République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE d'ORAN Mohamed Boudiaf

Faculté des Sciences
Département d'Informatique
Spécialité : *Informatique* ***Option :*** Systèmes, Réseaux et Bases de Données

MEMOIRE
Présenté par

Mr. LEKHAL Ilies Ali

Pour l'obtention du diplôme de Magister en Informatique
Thème

UN SYSTEME INTELLIGENT POUR LA LOCALISATION DE RESSOURCES DANS LES RESEAUX P2P

SOUTENU LE (24/10/2012, 10H, Salle de Soutenance).

Devant la commission d'examen composée de :

Qualité	Nom et Prénoms	Grade	Etb d'origine
PRESIDENT	**Mr M. BENYETTOU**	**PROFESSEUR**	**USTO**
RAPPORTEUR	**Mme R. MEKKI**	**M.CONF. A**	**USTO**
EXAMINATEUR	**Mme H. FIZAZI**	**M.CONF. A**	**USTO**
EXAMINATEUR	**Mr B. MESSABIH**	**M.CONF. A**	**USTO**
EXAMINATEUR	**Mme M. NOUREDDINE**	**M.CONF. A**	**USTO**

Année universitaire : 2011-2012

DEDICACES

« Louange à Dieu, le tout puissant »

A mes très chers parents,

Pour leurs soutiens permanents et inépuisables,

que Dieu les protège !

A mes très chers frères et sœur
Que j'adore

A mes adorables neveu et nièce que
j'aime tant

A mes grands parents

Et à tous ceux qui me sont chers,

A TOUS, je dédie ce travail en leur adressant tous mes sentiments

D'affection et de considération !

Ilies Ali

REMERCIEMENTS

*Je ne saurai remercier assez **Dr. R. MEKKI**, pour m'avoir proposé ce sujet et avoir dirigé mes travaux, pour son constante disponibilité, pour les conseils qu'elle m'a cessé de me prodiguer et enfin pour son encouragement.*

*Que **Pr. M. BENYETTOU**, Professeur à l'USTO, trouve ici l'expression de ma profonde reconnaissance et de ma très haute considération pour l'honneur qu'il m'a fait en acceptant de présider le jury.*

*Tous mes remerciements accompagnés de ma gratitude vont aux membres du jury : **Dr. H. FIZAZI**, **Dr. B. MESSABIH**, et **Dr. M. NOUREDDINE** pour la 'célérité' avec laquelle ils ont examiné, corrigé et critiqué ce travail.*

Ma profonde reconnaissance va spécialement à mes professeurs pour les orientations, la patience et l'intérêt qu'ils nous ont toujours manifesté durant nos études en Magister.

Que les Pr. Serridi professeur à l'université de Guelma, M.Ammad professeur à l'université de Bejaia, M.Mahmoud professeur a l'université de Saida, H.Boukli Professeur à l'université de SBA, trouve ici l'expression de ma profonde reconnaissance pour l'honneur qu'ils m'ont fait en acceptant de m'orienter.

Mes profonds remerciements vont à mes parents, Ma sœur, mes chers frères et Le petit ange Abdelilah.Leur amour et leur foi m'ont accompagné tout au long de ce travail.

Mes sincères remerciements vont encore à :

- *Mme Zehri Abbassia, Ahmed Bouhalouane et Kamel Belhadj, à d'une part, pour n'avoir ménagé aucun effort pour me faciliter le bon déroulement de ce travail et d'autre part, pour leur constante compréhension et ses encouragements.*
- *Mes cher amis : mohamed ,zine el abidin, youb et à tous qui me sont chers.*

Enfin, remercier mes parents serait se répéter, citer leur affection serait un pléonasme, parfois pour exprimer plus que ce qu'on a envie de dire, on a recours au silence.

Table des matières

Introduction Générale

Première partie : Etat de l'art

Chapitre I : Réseaux Peer to Peer et techniques de localisation de ressources

Chapitre III : Web Sémantique & ontologies

Chapitre IV : Relation des technologies P2P, agents mobiles et ontologies

Deuxième partie : Approche proposée

Chapitre V : Architecture Proposée

Conclusion Générale et perspectives

Communications et publications

Références bibliographiques

Annexes

Liste des Figures

Liste des Tableaux

Introduction Générale

INTRODUCTION

1. Contexte général

L'informatique est en train de changer de manière assez profonde. Au départ, confinée dans les ordinateurs, elle devient de plus en plus diffuse et distribuée dans de multiples objets et fonctionnalités qui sont amenés à coopérer. La décentralisation est donc la règle et une organisation coopérative entre modules logiciels est un besoin. De plus, la taille, la complexité et l'évolutivité croissantes de ces nouvelles applications informatiques font qu'une vision centralisée, rigide et passive atteint ses limites.

Le modèle client/serveur, est actuellement le modèle incontournable pour le développement d'applications réparties.

Néanmoins, ce paradigme n'arrive plus à satisfaire les nouveaux besoins des applications distribuées justifiés par l'évolution actuelle d'Internet. Nous pensons que les services doivent être adaptés à chaque utilisateur : trouver le meilleur produit au meilleurs prix, trouver l'information pertinente en temps voulu, accéder au meilleur service et ce, au moindre coût. De plus, à l'ère de la mobilité et la communication nomade, le client/serveur impose des coûts qui freinent la généralisation de ces technologies au grand public.

Ces besoins se sont traduits par des recherches intensives autour des techniques de mobilité dans des environnements répartis afin de développer d'autres approches pouvant servir efficacement les besoins des utilisateurs et utiliser intelligemment les ressources du réseau. C'est ainsi que les réseaux peer to peer sont apparus.

Les systèmes « pair à pair » (peer-to-peer ou P2P) sont des systèmes répartis à l'échelle du réseau Internet. Ils reposent sur le principe de mutualisation (échange et partage), au sein d'un réseau logique, de services et de ressources comme par exemple des données, des programmes, des capacités de stockage ou de calcul. Tous les participants (appelés pairs) sont à égalité de devoirs et de droits : chacun peut à la fois rendre accessibles des ressources dont il dispose (opération de publication) et exploiter des ressources fournies par d'autres. En ce sens, le modèle P2P est une alternative au modèle client-serveur classique, les pairs pouvant jouer en même temps le rôle de serveur et le rôle de client. Les systèmes P2P sont par ailleurs des systèmes ouverts : les pairs peuvent librement intégrer et quitter le réseau, et fournir les ressources de manière intermittente et dans une forme susceptible d'évoluer au cours du temps.

Au-delà des problèmes légaux (voire moraux) que posent quelques applications phares permettant l'échange de musique et de films, le P2P constitue un véritable modèle d'organisation répartie extensible et robuste qui permet la mutualisation et la capitalisation au sein de communautés ou d'entreprises. En particulier, il semble d'un grand intérêt pour la construction de systèmes d'information répartis à grande échelle qui, par nature, sont hétérogènes et instables.

2. Problématique

Par nature, l'instabilité des systèmes P2P et la volatilité des ressources sont fortes et incontrôlées. Un système P2P robuste doit donc être flexible et capable de s'adapter dynamiquement, de manière transparente pour l'utilisateur. Du fait de l'échelle (la dimension du réseau et le nombre de pairs, typiquement des milliers, des dizaines de milliers, voire davantage), la communauté ne peut pas être informée des changements (évolutions, connexions, déconnexions, pannes). Ainsi, lorsqu'un pair veut accéder à une ressource, il n'a a priori qu'une connaissance partielle (qui peut être en partie obsolète) de l'état du système P2P (de la disponibilité des serveurs et de la qualité des services). La localisation des ressources est donc une fonctionnalité essentielle qui suscite une étude et un développement des techniques efficaces pour soulever les problèmes de complexités de ces systèmes.

3. Objectifs de la recherche

L'objectif principal de cette recherche consiste à la localisation des ressources dans les réseaux P2P pur vu la complexité la nature décentralisée et ouverte de ces systèmes. Une telle localisation doit prendre en compte les évolutions fréquentes de la structure du réseau ainsi que la volatilité des ressources et des services. Les approches classiques pour le développement ne permettent pas de répondre de manière satisfaisante à ces contraintes. Nous proposons dans ce travail une approche à base d'agents (autonomie, mobilité, adaptabilité) et le web sémantique dont le déploiement, l'adaptation dynamique et la localisation sont assurés par ces agents et une ontologie qui facilite leur communication. Après avoir étudié ces différentes technologies nous proposons un Framework en couches basé sur qui assure la localisation des ressources dans un système P2P pur. Enfin, nous présentons quelques éléments d'implantation dans ce framework et les outils utilisés.

4. Organisation du mémoire

Après la présentation du contexte général, de la problématique et des objectifs de la recherche; le présent rapport est organisé en deux grandes parties :

La première partie est un état de l'art des domaines concernés par notre étude. Elle est composée des quatre chapitres suivants :

Chapitre I : Il introduit le concept Réseaux Peer to Peer et les techniques de localisation de ressources

***Chapitre II* :** Ce chapitre sera dédié à la présentation des systèmes multi-agents et les agents mobiles.

Chapitre III : Dans ce chapitre nous allons aborder le Web Sémantique et les ontologies.

***Chapitre IV* :** Suite aux études présentatrices et critiques précédentes, ce chapitre va contenir une étude sur la relation des technologies P2P, agents mobiles et ontologies.

La ***deuxième partie*** expose l'architecture que nous préconisons pour assurer la localisation des ressources dans les systèmes P2P en se basant sur les technologies agents mobiles et le web sémantique. Elle est composée du chapitre suivant :

***Chapitre V* :** Analyse et conception d'un système de localisation des ressources dans un système P2P pur.

Enfin, ce document sera clôturé par une conclusion qui résume l'apport essentiel de l'architecture proposée. Elle ouvre également de nouveaux éléments de réflexion et quelques perspectives de recherche.

Première partie

Etat de l'art

Chapitre I

Réseaux Peer to Peer et techniques de localisation de ressources

Introduction

L'expression "réseau peer to peer" (P2P), que l'on traduit généralement par réseau de "poste à poste", "pair à pair" ou encore "d'égal à égal" désigne une architecture de réseau où les postes connectés communiquent directement entre eux et partagent leurs ressources. Tous les postes ont un rôle équivalent, à la fois client et serveur par rapport à ces ressources (espace de stockage, puissance de calcul...), d'où leur appellation de "servent" (contraction de serveur et client). C'est la mise en commun de ces ressources qui fait la force et le succès de ces réseaux [Saroiu, 2002] [Kubiak, 2007].

Ce type d'architecture qui est à la base d'Internet dès l'origine, a été largement médiatisé ces dernières années, suite aux péripéties judiciaires liées aux partages de fichiers protégés par un copyright.

Les applications vont du partage de films ou de fichiers musicaux pour le grand public au travail collaboratif ou au calcul distribué pour l'entreprise.

On estime que le trafic p2p représente 60% du trafic des réseaux haut débit le jour et 90% la nuit preuve de sa popularité.

L'objectif de ce chapitre est de présenter de manière générale le Peer-To-Peer.

La première partie situe le modèle peer to peer dans les systèmes informatiques, définit les objectifs et caractéristiques de ce modèle. La deuxième partie traite des différentes architectures sur lesquelles il repose, et pour chaque architecture un exemple de réseau existant. Enfin, la dernière partie développe différentes utilisations du peer to peer.

1. Le modèle P2P

1.1. Taxonomie des systèmes informatiques

Les systèmes informatiques peuvent être classés en deux grandes catégories, présentés dans la figure 1.1 : les systèmes centralisés reposant sur des mainframes et les systèmes distribués. Ces derniers peuvent être construits selon deux modèles : le modèle client/serveur plat ou hiérarchique et le modèle pair à pair qui peut être pur ou hybride.

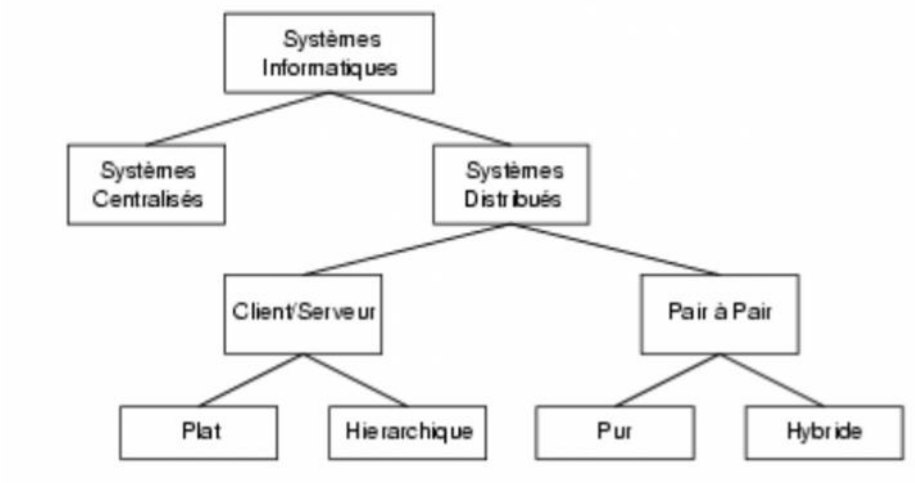

Figure 1.1 : *Classification des systèmes informatiques*

1.2. Topologie

Les réseaux peer to peer se greffent au réseau public qu'est internet. Ils présentent donc une topologie virtuelle (overlay network) (Fig 1.2), [Kubiak, 2007].

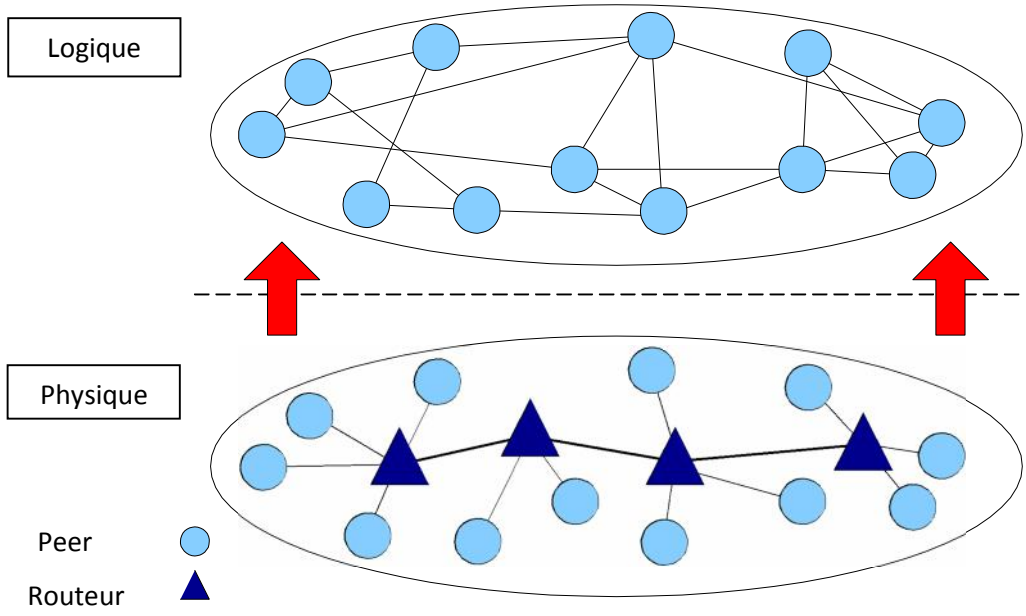

Figure 1.2 : *Topologie des réseaux P2P*

1.3. Objectifs

Le modèle P2P étant très général, des applications de nature très différente peuvent l'utiliser. En effet, les objectifs sont variés[Milojicic, 2002]:

- Partage et réduction des coûts entre les différents peers ;
- Fiabilité et passage à l'échelle, l'absence d'élément centralisé pour l'échange des données permet d'accroître la fiabilité en supprimant tout point central de panne et d'améliorer le passage à l'échelle en évitant les goulots d'étranglement ;
- Agrégation des ressources et interopérabilité, en mettant en commun des ressourcesvindividuelles comme de la puissance de calcul ou de l'espace de stockage ;
- Accroissement de l'autonomie en l'absence d'une autorité centrale, il est de la responsabilité de chacun de partager ou non des fichiers ;
- Anonymat pouvant être assuré par certaines applications, en utilisant par exemple desnalgorithmes de routage qui rendent quasiment impossible le pistage d'une requête ;
- Communication ad-hoc et collaborative.

1.4. Caractéristiques des systèmes P2P

Les principales caractéristiques des différents protocoles sont[Milojicic, 2002]:

- la localisation des fichiers dans un environnement distribué ;
- des index du réseau P2P ;
- la libre circulation des fichiers entre systèmes ;
- un mode de communication standard (TCP et http) ;
- des capacités de connexion variables suivant les modèles ;
- des peers non sûrs ;
- aucun peer n'a une vue globale du système.

1.5. P2P contre Client/Server

Conceptuellement, le P2P est une alternative au modèle client/serveur, qui est constitué d'un serveur ou d'un cluster de serveurs et de beaucoup de clients. Dans le modèle pur du P2P, il n'y a plus de serveur, tous les participants sont des peers.

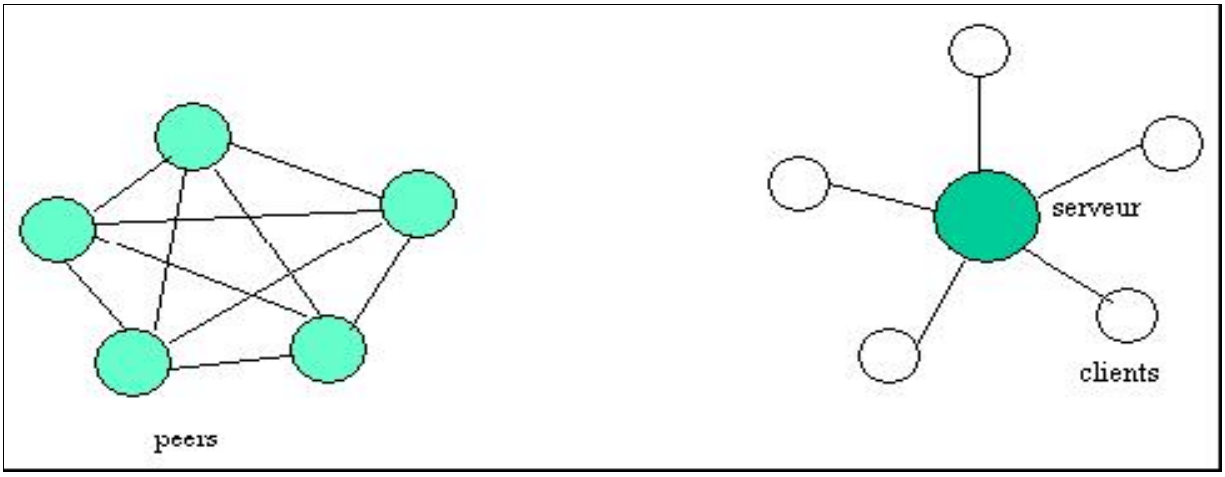

Figure 1.3 : *P2P versus Client/Serveur*

1.6. Comparaison

Peer-to-Peer	Client/Serveur
• Auto-organisé • Evolution dynamique, Ad-hoc • Découverte des peers • Flux distribué • Symétrie du réseau • Communication par Messages • Adressage dynamique au niveau application • Entités Autonomes • Attaques difficiles (mobilité , anonymat)	• Management centralisé • Configuré • Consultation de tables • Flux centralisé • Asymétrie du réseau • Orienté RPC • Adressage statique @IP • Entités dépendantes • Attaques plus simples

TAB1.1 Comparaison du Peer to Peer et architecture client/serveur

2. Les différentes architectures

Généralement, les systèmes peer to peer sont classés dans 2 grands modèles: les modèles purs et les modèles hybrides. Cependant, dans les modèles hybrides, on distingue deux architectures : une centralisée et une hybride entre le modèle centralisé et pur [Kubiak,2007].

2.1. Architecture centralisée

2.1.1. Présentation

La caractéristique de cette architecture est qu'elle repose sur un serveur central auquel se connectent les utilisateurs. Ce serveur est chargé de les mettre en relation directe. L'intérêt de cette technique réside dans l'indexation centralisée de tous les répertoires et intitulés de fichiers partagés par les abonnés sur le réseau [Schollmeier, 2002].

Lorsqu'un "peer" souhaite partager un fichier, il le déclare au serveur central. Celui-ci répertorie alors son adresse IP ainsi qu'un numéro de port donné par le client où il pourra être contacté pour un téléchargement.

Ainsi le serveur dispose principalement de deux types d'informations : celles sur le fichier (nom, taille, ...), et celles sur l'utilisateur (nom utilisé, IP, nombre de fichiers, type de connexion...

Si un utilisateur desire un fichier, il interroge l'index central du serveur qui lui indique alors la liste des postes (adresses IP) sur lesquels il est susceptible de le trouver. Interroger l'index se fait de la même manière qu'une recherche avec un moteur de recherche classique. L'utilisateur choisit alors parmi les réponses celle qui lui convient le mieux (pertinence de la réponse, bande passante, taille du fichier) et contacte directement le ou les postes choisis.

Le fichier en lui-même ne transite pas par le serveur central, qui fonctionne uniquement comme un annuaire.

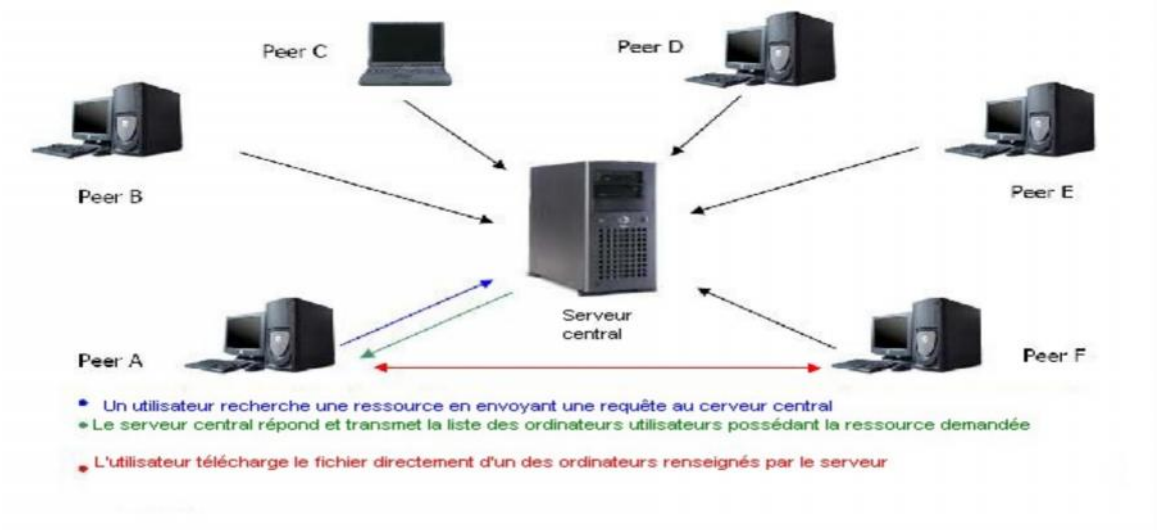

Figure 1.4 : *L'architecture centralisée*

2.1.2. Les avantage

- Avantages habituels d'un serveur central
- Facile à administrer
- Facile à contrôler
- Evite les recherches coûteuses sur le réseau
- Efficacité de la recherche par indexation centralisée
- Pas de routage
- Planification de la gestion des utilisateurs
- Tolérance aux fautes
- Par un sondage régulier des peers connectés, état cohérent

2.1.3. Les limites

- Pas d'anonymat partout
- Vous êtes connus du serveur
- et des peers sur lesquels vous téléchargez
- Limites habituelles d'un serveur central
- Réseau complètement dépendant du serveur
- Disponibilité
- Problème du passage à l'échelle :
- Saturation de la bande passante
- Saturation du nombreux processus
- Très facile de fermer le service car localisé
- Mauvaises informations du débit des peers pour ne pas être sollicités

2.1.4. L'Exemple du Réseau Napster

Le représentant le plus connu de ce mode de P2P est Napster. Il utilise un modèle P2P à répertoire centralisé pour les données administratives et les index des fichiers mp3 téléchargeables par les peers[Schollmeier, 2002].

Un peer se connecte directement sur le serveur. Le protocole basique est simple, avec des primitives de service : login request, login OK, login failed, user unknown, user not accepted, …

Le peer communique au serveur central la liste des fichiers qu'il partage, ainsi qu'un numéro de port TCP où il pourra être contacté pour un téléchargement. Le transfert de fichiers se fait selon les étapes citées précédemment.

2.1.5. Amélioration

Dans ce type d'architecture, il n'existe aucun recours dans le cas où le serveur tombe en panne ou se trouve encombré : le réseau se dématérialise.

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

Chaque serveur peut avoir accès aux informations des clients connectés sur les autres serveurs. L'accès aux données partagées est donc totalement transparent pour les utilisateurs.

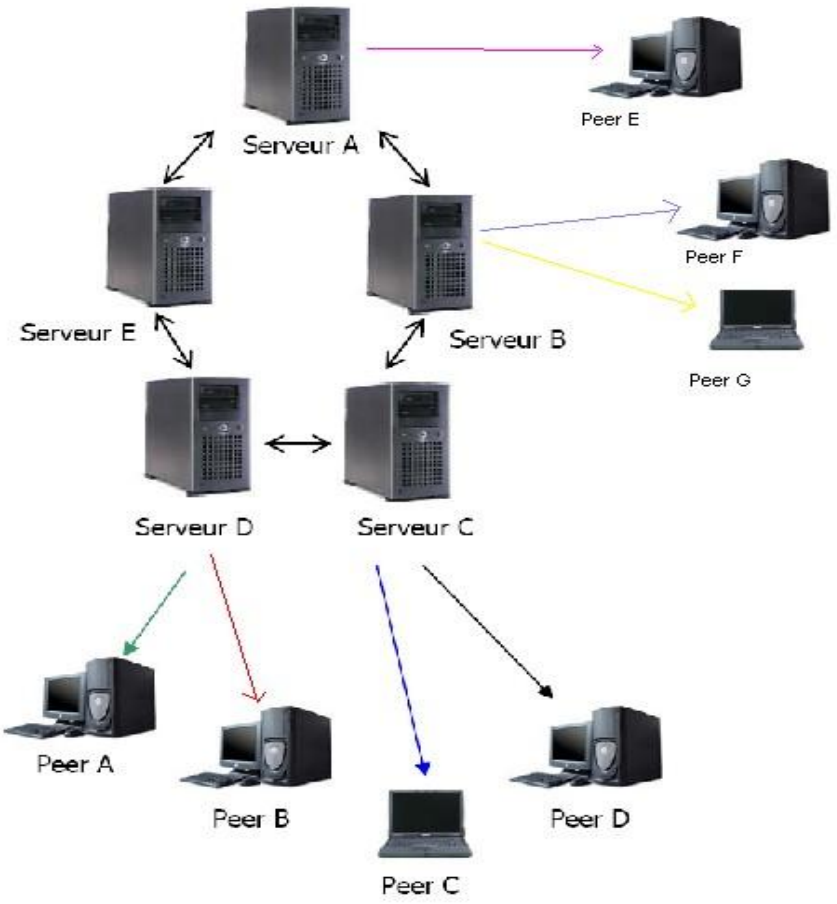

Figure 1.5: L'architecture *centralisée avec un anneau de serveur*

Du fait de la présence d'un ou de plusieurs serveurs, certains considèrent que ce modèle n'est pas entièrement p2p.

2.1.6. L'exemple du réseau eDonkey

Ce réseau, dont le but est de permettre le partage de gros fichiers multi-sources dispose de deux réseaux de pairs: les clients et les serveurs. Un serveur connaît la localisation des documents contenus chez les clients qui sont connectés à lui, alors que le client va stocker des morceaux de fichiers[Schollmeier, 2002].

Les fichiers

Les fichiers sont identifiés de manière unique grâce à l'algorithme MD4 sur les serveurs.

Afin de permettre un meilleur partage des fichiers, ceux-ci sont divisés en segments de 9 Mo chacun. Dès qu'un segment est complet, celui-ci est partagé par le client.

Ceci est permis grâce au protocole MFTP (Multisource File Transfert Protocol) qui s'exécute au niveau de la couche application.

Un dernier point intéressant est la propagation des sources entre les clients.

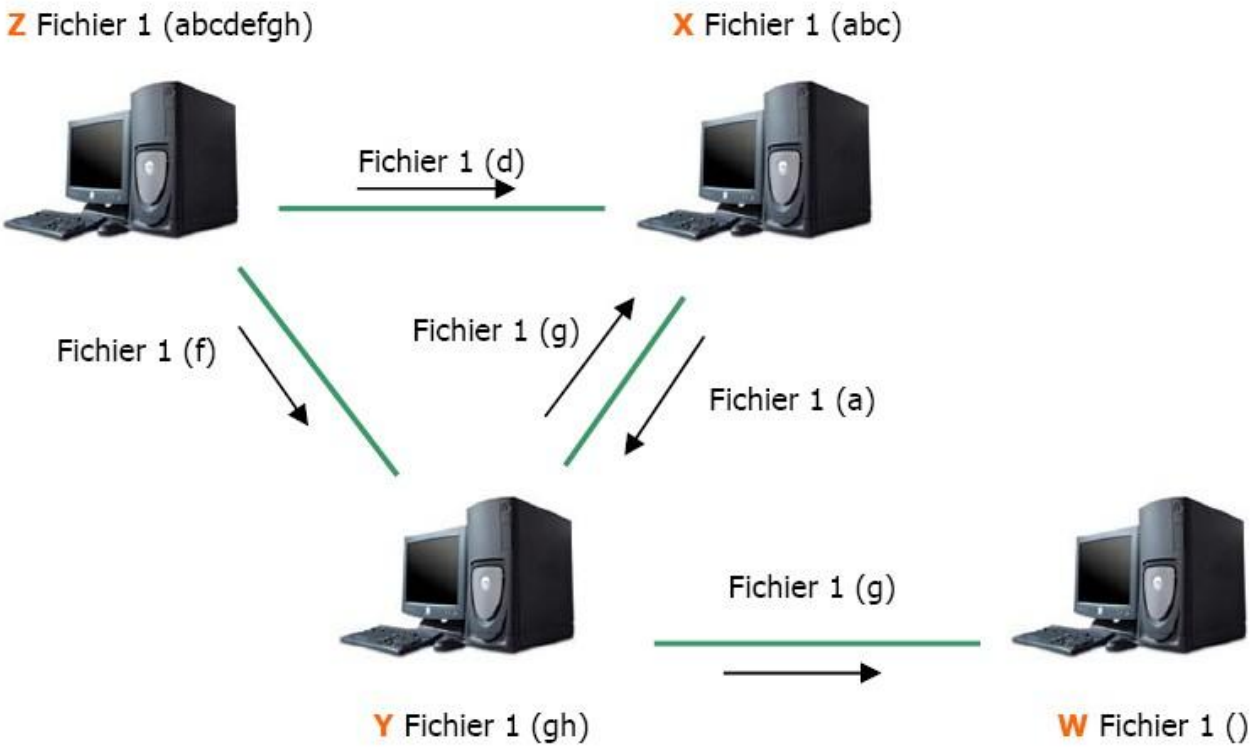

Figure 1.6 : *Le téléchargement partagé d'e-Donkey*

Ce schéma explique le principe du téléchargement partagé d'e-Donkey. Le client Z possède la totalité du fichier (les minuscules représentent les différentes parties du fichier). Les clients W, X, et Y veulent tous télécharger ce fichier. Puisque les clients X et Y ont tous les deux différents morceaux du fichier, ils peuvent non seulement télécharger ce fichier à partir de client Z mais ils peuvent aussi commencer à s'envoyer mutuellement ce qu'ils ont. Cela permet de

distribuer le plus rapidement possible ce fichier sans bloquer la bande passante de Z. En réalité cette bande passante sera bloquée (les clients X et Y vont télécharger au maximum de la vitesse) mais beaucoup moins longtemps (puisque certains morceaux seront arrivés d'autres sources).

De plus le client W peut commencer à charger ce qui lui manque chez les clients X et Y tant que Z n'a plus assez de bande passante.

En plus du MFTP, e-Donkey utilise un système de listes d'attente très ingénieux qui favorisent les personnes qui ont le plus de parties du fichier. Grâce à cela la diffusion se fait rapidement et plus il y a de sources qui disposent du fichier complet, plus la vitesse de téléchargement est rapide.

Point de vue du serveur :

Comme en mode client/serveur, les clients disposent ici de logiciels spécifiques leur permettant de se connecter explicitement au serveur.

Afin de faciliter le choix du client, le logiciel permet de voir plusieurs informations à propos du serveur, comme le nom ou le ping, mais aussi des informations plus spécifiques comme le nombre de clients maximum, le nombre de clients connectés, ou encore le nombre de fichiers partagés.

Les serveurs ne sont pas directement inter-reliés entre-eux. En revanche, les serveurs s'échangent, entre autre, la liste des serveurs connus.

En revanche, côté client, le fait d'être connecté à un unique serveur n'est pas aussi limitant qu'on pourrait le croire. En effet, chaque client demande des IP sources (c'est à dire les adresses IP des personnes ayant la ressource recherchée) à l'ensemble des serveurs qu'il répertorie. Cette possibilité d'interroger tous les serveurs peut aussi être utilisée lors d'une recherche. Un client ne publie la liste des fichiers qu'au serveur auquel il est connecté.

Le client :

A l'instar de la plupart des réseaux p2p cités ici, il n'existe pas un unique logiciel pour se connecter au réseau eDonkey. On peut par exemple citer eDonkey ou eMule.

2.2. Architecture décentralisée

2.2.1. Présentation

Nous avons vu que l'architecture centralisée pose des problèmes de sécurité, robustesse, et de limitation de la bande passante. Les problèmes sont directement issus de l'utilisation de serveurs dont le seul but est de posséder l'annuaire des clients[Kubiak,2007].

Si on désire supprimer les serveurs centraux, il faut donc trouver le moyen de constituer un annuaire sur chaque client, puis de les faire communiquer. C'est sur ces mécanismes que sont basés les réseaux Peer to Peer décentralisés. Il n'y a donc plus de serveurs centraux, ce sont tous les éléments du réseau qui vont jouer ce rôle. Chaque machine dans ses rôles est identique à une autre, c'est pour cela que l'on appelle ces types de réseaux pur peer to peer.

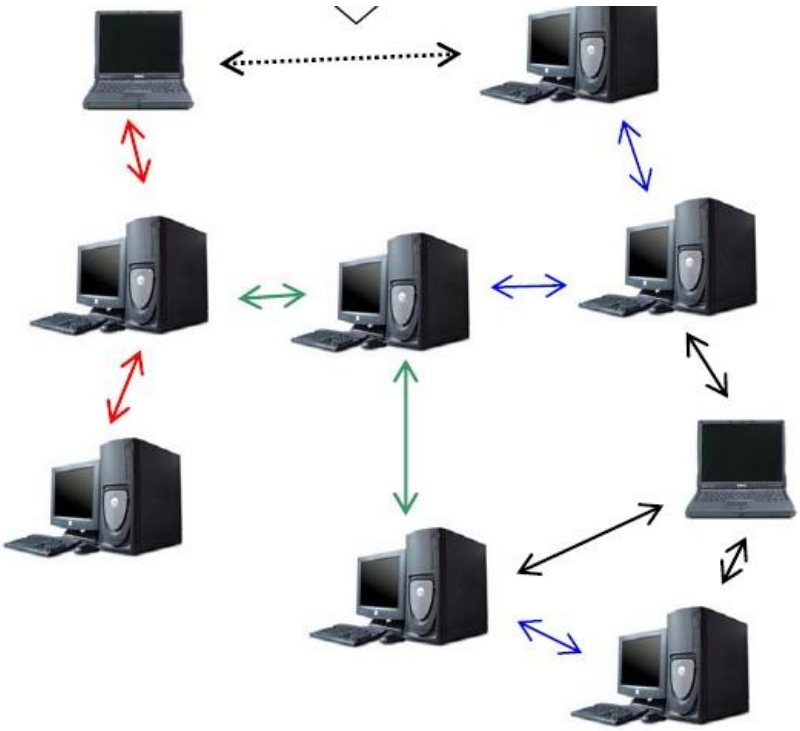

Figure 1.7 : *L'architecture décentralisée*

Lorsqu'un client souhaite se connecter, il va donc être nécessaire d'envoyer un message broadcast afin de savoir quelles autres personnes du réseau sont actives. Seules ces personnes répondront au message broadcast. On est alors connecté au réseau.

Afin de garder des informations cohérentes, un utilisateur n'est pas connecté directement à plus de 3 ou 4 noeuds, et connait tous les utilisateurs avec une profondeur d'arbre de 7

généralement. Chaque noeud a une vision limitée du réseau.

Pour obtenir une ressource, un poste transmet une requête à ses voisins qui font de même et ainsi de suite (notion d'inondation). Pour limiter la génération d'un nombre exponentiel de messages par ce procédé, on limite la propagation des messages jusqu'à un horizon, défini par un paramètre de durée de vie de la requête (TTL Time To Live), décrémenté à chaque poste. Une fois trouvé un poste possédant la ressource, une connexion directe s'établit entre les postes.

Ce mécanisme de recherche présente néanmoins une limite, une requête peut être stoppée par une expiration de TTL sans avoir parcouru l'intégralité du réseau et retourner une réponse négative.

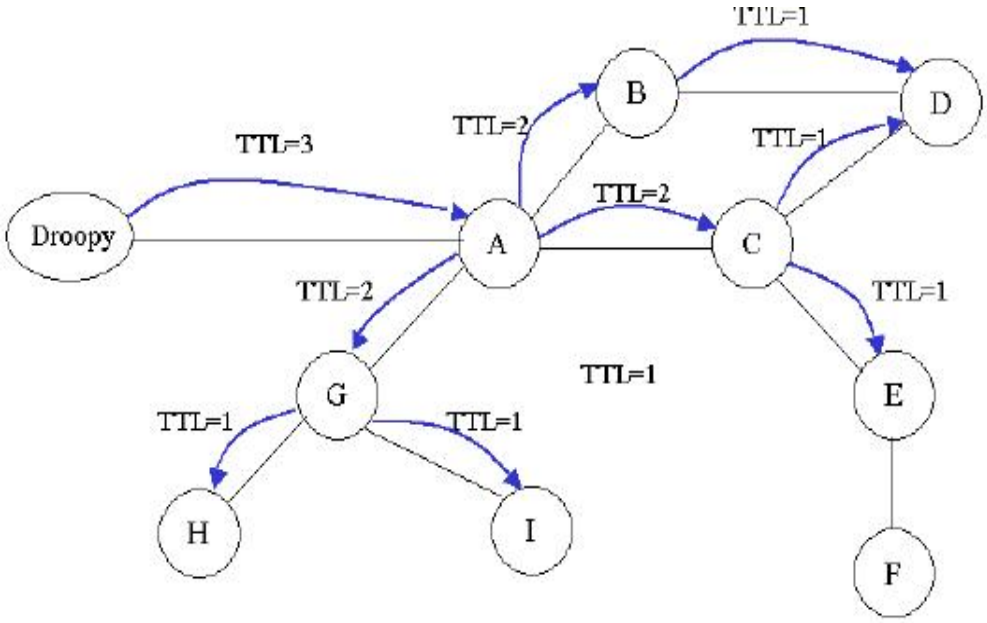

Figure 1.8 : *L'utilisation des TTL*

2.2.2. Avantages

- Administration simple et mutualisée
- Topologie évolutive (taille illimitée en théorie)
- Disponibilité du réseau, on ne peut l'arrêter.
- S'adapte bien à la dynamique du réseau (allées et venus des pairs)
- Total anonymat en théorie.

