UNIVERSITÉ DE MONTRÉAL

SOLUTION À BASE D'AGENTS MOBILES POUR LE COMMERCE ÉLECTRONIQUE

ALI CHAMAM

DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET DE GÉNIE INFORMATIQUE

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION

DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES

(GÉNIE ÉLECTRIQUE)

NOVEMBRE 2002

© Ali CHAMAM, 2002

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisitions et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-81534-X

Canada

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

CE MÉMOIRE INTITULÉ :

SOLUTION À BASE D'AGENTS MOBILES POUR LE COMMERCE ÉLECTRONIQUE

présenté par : CHAMAM Ali
en vue de l'obtention du diplôme de : maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen composé de :

Mme. BERNARD Jean-Charles, Ph. D., président
M. PIERRE Samuel, Ph. D., membre et directeur de recherche
M. GLITHO Roch, Ph. D., membre et codirecteur de recherche
M. QUINTERO Alejandro, Ph. D., membre

REMERCIEMENTS

Je tiens à exprimer mes plus vifs remerciements à mon directeur de recherches, le professeur Samuel Pierre, pour la qualité de son encadrement, ses précieux conseils et sa haute qualité humaine. Sa disponibilité et son support ont été un atout majeur pour le bon déroulement de ma maîtrise au sein du LARIM.

Je tiens à remercier également mon codirecteur de recherche, M. Roch GLITHO, chercheur à Ericsson Canada (Montréal), dont les conseils judicieux et les critiques de qualité tout au long de ce mémoire, m'ont été d'une aide considérable.

Ma profonde considération va aussi à mon cher père qui m'a encouragé à entreprendre cette maîtrise et qui n'a cessé de me fournir conseils et support moral et à ma chère mère dont l'encouragement et le soutien m'ont été d'une énorme motivation.

Je remercie aussi tous les membres du LARIM, pour leur aide et pour l'ambiance de travail qu'ils ont su maintenir, tout particulièrement Mme Sabine Kébreau, associée de recherche au LARIM, pour son aide technique et sa disponibilité.

Enfin, j'adresse un grand merci à tous mes collègues et amis qui, de près comme de loin, m'ont aidé à surpasser les moments de stress et de pression, tout particulièrement Riadh et Younes.

RÉSUMÉ

L'important essor qu'ont connu les technologies de l'information et les réseaux informatiques a favorisé l'émergence d'applications de plus en plus complexes telles les applications multimédia et le commerce électronique. Ces applications mettent en jeu des volumes importants de données transitant par le réseau. Le modèle client – serveur constitue l'architecture la plus employée et la mieux établie sur les réseaux de moyenne et grande taille tel Internet, pour la communication entre machines. Un client envoie une requête à un serveur qui la traite et renvoie le résultat au client. Cependant, face à l'émergence de nouvelles applications de plus en plus coûteuses en terme de taille et de temps d'exécution, les réseaux d'entreprises deviennent de plus en plus saturés, ce qui sanctionne directement le temps d'attente du client. Aussi, le développement des communications sans fil a permis aux utilisateurs du réseau de se connecter à partir de terminaux mobiles, souvent des téléphones cellulaires ou des ordinateurs portables sans fil, qui sont confrontés à une bande passante limitée et des liens pas toujours fiables. Pour faire face à ces nouveaux défis, une nouvelle approche a vu le jour il y a quelques années et est venue complémenter le mode client-serveur, c'est l'approche « agents mobiles ».

Les agents sont des programmes exécutables qui se déplacent d'une machine source vers une machine destination pour s'y exécuter. Ils ont comme principales caractéristiques l'autonomie, l'intelligence et surtout la mobilité. Malgré le manque d'applications à base d'agents mobiles sur le marché, beaucoup d'architectures sont fonctionnelles dans les laboratoires. Ces architectures mettent en jeu un ou plusieurs agents mobiles coopérant ensemble pour exécuter certaines tâches qui leur sont confiées par l'utilisateur.

Dans ce travail, nous étudions une application multi-agents mobiles pour le commerce électronique. Les agents mobiles doivent rechercher des prix admissibles, c'est-à-dire inférieurs à un certain seuil, d'un produit spécifique, sur un éventail de sites marchands répartis sur le réseau. Dès que les prix admissibles sont trouvés, la recherche est

achevée. Notre objectif est de trouver une stratégie de recherche de l'information, qui donnerait les meilleures performances. Les métriques de performance sont le temps de recherche global et la charge réseau induite.

Nous proposons, pour ce faire, deux stratégies de recherche de prix: une stratégie « individualiste » qui n'implique aucune collaboration entre agents et une stratégie « collaborative » qui implique une coordination entre agents mobiles. La coordination peut être de deux types : le premier repose sur un mécanisme de tableau noir (*blackboard*) à travers lequel les agents partagent et échangent les résultats de leurs recherches respectives. La seconde repose sur une communication directe entre les agents. Notre étude de performance est empirique, faute d'estimations précises sur certains paramètres comme les temps d'exécution sur une machine et la probabilité de trouver l'information cherchée sur un site donné. Nous procédons alors par scénarios : nous considérons deux cas de figure extrêmes, où les prix admissibles recherchés se trouvent respectivement sur les premiers et les derniers sites visités par les agents.

Les résultats montrent que chaque technique de collaboration peut être recommandée pour un besoin donné, mais dans tous les cas, la collaboration donne de meilleures performances qu'une stratégie « individualiste » sans coordination. D'une manière générale, le mécanisme du *blackboard* est recommandé lorsque nous cherchons absolument à optimiser le temps de recherche, sans souci pour la charge réseau. Si nous recherchons un « bon » compromis entre le temps de recherche et la charge réseau, le mécanisme de communication directe entre agents est recommandé, avec une condition limite sur le nombre d'agents.

Ce travail permet de préparer la voie à une utilisation plus efficace des agents mobiles dans un contexte sans fil. En effet, selon les besoins en qualité de service, l'une ou l'autre des stratégies peut être employée pour donner les meilleures performances. Sous un angle de vue d'applications de troisième génération, cette étude pourrait être étendue à un environnement sans fil et permettrait d'obtenir des résultats expérimentaux intéressants quant à l'utilisation des agents mobiles collaborant pour des applications de recherche d'information ou de commerce électronique.

ABSTRACT

The important development of information technology and computer networks have contributed to the emergence of more complex applications such as multimedia applications, voice over IP and electronic commerce. These applications involve important volumes of data conveyed through the network. The client-server model constitutes the most employed and the best established communication model over average and large sized networks such as Internet. In this model, the client sends a request to a server who process it and sends the result back to the customer. Nevertheless, facing the emergence of costly applications in terms of conveyed data, networks are more and more congested which negatively affects waiting times at customer side. Also, the development of wireless communications allowed users to connect to Internet from wireless terminals, often cell phones or laptops equipped with wireless cards which are faced to a narrow bandwidth and unreliable links. To handle these new challenges, a new approach has emerged in the last few years and came to complement the client-server mode, it is the « mobile agents » paradigm.

Agents are executable programs that move from a source site towards a destination one to execute itself. They are autonomous, mobile and potentially intelligent. Despite the lack of mobile agent-based killer applications on the market, plenty of architectures have been implemented in research laboratories. These architectures involve mobile agents collaborating to execute a certain task ordered by the user.

In this work, we present a multi-agent based application for electronic commerce. Mobile agents must look for k admissible prices, i.e, inferior to a certain threshold value. As soon as the k admissible prices are found, the search is terminated. Our objective is to find a search strategy that would give the best performance. Our performance metrics are the global search time and the generated network load. We propose, for this goal, two search strategies: the first one, called «individualistic», involves no collaboration between agents and the second one, called «collaborative», involves coordination between mobile agents. The coordination can be of two types: the first one is based on a

blackboard mechanism through which each agents share and exchange his search results with other agents. The second is based on a direct communication between the agents: every time an agent finds an admissible price, he sends a message to the other agents informing them.

Our performance study is empirical, as we cannot have a precise estimation of some parameters such as the execution times or the probability to find the sought information on a given site. We then proceed by scenarios: we consider two extreme cases, where the admissible prices are respectively located on the first sites visited by the agents and on the last visited sites. The results show that each collaboration policy can be recommended for a given need but, in any case, the collaboration gives better performances than non coordinative strategy. Generally, the blackboard mechanism is recommended when we absolutely aim to optimize the search time, without concern for the generated network load. If we want to have a «good» compromise between the search time and the network load, the inter-agents direct communication mechanism is recommended, with a small number of agents. This work prepares the way to a more efficient usage of mobile agents in a wireless context. In fact, according to the needs in quality of service, the one or the other strategy can be used, to give the better performance. From third-generation applications point of view, this study, extended to a wireless environment, could output interesting experimental results as for the use of collaborating mobile agents in information retrieval applications or in electronic commerce.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES SIGLES ET ABBRÉVIATIONS

API :	Application Interface
CORBA :	Common Object Request Broker Architecture
EDI :	Electronic Data Exchange
GH :	Grasshopper
JDK :	Java Development Kit
KIF :	Knowledge Interchange Format
KQML :	Knowledge Query and Manipulation Language
MASIF :	Mobile Agent System Interoperability Facility
m-Commerce :	Mobile Commerce
OMG :	Object Management Group
PDA :	Personal Digital Assistant
RPC :	Remote Procedure Call
SE :	Search Engine
SMA :	Système Multi-Agent
UML :	Unified Modeling Language
XML :	eXtended Marckup Language

