance). Les moyens de communication sont multiples tel que les envois des messages, le tableau blanc, les points de rendez-vous.
3. **La gestion des ressources :** l'agent mobile contrôle ses propres ressources afin d'optimiser leurs performance et satisfaire aux exigences des hôtes.
4. **La tolérance aux pannes :** comme un agent mobile s'exécute sur un site distant, il pourra tomber en panne (déconnexion, défaillance,…) il faut donc définir des mécanismes de récupération pour relancer l'agent à nouveau, une certaine redondance (copie par exemple) est nécessaires.
5. **La sécurité :** l'agent mobile doit s'assurer que ni son code ni ses données ne seront modifiés par le système hôte.
6. **Sérialisation / Désérialisation :** l'agent mobile est composé du code, données et état d'exécution. Avant d'être migré à une autre machine, ceux-ci doivent être codés sous la forme d'une chaine de caractère ; on dit que l'agent est sérialisé (processus de sérialisation). La désérialiation est le processus inverse. [Men 04]

2.4 Les types d'agents mobiles

Plusieurs types d'agents mobiles peuvent exister. On détaille ci-dessous les principaux types d'agents mobiles qu'on a rencontrés dans la littérature :

- **Agent statique :** Un agent statique est un agent mobile qui n'exécute en général qu'une seule migration. Cette migration s'effectue depuis la station de départ vers un nœud bien défini au lancement. A l'arrivée sur le nœud, l'agent mobile ne migre plus mais exécute une tâche prédéfinie.
- **Agent visiteur :** Un agent visiteur est un agent mobile qui visite successivement les différents nouds de la plate-forme afin d'y appliquer la même fonction d'administration du réseau, par exemple, en récupérant les versions des systèmes d'exploitation installées sur les éléments du réseau.
- **Collecteur de données** : Un agent collectant les données est un agent mobile qui nécessite un point de rendez-vous avec un ou plusieurs autres agents mobiles pour les décharger des données que ces derniers ont collectées. Après la collecte, cet agent transporte les données jusqu'à un système défini à l'avance.

- **Agent de suivi d'incident :** Un agent de suivi d'incident est un agent qui est programmé pour réagir aux données qu'il analyse.

2.5 La sécurité des agents mobiles

La sécurité est un problème inhérent à l'emploi des agents mobiles. Il constitue un frein à l'utilisation réelle de cette technologie [Hac 08]. En effet, deux types de risques sérieux de sécurité sont engendrés : i) les attaques pouvant être perpétrées par un agent mobile malveillant, ii) les attaques émanant d'un hôte malicieux hébergeant l'agent mobile. Le problème de la sécurité des agents mobiles s'énonce par la résolution des problèmes suivants :

1) La protection du système vis-à-vis des agents s'exécutant dans le système.
2) La protection de l'agent durant le transport dans le réseau.
3) La protection de l'agent vis-à-vis du système ou il s'exécute.

Le premier permet d'éviter l'intrusion d'agent malveillant, qui est aujourd'hui bien maitrisé. Il se résout classiquement par l'authentification et la protection. A l'opposé de la protection des systèmes, la protection des agents contre des sites malveillants ne dispose pas de solution éprouvée et reste à l'ère actuelle encore un champ de recherche ouvert [Cub 05]. Par contre le problème du transport d'un agent se résout par des techniques de signature digitale assurant qu'un message transmis n'a pas été modifié.

La sécurité est probablement le problème majeur des agents mobiles, car l'adaptation des mécanismes de sécurité tels que la cryptographie diminue la performance des agents. Plusieurs modèles et architecture de sécurité ont été proposés pour remédier ce problème.

2.6 L'apport des agents mobiles

Les agents mobiles ont rapidement suscité un intérêt tout particulier dans les domaines de recherche portant sur les applications réparties. Très rapidement, cette nouvelle méthode de programmation a été évaluée afin de voir ce qu'elle pouvait apporter comme caractéristiques propres. La raison de cet intérêt, est en fort simple, les agents mobiles permettent de dépasser les limitations des méthodes de programmation classiques (modèle client/serveur). Dans cette section, nous allons exposer ces différents apports.

2.6.1 La performance :

Les premières améliorations apportées par les agents mobiles portent sur le gain de performances dues à une meilleure utilisation des ressources physiques mises à disposition.

L'amélioration va intervenir à différents niveaux permettant d'optimiser la tâche globale des agents. [Cub 05]

a) Diminution de l'utilisation du réseau : Le déplacement des agents mobiles permet de réduire significativement, voir supprimer, les communications distantes entre les clients et les serveurs. En privilégiant les interactions locales, l'utilisation du réseau va se limiter principalement au transfert des agents. Cette situation présente trois principaux avantages.
   - la diminution de la consommation de bande passante.
   - la diminution des temps de latence.
   - des brèves périodes de communication.

b) Des calculs indépendants : Avec les agents mobiles, un client peut déléguer les interactions avec le service sans maintenir une connexion de bout en bout. Avec cette possibilité d'un calcul indépendant, le client peut demander un service, se déplacer (ou simplement terminer une session) puis venir récupérer les résultats plus tard.

c) Optimisation du traitement : L'optimisation des phases de traitement se produit à deux niveaux. Premièrement, en localisant les ressources et le savoir faire sur un même site, on supprime les phases de dialogue entre le client et le serveur qui sont perturbées par des temps de latence dus aux communications réseaux Ensuite, le déplacement du savoir faire va permettre de déléguer les calculs sur des machines serveurs, un super - calculateur par exemple, qui sont généralement plus puissantes qu'une machine cliente. Cela est particulièrement vrai dans l'informatique nomade où la miniaturisation s'accompagne d'une perte de puissance significative.

d) Tolérance aux fautes physiques : En se déplaçant avec leur code et données propres, les agents mobiles peuvent s'adapter facilement aux erreurs systèmes. Ces erreurs peuvent être d'ordre purement physique, disparition d'un nœud par exemple, ou d'ordre plus fonctionnel, arrêt d'un service par exemple. Si on prend le cas d'un site perdant une partie de ses fonctionnalités, un service tombant en panne, l'agent pourra alors choisir de se déplacer vers un autre site contenant la fonctionnalité désirée. Ceci permet une bien meilleure tolérance aux fautes que le modèle statique classique. Cependant, notons que dans le cas d'un agent unique, s'il vient à disparaître pour une erreur interne à son savoir faire ou pour une erreur système, il est généralement très

difficile de détecter sa disparition. énumère plusieurs mécanismes basés sur la réplication, transparents aux agents, permettant de résoudre en partie ce problème.

2.6.2 La conception :

Avec la méthode classique, il va être très contraignant de décrire des algorithmes d'exploration (de réseau) ou bien encore de caractériser les déplacements des utilisateurs nomades. Avec les agents mobiles, les concepteurs disposent d'une méthode qui permet de décrire naturellement ce genre de comportement. Ainsi, on peut facilement mettre en place un déploiement et/ou une maintenance d'application sur un réseau ou encore suivre les utilisateurs dans leurs déplacements.

2.6.3 Le développement :

Le domaine des agents mobiles étant encore relativement jeune, ils se heurtent à de fortes contraintes lors des phases de développement. Qu'il existe à l'heure actuelle un trop grand nombre d'intergiciels pour agents mobiles qui possèdent chacun leurs propres défauts et qualités. Avec cette offre pléthorique, il est difficile de parler de standardisation et aucune ne semble encore s'imposer.

Le manque de standardisation se retrouve aussi dans les interactions entre agents, ainsi il n'existe pas de langage partagé par toutes les plates-formes.

2.6.4 La sécurité :

Pour ce qui est des agents mobiles, d'un point de vue de la sécurité, nous sommes face à un problème qui n'est pas encore intégralement résolu. C'est d'ailleurs le principal argument avancé pour expliquer la faible utilisation de ce paradigme. En effet, les agents mobiles représentent un nouveau champ d'investigation pour le domaine de recherche en sécurité, d'une part dans la protection des sites vis-à-vis des agents malveillants et d'autre part dans la protection des agents vis-à-vis des sites malveillants.

Selon leurs caractéristiques les agents mobiles ne construisent pas une solution idéale pour tous les types des applications répartis, mais simplement une possibilité nouvelle pour la construction des applications.

2.7 les normes des agents mobiles

Afin d'assurer un niveau d'interopérabilité entre les différentes plateformes d'exécution d'agents mobiles. Il existe au temps présent deux normes principales : il s'agit de la norme MASIF (Mobile Agent System Interoperability Facility) [Mil 98] et de la norme FIPA (Foundation for Intelligent Physical Agents) [FIP 02].

2.7.1 MASIF

MASIF [Mil 98] est un standard, pour les systèmes à base d'agents mobiles, qui a été spécifiée par l'Object Management Group (OMG), qui se préoccupe généralement de l'hétérogénéité entre les systèmes [Ber 09]. L'objectif principal de ce travail était de proposer un ensemble de définitions et d'interfaces afin d'avoir d'un niveau d'interopérabilité entre différents agents mobiles. Il ne s'agit pas de standardisé les opérations locales des agents[3]. Plutôt MASIF standardise la manière de gérer le code des agents[4], leur identification, la migration[5] et l'adressage local.

MASIF est une collection de définitions et d'interfaces, permettant une interopérabilité pour les SMA mobiles, ce système présente deux interfaces CORBA[6] [Lou 10]: MAFAgentSystem et MAFFinder. Ces deux interfaces ont été définies au niveau de système d'accueil des agents et n'ont pas au niveau de l'agent.

L'interface MAFAgentSystem définit les opérations pour la gestion du cycle de vie de l'agent, assure aussi le transfert et la réception des classes d'agents mobiles. La deuxième interface MAFFinder se préoccupe de l'enregistrement et la localisation des agents.

2.7.2 FIPA

FIPA est une organisation de standardisation fondée en 1996 à Genève (suisse) [Lou 10]. La communauté d'origine de FIPA étant celle des systèmes multi-agents, plus proche de l'intelligence artificielle, elle va se situer à un niveau plus élevé, c'est-à-dire le niveau applicatif en décrivant les éléments nécessaires à la réalisation d'une application et principalement en détaillant la communication entre agents. [Cub 05] [Ber 09].

Les spécifications FIPA sont réparties en cinq catégories [Lou 10] :

[3] Opérations locales des agents telles que l'interprétation de l'agent, sa sérialisation et son exécution.
[4] Gestion des agents : création, suspension, reprise et terminaison.
[5] Sur des systèmes hétérogènes.
[6] CORBA : Common Object Request Broker Architecture.