2.2.3. Les limites

- Gros consommateur de bande passante à cause des broadcasts envoyés.
- Pas de garantie de succès, ni d'estimation de la durée des requêtes (utilisation de TTL).

2.2.4. L'exemple du réseau Gnutella

Protocole de base:

Types	Description	Information
Ping	Annonce la disponibilité, et lance une recherche de pair	Vide
Pong	Réponse à un ping	IP et N° de port.Nombre et taille des fichiers partagés
Query	Requête	Bande passante minimum demandée Critères de recherche
QueryHit	Réponse à query si on possède la ressource	IP + N° de port + Bande Passante.Nombre de réponses + descripteur
Push	Demande de téléchargement pour les pairs derrière un firewall	IP du pair ; index du fichier demandé. Adresse IP + N° de port où envoyer le fichier

Ajout d'un nœud :

Nous supposons ici que le pair A souhaite se connecter au réseau Gnutella. Afin de rendre les schémas plus lisibles, nous limitons le nombre de pairs à 5 (et donc aussi le nombre de propagations). Pour se faire, il va tout d'abord émettre un message ping (en traits pleins rouges sur la figure).

Ici, celui-ci est d'abord reçu par B, qui va alors faire deux choses:

1. Répondre à A un message pong (en tirets noirs pour B, en "tirets point" orange pour D, "tirets point point" pour C et pointillés pour E).

2. Transférer le message de A vers les deux clients qu'il connaît, C et D. Ceux-ci répètent alors la même chose (sauf si le nombre de propagations a déjà été atteint).

A va donc connaître les nœuds B, C, D et E. On peut d'ailleurs remarquer que E connaît l'existence de A grâce à C et à D, mais qu'il ne répond qu'une fois (via le chemin le plus court, ou le premier qui l'a contacté).

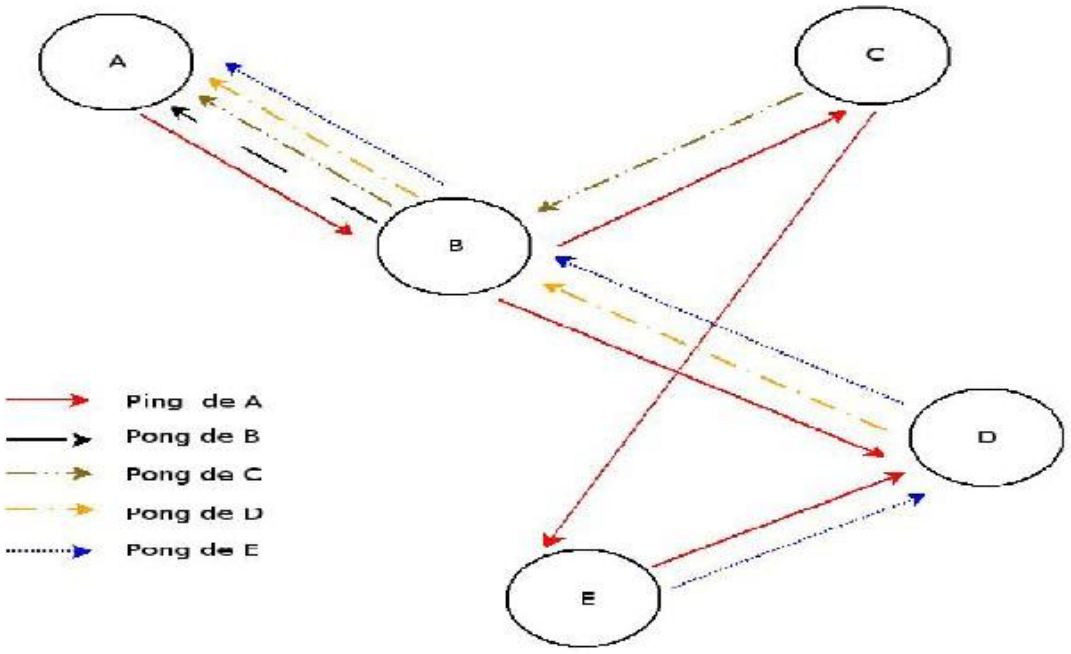

Figure 9 – *Ajout d'un noeud dans Gnutella*

Recherche :

Dans cet exemple, A cherche à obtenir le fichier a.txt, disponible chez C et E (ce que A ne sait pas bien évidemment).

A va donc envoyer un message Query (traits pleins rouges) au(x) premier(s) nœud(s) qu'il connaît (ici B), qui se charge:

1. de répondre s'il dispose de la ressource (ce qui n'est pas le cas)

2. de propager la question.

Les autres nœuds font de même, ce qui conduit C à répondre et à propager la requête. A peut alors envoyer à C le message get a.txt, et le transfert peut alors commencer.

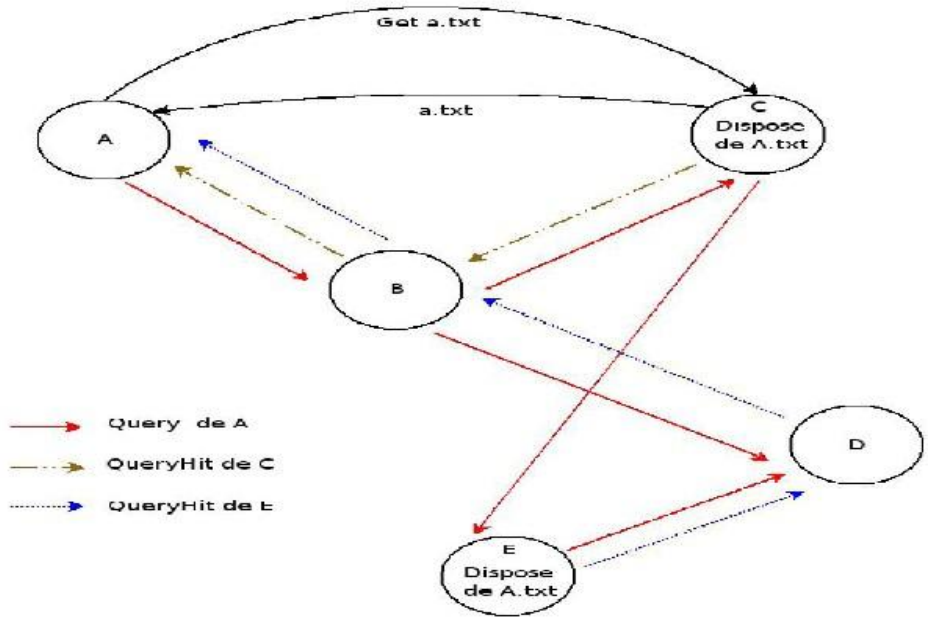

Figure 1.10 : *Exemple de recherche dans Gnutella*

L'échange de fichier

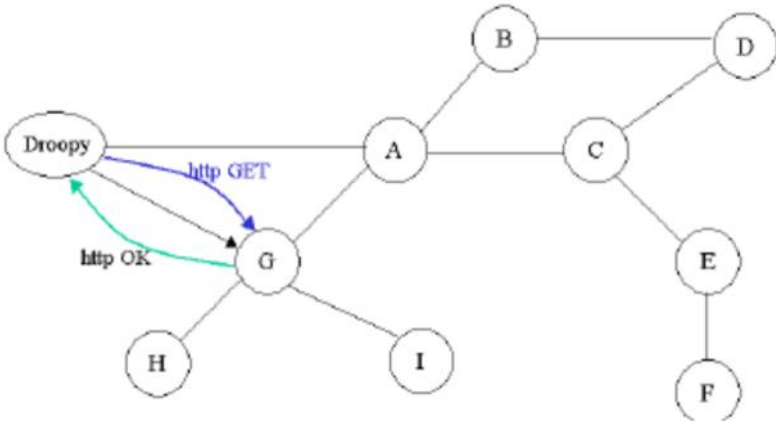

Figure 1.11 : *Exemple d'échange de fichiers dans Gnutella*

Observation de la Bande passante :

- Les requêtes occupent 25% du trafic, les ping 50%.
- En moyenne, un pair est connecté activement à 3 autres.
- Du point de vue de la bande passante, Gnutella ne passe pas à l'échelle (inondation).

2.3. Architectures hybrides

2.3.1. Présentation

Il s'agit ici d'un modèle que l'on pourrait qualifier d'hybride entre le modèle centralisé et pur. En effet, dans ce type de réseau il existe des machines particulières jouant le rôle de serveur local appelé "super node" ou "super peer" gérant un groupe de postes[Kubiak,2007].

Ainsi tous les noeuds ne sont plus égaux :

- Les noeuds disposant d'une bonne bande passante, de bonne performance CPU et d'une bonne accessibilité (temps de présence sur le réseau) sont organisés suivant le modèle décentralisé. Ce sont les super-peers.
- Les noeuds avec une faible bande passante sont reliés en mode client/serveur à un super-peer.
- Les super-peers disposent d'un index des ressources de leur cluster.

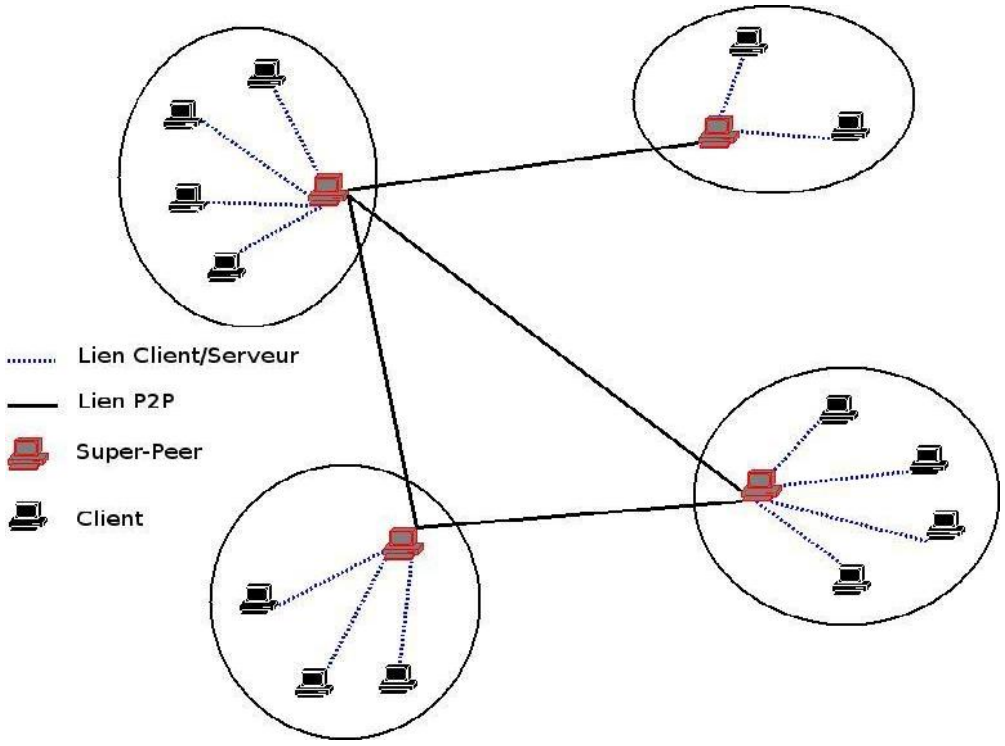

Figure 1.12 : *L'architecture hybride*

Lorsqu'un poste se connecte, il est affecté à l'un des groupes. Si le poste requiert une ressource, il effectue une requête classique client/serveur au "super node". Si un serveur ne possède pas le fichier demandé par un poste de son groupe, alors deux cas se présentent:

- soit il transmet la requête aux autres super nodes,
- soit il transmet l'adresse d'autres super nodes au client où renouveler la requête.

Une connexion s'établit ensuite entre les deux pairs si la ressource est trouvée, quel que soit leur niveau (node ou super node).

Entre deux superpairs, le système continue à fonctionner comme les réseaux décentralisés. Mais la propagation des données est plus rapide, puisqu'elle n'utilise plus que les connexions haut débit.

2.3.2. Avantages

- Indexation centralisée des ressources au niveau d'un groupe par les super nodes.
- Offre une meilleure bande passante.

2.3.3. Les limites

- Complexe à mettre en place.
- La fonction de super node consomme environ 10 % des ressources du poste.

2.3.4. Amélioration

Le choix d'un super-peer "à la tête" d'un groupe introduit de la sensibilité aux défaillances: si le super-peer n'est plus joignable, tous ses clients sont coupés du réseau. Une des améliorations possibles est donc de choisir parmi K super-peers au lieu d'un seul. Une restriction est que les super-peers soient partenaires. Dans ce cas, chaque partenaire est relié à chaque client, et possède un index de toutes leurs ressources. Les clients envoient leurs requêtes selon le principe du Round Robin, ce qui permet de faire baisser la charge d'un facteur K. En revanche, la charge du coût d'entrée d'un nouveau client est multipliée par K, comme le montre le graphique.

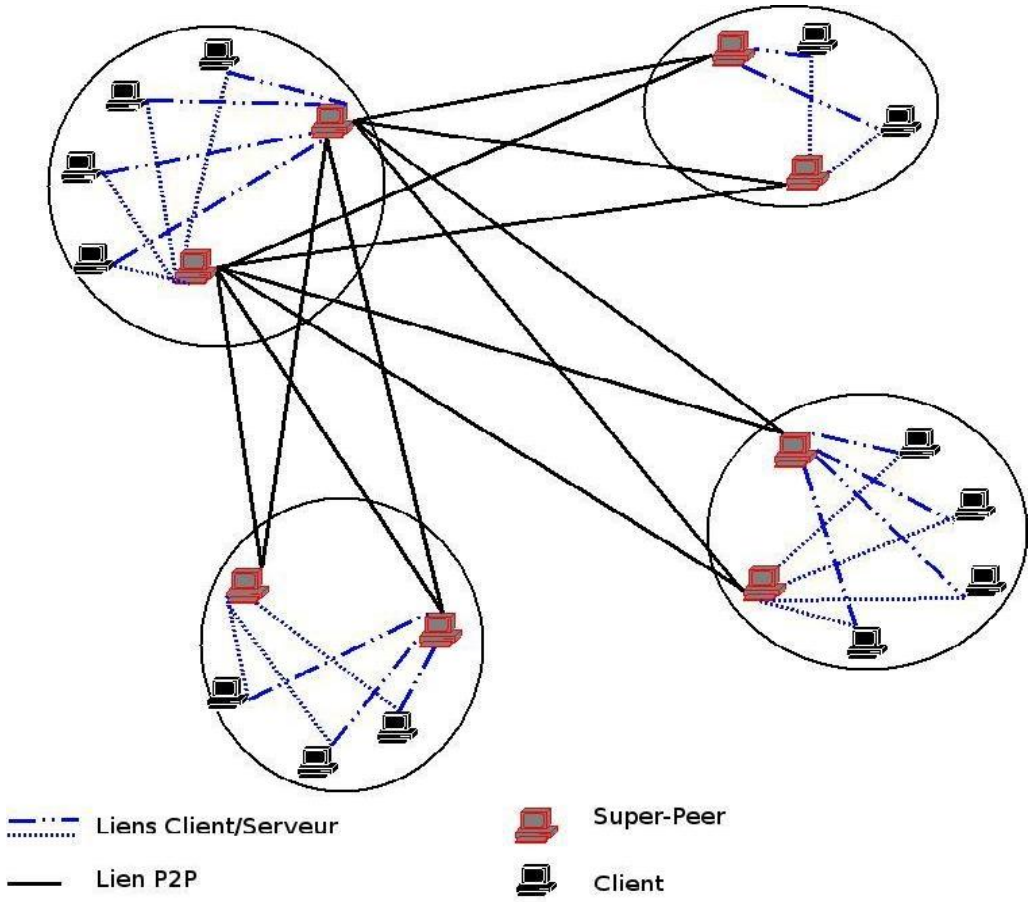

Figure 1.13 : *L'architecture hybrides améliorée*

2.3.5. Règles de conception d'un super-peer

Quelques règles de conception sont évoquées ici :

- Augmenter la taille d'un cluster diminue la charge agrégée, mais augmente la charge individuelle.
- La redondance de super-peer est favorable.
- Il faut maximiser le nombre de connexions des super-peers, ce qui permet de diminuer le nombre de sauts pour atteindre un résultat.
- Il est nécessaire de minimiser le TTL.

2.3.6. L'exemples du réseau Skype

Le logiciel Skype est un client peer to peer de voix sur IP développé par les concepteurs de kazaa, basé sur une architecture hybride.

Ajout d'un noeud :

Pour se connecter au réseau Skype un Client Skype (CS) doit se connecter sur un super node (SN).

A l'installation un CS connaît une liste de well-known hosts, des SNs. L'un d'eux au moins doit être accessible pour que le CS puisse se connecter. Après connexion, il construit au fur et à mesure une liste complémentaire de SN (limitée à 200). Une connexion TCP avec le SN est maintenue pendant toute la durée d'exécution du CS.

Recherche :

Lorsqu'on lance la recherche, il y a un échange entre le CS et son SN qui lui communique 4 (IP/Ports) de SN à interroger, ce que fait le CS. Si c'est infructueux, le CS informe son SN qui lui donne 8 autres adresses de SN à interroger et ainsi de suite. Dans le cas d'un CS derrière un firewall, c'est le SN qui fait la recherche. Des études ont déterminé que le résultat de la recherche est mis dans un cache sur le SN.

Communication:

L'appel entre deux machines qui ont des IP publiques, sans firewall, se fait par l'intermédiaire d'une communication directe.

L'appel entre deux machines dont l'une est derrière un firewall ou un NAT implique une connexion indirecte entre les deux hôtes qui passe par le SN.

3. Applications des technologies P2P

Au sens du grand public, le terme « peer-to-peer » (P2P) peut parfois se confondre avec les systèmes d'échange de fichiers. En réalité, le terme englobe également d'autres applications. D'une manière générale, un système est dit « peer-to-peer » lorsqu'il autorise la communication directe entre entités d'un réseau, sans passer par une autorité centrale, telle qu'un serveur. Dans un réseau P2P, chaque entité se comporte à la fois comme un client et un serveur[Kubiak,2007].

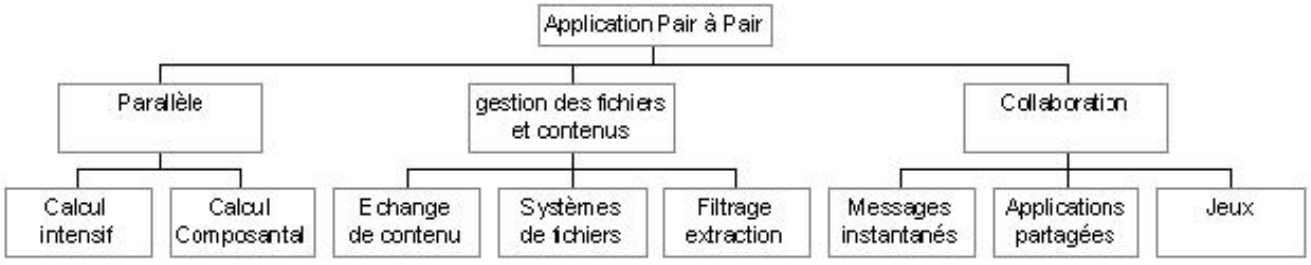

Figure 1.14 : *Taxonomie des applications P2P*

3.1. Le calcul distribué ou Grid Computing

Le P2P permet de mettre en commun de nombreux ordinateurs, disposant chacun de ressources limitées. Mais en "additionnant" les ressources de chacun, on obtient des performances théoriques considérables.

Un exemple parlant aux mathématiciens est le calcul de matrices : l'une des méthodes permettant l'inversion des matrices nécessite n3 opérations (n étant la taille de la matrice (qui comporte donc n² cases)). Un des challenges actuels est de réaliser un tel calcul grâce au P2P, avec n = 10^6 (un million), soit un total de 10^18 (un milliard de milliard) opérations ...

Le but est donc de diviser ce gros calcul qu'est l'inversion de la matrice en petits calculs plus ou moins indépendants que l'on pourra répartir entre les pairs. Cet algorithme demande cependant d'avoir déjà une partie des informations pour pouvoir continuer. Il faudra donc que les pairs échangent des données afin de pouvoir effectuer de nouveaux calculs.

Toutefois, outre le fait de trouver suffisamment de pairs pour réaliser ce test, d'autres problèmes surviennent :

- Les nœuds sont volatiles, c'est-à-dire qu'ils arrivent et partent quand ça leur plaît. Donc il va falloir confier le même calcul à plusieurs noeuds (notion de clouage).
- Du fait que l'algorithme a besoin des calculs précédents pour en effectuer de nouveaux, les temps de communication entre deux noeuds sont très supérieurs aux temps de calcul. Or sans vouloir faire intervenir la rentabilité ici, le projet ne présentera que peu d'intérêt si les

3/4 du temps sont passés à échanger des données et non à calculer.

Des solutions sont donc actuellement étudiées afin de réaliser de grands calculs grâce au P2P. On peut citer, entre autres projets :

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

3.2. Diffusion des données

La diffusion de données en utilisant le P2P permettrait de toucher beaucoup plus de personnes avec des moyens bien inférieurs à ceux nécessaires aujourd'hui. Actuellement, si on veut émettre de la vidéo (par exemple), il faut non seulement une machine qui diffuse la vidéo (logique), mais également une très très bonne connexion : plus le nombre de gens est important, plus la connexion doit être grosse.

En revanche, en utilisant le P2P, chaque personne devient à la fois réceptrice et émettrice. Ainsi, le diffuseur initial n'a plus que quelques clients connectés à lui, chacun de ces clients jouant également le rôle de diffuseurs pour d'autres clients.

Encore une fois, c'est plus complexe à mettre en œuvre qu'à expliquer, les principaux problèmes étant la volatilité des clients qui joue le rôle des serveurs pour d'autres utilisateurs, qui pose des problèmes de rattachement des clients, comme le montrent les figures suivantes.

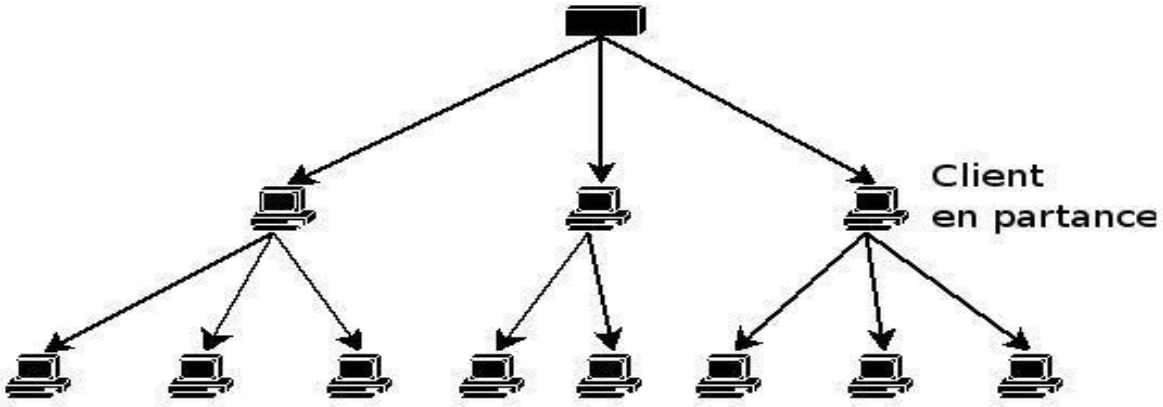

Figure 1.15 : *Modèle de diffusion avant le départ d'un pair*

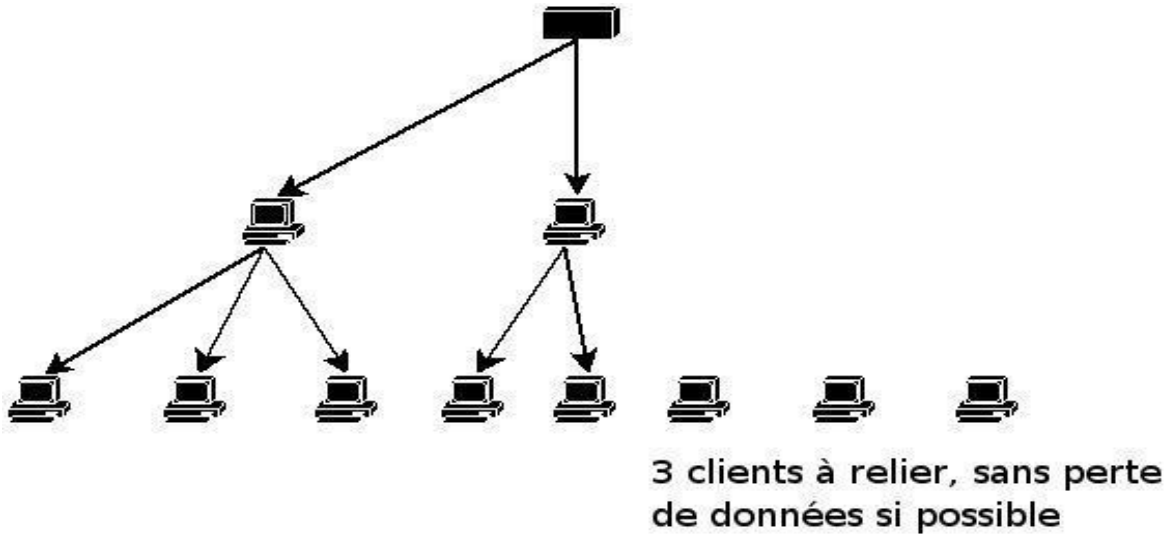

Figure 1.16 : *Situation au moment du départ*

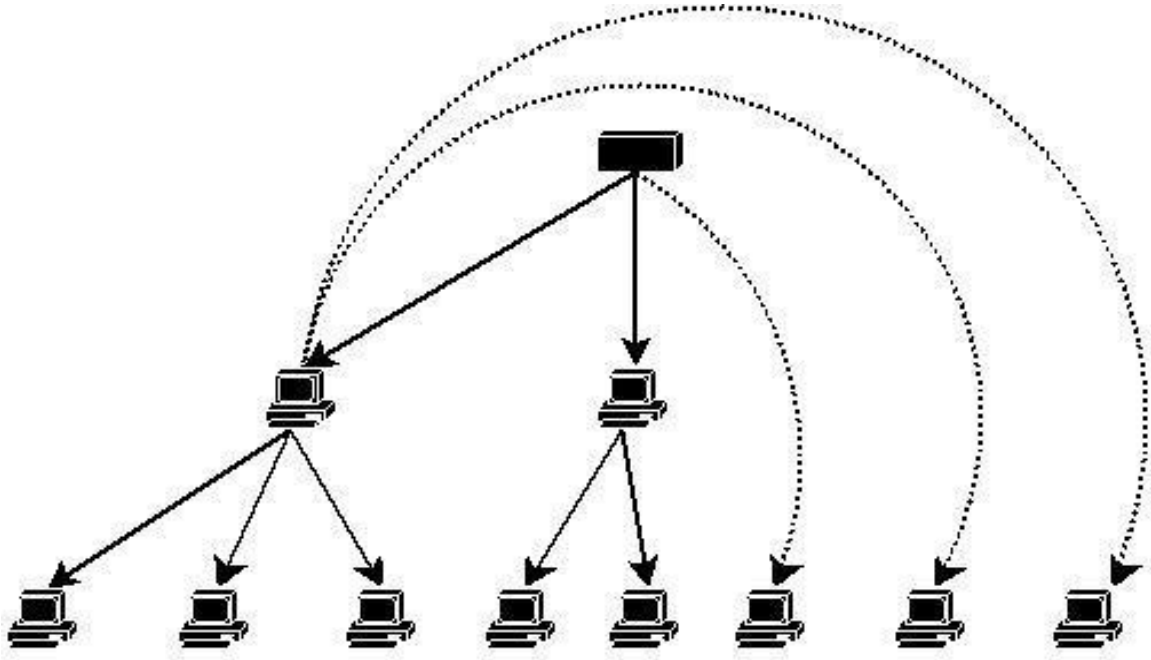

Figure 1.17 : *Une des manières de relier les 3 pairs orphelins*

3.3. Le travail collaboratif

Ce type d'applications permet à des usagers de collaborer en temps réel sans utiliser de serveur central.

Une application populaire est l'utilisation des messages instantanés comme Yahoo, AOL, Jabber. Le principe d'applications partagées ou de travail coopératif est très populaire. Les applications permettant de travailler de manière commune sur un projet distribué sont de plus en plus nombreuses : Groove, Magi, PowerPoint distribué, NextPage ou Kanari.

Par exemple, Groove permet ainsi aux employés d'une même société et aux partenaires commerciaux qui travaillent sur un même projet, de partager des documents, des emplois du temps ou de communiquer en temps réel dans une architecture entièrement décentralisée et cryptée pour assurer la confidentialité de l'ensemble.

Les jeux en réseaux, dont l'architécture est exempte de toute autorité centrale font également partie des applications de collaboration P2P.

Exemple: Doom

3.4. Les moteurs de recherches

Le but est de pallier les limites des moteurs traditionnels qui passent à côté de beaucoup de contenu. On estime qu'un moteur comme Google n'explore que 15% du web.

En effet, le principal problème auquel sont confrontés les moteurs de recherche vient du fait que de plus en plus de pages Web sont générées dynamiquement et que les informations qu'elles contiennent proviennent de bases de données auxquels les moteurs n'ont pas accès.

On peut citer InfraSearch comme moteur de recherches P2P. Il repose sur une architecture complètement distribuée, le réseau Gnutella. Ce type de moteur n'effectue pas lui-même la recherche et la récupération de l'information, il ne fait que transmettre les requêtes formulées par l'utilisateur aux systèmes qui lui sont connectés, qui eux-mêmes les répercutent à leurs voisins sur le réseau. Chaque hôte recevant la requête a la responsabilité totale d'effectuer lui-même la recherche sur son propre système et de renvoyer ensuite à InfraSearch les infos jugées pertinentes.

Autres exemples: Amoweba, OpenCola, Yaci.

3.5. Les bases de données distribuées

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

Exemple: Mariposa, Chord.

3.6. Plate-formes d'uniformisation

La plupart des logiciels P2P ont été développés de manière spécifique sans référence à des standards propres au p2p. Des initiatives ont été prises pour répondre à ce besoin. Elles proposent un environnement permettant de développer des applications P2P, en offrant les fonctions de base : gestion des pairs, nommage, découverte des ressources, communication entre pairs, sécurité. Ainsi Sun propose sa plate-forme JXTA basée sur la technologie Java et Microsoft intègre dans .Net des outils pour développer des applications p2p.

3.7 Plateformes de développement P2P existantes

Nous avons étudié quelques environnements de développement susceptibles de permettre la construction d'applications P2P, dont les deux derniers utilisant le paradigme agent, afin d'étudier leur adéquation à ce contexte et de situer notre approche.

1) XtremWeb6 est une plateforme pour la distribution de tâches en mode P2P sur des grilles de calcul, utilisée dans le cadre de l'ACI GRID. Elle a été conçue pour la distribution de calculs sur des pairs qui peuvent être demandeurs ou fournisseurs de ressources de calcul. La distribution est par nature centralisée même si une hiérarchie de serveurs permet de réduire la charge sur le serveur central.

2) Globus est une alliance pour le développement de technologies autour de la grille de calcul ouverte. Le Globus Toolkit7 est une implémentation en logiciel libre du standard OGSI (Open Grid Services Infrastructure), un modèle de Web Services adaptés à la grille. Cette boîte à outils offre les services et les outils de base requis pour construire une grille de calcul (sécurité, localisation des ressources, gestion des ressources, transmissions). La lourdeur du framework et les difficultés de déploiement le rendent moins adapté aux systèmes P2P les plus dynamiques.

3) JXTA8 est une initiative de Sun Microsystems, proposant un ensemble de protocoles ouverts pour interconnecter des dispositifs allant du téléphone cellulaire jusqu'aux serveurs. Le fonctionnement repose sur le découpage de parties du réseau réel en réseaux virtuels, dans lesquels chaque pair peut accéder aux ressources des autres sans se préoccuper de leur localisation ou de leur environnement d'exécution. JXTA propose une base de six protocoles (services de découverte des pairs, de rendez-vous, d'informations sur les pairs, de

communication, de routage et de résolution des pairs) dans lesquels la communication se fait par des messages XML. Plusieurs implémen- tations existent, par exemple en Java avec JXME9. Cette approche semble intéressante dans le contexte visé, néanmoins elle nous semble plutôt réservée à l'exécution de services distants, sans possibilité de personnalisation.

4) Jade10 (Java Agent DEvelopment Framework) est un framework Java pour l'implémentation de systèmes multi-agents au dessus d'un middleware conforme aux standards FIPA11, notamment pour permettre l'interopérabilité entre agents. Elle permet de réaliser des systèmes P2P dans lesquels chaque pair peut fonctionner de manière proactive, communiquer sans se soucier de la localisation, et se coordonner pour résoudre un problème complexe grâce à l'utilisation d'agents. Un agent spécial unique (Agent Managing System ou AMS) sert de superviseur pour la plateforme, et fournit les services d'annuaire, de cycle de vie . Tout agent créé doit s'enregistrer auprès de l'AMS pour obtenir son identifiant, indispensable à la communication.

Ce point de centralisation est un frein à l'adoption de Jade dans des systèmes répartis à grande échelle, et restreint son usage à des clusters de machines ou des réseaux de petite taille.

5) Un système Anthill12 est un ensemble de pairs sur lesquels est déployé un système multi-agents, dont l'interaction permet de résoudre des problèmes complexes (grâce à des comportements émergents et des algorithme génétiques) comme par exemple le routage des messages. Anthill s'inspire des colonies de fourmis, en proposant des agents aux comportements simples, autonomes, et pourvus d'un environnement sur lequel ils basent leurs actions . Chaque pair est un nid, qui fournit à l'application des services spécifiques au P2P : gestion de ressources, communication, gestion de la topologie, planification des actions. La dernière implémentation (version 1.1 de 2002) repose sur JXTA pour tout ce qui concerne la répartition. Les auteurs proposent également un logiciel pour le partage de fichiers (Gnutant), compatible avec Gnutella et Freenet. Ils montrent que le développement d'une telle application est simplifié par l'usage de leur système.

Hormis JXTA, aucune de ces plateformes n'est vraiment adéquate pour la construction de systèmes P2P purs. En particulier, la plupart des middleware pour la grille reposent sur des services centralisés (éventuellement hiérarchisés) d'allocation de ressources à des tâches indépendantes. Nous proposons une approche qui se démarque des technologies citées ci-dessus par la décentralisation, l'utilisation de composants génériques et la personnalisation des services au moyen d'agents mobiles.

Conclusion

On a vu que les réseaux peer to peer pouvaient reposer sur différentes architectures présentant chacune des avantages et des inconvénients.

Le P2P a connu et connaît toujours un franc succès auprès du grand public grâce aux logiciels de partage et de communication. Néanmoins, le Peer to peer se développe également auprès du monde professionnel de par son utilisation notamment dans le calcul distribué et le travail collaboratif. Dans de nombreux cas, les technologies P2P peuvent servir d'alternative au mode client serveur, ce qui permet de décharger les serveurs.

En revanche, deux problèmes se posent avec son utilisation. Le premier concerne le fait que les réseaux P2P n'ont pas été développés selon un standard. Conséquence, ces réseaux ne sont pas compatibles entre eux.

Le deuxième problème provient de la sécurité. En effet, un réseau P2P est constitué de peers inconnus et donc potentiellement dangereux. D'autre part, ce type de réseau est un moyen de diffusion de virus et de vers. Il convient ainsi de mettre en œuvre différents mécanismes de sécurité, parmi lesquels on trouve le cryptage des données avec des clés multiples, le sandboxing qui consiste à exécuter une application dans un environnement clos, la gestion des droits d'utilisation qui permet la protection intellectuelle des données, la facturation qui permet d'éviter le phénomène de pure consommation, et enfin la protection des hôtes par des pare-feux.

Chapitre **II**

Les systèmes multi-agents

Introduction

Les systèmes multi-agents (SMA) sont devenus l'un des courants systèmes les plus prometteurs. Le champ des applications des SMA s'est élargi avec le récent développement des réseaux et d'Internet en particulier. Le grand essor de la toile a fait apparaître de nouvelles opportunités de développement des SMA tels que le commerce électronique et récemment l'accélération des réseaux.

Les systèmes multi-agents sont l'un des paradigmes technologiques les plus prometteurs dans le développement de systèmes logiciels distribués, ouverts et intelligents. Nous entendons par systèmes ouverts, les systèmes auxquels peuvent se rattacher d'autres systèmes. La technologie agent commence par être utilisée pour concevoir des solutions facilitant la mise en place de nouveaux concepts notamment économique comme le e-commerce.

1. Positionnement historique

Les systèmes multi-agents se positionnent au carrefour de la programmation (ce sont des logiciels), de l'intelligence artificielle (leur autonomie de décision), et des systèmes répartis (leur décentralisation).

Historiquement, on peut replacer le concept d'agent et de système multi-agents dans l'histoire de l'intelligence artificielle et de manière duale dans l'histoire de la programmation. La notion d'agent est de fait à la base des débuts de l'intelligence artificielle (**IA**). Mais cette discipline s'est focalisée sur la modélisation des capacités intelligentes d'une unique entité pour résoudre des problèmes. Il en a résulté une première génération de programmes informatiques évolués, tels les systèmes experts.

Mais, même restreint à un domaine spécialisé (domaine expert), Un tel système expert était censé résoudre tout seul les problèmes de manière autarcique. L'accent a donc été mis progressivement à partir de la fin des années 70 sur une résolution distribuée de problèmes, par coordination d'un certain nombre d'agents, ce que l'on a alors commencé à appeler « systèmes multi-agents ». On utilise également le terme quasiment équivalent « **IA distribuée** » (avec son acronyme **IAD**, en anglais **DAI**[1]) pour bien montrer l'opposition à l'**IA** classique autarcique et centralisée.

[1] **DAI**: Distributed Artificial Intelligence.

Il est important pour la suite de pouvoir définir certains termes qui reviendront très souvent dans cet exposé à savoir Agent, Système multi-agents, etc. Il est important de notifier que les définitions que nous présentons ici s'inscrivent dans le cadre de notre travail (SMA: application dans le commerce électronique) et peuvent donc avoir d'autres connotations dans d'autres domaines.