CHAPITRE I
INTRODUCTION

La dernière décennie a connu un développement fulgurant des réseaux de données, notamment avec l'essor considérable qu'a connu Internet. Avec plus de 540 millions d'utilisateurs et environ 9 millions de sites dont 3.1 millions sont ouverts au public (soit une progression de 45% en quatre ans) (Le journal du net, 2002), Internet constitue la plus grande source d'information au monde et un moyen de communication de plus en plus incontournable. Son développement a permis l'expansion des applications multimédia et autres services à valeur ajoutée. Le commerce électronique, ou e-commerce (*electronic commerce*), fait partie de ces applications qui ont connu un essor considérable. En effet, en 2001, le chiffre d'affaires du e-commerce mondial avait augmenté de 39% par rapport à l'année 2000, s'établissant à 25,3 milliards de dollars (Le journal du net, 2002). La bande passante limitée et les délais élevés qui en résultent posent cependant de nouveaux défis pour le e-commerce. Pour les relever, un nouveau paradigme, celui des agents mobiles, a été proposé pour pallier certaines déficiences de l'architecture classique client-serveur. Dans ce mémoire, nous étudions le potentiel d'une architecture basée sur les agents mobiles pour supporter le commerce électronique avec un niveau acceptable de qualité de service. Dans ce chapitre, nous allons d'abord définir les concepts de base nécessaires à la compréhension des éléments de la problématique, puis nous préciserons nos objectifs de recherche, pour finalement esquisser le plan du mémoire.

1.1 Concepts de base et contexte de la recherche

Le *commerce électronique* englobe l'ensemble des applications et protocoles qui permettent aux utilisateurs du Web d'acheter des produits divers sur le Net. Les clients peuvent se connecter à des galeries ou magasins virtuels, naviguer à travers les différentes catégories de produits ou rechercher un produit spécifique, choisir un ou

plusieurs produits et les payer au moyen de leur carte de crédit ou d'argent virtuel dans le contexte d'une transaction sécurisée. L'essor du commerce électronique a été considérable au cours des dernières années et les prévisions sont assez prometteuses pour les années à venir. Les estimations sont de 169, 380 et 452 milliards de dollars US respectivement pour 2002, 2003 et 2004 (Le journal du net, 2002). Au Canada, on estime que les transactions d'achat en ligne vont produire 11.5 milliards $ en 2004, comparé à 1.2 milliards en 2001 (Le journal du net, 2002).

Pour rechercher un site marchand ou un produit particulier, tout comme pour rechercher une information quelconque sur le Net, l'utilisateur soumet sa requête de mots clés à un moteur de recherche spécialisé en commerce électronique ou à un *portail* de e-commerce. Un *portail* est un site qui regroupe un ensemble de renseignements divers permettant à l'utilisateur de trouver rapidement une information dans le secteur qu'il souhaite. Un portail est dit *spécialisé* lorsqu'il regroupe un ensemble d'informations abordant le même thème, comme la vente en ligne de billets d'avion ou la location de voitures. Ceux-ci fournissent au client un point d'entrée unique vers des boutiques virtuelles. Les portails classent généralement leurs sites selon les catégories des produits exposés. Comme exemple de portails de vente en ligne, on peut mentionner d'une part *travel.yahoo.com, ThomasCook.co.uk* et *ChanBrothers.com* qui sont spécialisés dans les voyages et, d'autre part, *shopping.yahoo.com* qui est un portail général de e-commerce.

Le commerce électronique, tout comme la plupart des applications sur Internet, repose sur l'architecture classique client-serveur. Le client se connecte à un site de commerce électronique (serveur), consulte les produits disponibles ou recherche un produit spécifique, et effectue éventuellement une opération d'achat. Toutes les données (pages Web, catalogues des produits, etc.) sont téléchargées sur le site du client et beaucoup de données intermédiaires sont aussi échangées entre client et serveur avant que l'information finale puisse être reçue par le client.

Par ailleurs, le développement considérable des télécommunications sans fil a rendu possible l'accès à Internet à travers des terminaux mobiles sans fil, comme les téléphones cellulaires et les PDAs (Personal Digital Assistant). La vente de ces petits

appareils sans fil a progressé, passant de 12.2 millions de PDAs vendus en 2000 à 22.4 millions prévus en 2002. Par ailleurs, plus d'un quart des sites Web américains seront accessibles à partir de terminaux mobiles avant la fin de cette année (Le journal du net, 2002). Le commerce électronique s'est donc accommodé au mode d'accès sans fil, d'où le concept de *commerce électronique mobile* ou *m-commerce* (Mobilecity, 2000). On estime que 15% des internautes dans le monde utilisent leur téléphone mobile pour accéder à des services de commerce électronique (Le journal du net, 2002).

Avant d'acheter un produit, le client a besoin de *services* comme la recherche d'un produit, du meilleur prix pour ce produit, la comparaison des rapports qualité / prix d'un certain produit parmi plusieurs fournisseurs, ou encore la participation à des ventes aux enchères. La recherche du produit spécifique à acheter, par exemple, est une tâche que beaucoup de personnes font en ligne. Un récent rapport du ministère du commerce des É-U a montré que 36% des Américains utilisent Internet pour rechercher des produits, un chiffre en hausse d'environ 10%. Tous ces services existent en commerce classique mais nécessitent en commerce électronique des considérations particulières comme la dynamique du marché et la diversité des plates-formes et des langages entre les différents sites marchands.

Le processus de recherche de produits et de prix passe par la consultation et l'interprétation du contenu de catalogues électroniques avant la prise de décisions. Pour automatiser ce processus, et pour que les documents soient compréhensibles par des machines, il faut qu'ils aient un format standard. EDI (Electronic Data Exchange) est un standard d'échange de documents qui est employé pour échanger des données en mode *Business-To-Business* (B2B). Mais la syntaxe EDI est complexe et nécessite des solutions d'intégration propriétaires chez le client et le marchand, et ce, pour chaque paire client-marchand, ce qui n'est pas pratique dans un marché ouvert comme sur Internet (Glushko et al., 1999).

Depuis quelques années, le standard XML (Sol, 1999) est venu s'imposer comme le protocole de description de données et d'échange de messages le plus approprié au Web (Glushko et al., 1999). Avec une syntaxe simple et riche, XML est un méta-langage qui

offre à l'utilisateur la possibilité de créer son propre protocole. La popularité de XML dans le commerce électronique l'a fait adopter par de grandes firmes telles *Netscape, CommerceOne, Sun* et *Microsoft* comme un standard pour la présentation des catalogues de produits dans leurs solutions commerciales de e-commerce (exemples: *CommerceXpert* de Netscape, *Net.Commerce* de IBM). Avec l'adoption de XML comme standard pour l'échange des documents et messages sur Internet, les catalogues des produits sont plus facilement interprétables par une machine. En fonction des données pertinentes sur les catalogues, comme les caractéristiques ou le prix d'un produit, une négociation automatisée peut avoir lieu entre le client et le marchand. XML se recommande pour présenter les catalogues et échanger les données entre client et marchand (Glushko et al., 1999).

Dans le même ordre d'idées, le paradigme d'*agent mobile* se prête bien à ce contexte de recherche d'informations sur le Web. Les *agents mobiles* sont des unités de code, ou des programmes exécutables, autonomes, qui se déplacent entre les machines du réseau pour y exécuter des tâches préprogrammées pour le compte de l'utilisateur qui les envoie (Falchuk et Karmouch, 1997; Gray et al., 2000). Munis éventuellement d'une certaine intelligence, les agents peuvent prendre des décisions plus ou moins complexes à la place du client, tenant compte de certains paramètres, comme le profil utilisateur et les paramètres dynamiques propres au site d'exécution.

1.2 Éléments de la problématique

Comme nous l'avons mentionné précédemment et tout comme le commerce classique, le commerce électronique comprend une étape de recherche du produit à acheter à travers plusieurs marchands. Une fois le produit trouvé, le client peut décider, selon ses critères et ses préférences, d'acheter ou d'ignorer le produit. S'il décide de l'acheter, une transaction englobant un paiement sécurisé a lieu. Dans chaque étape du processus, certains problèmes se posent, dont la sécurité de la transaction, la confiance dans les protocoles de paiement, les délais d'attente, la qualité de service, etc.

Malgré des prédictions assez optimistes (Selon une étude du cabinet Frost & Sullivan, en 2006, 15% des transactions en ligne se feraient à travers des terminaux sans fil), le m-commerce connaît plusieurs handicaps qui ralentissent son essor commercial. Les délais d'attente élevés et la bande passante limitée constituent les principaux obstacles à surmonter.

Par ailleurs, dans une architecture client-serveur, l'ensemble des interactions entre le client et le serveur peut engendrer des délais importants surtout si les sites du client et du marchand sont éloignés. Car, les données échangées entre le client et le serveur (site marchand) doivent à chaque fois passer par le réseau, souvent encombré. Au cas où une négociation aurait lieu entre le client et le marchand, le volume de ces données est encore plus grand. De plus, dans le cas d'un accès sans fil à partir d'un terminal mobile, la bande passante limitée et l'état de la connexion parfois bruitée et non fiable augmentent encore plus les délais d'attente. Les délais élevés et la piètre qualité de la connexion constituent donc autant de problèmes à résoudre en vue d'une progression commerciale du m-commerce.