- Les applications : des exemples de domaines d'application sur lesquels peuvent être déployé des agents FIPA.
- Les architectures abstraites : s'intéressent à la définition des entités abstraites nécessaires pour le développement d'un environnement d'agent.
- La gestion des agents : traitent le contrôle et la gestion des agents dans/entre les plateformes d'agents.
- Le transport des messages d'agents : traitent le transport et la représentation des messages à travers différents protocoles réseaux.
- La communication des agents : ces spécifications traitent les messages ACL (Agent Communication Lanquage), les protocoles d'échange de message, etc.

FIPA a publié plusieurs documents qui présentent les spécifications pour ces différentes catégories. Ces spécifications ont passé par plusieurs versions FIPA97, FIPA98 et FIPA2000.

Ces deux normes tendent plus vers la complémentarité que vers la divergence [Ber 09]. Leurs différences résident dans le fait que MASIF s'intéresse à l'interopérabilité des agents mobiles, elle lui permettre de migrer entre les systèmes d'agents mobiles du même profil par l'intermédiaire des interfaces normalisées de CORBA, contrairement à FIPA qui permet l'interopérabilité des agents intelligents par l'intermédiaire de la communication normalisée des agents et des langues.

2.8 Travaux concernant les agents mobiles

Pour déployer une application à base d'agent mobile, if faut disposer d'une plateforme appropriée.

On distingue trois approches pour concevoir et implanter une plateforme d'agent mobile. La première consiste à recourir à un langage de programmation qui comprend des instructions pour les agents mobiles. La deuxième approche consiste à implanter le système d'agents mobiles comme des extensions du système d'exploitation. Enfin, la dernière approche construit la plateforme comme une application spécialisée qui tourne au –dessus d'un système d'exploitation. [Pie 03]

Dans cette section, on va présenter quelques travaux concernant les agents mobiles. Ces plateformes se diffèrent dans leurs buts, motivations et implémentation mais elles fournissent toutes des fonctionnalités communes qui supportent la migration, la communication entre agents, plusieurs langages de programmation et plusieurs formes de sécurité. Nous allons

commencées par les travaux antérieurs, qui constituent les prémices de la notion d'agent mobile, allant jusqu'aux travaux qui sont apparut récemment.

2.8.1 Telescript

La notion d'agent mobile dans les réseaux est apparue en 94 avec Telescript, qui est développé par la société américaine General Magic, il est appliqué dans le commerce électronique.

Il utilise différents concepts : les places et les agents.

Les places sont constituées par les zones de services et les stations clientes du réseau.

Les agents sont des entités capables de :

- Migrer entre les places par l'instruction *go*.
- Rencontrer des autres agents sur la même place par l'instruction *meet*.
- Communiquer par établir des connexions permettant l'envoie de message à des agents qui se trouvent dans d'autres places.

Telescript est un langage interprété proche de LISP intégrant la notion d'objets.

Il fournit deux niveaux de langage : High Telescript et Low Telescript.

La migration d'un *agent* est réalisée au niveau de la machine Low Telescript.

2.8.2 Agile Applets (Aglets)

Aglets est une plateforme d'agents mobiles développée par IBM Japon, un Aglets est un objet Java mobile qui visite des environnements d'exécution (serveur). L'architecture des Aglets est similaire à celle des applets Java. Le système Aglets a son propre protocole pour le transfert des Aglets entres les hôtes : Aglet Transfer Protocol. [Pie 03]

2.8.3 Mole

Mole est une plateforme d'agents mobiles construite à l'Université de Stuttgart en Allemagne. [Pie 03] Elle implémente la migration partielle, ou seuls les données et l'état de l'agent sont transférés. La migration partielle a été développée après que les bâtisseurs du système se soient rendu compte que la migration totale (déplacement de processus d'exécution de l'agent) était trop couteuse quand il s'agissait d'agents « multi –threads ». Un agent est traité comme une grappe d'objet Java, un ensemble fermé sans aucune référence avec l'extérieur, excepté avec le système hôte.

2.8.4 Voyager

Voyager est un produit commercial proposé par ObjectSpace. Il s'agit d'un système « branché » incorporant plusieurs normes industrielles populaires à l'heure actuelle, notamment CORBA, DCOM et Distributed JavaBeans. Différent des autres systèmes, il ne s'agit pas d'un système dédié uniquement aux agents mobiles : toute sorte de développement distribué est supportée. Dans leur documentation, cependant, le concept d'agent mobile est explicitement présenté. Puisqu'il s'agit d'un système bien soigné, d'une nature générale, et compatible avec les normes existantes, sa popularité est croissante. [Objectspace @]

2.8.5 Java Agent DEvelopment Framework (Jade)

JADE (Java Agent Development Framework - Bellifemine, Poggi, Rimassa, 1999) est une plate-forme qui permet de construire des systèmes multi agents (SMA) entièrement implémenté en JAVA ; développé par le groupe CSELT1 Telecom Italia avec la coopération de l'université de Parme (Italie). [Ber 09]

JADE est un middleware qui facilite le développement des systèmes multi agents (SMA). JADE contient :

- **Un runtime Environment :** l'environnement ou les agents peuvent vivre. Ce runtime environment doit être activé pour pouvoir lancer les agents.
- **Une librairie de classes :** que les développeurs utilisent pour écrire leurs agents.
- **Une suite d'outils graphiques :** qui facilitent la gestion et la supervision de la plateforme des agents [Cai 07].

JADE est basée sur le protocole IIOP conforme aux normes FIPA (Foundation four Intelligent Physical Agents). [Ber 09] La communication des agents est basée sur l'envoi de messages FIPA-ACL (langage de communication des agents), qui est le langage standard des messages et impose le codage et la sémantique des messages.

Jade est compatible avec la plupart des configurations matérielles (PC, MAC, OS) et logicielles (Windows, Unix). Le seul système nécessaire pour son fonctionnement est la machine virtuelle JAVA.

Chaque instance du JADE est appelée conteneur " container ", et peut contenir plusieurs agents. Un ensemble de conteneurs constituent une plateforme. Chaque plateforme doit contenir un conteneur spécial appelé main-container et tous les autres conteneurs s'enregistrent auprès de celui-là dés leur lancement

La figure suivante illustre les concepts de base du Jade en montrant un petit exemple de deux plateformes jade composées respectivement de trois et un conteneur Chaque agent est identifié par un identifiant unique et peut communiquer avec n'importe quel autre agent sans avoir besoin de connaître son emplacement :

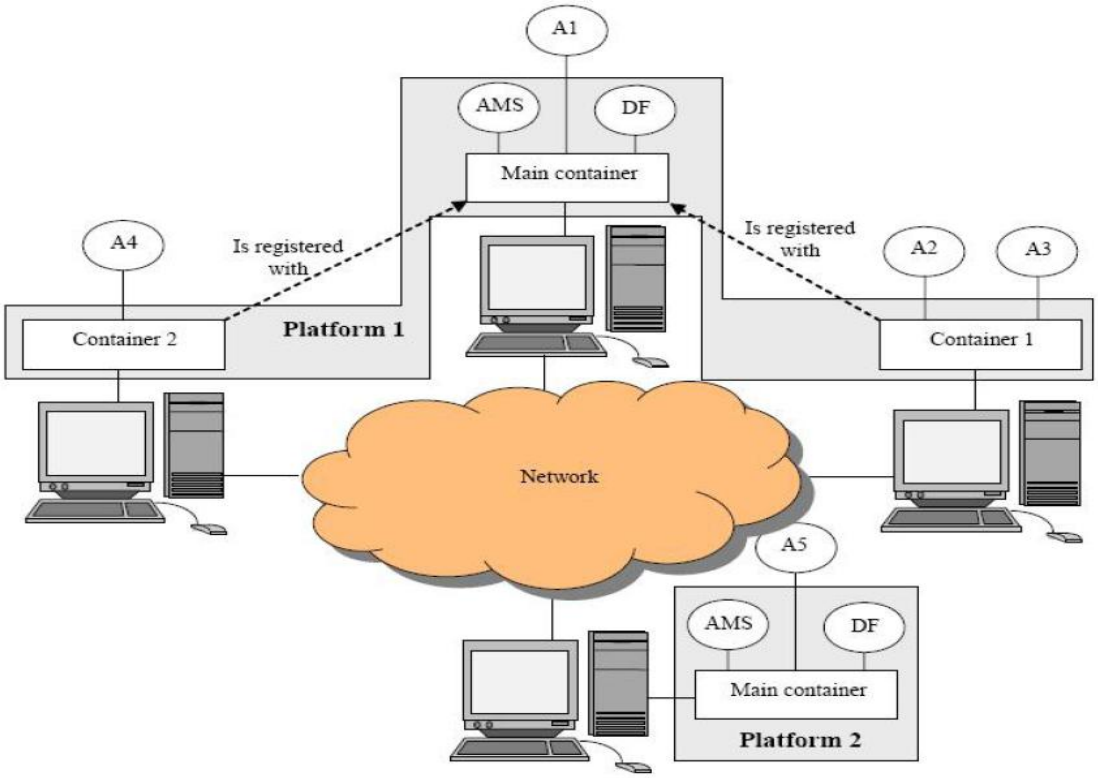


Figure 2.7. : Les concepts de base de la plateforme Jade.

- Dans le même conteneur (exemple agents A2 et A3)
- Dans la même plateforme mais dans des conteneurs différant (ex A1 et A2)
- Dans deux plateformes différentes (ex. A4 and A5).

Un main-container se distingue des autres " simples " conteneurs par une autre chose; il contient toujours deux agents spéciaux appelés **AMS** et **DF** qui sont lancés automatiquement au lancement du main-container.

- **AMS** (Agent Management System) qui fournit le service de nommage (pour assurer par exemple que chaque agent possède un identifiant unique dans la plateforme) et qui représente l'autorité de la plateforme.
- **DF** (Directory Facilitator) qui fournit un système de pages jaunes qui permet aux agents de retrouver les agents fournisseurs de services. [Cai 07].