2. Qu'est-ce qu'un agent ?

Il y a un nombre important d'ouvrages offrant des définitions des agents et des systèmes multi-agents. Les difficultés sont semblables en quelque sorte à celles rencontrées par les scientifiques qui ont essayé de définir la notion d'intelligence artificielle. Pourquoi a-t-il été si difficile de définir l'intelligence artificielle et pourquoi est-il si difficile de définir les systèmes d'agents, quand d'autres concepts de l'informatique, tels que l'objet et l'orienté objet, le calcul distribué, etc., n'ont pas rencontré une si grande résistance à être définis.

Une réponse possible est que la notion d'agent, ainsi que la notion d'intelligence artificielle, ont émergé des humains et de la société humaine. Il est évidement difficile de modéliser ou de simuler le comportement spécifique humain au moyen de programmes informatiques.

Il y'a plus de 30 ans, les scientifiques ont essayé d'écrire des programmes pour mimer le comportement humain intelligent, dont le but était de créer un système artificiel ayant les mêmes capacités qu'une personne intelligente.

Il n'y a pas une définition acceptée en unanimité pour la notion d'agent. Dans ce qui suit, on présente les définitions les plus importantes.

- Un agent est une entité qui perçoit son environnement et agit sur celui-ci [Bensaid and P. Mathieu,1997].
- Un agent est un système informatique, situé dans un environnement, et qui agit d'une façon autonome pour atteindre les objectifs pour lesquels il a été conçu [Bensaid and P. Mathieu,1997].
- Un agent est une entité qui fonctionne continuellement et de manière autonome dans un environnement où d'autres processus se déroulent et d'autres agents existent [Bensaid and P. Mathieu,1997];
- Un agent est une entité autonome, réelle ou abstraite, qui est capable d'agir sur elle-même et sur son environnement, qui dans un univers multi-agents, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents [Chaib-draa, et al,2001].

Nous pouvons identifier deux tendances principales en comparant les définitions données ci-dessus pour les agents et les agences. Quelques chercheurs considèrent que nous pouvons définir un agent en isolation, alors que d'autres considèrent les agents principalement comme entités agissant dans une société d'autres agents, à savoir le paradigme des systèmes multi-agents.

Les deux tendances ont déjà donné des résultats. C'est vrai qu'on pense que c'est le paradigme des **SMA**[2] qui va s'imposer comme prépondérant car il est plutôt difficile de compter qu'un agent existe seulement comme une entité pour lui seul et ne rencontrera pas d'autres agents (soit artificiels ou humains) dans son environnement.

Les agents personnels, ou les agents d'information, qui ne sont pas censés principalement travailler collectivement pour résoudre des problèmes, auront certainement beaucoup à gagner s'ils agissent par interaction avec d'autres agents et bientôt, avec la diffusion large de la technologie multi-agents, ils ne pourront pas réaliser leurs tâches en isolation. Par conséquent, les chercheurs considèrent la dimension sociale d'un agent comme une de ses caractéristiques essentielles.

D'autres considèrent la mobilité en tant qu'une des caractéristiques des agents informatiques. On a des réserves sur cette opinion parce que la mobilité est un aspect relié principalement à la mise en place ou à la réalisation des agents, soient-ils logiciels (agents mobiles sur la toile) ou matériels (robots qui se déplacent dans le monde physique) et peut être comprise dans la capacité des agents d'interagir avec l'environnement.

En partant de l'ouvrage de [Chaib-draa, et al,2001], et des définitions citées, on peut identifier les caractéristiques suivantes pour la notion d'agent:

- *situé* – l'agent est capable d'agir sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement;
- *autonome* – l'agent est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ses propres actions ainsi que son état interne;
- *proactif* – l'agent doit exhiber un comportement proactif et opportuniste, tout en étant capable de prendre l'initiative au bon moment;
- *capable de répondre à temps* – l'agent doit être capable de percevoir son environnement et d'élaborer une réponse dans le temps requise;
- *social* – l'agent doit être capable d'interagir avec d'autres agents (logiciels ou humains) afin d'accomplir des tâches ou aider ces agents à accomplir les leurs.

Bien que presque toutes les caractéristiques ci-dessus des agents puissent être considérées en tant que partage de quelque chose avec le comportement intelligent, les

[2] **SMA**: **S**ystème **M**ulti-**a**gents.

chercheurs ont essayé de donner une différence nette entre les agents logiciels (software) et les agents logiciels intelligents, glissant dans le monde des agents la différence si recherchée entre les programmes et les programmes intelligents. D'un point de vue, il est clair que, si dans la conception d'un système agent ou multi-agents, nous utilisons des méthodes et des techniques spécifiques à l'intelligence artificielle, alors ces agents peuvent être considérés comme étant intelligents, évidemment dans le sens de l'intelligence artificielle.

L'agent peut également posséder d'autres propriétés (de second ordre) notamment la mobilité, la faculté d'apprentissage, etc. Il est important de mentionner que celles-ci ne sont pas essentielles pour l'agent. En combinant diverses qualités propres aux agents, on peut définir différents types d'agents comme l'indique la **FIG 2.1.**

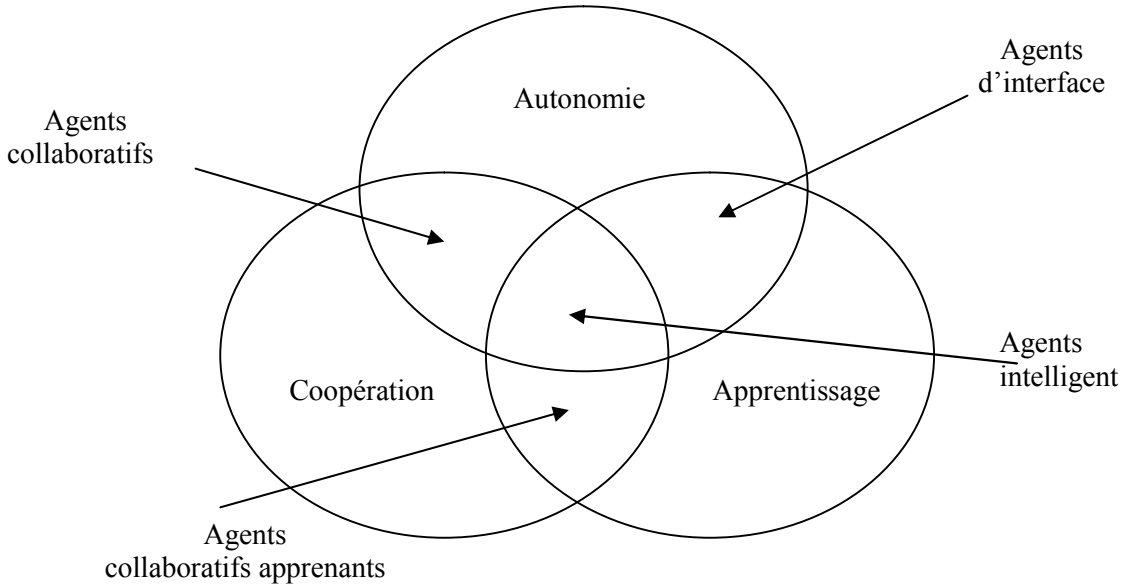

Figure 2.1 : *Différents types d'agents.*

Par exemple, dans la **FIG 2.1**, les propriétés telles que la coopération, l'apprentissage et l'autonomie permettent de définir quatre types particuliers d'agents :

- les agents intelligents,
- les agents collaboratifs,
- les agents collaboratifs apprenants,
- les agents d'interface.

2.1 L'architecture d'agent

La représentation d'agent en tant que boîte noire (« black-box ») est considérée comme une architecture minimale, commune à tous les domaines de recherche d'**IA**[3] [Chaib-draa, et al,2001].

Figure 2.2 L'architecture minimale d'un agent

L'architecture minimale (**FIG 2.2**) est composée de trois modules : le module de perception, le module d'action et le module de traitement intelligent. Ce dernier détermine le comportement d'agent qui peut ainsi analyser les données reçues et construire la réponse sous la forme d'action ou de message.

En général, les architectures existantes sont divisées en trois familles principales : les agents cognitifs (délibératifs), les agents réactifs et les agents hybrides.

2.2 Une comparaison avec les objets

Bien des programmeurs croient encore que les agents ne sont en réalité que des simples objets, au sens attribués à ce terme dans le domaine de la programmation. En fait, un objet est une certaine modélisation d'une entité réelle ou abstraite. Cet objet encapsule un état et il est possible de modifier cet état en invoquant différentes méthodes faisant aussi partie de l'objet. Les agents sont aussi des entités informatiques qui possèdent un état interne (privé), qui sont capables d'agir et de communiquer par échanges de messages. Ce qui différencie les objets des agents, c'est que ces derniers reposent sur un concept très fort d'autonomie leur permettant d'agir de manière très flexible. La négociation, processus par lequel différents groupes arrivent à un accord, est un concept étranger au monde objet. La démarche en vue d'une négociation requiert un comportement flexible afin de pouvoir faire certaines concessions de part et d'autre.

De plus, les agents contrôlent d'une certaine façon leur état interne. Par exemple, si un objet **O1** invoque une méthode M1 sur l'objet **O2**, cette méthode sera exécutée si aucune erreur ne survient. Cependant, si un agent **A1** effectue une requête **R1** à l'agent **A2**, c'est ce dernier qui décidera si la requête sera effectuée ou non. Toute la différence se situe dans les termes invocation et requête.

[3] **IA: I**ntelligence **A**rtificielle.

Le tableau ci-dessous **(Tab 01)** présente un récapitulatif des éléments de base de l'approche orientée objet versus l'approche orientée agent présenté par **Shoham**[4] [Esmahi,2001].

Propriétés	**AOO**[5]	**AOA**[6]
Élément de base	Objet	Agent
Paramètres définissant l'élément de base	Sans contraintes	Croyances, engagements, capacités, choix
Type d'exécution	Envoie et réception de messages	Envoie et réception de messages
Type de messages	Sans contraintes	Actes du langage
Contraintes sur les méthodes	Sans contraintes	Honnête, consistant

TAB 01 – Approche orientée objet (**AOO**) versus approche orientée agent (**AOA**) [Esmahi,2001].

2.3 Classes d'agent

De manière générale, il est possible de classer un agent selon les trois principaux points suivants :

a)- **Selon leur nature** :

– *Agents réactifs* : Ce sont des agents qui n'ont pas une représentation explicite de leur environnement, ni de mémoire de leur passé, ni de but explicite et leur comportement est de type stimulus réponse.

– *Agents délibératifs :* Ce sont des agents qui possèdent une représentation explicite de leur environnement. Ils se basent sur la connaissance qu'ils ont de leur environnement et leur habileté à raisonner sur leurs connaissances. Les agents dirigés par leurs buts ou par une fonction d'utilité font partie de ce groupe.

– *Agents hybrides :* De façon habituelle, un agent n'est pas complètement de type réactif ni de type délibératif. C'est plutôt une combinaison de ces deux approches.

[4] **Yoav Shoham** : Professeur à Université de Stamford.
[5] **AOO**: **A**pproche **O**rientée **O**bjet.
[6] **AOA**: **A**pproche **O**rientée **A**gent.

Dans certaines circonstances, un agent doit agir très rapidement alors que dans d'autres, il aura la possibilité de prendre plus de temps pour avoir un processus délibératif de meilleure qualité. Dans ce cas, les agents sont dits hybrides.

b)- **Selon l'utilisation** :

– *Agents collaboratifs :* Ces agents ont des habiletés de coopération. Un regroupement de ces agents permet, entre autres, de réduire un problème complexe en sous-problèmes moins complexes.

– *Agents d'interface :* Ces agents collaborent avec l'utilisateur pour effectuer certaines tâches.

– *Agents pour la recherche d'informations :* Ces agents effectuent, en premier lieu, une recherche d'informations parmi une collection de données et, en second lieu, procèdent à une analyse des informations utiles trouvées afin de découvrir de nouvelles connaissances.

– *Agents pour le commerce électronique :* La montée de l'Internet a bien entendu créé de nouvelles nécessités. Les agents issus de cette tendance permettent la promotion, la vente ainsi que l'achat de produits et de services par l'entremise des réseaux informatiques, etc.

c)- **Selon la technologie employée** :

– *Agents stationnaires :* Il s'agit du cas où l'agent s'exécute toujours sur la même machine.

– *Agents mobiles :* Ces agents s'exécutent sur différentes machines en se promenant d'un hôte à l'autre. Typiquement, ils suivent ce que l'on appelle un "itinéraire".

Le plus grand avantage réside dans le fait qu'ils réduisent la charge des réseaux en se rendant directement sur la machine "utile" pour y effectuer les actions de manière locale.

D'un point de vue général, les chercheurs aimeraient que les agents soient des composants logiciels réutilisables qui permettraient d'accéder et/ou contrôler des services et des ressources informatiques. Ces agents seraient les unités de base des applications informatiques et seraient organisés en réseau dans une optique de collaboration.

L'ensemble des entités (Agents) en interaction et en vue de réaliser un but commun forme ce qu'on appel un système multi-agents, Alors la question qui se pose est :

3. Qu'est-ce qu'un système multi-agents (SMA)?

3.1 Les systèmes multi-agents

Les systèmes multi-agents font partie de l'Intelligence Artificielle Distribuée (**IAD**), qui, à la différence d'**IA** classique, s'intéresse aux comportements intelligents, résultant de l'activité coopérative de plusieurs agents [Chaib-draa, et al,2001].

Bien qu'il soit difficile de donner une définition générale pour l'expression « **S**ystème **M**ulti-**A**gents "**SMA**" », les différentes caractérisations proposées par [Chaib-draa, et al,2001]

[Mathieu, et al,2002] et autres permettent de dégager des idées fondamentales communes. La définition, mentionnée ci-dessous, est conforme à la plupart de ces idées.

3.2 Définition: Système multi-agents (SMA)

Le système multi-agents est considéré comme un système distribué composé d'un certain nombre d'entités autonomes[7] *(les agents), qui travaillent selon les modes complexes d'interaction, pour réaliser leurs propres buts et par-là même atteindre l'objectif global désiré. Les agents peuvent interagir en communiquant directement entre eux ou par l'intermédiaire d'un autre agent ou en agissant sur leur environnement.*

En général, dans les systèmes multi-agents, deux types d'agents sont identifiés : les agents **IAD**[8] et les agents **mobiles**. Les agents **IAD** sont capables de communiquer et de coopérer à l'aide de protocoles basés sur les réseaux contractuels et les actes de langage. Les agents **mobiles** peuvent se déplacer de site en site dans le réseau pour accomplir des tâches spécifiques.

En général, ils peuvent avoir deux types de mobilité :

- ✓ L'exécution distante:

  L'agent (le programme et les données) est transféré sur le site distant, où il est exécuté.

- ✓ La migration:

  Pendant son exécution, l'agent se déplace vers un autre site afin d'accomplir progressivement sa tâche. Autrement dit, l'agent est capable de suspendre son exécution sur un site, de se déplacer vers un autre en transportant le code, les données et l'état d'exécution, et de reprendre l'exécution depuis le point de suspension.

D'après la définition du système multi-agents, proposée ci-dessus, pour qu'un système soit considéré comme SMA, il est nécessaire de satisfaire certains critères :

- Disposer d'un ensemble des agents autonomes, fonctionnant en parallèle et cherchant à satisfaire un but ;
- Les agents doivent posséder un mécanisme d'interaction de haut niveau, indépendant de problème à résoudre. Ils peuvent utiliser les langages de communication d'agents (*ACL*), par exemple, *KQML*[9], *FIPA ACL*, etc. ;
- L'agent doit percevoir une partie de son environnement qui peut être le monde physique, d'autres agents, le réseau Internet, etc. Il doit répondre dans un délai acceptable.

[7] **L'autonomie** signifie que l'entité (l'agent) est capable de travailler sans l'intervention d'un humain ou des autres agents, et de contrôler ses actions ainsi que son état interne

[8] **IAD** : Intelligence Artificiel Distribué.

[9] **KQML**: **K**nowledge and **Q**uery **M**anipulation **L**anguage.

On peut déduire, par conséquent, que la distribution modulaire d'une application ne constitue pas forcement un système multi-agents. En effet, les modules n'ont pas de buts à atteindre ou la fonction de satisfaction. Leurs mécanismes d'interactions sont de bas niveau : appels de procédures, etc. En ce qui concerne les agents, ils reçoivent des messages qui peuvent être des demandes d'exécution, mais, aussi, des requêtes d'information sur leurs capacités. Ils tentent de satisfaire les objectifs, ils disposent pour cela d'une autonomie supplémentaire par rapport aux objectifs. L'agent peut refuser une tâche donnée à cause de son manque de compétences ou de sa trop grande occupation.

3.3 Les interactions et les agents

L'interaction est une notion importante dans les systèmes multi-agents. Au dire de **J**.**Ferber** [Ferber,1995], l'interaction permet d'avoir une relation dynamique entre deux ou plusieurs agents par le biais d'actions réciproques. Les situations d'interactions sont diverses (l'aide d'un robot à un autre, l'échange de données entre deux serveurs, etc.).

Pour un agent, interagir avec un autre agent constitue la source de sa puissance et l'origine de ses problèmes [Ferber,1995]. En effet, seulement en coopérant les uns avec les autres, les agents peuvent accomplir leurs tâches collectives. D'autre part, à cause de leurs interactions avec les autres agents, ils doivent coordonner leurs actions et résoudre des conflits.

Traiter le problème d'interaction revient non seulement à décrire les mécanismes permettant aux agents d'interagir, mais aussi à analyser et à concevoir les différentes formes d'interaction, utilisées par les agents pour accomplir leurs tâches et satisfaire leurs buts. Nous présentons le concept d'une de ces formes – la communication - sur laquelle repose toute l'interaction.

3.3.1 La communication

La communication est essentielle dans la résolution coopérative des problèmes [Mathieu, et al,2002] [Ferber,1995]. Elle permet de synchroniser les actions des agents et de résoudre les conflits des ressources par la négociation. D'après [Chaib-draa, et al,2001], la communication définit l'ensemble des processus physiques et psychologiques par lesquels s'effectue l'opération de mise en relation d'un agent émetteur avec un ou plusieurs agents récepteurs, dans l'intention d'atteindre les objectifs prévus. Les processus physiques décrivent les mécanismes d'exécution des actions, par exemple, l'envoi et la réception de messages. Les processus psychologiques désignent les changements opérés par la communication sur les buts et les croyances des agents.

En général, les actions de communication entre les agents sont considérées comme les actions d'échange d'information.

Il existe plusieurs types de communication, à savoir, la communication par tableau noir, la communication par partage d'informations et la communication par envoi des messages qu'on va détailler dans ce qui suit.

3.3.1.1 La communication par envoi des messages

Ce type de communication permet aux agents d'envoyer leurs messages directement aux destinataires par les mécanismes spécifiés (les canaux ou les ports). Il existe trois types de messages : les questions, les réponses et les informations. Au niveau protocolaire, un envoi de message peut être synchrone (un agent émetteur attend la réponse de son récepteur) et asynchrone (un agent émetteur peut agir immédiatement après avoir placer son message dans une fille d'attente).

La communication par l'envoi de messages peut être organisée suivant trois formes différentes :

- *La communication point à point.*

L'agent émetteur connaît l'agent destinataire et lui transmet directement son message. L'agent destinataire est le seul à recevoir le message envoyé ;

- *La distribution généralisée.*

Il s'agit d'un envoi d'un même message à tous les agents du système multi-agents. L'agent émetteur ne connaît pas forcement les destinataires du message envoyé.

- *La distribution restreinte.*

Il s'agit d'un envoi d'un même message à un certain groupe d'agents du système multi-agents. L'agent émetteur ne connaît pas forcement tous les destinataires, mais il doit être capable de les atteindre en s'appuyant, soit sur leurs caractéristiques, soit sur la notion de groupe auquel ils appartiennent.

3.4 Les langages de communication

Le concept de communication permet de réaliser les échanges locaux d'information entre des agents. Le problème est alors de combiner ces actions de communication avec les compétences et les connaissances des agents, pour obtenir un comportement collectif entre ces derniers. Dans les systèmes multi-agents, la problématique susmentionnée est étudiée à travers le concept de l'interaction des agents. Cette dernière permet d'avoir une relation dynamique entre deux ou plusieurs agents par le biais d'actions réciproques [Ferber,1995] .La question qui se pose alors, est comment réaliser les interactions des agents ?

D'après [Esmahi,2001], pour que les agents puissent interagir de manière efficace, ils doivent posséder un langage de communication commun, leur permettant de se comprendre ainsi que de s'échanger des informations et des connaissances.

Avant d'aborder les langages de communication existants, nous présentons brièvement la théorie des actes de langage, considérée en intelligence artificielle comme un modèle général de communication entre les agents.

3.4.1 La théorie des actes de langage

La théorie des actes de langage (« ***Speech Act Theory*** »), constitue un fondement théorique de la communication, basée sur l'idée suivante : « *Lorsqu'on parle, on effectue des actions* »[10] [Esmahi,2001]. Un acte de langage définit un message, qui contient l'affirmation positive ou négative, et provoque les changements de l'environnement.

Chaque acte de langage comprend trois composants :

- Le composant ***locutoire***, qui décrit l'expression d'un message ;
- Le composant ***illocutoire***, qui définit les intentions de l'émetteur, associées implicitement au message ;
- Le composant ***perlocutoire***, qui décrit les effets d'un acte de langage sur l'environnement.

Dans la théorie des actes de langage, les intentions des émetteurs sont identifiées en utilisant les verbes performatifs. Ces derniers sont classifiés en plusieurs catégories : **les affirmatifs** (informer), **les directifs** (ordonner), **les promissifs** (promettre), **les déclaratifs** (déclarer) et **les expressifs** (exprimer).

D'après cette théorie, chaque acte de langage multi-agents peut être décrit sous la forme d'un message, dont le type est défini par le verbe performatif, tel que « *Request* » ou « *Inform* ». Le contenu de ce message est décrit en utilisant les langages de communication (**ex**: *KQML, FIPA ACL*).

3.4.2 KQML

KQML (« ***K****nowledge and* ***Q****uery* ***M****anipulation* ***L****anguage* ») [Esmahi,2001] est une approche basée sur les actes de langages, qui permet de réaliser les interactions des agents tenant en compte la diversité des langages de communication.

La communication est considérée comme un ensemble d'échanges de messages *KQML*. Chaque message comprend trois couches [Mathieu, et al,2002]:

- La couche de *contenu*, qui spécifie le contenu réel du message d'agent ;

[10] Cette théorie est introduite en 1962, par le philosophe et le linguiste anglais **J.-L.Austin** dans son livre « *How to do things with words* ».

- La couche de *communication*, qui décrit tous les paramètres de communication de bas niveau, par exemple, l'identificateur de l'agent émetteur et celui de l'agent récepteur, l'identificateur de la communication, etc.
- La couche de *message*, qui est considéré comme le noyau de *KQML*. Sa fonction principale s'agit d'identifier le protocole de réseau, utilisé pour envoyer le message, et de déterminer le performatif, indiquant le type de ce message (**ex**: une affirmation, une requête, une commande, etc.).

Le message *KQML* est représenté sous la forme d'une liste, contenant le performatif, qui correspond au type particulier d'acte de langage (**ex:** *tell* (transférer l'information aux autres agents), ***ask-one*** (demander la réponse à l'agent correspondant), etc.), et les arguments associés à ce performatif (**FIG 03**).

```
(KQML-performative
    :sender     <name>          le nom de l'agent émetteur du message
    :receiver   <name>          le nom de l'agent récepteur du message
    :language   <text>          le langage, utilisé pour décrire le contenu du message
    :ontology   <text>          le vocabulaire du contenu de message
    :content    <expression>    le contenu réel du message
                    )
```

FIG 2.3 : *La structure d'un message **KQML**.*

Remarque: Il est à noter que l'ordre des arguments dans une liste n'est pas important.

Le langage de *KQML* permet d'utiliser les différents protocoles de communication, notamment, *TCP/IP*, *SMTP* (*e-mail*), *HTTP*, *CORBA*, etc. Parmi les travaux réalisés concernant le développement de nouveaux protocoles, basés sur KQML, on peut mentionner les langages de coordination ***COOL*** et ***AgenTalk*** [Mathieu, et al,2002].

Bien qu'il présente un grand intérêt pour les utilisateurs, *KQML* montre quelques lacunes:

- La signification floue de certains performatifs (**ex**: le cas d'un performatif « dénier ») ;
- Le manque des performatifs promissifs, exprimant l'engagement, auprès d'un tiers, d'accomplir une action ;
- L'utilisation de ce langage que pour des communications isolées.

3.4.3 FIPA ACL

Pour résoudre certains problèmes inhérents à *KQML*, *FIPA*[11] a proposé un nouveau langage de communication d'agents – *FIPA ACL*[12].

Comme *KQML*, ce langage est fondé sur la théorie des actes de langages. Sa spécification consiste en la définition d'un ensemble des types de messages et en description de leurs pragmatiques, c'est-à-dire, les effets sur les attitudes mentaux des agents émetteurs et des agents récepteurs. La sémantique formelle de *FIPA ACL* se compose de cinq niveaux:

- Le protocole, qui décrit les règles sociales des dialogues entre les agents ;
- Les actes communicatifs (*AC*), qui définissent le type de communication entre les agents (par exemple, la demande, la confirmation, etc.) ;
- Le méta-information concernant le message (l'identification d'un agent émetteur et d'un agent récepteur, le contexte, etc.) ;
- Le langage du contenu, qui décrit la grammaire et la sémantique associée, utilisées pour exprimer le contenu d'un message ;
- L'ontologie, qui définit le vocabulaire et les significations des termes et des concepts, employées dans le contenu.

La syntaxe de *FIPA ACL* est identique à celle de *KQML*, sauf les noms de certains primitifs réservés. Le message *FIPA ACL* est représenté sous la forme d'une liste, contenant le type de l'acte communicatif (par exemple, *INFORM*, *REQUEST*), le nom de l'agent émetteur et celui de l'agent récepteur, le contenu et le contexte du message, l'ontologie à utiliser pour interpréter ce contenu, et le protocole :

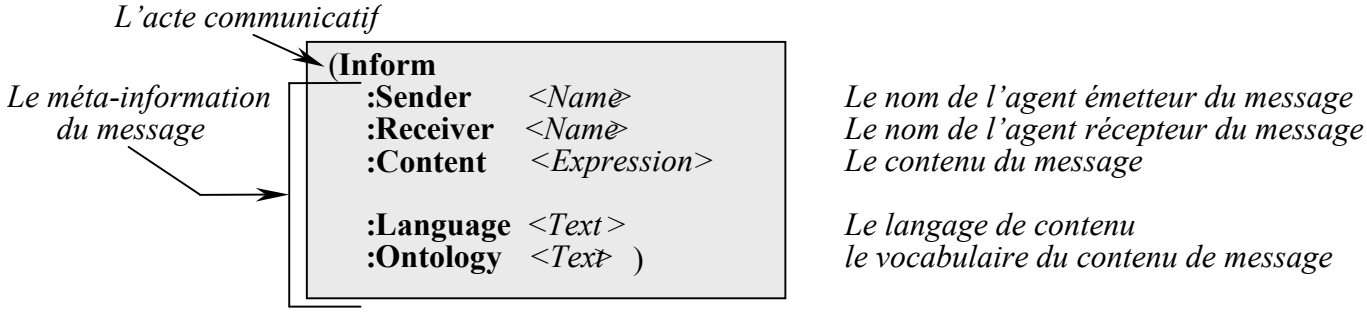

FIG 2.4 : *La structure d'un message FIPA ACL*

FIPA ACL peut être considéré comme l'extension de *KQML* possédant deux langages différents. Le langage externe définit la signification intentionnelle du message. Le langage interne (ou le contenu) décrit l'expression à laquelle s'appliquent les croyances, les désirs et les intentions des agents, décrites dans le primitif de communication.

[11] **FIPA**: **F**oundation for **I**ntelligent **P**hysical **A**gents.
[12] **FIPA ACL**: **FIPA A**gent **C**ommunication **L**anguage.

Différemment de *KQML*, *FIPA ACL* est basé sur la sémantique logique de la communication. Ceci facilite la description des formats de la communication. Cependant, les agents ne possèdent pas toujours les capacités logiques nécessaires. Il est à noter que la sémantique de *FIPA ACL* est basée en grande partie sur les croyances des agents, qui peuvent être inconnus pour les autres agents.

Une autre grande différence entre *FIPA ACL* et *KQML*, concerne les actes communicatifs. *FIPA ACL* contient un ensemble d'actes communicatifs normatifs, qui peuvent être primitifs ou composés. Les nouveaux actes communicatifs ne peuvent être définis qu'en combinant les actes existants et en utilisant les opérateurs prédéfinis. Ceci permet de maintenir l'intégrité sémantique du langage.

3.5 Sémantique des messages

Pour que les agents puissent comprendre les messages inter-changés, il faut qu'ils partagent un vocabulaire commun. Il s'agit de garantir que les concepts et les entités véhiculées au travers des applications ont la même signification, même si différentes applications utilisent des noms différents les référant.

Ce vocabulaire ce n'est autre qu'une ontologie qui représente une sorte d'un vocabulaire d'un domaine bien précis.

Et voici un exemple d'ontologie utilisée dans le commerce électronique (Vente & achat des livres).

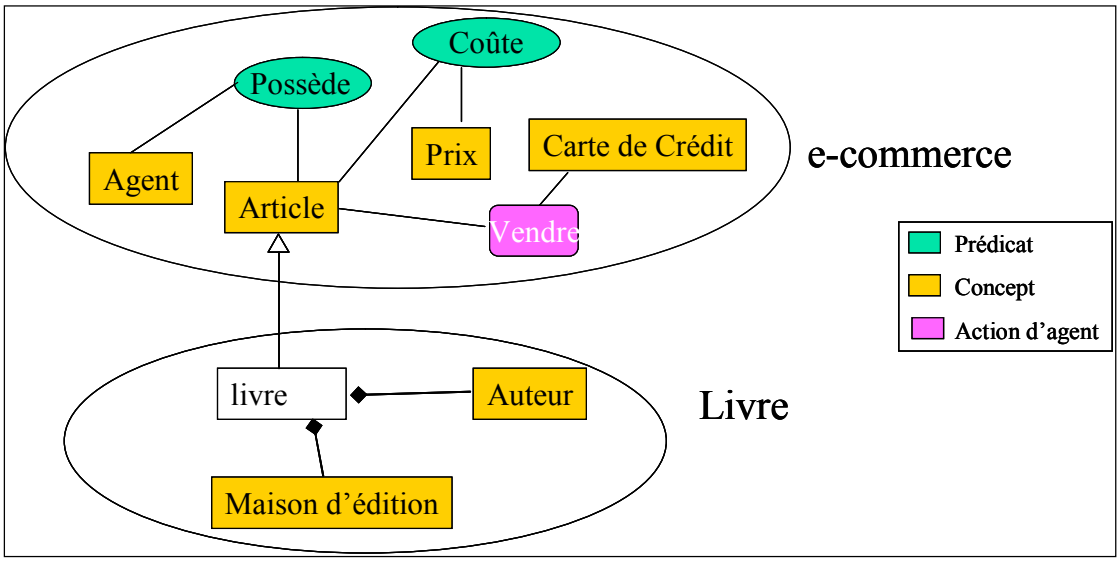

FIG 2.5: *La représentation de l'ontologie pour la vente&achat des livres*

On remarque qu'il existe trois catégories des composants (suivant les trois couleurs), ceux en couleur verte sont les **prédicats,** ceux en couleur jaune sont les **concepts** et dernièrement, ceux qui sont en couleur violet sont les **Action d'agents.**

Ces trois composants sont les composants de l'ontologie (vocabulaire) partagée entre ces agents.

L'intérêt de cette ontologie est essentiellement de décrire le vocabulaire d'un domaine (comme le commerce électronique), elle permet la réutilisation des connaissances, en outre, elle ajoute (enrichit) une sémantique aux messages inter-changés entre les agents de notre système.

3.6. La coopération

La coopération est l'une des caractéristiques fondamentales des systèmes multi-agents.

- D'après [Ferber,1995] ,le problème de la coopération peut se ramener à résoudre les différents sous problèmes qui comprennent la collaboration des agents par répartition des tâches, la coordination d'actions et la résolution de conflits. **J**.**Ferber** présente deux points de vue sur la coopération [Ferber,1995]:
  - ✓ *Une attitude intentionnelle* d'agents qui décident de travailler ensemble. Dans ce cas, les agents coopèrent s'ils effectuent une action commune, après avoir identifié et adopté un but commun. Ce concept présente deux inconvénients principaux : d'une part, on considère que la coopération existe, même si les agents obtiennent meilleurs résultats en travaillant individuellement ; d'autre part, d'après l'attitude intentionnelle, la possibilité pour les agents réactifs de pouvoir coopérer est supprimée ;
  - ✓ *Une qualification d'une activité du groupe d'agents, observée par l'observateur*, qui interprète les comportements à partir de critères physiques et sociaux. Dans ce cas, plusieurs agents coopèrent si (1) l'ajout d'un nouvel agent permet d'accroître les performances d'un groupe, et (2) l'action d'agent sert à résoudre ou à éviter les conflits. Ces indices de coopération représentent la collaboration entre les agents ainsi que la résolution des conflits.

S'appuyant sur ces points de vue, on peut distinguer deux types de coopération : la coopération intentionnelle (ou celle d'agents cognitifs), où les agents ont l'intention de coopérer, et la coopération réactive.

J.Ferber [Ferber,1995] précise plusieurs méthodes de coopération : le regroupement et la multiplication, la communication, la spécialisation, la collaboration par partage de tâches et de ressources, la coordination d'action ainsi que la résolution de conflit par arbitrage et la négociation. Toutes ces méthodes nécessitent d'être structurées au sein de l'organisation. Cette dernière permet de décrire la manière dont les agents sont positionnés dans un groupe ainsi que les techniques de travail coopératif efficace.

Dans la section ci-dessus on va présenter la plateforme Jade, par laquelle, on va créer et gérer les agents de notre application, qu'on va la présenter à la fin de notre exposé afin de l'enrichir.

4. Outils & plateformes des SMA

Le meilleur moyen pour construire un système multi-agents (SMA) est d'utiliser une plate-forme multi-agents. Cette dernière est un ensemble d'outils nécessaire à la construction et à la mise en service d'agents au sein d'un environnement spécifique. Ces outils peuvent servir également à l'analyse et au test du SMA ainsi créé. Ces outils peuvent être sous la forme d'environnement de programmation (API) et d'applications permettant d'aider le développeur.

Actuellement, il existe plusieurs plates-formes developements des agents et des SMAs parmi ceux on cite :

- **JADE**
- ZEUS
- SWARM
- MOZART
- MADKIT
- AGENT SHEET

On va étudier ci-après la plate-forme **JADE**.

JADE est une plate-forme multi-agents développée en Java par **CSELT** (Groupe de recherche de Gruppo Telecom, Italie) qui a comme but la construction des systèmes multi-agents et la réalisation d'applications conformes à la norme **FIPA**[13] (FIPA, 1997). **JADE** comprend deux composantes de base : une plate-forme agents compatible FIPA et un paquet logiciel pour le développement des agents Java.

4.1 Pourquoi la plateforme JADE ?

On a développé notre application sous la plate forme multi agent JADE qui offre les avantages suivants :

- Plate forme assez facile à mettre en place.
- Disponibilité de packages sur lesquels nous nous sommes appuyés pour développer notre application.
- Documentation claire et complète.
- Licence gratuite.

4.2 La norme FIPA pour les systèmes multi-agents

Les premiers documents de spécification de la norme FIPA, appelés spécifications **FIPA97**, établissent les règles normatives qui permettent à une société d'agents d'inter-opérer. Tout d'abord, les documents FIPA décrivent le modèle de référence d'une plate-forme multi-agents (**FIG 2.6**) où ils identifient les rôles de quelques agents clés nécessaires pour la gestion de la plate-forme, et spécifient le contenu du langage de gestion des agents et l'ontologie du langage.

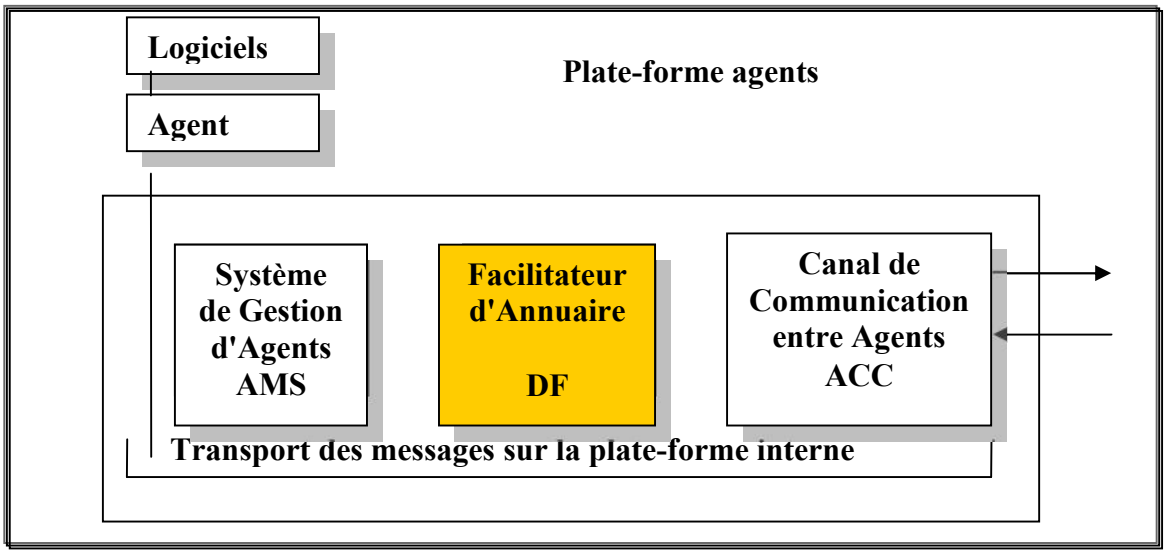

FIG 2.6 : *Le modèle de référence pour une plate-forme multi-agents FIPA*

Dans la **FIG 2.6**, on voit qu'il existe trois rôles principaux dans une plate-forme multi-agents FIPA :

- Le **Système de Gestion d'Agents** (Agent Management System - **AMS**) est l'agent qui exerce le contrôle de supervision sur l'accès à l'usage de la plate-forme; il est responsable de l'authentification des agents résidents et du contrôle d'enregistrements.
- Le **Canal de Communication entre Agents** (**ACC**) est l'agent qui fournit la route pour les interactions de base entre les agents dans et hors de la plate-forme; c'est la méthode de communication implicite qui offre un service fiable et précis pour le routage des messages; il doit aussi être compatible avec le protocole **IIOP** pour l'interopérabilité entre les différentes plates-formes multi-agents.
- Le **Facilitateur d'Annuaire** (**D**irectory **F**acilitator - **DF**) est l'agent qui fournit un service de pages jaunes à la plate-forme multi-agents.