Dans un autre ordre d'idées, en mode client serveur, les traitements sur les catalogues des produits (exploration des produits, des prix, des offres spéciales, etc.) se font localement sur le site du client. Ainsi, pour accéder aux catalogues, le client doit se connecter au serveur du marchand et télécharger le (ou les) catalogue(s) en entier pour en extraire les informations cherchées (caractéristiques, prix, etc.), ce qui peut induire des délais d'attente élevés. On a donc besoin de mécanismes plus souples et plus efficaces pour réduire les délais d'accès et de consultation de ces catalogues.

Le e-commerce actuel repose beaucoup sur l'intervention de l'utilisateur pour effectuer les transactions et décider des principales étapes du processus. Les actions de l'utilisateur sont basées sur des décisions visuelles successives. En visualisant un catalogue virtuel de produits, le client peut choisir le produit qui correspondrait le plus à ses besoins, à ses critères et à sa fourchette de prix « admissible ». Ces opérations nécessitent donc une intervention continue de l'utilisateur, ce qui lui coûte du temps et de l'argent, surtout quand il s'agit d'opérations telles que le lancement d'une commande

d'un stock de matériel dont les articles proviennent de différents fournisseurs et possèdent des caractéristiques différentes (fourchette de prix, délais de livraison, etc.).

D'autre part, la notion de négociation électronique n'est pas encore assez développée, faute de standards et mécanismes d'interopérabilité entre les clients et les marchands, opérant éventuellement dans des environnements hétérogènes. Beaucoup de travaux (Benyoucef et al., 2000; Collins et al., 2001) s'intéressent à développer cette notion car on pense que beaucoup de temps et d'argent peuvent être sauvés en automatisant efficacement la négociation.

Tout compte fait, les solutions client-serveur qui existent actuellement sur le Web pour rechercher des informations, telles des prix admissibles pour des produits spécifiques, ne sont pas encore assez efficaces faute d'interopérabilité des protocoles de présentation des produits et de communication. Le besoin se fait donc sentir pour des solutions qui garantissent autant la convivialité des interactions que l'efficacité des transactions.

Pour remédier à certains des problèmes cités ci-dessus, une idée a été proposée depuis quelques années, c'est de ramener l'application, en tout ou en partie, vers les données, par opposition au client-serveur où les données sont ramenées (téléchargées) vers l'application pour y être traitées. Les applets Java et les RPC (Remote Procedure Call) reposent sur ce principe. Dans ce mémoire, nous privilégions une approche par agents mobiles. Les agents peuvent automatiser certaines tâches et optimiser le temps d'attente de l'utilisateur grâce à l'exécution en mode local. Dans la recherche du meilleur prix par exemple, l'agent parcourt les sites marchands à la recherche du prix le plus bas d'un produit correspondant aux critères du client. Dans les enchères, l'agent conduit l'enchère selon une stratégie pré-programmée par le client. Il peut même suivre et diriger plusieurs enchères corrélées simultanément, l'état de l'une influençant les décisions à prendre sur l'autre (augmenter la mise et de combien, quitter l'enchère, etc.).

Cependant, le paradigme « agents mobiles » est confronté à plusieurs problèmes: l'agent qui doit s'exécuter « librement » sur une machine du réseau pose un problème de sécurité, il faut donc authentifier l'agent et s'assurer de sa bonne conduite. D'autres

problèmes de sécurité (Green et al., 1997) comme l'enlèvement d'un agent ou sa corruption par la plate-forme peuvent se poser. Par ailleurs, quand plusieurs agents sont employés pour exécuter une tâche donnée, des mécanismes de coopération et de coordination doivent être mis en place (Green et al., 1997). La coordination consiste à implanter des mécanismes de communication et d'échange d'information pertinente entre les agents pour qu'ils puissent effectuer plus efficacement leurs tâches (Kay et al., 1998) (Cabri et al., 1998). C'est là un échantillon de problèmes à prendre en compte si on veut concevoir une architecture efficace de commerce électronique basée sur les agents mobiles.

1.3 Objectifs de recherche

L'objectif principal de ce mémoire est de concevoir une architecture à plusieurs agents mobiles pour la recherche sur le réseau d'un produit ayant des critères bien déterminés (tel le prix). D'une manière plus spécifique, nous visons à :

- analyser les meilleurs mécanismes de recherche d'information identifiés dans la littérature sous l'angle de leur applicabilité au contexte du commerce électronique ;
- concevoir une architecture à base d'agents mobiles pour la recherche de produits à travers des sites marchands répartis sur le réseau ;
- proposer des algorithmes de collaboration entre les agents et les adapter au contexte de la recherche d'information en commerce électronique ;
- étudier les performances de l'architecture proposée en comparant une approche à un seul agent mobile à celle à plusieurs agents mobiles afin d'étudier l'évolutivité (scalability) de cette architecture ;
- comparer les performances des algorithmes proposés afin de dresser un schéma optimal de leur applicabilité.

1.4 Plan du mémoire

Faisant suite au chapitre I d'introduction, le chapitre II présente d'abord le commerce électronique et son architecture classique, introduit ensuite les agents mobiles, leurs

avantages et inconvénients ainsi que leur apport réel ou potentiel au commerce électronique. Le chapitre III présente une nouvelle architecture à base d'agents mobiles ainsi que les détails de sa mise en oeuvre. Il expose aussi trois algorithmes de collaboration entre agents pour la recherche de produits spécifiques sur le Net. Le chapitre IV expose les résultats de tests et d'évaluation de performance qui seront analysés par la suite. Le chapitre V, en guise de conclusion, présente une synthèse des travaux réalisés avant d'esquisser des orientations de recherche futures.

CHAPITRE II
AGENTS MOBILES POUR LE COMMERCE ELECTRONIQUE

Le commerce électronique a connu un développement important au cours de la dernière décennie. Beaucoup de travaux de recherche récents ont montré que le concept d'*agents mobiles* peut, dans certains cas, bien s'adapter au commerce électronique et permet d'automatiser voire d'optimiser certaines applications comme les enchères en ligne, la gestion des chaînes de production des entreprises, la négociation électronique, etc. Dans ce chapitre, nous faisons une synthèse sur les agents mobiles et leur application au domaine du commerce électronique. Nous présentons d'abord les agents mobiles, leurs avantages par rapport à l'architecture client-serveur et quelques uns de leurs domaines d'application, parmi lesquels le commerce électronique. Ensuite, nous examinons certaines applications spécifiques des agents mobiles au commerce électronique.

2.1 Les agents mobiles

Avec des applications Internet de plus en plus complexes et un trafic de plus en plus dense sur le réseau, l'architecture client-serveur est confrontée à des délais élevés principalement dus à une bande passante limitée, surtout en environnement sans fil. Ce défi est d'autant plus important que l'accès à Internet sans fil est en progression et que les applications supportées par les terminaux mobiles sans fil (PDAs, PocketPCs, etc.) sont de plus en plus complexes et donc exigeantes en bande passante. L'intérêt s'est donc manifesté pour une décentralisation partielle de l'exécution du code. Les applets Java et les appels de procédures distantes (RPC) font partie des solutions fonctionnelles sur Internet. Dans cette section, nous définissons un autre paradigme, celui des agents mobiles. Nous en exposons quelques avantages et inconvénients ainsi que certains de leurs domaines d'application. Enfin, nous passons en revue quelques solutions existantes à base d'agents pour le commerce électronique.

2.1.1 Concept d'agents mobiles

Un *agent* est une entité de code exécutable, capable d'exécuter des tâches pour le compte de l'utilisateur qu'elle représente. Elle est autonome, capable de prendre des décisions avec ou sans intervention externe. Un agent peut être fixe, et donc résidant sur un nœud du réseau, ou mobile (Green et al., 1997). Un *agent mobile* peut se déplacer d'une manière autonome entre les nœuds du réseau pour y exécuter des tâches pré-définies par l'utilisateur. L'agent peut choisir quand et où migrer et peut décider d'interrompre son exécution pour l'achever sur un autre nœud du réseau. Quatre éléments principaux définissent le paradigme d'agents mobiles et sont indispensables à son application (Falchuk et Karmouch, 1997) :

- l'agent lui même : c'est un programme ou un script exécutable;
- un environnement d'exécution (plate-forme): c'est une machine virtuelle qui supporte les agents mobiles, leur permet de s'exécuter et permet à l'utilisateur de les contrôler et de les manipuler;
- des ressources physiques : généralement des machines connectées au réseau, comportant CPU, mémoire, disque, services (Web, bases de données, etc.) et protocoles réseau;
- un protocole de communication inter-agents : c'est la syntaxe d'échange de messages entre agents;

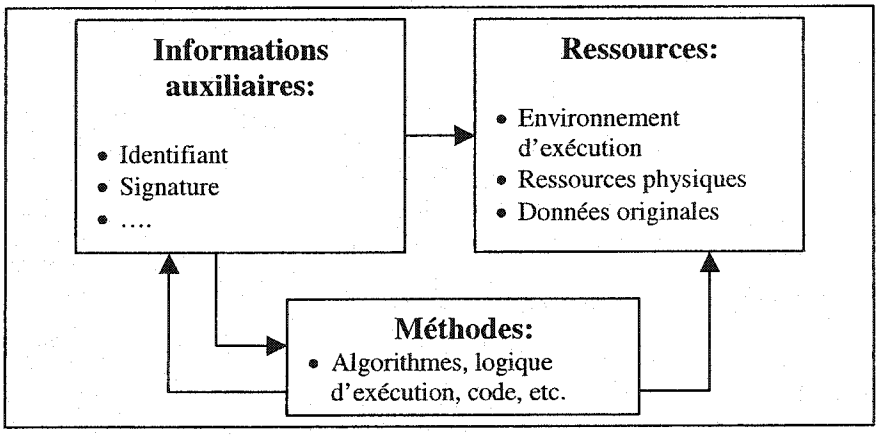


Figure 2.1 Modèle général d'un agent mobile

Les agents peuvent fonctionner en mode déconnecté : un agent peut être envoyé à partir de la machine de l'utilisateur qui peut ensuite se déconnecter. L'agent exécute alors les tâches qui lui sont confiées et lorsque l'utilisateur se reconnecte de nouveau au réseau, l'agent l'informera du résultat de sa requête. L'agent peut s'exécuter successivement sur plusieurs nœuds du réseau et transporter avec lui l'état de l'exécution et les résultats partiels.