2.9 Comparaison des quelques modèles de plateformes d'agents mobiles

Dans la section précédante nous avons essayés de décrire quelques plateformes jugées importantes. Pour des contraintes de forme, nous ne pouvons pas présenter la totalité des autres modèles. Afin d'en avoir un aperçu rapide de ces plateformes, nous avons regroupé les principales caractéristiques dans le tableau ci-dessous. [Ber 09]

plateformes	Agents		communication		propriété		Protocoles	langages
	Type	Mobilité	Mode	Type	Etat	Annuaire		
Aglets	proactif	Faible	asynchrone, synchrone	asynchrone, synchrone	Oui	Non	ATP RMI CORBA	Java
Ajanta	Proactif	Faible	Asynchrone, Synchrone	Asynchrone, Synchrone	Non	Oui	RMI	Java
JADE	Proactif, Réactif	Faible	Asynchrone	Asynchrone, Synchrone	Oui	Oui	RMI, http, IIOP	Java
JavAct	Proactif	Faible	Asynchrone, Synchrone	Asynchrone, Synchrone	Oui	Non	RMI, CORBA	Java
LIME	Supportées[7]		Synchrone	Asynchrone, Synchrone	Oui	Non	Sockets	Java
PLAGENT	Proactif	Faible	Synchrone	locale	Oui	Non	Sockets	Java
TACOMA	Proactif	Faible	les deux	locale	Oui	Non	Sockets	Multiple
Voyager	Proactif	Faible	Asynchrone, Synchrone	Asynchrone, Synchrone	Non	Oui	RMI	Java

Tableau 2.1. : Tableau comparatif des plates-formes mobiles

[7] LIME propose un médium de synchronisation. Il supporte donc les différents types d'agents en fonction du système sous- jacent.

2.10 Domaine d'application des agents mobiles

La technologie des agents mobiles est une approche très puissante pour les applications réseau avec des moyens intelligents. Une partie des applications ne nécessiteront pas cette technologie, mais certaines se révèleront plus efficaces en l'utilisant. Voici quelques exemples :

- **La collecte de données à partir de sources multiples** : les agents mobiles sont capables de collecter des informations réparties sur plusieurs machines reliées à un réseau.
- **Recherche et filtrage** (recherche personnalisée): un agent mobile pourrait visiter plusieurs sites, cherche à travers les informations disponibles dans chaque site et construire un index des liens pour les portions d'informations qui répondent aux critères de sélection [Maa 98].
- **La surveillance** [Ber 09] si tel évènement se produit sur le réseau, alors l'agent prévient le client. Exemple : un agent peut aller à une place boursière, attendre qu'une action particulière atteigne certain prix puis en acheter à la place de l'utilisateur [Esp 10].
- **Le calcul parallèle**, l'agent peut se dupliquer sur différents serveurs avec les différentes parties d'un calcul à effectuer.
- **Négociation** : l'agent part avec une offre du client et revient avec le meilleur prix obtenu.
- **Le commerce électronique**, l'agent pourrait localiser l'élément le plus approprié pour l'utilisateur.

Il y'a d'autre application comme la détection d'intrusion, jeux sur les réseaux, etc.

2.11 Les agents mobiles et les réseaux pair à pair

Les agents mobiles est une extension directe du système classique client/serveur [Maa 98], par conséquent, l'utilisation des agents mobiles reste limitée dans ce paradigme. Sauf dernièrement, certains chercheurs sont dirigés vers l'intégration de la technologie agents mobiles dans le paradigme pair à pair. Dans ce domaine on peut citer les travaux suivants :

- **A-Peer**[8]

A –Peer est une plateforme agent basé P2P[9], qui autorise les agents à être déployer dans un environnement pair à pair [Tie]. Son implémentation basée sur les Aglets et la plateforme JXTA[10]. La figure suivante illustre l'architecture de la plateforme A-Peer. La conception de l'architecture de l'ensemble APEER a un seul esprit: faire les agents exécutant au niveau des pairs [Tie]. Cette plateforme offre des moyens tente à facilite la programmation des agents dans un environnement pair à pair. Les fonctionnalités de sécurité et de gestion sont encore en cours de développement.

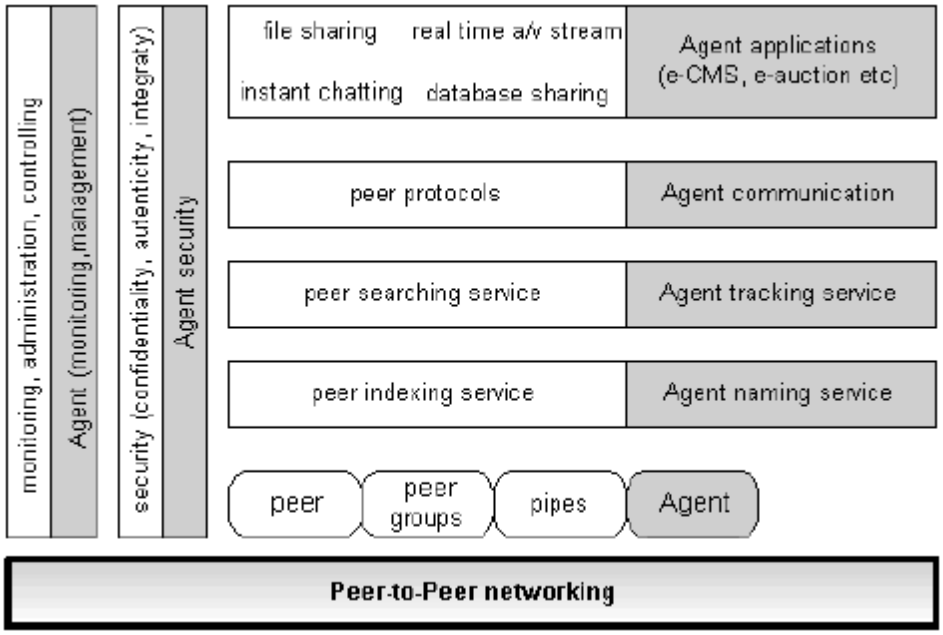


Figure 2.8 : L'architecture de la plateforme A –Peer [Tie]

- **UbiMAS** (ubiquitous mobile agent system)

C'est un système d'agents mobiles ubiquitaire, qui est utilisé pour accompagner et conduire les personnes dans un immeuble de bureaux avec plaques de porte à puce [Far 03]. UBiMAS est un système d'agent mobile qui fonctionne comme un service au-dessus de l'intergiciel ubiquiste à côté d'autres services comme l'emplacement de suivi des services. La figure suivante montre l'architecture de système UbiMAS.

[8] HTTP://sourceforge.net/projects/a-peer/
[9] Peer To Peer ou Pair à Pair en français.
[10] Voir chapitre IV.

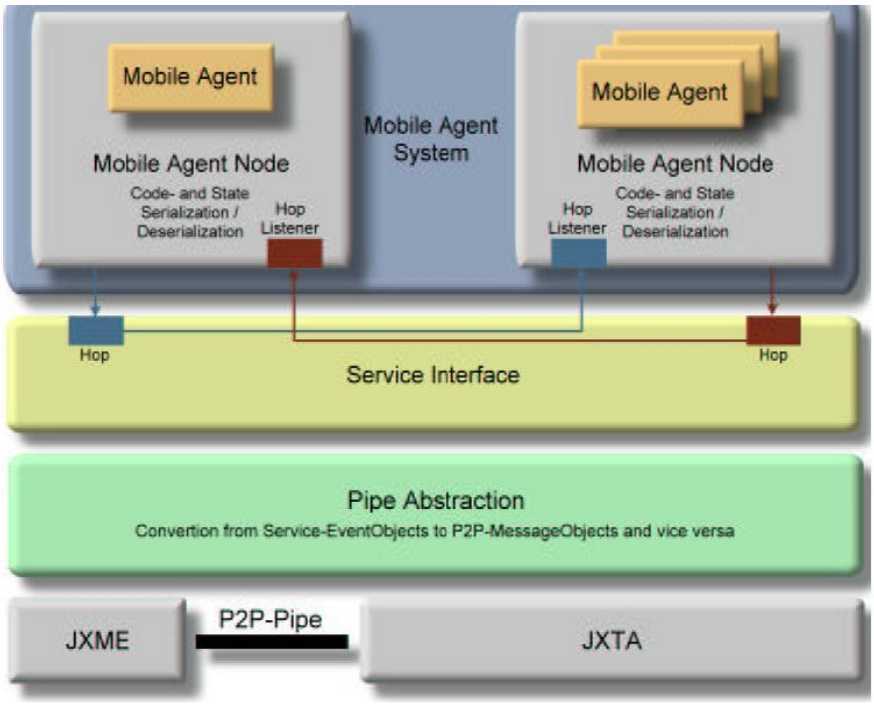


Figure 2.9 : Architecture d'UbiMAS. [Far 03]

- **SMAP** (Simple Mobile Agent Protocol)

C'est un protocole développé par Rita Yu Chen et Bill Yeager de Sun Microsystems, son implémentation basé sur le framework JXTA. D'après [Che 02] la raison de créer SMAP est de cibler les réseaux P2P plutôt que le traditionnel réseau client/serveur ou les précédents protocoles d'agents mobiles ont été appliqués. SMAP est un prototype, il est en cours de développement. [Che 02]

Conclusion

Les agents mobiles sont une catégorie particulière des agents logiciels dont la caractéristique prédominante est leur capacité de se migrer entre les nœuds d'un réseau ou de plusieurs réseaux. Les agents mobiles vu dés le début comme une extension directe du paradigme client/serveur afin de dépasser les limites de ce dernier. Alors les plateformes cité précédemment offre des avantages facilitant l'implémentation d'un agent mobile dans un réseau client/serveur. Mais dans notre cas en tente de pouvoir concaténer entre les deux technologies agents mobiles et peer to peer, par conséquence nous sommes besoins d'une plateforme qui répond mieux à la nature du réseau pair à pair (hétérogénéité, l'ouverture, la

dynamique,…). Pour cette raison nous avons choisi la plateforme JXTA. Dans le prochain chapitre nous présenterons en détail la plateforme JXTA que nous utiliserons, dans la suite de notre travail, pour combiner les concepts P2P avec les propriétés des agents mobiles.

Chapitre III
La plateforme JXTA

Introduction

Une analyse des systèmes pair-à-pair existants permet de constater que ces réseaux partagent un grand nombre de fonctionnalités communes. Malheureusement, ces fonctionnalités sont souvent volontairement implémentées par des mécanismes propriétaires, fermés. Afin de mettre en œuvre un nouveau type de service pair-à-pair, il est souvent nécessaire de construire un système complet, alors qu'un grand nombre de composants du système reposent sur des mécanismes déjà implémentés précédemment. Par conséquent, un investissement important en temps et ressources matérielles et humaines est nécessaire lors de la validation de nouvelles techniques pair-à-pair, car cela nécessite à chaque fois la mise en œuvre des mécanismes de base. Malgré leur grand succès les applications peer to peer restent incompatibles.