Le standard spécifie aussi le **Langage de Communication d'Agents** (**A**gent **C**ommunication **L**anguage - **ACL**), la communication des agents est basée sur l'envoi de messages. Le langage **FIPA ACL** est le langage standard des messages et impose le codage,

la sémantique et la pragmatique des messages. La norme n'impose pas de mécanisme spécifique pour le transport interne de messages. Plutôt, puisque les agents différents pourraient s'exécuter sur des plates-formes différentes et utiliser technologies différentes d'interconnexion, *FIPA* spécifie que les messages transportés entre les plates-formes devraient être codés sous forme textuelle. On suppose que l'agent est en mesure de transmettre cette forme textuelle.

5. Formalismes de modélisation

La mise au point de tout système doit en principe passer par une étape de modélisation, cette dernière est donc une étape incontournable et doit se baser sur un formalisme de représentation.

Actuellement il existe une multitude de formalisme de représentation, mais certains formalismes sont plus adaptés que d'autres et la tendance actuelle veut privilégier les formalismes graphiques qui sont jugés pus faciles à comprendre. Cela s'applique dans notre cas dans la description des comportements d'agents.

Dans cette situation, on peut citer les formalismes de modélisation des comportements des agents, a savoir AUML (***Agent UML***), le réseau de Pétri (***RDP***) et le CATN (***C****oupled* ***A****ugmented* ***T****ransition* ***N****etwork*) qui se présente sous la forme d'une machine à transition d'états présentant l'exécution en parallèle des différents agents en interaction.

6. Domaines d'application des Systèmes multi-agents

Il y a évidemment plusieurs domaines d'application pour les applications agents dus au fait que les architectures basées sur les agents fournissent une manière bien particulière afin d'aborder des problèmes rapidement. C'est pour cette raison que les agents sont largement utilisés dans les domaines suivants:

L'énergie : l'achat de la puissance d'une manière intelligente, la gestion des réseaux, le support d'un centre de crise.

L'Industrie : l'automatisation des processus et de la production, la logistique, les robots coopératifs.

La Communication (y compris télécommunication): la gestion de réseaux, le commerce électronique, les services du réseau personnel, le calcul mobile.

L'information : l'assistance personnelle, la recherche d'Information, la gestion du workflow.

La médecine : la supervision des malades.

La transportation : la logistique, le support de la mobilité, l'information du voyage**.**

Le grand essor d'Internet a fait apparaître de nouvelles opportunités de développement de SMA tels que le commerce électronique et récemment l'accélération des réseaux.

Le commerce électronique est une forme de commerce dans lequel les transactions financières se déroulent entièrement ou partiellement sur un réseau informatique avec un paiement et une livraison physique ou numérique.

7. Approche agents mobiles

Récemment, la recherche en systèmes répartis a vu l'émergence d'un nouveau modèle pour la structuration d'applications réparties : La programmation par agents mobiles. Les agents mobiles visent principalement des applications réparties sur des réseaux à grande distance, car ils permettent de déplacer l'exécution vers les serveurs et de diminuer ainsi le coût d'accès à ces serveurs [Hagimont,2001]. Le concept d'agent mobile est un peu particulier. Il repose non plus sur le modèle classique client-serveur mais sur un modèle d'exécution distribué.

7.1 Définition d'un agent mobile

Un agent mobile est un agent autonome qui peut se déplacer à travers un réseau hétérogène sous son propre contrôle, migrant d'un site vers un autre et interagissant avec d'autres agents et ressources sur chacun d'eux, puis éventuellement revenir à son site natal une fois sa tâche accomplie [Gherbi,2000].

Un agent mobile est en quelque sorte une capsule contenant des éléments d'authentification et du code; qui est dotée de mécanismes de migration et de négociation. Ainsi, l'agent mobile est capable de suspendre son exécution pour migrer et négocier lui même avec le système cible son exécution (autorisation, contexte, etc.).

7.2 Caractéristiques des agents mobiles

Après avoir évoqué le concept d'agent mobile, nous nous attacherons aux propriétés de mobilité, c'est-à-dire à celles qu'ont les agents qualifiés de mobiles.

7.2.1 Mobilité [Magedanz,1998], [Dimakopoulos,2003]**:**

La mobilité est la première caractéristique d'un agent mobile, qui lui permet de s'exécuter dans différents noeuds d'un réseau. On distingue deux types de mobilité :

• **L'exécution à distance (Remote Execution)** :

L'agent est transféré vers le site de destination avant le début de l'exécution (code et données), il y reste jusqu'à la fin de l'exécution la tâche.

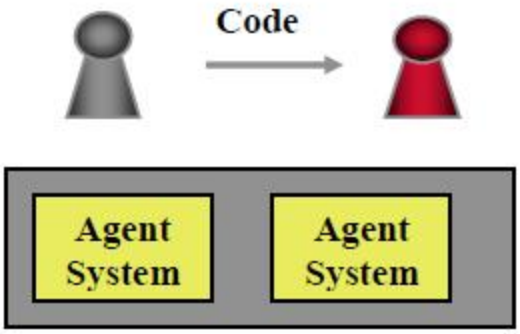

FIG 2.7 : *Exécution à distance (Remote Execution) [Magedanz,1998]*

La migration : Cette migration est réalisée grâce à des performatifs de migration pendant l'exécution de l'agent; leurs invocations provoquant la capture de son état, son encapsulation, et son transfert au site de destination.

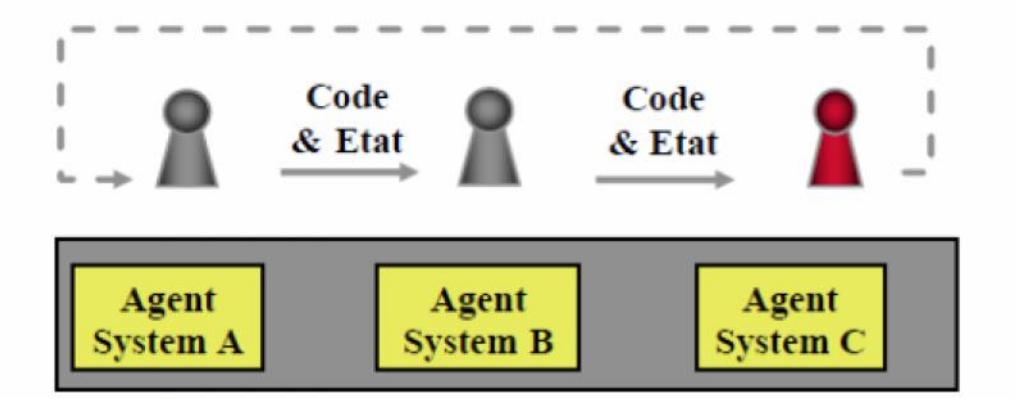

FiG 2.8 : *La migration [Magedanz,1998]*

Il existe deux degrés de migration :

- **Migration forte (orientée Etat) :** Pour une migration forte le contexte de l'agent est déplacé en même temps que son code et ses données. Une fois reconstitué à destination, l'agent continue son exécution à l'instruction qui suit le performatif de mobilité (point d'interruption sur le site de départ).
- **Migration faible (Orientée Non-Etat) :** Elle consiste à faire migrer uniquement le code et les données. Après sa migration, l'agent mobile reprend son exécution dès le début tout en possédant les valeurs mises à jour de ses données. Ce type de migration est souhaitable si l'agent répète une suite de tâches à chaque destination.

Un sous-produit de cette opération de migration est le **clonage** distant, où un double de l'agent (même code et même état) a migré sur un site distant en conservant l'agent original, ces deux agents évoluent ultérieurement d'une façon autonome.

7.2.2 Autonomie :

Un agent agit indépendamment du client. Il est pourvu d'un programme qui lui dicte quelles tâches il doit accomplir. On dit qu'il agit par *délégation* pour le client. Il décide lui-même là où il va et ce qu'il doit y faire, en fonction du comportement qui lui a été donné.

Cela implique que l'on ne peut pas toujours prévoir l'itinéraire des agents et c'est pour cela qu'on utilise des *proxies* pour faire abstraction de leur localisation.

7.2.3 Communication :

Un agent doit pouvoir interagir avec les différents environnements (le client et les hôtes) et les autres agents (locaux ou distants). Les agents mobiles disposent généralement de deux méthodes pour communiquer, la différence résidant dans la localisation des communicants :

• **Orientée Réseau :** C'est un procédé de communication qui s'étend à l'ensemble du réseau, les agents communiquent sans avoir à être sur le même noeud du réseau.

• **Orientée Noeud :** Cette méthode impose aux parties communicantes de résider sur le même noeud, du fait que ce procédé utilise des fichiers, la mémoire partagée ou les *pipes* anonymes.

7.2.4 Coopération :

La capacité qu'ont les agents d'effectuer un travail collaboratif pour accomplir une tâche donnée.

7.2.5 Délibération :

l'agent doit être capable de comprendre son environnement et de prendre une décision seul.

7.2.6 Acquisition des données :

Au cours de leur déplacement, les agents peuvent collecter des informations afin d'arriver à leur objectif. Ces informations sont filtrées pour être stockées ou transmises à une certaine destination (comme le site d'origine). [Baghdad et Zerhouni,2003]

7.2.7 Détermination de la route :

Trois méthodes existent pour la détermination de la destination de l'Agent mobile :

• **Prédéterminée :** Cette méthode est souhaitable quand l'ordre des sites visités est important.

• **Dynamique :** L'agent lui-même décide (comme un grand !) du choix du prochain site à visiter.

• **Hybride :** La méthode utilisée est un amalgame des deux méthodes précédentes.

7.2.8 Pro-Activité :

Ceci caractérise la capacité d'un agent à anticiper des changements plutôt qu'à réagir au changement. Par exemple, il s'agit d'appliquer une règle d'administration particulière dès qu'une variable de l'environnement atteint un certain seuil, c'est-à-dire agir avant que le seuil ne soit atteint. [Reuter, 2000]

7.2.9 Réactivité :

Ceci caractérise la capacité d'un agent à effectuer une nouvelle tâche lorsqu'un événement survient.

7.2.10 Sécurité :

Un agent doit se protéger des attaques extérieures : par rapport aux autres agents, à ses environnements d'exécution et pendant son déplacement sur le réseau. Son intégrité et la confidentialité des données qu'il transporte doivent être assurés. [Esposito,2000]

8. Intérêts de l'approche d'agents mobiles

Les agents mobiles offrent plusieurs avantages améliorant la performance de plusieurs applications distribuées. Cette amélioration peut être ressentie dans l'utilisation du réseau, dans le temps de calcul, dans la commodité par rapport aux programmeurs ou simplement dans l'habilité de continuer l'interaction avec un utilisateur durant une déconnexion du réseau. [Esposito,2000]

• Tout d'abord, les agents mobiles vont permettre de personnaliser le serveur, qui fournit un ensemble fixe d'opérations, pour qu'il réponde à des besoins spécifiques qu'il ne connaît pas forcément à l'avance. [Esposito,2000]

• Réduction du trafic réseau (il y a moins d'utilisation de la bande passante) : Le rapprochement du client (représenté par un agent) et du serveur permet de diminuer les interactions distantes.

En général, il y aura simplement deux échanges : l'envoi d'un agent et la réception des résultats. Le déplacement de l'agent vers le serveur qui héberge les informations (données ou services) auxquelles le client accède fréquemment permet de profiter de la localité des informations et de diminuer le trafic réseau.

• Répartition dynamique de charge : Les agents mobiles des applications distribuées peuvent se déplacer vers de nouveaux sites et profiter ainsi des ressources physiques des sites d'accueil, telles que la mémoire ou le processeur. Un déplacement judicieux des agents permet d'équilibrer dynamiquement la charge à travers le système distribué. [Bouchnak, 2001]

• Exécution asynchrone et autonome (mode déconnecté) : Le principe de délégation permet à l'agent d'opérer en étant déconnecté du client. Une fois l'agent lancé, l'usager peut se déconnecter; la réponse lui sera communiquée lors d'une prochaine reconnexion, ce qui permet de diminuer le temps de connexion au réseau. [Thierno, 2000].

• Tous ces avantages font des agents mobiles une technologie adaptée aux clients mobiles qui ont typiquement des connexions peu fiables à faible débit [Thierno, 2000]. En effet les ordinateurs mobiles souffrent de nombreux problèmes qui, en adoptant l'approche d'agents mobiles, se règlent avec flexibilité [Harrison, 1995]:

- Ils sont souvent déconnectés du réseau : grâce à l'exécution asynchrone et autonome des agents mobiles, le client mobile peut surmonter ce problème.
- Faible bande passante : Quand un agent mobile se déplace vers le site serveur, il peut effectuer et la consultation et le filtrage des données, ensuite revenir au client sur l'ordinateur mobile avec uniquement l'information utile. Ceci réduit le volume des informations circulant sur le réseau et répond bien au problème de la faiblesse de la bande passante.
- Faible capacité de calcul : En plus, en se déplaçant d'un ordinateur mobile vers un ordinateur fixe, l'agent mobile peut profiter des capacités importantes de calcul et de stockage offertes par ce dernier.

• Neutralisation de la latence réseau ce qui facilite les interactions temps réel avec le serveur. Par exemple lorsqu'un serveur gère un équipement externe (un robot), et si la durée de transmission à travers le réseau est plus grande comparée aux contraintes (en temps réel) imposées par cet équipement, alors, un programme déclenché par le client pour contrôler l'équipement doit se déplacer pour s'exécuter de préférence sur le serveur. Ce schéma d'exécution est naturellement supporté par les agents mobiles [Gherbi,2000].

• Administration du système : L'administration des systèmes distribués nécessite parfois la réinitialisation des machines sans pour autant vouloir interrompre certains services (applications) en cours d'exécution sur ces machines. Les agents mobiles permettent la reconfiguration dynamique d'applications en transférant l'exécution de ces applications vers d'autres machines où ils reprendront leur exécution. [Bouchnak, 2001]

• Les agents étant dynamiques, ils peuvent évoluer au cours du temps : ils peuvent apprendre des choses et les ramener au ``client'', leur stratégie tant du coté client que serveur peut évoluer, ils n'agissent pas de manière automatique mais en fonction de leur passé.

Tolérance aux pannes : La mobilité des agents et leur autonomie améliorent la sûreté par redéploiement des agents sur des noeuds en service. [Leriche, 2004]

En ce qui concerne les performances, dans certains types d'applications et à partir d'un certain seuil (puisqu'il faut tenir compte de la taille de l'agent en lui-même), les agents mobiles deviennent beaucoup plus intéressants que le modèle client/serveur.

De plus, les clients peuvent programmer les serveurs et réciproquement; ce qui étend largement les fonctionnalités que les développeurs d'applications et de serveurs peuvent fournir à leurs consommateurs [Gherbi,2000].

9. Les domaines d'applications des agents mobiles

Tout ce qui est fait en mode client/serveur pourra être fait avec des agents mobiles. Une partie des applications ne nécessiteront pas ce nouveau modèle, mais certaines se révèleront plus efficaces en l'utilisant. Voici quelques exemples :

9.1 La recherche d'information

La recherche d'information est une application les plus souvent mise en avant lorsque l'on parle d'agents mobiles. En effet, la quantité d'information disponible sur le web croît sans cesse. Les outils de recherches disponibles deviennent inadaptés car ils ne peuvent pas traiter une grande quantité de données ni de modifier la requête. Les agents mobiles peuvent visiter plusieurs sites web, coopérer et trouver les sites d'intérêts et revenir avec les meilleurs résultats, on peut même modifier la requête en lançant un nouvel agent qui va informer ses collègues; [Thierno, 2000]

9.2 La distribution d'information

Un autre champ d'application des agents mobiles est la distribution automatique d'information vers un ensemble d'utilisateurs.

9.3 Le commerce électronique

Avec le développement du commerce électronique, l'utilisation d'un agent qui fait des courses est de plus en plus envisageable. Un agent mobile peut être utilisé pour fournir un accès personnalisé au magasinage en ligne. L'usager peut sélectionner un profil qui correspond à ses habitudes d'achat ou choisir des produits et lancer un agent qui va s'occuper de faire le magasinage.

La négociation : l'agent part avec une offre du client et revient avec le meilleur prix obtenu. [Thierno, 2000] .

9.4 Le calcul parallèle

Un calcul parallèle peut s'exécuter facilement sur plusieurs machines en utilisant des agents mobiles : Il suffit d'envoyer des agents effectuer de petites paries du calcul sur des machines différentes. Ainsi on utilise plusieurs ordinateurs pour effectuer le calcul et le résultat est obtenu plus rapidement. Cela permettra d'équilibrer la charge du système. [Menacer, 2004]

9.5 L'administration des réseaux

Les agents mobiles peuvent effectuer tous les traitements nécessaires sur l'information recueillie et la transmettre de façon concise à la station d'administration. Celle-ci n'a plus besoin d'être particularisée et ne nécessite plus d'être sécurisée puisque n'importe quel système du réseau peut devenir à tout moment la station de supervision. De plus, la CPU utilisée pour le traitement des informations se trouve répartie sur les équipements du réseau et non plus sur une station unique.

Par un traitement déporté des données, la taille de l'agent mobile en transit reste suffisamment petite pour sauver de la bande passante et éviter le goulot d'étranglement connu des stations d'administration centralisées. [Reuter, 2000]

9.6 La surveillance

Par fois le problème n'est pas de savoir où se trouve l'information mais quand elle va s'y trouver.

Dans ce cas, les agents mobiles peuvent aussi se révéler utiles.

Prenons l'exemple des cours boursiers : Les cours boursiers sont disponibles sur Internet (Bourse de New York, Yahoo – Finance, etc.). On sait donc où est l'information, il faut maintenant savoir quand l'information va devenir intéressante. Ainsi, on peut envoyer un agent mobile sur le serveur de la bourse attendant qu'une action atteigne un certain cours puis de revenir faire un rapport. [Menacer, 2004]

9.7 Télécommunication

Les architectures actuelles sont basées sur les réseaux intelligents. Actuellement, le principal avantage de la téléphonie sur Internet est son coût peu élevé. Cependant la différence de prix devient de moins en moins significative au niveau des services offerts (ex. redirection d'appel).

Les agents mobiles agiraient comme des dossiers qui transporteraient les services auxquels est abonné l'usager. Au lieu d'avoir à gérer les services un par un, on pourrait gérer tous les services avec un seul agent mobile. [Thierno, 2000]

10. Langages de programmation des agents mobiles

La notion d'agent mobile pose un problème d'intégration dans les langages. L'agent est une partie de l'application qui va de par sa mobilité être amenée à s'en détacher. Le système doit gérer l'isolement de l'agent de manière à ce qu'il devienne autonome pour sa mission. [Perret, 1997] Les systèmes d'agents mobiles sont basés sur des langages qui supportent la mobilité du code.

Parmi ces langages, Java, Obliq, Telescript et Tcl. Mai Java reste le langage le plus utilisé. Grâce au concept de la machine virtuelle, Java est portable sur tous les systèmes d'exploitation connus.

Avec Java, les agents mobiles peuvent migrer sur des sites hétérogènes. Cela permettra d'étendre la portée des agents mobiles sur des réseaux de grande envergure comme l'Internet.

L'une des fonctionnalités la plus intéressante qu'offre Java est la **sérialisation.** Ce mécanisme permet de capturer et de restaurer l'état des objets avant et après migration.

Cependant, Java ne permet la migration forte. C'est à la charge du programmeur après sérialisation des objets nécessaires, d'empaqueter l'état et les données variables avant la migration. L'agent est régénéré sur le site distant et le processus (thread) correspondant est redémarré avec les informations empaquetées. La dé-sérialisation permet alors de retrouver les objets accédés par le processus. [Menacer, 2004]

11. Les Systèmes d'agents mobiles

11.1 Définition

Un système d'agents mobiles est un environnement de développement et d'exécution d'applications réparties à base d'agents mobiles.

Toute machine hôte susceptible d'accueillir un agent doit disposer d'un environnement d'exécution qui permet aux agents de s'exécuter et d'accéder aux différents services de l'hôte. Un système d'agents mobiles fournit une interface de programmation et un environnement d'exécution offrant des mécanismes de création et d'hébergement locaux d'agents. Afin de pouvoir être informé des variations de l'environnement. [Leriche, 2004]

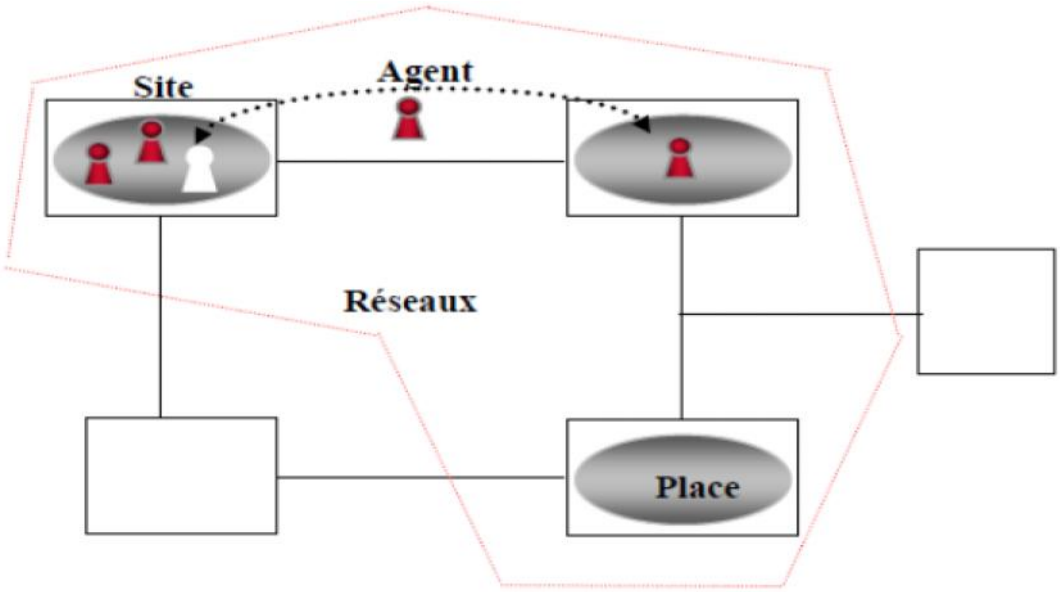

FIG 2.9 : *Environnement d'exécution d'agents mobiles*
Dans un système distribué. [Sahai, 1999]

Un environnement d'exécution d'agents mobiles est constitué d'un ensemble de programmes statiques, appelés *places*, s'exécutant sur les sites du système susceptible d'accueillir des agents.

Les places sont des programmes qui fournissent aux agents l'infrastructure de base pour leur exécution et leur permet de se déplacer. [Sahai, 1999]

11.2 L'environnement d'un système d'agents mobiles

Les systèmes d'agents mobiles doivent fournir un support système pour l'exécution, la communication, la migration et la prise en compte de la sécurité.

11.2.1 Support d'exécution

Pour pouvoir s'exécuter sur un site, un agent doit être admis et des ressources doivent lui être allouées. Le support d'exécution doit permettre la création, l'enregistrement et la réception de l'agent (code, état et attributs) ainsi que l'activation du code (lancement de l'exécution). Il doit également fournir l'accès aux ressources locales du système hôte et de prévoir la gestion des erreurs afin que le propriétaire de l'agent soit informé de sa terminaison normale.

11.2.2 Support de communication

On identifie trois types de communication dans un système d'agents mobiles :
la communication d'un agent mobile avec un agent statique, la communication entre agents mobiles et la communication entre un agent et l'environnement client.

La communication entre agents mobiles pose le problème du rendez-vous d'entités mobiles et nécessite des mécanismes de localisation et de notification pour réaliser la mise en correspondance des agents. [Perret, 1997] La communication entre l'environnement client et l'agent qui a été délégué pour une action est peu abordée. Les systèmes utilisent souvent la communication classique par échange de messages ou proposent d'attendre que l'agent revienne dans l'environnement client pour communiquer localement. Certains systèmes proposent d'utiliser un langage de communication uniforme entre agents pour la collaboration appelé KQML (Knowledge Query and Manipulation Language). [Perret, 1997]
Dans le cas de la communication avec un agent ou un service distant, le choix entre la communication distante et la migration sur le site distant est un problème complexe (compromis entre le coût de la communication et le coût de la migration).

11.2.3 Support de migration

Tout environnement d'agents mobiles fournit une primitive de migration volontaire de l'agent sur le site distant (migration proactive). C'est l'agent qui détermine quand, et où, se déplacer. Le système local doit fournir un support pour réaliser cette opération : emballage de l'agent (code, état, attributs), envoi sur le réseau, gestion des erreurs. En cas de succès, les ressources occupées localement par l'agent sont libérées.

Le premier problème posé par la migration est la finesse de la capture de l'état de l'agent.

En effet, celui-ci est constitué de l'état d'exécution et des données ; le programmeur doit donc préserver dans les données les informations d'état permettant la poursuite de l'exécution au point d'avancement atteint ce qui est une contrainte importante.
Le deuxième problème est relatif à l'utilisation par l'agent de ressources telles que fichiers ou bases de données. Idéalement l'accès à ces ressources devrait pouvoir être conservé après

migration, mais la nature des ressources (transférables ou pas) et la forme de liaison avec l'agent (par identificateur, par valeur ou par type) sont souvent imposées par la définition ou l'implémentation du langage ce qui limite les possibilités.

Le système doit supporter également, un mécanisme de localisation (Traçabilité) et de contrôle permettant de terminer l'exécution d'agents dispersés afin d'éviter la consommation inutile de ressources (par exemple lorsqu'un critère d'arrêts de recherche et satisfait).

11.2.4 Support de sécurité

Le sens de sécurité doit être entendu dans le sens le plus large, englobant les notions d'autorisation, d'authentification, de protection et de confidentialité.

Le site d'accueil d'un agent doit se protéger contre la sur-utilisation des ressources et les actions malveillantes des agents accueillis. Ceci suppose l'identification du propriétaire, du site de provenance et la vérification du code. Des limitations sur la consommation des ressources peuvent également être fixées.

Inversement, l'agent doit avoir l'assurance que ni son code ni ses données ne seront modifiés par le système hôte.

11.3. Les systèmes d'agents mobiles existants

La première plate-forme d'agents mobiles est Telescript, née dans la première moitié des années 90. Elle a depuis donné lieu à de nombreux descendants, extensions du langage Java (Aglets, Voyager, etc.). Nous allons présenter quatre choisies en raison de leur maturité.

Ces plateformes diffèrent dans leurs buts, motivations et implémentations, mais elles fournissent toutes des fonctionnalités communes qui supportent : la migration, la communication entre agents, plusieurs langage de programmation et plusieurs formes de sécurité.

11.3.1 Ara (Agents for Remote Action)

Ara est un système multi-langage (C,C++ et Tcl). Sur chaque machine susceptible d'accueillir des agents ARA, doit être lancé un interpréteur Tcl.

ARA supporte la migration proactive avec mobilité forte et la gestion des exceptions, ainsi que le clonage. L'accès aux ressources externes est perdu après migration d'un agent.

Les agents peuvent communiquer entre eux par la notion « point de service ». Un point de service est créé et publié par tout agent souhaitant recevoir des messages, cet agent joue alors le rôle de serveur pour ce point de service, la communication entre l'agent serveur et les agents clients est synchrone.

Aucune limitation n'existe pour les sites de provenance des agents, pour l'accès au système de fichiers des sites visités et pour l'exécution de processus locaux.

11.3.2 D'Agent

D'Agent a été développé à l'université de Dartmouth, initialement sous le nom de Agent-Tcl. La conception et l'implantation étant voisine de celle D'ARA. Sur chaque machine susceptible de créer ou d'accueillir des agents, trois processus doivent être lancés, l'un exécutant l'interpréteur Tcl, le deuxième étant chargé de l'accueil des agents externes et le dernier du contrôle de l'utilisation des ressources. D'Agents permet une migration forte et proactive, le clonage, le suivi et le contrôle des agents dans le système. [Guy, 1999]

Le système D'Agent propose un mécanisme de localisation utilisant un intermédiaire statique qui maintient la localisation d'un agent. L'agent qui veut être contactable à tout moment notifie sa localisation à cet intermédiaire lors de chaque migration. Les agents peuvent communiquer entre eux (un à un) par passage de messages de manière synchrone ou asynchrone. [Per97] La limitation de la consommation des ressources s'effectue par l'intermédiaire d'un fichier de configuration utilisé par les sites serveurs à l'initialisation du système. [Cfs99]

11.3.3 Aglet

Aglets (contraction de « agent » et « applet ») est le nom donné par Tokyo research Laboratory d'IBM à un environnement de programmation et d'exécution d'agents mobiles écrits en langage Java. Cet environnement comporte un ensemble de classes Java pour le support de la mobilité et de la communication. Sur chaque machine susceptible d'accueillir des Aglets, un serveur doit être lancé pour fournir l'environnement d'exécution des Aglets.

Aglets supporte la migration proactive avec faible mobilité. Ceci provient de l'architecture actuelle de la machine virtuelle Java, qui ne permet pas à un programme d'accéder directement à son état d'exécution.

La communication entre les agents sur le même site ou sur des sites différents peut être synchrone ou asynchrone. La gestion des messages est très souple, permettant par exemple d'affecter des priorités différentes aux messages suivant leur type. Enfin, le multicast est possible pour un ensemble d'Aglets situés sur le même site.

La sécurité des Aglets bénéficie des mécanismes de sécurité propres à Java (vérification du bytecode avant son exécution). Sur chaque site, les serveurs d'Aglets ont la possibilité de définir des droits d'accès au système de fichiers, les droits d'utilisation des réseaux, les droits d'utilisation des protocoles d'invocation de méthodes distantes. Il n'est pas possible en revanche de limiter l'accès au ressources CPU et mémoire à l'impossibilité d'accéder à l'état d'exécution. [Guy, 1999]

11.3.4 Voyager

Voyager est développé par la société ObjectSpace de Dallas. C'est un environnement de développent d'applications réparties à base d'objets Java. Le principe de mise en place pour un Agent créé par Voyager est similaire à celui des Aglets d'IBM.

Dans Voyager, un Agent est une classe qui hérite de la classe Agent de l'API Voyager. Celui-ci est identifiable par son nom et par une référence virtuelle. Les Agents peuvent communiquer grâce à différents types de messages.

Voyager supporte seulement la migration faible. La principale différence avec Aglets est que le nom de la méthode utilisée comme point d'entrée pour la reprise de l'exécution après la migration n'est pas imposée par le système, mais passé en paramètre de la primitive de migration.

La communication par messages entre agents est asynchrone et « oneway » (aucune réponse attendue). Le multicast est possible, y compris, entre agents distants. [Guy, 1999]

12. Les principaux problèmes des agents mobiles

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

12.1 La sécurité

Garantir la sécurité dans les environnements d'agents mobiles est important du fait de la nature du modèle d'exécution des agents mobiles et du fait des interactions des agents mobiles avec plusieurs systèmes et ressources. Quatre types de problèmes de sécurité ont été identifiés pour les environnements d'exécution d'agents mobiles [Sahai, 1999] :

1. sécurité site-agent
2. sécurité inter-agent
3. sécurité inter-sites
4. sécurité entre les sites et les parties tierces non autorisées

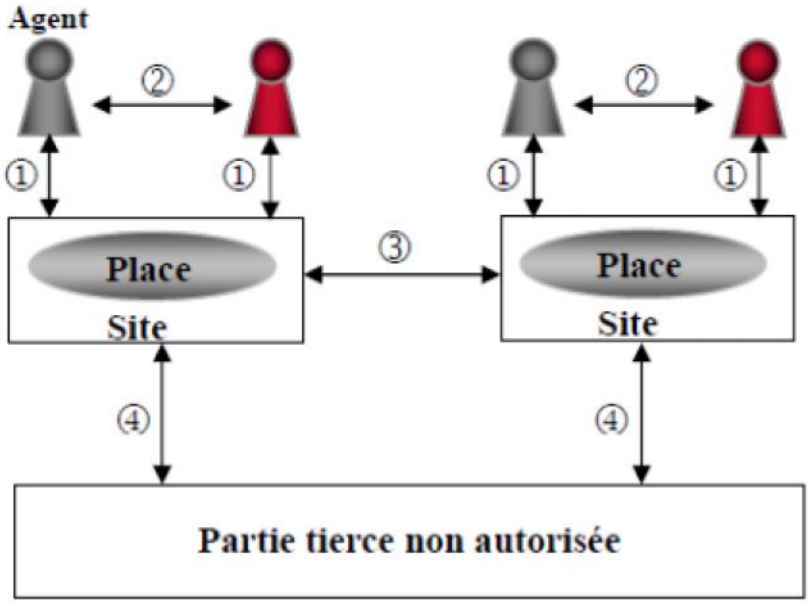

FIG 2.10 : *Illustration des problèmes de sécurité dans les environnements d'agents mobiles [Sahai, 1999]*

Les agents mobiles, par leur nature, sont exposés à des risques de sécurité spécifiques en plus des risques usuels. Chaque agent doit se protéger des attaques extérieures : par rapport aux autres agents, à ses environnements d'exécution et pendant son déplacement sur le réseau. Son intégrité et la confidentialité des données qu'il transporte doivent être assurées. Par exemple on ne doit pas modifier son programme ni de pouvoir pratiquer la mutation d'un agent à son insu. De ce fait, les agents doivent être créés par des outils sécurisés. [Bag03]

Les agents mobiles peuvent eux-mêmes être perçus comme une menace à la sécurité des systèmes si aucun mécanisme n'est prévu pour y remédier. En effet, le concept d'agent mobile pose un problème d'identification de la source des données, car cette identification est fournie par l'agent lui-même et non par le protocole réseau sous-jacent puisque l'entité a pu visiter de nombreux hôtes avant d'arriver sur un nouveau. On imagine aisément les dangers que représente l'introduction pernicieuse d'agents ``pirates'' au sein d'un tel système.

L'environnement d'exécution doit se protéger des attaques, par exemple d'agents virus.

Il nous faut donc un mécanisme d'authentification des agents pour gérer les droits qu'ils auront sur l'environnement. Cela permettra notamment à un hôte d'interdire l'accès à certaines informations pour tous les agents n'appartenant pas à un groupe autorisé.

12.2 L'interopérabilité

Les travaux sur les agents mobiles ont été jusqu'ici déconnectés les uns des autres [Guy, 1999].

Les plateformes qui voient le jour, bien que de plus en plus souvent basées sur Java, restent incompatibles entre elles. Plus exactement un agent d'une plateforme ne peut pas s'exécuter sur une des autres. Si cela ne pose pas de problème dans un univers « fermé », il est tout autrement dans le cas d'Internet. Un tel réseau est un environnement ouvert, peuplé d'applications fort hétérogènes, des services divers, mais surtout des plateformes d'exécution d'agents incompatibles. [Magnin ,1999]

En effet, que faire si l'on veut par exemple, concevoir un agent destiné à trouver sur le réseau un billet d'avion aux meilleures conditions entre Montréal et Paris et que le serveur d'Air France et d'Air Canada imposent de recevoir des agents devant s'exécuter sur des plateformes incompatible entre elles ? Dans ce cas, il semble bien difficile d'imposer une modification d'au moins un des deux serveurs…Faudrait-il prévoir dès lors deux types d'agents, voire plus si d'autres serveurs de compagnies aériennes sont aussi incompatibles ?

Une façon de contourner cette difficulté serait de prévoir une norme capable de s'imposer sur ces différents « serveurs d'agents » afin de permettre l'interopérabilité des différentes architectures d'agents mobiles. [Magnin ,1999]

Dans le domaine de l'interopérabilité entre les agents, il existe deux standards : Mobile Agent System Interoperability Facility (MASIF) de l'OMG (l'organisme qui gère la norme CORBA), et Foundation of Intelligent Physical Agents (FIPA). Mais les plates-formes à agents mobiles qui se conforment à ces normes sont rares.

Conclusion

Les SMA, nouveau paradigme, semblent ouvrir une nouvelle voie pour la conception future d'applications réparties grâces aux multiples avantages qu'ils offrent notamment leurs qualités asynchrones et leur utilisation modérée des ressources du réseau. [Menacer, 2004]

Les agents mobiles visent principalement des applications réparties sur des réseaux à grande distance, car ils permettent de déplacer leur exécution vers d'autres serveurs et de diminuer ainsi le coût d'accès à ces serveurs. Récemment la technologie d'agents mobiles a suscité beaucoup d'intérêt. Par conséquent, le développement de systèmes d'exécution d'agents mobiles est actuellement en plein essor [Sahai, 1999].

Or, le manque de standard a entraîné le développement de nombreuses plateformes qui ne sont pas compatibles, et les agents ne sont pas forcément conçus avec le même langage et ne peuvent de ce fait pas communiquer. Ces deux derniers points nécessitent de rendre les plateformes compatibles et interopérables [Bouchnak, 2001].

La conscience de ce besoin a poussé l'OMG (Object Management Group) à définir les spécifications de MASIF (Mobile Agent System Interoperability Facilty) pour l'interopérabilité entre les différents systèmes à agents mobiles. Un autre effort est lancé par FIPA (Foundation for Intelligent Physical Agents), afin de spécifier l'architecture ainsi que les sémantiques de communications entre des agents hétérogènes; l'interopérabilité reste un but important [Hagimont,2001].

Chapitre **III**

Web Sémantique & ontologies

1. Introduction

Le Web sémantique (plus techniquement appelé « le Web de données ») permet aux machines de comprendre la sémantique, la signification de l'information sur le Web. Il étend le réseau des hyperliens entre des pages Web classiques par un réseau de lien entre données structurées permettant ainsi aux agents automatisés d'accéder plus intelligemment aux différentes sources de données contenues sur le Web et, de cette manière, d'effectuer des tâches (recherche, apprentissage, etc.) plus précises pour les utilisateurs. Le terme a été inventé par Tim Berners-Lee, co-inventeur du Web et directeur du W3C, qui supervise l'élaboration des propositions de standards du Web sémantique.

La plupart du temps, lorsque l'on prononce le terme de Web sémantique, on parle des différentes technologies qui se cachent derrière. Parmi les plus connues, on peut citer RDF (Ressource Description Framework) qui correspond à un modèle d'information, et les formats d'échanges de données en RDF pour communiquer entre différentes applications (RDF/XML, RDF/JSON, N3, Turtle, N-Triples et d'autres). Dans le domaine du Web sémantique, la sémantique des données est décrite par des ontologies - ce terme sera défini plus loin dans l'article - avec des langages prévus pour fournir une description formelle de concepts, termes ou relations d'un domaine quelconque. Ces langages sont RDFS (Ressource Description Framework Schema) et OWL (Web Ontology Language). Il existe aussi des langages de description des données structurées dans du XHTML afin que des outils effectuent un traitement automatique de ces différentes données. Ces langages sont RDFa et Microformat et, nouvellement arrivé avec HTML 5, Microdata. Voici d'ailleurs un article qui vous introduit le langage RDFa et un autre sur les Microdata. Ensuite, pour finir avec la liste des technologies, il existe un langage de requête, au même titre que SQL pour les bases de données relationnelles, SPARQL, qui effectue des requêtes mais sur des triplets RDF. Il en existe d'autres (RQL et RDQL), mais ils sont bien moins utilisés.