2.1.2 Avantages et limitations des agents mobiles

Grâce à leur capacité de migrer sur le serveur des données pour s'y exécuter, les agents mobiles présentent plusieurs avantages par rapport à l'architecture client-serveur, parmi lesquels nous pouvons citer:

- Équilibre de charge : Puisque le serveur prend en charge l'exécution de la partie du processus (ou de tout le processus) rapportée par l'agent, le site du client se trouve allégé.

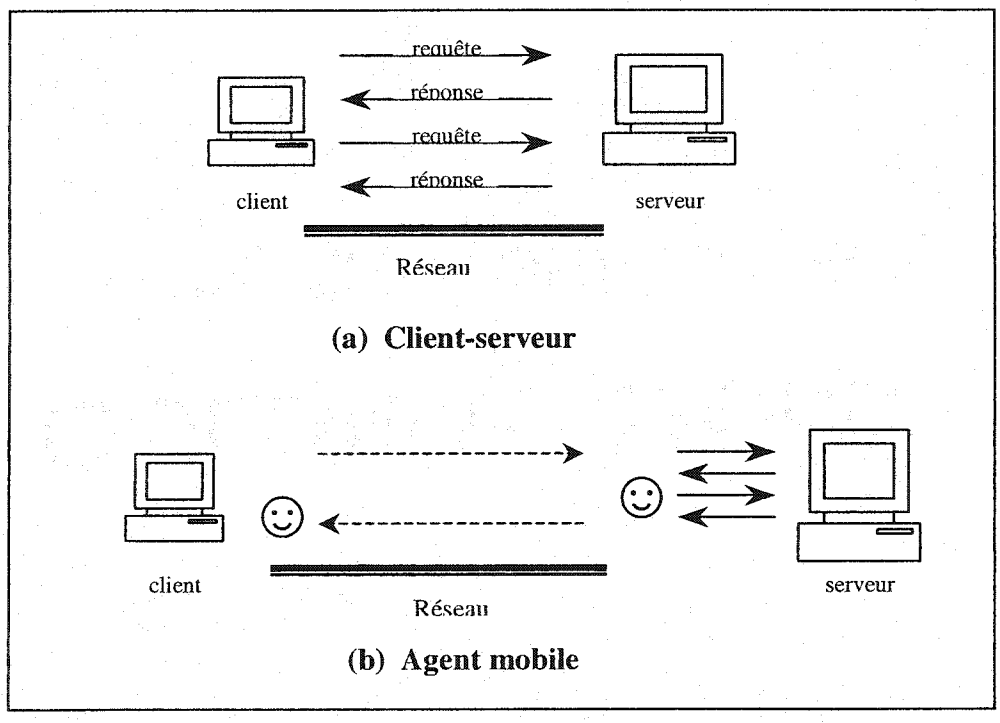


Figure 2.2 Architecture client-serveur vs architecture à base d'agents mobiles

- La bande passante est moins utilisée: En client serveur, beaucoup de données intermédiaires (comme les données de contrôle et de session du protocole TCP/IP) transitent à travers le réseau, ce qui augmente la densité du trafic. En mode agent mobile, l'agent migre une seule fois à travers le réseau et retourne avec les résultats utiles, comme le montre la Figure 2.2. Les données intermédiaires ne passent plus par le réseau mais restent locales au niveau du serveur.
- Délais d'attente moins élevés: Comme le montre la Figure 2.2, en mode agent mobile, les données intermédiaires sont échangées en moins de temps qu'en client-serveur, car elles ne passent plus par le réseau et donc le délai d'attente global est faible.
- Travail en mode déconnecté : L'agent offre à l'utilisateur qui l'envoie la possibilité de travailler en mode asynchrone. L'utilisateur peut envoyer l'agent puis se déconnecter. L'agent peut revenir à la machine qui l'a émis quand celle ci se reconnecte de nouveau au réseau. Ceci peut faire gagner du temps à l'utilisateur mais aussi de l'argent si la communication est payante.
- Personnalisation et gestion du profil utilisateur : Les agents mobiles peuvent représenter le client. Ils portent ses préférences, sa logique de prise de décision et son profil (Pierre et al.,2000).

Malgré leurs avantages prouvés et leurs capacités potentielles à pallier certains problèmes de latence et de bande passante de l'architecture client-serveur, les agents mobiles ont encore du mal à s'imposer en industrie et ce, pour plusieurs raisons parmi lesquelles nous pouvons citer:

- **Absence d'application type (*killer application*):** Beaucoup d'applications dans le cadre de travaux de recherche, ont prouvé que la performance des agents mobiles dépassait, dans certains cas, celle du client-serveur, comme dans des applications de recherche et d'extraction de fichiers multimédia (Falchuk et Karmouch, 1997), de recherche d'informations pour l'organisation électronique

de rendez-vous (Glitho et al., 2002), de gestion de réseau (Hu et al., 2002), etc. Cependant, il n'existe toujours pas d'application type qui propulserait l'utilisation des agent mobiles à l'échelle de l'industrie, et ce, entre autres, pour les deux raisons qui suivent.

- **Problèmes d'interopérabilité:** L'avantage principal dont bénéficient les agents est sans doute la mobilité. Un agent passe par plusieurs nœuds du réseau et s'y exécute. Pour ce faire, il faudrait qu'il puisse communiquer avec les plates-formes qui l'accueillent. Aussi, la communication entre agents ne peut se faire sans un standard commun de communication et de coopération. L'interopérabilité entre plate-formes et agents hétérogènes est donc une condition essentielle à la commercialisation des agents mobiles. Or, actuellement, la diversité des plates-formes, des langages et des protocoles de communication inter-agents entrave l'évolution vers une interopérabilité complète entre les environnements hétérogènes. Plusieurs efforts de standardisation ont été cependant réalisés et sont entrain d'être développés pour pallier ce problème, parmi lesquels KIF (Knowledge Interchange Format), un langage d'échange de connaissances entre agents (données, messages, requêtes, etc.) et MASIF (Mobile Agent System Interoperability Facility), un standard gérant l'interopérabilité entre les plates-formes.
- **Problèmes de sécurité :** Plusieurs problèmes de sécurité se posent quand l'agent arrive sur un hôte du réseau pour y exécuter une certaine tâche. Comme décrit dans (Qi et Yu, 2001), ces problèmes sont de deux types:
    - Agent malicieux : Il faut protéger la machine hôte ainsi que les autres agents contre les agents malicieux ;
    - Plate-forme malicieuse : Il faut protéger l'agent contre les potentielles attaques de la plate-forme et celles provenant du réseau.

  Les principaux scénarios d'attaques qui peuvent survenir sont :
    - Révélation : L'information contenue dans l'agent est espionnée et révélée à la plate-forme ou à d'autres agents ;

- Répudiation : une des parties engagées dans la communication nie avoir participé à l'opération ;
- Déni de service : Les utilisateurs nient avoir connaissance des actions exécutées par leurs agents ou les refusent ;
- Usurpation d'identité: lorsqu'un agent se présente sous une fausse identité, possiblement usurpée à un autre agent ;
- Épuisement de ressources: une ressource abuse d'une autre, faisant croire à une opération « honnête », dans le but de freiner ou d'enfreindre son fonctionnement, ce qui se répercute sur les autres agents «de bonne foi» ;
- Renvoi: une ancienne copie de l'agent est renvoyée à la place de la vraie ;
- Modification: Les données de l'agent sont modifiées ou substituées.

Malgré un nombre de travaux et de scénarios réalisés pour pallier les failles de sécurité, beaucoup de travail reste encore à faire et la sécurité demeure un point faible dans le paradigme agents mobiles.

2.1.3 Les agents mobiles comme solution à l'Internet sans fil

L'accès à Internet sans fil est de plus en plus populaire (15% des internautes dans le monde utilisent leur téléphone mobile pour accéder à des services en ligne (Le journal du net, 2002)). Le commerce mobile (*m-commerce* ou *mobile commerce*) est l'une des applications phares qui utilisent l'Internet mobile. Les réseaux hertziens vont des réseaux sans fil à faible bande passante couvrant de longues distances, comme le GSM (son taux de transfert de données est de 9.6 Kbits/s) aux réseaux à moyenne bande passante couvrant de courtes distances, comme le *WaveLAN* (2 Mbits/s). Tous les réseaux sans fil souffrent généralement d'une bande passante limitée par rapport aux réseaux filaires, ce qui produit des délais élevés quand il s'agit de transporter des applications complexes (vidéo, images, etc.) en plus des taux d'erreur, de la latence qui agissent sur la qualité de service. L'architecture client-serveur se trouve donc inappropriée à certaines applications accédant à Internet sans fil (À travers un PDA ou

un ordinateur portable par exemple) et parce qu'elle est synchrone, l'utilisateur se trouve confronté à des délais élevés en attendant le transfert d'un document volumineux (multimédia, etc.). Les agents mobiles peuvent constituer ici une alternative intéressante et plusieurs travaux (Perret et Duda, 1996 ; Delord et al., 1997) ont prouvé qu'elle est plus performante dans certains cas.