La compagnie Sun MicroSystem propose une nouvelle technologie le projet JXTA, un ensemble de protocoles pair à pair permettent de gérer un certain nombre de fonctionnalités communes, comme la gestion des pairs, la découverte de ressources, l'invocation de services sur le réseau, la communication inter-pair, la sécurité, etc. Ces protocoles peuvent servir de briques de base pour la mise en œuvre de services pair-à-pair.

En effet JXTA est une évolution majeur de l'informatique distribué de la même manière que la été JAVA / XML pour les langages de programmation, et bien sur UNIXBSD Berkly pour les systèmes d'exploitation. Il ya un facteur commun entre eux :
Bill Joy et l'histoire de SUN
JAVA : est indépendant de la plateforme.
XML : indépendant des applications.
JXTA : indépendant des réseaux.
JVN JXTA Virtuel Network est une virtualisation des réseaux de la même manière que la JVM[1] est une virtualisation des processeurs physiques.

Ce chapitre fournit une introduction au projet JXTA, ses blocs de construction de base, ces concepts et ces protocoles.

[1] Java Virtuelle Machine.

1. Les objectifs JXTA

Le but du Framework JXTA est de créer une plateforme qui permette de construire, simplement et facilement, un grand ensemble de services et d'application distribués pour tout dispositif qui pourrait être un Peer. JXTA permet, entre autre, aux développeurs de se connecter sur le développement de leurs application, en créant des logiciels distribuées flexibles.

Le Framework JXTA a quatre Objectifs :

1-l'interopérabilité

2-l'indépendance vis-à-vis des plateformes logicielles et matériel

3-l'ubiquité

4-Sécurité à différents niveaux

- Interopérabilité:

Les applications pair-à-pair (P2P) actuelles sont pour un seul type de service (échanger de fichiers, audio,…..). Comme il n'y a aucune infrastructure commune, chaque créateur de software pair à pair crée sa propre communauté et développe les primitives de base des systemes. Cela rend les systèmes incompatibles et en plus constitue des offerts inutiles, JXTA tient à résoudre ce proposant des primitives et services fondamentaux communs. [Kra 02]

- L'indépendance de la plateforme :

La plupart des softs pairs à pair (P2P) actuels sont basés sur Windows et TCP/IP. Or ceci n'est pas l'unique type de plate-forme existant. Si une application doit tourner sur d'autres systèmes et communiquer avec le système de base, soit les services doivent être redéveloppés, soit des ponts entre les technologies doivent être prévus. JXTA propose d'assurer un système garantissant les échanges entre tous les types de plate-forme et de protocoles. [Kra 02]

- Ubiquité :

Il matérialise le désir que le Framework JXTA soit imprésentable sur n'importe quel périphérique doté d'un cœur digital (digital heartbeat). Cette notion est très générale puisqu'elle comprend les senseurs, l'électronique grand public, les PDA, les GSM, les routiers réseaux, les ordinateurs de bureau, les serveurs de données ou de traitements.
[Kra 02]

- Sécurité à différents niveaux :

Prise en compte dans le noyau JXTA (JXTA permet de réaliser des applications pour les entreprises soucieuses de la sécurité).

2. Définition

"*JXTA is an open network computing platform designed for peer-to-peer (P2P) computing by way of providing the basic building blocks and services required to enable anything anywhere application connectivity. The name "JXTA" is not an acronym. It is short hand for juxtapose, as in side by side. It is a recognition that P2P is juxtaposed to client-server or Web-based computing, which is today's traditional distributed computing model*". [Jpg 07]

JXTA est une plate-forme informatique d'un réseau ouvert conçu pour le peer-to-peer (P2P) au moyen de fournir les éléments de base et les services nécessaires pour permettre à n'importe quel appareil connecté sur le réseau, allant des téléphones cellulaires et assistants numériques sans fil pour PC et serveurs, de communiquer et de collaborer d'une manière P2P. JXTA créer un réseau virtuel où tout pairs peuvent interagir avec d'autres pairs et des ressources directement, même si quelques-uns des pairs ou les ressources sont derrière des firewalls et des traductions d'adresse réseau (NAT) ou sur des transports réseau différents.

Le nom "JXTA" n'est pas un acronyme. Il est raccourci pour juxtaposer, ou bien côte à côte. C'est une reconnaissance que le P2P est juxtaposée à client-serveur, ce qui est d'aujourd'hui un modèle de calcul distribué traditionnel.

JXTA fournit un ensemble commun de protocoles ouverts soutenu avec des applications de référence open source pour le développement des applications peer-to-peer.
Les protocoles JXTA normalisé la manière dont les peers :

- Se découvre les uns les autres.
- S'auto-organisent en peer groupes.
- S'annoncer et se découvrir les ressources réseau.
- Se communiquer les uns avec les autres.

Les protocoles JXTA sont conçus pour être indépendants des langages de programmation et de protocoles de transport aussi bien. Les protocoles peuvent être mis en œuvre dans les langages de programmation Java, C / C + +, NET, Ruby, et de nombreux autres. En outre, ils peuvent être mis en œuvre au-dessus de TCP / IP, HTTP, Bluetooth, et d'autres réseaux de transport.

3. Le réseau virtuel JXTA

Comme d'autre architectures pair-à-pair, JXTA crée un réseau virtuel appelé OVERLAY, au dessus des autres réseaux issus du monde IP, permettant aux pairs d'interagir et de s'organiser indépendamment de leur mode de connexion réseau. Un exemple d'overlay est présenté figure (3.1).

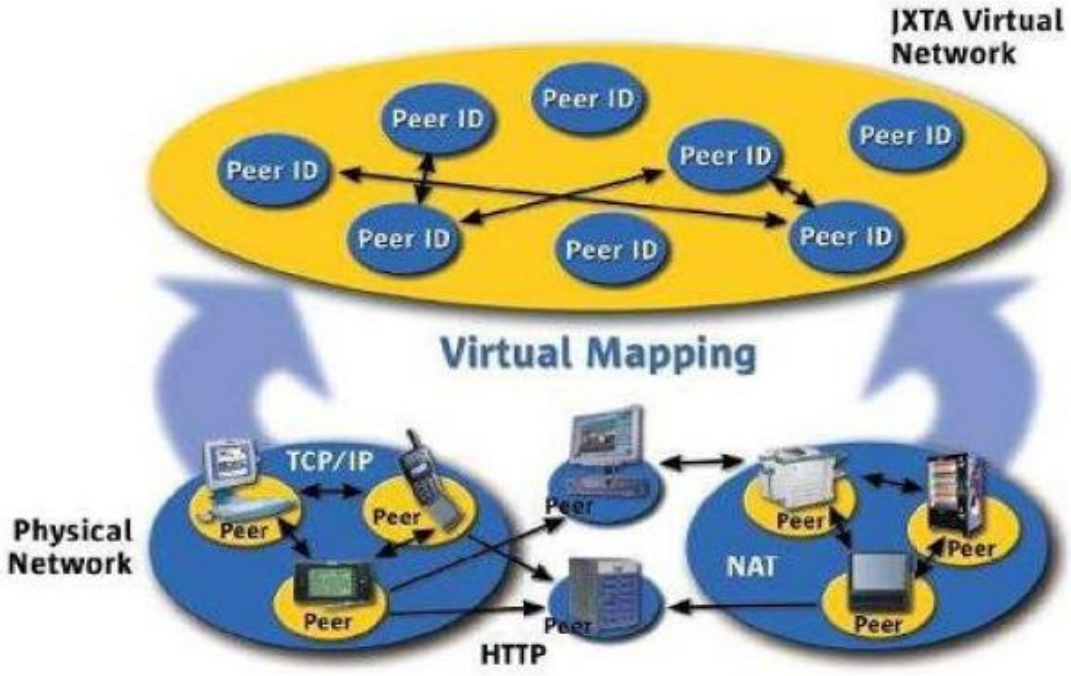


Figure3.1 : le réseau virtuel JXTA

4. Architecture logicielle

Pour permettre à ces protocoles de travailler ensemble et de former un système complet, une architecture doit être mise en place. JXTA est basé sur un modèle en couche, ce qui lui apporte de nombreux avantages chaque couche apporte ainsi ses services, permettant aux couches supérieures de les utiliser.

L'architecture logicielle du projet JXTA se divise en trois couches, comme il est illustré dans la figure 4.2.

4.1 Le noyau (Core) JXTA :

La couche plate-forme, également connu comme le noyau JXTA, encapsule primitives minimal et essentiels qui sont communs aux réseaux P2P. Elle comprend les blocs de

construction afin de permettre un mécanisme clé pour les applications pair à pair, y compris la découverte, le transport, la création des pairs et des groupes de pairs, et des primitives de sécurité associées. [Haj 03]

4.2 La couche service :

La couche service inclut des services réseau qui ne sont pas nécessaires mais très utiles (programmation très rapide, efficacité accrue). Exemples de services de réseau comprend la recherche et l'indexation, les systèmes de stockage, le partage de fichiers, la conversion de protocoles et d'authentification. [Haj 03]

4.3 La couche application :

La couche application comprend l'implémentation d'applications intégrées, comme la messagerie instantanée P2P, le partage des documents et des ressources, et beaucoup d'autres applications de haut niveau. [Haj 03]

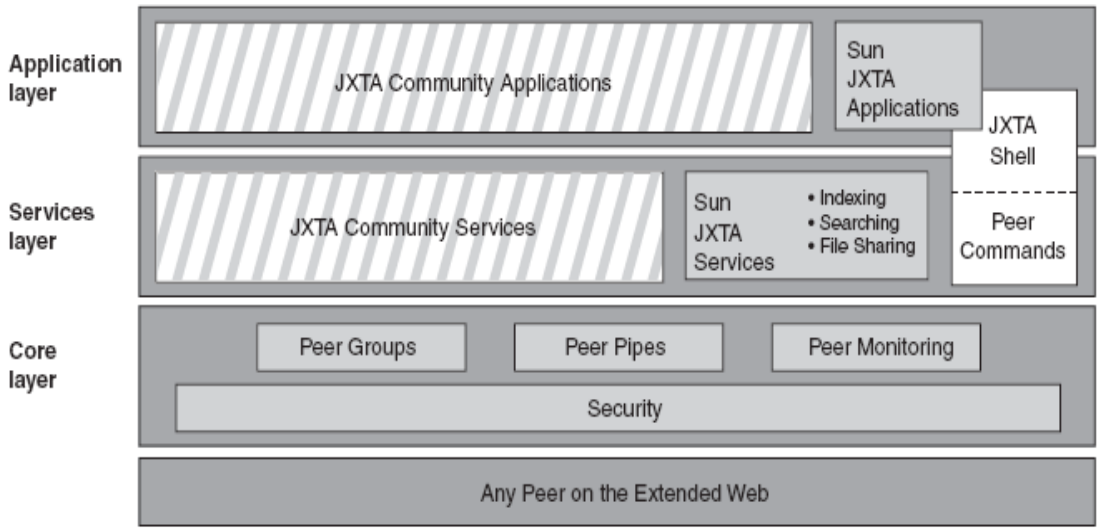


Figure 3.2 Architecture logiciel de JXTA. [Haj 03]