2. Histoire du Web Sémantique

En 1994, lors de la première conférence WWW à Genève, plus précisément au CERN, a lieu l'annonce de la création du W3C. C'est d'ailleurs à cette période que Tim Berners-Lee dresse les objectifs du W3C et montre les besoins d'ajouter de la sémantique au Web futur. Il montre alors en quoi les liens hypertextes ou, plus précisément, la façon dont on met en relation les documents sur le Web est trop limitée pour permettre aux machines de relier automatiquement

les données contenues sur le Web à la réalité. Compte tenu de l'ambition d'un tel projet, cette idée suscite quelques résistances et controverses qui sont classiquement rencontrées dès qu'on aborde des problématiques liées au domaine de l'intelligence artificielle.

Mise à part la mise en place des recommandations nécessaires à la construction des documents, le W3C nouvellement créé entame les premières réflexions sur la mise en place du Web sémantique. Ces réflexions aboutissent à la publication d'un premier draft de recommandations sur le Web sémantique en octobre 1997 et d'une seconde en avril 1998. Cette même année, Tim Berners-Lee publie un document sur les toutes premières moutures de ce qui sera plus tard appelé le Web sémantique. Ces moutures consistent à mettre en place les différentes technologies du Web sémantique. Dans ce document, il présente le Web sémantique comme une sorte d'extension du Web des documents, qui constitue une base de données à l'échelle mondiale, afin que toutes les machines puissent mieux lier les données du Web. Cette feuille de route se matérialise par une représentation graphique, le « layer cake », qui montre l'agencement des différentes briques technologiques du Web sémantique.

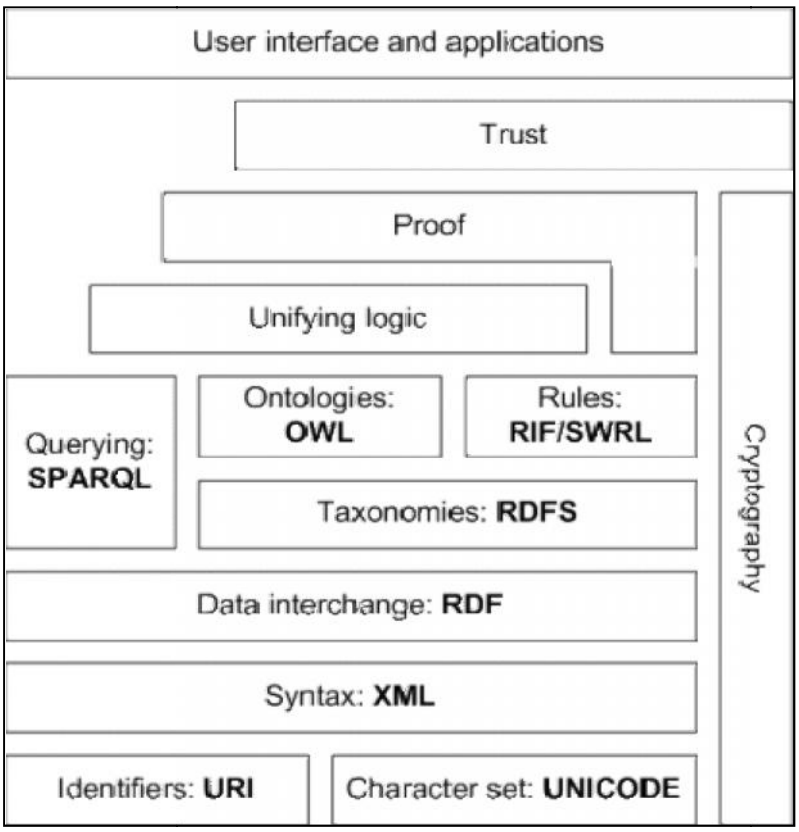

Figure 3.1 .layer cake

2.1Objectifs du Web Sémantique

Un des principaux objectifs du Web sémantique est de permettre aux utilisateurs d'utiliser la totalité du potentiel du Web : ainsi, ils pourront trouver, partager et combiner des informations plus facilement. Aujourd'hui tout le monde est capable d'utiliser des forums, d'utiliser des réseaux sociaux, de chatter, de faire des recherches ou même d'acheter différents produits. Néanmoins, il serait mieux que la machine fasse tout ceci à la place de l'homme, car actuellement, les machines ont besoin de l'homme pour effectuer ces tâches. La raison principale est que les pages Web actuelles sont conçues pour être lisibles par des êtres humains et non par des machines. Le Web sémantique a donc comme principal objectif que ces mêmes machines puissent réaliser seules toutes les tâches fastidieuses comme la recherche ou l'association d'informations et d'agir sur le Web lui-même.

3. Les ontologies

Les années 80 ont vu le développement de nombreux Systèmes Experts (SE) réalisant des tâches variées (conception, maintenance) dans des domaines également variés (médecine, ingénieries mécanique et électronique, robotique, finance). L'expérience de leur développement a toutefois montré que la construction d'une Base de Connaissances était un processus complexe et nécessitant un temps considérable. Le souhait des développeurs a été dès lors de pouvoir **réutiliser** et **partager** des BCs ou, au moins, des parties de **BC**s.

Cette question de la réutilisation et du partage de BCs est donc difficile et implique plusieurs dimensions. Au début des années 90, des chercheurs réunis au sein du projet américain « Knowledge Sharing Effort », soutenu notamment par la **DARPA**[1] , décident de s'attaquer au problème en privilégiant la **représentation explicite du sens**. Ils nomment « ontologie » une telle représentation.

3.1 Définitions

Ontologie : Ce terme est un emprunt à la philosophie, car il désigne déjà [P.ROB, 79] : *la partie de la métaphysique qui s'applique à l'être en tant qu'être, indépendamment de ses déterminations*.

L'Ontologie, en tant que discipline, se donne pour objectif de caractériser les différents modes d'existence des objets, suivant les espèces d'objets (naturels, artificiels, esthétiques, etc.). On retrouve bien ce souci de caractériser les concepts dans différents domaines.

[1] **DARPA**: Defense Advanced Research Projects Agency.

Définition

*une **ontologie** est une spécification explicite d'une conceptualisation*. [GRUB 93]

Le terme « **conceptualisation** », dans la définition, fait référence à un système de concepts, autrement dit à un ensemble structuré de concepts. L'expression « **spécification explicite** » signifie, pour sa part, que la conceptualisation est représentée dans un langage, ce dernier peut être une langue naturelle (français, anglais) ou un langage formel (logique du 1er ordre, réseau sémantique).

3.1.1 Nature d'un concept

Selon une tradition remontant à Aristote, un concept peut se définir comme une entité composée de trois éléments distincts :

- Le(s) terme(s) exprimant le concept en langue.
- La signification du concept, appelée également « notion » ou « intension » du concept.
- Le(s) objet(s) dénotés par le concept, appelé(s) également « réalisation » ou « extension » du concept.

Ces éléments sont habituellement présentés comme constituant les sommets d'un « triangle sémantique ».

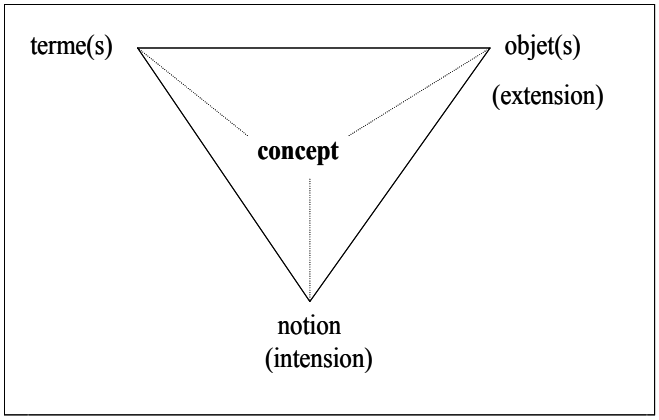

Figure 3.2 : *Le triangle sémantique*

Prenons l'exemple du concept ÉTOILE :

- Son terme est le nom commun *étoile*.
- Sa notion est *d'être un point brillant dans le ciel, la nuit*.
- Son extension est justement *l'ensemble des points brillants* que l'on peut découvrir dans un ciel nocturne … dégagé !

Le concept ÉTOILE a pour extension un ensemble d'objets. Nous le nommons pour cette raison **concept générique**. Il faut noter que le triangle sémantique permet de rendre

compte également de **concepts individuels**, ayant pour extension un unique objet. Pour s'en persuader, on peut prendre l'exemple du concept ÉTOILE DU SOIR :

- Ce concept s'exprime par le syntagme (groupe de mots) *étoile du soir*.
- Sa notion est *d'être la première étoile à briller dans le ciel, le soir*.
- Son extension est … *la planète Vénus*.

Cette clarification de la notion de concept permet du même coup de clarifier la notion d'ontologie :

- Une ontologie est concernée en priorité par les **notions**, puisque l'objectif est de partager du sens. Il faut toutefois préserver un lien avec le(s) terme(s).
- Une ontologie concerne principalement les **notions de concepts génériques** qui ne dépendent pas de situations particulières du monde (l'équivalent d'une base de faits pour un SE) et sont donc plus susceptibles d'être partagées.

La nouvelle définition ci-dessous tient compte des précisions qui viennent d'être apportées :

Définition plus précise :

Une **ontologie** est la spécification explicite d'un ensemble de notions de concepts génériques.

Pour gagner encore en précision et permettre d'appréhender ces objets, les visualiser et les manipuler, il reste à voir comment – pratiquement - on peut spécifier une notion.

3.1.1.1 Spécification d'une notion

Faisons pour cela un détour par les dictionnaires dont la fonction est – justement – de rendre compte du sens des mots de la langue. Ci-dessous, nous avons rassemblé quelques définitions extraites du [P.ROB, 79] :

- Un « camion » est un véhicule automobile de grande taille transportant des marchandises.
- Un « cartable » est un sac d'écolier.

On peut observer dans la forme de ces définitions une régularité : le concept à définir (nous l'avons placé, plus exactement son terme, entre guillemets) est systématiquement défini par rapport à un autre concept (nous avons souligné son terme) en précisant une différence ; cette différence correspond à des propriétés vérifiées par les objets réalisant le nouveau concept (ex : CAMION) et que ne partagent pas les objets réalisant le concept de référence (ex : VÉHICULE AUTOMOBILE).

Ce mode de définition relève en fait d'une longue tradition, que l'on peut faire remonter à Porphyre. Pour classer l'ensemble des objets du monde en catégories abstraites, Porphyre avait adopté le mode de définition suivant :

nouveau genre = genre proche + différence.

L'arbre de Porphyre (FIG 06) peut être considéré comme un premier exemple d'ontologie.

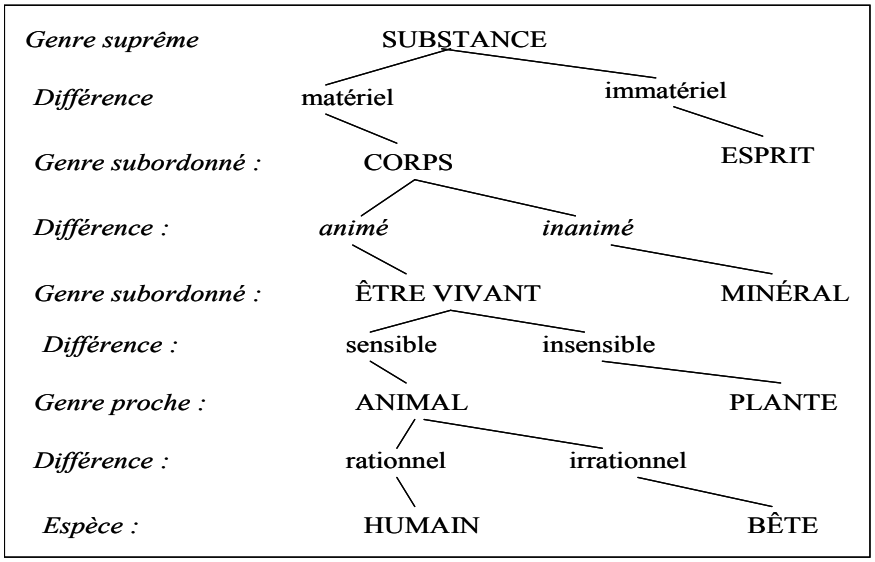

Figure 3.3 : *L'arbre de Porphyre*

Définir un concept par rapport à un autre revient à introduire un ordre sur les concepts. Cette **relation**, exprimée en français par l'expression « **est un(e)** », est appelée relation de « **généralisation** », et son inverse relation de « **spécialisation** ». L'existence d'un tel lien entre concepts se justifie par l'existence d'une relation d'inclusion entre les extensions de ces concepts.

Définition

Le concept CONCEPT1 **généralise** le concept CONCEPT2 (resp. le concept CONCEPT2 **spécialise** le concept CONCEPT1) si et seulement si l'extension du concept CONCEPT2 est incluse dans l'extension du concept CONCEPT1.

Dans l'arbre de Porphyre, le concept ÊTRE VIVANT généralise le concept ANIMAL, ce qui signifie que tout animal est un être vivant. Il faut noter qu'au fur et à mesure que l'on descend dans l'arbre, on ajoute des propriétés (les différences), ce qui a pour effet de réduire la taille des ensembles d'objets réalisant les concepts.

Le terme « subsomption » (encore un emprunt à la philosophie !) tend à remplacer le terme « généralisation » dans le domaine de l'ingénierie ontologique, concerné par la construction d'ontologies.

On parlera ainsi de « liens de subsomption » et on dira, dans le cas de l'arbre de Porphyre, que le concept ÊTRE VIVANT « subsume » le concept ANIMAL, ou inversement, que le concept ANIMAL « est subsumé par » le concept ÊTRE VIVANT.

Ces relations permettent de rendre compte d'autres propriétés vérifiées par les objets réalisant le concept et de contribuer ainsi à exprimer le sens des concepts. De cette remarque, nous déduisons la définition suivante pour la notion de « notion » :

Définition

La **notion** d'un concept est un ensemble de propriétés satisfaites par les objets réalisant le concept.

3.2 Construction d'une ontologie opérationnelle

Le processus général de représentation des connaissances peut donc être découpé en 3 phases (FIG 07) :

La conceptualisation : identification des connaissances contenues dans un corpus représentatif du domaine. Ce travail doit être mené par un expert du domaine, assisté par un ingénieur de la connaissance ;

L'ontologisation : formalisation, autant que possible, du modèle conceptuel obtenu à l'étape précédente. Ce travail doit être mené par l'ingénieur de la connaissance, assisté de l'expert du domaine ;

L'opérationnalisation : transcription de l'ontologie dans un langage formel et opérationnel de représentation de connaissances. Ce travail doit être mené par l'ingénieur de la connaissance.

Il est à noter que ce processus n'est pas linéaire et que de nombreux **aller-retour** sont a priori nécessaires pour bâtir une ontologie opérationnelle adaptée aux besoins. Remarquons pour finir que ce modèle de construction d'ontologie est ascendant, c'est-à-dire que l'on part des connaissances à représenter, pour aboutir à une représentation formelle. Mais une construction descendante est possible et qui consiste à choisir un modèle opérationnel de représentation, en fonction de l'objectif d'utilisation de l'ontologie, puis à instancier ce modèle avec les connaissances du domaine.

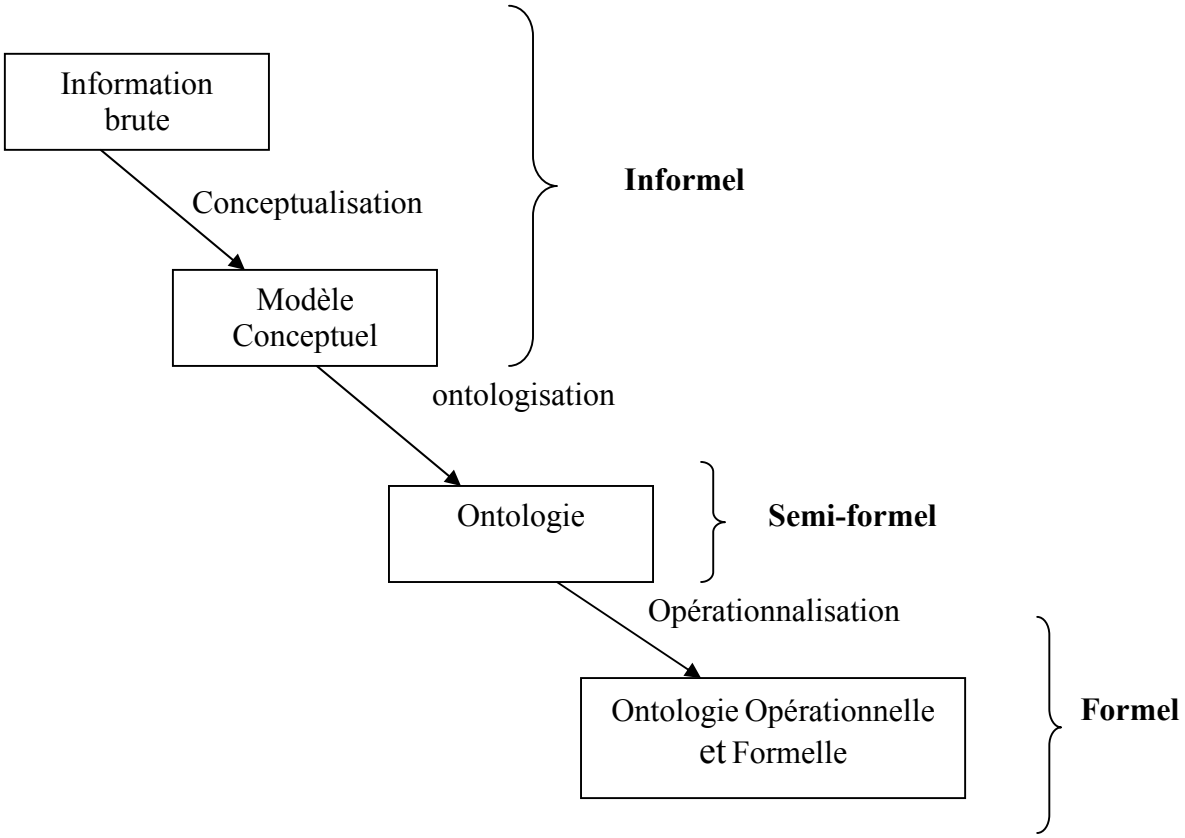

Figure 3.4: *Construction d'une ontologie opérationnelle.*

Le rôle des ontologies au sein du processus de représentation des connaissances étant désormais fixé, il convient maintenant de préciser quelles sont les primitives cognitives utilisées dans la construction des ontologies.

3.3 Rôle des ontologies dans les agents et les SMA

Les ontologies ont été développées pour fournir des vocabulaires spécifiques dépendants du domaine d'application pour la communication entre les agents. Une ontologie définit les concepts et les relations qui existent entre les mots d'un vocabulaire formel pour les agents qui l'utilisent. Il faut noter que les agents d'un système multi-agent partagent la même ontologie (le même vocabulaire) mais ceci ne veut pas pour autant dire qu'ils possèdent la même base de connaissances.

Un problème qui est étroitement lié à la sémantique des langages de communication est le traitement approprié des ontologies au niveau des langages de communication.
FIPA-ACL et *KQML* incluent un élément qui est employé pour identifier la source de vocabulaire utilisé dans le contenu de message. Ceci est conçu pour rendre ces langages de communication indépendants des vocabulaires particuliers d'application, et pour donner aux destinataires des messages une manière d'interpréter les termes non logiques au niveau du

contenu des messages. Dans les spécifications originales de *KQML*, cet élément a été conçu pour se rapporter à une ontologie indiquée dans **Ontolingua**[2].

Dans *FIPA-ACL*, la sémantique de l'étiquette d'ontologie est efficacement définie pour l'utilisateur. Évidemment, une ontologie qui a une large couverture, concernant son domaine, et extensible, est nécessaire dans beaucoup de langages de communication. Ainsi, une ontologie qui exhibe une couverture de son domaine permet aux agents multiples de partager leurs connaissances dans plusieurs contextes. Il est cependant très important de garder à l'esprit qu'une large couverture peut mener à une ontologie volumineuse et ainsi les agents peuvent passer beaucoup de temps à essayer de trouver la signification du contenu de leur langage de communication au lieu d'interagir les uns avec les autres. Une bonne ontologie devrait également être extensible afin de permettre aux concepteurs d'ajouter de nouveaux éléments. Finalement, une ontologie doit être dépendante de son domaine ("domain-dépendant") et ses rapports devraient montrer clairement leur pertinence avec ce domaine.

La manière avec laquelle un agent se servirait des spécifications des ontologies de KQML ou de FIPA-ACL pour interpréter des parties peu familières d'un message ACL n'a jamais été définie de manière précise. Le fait d'approvisionner une étiquette d'ontologie ne résout pas le problème de la façon dont les agents acquièrent et emploient la base de connaissance ontologique commune qui est un préalable à une bonne communication. Ceci représente un problème particulièrement aigu dans les systèmes ouverts qui incluent des agents basés dans différents organismes. Au niveau des problèmes liés à l'étude des significations et au raisonnement avec un nouvel ensemble de terminologie, il devrait exister des règles de traduction qui permettent de convertir des termes significatifs d'une ontologie vers une autre. Bien qu'un fonctionnement humain avec des représentants de chaque communauté terminologique puisse souvent hacher hors d'un ensemble de règles satisfaisant, il est presque impossible de construire ces règles entièrement de manière automatique. En conséquence, les agents peuvent seulement entièrement communiquer que s'ils partagent déjà une ontologie commune, ou si un ensemble de règles de préexistence de traduction est fourni. Le problème ontologique général fait toujours l'objet d'une recherche active.

3.4 Fonctions de l'ontologie et du langage de contenu dans JADE

Quand un agent A communique avec un agent B, une certaine quantité d'information I est transférée à partir de A à B au moyen d'un message ACL (AclMessage). A l'intérieur du message ACL, I est représentée comme une expression de contenu conformée à un langage de contenu approprié (par exemple ***SL***[3]) et codé dans un format approprié (Par exemple *String*).

A et B ont probablement leurs propres manières intérieures de représentation de I. Tenant compte que la manière dont un agent représente intérieurement un morceau d'information doit permettre une manipulation facile de cette information, Il est bien évident que la représentation utilisée dans une expression de contenu d'ACL n'est pas appropriée à la représentation à l'intérieure d'un agent. Par exemple l'information concernant une personne dont le nom est Mohamed et qui a 33 ans dans une expression de contenu d'ACL peut être représentée comme un *String*

(Personne: nommée Mohamed, âge : 33 ans).

Stocker cette information à l'intérieur d'un agent simplement comme un *String* n'est pas approprié pour manipuler l'information comme par exemple pour obtenir l'âge de Mohamed il faudrait analyser le *String* à chaque fois.

Considérons des agents logiciels écrits dans Java (Comme le sont les agents de Jade), l'information peut être convenablement représentée à l'intérieur d'un agent comme des objets Java. Par exemple, représentons l'information ci-dessus concernant Mohamed comme une instance (un objet Java) d'une classe spécifique à une application

```
class Person {
        String name;
         int age;
    public String getName() {
                        return name;
                         }
    public void setName(String n) {
                              name = n;
                        }
     public int getAge() {
                        return age;
                    }
     public void setAge(int a) {
                          age = a;
                  }
                  ….
          }
Initialisé avec    name = "Mohamed ";
                   age = 33;
```

permettra de manipuler facilement cette information.

Il est clair cependant que si d'une part l'exploitation de l'information à l'intérieur d'un agent est soulagée, d'autre part, chaque fois que l'agent A envoie une information I à l'agent B alors :

1- L'agent A a besoin de convertir sa représentation interne de I en représentation d'expression de contenu d'ACL correspondant et l'agent B doit exécuter la conversion opposée.
2- D'ailleurs B devrait également exécuter un certain nombre de contrôles sémantiques pour vérifier que I est une information significative, c à d qu'elle est conforme aux règles (par exemple l'âge de Mohamed est réellement une valeur de nombre *Entier*) de l'ontologie au moyen de laquelle les agents A et B attribuent une signification appropriée à l'information I.

Le support des langages de contenu et des ontologies fourni par Jade est conçu pour exécuter automatiquement toutes les conversions et vérifier les opérations comme représenté sur le schéma ci-dessous.

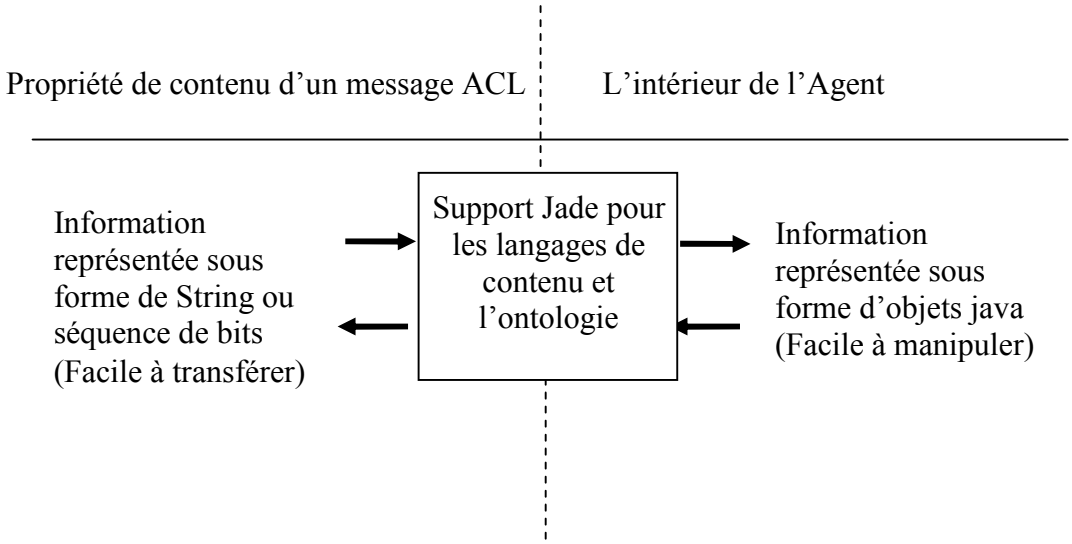

FIG 3.5 : *Le mécanisme de conversion de contenu de message suivant une* ontologie

3.4.1 Eléments principaux

Les opérations de conversion et de contrôle décrites dans le schéma précédent sont effectuées par l'objet gestionnaire de contenu (une instance de la classe *ContentManager* incluse dans le package *Jade.content*).

Chaque agent jade possède un gestionnaire de contenu accessible par la méthode *getContentManager*() de la classe Agent, la classe *ContentManager* fournit toutes les méthodes pour transformer des objets Java en *String* (ou séquence de bits) et pour les insérer dans la propriété de contenu d'un message *ACL* et vice versa.

Le gestionnaire de contenu fournit les interfaces appropriées pour accéder à la fonctionnalité de conversion, mais actuellement délègue les opérations de conversion et de contrôle à une ontologie (une instance de la classe *ontology* inclus dans le package *Jade.content.onto*) et un codec[4] de langage de contenu (une instance de l'interface *codec* inclus dans le package *Jade.content.lang*). Plus spécifiquement l'ontologie valide l'information à convertir du point de vue sémantique tandis que le codec effectue la traduction en *String* (ou séquence de bits) selon des règles syntaxiques du langage de contenu.

3.4.2 Le modèle de contenu de référence

Pour que Jade exécute les contrôles sémantiques appropriés sur une expression de contenu donnée, il est nécessaire de classifier tous les éléments possibles dans le domaine de discours (les éléments qui peuvent apparaître dans une information valide employée par un agent comme contenu d'un message *ACL*) selon leurs caractéristiques sémantiques génériques. Cette classification est dérivée du langage ACL défini dans FIPA qui exige du contenu de chaque message (*ACLMessage*) d'avoir une sémantique appropriée selon la performative du message *ACL*. Plus en détails, en premier lieu nous faisons la distinction entre **prédicats** et **termes.**

- **Les prédicats** sont des expressions qui indiquent quelque chose au sujet du statut du monde et peuvent être vraies ou fausses par exemple :

  (Works-for (Person : name Mohamed) (Company : name TILAB)) déclare que

  "La personne Mohamed travaille pour la compagnie TILAB ".

  Des prédicats peuvent être clairement employées par exemple comme

  contenu d'un message INFORM ou d'un message QUERY-IF, tandis qu'il ne

  semblerait pas raisonnable qu'il soient utilisés comme contenu d'un message

de demande (REQUEST).

- **Les Termes** sont des expressions identifiant des entités (abstraites ou concrètes) qui existent dans le monde et qui font un sujet de raisonnement et de communication des agents. Il sont classifiés en:

➢ **Les expressions de concepts** sont des expressions qui indiquent des entités ayant une structure complexe qui peut être définie en termes de champs.

par exemple (Person : name Mohamed : age 33).

Les concepts n'ont pas de sens s'ils sont utilisés directement comme contenu d'un message ACL. Généralement, il sont référencés dans des prédicats et d'autres concepts tels que :

(Book: title "The Lord of the rings": author (Person: name "Abdelkader")).

➢ **Les Actions d'agents :** sont des concepts spéciaux qui indiquent les actions qui peuvent être effectuées par quelques agents par exemple :

(Sell (Book: title" the lord of the rings")(Person: name Mohamed)).

Il est utile de traiter des actions d'agents séparément puisque, à la différence des concepts normaux, ils sont les contenus significatifs de certains types de message *ACL* tels que REQUEST. Les actes communicatifs (message *ACL*) sont eux-mêmes des actions d'agents.

➢ **Les primitifs** : sont des expressions qui indiquent les entités atomiques telles que *String* et *Integer*.

➢ **Les agrégats :** sont des expressions qui indiquent les entités qui sont des groupes d'autres entités, par exemple

(Séquence (Person: name Mohamed) (Person: name Boudina)).

➢ **Les expressions référentielles identifiantes (IRE[5])**: sont des expressions qui identifient l'entité (ou les entités) pour lesquelles un prédicat donné est vrai. par exemple :

(all ?x (Works-for ?x (Company :name TILAB))

Cette expressions identifie tous les éléments x pour lesquels le prédicat :

(Works-for x (Company :name TILAB)) est vraie,

Ces expressions sont typiquement employées dans les messages d'interrogation et exigent des variables.

Les variables : sont des expressions (Typiquement utilisées dans les interrogations "*Queries*") qui indiquent un élément générique non connu à priori.

Un langage de contenu entièrement expressif devrait pouvoir représenter et distinguer tous les types cités ci-dessus.

Une ontologie pour un domaine donné est un ensemble de schémas définissant la structure des prédicats, des actions d'agents et des concepts (fondamentalement leurs noms et leurs propriétés) pertinents à ce domaine.

4. Conclusion

Pour récapituler nous pouvons dire que l'ontologie assure une sémantique commune à la terminologie utilisée dans les messages échangés entre les agents,

Nous avons présenté toutes les notions de base contribuant au développement de l'ontologie utilisée par les agents pour communiquer et raisonner. Ces notions sont propres à la plateforme JADE, elles présentent une certaine analogie avec les notions utilisées dans d'autres plateformes.

Nous détaillerons plus ces notions dans la partie implémentation, où nous expliciterons toutes les étapes suivies pour développer notre ontologie.

Chapitre **IV**

Relation des technologies P2P, agents mobiles et ontologies

1. Introduction

Les applications réparties à l'échelle du réseau mondial, telle que les systèmes peer to peer sont complexes par leur nature décentralisée et ouverte. Leur construction doit prendre en compte les évolutions fréquentes de la structure du réseau ainsi que la volatilité des ressources et des services (localisation, disponibilité. . .). Les approches classiques pour le développement ne permettent pas de répondre de manière satisfaisante à ces contraintes. Nous proposons dans ce chapitre la relation et les travaux connexes entre les technologies peer to peer, agent mobile et ontologies.

2. Apport réseaux P2P et Agents Mobiles

2.1 Motivation d'utilisation des agents mobiles dans les systèmes peer to peer

Jusqu'à présent, la motivation d'utilisation des agents mobiles en peer-to-peer vient des aspects suivants [Lübke, 2004] [Dimakopoulos, 2003] [Braun, 2005] [Genco, 2007] [Dunne, 2001]:

- Les agents mobiles réduisent le besoin en bande passante. Très souvent, les pairs établissent un canal de communication en utilisant un protocole distribué. Puis effectuer les connexions multiples sur ce canal. Chacune de ces interactions génère du trafic sur le réseau. Les agents mobiles permettent de conditionner ces interactions ensemble, et les envoyer comme une pièce discrète dans le réseau. Ces derniers, permettent toutes les interactions localement. Les agents mobiles permettent aussi de joindre toutes les données nécessaires entre elles. Par conséquent, lorsqu'un agent mobile arrive sur un ordinateur, il n'a pas besoin de communiquer avec tous les autres ordinateurs, car il a toutes ses données avec lui. Dans un protocole de recherche habituel, toutes les données brutes se déplacent sur le réseau pour être traitée, même si seulement une partie de ces données peut être nécessaire. Dans ce scénario, le trafic réseau est réduit par des agents mobiles tout en déplaçant le traitement des données brutes plutôt que le déplacement des données brutes pour le traitement. Enfin, les agents mobiles peuvent être de très petite taille, mais ils peuvent se développer de manière dynamique car ils ont besoin d'accueillir davantage de données.

- Deuxièmement, les agents mobiles sont asynchrones. Par conséquent, quand un agent mobile est posté il n'est pas nécessaire d'attendre son retour. En effet, tandis que les agents mobiles sont sortis, l'homologue d'origine n'a même pas besoin de rester connecté au réseau. Les agents mobiles peuvent attendre jusqu'à ce que le peer d'origine soit de retour sur le réseau avant d'essayer d'y revenir.

- Troisièmement, les agents mobiles sont autonomes. Cette principale caractéristique du réseau peer-to-peer, parce que l'agent mobile apprend sur le réseau tout en se déplaçant. L'agent mobile visite les pairs qui étaient inconnus quand il a été initialement expédié. A chaque pair il peut prendre des décisions en fonction de son historique de pairs visités et l'actuel peer.

- Quatrièmement, l'information est diffusée à tous les pairs que l'agent mobile les a visité. Tous les pairs auront l'avantage d'accepter la visite d'un agent mobile, parce que ce dernier aura de nouvelles informations plus récentes sur les ressources. En outre, c'est avantageux qu'agent mobiles rend visite à un pair, car il apprendra soit de nouvelles ressources ou il fait une mis à jour. Dans le cas ou des agents mobiles n'auront aucune information nouvelle, ils peuvent être détruits. Accepter et accueillir des agents mobiles à besoin l'utilisation des ressources physiques, comme la mémoire et les cycles informatiques. Dans le cas de la disponibilité des ressources physiques, il est facile pour le peer de refuser de nouvelles requêtes pour accepter des agents mobiles.

- Cinquièmement, les agents mobiles peuvent facilement être reproduits et envoyés dans des directions différentes. Cela leur permet de fonctionner en parallèle. Bien que cela provoque des agents mobiles en plus d'être actif sur le réseau, ils assurent la découverte des ressources dans réseau rapidement, donc ces agents mobiles passent moins de temps sur le réseau.

- Sixièmement, une solution d'agent mobile sur la base est très acceptée. Même si quelques-uns des agents mobiles sont détruits, tous ceux qui ont survécu auront une influence positive. Certes, les agents détruits ont bénéficié de tous les pairs à l'endroit où ils ont été détruits. Enfin, une solution basée sur l'agent mobile peut être combiné avec des caractéristiques positives de peer-to-peer d'autres systèmes .

2.2 Application des agents mobiles sur des réseaux peer to peer pour la localisation des ressources travaux de Dasgupta et Dunne :

Dans les réseaux de partage de fichiers, tels que Napster, Kazaa ou Gnutella, la localisation des ressources partagées est l'une des caractéristiques-clés. Les technologies actuelles utilisent diverses méthodes pour localiser les ressources [Milojicic, 2002]. Napster utilise le modèle d'annuaire centralisé, un modèle p2p hybride, où un seul serveur centralisé inclus des informations sur les ressources. Quand quelqu'un voulait trouver un fichier mp3 par exemple, il demande au serveur, il obtient l'adresse du pair ayant le fichier désiré, et contacté par les pairs que pour le téléchargement.

La méthode utilisée dans Gnutella, est l'inondation de requêtes. Dans ce modèle, la requête est inondé dans le réseau p2p, et le peer qui a la ressource recherchée, répond à la requête.

Les ressources recherchées à partir du réseau ne sont pas seuls les fichiers. Aussi, par exemple la puissance de calcul peut être partagée en tant que ressource.

Dans le modèle traditionnel ou les ressources sont partagées les hôtes établissent un canal de communication, à travers lequel toutes les informations sont envoyées et reçues. Avec ce genre de système la bande passante est nécessaire en permanence .Avec l'utilisation d'agents mobiles, ce besoin de communication peuvent être emballés pour quelques paquets simples, et envoyé sur le réseau [Dunne, 2001]. Les agents mobiles permettent donc toutes les interactions localement sur l'hôte local, l'agent rassemble les informations à partir du réseau et retourne à l'hôte mère.

Dans ce sens Dasgupta [Dasgupta, 2004] présente une méthode basée sur l'agent mobile pour découvrir des ressources dans un réseau. Dasgupta utilise des agents à la fois, fixes et mobiles. Les agents de la méthode Dasguptas sont les suivants:

- Agents de Tâche (stationnaire).
- Agents de reconnaissance (mobile).
- Agents de recherche (mobile).

- Agents Téléchargement (mobile ou fixe).
- Agents d'information (stationnaire).
- Agents d'interface (stationnaire).
- Les agents de sécurité.

Les agents les plus importants sont l'agent de reconnaissance, de recherche et de téléchargement. La méthode permet la détermination et le téléchargement d'une ressource du réseau par la méthode suivante: l'Agent de tache envoie des agents de reconnaissance dans tous les pairs voisins. La tâche de ces agents est de découvrir la disponibilité des ressources, y compris par exemple la connectivité réseau et les capacités de calcul de ce nœud. Après avoir pris tous les renseignements nécessaires, l'agent de reconnaissance rend visite à tous les pairs voisins du pair. Lorsque cet agent a visité un nombre limité (ce qui est décidé lors premier lancement) de pairs ; il retourne à la paire mère et donne à l'agent de tâche des adresses et autres informations obtenues.

Deuxièmement, lorsque nous voulons trouver une ressource (fichier, etc) un agent de recherche est créé par les pairs concernés avec un itinéraire qui se compose de nœuds que l'agent reconnaissance les avait visité. Sur chacune des hôtes visités, l'agent de recherche enregistre toutes les informations pertinentes. Après avoir visité tous les hôtes sur l'itinéraire, l'agent retourne chez lui et donne à l'agent de tâche les informations.