Perret et Duda (1996) présentent l'architecture MAP qui est basée sur des *agents assistants* pour la recherche d'informations sur le Web. Dans MAP, le client lance une requête de recherche d'information à partir d'un terminal mobile. Un agent *assistant* est alors créé sur un nœud distant et activé par du code qu'il interprétera. Un agent *collecteur de résultats* est alors créé sur un nœud *résultat*. Un agent *interpréteur* interprétera le message de l'*assistant* et, en cas de panne d'un nœud, l'assistant peut migrer vers d'autres nœuds distants et cloner d'autres agents assistants pour terminer le travail dont l'état au moment de l'interruption a été stocké dans une *mémoire persistante* (*persistant storage*).

Delord et al. (1997) présentent *Alycta*, une architecture basée sur MAP, et qui améliore les performances d'accès au Web à travers des téléphones cellulaires GSM. *Alycta* met en jeu des agents mobiles qui recherchent des documents sur le Web, dégradent leur qualité jusqu'à un seuil « acceptable » et les renvoient au téléphone cellulaire du client. La recherche et le filtrage des documents se font en mode déconnecté. L'utilisation des agents mobiles améliore le temps de réponse et limite le temps de connexion du client.

2.1.4 Architectures multi-agents mobiles

Les systèmes multi-agents mobiles (SMA) sont des architectures où plusieurs agents collaborent pour exécuter une tâche commune. Les SMA constituent des solutions intéressantes pour des problèmes décentralisés où les sources de données sont réparties. Par l'exécution des processus en parallèle, les SMA peuvent diminuer les délais d'exécution, améliorer l'évolutivité et la fiabilité par rapport à une architecture à un seul agent. Une architecture multi-agents mobiles implique une communication et une

coordination entre les agents. Selon le type de relation entre les agents, on distingue deux types de SMA (Green et al., 1997):

- SMA réparti coopératif : les agents coopèrent pour réaliser une tâche commune. Le but de chacun des agents est d'améliorer la performance globale du système ou d'aboutir au même but commun. Comme exemple, nous mentionnons un SMA où les agents cherchent une information sur le réseau ou cherchent le meilleur prix d'un produit spécifique.
- SMA compétitif (*self-interested*): chaque agent essaie de réaliser une tâche propre à lui, indépendamment des autres agents. Les agents sont en compétition et chacun doit réaliser une meilleure performance que les autres, souvent à leurs dépens. Par exemple, dans un système où des agents négocient un contrat ou essaient de trouver la meilleure plage horaire pour un rendez vous, chaque agent essaie de réaliser son but aux dépens des autres.

2.1.5 Communication et coordination entre agents

Dans une architecture multi-agents mobiles, les agents peuvent communiquer et coopérer entre eux. La coopération a lieu lorsqu'un agent est confronté à une alternative de choix pouvant affecter son environnement (les autres agents de son domaine). Plusieurs des protocoles de négociation et de coopération entre les agents proposés dans la littérature se basent sur les réseaux de contrats (*Contract net). Contract net* (Sandholm, 1993) est le premier protocole pour l'allocation des tâches et des ressources entre entités informatiques (Sandholm, 1993).

Par ailleurs, des efforts ont été réalisés pour standardiser la communication entre les plates-formes (Perdikeas et al., 1999) et ont donné naissance à FIPA (Foundation for Intelligent Physical Agents, *www.fipa.org*) qui est une fondation établie depuis 1996 pour promouvoir et éditer les spécifications d'interopérabilité entre les systèmes d'agents intelligents dans un contexte industriel. Le standard MASIF (Mobile Agent System Interoperability Facility), agréé par l'OMG (Object Management Group, *www.omg.org*) a également été créé pour gérer l'interopérabilité entre les plates-formes.

KIF (Knowledge Interchange Format, conforme à ANSI, *http://Logic.Stanford.EDU/kif/*) est un langage spécifiant le format d'échange des connaissances. Sa sémantique spécifie les règles de définition des objets, des règles de logique, des fonctions et des relations. En ce qui concerne la communication inter-agents, KQML (Knowledge Query and Manipulation Language, *http://www.cs.umbc.edu/kqml/*) est considéré comme le standard le plus répandu. KQML est un protocole spécifiant les formats de messages entre systèmes intelligents en général et agents en particulier, et des actions de prise en charge de ces messages (*message-handilng*).

COOL (*COOrdination Language*) (Barbuceanu et Fox, 1996) est un exemple de systèmes à base d'agents mobiles pour la gestion électronique de la chaîne d'approvisionnement d'une entreprise. COOL établit un système de communication entre les agents basé sur les formats de message KQML; il permet au développeur de spécifier son vocabulaire et ses types d'action.

2.1.6 Plates-formes d'agents mobiles

Pour s'exécuter, communiquer et migrer, les agents ont besoin d'un *environnement d'exécution* qu'offrent les *plates-formes* d'agents mobiles. Une plate-forme d'agents mobiles est une machine virtuelle qui supporte les agents et permet à l'utilisateur de les programmer et de les manipuler. La plate-forme offre aux agents des services comme:

- La mobilité: Les agents peuvent migrer entre des nœuds du réseau qui sont munis de plates-formes d'agents mobiles ;
- La communication: Les plate-formes permettent la communication inter-agents par échanges de messages, et à un niveau plus bas, entre les agents et la plate-forme ;
- L'auto-exécution: la plate-forme permet à l'agent de s'auto-exécuter en interfacant avec le système d'exploitation ;
- La nomination et la localisation: la plate-forme permet de localiser tous les agents qu'elle émet. Elle attribue à chaque agent un identifiant unique qui permet de l'invoquer ;

- La sécurité : l'exécution des agents est assujettie à des règles de sécurité afin de protéger les ressources des sites visités des agents malicieux, de protéger les agents de leurs homologues et des plates-formes.

Il existe sur le marché plusieurs plates-formes. Certaines sont multi-langages (C, C++, Tcl, etc.) comme *D'agent* et *Ara*, mais les plus populaires implémentent des librairies (API) en java qui permettent de les manipuler en ayant simplement une machine virtuelle Java installée. Parmi celles ci, on peut citer:

- ***Grasshopper*** (*www.grasshopper.de*): plate-forme basée sur des APIs en Java, qui permettent au programmeur de faire migrer, communiquer et exécuter des agents d'une manière transparente. Grasshopper est conforme à la première version de MASIF, standard établi par l'OMG pour faciliter l'interopérabilité entre les différentes plate-formes d'agents mobiles. Grasshopper supporte plusieurs mécanismes de migration et de communication entre agents, comme RMI (Remote Method Invocation), CORBA (Common Object Request Broker Architecture) et IIOP (Internet Inter ORB Protocol).
- ***Aglet*** (http://www.trl.ibm.com/aglets/index_e.htm) : Aglet est une plate-forme d'agents actifs en java spécifié par IBM en 1997 (IBM Tokyo Research Lab). Il se compose d'un environnement de développement (l'Aglets Software Development Kit 1.0.3). Un Aglet est un agent mobile qui peut voyager d'un serveur à un autre via le réseau, en exécutant des opérations arbitraires dans les limites de sécurité établies par chaque serveur. Un "serveur d'Aglets" est défini par l'Aglet *deamon*. C'est la couche logicielle au-dessus de la JVM qui assure l'exécution des Aglets, leur transfert d'une machine à une autre, et des contrôles de sécurité. Il fournit aussi des outils graphiques d'administration. Cette plate-forme fournit les services de base d'un Système à base d'agents mobiles tels que l'envoi de messages synchrones et asynchrones. Elle permet aussi la gestion d'événements permettant de réagir au changement d'état d'un Aglet. Un des intérêts majeurs des plates-formes multi-agents est d'assurer la transparence du mécanisme de transport des agents. La plate-forme Aglet est organisée en couches de façon à cacher au maximum ces mécanismes via l'Aglet *deamon* et l'Aglet *context*

(contexte d'exécution d'un Aglet). Cela permet aussi d'assurer un niveau de sécurité non négligeable dans un environnement ou l'accès aux ressources locales est critique (via l'Aglet *proxy*). Avec Java comme support d'exécution, les problèmes de déploiement sur des plates-formes hétérogènes sont réduits. Bien que *Aglet* soit utilisé dans l'industrie, cette architecture reste pour IBM un environnement de recherche et de développement sur les agents mobiles.

- Grasshopper et Aglet sont des exemples de plate-formes, il en existe d'autres (que nous ne jugeons pas utile de détailler), comme *D'Agents* du *Dartmouth College* (http://agent.cs.dartmouth.edu/), *Odyssey* de *General Magic* (www.generalmagic.com) et *Voyager* de *ObjectSpace* (www.objectspace.com).