5. Les composants JXTA

Le réseau JXTA se compose d'une série de nœuds interconnectés, ou pairs. Les pairs peuvent s'auto organiser en groupes de pairs, qui fournissent un ensemble commun de services. Les pairs JXTA publié leurs services dans des documents XML appelé annonces. Les annonces permettre à d'autres pairs sur le réseau de savoir comment se connecter et interagir avec les services d'un pair. [Haj 03]

Pairs JXTA utilisent des pipes pour l'envoi des messages. Les pipes sont un mécanisme asynchrone et unidirectionnel de transfert de message utilisé pour la communication de

services. Les messages sont simples documents XML dont l'enveloppe contient le routage et les informations d'identification. Les pipes sont liés à des paramètres spécifiques, comme un port TCP et l'adresse IP associée. [Haj 03]

Trois aspects essentiels de l'architecture JXTA le distinguent des autres modèles de réseaux distribués:

- L'utilisation de documents XML (advertisements) pour décrire les ressources réseau.
- Abstraction des pipes à des pairs, et les pairs aux points de terminaison.
- Un schéma d'adressage uniforme par les pairs (peer ID).

Ces concepts sont décrits dans les sections suivantes.

6. Concepts de base

6.1 Pair (peer) :

Pair est l'unité structurelle de base de tout réseau pair-à-pair. Un pair peut être toute ressource capable de ce connecter au réseau et de fournir un service. JXTA peut distingue plusieurs types de pairs. [Ant 02]

- Pairs minimaux (Minimals Edges peers): Ces pairs sont réduits à des fonctionnalités élémentaires de communication, envoi et réception de message. Ils ne sont pas capables ni de stocker de l'information, ni de router les messages.
- Pairs simples (Edges peers): ces pairs peuvent envoyer et recevoir des messages et, en plus stocker dans un cache local des informations sur les ressources du réseau, sous forme d'annonces. Ces annonces leur permettent de répondre à des requêtes d'informations et de participer ainsi au protocole de découverte de ressources.
- Pairs de rendez-vous (Rendezvous Super Peer) : Ces pairs sont essentiels à la constitution du réseau. Ces pairs permettent de faire suivre les requêtes vers d'autres pairs connus (pairs simples ou pairs de rendez-vous).
- Pairs de routage (Relay Super Peer): Ces pairs stockent des informations de routage vers les autres pairs. En plus, ils permettent de transmettre les requêtes des pairs isolés du réseau par des mécanismes de pare-feu.

6.2 Groupe de pairs (peer group)

Peer group est un ensemble de peers qui partage les mêmes services et les mêmes spécifications dans un réseau. Chaque peer Group peut définir sa propre politique d'appartenance qui peut être ouverte (tout le monde peut joindre) ou hautement sécurisée (entreprend des mécanismes d'identification et des pièces de justification). Un peer Group est également identifié par un ID. Un peer peut être présent dans plusieurs groupes à la fois.

6.3 Service :

Les services sont des traitements fournis au sein d'un réseau pair-à-pair (recherche, partage, etc.). Ce sont eux qui motivent la constitution du réseau. On distingue deux types de services: des services individuels, fournis par des pairs individuels et des services de groupe, fournis par un group tous ses membres. Plusieurs membres du groupe peuvent contribuer, de manière redondante, à ce dernier type de service, qui reste disponible tant qu'au moins un membre du groupe reste connecté. [Ant 02]

6.4 Annonce (Advertisement) :

Toutes les ressources d'un réseau JXTA sont représentées par une annonce. Les peers, les peer Group, les services, les pipes,… sont tous représentés par une structure de métadonnées représentée sous forme de fichier XML et appelée advertisement (annonce). Un peer découvre une ressource dans le réseau en cherchant l'annonce correspondante. L'annonce d'une ressource est distribuée partout dans le réseau selon un mécanisme de publication. Chaque annonce est publiée avec une durée de vie permettant de spécifier le temps d'activité de la ressource associée.

JXTA distingue six types d'annonces :

- Annonce de pair.
- Annonce de groupe de pair.
- Annonce de pipe.
- Annonce de service.
- Annonce de contenu.
- Annonce de Endpoint.

6.5 Message :

Les applications et services de JXTA communiquent en utilisant les messages JXTA. Ces derniers sont l'unité de base de l'échange des données entre les peers.

Chaque protocole de JXTA est défini comme un ensemble de messages échangés par les peers.

6.6 Pipe :

Les peers utilisent les pipes pour envoyer et recevoir des messages. Une pipe représente un canal virtuel et unidirectionnel de communication et de transfert de données permettant de lier deux peers qui n'ont pas physiquement un lien direct. Le transfert des données sur le tube de communication est indépendant de toute topologie du réseau ou de la localisation physique des peers. Ils permettent d'envoyer tout type de données : XML, texte, image, vidéo et même les objets de java.

Il existe trois types de pipes :

- Unicast : unidirectionnel, non sécurisé et non fiable.
- Unicast secure : unidirectionnel, sécurisé et non fiable.
- Propagating : tube de propagation, non sécurisé et non fiable.

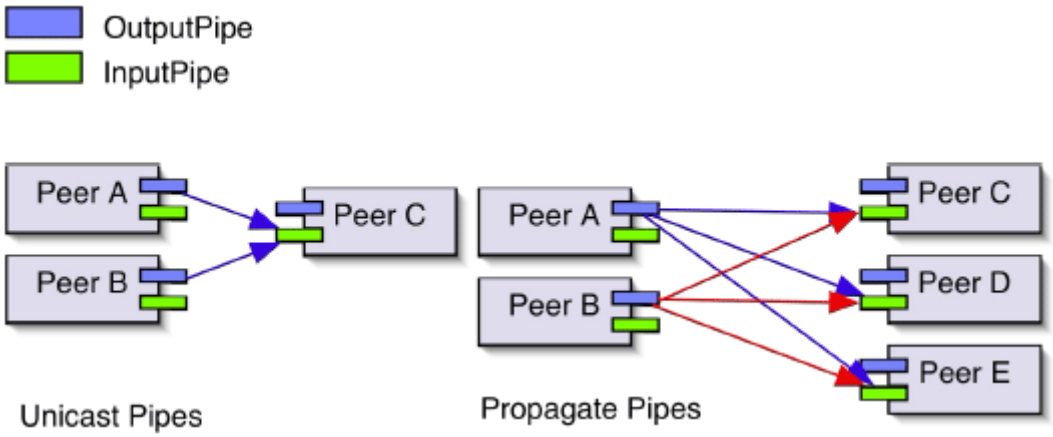


Figure 3.3. : pipes JXTA

6.7 Socket :

Les sockets de JXTA sont essentiellement basés sur les pipes. Ils représentent des canaux de communication fiables, sécurisés et bidirectionnels destinés au transfert de données.

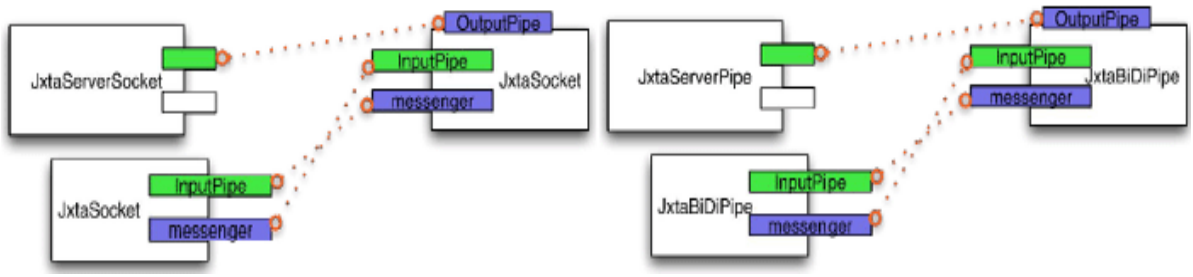


Figure 3.4. : Sockets JXTA

6.8 Module :

Ce sont soit des services, soit des applications.

6.9 IDs :

Il s'agit d'une chaîne de caractère unique, utilisée pour identifier six types de ressources : les peers, les groupes de peers, les tubes de communications, les contenus, les spécifications de modules et les classes de modules.

Ces identifiants peuvent en fait être considérés comme étant le système d'adressage de JXTA.

Un identifiant est composé de trois parties :

- un identifiant de namespace = jxta ;
- une spécification de format = urn ;
- un ID = valeur unique.

Exemple : urn:jxta:uuid

5961626164616261 4A78746150325033F3BC76FF13C2414CBC0AB663666DA53903

7. Les protocoles JXTA

JXTA est une spécification d'un ensemble de six protocoles, qui permettent le partage, la communication et la collaboration entre les peers, les six protocoles sont indépendants. Chaque protocole est mis en œuvre de chaque coté de la communication par un service, qui offre les primitives nécessaire à la réalisation de ce protocole. Ces protocoles sont basés sur l'échange de message XML.

7.1 Peer Discovery Protocol (PDP)

La localisation des ressources dans un réseau JXTA est basée sur la notion d'annonce. Ainsi, trouver un peer ou une autre ressource revient à découvrir l'annonce de celui-ci. Le *Peer Discovery Protocol* définit un ensemble de protocoles permettant de publier et de découvrir des annonces JXTA sur le réseau.

7.2 Peer Information Protocol (PIP)

JXTA inclut un Protocol nommé le *Peer Information Protocol* utilisé par un peer afin d'obtenir des informations sur l'état d'un autre peer. [Gra 02] Il s'agit d'un protocole optionnel dans tout type de liaison. En d'autres termes, JXTA ne pose aucune restriction sur un peer distant pour répondre aux messages PIP.