Enfin, l'agent décide quelle pairs possède la ressource, et crée un agent de téléchargement. Sa principale fonction est de télécharger des ressources.

Une autre méthode de découverte des ressources est décrite par Dunne [Dunne, 2001] en utilisant des agents mobiles. Dans cette méthode, les agents mobiles sont également lancés à partir de pair local sur le réseau pour trouver les ressources. En plus les agents sont également capables de se cloner et ainsi être en mesure de rechercher plus efficacement dans des réseaux plus vastes. Plus de fiabilité est atteinte en reproduisant plus d'agents, aussi un ou plusieurs des agents peuvent être détruits tandis que d'autres essayent encore de localiser les ressources. Ces méthodes pour la localisation des ressources dans le réseau peuvent également être appliquée si

la ressource doit être recherché n'est pas un fichier. Par exemple, la recherche de la puissance de calcul pour le calcul distribué qui pourrait être fait par des agents mobiles.

3. Apport systèmes multi agent et Ontologie

3.1 Le Système Multi-Agent et l'ontologie

Dans la revue de la littérature, nous remarquons l'existence du concept agent ou Système Multi-Agent et les ontologies comme deux domaines distincts mais en y regardant de plus près, il s'agit bien d'éléments complémentaires. Ainsi, il s'agit que l'usage des 1ontologies présente un atout pour le fonctionnement des Systèmes Multi-Agents. En effet, l'usage de ces derniers soulève des problèmes liés à la communication entre les agents. Et comme nous avons déjà présenté les ontologies permettent de s'attaquer à des problèmes communs de langage et de représentation du monde. Nous allons donc examiner les différentes relations présentées entre les Systèmes Multi-Agents et les ontologies.

3.2 Les ontologies pour les Systèmes Multi-Agent

En premier lieu, on va concevoir la relation entre l'ontologie et l'agent puis sa relation avec le Système Multi-Agent en entière. En effet, on trouve parfois dans des travaux que chaque agent possède sa propre ontologie dont le but de lui rendre capable de la modifier grâce aux interactions avec d'autres agents [Siebes, R et al. 2002]. Ainsi, dans autres cas un agent peut être un porteur de deux ontologies comme pour le projet de l'Université Utrecht ANEMONE [Van, D et al. 2006] (AN Effective Minimal Ontology Negotiation Environment).

Dans ce projet chaque agent possède deux versions de son ontologie: une version dont l'ontologie native et une version acquise lors de la négociation et évolue au cours de temps.

Dans ce cas, les ontologies utilisées sont comme des représentations du monde pour l'ensemble des agents d'un système ouvert. Ainsi, pour ce dernier, il est tout à fait possible d'avoir plusieurs ontologies d'un même domaine donné. Mais dans ce cas, il existe les problèmes de communication entre ces agents, ce qui justifie le besoin de système d'alignements d'ontologies [Kevin, O. 2009].

Alors que la relation de l'ontologie avec le SMA réside dans son utilisation pour faire fonctionner les Systèmes Multi-Agents [Gandon, F. 2002]. Dans ce cas, les agents se réfèrent à une ontologie connue pour mettre en œuvre un langage commun et être capable d'interagir. Les ontologies sont alors un moyen de médiation entre agents logiciels, ce qui est une des raisons de leur définition. En effet d'après [Antoine, I. 2005], les ontologies constituent des référentiels de sens pouvant améliorer les communications entre agents, tout en garantissant un degré de pertinence suffisant pour les traitements qui seront effectués de manière automatique à partir des éléments de spécification qu'elles fournissent.

Selon [Mark, T et al.2003], les ontologies interprètent un flux de données en provenance de l'extérieur du Système Multi-Agent qui regroupe un ensemble d'agents utilisant un langage est commun. Dans ce cas, un ensemble fixe d'ontologies donné initialement au système sera utilisé par tous les agents pour effectuer un traitement sur un ensemble de données.

On trouve aussi le système COMMA (Corporate Memory Management through Agents), [Federico, B et al. 2002] qui couple le concept agent et ontologie. Ce système est composé de quatre sociétés d'agents. Trois parmi eux sont dédiées aux ressources :

L'ontologie et les modèles; les annotations; les pages jaunes pour gérer les interconnexions entre agents, et une dédiée aux utilisateurs (figure ci-dessous) [Gandon, F et al. 2004].

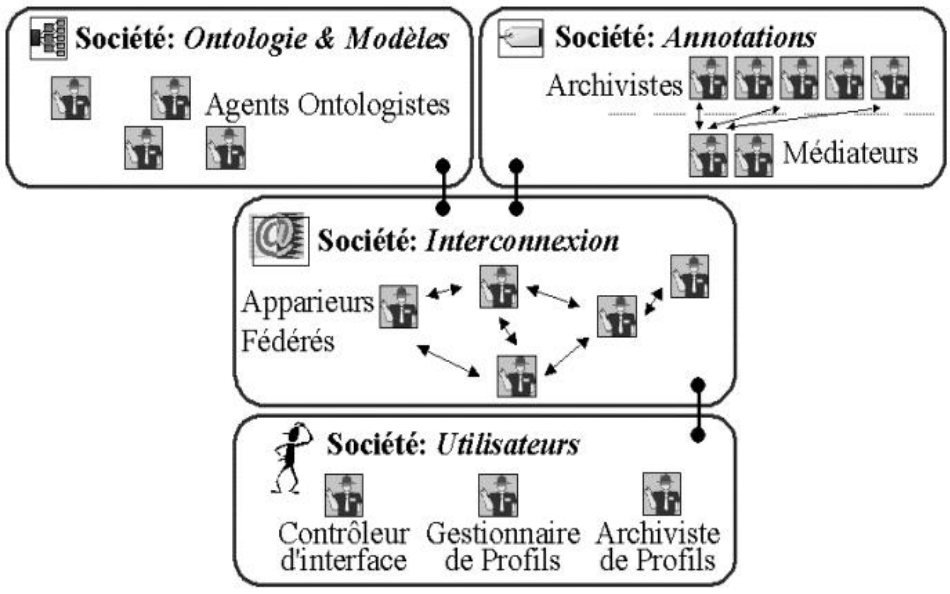

Figure 4.1 : *Les sociétés pour la gestion de la mémoire.*

En analysant les sociétés dédiées aux ressources, les auteurs identifient un ensemble récurrent d'organisations possibles: hiérarchique, égalitaire ou réplication. Selon la taille et la complexité des ressources manipulées et selon le type de tâches devant être exécutées, le choix d'une telle ou telle organisation de la société d'agents sera fixé. Concernant, la société dédiée à l'ontologie et aux modèles, elle prend en charge de rassembler des agents ontologistes ayant chacun une copie complète de l'ontologie. Aussi, elle est organisée comme une société de réplication. Par contre, la société dédiée aux annotations est une organisation hiérarchique où les agents médiateurs d'annotations s'allouent les nouvelles annotations et les tâches de résolution distribuée des requêtes, aux agents archivistes d'annotations.

3.3 Les systèmes Multi-Agents pour les ontologies

[Kevin, O. 2009] a utilisé le Système Multi-Agent pour la construction des ressources ontologiques à partir de textes où le système lui-même sera l'ontologie, et les composants du système (les agents) seront alors des entités de l'ontologie (concepts, relations, instances…..).

On trouve aussi des SMA où les ontologies assurent la bonne communication entre les agents ainsi grâce à l'alignement entre eux. Alors que [Alexandre, V. 2003] a définis un protocole de communication entre agents dont le but de faire l'alignement des ontologies c'est-à-dire de mettre en correspondance leurs ontologies pour pouvoir partager du sens. Aussi ce protocole permet d'aider les agents à se retrouver dans une situation où un message venant d'un autre agent peut être interprété. Dans le même contexte, [Kendall, L et al. 2006] a mis en œuvre les Systèmes Multi-Agents pour faire concilier les différences ontologiques, sans explicitement ontologies formelles. Et pour se faire ces systèmes exercent des calcules sur des ontologies différentes en jouant le rôle de médiateur entre eux.

Dans le même contexte, [Francisco, G et al. 2008] présentent le SEMMAS (Services

Web sémantique et Multi-Agents System) dont est l'architecture composée de trois éléments principaux: un ensemble d'agents intelligents qui constituent le SMA, et quatre dépôts d'ontologies, et trois interfaces pour interagir avec les acteurs extérieurs qui ont été identifiés, à savoir, les fournisseurs de services, les demandeurs de services et les développeurs de logiciels.

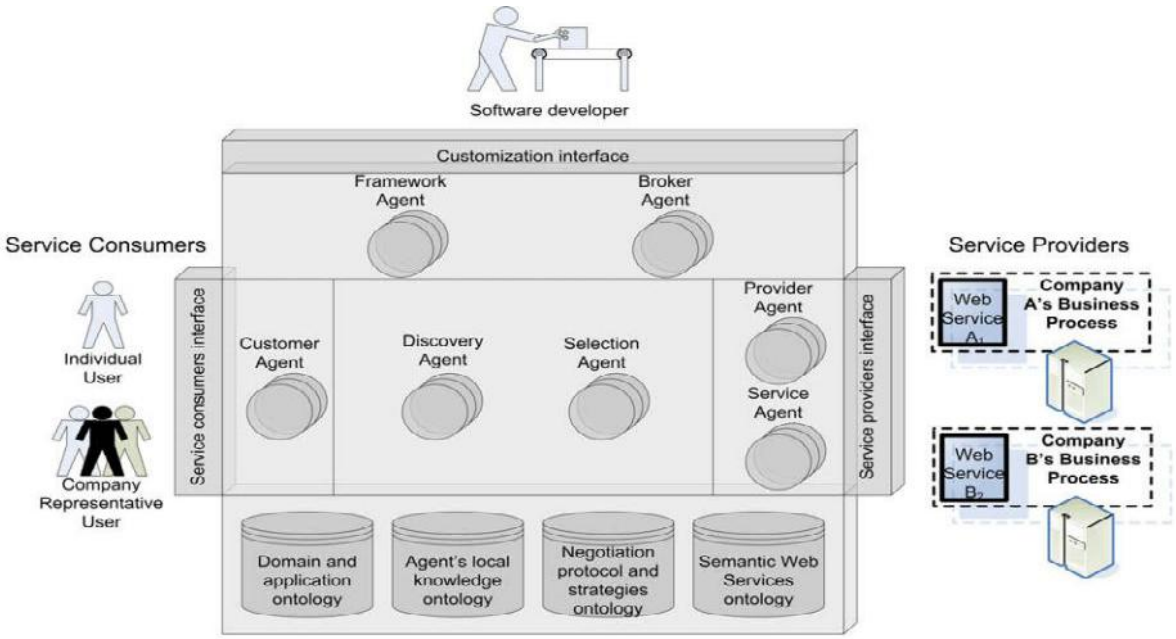

Figure 4.2 : *Services Web sémantique et Multi-Agents System (SEMMAS).*

En se basent sur la plate-forme proposée d'après ces auteurs, il existe sept types d'agents. Ces derniers sont regroupés en trois catégories principales: une catégorie renferme (« agent prestataire » et « agent de service ») qui agit sur les noms des propriétaires de service, mais aussi gère l'accès aux services et veille à ce que les contrats soient remplies, une autre regroupe (« agent de la clientèle », « agent de découverte », et « agent de sélection ») pour agir sur les noms de consommateurs de services ainsi ils localisent les services et présentent les résultats, et les agents qui gèrent les tâches (« agent-cadre » et « courtier »). L'objectif du système proposé est de gérer à la fois des erreurs inattendues et d'améliorer sa performance globale. Concernant, les ontologies il existe quatre types figurant comme des dépôts de données :

- Application et l'ontologie de domaine : contient les entités des connaissances (concepts, attributs, relations et axiomes) que le modèle application doit employer.
- Ontologie Agent connaissances locales: pour chaque agent, elle contient les connaissances sur l'environnement dont elle dispose. Ainsi, elle comprend les connaissances sur les tâches assignées, ainsi que les mécanismes et les ressources disponibles pour réaliser ces tâches.
- Ontologie de négociation: elle concerne la négociation des protocoles et des stratégies de négociation qui forment les mécanismes de négociation que les agents doivent

utiliser pour coordonner leurs interactions.

- Services Web sémantiques ontologies: dans ce dépôt, les ontologies contiennent la description sémantique des Services Web stockés.

4. Conclusion

En conclusion, étudiant Les caractéristiques des technologies d'agents mobiles et d'ontologies et voir la relation qui relie entre les deux on constate que le web sémantique aide beaucoup l'optimisation, le comportement coopératif des agents et permet d'avoir une vision partagée de la connaissance, en dépit de l'hétérogénéité de leurs sources.

D'où notre idée d'utiliser l'avantage du couplage de ces deux technologies dans des systèmes ouverts et dynamiques tel que les systèmes peer to peer pour assurer la localisation des ressources.

Le chapitre suivant étudiera en détails l'architecture de notre système à base de ces technologies.

Deuxième partie
Approche proposée

Chapitre V

Architecture Proposée

ANALYSE ET CONCEPTION D'UN SYSTEME DE LOCALISATION DES RESSOURCES DANS UN SYSTEME P2P PUR

1. Introduction

Les systèmes multi-agents sont l'un des paradigmes technologiques les plus prometteurs dans le développement de systèmes logiciels distribués, ouverts et intelligents.

L'usage des ontologies présente un atout pour le fonctionnement des Systèmes Multi-Agents. En effet, l'usage de ces derniers soulève des problèmes liés à la communication entre les agents.

Dans cette optique, Nous avons proposé un Framework intelligent pour la découverte de ressources dans un système P2P pur basé sur les agents intelligents et le web sémantiques, baptisé i-PRL (Intelligent P2P Resource Location). Le couplage de ces deux technologies permet d'accéder à des ressources par l'utilisation de descriptions sémantiques utilisables et compréhensibles par les agents pour résoudre tous les conflits sémantiques apparaissant.

Le concept partagé du domaine des connaissances est basé sur une ontologie qui aide les agents mobiles de migrer vers des ressources relatives et d'optimiser la coopération entre eux.

Dans ce chapitre, nous présentons l'architecture globale de notre système de localisation des ressources conteneurisées. Nous détaillons les différentes couches qui la composent.

Dont le bût de l'analyse et la conception du système i-PRL, nous choisissons la méthodologie MaSE (Multi-agent System Engineering). Pour l'implémentions de notre système i-PRL, nous adoptons la plate-forme JADE (Java Agent Développent Framework), et l'outil protégé 2000 pour construire l'ontologie.

2. Architecture globale du système i-PRL (Intelligent P2P Resource Location)

Comme nous l'avons déjà évoqué, l'intégration des informations est importante dans tous systèmes de localisation. Et, du fait que la popularisation de l'Internet et de la technologie du Web, la réalisation des applications coopératives dans un environnement distribué est de plus en plus nécessaire. Par la suite, les utilisateurs peuvent directement utiliser une plate-forme Web pour l'accès aux données. De ce fait, notre réflexion porte sur une architecture intelligente adaptée pour le système de localisation i-PRL. Ce système est composé d'un ensemble d'agents pour traiter et assurer la fiabilité des informations associées et agir rapidement, correctement et intelligemment. Ainsi, ces agents doivent

intégrer les informations de leurs environnement afin de créer des concepts globaux, c'est-à-dire, posséder une connaissance étendue et riche.

L'interaction, la communication et la coopération entre les peers sont réalisés grâce aux agents. De ce fait, les informations exigées pour la résolution d'un tel problème ou pour qu'un agent prend une telle décision seront accessibles et partagées grâce à la transparence au niveau des échanges. Ces informations sont représentées sous des formats hétérogènes. L'existence des agents spécifiques destinés à la transformation de ces formats est indispensable afin de les rendre homogènes dont le bût de construire des vocabulaires bien définis. Ainsi, grâce à l'ontologie, les informations seront plus flexibles et compréhensibles pour l'exécution de différentes tâches des agents.

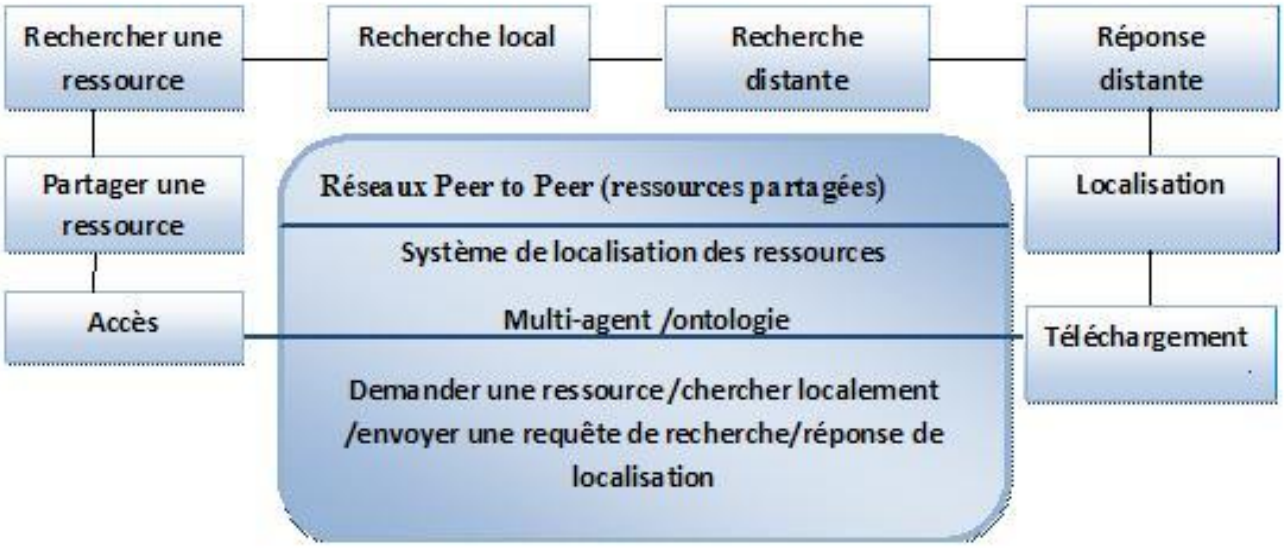

FIG 5.1 *Architecture globale du système i-PRL*

Afin de cerner au mieux les particularités du système i-PRL proposé, nous présentons, dans la suite de ce chapitre, l'architecture détaillée en décrivant les déférentes parties qui la constituent.

3. Architecture détaillée du système i-PRL

L'architecture détaillée du système i-PRL est composée de cinq couches présentées comme le suit (figure 5.2) :

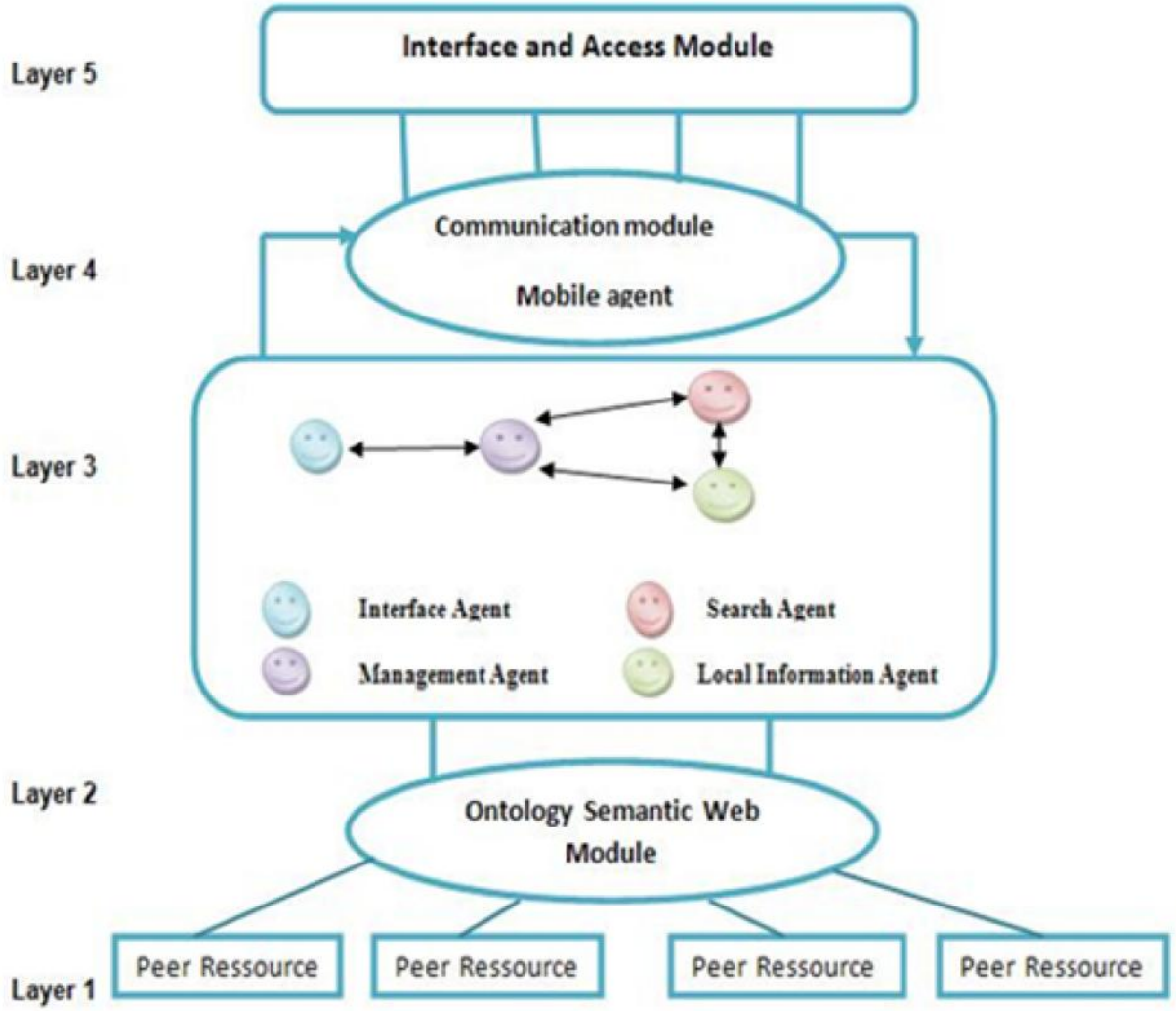

FIG 5.2 *Architecture détaillée du système i-PRL*

3.1 Couche 5 : module de connexion et d'accès

Elle permet aux utilisateurs de s'interfacer avec le système afin de les alimenter à travers les différents Peer (les ordinateurs de bureau, les téléphones cellulaires, les ordinateurs portables et les PDA …). Grâce à cette couche, le système i-PRL peut ainsi fournir différents moyens pour que les utilisateurs entrent dans le portail de manière ubiquitaire. Aussi, elle se caractérise par une large accessibilité, car les utilisateurs peuvent accéder à l'i-PRL de n'importe où et n'importe quand aussi bien par une multitude de formats d'entrées.

3.2 Couche 4 : module de communication

Elle prend en charge la standardisation des différents formats d'informations en entrées et en sorties. Pour accomplir sa fonctionnalité, cette couche comprend l'interaction entre l'interface et « Location Module ». Cette couche utilise interface agent. Le module réception reçoit l'information à partir de l'interface qui sera par la suite évaluée par le module évaluation. Si l'évaluation est positive, alors le module transformation change le format de l'information et l'envoie au système i-PRL. Ce dernier traite l'information obtenue puis la transfère au module réponse pour qu'elle soit affichée sur l'interface des utilisateurs.

3.3 Couche 3 : module d'applications

Elle représente la partie centrale du système i-PRL. Elle est constituée de plusieurs agents qui travaillent en collaboration et en coordination dans un système où les compétences et les connaissances sont distribuées pour assurer la localisation des ressources.

Par la suite nous présentons les agents de cette couche :

- **Interface agent (IA) :**

Est une interface pour l'interaction entre le système et l'utilisateur. Plusieurs tâches sont assignées: IA récupère une demande faite par l'utilisateur, il traite cette requête pour extraire les éléments nécessaires pour les diriger vers l'agent de gestion (MA), et présente les résultats à l'utilisateur (retour de l'agent de gestion (MA).

- **Management Agent (MA) :**

Chaque nœud possède un agent de gestion statique (MA) qui gère les informations des ressources. MA interagit également avec l'interface agent, l'agent mobile local d'information et de recherche Agent.MA reçoit la demande de l'agent d'interface (IA), crée un agent de recherche mobile (SA), renvoie les résultats trouvés par SA à IA et mises à jour par MA une nouvelle information.

- **Search Agent (SA) :**

Après avoir accepté une requête utilisateur, l'agent de gestion (MA) crée un agent de recherche mobile (SA). D'abord, ce dernier il interagit avec l'agent local d'information pour voir quelle peer disposant la ressource. Distribué localement SA se déplace à travers le réseau pour trouver les ressources demandées. Apres son expédition Il renvoie les résultats trouvés à l'agent de gestion (MA).

- **Local Information Agent (LIA):**

Chaque noeud possède un agent local d'information (LIA) qui gère toutes les informations sur un pair Un LIA a une table qui contient les adresses IP des voisins et d'autres informations comme le numéro de port, nom de la ressource, type de ressource,. (figure 5.3)

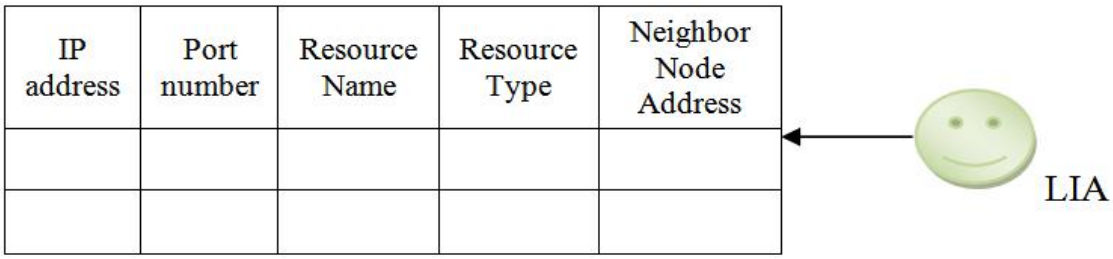

FIG 5.3 *Local Information Agent (LIA)*

Les étapes du “the intelligent Resource discovery algorithm” pour le location module sont:

- L’agent interface (IA) reçoit la requête de l’interface utilisateur puis il la transmet vers l’agent de gestion (MA).
- L’agent de gestion (MA) faits des interactions avec l’agent d’information locale (LIA) pour lui demander une ressource ; si la ressource existe localement ce dernier va répondre positivement sinon il va créer un agent de recherche (SA) avec une métrique du nombre de peer a visiter avant d’abandonner la recherche si la ressource n’est pas trouvé.
- L’agent de recherche fait des interactions avec l’agent d’information locale pour savoir quelle peer dispose la ressource recherchée .L’agent de recherche commence son expédition dans le réseau visitant les peers dans son chemin ,en collaborant avec chaque agent d’information locale des peers visitées si le résultat est positive ce dernier est dirigé vers l’agent de gestion (MA) .
- L’agent de gestion (MA) retourne le résultat pour l’agent interface quant a ce dernier va mettre a jour l’agent d’information locale par les nouvelles informations.

3.4 Couche 2 : module d’ontologie

Elle représente l’ensemble de vocabulaires qui décrivent l’univers du discours du domaine Intégré dans notre système. Cet ensemble de vocabulaire sera ensuite partagé par les agents du système i-PRL pour assurer une bonne communication. Aussi, cette couche gère les concepts et les termes utilisés dont le bût d’améliorer la flexibilité. Elle comprend les relations sémantiques entre eux afin de construire les bases de connaissances des agents et les rendre de plus en plus enrichies et compréhensibles garantissant ainsi une interprétation identique des données.

3.5 Couche 1 : les Peers

Représente le niveau le plus bas du système, il contient tous les pairs participant au système. (Ils peuvent rendre les données accessibles aux utilisateurs pour une recherche et de téléchargement).Elles permettent de mettre des données à la disposition des utilisateurs pour une consultation, une recherche et de téléchargement.

4. Modélisation et implémentation du système i-PRL

4.1. Présentation de la méthodologie de conception et le langage adapté

Nous opterons pour la méthodologie MaSE (Multi-agent System Engineering), que nous trouvons assez simple dans sa démarche et rigoureuse car les liens entre ces étapes et les résultats attendus pour chaque étape sont assez bien spécifiés et claire dont les détails de chacune des étapes sont précisés. Nous présentons dans cette partie la méthodologie MaSE développée par Scott DeLoach et ses collaborateurs au sein du Laboratoire d'Intelligence Artificielle de l'Air Force Institute of Technology (AFIT) [Deloacha, S et al, 2001]. Nous ajoutons aux critères cités précédemment qui justifient notre choix de la méthodologie MaSE que nous avons retenue pour la conception de notre système i-PRL l'abondante littérature et la documentation disponibles qui ont le mérite de décrire dans le détail les différentes étapes de la méthodologie. Ainsi, elle part d'une définition à l'agent simple, très modeste et très pragmatique et par la suite au Système Multi-Agents: ici, un agent est considéré comme un ensemble de processus informatiques qui communiquent entre eux pour atteindre un objectif global donné.

En effet, MaSE est orienté objet où un agent considéré comme une abstraction de plus haut niveau qu'un objet actif. De plus cette méthodologie s'appuie sur un environnement qui propose la génération automatique du code architectural des agents et se base sur le formalisme UML (Unified Modeling Language). En effet, l'outil AgentTool fournit des diagrammes pour représenter les modèles correspondant aux étapes de MaSE (figure ci-dessus) [Aloys, M. 2003].

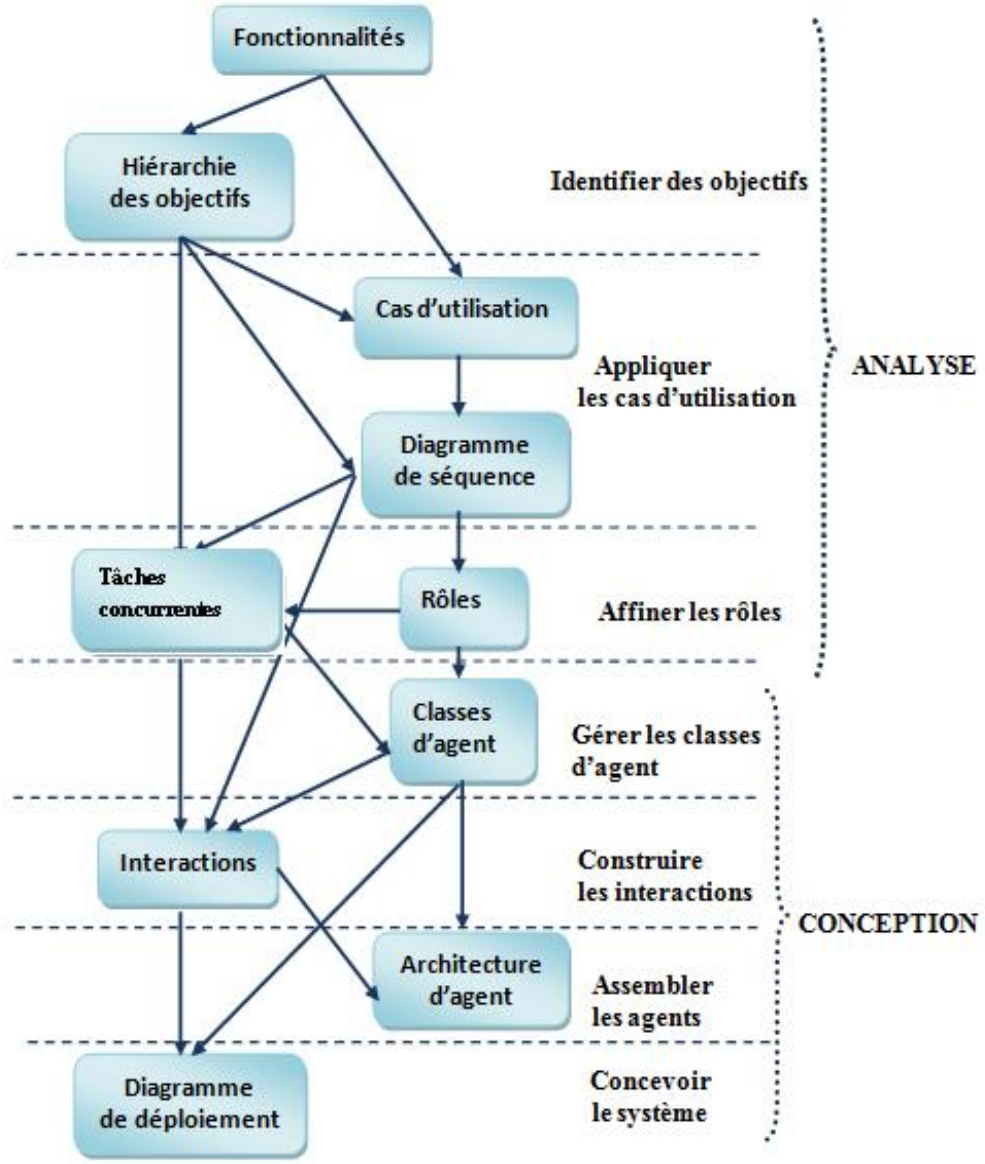

FIG 5.4 *Les étapes et les phases de la méthodologie MaSE*

La méthodologie MaSE met l'accent sur la façon de coordonner le comportement de chaque agent afin d'obtenir un comportement du système tout entier. Pour cela, MaSE utilise un certain nombre de modèles graphiques pour décrire les types d'agents, les interfaces inter-agents, l'architecture indépendante de la structure interne des agents. En se basant sur la figure ci-dessus, nous constatons qu'il s'agit de deux phases (analyse et conception) pour MaSE renfermant un ensemble d'étapes détaillées comme suivant :

La phase d'analyse comporte trois étapes :

- Identification des objectifs du système : le résultat de cette étape c'est un diagramme hiérarchisé des objectifs où les relations sont maintenues.
- Identification des cas d'utilisation, en se basant sur les exigences et le modèle des bûts, après l'extraction des scénarios qui représentent le comportement du système dans les cas spécifiques on obtient les diagrammes de séquence.
- Affiner les rôles, en se basant sur les diagrammes de séquences et de bûts dans les étapes précédentes, nous identifions les rôles du système pour déterminer les diagrammes de rôle.

La phase de conception comporte quatre étapes :

- Création les classes d'agents : le résultat de cette étape c'est le diagramme de classes D'agent.
- Construction des interactions entre les agents pour avoir un ensemble de diagrammes D'interactions.
- Assemblage des classes d'agents, dont le résultat est de concevoir l'architecture des agents.
- Concevoir du système : c'est la dernière étape pour la phase de conception qui dérive le diagramme de déploiement.

D'après ce qui précède nous remarquons que les différentes étapes sont enchainées où l'étape en amont fait servir les étapes en aval. Ainsi, les améliorations de cette méthodologie MaSE par rapport à la méthode orientée objet c'est qu'elle oriente le bût du système, le rôle dans système et les protocoles entre les agents.

4.2. Analyse et conception du système i-PRL selon MaSE

4.2.1. Phase d'analyse

Nous commençons par la phase d'analyse qui a pour finalité de produire un ensemble de rôles dont les tâches décrivent ce que doit faire le système i-PRL pour atteindre ses objectifs. Un rôle décrit une entité qui remplit une fonction dans un système. Ainsi, il aide à l'accomplissement d'un objectif ou un sous-objectif qui résume l'ensemble de fonctionnalités du système. En effet, la phase d'analyse permet de définir les objectifs et les rôles nécessaires pour leur accomplissement, plutôt que d'essayer de déduire les rôles à partir des objectifs. Multi-

agent System Engineering s'appuie sur l'utilisation des cas d'utilisation afin de définir et de valider les rôles. Les cas d'utilisation sont exprimés à partir des diagrammes de séquence permettant ainsi de structurer les interactions entre les différents rôles.

a. Identification des objectifs : structuration globale du système i-PRL

C'est la première étape de la modélisation. Nous concevons le système i-PRL pour structurer ses objectifs d'une façon hiérarchisée comme montre (figure 5.5)

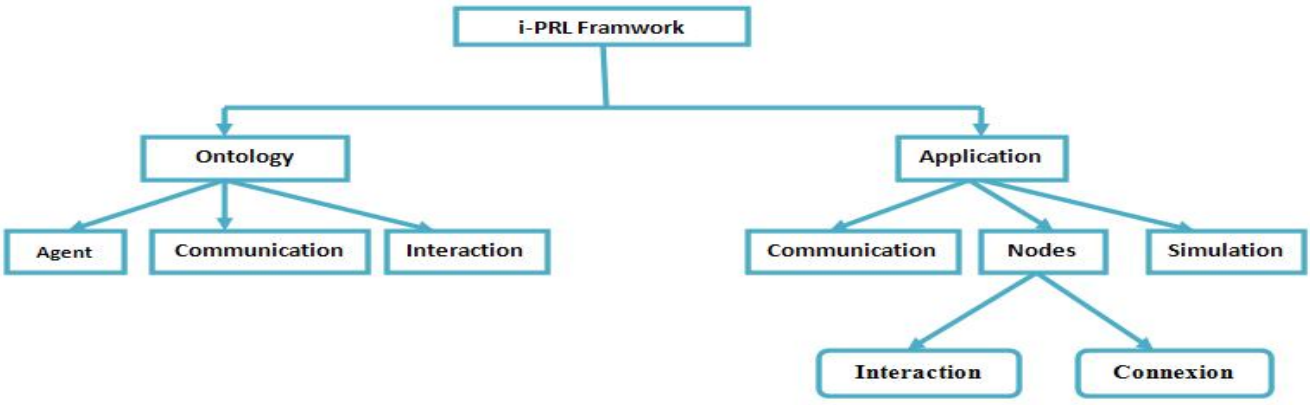

FIG 5.5 *structuration globale du système i-PRL*

b. Identification des cas d'utilisation

Les cas d'utilisations permettent de structurer les différents rôles et les bûts correspondants de notre système. En effet, ils sont explicités dans des diagrammes de séquence et définissent des Scénarios de base que le système peut exécuter. Cette étape a aussi pour bût de permettre au concepteur d'identifier un ensemble initial de rôles et d'interactions. Les différents diagrammes matérialisent l'ensemble des rôles et des interactions. Dans la spécification de i-PRL, nous avons identifié six cas d'utilisation. Nous présentons dans la suite la description de tous les cas d'utilisation de i-PRL. Et à partir de l'identification de ces cas d'utilisations, il convient de réaliser les diagrammes de séquence correspondants. Ces derniers permettent de représenter des collaborations entre les rôles d'agents selon un point de vue temporel dont le bût de mettre l'accent sur la chronologie des conversations afin de satisfaire un bût. Nous présentons par suite les quelques diagrammes de séquence.

- ***Cas d'utilisation « Localisation» :***

Dans le contexte de la localisation, la gestion d'accès représente la première étape de recherche des ressources. Pour accomplir cette étape, plusieurs acteurs et agents intégrant notre système collaborent et échangent des messages entre eux.