2.1.7 Domaines d'application des agents mobiles

Bien que le paradigme d'agents mobiles souffre de l'inexistence d'application type (*killer application*), plusieurs applications intéressantes ont vu le jour depuis quelques années et ont ranimé l'espoir de voir les agents mobiles sortir des laboratoires de recherche pour s'établir en industrie. Parmi les domaines d'application des agents mobiles, on peut citer :

- Les télécommunications: Plusieurs travaux se sont intéressés à utiliser les agents pour les télécommunications sans fil. Par exemple, dans (Delord et al., 1997), un agent est utilisé pour chercher et télécharger sur un téléphone cellulaire GSM des documents volumineux à partir d'Internet. Un agent est lancé à partir du téléphone cellulaire pour chercher un document (une image) sur le Net. Ensuite, l'image est traitée en résolution de façon à ce que sa taille engendre un délai de transport acceptable. L'agent mobile transmet enfin le document à un agent fixe sur l'appareil cellulaire. L'étude a prouvé que l'agent pouvait contribuer à diminuer les temps de latence.
- La gestion de réseau : La gestion du réseau comprend la surveillance, la mise à jour et la maintenance du réseau et de ses ressources afin de détecter et corriger les dysfonctionnements. L'utilisation des agents peut ajouter de la flexibilité, décentraliser l'exécution et réduire la charge réseau. MAGENTA (Van Thanh, 2001) est un système

de gestion de réseau basé sur les agents mobiles. Des mesures réalisées avec un *Gestionnaire de Réseaux Mobiles* ont prouvé que les agents étaient plus performants que le client-serveur du point de vue de l'utilisation de la bande passante et du temps total d'exécution (Van Thanh, 2001).

- La recherche d'information: C'est un domaine où l'utilisation des agents mobiles est très prometteuse. Parmi les travaux réalisés dans ce sens, Glitho et al. (2002) mettent en place une architecture à base d'agents mobiles pour la planification de rendez-vous ou de réunion. Un agent se déplace sur un serveur de calendriers et cherche une date à laquelle les participants convoqués à une réunion sont disponibles. L'agent revient ensuite avec l'information sur la date de la réunion. L'utilisation des agents pour la recherche d'information sur Internet à partir d'un terminal sans fil s'avère être, dans certains cas, plus performante que le client-serveur classique. Jain et Anjum (2000) montrent que, dans certaines conditions, les délais de recherche engendrés par la stratégie agents mobiles sont nettement inférieurs à ceux du client-serveur.
- Le commerce électronique: Les agents mobiles dans le commerce électronique interviennent dans la recherche d'un produit spécifique sur Internet, la recherche d'un meilleur prix pour un produit parmi plusieurs marchands, la gestion automatisée des commandes en B2B (*business to business*), la participation à des enchères en ligne ou encore la négociation électronique de contrats. Dans la section suivante, nous passons en revue différents travaux impliquant l'utilisation des agents mobiles dans le commerce électronique en général et la négociation en particulier.

2.2 Les agents mobiles dans le commerce électronique

Les agents sont bien adaptés au commerce électronique, ils peuvent être intégrés dans une architecture de e-commerce pour faire la recherche d'articles à travers des sites marchands. Ils peuvent aussi négocier des contrats et prendre des décisions pour le compte du client qui les envoie.

Parmi les avantages que présentent les agents, citons:

- **Gestion du profil utilisateur**: l'agent peut transporter le profil de l'utilisateur qui le lance, c'est-à-dire, dans le cas du commerce électronique, ses préférences pour un produit particulier, sa stratégie de négociation, son prix seuil, etc.
- **Prise de décisions :** L'agent peut décider à la place de l'utilisateur des actions à entreprendre en fonction de paramètres pré-programmés.

Cependant, une architecture client-serveur peut aussi nous procurer ces avantages, moyennant une application sur le poste du client qui joue le rôle d'un agent fixe. L'envoi d'un agent mobile doit en effet être motivé par d'autres critères parmi lesquels :

- La qualité de la liaison entre la machine cliente et le reste du réseau. En effet, si la liaison n'est pas suffisamment fiable ou si elle a un taux d'erreur élevé, comme dans le cas d'une liaison sans fil, la migration de l'agent peut être efficace à cause des risques d'interruption de la liaison (et donc de la session) et de la perte des informations intermédiaires dans le cas d'une liaison client-serveur. Ce cas s'identifie bien au m-commerce où l'accès aux sites marchands se fait à partir d'un poste mobile sans fil.
- La complexité des actions à exécuter localement sur les machines distantes. Dans le commerce électronique, la recherche d'informations et la négociation électronique constituent des tâches plus ou moins complexes où l'utilisation d'agents mobiles peut s'avérer intéressante.

2.2.1 Concepts de base du commerce électronique

Le *commerce électronique* ou *e-commerce* englobe l'ensemble des opérations, processus, terminaux informatiques, réseaux de communication, applications et protocoles qui permettent aux utilisateurs du Web d'acheter des produits sur le Net. Les clients peuvent se connecter à des boutiques ou magasins virtuels, naviguer à travers les différentes catégories de produits ou rechercher un produit spécifique, choisir un ou plusieurs produits et les payer au moyen de leur carte de crédit ou d'argent virtuel dans le contexte d'une transaction sécurisée. Du côté client, le e-commerce couvre plusieurs volets allant de la recherche d'un produit spécifique sur un éventail de marchands

virtuels sur le Net, à la commande et au paiement sécurisé de l'article acheté. Du côté marchand, il va de la présentation des produits sous forme de catalogues virtuels (pages Web, documents texte, etc.) au processus de paiement en passant par la gestion de la clientèle et la mise à jour de la base de données des produits.

Parmi les applications spécifiques du commerce électronique, nous pouvons citer: la banque à distance *(e-banking)*, la vidéo sur demande, la chaîne d'approvisionnement automatique des entreprises *(chain supply)*, le marketing et la publicité en ligne, la vente/achat et la livraison, les ventes aux enchères, etc.

Le commerce électronique comprend principalement quatre volets :

1) Attraction du client : publicité, marketing, etc. ;
2) Interaction avec les utilisateurs : catalogues, moyens pratiques de recherche dans ces catalogues, négociation de contrats, etc. ;
3) Gestion des commandes : réception des commandes, paiement, gestion des transactions, livraison des produits achetés ;
4) Assurer les rétroactions aux clients : fidélisation, service après vente, historiques des transactions des clients.

Un client qui cherche un produit spécifique à acheter sur le Net est généralement confronté à un problème d'abondance et/ou de repère : soit il ne connaît pas l'adresse du marchand qui aurait l'article cherché, soit il y a trop de marchands et il doit les visiter un à un pour trouver son article, ce qui peut lui prendre beaucoup de temps tout en n'ayant aucune garantie d'avoir la meilleure offre. Sur Internet, certains sites comme BizRate.com et Price.com proposent de rechercher le meilleur prix d'un produit à travers plusieurs boutiques virtuelles offrant ce produit. D'autres, comme pricingcentral.com, font une comparaison entre plusieurs marchands des caractéristiques d'un produit. Ces sites utilisent des robots (*bots*), qui sont essentiellement des moteurs de recherche collectant des prix de produits de la part de plusieurs vendeurs.

D'autre part, si le client veut négocier le prix d'un produit, comme c'est généralement le cas dans les enchères *(auctions)*, il doit rester connecté au site qui héberge l'enchère

ou au marchand avec qui il négocie. De plus, quand il s'agit d'une négociation multi-parties, avec plusieurs marchands simultanément, la tâche devient plus compliquée si les décisions à prendre pour les différentes négociations sont corrélées. Ici aussi, il existe sur Internet des sites qui proposent de participer à une enchère à la place du client qui doit leur fournir son prix maximal qu'il peut payer et sa marge d'augmentation du prix.

En passant en revue les principaux serveurs de commerce électronique disponibles sur le marché[1], nous pouvons dégager une architecture générale commune à ces serveurs. Cette architecture est illustrée à la Figure 2.3 et nous pouvons y distinguer trois parties principales :

- Une partie temps réel d'«avant boutique» (*front-office*) : Généralement composée d'une partie « application » et d'une partie « données », l'application a une composante Web qui interagit avec les clients en temps réel et leur offre plusieurs interfaces de communication, pour visualiser les produits, rechercher des articles, effectuer des transactions, payer les commandes, etc. Elle a aussi une autre composante *métier*[2] qui comporte la logique fonctionnelle du site (stratégie commerciale, présentation des catalogues, gestion des profils clients et des transactions temps réel, etc.). L'application est généralement placée sur un serveur d'application public et séparé des données pour des fins de sécurité, de décentralisation, d'équilibre de charge et de relève en cas de panne. Un moteur de recherche interne permet aux clients de rechercher des produits sur le site. Il est mis à jour régulièrement grâce à des robots (*bots* ou *crawlers* en anglais).

---

[1] Nous avons passé en revue trois serveurs de commerce électronique parmi les plus connus sur le marché : *Net.Commerce Start* de Lotus, *Commerce Server* de Microsoft et *WebSphere Commerce* de IBM.

[2] C'est, dans le cas du e-commerce, la stratégie commerciale : politique de vente, d'application des prix et des rabais, négociation, marketing, sollicitation des clients, etc.