7.3 Peer Resolver Protocol (PRP)

Le PRP est une suite de messages génériques (requêtes et réponses) écrits dans le but de créer un système de messagerie commun entre les peers dans un réseau JXTA. [Gra 02] Il permet à un peer d'envoyer une requête générique vers un ou plusieurs peers et de recevoir une ou plusieurs réponses. La spécification de ce protocole permet aux services de JXTA d'échanger tout type d'information.

7.4 Peer Binding Protocol (PBP)

Le PBP est utilisé afin d'établir des canaux et des pipes virtuels de communication entre deux ou plusieurs peers. [Gra 02] Le PBP est responsable d'établir la liaison entre un pipe et sa localisation physique (adresse IP,…).

7.5 Endpoint Routing Protocol (ERP)

L'ERP est utilisé par un peer afin de trouver des chemins (routes) vers des ports spécifiques du peer de destination. [Haj 03] Les informations sur les routes incluent des séquences ordonnées de peer IDs qui peuvent être sollicités afin d'envoyer le message à sa destination finale. Il fournit une façade permettant à deux peers qui n'ont pas de connexion directe d'apparaitre comme étant directement connectés.

7.6 Rendez-vous Protocol (RVP)

Le rendez-vous protocole a été conçu dans le but de propager les messages entre les peers à travers un peer group. Il est basé sur le concept de rendez-vous Peer. [Haj 03] Un rendez-vous peer est tout simplement un peer qui propage tous les messages qu'il reçoit. Ce broadcaste de messages permet de partager des informations concernant les ressources du réseau.

Les deux protocoles PEP et PRP sont obligatoires pour la communication, ils forment les protocoles de la couche noyau. Les autres protocoles sont optionnels mais communs dans les applications, et forment les protocoles de la couches service.

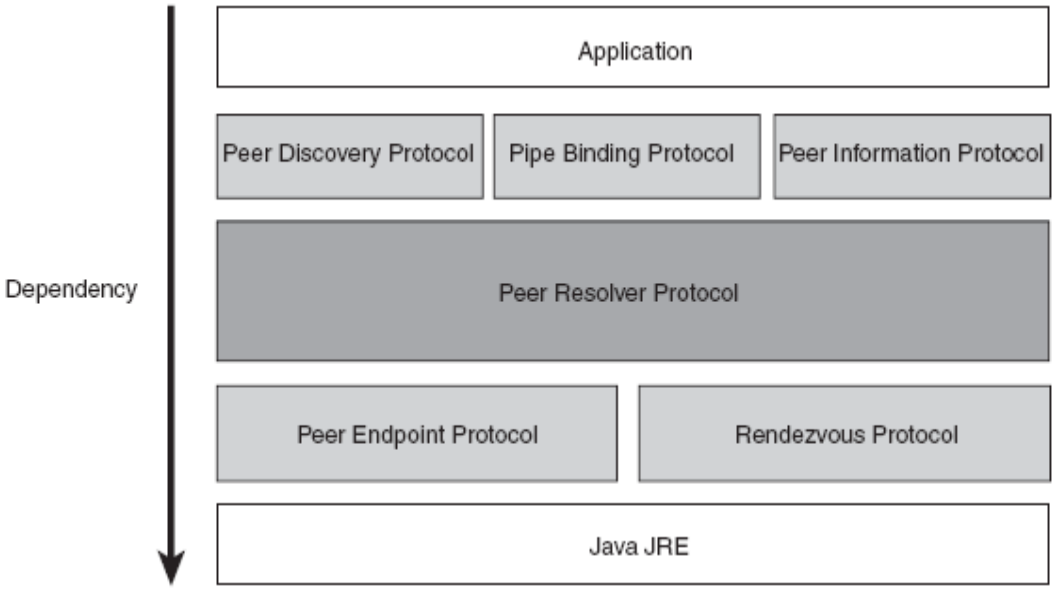


Figure 3.5 L'hiérarchique des protocoles JXTA [Haj 03]

8. Les avantages et les limites de JXTA

8.1Avantages

Comme dit précédemment, JXTA permet d'implémenter n'importe quel langage sur n'importe quel systèmes d'exploitations (ouverture vers le matériel 'non PC' comme les téléphones ……).

Tout en étant décentralisé pair-à-pair, un module centralisé (comme une application de type messagerie instantanée) pourra être facilement ajouté à une autre application (si les deux applications supportent les protocoles JXTA).

8.2 Limites

- JXTA est neuf, change rapidement et est faiblement documenté.
- Seul un début de noyau est aujourd'hui disponible.
- Il existe des incertitudes liées au potentiel d'évolution de la plate-forme et à son adoption par les utilisateurs développeurs.

9. Conclusion

Le framework JXTA définit une suite de protocoles désignés pour adresser les fonctionnalités communes requises pour permettre aux peers de former des réseaux robustes, envahissants et indépendants du système d'exploitation, langage de programmation ou protocoles de transport utilisés par chaque peer. JXTA a été développé d'une façon à viser les besoins du plus large ensemble d'applications pair à pair.

Grace aux caractéristiques et aux avantages offerts par la plateforme JXTA, la programmation des applications peer to peer fait des progrès significatifs.

Chapitre IV Implémentation D'un agent mobile dans un environnement pair à pair

Introduction

Après la modélisation des besoins puis l'organisation de la solution et passer par la littérature, ce chapitre vient, pour fournir une description de notre système.

1. Représentation de l'application

Notre réseau se compose de quatre hôtes, un hôte est le propriétaire de l'agent et les trois autres hôtes sont des magasins. L'agent migre à travers se réseau pour effectuer sa tache.

La tache dont nous avons choisi est un exemple classique de commerce électronique (Un agent mobile qui pourrait déterminer le meilleur prix pour un produit proposé par les hôtes magasins, en les consultant successivement, les un après les autres, puis en revenant sur l'ordinateur de l'utilisateur avec la meilleure offre). La première forme du e-commerce a été le magasin on-line ou les vendeurs communiquaient avec les consommateurs via leurs sites web en utilisant les méthodes de marketing traditionnelles.

- Le client crée le programme agent est le lancer dans le réseau en lui attribut une tache près défini qui est la recherche de produit au prix minimale et un itinéraire à suivre.
- L'agent se déplace dans le réseau d'un magasin à un autre en suivant son itinéraire, à chaque fois il arrive sur une machine, il accès à ses ressources effectue ses traitement, stocke son résultats, et migre à la prochaine destination.
- Une fois l'agent parcourir son itinéraire, il retourne à son emplacement initiale avec les résultats obtenus.

2. Diagramme de classes

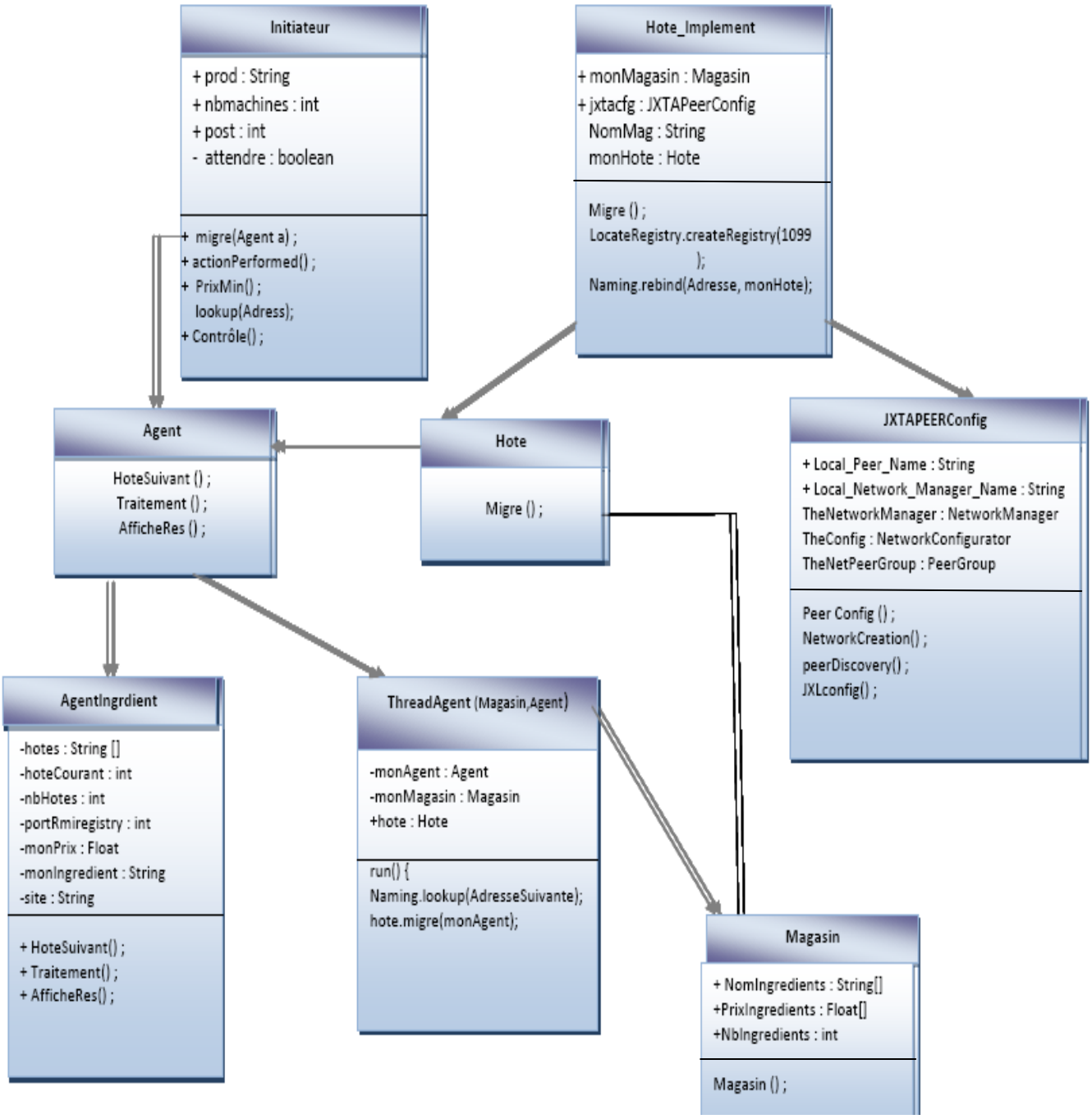


Figure 4.1. : Diagramme de classes

L'application se compose de huit classes

- **La structure du programme agent**

Notre agent mobile est composé de trois classes

La classe Agent

Cette classe est l'interface de notre objet déplaçable, sérializable qui est notre agent mobile.