Au premier lieu, l'acteur expéditeur (peer) désire avoir une ressource. Suite à ce désir s'interfacer au système i-PRL l'un des agents est activé (interface agent).

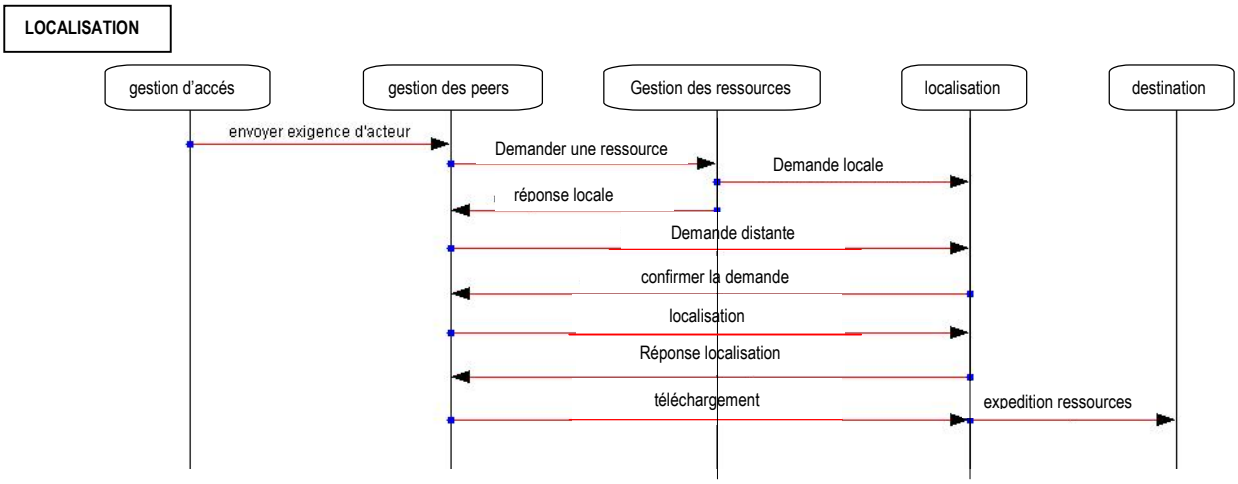

FIG 5.6 *Diagramme de séquence : localisation.*

- ***Cas d'utilisation « demande ressource» :***

Avant de commencer l'opération de localisation, l'agent interface fournit une demande de localisation.

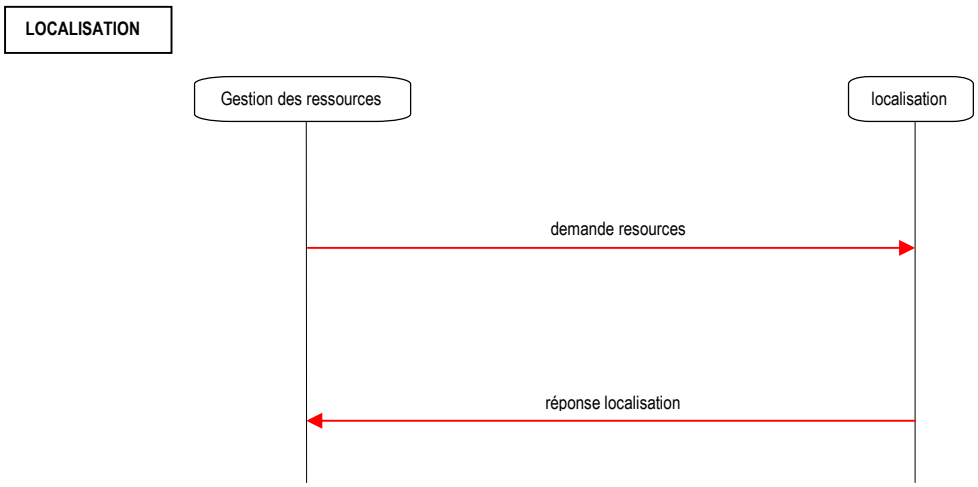

FIG 5.7 : *Diagramme de séquence : demande ressources.*

c. Affinage des rôles

C'est la troisième étape de la phase d'analyse permettant de transformer les objectifs hiérarchisés et les diagrammes de séquence issus des différents cas d'utilisation en un diagramme de rôles et de tâches concurrentes associées. Ces tâches sont des processus indépendants que le rôle doit les exécuter pour remplir ses objectifs. Etant indépendants, ilss'exécutent de façon parallèle, c'est pourquoi l'on parle de tâches concurrentes [Aloys, M.2003].

Afin de s'assurer que tous les objectifs du système i-PRL seront remplis, chacun des objectifs ou sous-objectifs du diagramme hiérarchisé des objectifs est assigné à un rôle particulier.

La méthodologie MaSE indique que chaque objectif doit être assigné à au moins un rôle, laissant ainsi supposer qu'un objectif peut être réalisé par plusieurs rôles mais AgentTool, l'environnement mis au point pour instancier MaSE, oblige à assigner un objectif à un rôle et un seul.

4.2.2. Phase de conception

La conception représente la deuxième phase pour la méthodologie MaSE comportant quatre étapes. La première étape concerne la création des classes d'agents où les rôles fixés précédemment sont associé à des types d'agent bien définis. Dans la deuxième étape, des interactions sont définies entre les différentes classes d'agents. La troisième étape permet de définir les processus de raisonnement des agents aussi bien leurs architectures interne.

Enfin, la dernière étape consiste à déployer le système où le concepteur doit définir le nombre et la localisation des agents.

a. Création des classes d'agents

Les rôles détaillés dans le diagramme de rôle de la phase d'analyse sont associés pour la création des classes d'agents. Chaque rôle identifié plus haut doit être rempli par au moins une classe d'agents. Alors qu'une classe d'agents peut jouer plusieurs rôles.

Le résultat de cette étape est le diagramme de classes d'agent qui montre l'organisation du système i-PRL en sept classes d'agents et les interactions entre elles. Le diagramme de classes d'agents est la première entité de MaSE décrivant le SMA tout entier et la façon dont il sera éventuellement implémenté. Une classe d'agent est similaire aux classes d'objets dans la POO (Programmation Orienté Objet) [Aloys, M. 2003].

Cependant, il existe deux différences. D'une part, les classes d'agents ne sont pas définies par les attribûts et les méthodes mais plutôt par les rôles à jouer. Et d'autre part, dans les classes d'agents, toutes les relations représentent des interactions.

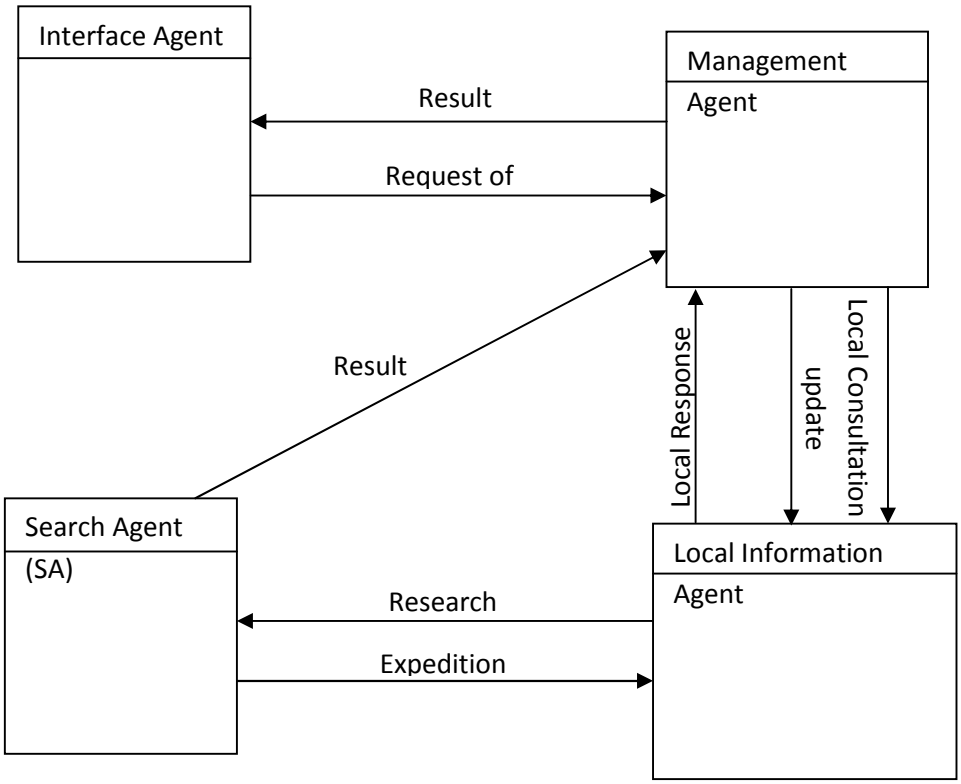

FIG 5.8 : *Diagramme de classes d'agents.*

b. Construction des interactions

Cette étape a pour bût de construire le détail des conversations entre agents. Dans la méthodologie MaSE, une conversation ou interaction définit un protocole de coordination entre deux agents. Ainsi, elle est représentée par deux diagrammes d'interaction: l'un pour l'émetteur et l'autre pour le destinataire. Ainsi, pour chaque diagramme une succession d'états (state) est présentée. En effet, le message de l'émetteur commence une telle communication et lorsque le destinataire reçoit le message, il le compare avec ses communications actives.

A cette étape deux cas se présentent: la première, s'il trouve une correspondance donc il opère la transition nécessaire et migre vers un nouvel état. Puis il effectue les actions nécessaires contenues dans la transition où le nouvel état dans lequel il se trouve.

La deuxième, si le destinataire ne trouve pas de correspondance avec le message reçu donc il s'agit d'une nouvelle communication et il la compare avec toutes les interactions pouvant apparaître avec l'agent émetteur. Ainsi s'il trouve une correspondance donc il engage une nouvelle communication. Comme montrer dans les figures suivantes, chaque diagramme

commence toujours par un cercle (vert) indiquant le début de la conversation et finit par deux cercles (rouge). Le rectangle est composé de deux parties: la première partie c'est le nom de l'état et la deuxième c'est l'action de l'état.

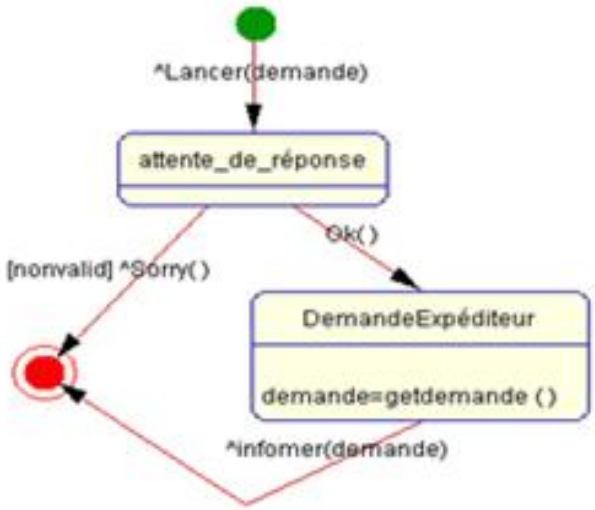

FIG 5.9 : *Diagramme de conversation initiale.*

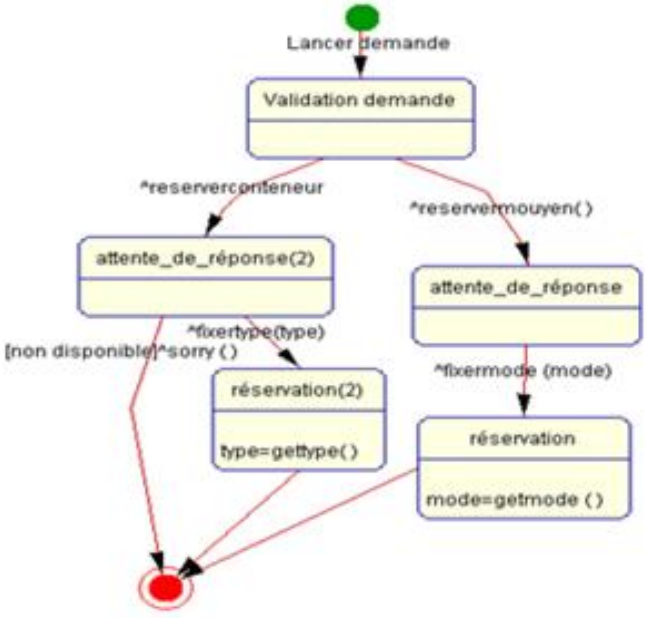

FIG 5.10 : *Diagramme de conversation réponse***.**

c. Assemblage des classes d'agents

Durant cette étape, l'architecture interne des agents est définie. Elle renferme deux sous- étapes : définition de l'architecture et définition des composants de cette architecture.

D'après [Aloys, M. 2003], les auteurs de MaSE perçoivent la possibilité de définir directement une architecture à partir des tâches identifiées dans la phase d'analyse. L'idée est que chaque tâche d'un rôle devienne un composant dans la classe d'agents correspondante. Chacun de ces composants est constitué en blocs regroupant tous les paramètres et les variables nécessaires à la réalisation de la ou les tâches destinées à l'agent et toutes les actions définies sous forme de méthodes. Les paramètres utilisés sont précisés pour chaque méthode.

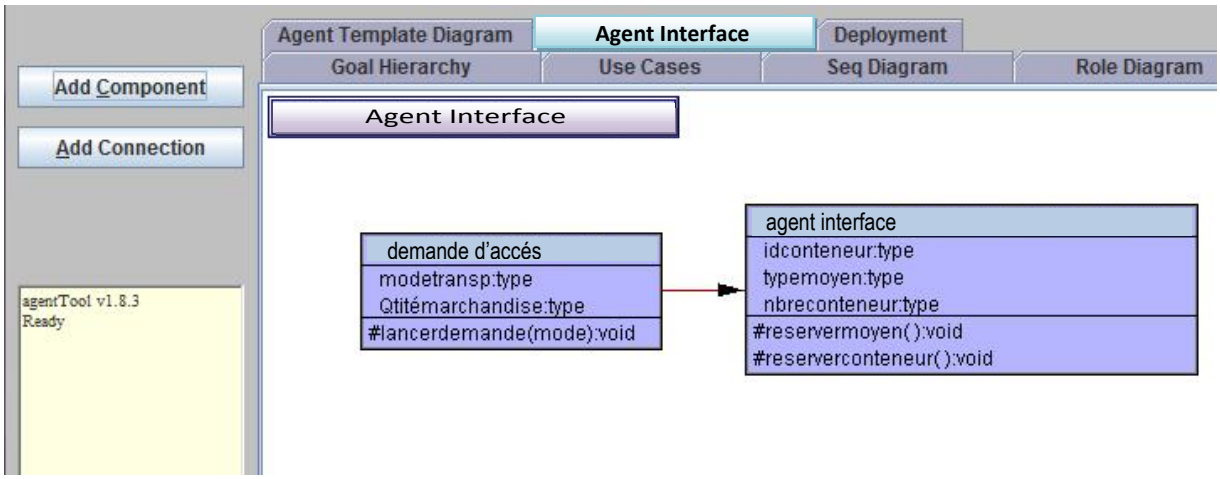

Figure 5.11 : *Architecture Agent « Interface».*

d. Conception du système

C'est la dernière étape pour la méthodologie MaSE : la conception du système est en fait la plus simple phase dans la MaSE, puisque la plupart des travaux ont été effectués dans les étapes précédentes. En effet, elle consiste à instancier les classes d'agents définies précédemment. Dans MaSE, le diagramme de déploiement permet de définir l'architecture globale du système en précisant le nombre, les types et emplacements des agents au sein du système [Deloacha, S et al, 2001].

En outre, comme le montre la figure 3.19, dans le diagramme de déploiement du système i-PRL, les boîtes en trois dimensions indiquent les agents alors que les lignes représentent les conversations réelles entre ces agents. Ces derniers sont nommés d'après le diagramme de classes d'agents et toute conversation entre classes d'agents apparaît dans la

conception du système. Cependant, la boîte en pointillé indique que les agents s'exécutent sur la même plate-forme physique.

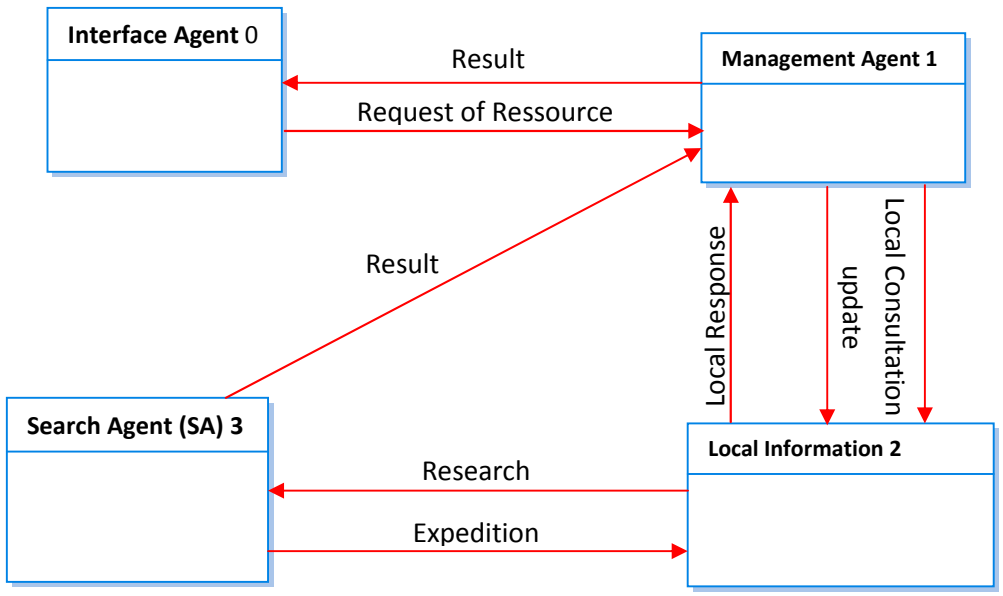

Figure 5.12 : *Diagramme de déploiement.*

5. Implémentation du système i-PRL

Pour développer des SMA, il est pratique de choisir une plate-forme. Et après une étude détaillée des plates-formes existantes (cf. chap 4), nous avons choisi la plate-forme JADE (Java Agent Development Framework). Ce choix de la plateforme JADE a été motivé premièrement par sa disponibilité, elle est open source. En fait, il s'agit d'un logiciel libre et facile à mettre en place et bien documenté. D'autres raisons nous ont poussées à sélectionner cette plateforme, comme la conformité aux spécifications de FIPA (Foundition for Intelligent Physical Agent) et la documentation abondante sur sa mise en œuvre. De plus, disposant des méthodes d'interfaçage avec le langage JAVA, elle offre la possibilité d'introduire les fichiers Java de l'ontologie utilisée. Aussi JADE n'exige pas une méthodologie de conception de SMA, donc il nous laisse le choix de la méthode la plus adaptée pour notre système.

JADE comprend deux composantes de base : une plate-forme agents compatible FIPA (Foundition for Intelligent Physical Agent) et un paquet logiciel pour le développement des agents Java. JADE possède des agents prédéfinis comme le Directory Facilitator (DF), l'Agent Management System (AMS) et le Remote Monitoring Agent (RMA).

– **L'agent DF** enregistre les descriptions et les services des agents qui permettent de mettre en relations les agents avec leurs compétences et son GUI (Graphical User Interface) peut être lancé du menu outils du RMA. Un agent peut enregistrer ses compétences dans le DF ou interroger le DF pour connaître les compétences proposées par les autres agents.

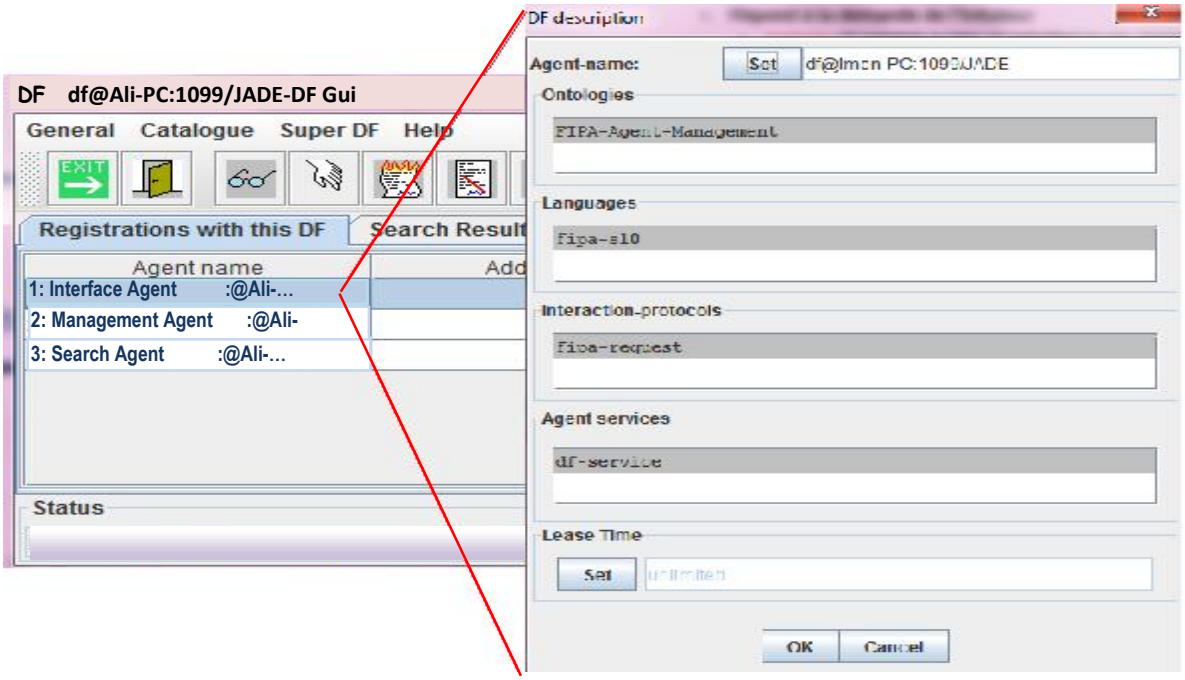

FIG 5.13 : Interface graphique de l'agent DF de JADE

- **L'Agent AMS** supervise la plate-forme entière. Il est le point de contact pour tous les agents qui ont besoin d'interagir pour accéder aux pages blanches de la plate-forme ainsi que de gérer leur cycle de vie. Chaque agent est requis pour s'enregistrer avec l'AMS (effectuées automatiquement par Jade à agent start-up).

– **L'agent RMA** sert d'administrateur pour les agents. Avec son interface graphique (Figure 3.21), il est possible de créer, supprimer et de migrer des agents. L'agent RMA permet également la création et la suppression des conteneurs où vivent les agents. Il y a un Container qui s'appelle Main Container qui gère les autres Containers dans le système (c'est Container-1 pour notre cas). Comme le montre la figure, nous avons implémenté système i-PRL.

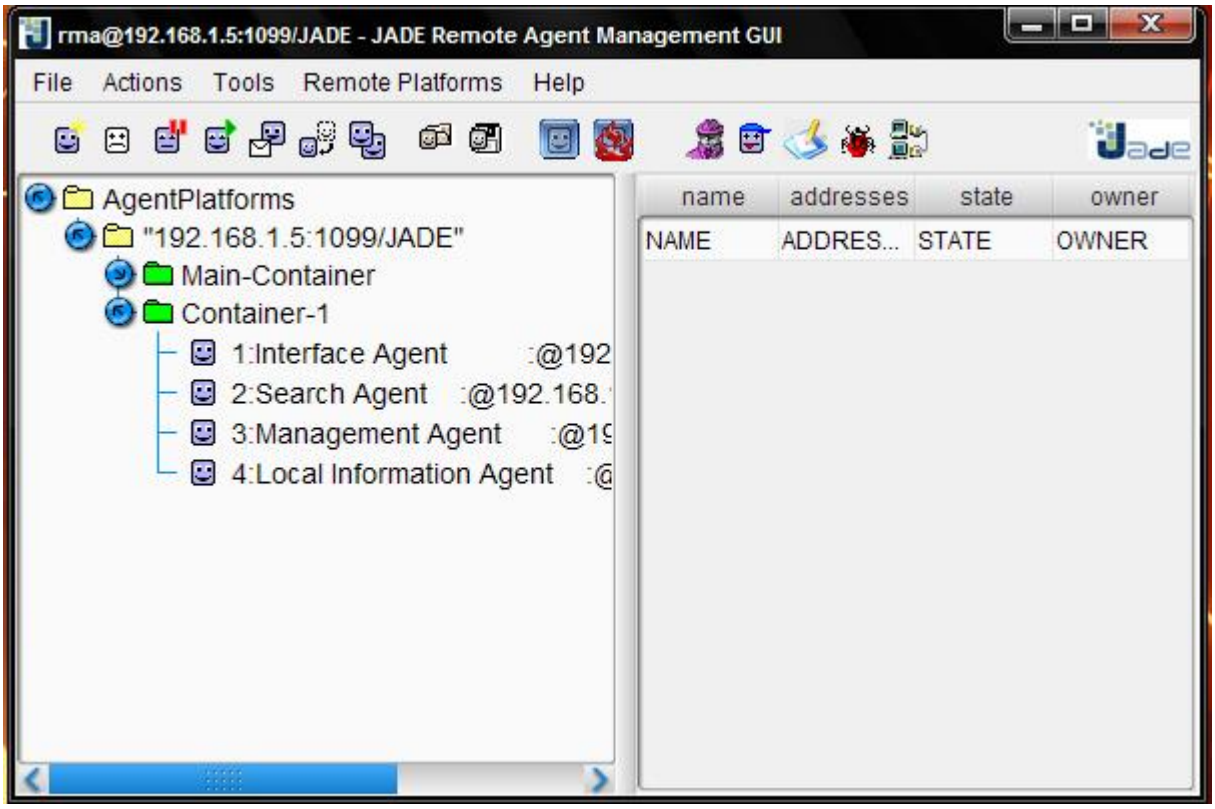

FIG 5.14 : *Interface graphique de l'agent RMA de JADE*

Scénario d'exécution : communication entre les agents (Sniffer)

La communication entre deux agents (agent A1 et agent A2) avec JADE fait intervenir deux actions, celle de l'envoi du message, matérialisée par la méthode send ( ), et celle de la réception du message, correspondant à la méthode receive ( ). Ainsi, le support des actes de communication fourni par JADE est conforme à la spécification FIPA. Afin que les agents puissent communiquer, ils doivent partager le même langage, le même vocabulaire et les mêmes protocoles. Tous ces actes de communication obéissent au vocabulaire d'une ontologie dont nous présentons le détail dans la section suivante. En effet, quand l'utilisateur décide de renifler un agent, ou un groupe d'agents, chaque message dirigé vers ou venant de cet agent, ou du groupe, est dépisté et montré dans la fenêtre de Sniffer. L'utilisateur peut alors regarder, sauvegarder, charger chaque message pour une analyse postérieure.

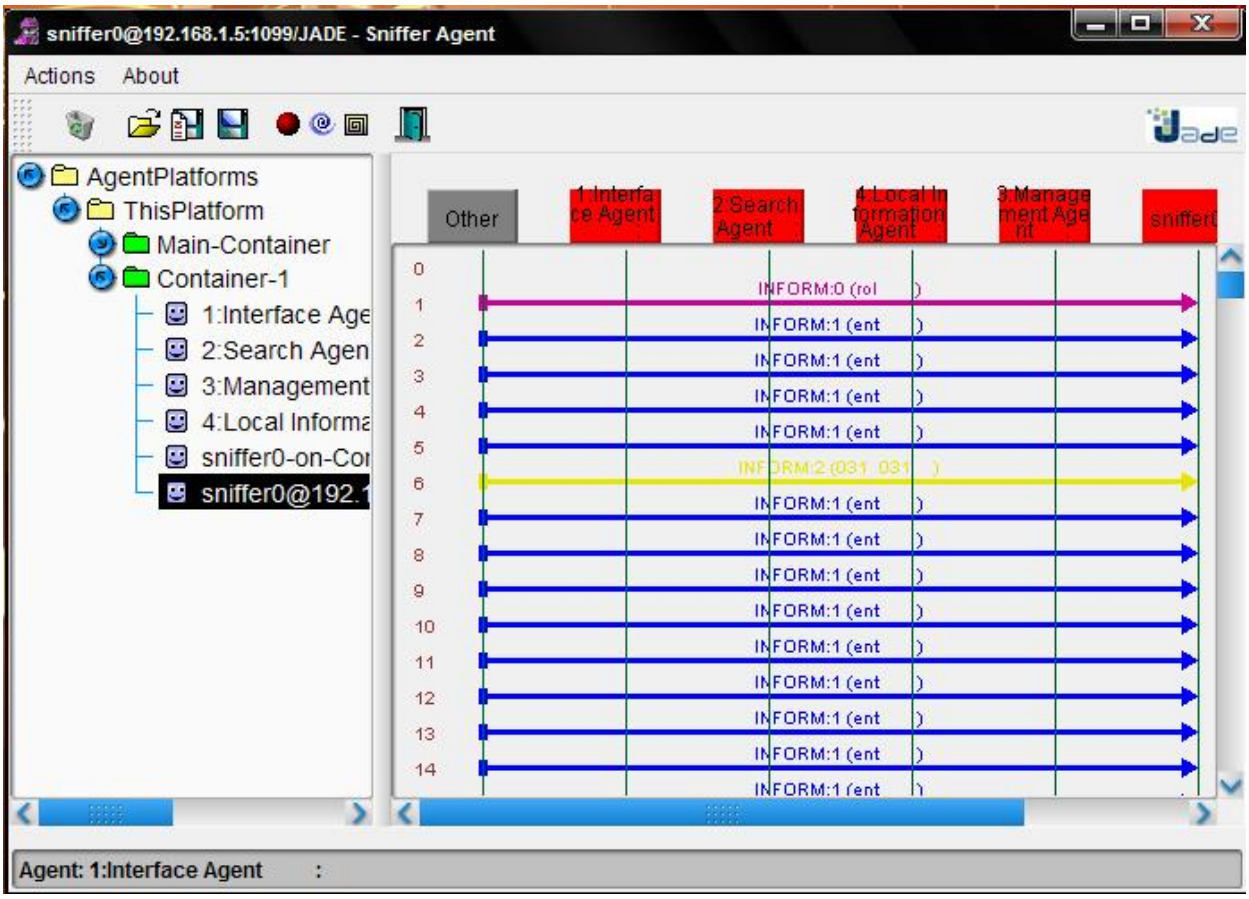

FIG 5.15 : *Interface graphique de l'agent Sniffer de JADE*

6. Construction de l'ontologie

L'ontologie « Traçabilité » décrit les concepts qui peuvent être utilisés dans le contenu des messages transmis entre les agents du système i-PRL. Cette ontologie est composée de deux parties : un vocabulaire qui décrit la terminologie des concepts utilisée dans la communication et la nomenclature des relations entre ces concepts et une sémantique. Pour construire l'ontologie, nous avons eu recours à un éditeur d'ontologie. Celui-ci comporte un outil et un environnement. Il est utilisé non seulement pour construire une nouvelle ontologie, mais aussi pour réutiliser des ontologies existantes. L'éditeur pour lequel nous avons opté est protégé2000. Cet éditeur est facile pour comprendre comment construire une ontologie. Aussi, il permet au développeur d'accéder aux informations pertinentes rapidement au moment souhaité, et d'utiliser une manipulation directe pour naviguer et gérer une ontologie. Dans la figure 5.23, nous exposons l'ontologie de communication dans i-PRL.

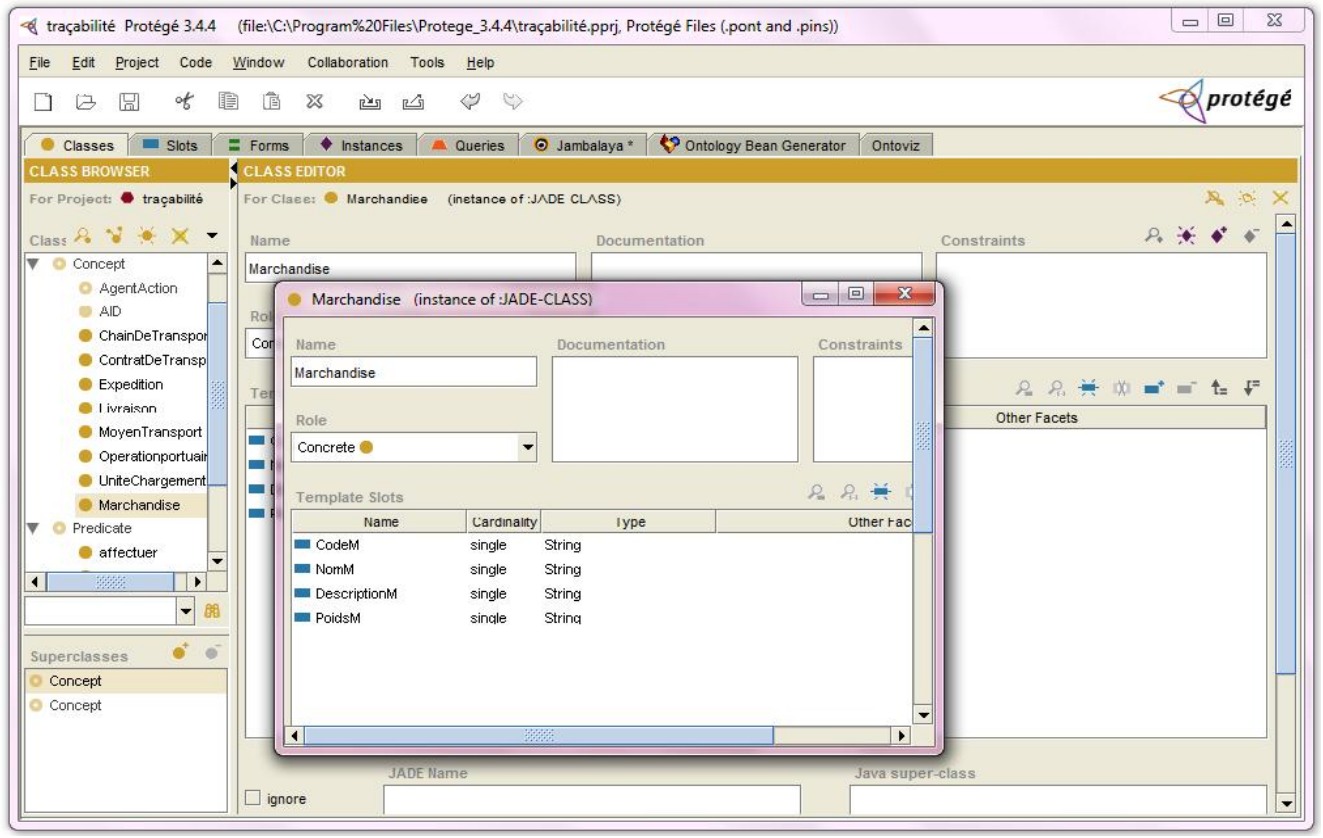

FIG 5.16 : *Création de l'ontologie par l'éditeur protégé2000*

Un des avantages de l'éditeur protégé2000 est de pouvoir générer des fichiers Java sur l'ontologie construite. Ces fichiers peuvent être introduits dans la plate-forme JADE. En effet, il suffit de les construire avec l'éditeur. Et avec le plug-in BeanGenerator, la génération des fichiers java sur l'ontologie se fera d'une manière automatique. Dans la figure 5.17, nous illustrons une génération automatique des fichiers java à partir du plug-in BeanGenerator.

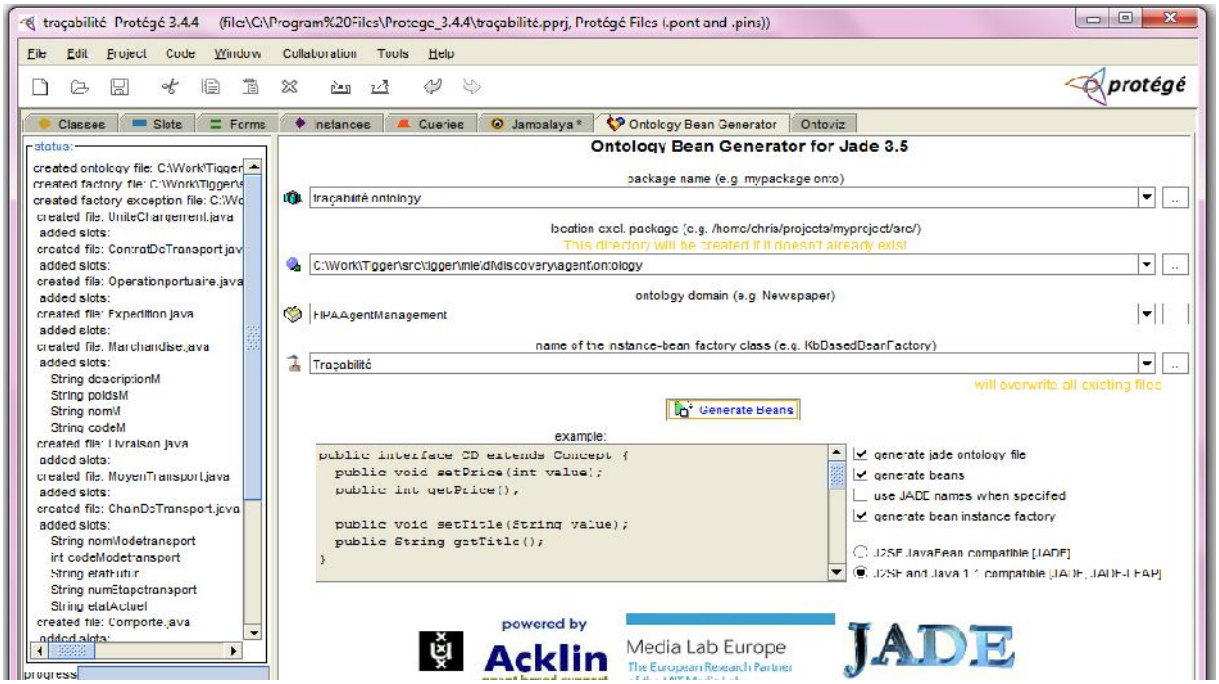

FIG 5.17 : *Génération des fichiers Java de l'ontologie avec le plug-in BeanGenerator*

7. Quelques interfaces du système i-PRL

- **Page d'accueil :**

A partir de la page d'accueil, l'utilisateur peut accéder à la page principale du i-PRL en cliquant sur " Entrer " puis en saisissant un login et un mot de passe valide.

FIG 5.18 : *Page d'accueil*

- **Interface d'inscription :**

Elle permet à un nouvel utilisateur de s'inscrire dans système. Il s'agit d'un formulaire renfermant les informations personnelles de l'utilisateur à enregistrer.