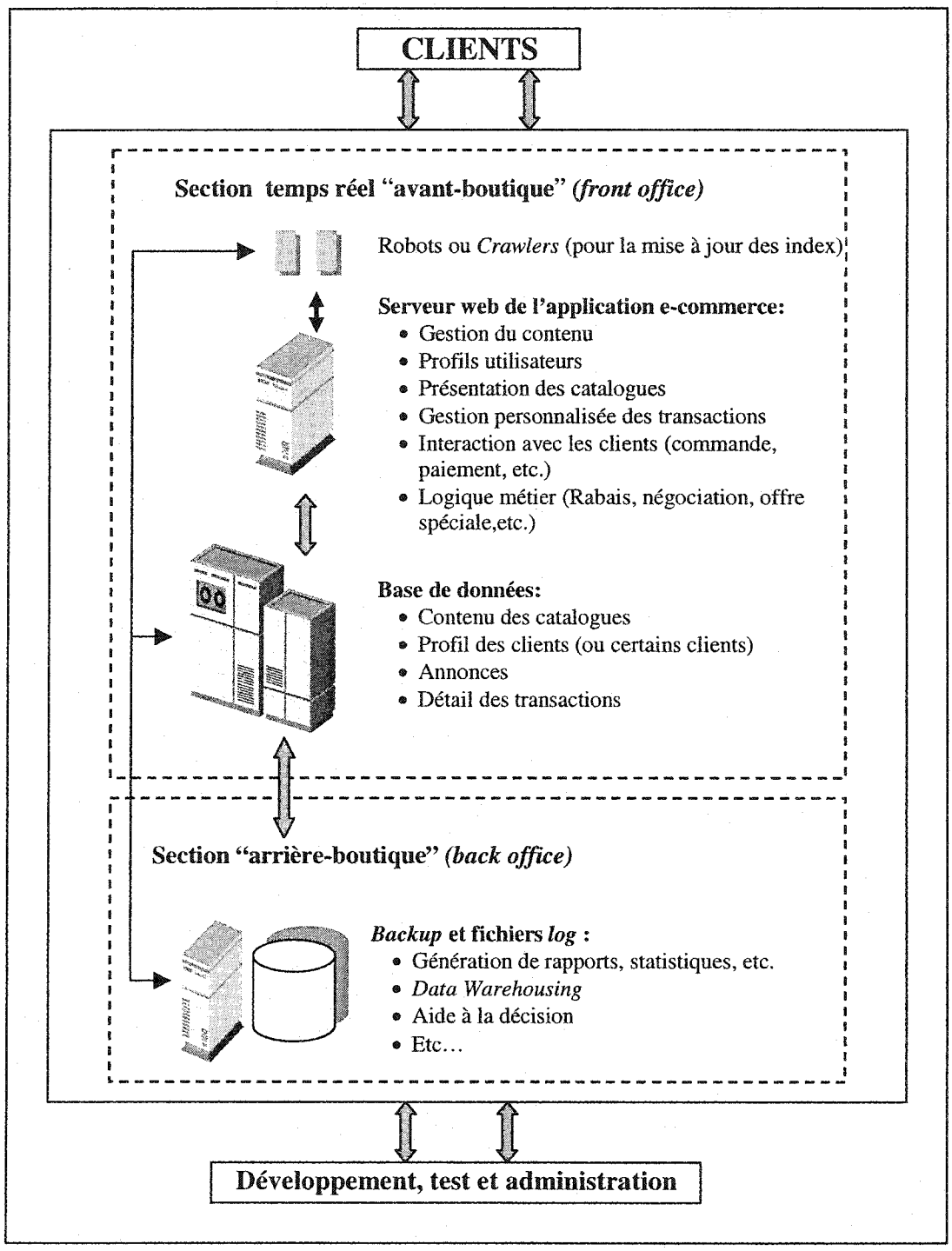


Figure 2.3 Architecture générale d'un serveur de commerce électronique

- Une partie « arrière-boutique » (*back office)* : qui contient les données du site stockées dans des bases de données, à partir desquelles sont générés les rapports, les statistiques et sont prises les décisions (*Data warehousing*).

Ces deux parties sont reliées aux modules d'administration qui assurent la mise à jour, et aux modules de développement et de test.

2.2.2 Les agents mobiles dans le e-commerce

Les agents peuvent jouer un rôle de médiateurs dans le commerce électronique. Ils peuvent intervenir sur plusieurs niveaux entre le client et le marchand, en automatisant certaines tâches (comparaison de prix entre plusieurs marchands, chercher un produit correspondant aux critères du client, etc.), en prenant des décisions plus ou moins complexes pour le compte du client ou en lui facilitant la prise de décision (tri ou filtrage des données avant leur présentation). Dans tous les cas, c'est le client qui prend la décision finale ou qui délègue à l'agent de décider à sa place. Par exemple, imaginons qu'un client qui veut faire un appel de longue distance et qui a le choix entre plusieurs opérateurs téléphoniques; un agent peut intervenir, avant de faire l'appel, pour chercher l'opérateur qui offre, au moment de l'appel, le tarif le moins cher. Les agents peuvent donc jouer le rôle de médiateurs intelligents dont le degré d'autonomie et d'intelligence est spécifié par le client.

Le e-commerce, s'inspirant du commerce classique, comporte six étapes, comme décrit par le modèle du « comportement d'achat du consommateur » ou CBB (*Consumer buying behavior*) (Guttman et Maes, 1998). Ces étapes sont :

1) **Identification du besoin:** le client doit spécifier le produit qu'il veut acheter et sous quels critères;
2) **Filtrage et/ou tri des produits:** selon les besoins spécifiques du client et son évaluation des produits par rapport à ses besoins. Cette étape aboutit à un ensemble admissible de produits;

3) **Filtrage des marchands/fournisseurs:** choix d'un ensemble de marchands qui répondent le mieux aux besoins du client par rapport à l'ensemble admissible de produits. Cette étape aboutit à un ensemble admissible de marchands ;
4) **Négociation:** le client peut négocier les termes d'une transaction avec le(s) marchands(s) ;
5) **Achat et livraison:** un contrat d'achat est conclu et le client devrait recevoir la marchandise en bonne et due forme ;
6) **Service après-vente:** le marchand doit s'assurer de la satisfaction du client et faire le suivi nécessaire pour améliorer le service et mieux s'adapter aux besoins du marché.

En commerce électronique, les agents peuvent intervenir au niveau des étapes 2, 3 et 4. Par exemple, ils peuvent être incorporés dans plusieurs volets du e-commerce comme l'approvisionnement automatisé des entreprises (prise de contact avec les fournisseurs, recherche de la meilleure offre, commande, négociation et paiement), le commerce multimédia (achat et vente de fichiers de chansons, films, etc.), le commerce dit « domestique » ou le B2C (*Business to customer*), la notification (notification du client de la parution d'un nouvel article, d'une offre spéciale, etc.), les services de médiation (trouver des informations à propos d'un produit, chercher le meilleur prix pour un article, négocier un prix, participer à des ventes aux enchères, etc.). Nous donnons dans la suite quelques exemples d'applications de commerce électronique à base d'agents.

Les agents comme médiateurs dans le B2C (Product brokering)

Grâce à leur aspect personnalisé et autonome (ou semi-autonome), les agents peuvent jouer le rôle de médiateurs (*brokers*) dans le commerce électronique. Après que le client eut identifié ses besoins et les eut formulés électroniquement (par une interface utilisateur, un fichier texte structuré, etc.), un agent peut rechercher et identifier les produits qui correspondent aux critères exigés par le client et qui concordent avec son profil. Ensuite, il constitue à partir de catalogues électroniques (*e-catalogs*) distribués une liste de produits admissibles, c'est à dire conformes aux critères du client, qu'il

présente à ce dernier. ***PersonalLogic*** (www.personallogic.com) est un logiciel à base d'agents qui permet de filtrer l'ensemble des produits admissibles selon le profil du client et ses critères de préférence. Les clients peuvent alors naviguer dans un « espace de produits » et imposer leurs contraintes sur ces produits, ce qui limitera leur espace admissible. ***FireFly*** (www.firefly.com) est une autre solution à base d'agents qui aide le client à prendre ses décisions en lui recommandant des produits spécifiques selon un classement et une cotation des marchands. Les agents interviennent aussi dans la recherche et le choix des marchands. L'agent peut parcourir une liste d'adresses de sites marchands, les interroge et sélectionne le marchand proposant la meilleure offre. Plusieurs travaux ont été réalisés dans ce sens: ***BargainFinder*** (Guttman et Maes, 1998) est un projet pilote de *PriceWaterHouse & Coopers* qui fut le premier où un agent mobile est employé pour chercher le meilleur prix d'un produit spécifique. L'agent parcourt neuf sites de vente de disques compacts de musique, et retourne au site du client avec le meilleur prix. *BargainFinder* ne prend en considération que le facteur prix, il ne permet pas la négociation du prix et utilise un seul agent mobile. ***Jango*** (Guttman et Maes, 1998) de *NetBot* est une version avancée de *BargainFinder*. *Jango* tient compte des critères de chaque client pour le produit cherché, ce qui permet au client de choisir le produit en fonction du rapport qualité/prix, plutôt que par le prix absolu. *Jango* suppose que les produits des marchands sont exposés dans des catalogues publics, mais ne permet pas de négociation avec le marchand. *Jango* repose sur des « adaptateurs d'information » intelligents qui identifient et retrouvent l'information que demande le client à partir de pages HTML et de scripts CGI dont les formats changent fréquemment. ***KASBAH*** (Chavez et Maes, 1996) est un système où plusieurs agents interrogent les marchands, négocient avec eux leurs offres et décident de les accepter ou de les refuser (*KASBAH* sera revu dans le paragraphe « négociation »).