La classe AgentIngredient

Constitue le corps de l'agent, elle contient les traitements (le code de l'agent) que l'agent doit accomplir. AgentIngredient fournit trois méthodes : *hoteSuivant ()*. Cette methode permet à l'agent d'indiquer à l'hôte qui l'héberge le site suivant a visiter. *afficheResultat()* qui affiche le résultat du traitement effectue par notre agent. *Traitement ()* qui consiste à chercher dans le magasin de l'hôte courant l'ingrédient et son prix.

Attributs		
Nom	**Type**	**Rôle**
hotes	String	La liste des hôtes à visiter
hoteCourant	int	indice permet a l'agent de savoir quel est le prochain hôte à visiter
nbHotes	int	
portRmiregistry	int	Donne le contexte de designation
monPrix	Float	Stocke le prix minimal trouve
monIngredient	String	Stocke le nom de l'ingredient à rechercher
site	String	Stocke le site sur lequel le prix minimum a été trouvé
Méthodes		
Nom	**Résultat retourné**	**Rôle**
hoteSuivant()	Liste des hôtes reste à visiter	Donne la prochaine destination
afficheResultat()		Stocker et afficher les résultats des traitements
Traitement()		Recherche le produit de prix minimal.

La classe threadAgent

Cette classe définit le thread qui va être lancé à chaque fois qu'un agent arrive sur un hôte. threadAgent est donc le support système offert par un hôte pour exécuter un agent.

- **La structure du programme Hote**

Le programme Hôte est constitué de deux classes :

La classe Hôte

Cette classe est une interface de l'objet reparti Hôte (qui représentera l'interface des pairs du réseau). Permet de représenter un poste de travail et une destination pour l'agent.

La classe HoteImplement

Contient les ressources recherchées par l'agent, elle définir aussi la méthode *migre()* qui est le point d'entrée de l'agent. Elle permet de déplacer l'agent à travers le réseau. La méthode *migre()* est invoquée par la classe ThreadAgent.

Attributs		
Nom	**Type**	**Rôle**
monMagasin	Magasin	Encapsule les informations sur les ingrédients
jxtacfg	JXTAPeerConfig	Crée pair JXTA
NomMag	String	Le contenu du magasin
monHote	Hote	Crée l'objet reparti et publication sur le réseau
Méthodes		
Nom	**Résulta retourné**	**Rôle**
Migre()		Permet la migration de l'agent

- **La structure du programme initiateur**

Il est constitué d'une seule classe, cette classe implémente l'entité qui va créer l'agent, l'envoyer sur le réseau et attendre son retour. Notez que cette classe est aussi une implantation de l'interface "Hôte", cette classe agit comme un initiateur pour l'agent et aussi comme un hôte (peer) en cas ou l'initiateur et aussi un magasin (cherchant des fournitures). Cette classe utilise des fonctionnalités JXL PeerMi tout en faisant appel à des fonctions de base du RMI classique.

- **Magasin**

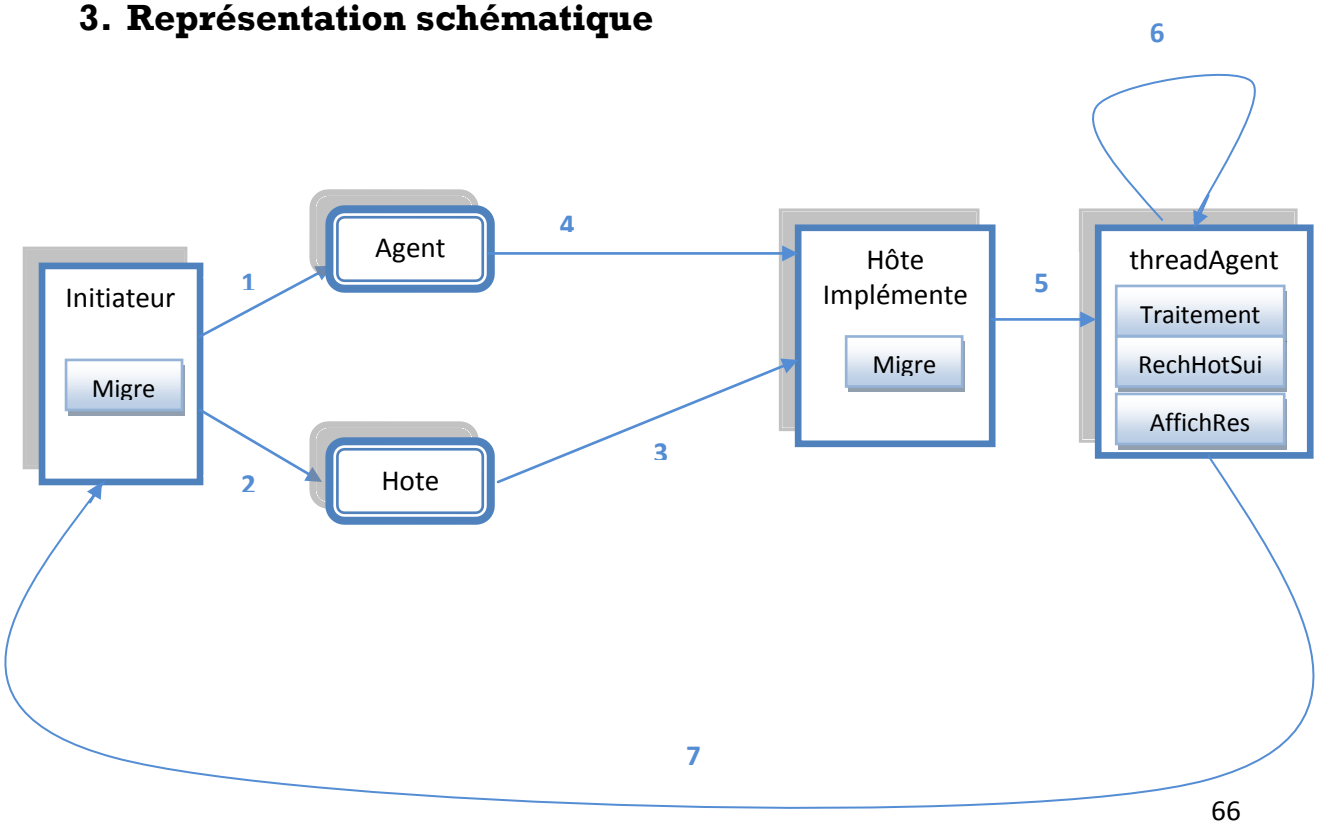
Cette classe contient les informations sur les ingrédients et leur prix.

- **JXTAPEERConfig**

Cette classe permet avant tout de créer un pair, créer des Advertisements et de lancer la découverte des voisins, mais avec l'apparition de la librairie JXL ceci est devenu plus simple, donc cette classe ne couvre plus que quelques fonctionnalité supplémentaires concernant le réseau P2P où l'agent doit se déplacer.

Attributs		
Nom	**Type**	**Rôle**
Local_Peer_Name	String	Donne le nom de pair
Local_Network_Manager_Name	String	Donne le nom du réseau local
TheNetworkManager	NetworkManager	Créer le réseau JXTA
TheConfig	NetworkConfigurator	Configuration du réseau
TheNetPeerGroup	PeerGroup	Crée le peer groupe
Méthodes		
Nom	**Résulta retourné**	**Rôle**
PeerConfig()		Configuration des pairs
NetworkCreation()		Crée et lance le réseau

3. Représentation schématique



1- l'initiateur crée une interface Agent pour lancer l'agent.
2- l'initiateur crée une interface Hôte pour faire déplacer cet agent.
3- L'interface Hôte fait appel à un objet distant HoteImplement (appel RMI)
4- L'exécution de la méthode migre() de HoteImplement permet de migrer l'agent vers la machine hôte sur laquelle elle s'exécute.
5- L'exécution de la méthode migre() consiste également à créer une instance de la classe threadAgent qui est le support de l'agent sur la machine hôte.
6- Quand l'agent arrive sur l'hôte destination, il exécute ses méthodes :
   - Traitement (rechercher le prix minimal)
   - Rechercher l'adresse de l'hote suivante
   - Effectuer un appel d'objet distant (appeler la méthode migre de HoteImplement suivante) pour faire déplacer l'agent
7- Le retour de l'agent : après la consultation de la liste des adresse demandée, de la même manière, l'agent fait appel à la méthode migre de l'initiateur pour que ce dernier lui récupéré avec les résultats obtenus.

Conclusion

Ce travail nous a permis d'aborder le domaine des réseaux pair à pair, et celui des agents mobiles.

Nous somme parvenu à faire fonctionner un agent mobile dans un environnement pair à pair, et comme nous avons mentionné précédemment, notre but est l'intégration de la technologie des agents mobiles dans les réseaux pair à pair.

La technologie des agents mobiles est intéressante mais très exigeante.

La plateforme JXTA a géré le problème de l'hétérogénéité mais elle montre des insuffisances coté mobilité, nous été imposées de travailler avec la technique RMI de l'API JXL.

Conclusion générale Et Perspectives

Conclusion générale et Perspectives

Ce travail nous a permis d'aborder le domaine des réseaux pair à pair, et celui des agents mobiles.

L'objectif de notre projet était de concevoir et de faire fonctionner un agent mobile dans un environnement pair à pair, dont le principe et de faire migrer l'agent de son pair origine vers un pair distant, effectuer sa tache et retourner des résultats.

Afin d'atteindre ce but nous avons commencé par étudier la technologie du pair à pair et les différents systèmes et architectures existants. Ensuite nous avons étudié la technologie agents mobiles, et on a vu les différents travaux antérieurs et récents concernant cette technologie. On a finalisé cette étape par une étude de la plateforme JXTA, ses concepts et ses différents protocoles de découverte, de routage,….qui sont armés pour réaliser un système pair à pair robuste.

Compte tenu de l'étude bibliographique effectuée et des objectifs fixés au début nous avons abouti à l'implémentation d'un agent mobile dans un réseau pair à pair, un réseau décentralisé, auto organisé et la communication entre les pairs se fait d'une manière automatique et dynamique.