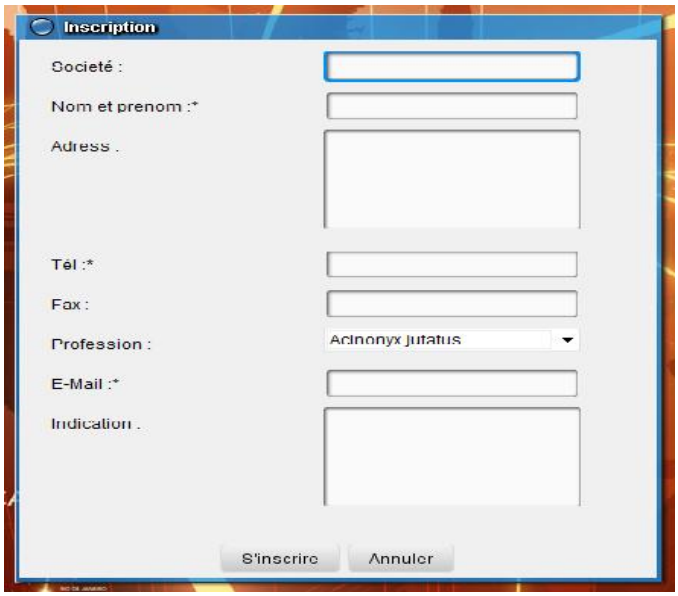

Figure 5.19 : Interface d'inscription

- **Interface principale du système i-PRL :**

Comme montre la figure ci-dessous, cette interface contient également plusieurs fonctionnalités comme : la recherche des ressources selon un critère fixé par l'utilisateur, la manipulation des listes de formulaires exploitées par les agents du système pour l'échange d'informations entre eux et l'affichage de la communication entre les agents.

FIG 5.20 : *Interface principale du système i-PRL*

- **Interface de formulaire demande de ressource :**

Tout d'abord, il faut saisir les différentes données qui constituent cette interface. Puis, en cliquant sur " confirmer ", ces informations seront transférées vers l'agent interface.

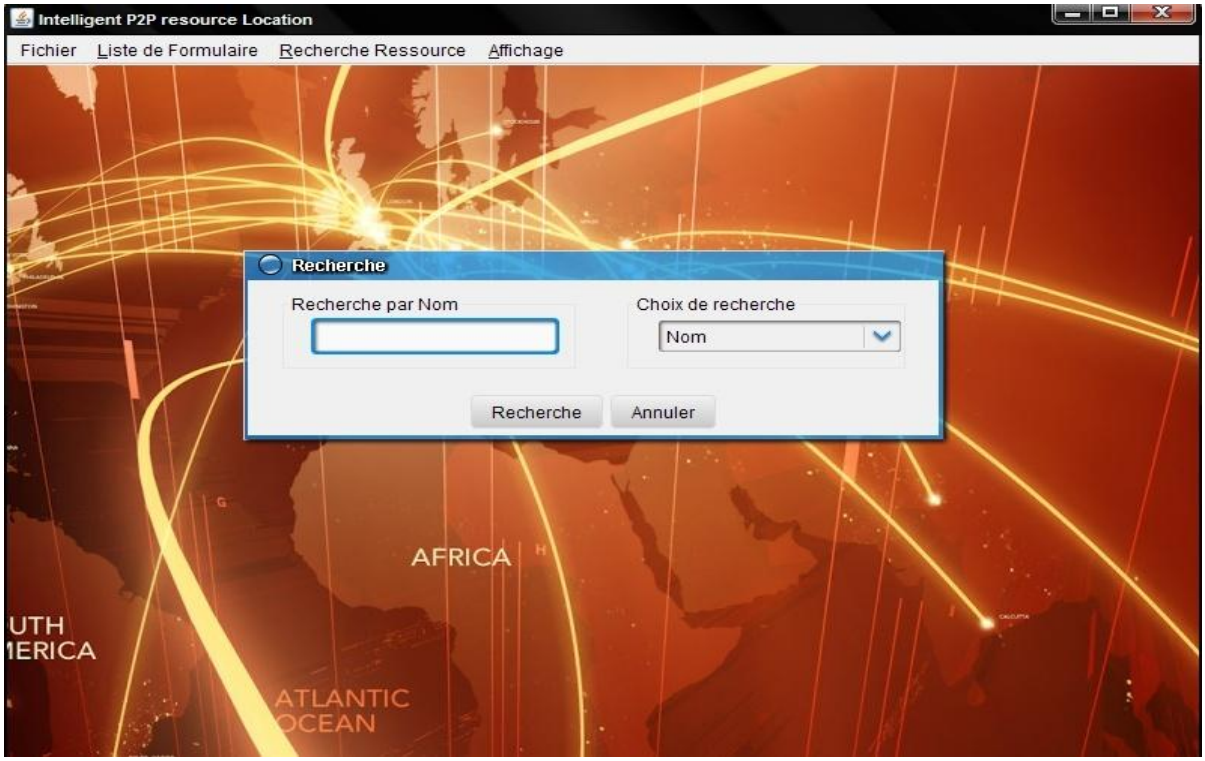

FIG 5.21 : *Interface de Recherche de ressource*

8. Résultats et simulation

En comparant notre approche avec la méthode flooding (inondation de requêtes), les premières expériences numériques à travers la simulation ont montré une réduction significative de messages générés "figure 5.22", également en utilisant le renifleur de jade, il a été vu une bonne coopération et communication entre les agents. Notre système est encore en expérience, nous allons présenter de nouveaux résultats dans l'avenir.

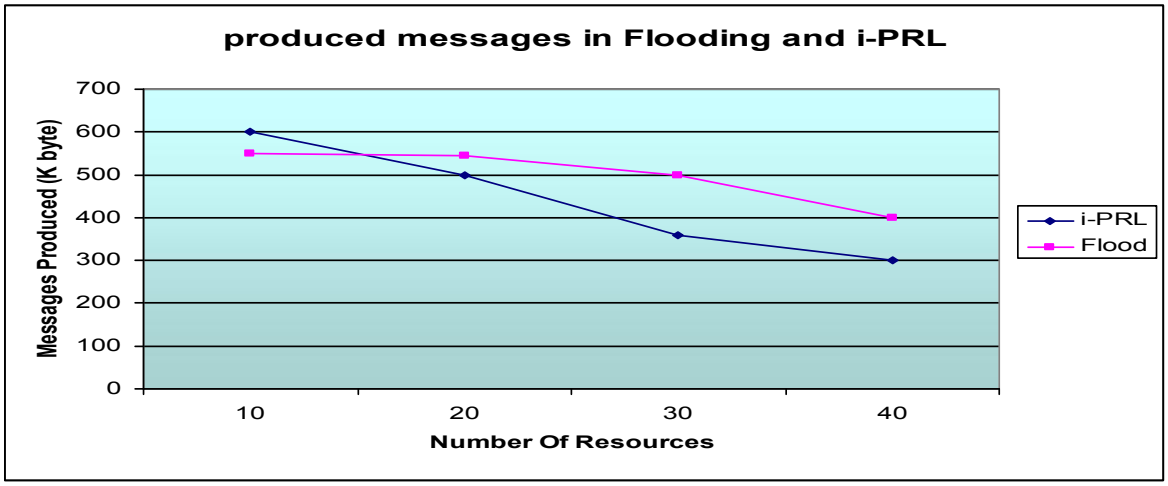

FIG 5.22 *Comparaison entre la méthode Flooding et i-PRL Framework.*

9. Conclusion

Au cours de ce chapitre nous avons présenté notre contribution pour assurer la localisation des ressources dans un système peer to peer pur à travers la modélisation et le développement d'un système basé sur les technologies des Systèmes Multi-Agent et le web sémantique.

Nous avons commencé ce chapitre par la présentation de l'architecture générale du système i- PRL puis nous avons présenté son architecture détaillée composée de cinq couches.

Ensuite, nous avons proposé l'analyse et la conception du système i-PRL par la méthodologie MaSE (Multi-agent System Engineering).

En fin, ce chapitre s'achève par l'implémentation du système de localisation en se basant sur la plate-forme JADE pour le développement des agents, Et on se basant sur l'éditeur protégé2000 et le plugin Beangenerator pour introduire l'ontologie .

Conclusion Générale

CONCLUSION ET PERSPECTIVES

A travers ce mémoire de magister, nous nous sommes intéressés à un sujet qui porte d'importance dans plusieurs travaux de recherche celui de la localisation des ressources dans les systèmes P2P.

Ce travail est la première étape dans notre recherche sur l'application des agents mobile et le web sémantique dans les systèmes P2P, Pour se faire, nous avons élaboré dans un premier temps, une étude sur le contexte de travail. Nous avons présenté les concepts de bases des systèmes peer to peer et les différentes architectures. Dans un second temps, nous avons exposé les différents modèles et approches abordant la problématique de la localisation.

Nous avons présenté également les Systèmes Multi-Agent (SMA), les ontologies et leurs concepts fondamentaux. Nous avons mis en relief la relation entre ces deux approches.

En se basant sur l'état de l'art, nous avons élaboré une architecture intelligente basée sur l'accouplement du paradigme agent et ce lui de l'ontologie pour assurer la localisation des ressources. Cette architecture est composée de cinq couches : accès et connexion, communication, application, ontologie et Peer. Nous avons présenté un système intelligent baptisé i-PRL (intelligent P2P Resource Location) tout en couplant les deux technologies.

Les premières expériences ont montré une réduction significative de messages générés en le comparant avec la méthode flooding et une bonne coopération entre les agents, dans la prochaine étape nous allons travailler sur la performance de notre système

Ce système à été l'objet de communication et publication dans des conférences nationales et internationales.

Cette proposition reste une tentative dans le domaine de la recherche, Nous espérons que cette modeste contribution gagnerait à susciter l'intérêt chez d'autres chercheurs pour des développements futurs dans le domaine de la technologie des P2P et agents mobiles et plus particulièrement la localisation des ressources.

Plusieurs pistes peuvent être exploitées notamment La sécurisation du système contre les pairs malveillants, l'augmentation de la fiabilité et la sécurisation des agents mobiles.

Bibliographie

Références Bibliographiques

[Aloys, 2003] Aloys Mbala Hikolo « Analyse, conception, spécification et développement d'un système multi-agents pour le soutien des activités en formation à distance », l'Université de Franche-Comté, 2003

[Antoine, 2005] Antoine Isaac « Conception et utilisation d'ontologies pour l'indexation de documents audiovisuels », 2005

[Briot, 2001] Jean-Pierre Briot et Yves Demazeau, Principes et architecture des systèmes multi-agents, Collection IC2, pages 17-25. Hermes Science Publications, Paris, France, 2001.

[Baghdad et Zerhouni,2003] L. Baghdad et A. Zerhouni, *Agents Mobiles Anonymes*, Département Informatique, USTHB, 2003

[Bensaid et P. Mathieu,1997] N. E. Bensaid and P. Mathieu, "A hybrid and hierarchical multi-agent architecture model," presented at the Proceedings of the Second International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'97), London-United Kingdom.

[Bouchnak, 2001] S. Bouchnak, Mobilité et Persistance des Applications dans l'Environnement Java, Institut National Polytechnique de Grenoble, 2001. http://sardes.inrialpes.fr/papers/files/01-Bouchenak-PhD.pdf

[Bouron,1992] T. Bouron, "Structures de communication et d'organisation pour la coopération dans un univers multi-agents," Université Pierre et Marie Curie (Paris VI), 1992

[Braun,2005] Braun.P,Wilhelm R. Rossak "Mobile Agents: Basic Concepts, Mobility Models, and the Tracy Toolkit ", Morgan Kaufmann, 1 edition, Vol 464,p 15-70

[Chaib-draa, et al,2001] B. Chaib-draa, *et al.*, systèmes multi-agents : principes généraux et applications: Hermès Sience Publications, Lavoisier, 2001.

[Dimakopoulos,2003].DimakopoulosV.V.,Pitoura,E"A peer-to-peer approach to resource discovery in multi-agent systems," In Proc. of CIA2003, Springer-Verlag, Lecture Notes on Computer Science, Vol. 2782,p 62-77.

[Deloacha, S et al, 2001] Deloach, S, Wood, A, Sparkman, C «Multiagent Systems Engineering. International Journal on Software Engineering and Knowledge Engineering ». World Sientific Publishers, 2001

[Dunne,2001] Dunne.C.R " Using Mobile Agents for Network Resource Discovery in Peer-to-Peer Networks" School of Computer Applications, Dublin City University, Dublin 9, Ireland.

[Esmahi,2001] L. Esmahi, "Modèles formels pour la coopération dans les SMA," presented at the Actes des 9ème Journées Francophones d'Intelligence Artificielle et Systèmes Multi-Agents (JFIADSMA'01), 2001.

[Esposito,2000] Nicolas Esposito, *Les agents mobiles – Une introduction*, DASSAULT SYST`EMES – Recherche et nouvelles technologies, France, 2000 http://www.utc.fr/~nesposit/publications/esposito2000.pdf

[Federico, B et al. 2002] Federico Bergenti, Agostino Poggi, Giovanni Rimassa et Paola Turci « Comma: a multiagent system for corporate memory management » AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems, pages 1039–1040, 2002

[Ferber,1995] J. Ferber. "Les systèmes multi-agents, vers une intelligence collective". InterEditions, 1995.

[Francisco, G et al. 2008] Francisco García-Sánchez, Jesualdo Tomás Fernández-Breis, Rafael Valencia-García, Juan Miguel Gómez, Rodrigo Martínez-Béjar «Combining Semantic Web technologies with Multi-Agent Systems for integrated access to biological resources », Journal of Biomedical Informatics, Volume 41, Issue 5, Pages 848-859, October 2008

[Gandon, F. 2002] Gandon Fabien « Ontology Engineering: a Survey and a Return on Experience ». INRIA, 2002

[Gaudette,2003] Dan Gaudette, *Mobile Agents: An Introduction*, Lakehead University, Canada, 2003 http://ccc.cs.lakeheadu.ca

[Genco,2007]Genco,A. "Mobile Agents: Principles of Operation and Applications (Advances in Management Information",WIT Press; 1 edition, Vol 304,p8-105.

[Gherbi,2000] Tahar Gherbi, *Structuration d'applications reparties dans un environnement mobile*, Thèse Magister, l'U.S.T.H.B., 2000

[Guy, 1999] Guy Bernard, Applicabilité et performances des systèmes d'agents mobiles dans les systèmes répartis, Première Conférence Française en Systèmes d'Exploitation (CFSE'1), Rennes, France, Juin 1999.

[Hagimont,2001] D. Hagimont, L. Ismail**,** *Agents mobiles et client/serveur : évaluation de performance et comparaison*, Article de recherche, INRIA, France, 2001 http://taha.inrialpes.fr/publi/TSI.pdf

[Harrison, 1995] Harrison C. G., Chess D. M., Kershenbaum A., Mobile agents: Are they a good idea*?*, Technical Report, IBM T. J. Watson Research Center, March 1995. http://www.research.ibm.com/massive/mobag.ps

[Kevin, O. 2009] Kevin Ottens « Un système multi-agent adaptatif pour la construction d'ontologies à partir de textes », Laboratoire d'accueil : Institut de Recherche en Informatique de Toulouse Equipe d'accueil : Systèmes Multi-Agents Coopératifs. 2009

[Kubiak,2007] Kubiak.S P2P Networks: principles and simulation, Universität Duisburg-Essen (Lehrstuhl Technik der Rechnernetze), 22 Eintragungen im Literaturverzeichnis GRIN Verlag. 44.

[Leriche, 2004] S.Leriche, J. P. Arcangeli,Une architecture pour les agents mobiles adaptables,Toulouse,France,004.http://www.lifl.fr/jc2004/articles/leriche-arcangeli.pdf

[Lübke,2004]Lübke.D ,J. M. Gómez. "Applications for mobile agents in Peer-to-Peer-Networks". In Engineering ofComputer-Based Systems.

[Magedanz,1998] T. Magedanz, *Agent Activities within ACTS - An Overview and Impacts on TINA*, IKV++ GmbH. ACTS. CLIMATE. TINA Forum Meeting - Heidelberg, 1998.

[Magnin ,1999] Laurent Magnin, *Internet, environnement complexe pour agents situés*, Centre de Recherche en Informatique de Montréal, Canada, 1999

[Mark, T et al.2003] Mark T. Elmore, Thomas E. Potok et Frederick T. Sheldon « Dynamic data fusion using an ontology-based software agent system. » Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics, 2003

[Mark, T et al.2003] Marc Savall « Une architecture d'agents pour la simulation---Le modèle YAMAM sa plate-forme Phoenix, »THESE présentée pour l'obtention du Doctorat de l'INSA de Rouen (Spécialité informatique), 2003

[Mathieu, et al,2002] P. Mathieu*, et al.*, "Dynamic organization of multi-agent systems," presented at the Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1, Bologna, Italy, 2002.

[Menacer, 2004] Menacer Djamel eddine, Un modèle d'architecture à base d'agents mobiles pour les applications réparties, Thèse Magister, Institut National d'Informatique (INI), Novembre 2004.

[Milojicic,2002] Milojicic. D. S, Kalogeraki. V, Lukose. R, Nagaraja. K,. Pruyne. J, Richard. B, Rollins. S,. Xu.v" Peer to-peer computing". Technical report, HP Laboratories Palo Alto.

[Moro,2005] Moro.G, Sartori.C , Munindar.P. Singh "Agents and Peer-to-Peer Computing": Second International Workshop, AP2PC 2003, Melbourne, Australia, July 14, 2003, Revised and Invited Papers (Lecture ... / Lecture Notes in Artificial Intelligence).

[Perret, 1997] Stéphane Perret, Agents mobiles pour l'accès nomade à l'information répartie dans les réseaux de grande envergure, Thèse Doctorat, Uversité Joseph Fourier, Grenoble I, 1997,

http://rangiroa.essi.fr/rapports/1900/96-these-perret.pdf

[Reuter, 2000] Emmanuel Reuter, Les Agents Mobiles et Actifs pour l'Administration de Réseaux, DEA Réseaux et Systèmes Distribués, INRIA, France, 2000 http://www.iufm.unice.fr/~reuter/rapport-dea.pdf

[Sahai, 1999] Akhil Sahai, Conception et réalisation d'un gestionnaire mobile de réseaux fondé sur la technologie d'agent mobile, Thèse Doctorat, Université de Rennes1, 1999. http://ftp.irisa.fr/techreports/theses/1999/sahai.ps

[Saroiu, 2002] Saroiu. S , et al., "A Measurement Study of Peer-to-Peer File Sharing Systems," presented at the Multimedia Computing and Networking 2002 (MMCN'02), San Jose, California, United States.

[Schollmeier, 2002] Schollmeier, A. 2002. A Definition of Peer-to-Peer Networking for the Clas- sification of Peer-to-Peer Architectures and Applications. *In : Proceedings of the First International Conference on Peer-to-Peer Computing*.

[Siebes, R et al. 2002] Siebes Ronny et Frank van Harmelen « Ranking agent statements for building evolving ontologies. Workshop on Meaning Negotation », in conjunction with the Eighteenth National Conference on Artificial Intelligence, July 2002

[Thierno, 2000] B. Thierno, Le problème d'interopérabilité entre les plate-formes d'agents mobiles, Erricson, Canada, CAT 2000 http://www.iro.umontreal.ca

[Van,D et al.2006]Van digglen Jurriaan, Beun Robbert-Jan, Dignum Frank, Van Eijk Rogier, and Meyer John-Jules. ANEMONE: An effective minimal ontology negotiation environment. In Proc. of AAMAS, 2006

Annexes

1. Représentation de la plate-forme JADE

Le meilleur moyen pour construire un système multi-agent (SMA) est d'utiliser une plate-forme multi-agent. Cette dernière est un ensemble d'outils nécessaire à la construction et à la mise en service d'agents au sein d'un environnement spécifique. Ces outils peuvent servir également à l'analyse et au test du SMA ainsi créé. Ces outils peuvent être sous la forme d'environnement de programmation (API) et d'applications permettant d'aider le développeur. Nous allons étudier dans cette section la plate-forme JADE[1].

JADE est une plate-forme multi-agents développée en Java par **CSELT** (Groupe de recherche de Gruppo Telecom, Italie) qui a comme but la construction des systèmes multi-agents et la réalisation d'applications conformes à la norme **FIPA**[2] (FIPA, 1997). JADE comprend deux composantes de base : une plate-forme agents compatible FIPA et un paquet logiciel pour le développement des agents Java.

1.1 Pourquoi la plateforme JADE?

On a développé notre application sous la plate forme multi agent JADE qui offre les avantages suivants :

- Plate forme assez facile à mettre en place.
- Disponibilité de packages sur lesquels nous nous sommes appuyés pour développer notre application.
- Documentation claire et complète.
- Licence gratuite.

1.2 La norme FIPA pour les systèmes multi-agents

Les premiers documents de spécification de la norme FIPA, appelés spécifications FIPA97, établissent les règles normatives qui permettent à une société d'agents d'inter-opérer. Tout d'abord, les documents FIPA décrivent le modèle de référence d'une plate-forme multi-agents (**FIG 1**) où ils identifient les rôles de quelques agents clés nécessaires pour la gestion de la plate-forme, et spécifient le contenu du langage de gestion des agents et l'ontologie du langage.

[1] **JADE**: **J**ava **A**gent **DE**velopment Framework.
[2] **FIPA**: **F**oundation for **I**ntelligent **P**hysical **A**gents.

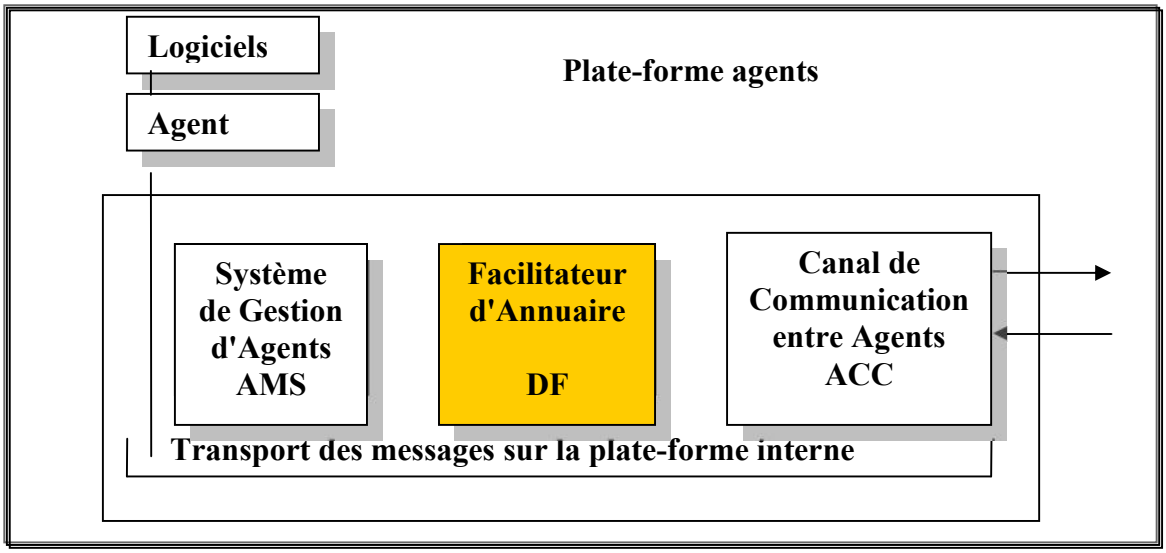

FIG 1. Le modèle de référence pour une plate-forme multi-agents FIPA

Dans la **FIG 1**, on voit qu'il existe trois rôles principaux dans une plate-forme multi-agents FIPA :

- Le **Système de Gestion d'Agents** (Agent Management System - **AMS**) est l'agent qui exerce le contrôle de supervision sur l'accès à l'usage de la plate-forme; il est responsable de l'authentification des agents résidents et du contrôle d'enregistrements.
- Le **Canal de Communication entre Agents** (**ACC**[3]) est l'agent qui fournit la route pour les interactions de base entre les agents dans et hors de la plate-forme; c'est la méthode de communication implicite qui offre un service fiable et précis pour le routage des messages; il doit aussi être compatible avec le protocole **IIOP**[4] pour l'interopérabilité entre les différentes plates-formes multi-agents.
- Le **Facilitateur d'Annuaire** (Directory Facilitator - **DF**) est l'agent qui fournit un service de pages jaunes à la plate-forme multi-agents.

Le standard spécifie aussi le **Langage de Communication d'Agents** (Agent Communication Language - **ACL**), la communication des agents est basée sur l'envoi de messages. Le langage FIPA **ACL** est le langage standard des messages et impose le codage, la sémantique et la pragmatique des messages. La norme n'impose pas de mécanisme spécifique

[3] **ACC**: **A**gent **C**ommunication **C**hannel.
[4] **IIOP** : Protocole de communication permettant d'intégrer et de mettre en réseau des applications de provenance diverse

pour le transport interne de messages. Plutôt, puisque les agents différents pourraient s'exécuter sur des plates-formes différentes et utiliser technologies différentes d'interconnexion, *FIPA* spécifie que les messages transportés entre les plates-formes devraient être codés sous forme textuelle. On suppose que l'agent est en mesure de transmettre cette forme textuelle.

1.3 L'environnement JADE

Le but de *JADE* est de simplifier le développement des systèmes multi-agents en conformité avec la norme *FIPA* pour réaliser des systèmes multi-agents interopérables. Pour atteindre ce but, *JADE* offre la liste suivante de **caractéristiques** au programmeur d'agents :

- **La plate-forme multi-agents compatible FIPA**, qui inclut le Système de Gestion d'Agents (**AMS**), le Facilitateur d'Annuaire (**DF**), et le Canal de Communication entre Agents (**ACC**). Ces trois agents sont automatiquement créés et activés quand la plate-forme est activée.

La plate-forme d'agents peut être distribuée sur plusieurs hôtes, à condition qu'il n'y ait pas de pare-feu entre ces hôtes. Une seule application Java, et donc une seule Machine Virtuelle Java, est exécutée sur chaque hôte. Les agents sont implémentés comme des threads d'exécution Java et les événements Java sont utilisés pour la communication efficace et légère entre agents sur un même hôte. Un agent peut exécuter des tâches parallèles et *JADE* planifie ces tâches d'une manière plus efficace (et même plus simple pour le programmeur) que la planification faite par la Machine Virtuelle Java pour les threads d'exécution.

- Un certain nombre de **DF** (Facilitateurs d'Annuaire) compatibles *FIPA* qui peuvent être activés quand on lance la plate-forme pour exécuter les applications multi-domaines.
- Une interface de programmation pour simplifier l'enregistrement de services d'agents avec un ou plusieurs domaines de type **DF**.
- Le mécanisme de transport et l'interface pour l'envoi et la réception des messages.
- Le protocole **IIOP** pour connecter des plates-formes multi-agents différentes.
- Le transport léger de messages **ACL** sur la même plate-forme d'agents. Dans le but de simplifier la transmission, les messages internes (sur la même plate-forme) sont transférés et codés comme des objets Java et non comme des chaînes de caractères.

Quand l'expéditeur ou le récepteur n'appartient pas à la même plate-forme, le message est automatiquement converti en chaîne de caractères. De cette façon, la conversion est cachée au programmeur d'agents qui a seulement besoin de traiter la classe d'objets Java.

- L'enregistrement automatique d'agents dans le Système de Gestion d'Agents (**AMS**).
- Un service d'attribution de noms compatible *FIPA* ; quand on lance la plate-forme, un agent obtient un identificateur unique (GUID[5]).
- Une interface graphique utilisateur pour gérer plusieurs agents et plates-formes multi-agents en partant d'un agent unique. L'activité de chaque plate-forme peut être supervisée et enregistrée.

1.4 L'architecture de la plate-forme multi-agents

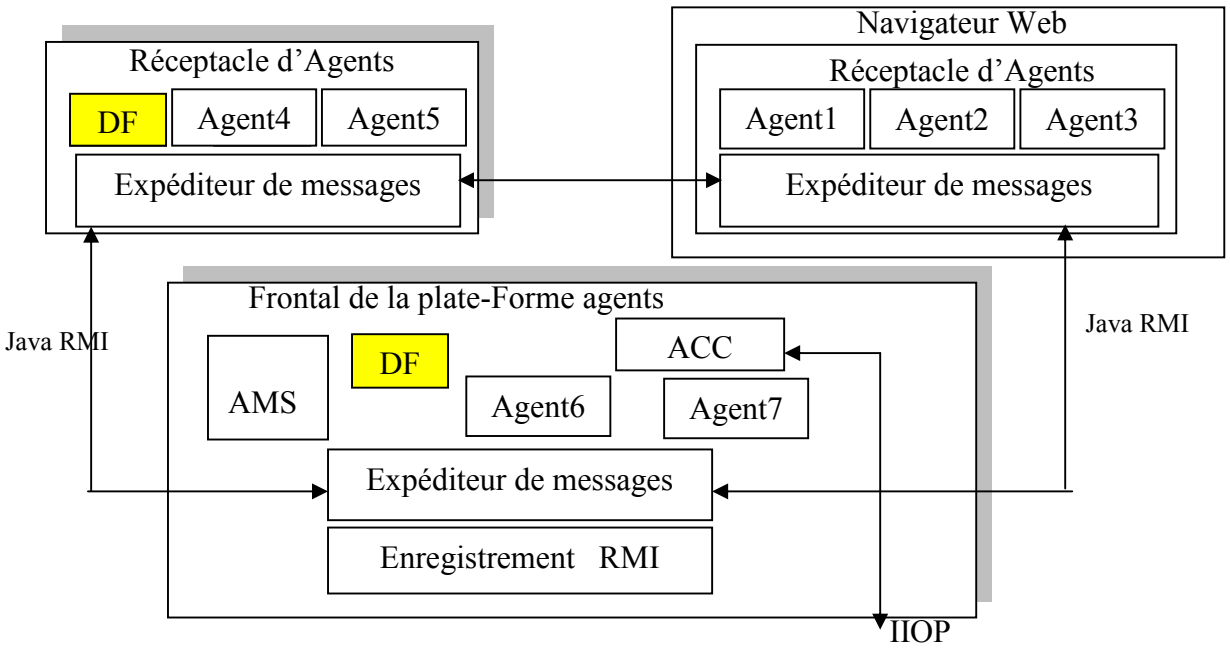

FIG 2. Architecture logicielle de la plate-forme multi-agents JADE

La plate-forme offre une interface graphique utilisateur (**GUI**[6]) pour la gestion à distance qui permet de contrôler et superviser les états des agents, par exemple arrêter et remettre en marche un agent. L'interface graphique permet aussi de créer et de commencer l'exécution d'un agent sur un hôte éloigné, à condition qu'un réceptacle d'agents s'exécute déjà sur cet hôte. L'interface elle-même a été implémentée comme un agent, appelé **RMA** (**R**emote **M**onitoring **A**gent).

[5] **GUID**: **G**lobally **U**nique **ID**entifier.
[6] **GUI**: **G**raphical **U**ser **I**nterface.

La propriété importante d'un agent est son autonomie : un agent ne doit pas se limiter à réagir aux événements externes, mais il doit être aussi capable de prendre l'initiative de nouveaux actes communicatifs d'une façon autonome. Ceci exige que chaque agent ait un thread interne de contrôle ; cependant, un agent peut engager des conversations simultanées multiples, tout en poursuivant d'autres activités qui n'impliquent pas d'échanges de messages.

JADE utilise l'abstraction *Comportement* pour modéliser les tâches qu'un agent peut exécuter et les agents instancient leurs comportements selon leurs besoins et leurs capacités.

De point de vue de la programmation concurrente, un agent est un **objet actif**, ayant un thread de contrôle. JADE utilise un modèle de programmation concurrente "un thread par agent" au lieu d'un modèle "un thread par comportement" pour éviter une augmentation du nombre de threads d'exécution exigés sur la plate-forme d'agents. Ceci signifie que, pendant que les agents différents s'exécutent dans un environnement multi-threads de préemption, deux comportements d'un même agent sont planifiés coopérativement.

En dehors de la préemption, les comportements travaillent tous comme des threads d'exécution coopératifs, mais il n'y a pas de pile qui ait besoin d'être sauvée. Un planificateur (Scheduler), exécuté par la classe de base *Agent* et caché au programmeur, exécute une politique de "round robin" de non préemption entre tous les comportements disponibles dans la file des processus prêts. Ainsi, il permet l'exécution d'une classe dérivée de la classe Comportement jusqu'à ce qu'elle abandonne le contrôle d'exécution par elle-même. Si la tâche qui a le contrôle n'est pas encore finie, elle sera re-planifiée pendant le prochain tour du round-robin à moins qu'elle ne soit pas bloquée ; en fait, un comportement peut se bloquer lui-même, par exemple pendant qu'il attend des messages, pour éviter le gaspillage de temps de CPU, réalisant ainsi un comportement d'attente occupée.

Donc, le développeur d'agents doit étendre la classe Agent et implémenter les tâches spécifiques de l'agent par une ou plusieurs classes Comportement, les instancier et les ajouter à l'agent. La classe Agent représente une super-classe commune pour tous les agents définis par l'utilisateur. Du point de vue du programmeur, la conséquence est qu'un agent JADE est simplement une classe Java qui étend la classe de base Agent. Cela permet à l'agent d'hériter un comportement fondamental caché (qui traite toutes les tâches liées à la plate-forme, telles que l'enregistrement, la configuration, la gestion à distance, etc.), et un ensemble de méthodes qui peuvent être appelées pour implémenter les tâches spécifiques à l'agent, par exemple

envoi des messages, utilisation des protocoles d'interaction standard, enregistrement sur plusieurs domaines, etc. De plus, il y a encore deux méthodes qui sont héritées pour gérer la file de comportements d'agents : *addBehaviour* (*Behaviour*) et *remove Behaviour* (*Behaviour*). JADE inclut aussi quelques comportements prêts à être utilisés pour les tâches les plus communes dans la programmation des agents, tels que l'envoi et la réception des messages et la décomposition des tâches complexes en des agrégations de tâches plus simples.

1.5 **Les Actes de communication**

Les actes de communications proposés par JADE sont : **agree**, **cancel**, **confirm**, **cfp**, **disconfirm**, **failure**, **inform**, **inform_if**, **query_if**, **query_ref**, **reject_proposal**, **request**, **request_when**, **request_whenever**, **subscribe**, **null**, **propose**, **not_understood**.

L'acte de communication traduit la nature du message envoyé entre agents. Les actes se classent dans des catégories, certaines permettent d'amorcer des protocoles alors que d'autre sont utilisés pour répondre. Ainsi **request**, **cfp**, **request_when**, **request_whenever**, **query_if**, **query_ref**, **subscirbe** sont utilisés pour amorcer un questionnement ou une demande de proposition et **agree, inform**, **failure, propose, not_understood**, sont des réponses possibles .Le plus usité est le **not_understood**, dès qu'un agent ne comprend pas le message reçu, il répond par un message **not_undertood** pour avoir plus de clarté.

Et voici, le **Tab 02** qui explique tous les performatifs :

Actions	Syntaxe	Définition - Sens
Accept Proposal	accept-proposal	Communication de l'accord de l'envoyeur d'effectuer une action qui lui a été préalablement soumise.
Agree	agree	Communication de l'accord de l'envoyeur pour effectuer une action, sans doute dans le futur.
Cancel	cancel	Communication de l'annulation de l'accord donnée préalablement par l'expéditeur pour effectuer une action.

Call for Proposal	Cfp	Communication par l'envoyeur d'une demande d'effectuer une certaine action.
Confirm	confirm	Communication par l'envoyeur de la confirmation de la validité (selon les règles de l'agent) de la proposition préalablement reçue.
Disconfirm	disconfirm	Communication par l'envoyeur de la confirmation de la non validité (selon les règles de l'agent) de la proposition préalablement reçue.
Failure	failure	Communication par l'envoyeur de l'échec d'une action essayée.
Inform	inform	Communication par l'envoyeur d'une proposition, pensée vrai par celui-ci.
Inform If	inform-if	Communication par l'envoyeur d'une proposition (pensée vrai par celui-ci), et demande au receveur une confirmation ou une non-confirmation.
Inform Ref	inform-ref	Communication par l'envoyeur d'une demande de l'objet qui correspond à une description envoyée.
Not Understood	not-understood	Communication par l'envoyeur d'une non compréhension d'une action effectuée par le destinataire.
Propagate	Propagate	Communication par l'envoyeur d'un message à propager à des agents dont la description est fournie. Le

		destinataire du message traite le sous message à propager comme s'il lui était directement destiné et envoie le message "propagate" au agent qu'il a identifié.
Propose	propose	Communication par l'envoyeur d'une proposition d'action conditionnée à certains prés conditions données.
Proxy	proxy	Communication par l'expéditeur d'une demande d'une transmission d'un message à des agents dont la description est donnée.
Query Ref	query-ref	Communication par l'envoyeur d'une demande par l'expéditeur de l'objet référencé par une expression.
Refuse	refuse	Communication par l'envoyeur de son refus d'effectuer une action donnée, et en donne les raisons.
Reject Proposal	reject-proposal	Communication, pendant une négociation, par l'envoyeur de son refus d'effectuer des actions.
Request	request	Communication par l'envoyeur d'une demande au destinataire d'effectuer une action.
Request When	request-when	Communication par l'envoyeur d'une demande, au destinataire, d'effectuer une action quand une proposition donnée devient vraie.
Request Whenever	request-whenever	Communication par l'envoyeur d'une

		demande, au destinataire, d'effectuer une action dès qu'une proposition donnée devient vraie, et à chaque fois que celle-ci redevient vrai.
Subscribe	subscribe	Communication par l'envoyeur d'une demande d'un objet donné par une référence envoyé par l'expéditeur, et de notifier l'agent ayant souscrit dès que l'objet en question change.

Tab 02 : Les actes de communication de FIPA-ACL.

JADE est un environnement de développement d'agents selon les spécifications de FIPA et implémenté totalement en JAVA. Il comprend tous les composants qui permettent la gestion de la dite plate-forme selon FIPA. Ces composants sont : ACC, AMS, et DF.

La plate-forme peut être répartie sur plusieurs hôtes. Une seule application JAVA, et par conséquent une seule machine virtuelle Java, est exécutée sur chaque hôte.

Chaque machine virtuelle Java est un conteneur d'agents fournissant un environnement complet d'exécution de ces agents et permettant à plusieurs agents concurrents de s'exécuter sur le même hôte.

Les agents communiquent à travers des messages représentés en FIPA-ACL. Le concept d'agent est vu par JADE comme un processus autonome et indépendant qui a une identité, qui requiert la communication (collaboration, compétition…) avec les autres agents dans le but de remplir ses missions.

En plus il comporte plusieurs outils (Dummy Agent, DF, RMA, Sniffer Agent, Introspector Agent) permettant de gérer et contrôler le cycle de vie de la plate forme, l'envoi de messages ACL et maintient une liste de messages ACL envoyés et reçus, voir la description des agents enregistrés, ajouter ou supprimer des agents, modifier la description sur les agents enregistrés.