L'hétérogénéité, est toujours un problème rencontré par la migration des agents, le pair à pair via JXTA et grâce à ses caractéristiques et ses protocoles arrive à géré parfaitement ce problème.

La sécurité à son tour présente un vrai obstacle, au niveau d'agent et au niveau pair, le JXTA grâce à ses techniques de sécurité intégrés (authentification, menbership, cryptographie et TLS) aide à palier cet obstacle.

Dans notre application, les API de la plateforme JXTA et JXL, ont facilité l'implémentation, la migration et la découverte de l'agent mobile et des pairs du réseau.

Après avoir implémenter un agent mobile en P2P, une première perspective est évidemment d'avoir plusieurs agents qui coopèrent et collaborent entre eux dans un environnement pair à pair. Par exemple avoir un agent négociateur qui prend en charge la négociation sur le prix du l'article, et un agent acheteur qui s'occupe de paiement et de la livraison.

La deuxième perspective est d'intégrer les agents mobiles dans d'autres scénarios d'applications pair à pair, qui peuvent fonctionner beaucoup mieux en utilisant les agents mobiles, par exemple utilisation d'un agent mobile dans les applications de partage de fichiers, ou la découverte et le téléchargement de fichier peut être réalisé par l'agent. On a

aussi l'utilisation des agents mobiles dans le calcul distribué, l'agent peut être utilisé pour gérer le calcul et envoyer les données pour être traiter dans les pairs distants.

L'intégration des agents mobiles dans les réseaux pair à pair, forme une tendance intéressante dans le domaine de l'informatique distribuée.

Bibliographie

Bibliographie

[And 04] S. Androutsellis-Theotokis et D. Spinellis, A Survey of Peer-to- Peer Content Distribution Technologies, ACM Computing Surveys, Décembre 2004.

[Ant 02] G. Antoniu, L. Bougé, T. Priol, Partage de mémoire à très grande échelle sur des réseaux pair-à-pair, rapport de recherche. INRIA institut national de recherche en informatique et en automatique. Mars 2002. Mars 2002.

[Bag 03] F. Bagci, J. Petzold, W. Trumler, and T. Ungerer, Ubiquitous Mobile Agent System in a P2P-Network, University of Augsburg, Germany, 2003. Article.

[Bro 86] R. A. Brooks, (1986). A robust layered control system for a mobile robot. IEEE Journal of Robotics and Automation.

[Bar 03] A. M. Barika, Vers un IDS Intelligent à base d'agents mobiles. 2003. Thèse doctorat.

[Bas 04] S. A, Baset, et H. Schulzrinne. (2004). An analysis of the skype peer-to-peer internel telephony protocol. Rapport technique, Departement of Computer Science, Columbia University.

[Ber 09] M. Bernichi, Surveillance logicielle à base d'une communauté d'agents mobiles, doctorat en informatique, université Paris 12 Val de Marne, 2009.

[Bra 05] P. Braun, and W. Rossak. Mobile Agents Basic Concepts, Mobility Models, and the Tracy Toolkit. San Diego. Elsevier Inc. (USA) and dpunkt.verlag (Germany), 2005. Article.

[Cai 07] G. Caire (TILAB, formerly CSELT). Jade Tutorial, Jade Programming for Beginner.2007. Livre.

[Che 02] R. Y. Chen, B. Yeager, Java Mobile Agents on Project JXTA Peer-to-Peer Platform, Sun Microsystems, 2002. Article.

[Cla 00] I.Clarke, O. Sandberg, B.Wiley et T. W. Hong. (2000). Freenet :
A distributed anonymous information storage and retrieval system. In Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA.

[Cub 05] C. Cubat Dit Cros. Agents Mobiles Coopérants pour les Environnements Dynamiques.2005. Thèse doctorat, Institut National Polytechnique de Toulouse.

[Das, 98] P.Dasgupta, N.Narasimhan, L.E.Moser, and P.M.Milliar Smith, A Suppler-Driven Electronic Marketplace Using Mobile Agents, university of California, Santa Barbara.

[Dij 06] A. Dijoux, E. Samuel, Peer-To-Peer, Master Professionnel Système informatique et réseaux, Université Claude Bernard LYON 1. 2006.

[Dua 07] E. Duarte, Technologie JXTA, CNAM Paris 2007-2008, Base de données avancées - NFE204, Vendredi 14 décembre 2007.

[Esp 10] B. Espinasse, Université d'Aix-Marseille, Brève introduction aux agents logiciels, 2010. Support de cours, disponible sur : www.lsis.org/espinasseb/Supports/RIWS-2010/IntroAgents-2010-4p.pdf

[Fel 02] G. Feltin, G. Doyen, O. Festor; les protocoles Peer-to-Peer, leur utilisation et leur détection; LORIA 2002.

[Fer 95] J. Ferber. Les Systèmes Multi Agents: vers une intelligence collective.1995 Livre.

[FIP 02] Foundation for Intelligent Physical Agents (www.fipa.org). FIPA Abstract architecture Specification, December 2002.

[Flo 02] A. M. Florea, Agents et systèmes multi-agents, University of Bucharest – 2002. Support de cours, disponible sur le site : http://turing.cs.pub.ro/auf2/

[Gra 02] J. D. Gradecki, Mastering JXTA Building Java Peer-to-Peer Applications. Livre.

[Gue 01] Z.Guessoum, & M. Occello. "Environnements de développement. Dans Principes et architectures des systèmes multi-agents", Hermès, Lavoisier, 2001.

[Gui, 02] V. Guilloux, agents électroniques, prise de décision en commerce électronique : synthèse et pistes de recherche, Université Paris, 2002.

[Hac 08] S. Hacini, Sécurité des Systèmes d'Information : Mise en œuvre de la confiance et de l'adaptabilité pour la protection de l'agent mobile, doctorat en science, Université Mentouri – Constantine, faculté des Sciences de l'Ingénieur – Département d'Informatique, 2008.

[Haj 03] S.H. Hajamohideen, A Model for Web Service Discovery and Invocation in JXTA, Information and Media Technologies, mars 2003. Livre.

[Hon 07] L. Hong. Architectural Design of Multi-Agent Systems: technology and techniques. Harshy New York : Information Science Reference, 2007.

[Jef 06] Lu, Ma, and J. P. Jeffrey. Tsai. Security Modeling and Analysis of Mobile Agent Systems. Londre : Imperial college press, 2006. Article.

[Jen 98] N. Jennings, K. Sycara, M. Wooldridge. A roadmap of agent research and development, Autonomous Agents and multi-agent Systems, Kluwer Academic Publishers, Boston. Manufactured in The Netherlands. 1998.

[Jha 99] R. Jha, Mobile Agents for e-commerce, KR School of information Technology-Indian Institue of Technology, Bombay, 1999.

[Jpg 07] JXTA Java™Standard Edition v2.5: Programmers GuideSeptember 2007. Livre.

[Kra 02] S. Krasinsky, Project JXTA Overview. 2002. Support de cours disponible sur le site : www.mnlab.cs.depaul.edu/seminar/.../JXTAOverview.pdf

[Lou 10] M. Loulou, Approche Formelle pour la Spécification, la Vérification et l'Imposition des politiques de Sécurité Dynamiques dans les Systèmes à base d'Agents Mobiles, thèse de doctorat en informatique de l'université Sfax en cotutelle avec l'université de Bordeaux 1, 2010.

[Maa 98] Z. Maamar, Aperçu général sur la technologie des agents mobiles, University of Waterloo, Waterloo, Ontario, Canada, 1998.

[Mar 06] P. Marlier, Sécurité du Peer-to-Peer, LABO (www.labo-asso.com), 2006. Article.

[Men 04] D.E. Menacer, Un modèle d'architecture à base d'agents mobiles pour les applications réparties. 2004. Thèse doctorat.

[Men 04] D.E. Menacer, Un modèle d'architecture à base d'agents mobiles pour les applications réparties. 2004. Thèse doctorat.

[Mil 98] D. Milojicic, M. Breugst, I. Busse, J. Campbell, S. Covaci, B. Friedman, K. Kosaka, D. Lange, K. Ono, M. Oshima, C. Tham, S. Virdhagriswaran, and J. White. MASIF: The OMG Mobile Agent System Interoperability Facility. In K. Rothermel and F. Hohl, editors, Proceedings of the 2nd International Workshop on Mobile Agents, Heidelberg, Germany, 1998.

[Mil 02] S. D. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins et Z. Xu, Peer to Peer computing, Rapport technique 57, Laboratoires Hewlett-Packard, Palo Alto, Californie, Mars 2002.

[Pan 08] D. Panzoli, Proposition de l'architecture « Cortexionist » pour l'intelligence comportementale de créatures artificielles, thèse doctorat de l'université de Toulouse. 2008.

[Per 97] S. Perret, « Agents mobile pour l'accès nomade à l'information répartie dans les réseaux de grande envergure », Thèse pour obtenir le titre de docteur de l'université Joseph Fourrier – Grenoble I, Novembre 1997.

[Rhe 03] S. C. Rhea, P. R. Eaton, D. Geels, H. Weatherspoon, B. Y. Zhao et J. D. Kubiatowicz. (2003). Pond : the oceanstore prototype. In the 2nd USENIX conference on File And Storage Technologies, San Francisco, CA, USA.

[Sch 01] R. D. Schollmeier, A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications, Institute of Communication Networks, Germany 2001.

[Soy 05] O. Soyez, Stockage dans les systèmes pair à pair, doctorat en informatique, l'Université de Picardie Jules Verne d'Amiens, 2005.

[Sun 00] Sun Microsystems et Artima.com. The Jini User Interface Specification, Avril 2000.

[Pie 03] S. Pierre, Réseaux et Systèmes Informatiques Mobiles : Fondements, Architecture. Livre

[Tie] Tie-Yan Li, Zhi-Gang Zhao & Si-Zhen You, A-peer: An Agent Platform Integrating Peer-to-Peer Network, Institute for Infocomm Research, Singapore. Article.

[Wai 07] A. Wai-Sing Loo, Peer to Peer Computing: Building Supercomputers with WebTechnologie, 2007. Livre.

Sites Web

[eMule, 2002] http://www.emule-project.net/

[jxta@] http://jxta.dev.java.net/

[Jxl @] http://jexcelapi.sourceforge.net/

[Net @] http://technet.microsoft.com

[Napster, 1999] http://www.napster.com.

[Objectspace @] http://www.ObjectSpace.com

[skyp@] http://www.skype.com/

[Seti@home, 1997] http://setiathome.ssl.berkeley.edu/