Les agents mobiles dans les ventes aux enchères (auctions)

Les enchères sur Internet se sont développées rapidement lors des dernières années. Selon une étude de *Forrester Research,* la fréquentation des salle d'enchères a augmenté

de 15% au premier trimestre 2001 enregistrant un chiffre d'affaires de 6.5 Milliards USD en 2000. *Yahoo!auction*, *eBay*, et *UBid* font partie des salles d'enchères les plus populaires sur Internet.

Dans une enchère, plusieurs participants soumettent des offres de prix pour acheter un produit qui les intéresse parmi ceux exposés sur le site de l'enchère, et ce, dans une période de temps limitée. À la fin de cette période, c'est le participant qui a soumis la meilleure offre qui gagne l'enchère et achète le produit. Il existe plusieurs types d'enchères selon le prix initial (le plus bas ou le plus haut) et le prix retenu à la fin de l'enchère (le meilleur prix ou le second meilleur prix). Les plus connues sont: l'enchère anglaise, hollandaise et Vickrey (Antony et al., 2001). Pour mener une enchère, chaque participant utilise une tactique propre à lui, qui dépend de ses paramètres de décision comme le prix maximum qu'il peut payer, le pas d'augmentation du prix proposé à chaque étape, etc. Pour automatiser les enchères, certains sites proposent aux clients de conduire leurs enchères pour eux, grâce à des robots (*bots*) comme SpotBot (www.spotbot.com). Les clients doivent alors fournir le prix maximal qu'ils peuvent payer, et le robot peut conduire plusieurs enchères en parallèle. Cependant, un problème de confidentialité peut se poser, car le robot, connaissant le prix maximal qu'un client peut payer, il tricher contre lui en lui faisant payer ce prix maximum. Les agents peuvent constituer une solution à ce problème. Un agent peut conduire plusieurs enchères simultanément pour le compte du client, sans révéler sa tactique, en l'occurrence son prix maximal. Il connaît aussi les préférences du client, donc il peut décider quelle enchère privilégier au cas où il doit faire ce choix. Antony et al. (2001) présentent un agent qui peut participer à des enchères pour le compte du client. L'agent est muni d'un algorithme à plusieurs tactiques pour mener les enchères simultanées, et de certains paramètres comme le prix maximal. D'autres serveurs d'enchères à base d'agents mobiles ont été réalisés dans un cadre académique ou de recherche, comme *Internet AuctionBot* (Guttman et Maes, 1998) de l'université du Michigan, qui permet au développeur, grâce à une interface d'API, de créer ses propres agents qui iront sur le serveur d'enchères retrouver des produits et participer aux enchères.

Dans *MAGICS* (**M**obile **A**G**e**nt-based **I**nternet **C**ommerce **S**ystem) (Chan et al., 2001), le client peut envoyer un ou plusieurs agents sur des sites d'enchères à partir de son appareil WAP, et contrôler leurs activités en tout temps. De plus, un serveur *MAGICS* est mis à la disposition des clients pour leur permettre de créer et de gérer leurs agents à distance sans avoir à installer de logiciels particuliers sur leurs terminaux (en l'occurrence une plate-forme d'agents mobiles).

Le *Biddingbot* (Ito et al., 2000) est un système multi-agents qui permet aux utilisateurs de participer à des enchères multiples, de les gérer et de les surveiller, en introduisant et en modifiant ses offres. Les agents surveillent simultanément des prix de l'article dans plusieurs enchères. Le système se compose d'un agent chef et de plusieurs agents soumissionnaires, qui sont assignés aux différents sites d'enchère. Les agents coopèrent : ils recueillent l'information, la surveillent, et font des offres simultanément dans les sites d'enchère. L'agent chef facilite la coopération des agents soumissionnaires et organise leur couplage (agit comme un matchmaker), envoie les requêtes de l'utilisateur aux agents soumissionnaires, en récupère l'information et la présente à l'utilisateur.

Par ailleurs, l'utilisation des agents pourrait être très prometteuse pour l'accès sans fil aux salles d'enchères. En effet, comme les enchères peuvent durer plusieurs jours, il est irréaliste que le client reste constamment connecté pour suivre en temps réel l'évolution des offres. L'agent peut alors faire ce suivi et aviser son client quand ce dernier se connecte, ou quand il y a une évolution importante du cours de l'enchère.

Les agents mobiles dans la chaîne d'approvisionnement des entreprises

Une chaîne d'approvisionnement (*supply chain*) est un ensemble d'entités commerciales autonomes ou semi-autonomes distribuées (généralement appelées entreprises virtuelles), responsables de la fabrication et la distribution de produits et services relatifs à une famille de produits. Chaque entité dans la chaîne d'approvisionnement a un objectif précis et est assujettie à un ensemble de contraintes (Swaminathan et al., 1998). Pour produire un bien ou un service quelconque, une

entreprise a besoin, entre autres, de produits et/ou services de base qu'elle se procure chez ses fournisseurs. Il a été prouvé que l'automatisation de la chaîne de production fait épargner beaucoup d'argent aux grandes entreprises dont une partie des dépenses est destinée à l'approvisionnement (Bichler et al., 1998) (Swaminathan et al., 1998). Ceci a suscité beaucoup d'intérêt quant à l'automatisation et l'optimisation des chaînes de production, notamment dans les entreprises virtuelles où l'approvisionnement en biens repose sur le commerce électronique, et se base sur la commande, l'achat et le paiement des produits et services sur Internet.

Beaucoup de travaux ont été réalisés offrant une solution répartie pour la gestion décentralisée de chaînes de production (Swaminathan et al., 1998). Grâce à leur autonomie, les agents peuvent contribuer à plus d'efficacité dans l'organisation des chaînes d'approvisionnement. Les agents peuvent avoir des tâches très spécifiques dans lesquelles ils doivent prendre des décisions, gérer la production et collaborer avec les autres acteurs (les autres agents). Un agent peut représenter un fabricant, un revendeur, un fournisseur de services, un intermédiaire, un client, un courtier ou n'importe quel acteur de la chaîne. Dans (Brugali et al., 1998), une architecture à base d'agents statiques a été conçue pour gérer une chaîne de production dans le domaine du textile. Les agents sont coopérants et communiquent à travers des messages et des agents mobiles *Aglets* (d'IBM) porteurs de messages. L'architecture repose sur plusieurs niveaux :

- Un niveau de *Gestion globale de la chaîne de Production* : avec des agents statiques responsables de la gestion des produits et de l'intégration des fonctions commerciales de l'entreprise (gestion des commandes, stocks, etc.) dans le système de planification des ressources. C'est un module qui gère aussi les agents provenant des unités de manufactures (usines, ateliers, etc.).
- Un niveau *Ateliers (usines)*: formé d'agents hiérarchiquement organisés pour gérer la fabrication.

- Un niveau *utilisateur*: basé sur des agents *Interface* accessibles à l'aide d'un navigateur Web, et qui facilitent à l'utilisateur l'accès aux services et moyens de contrôle de la chaîne.

MAGNET (Dasgupta et al., 1999) est un autre exemple de travaux intéressants qui utilisent les agents mobiles pour la gestion des chaînes de production des entreprises. L'architecture de MAGNET est formée par des agents *fournisseurs* statiques interagissant avec des agents clients (ou acheteurs) mobiles qui se déplacent sur les sites marchands. Les agents sont des *Aglets* (IBM) et les produits à acheter sont décrits en XML. MAGNET gère plusieurs niveaux de sous-fournisseurs et permet à un marchand qui doit livrer une matière première à plusieurs revendeurs, de savoir si les commandes des revendeurs est destinée à satisfaire la commande du même client ou pas.

Les agents mobiles dans la négociation électronique

La négociation électronique a été définie par Bussmann et Muller (1992) comme: « un processus de communication entre un groupe d'agents afin de réaliser un accord mutuel acceptable à propos d'un sujet spécifique». Cet accord est généralement un compromis dans lequel les parties négociatrices font des concessions acceptables. Le sujet de la négociation peut être le prix d'un bien, l'horaire d'un rendez-vous, etc. Une négociation est donc une interaction entre deux entités, basée sur des *protocoles* et des *stratégies,* que chacune des entités utilise pour augmenter son *utilité*. Un protocole est un ensemble de règles de jeu (actions valides), et pour un protocole donné, une ou plusieurs stratégies (plans d'action) peuvent être employées. L'utilité est une fonction d'appréciation associée par chaque agent au produit qui est sujet de la négociation. Elle exprime le degré de satisfaction d'une entité négociatrice par l'offre proposée et dépend directement des paramètres du contrat négocié (Zlotkin et Rosenschtein, 1989). Des négociations sont dites *combinées* si elles ont lieu simultanément, sur des serveurs entre une entité A et deux autres entités B et C situées sur des serveurs différents, à propos de produits complémentaires et non dissociables pour composer une offre globale. On distingue deux types de négociation:

- Négociation compétitive: Les agents négociateurs ont des intérêts opposés et essaient de se mettre d'accord sur un choix qui serait accepté par tous (Zlotkin et Rosenschtein, 1989). Chaque agent suit un ensemble de mécanismes de négociation, ou stratégie, qui dépend de ses intérêts, ses préférences et ses besoins. Comme exemple, on peut mentionner un agent client et un agent marchand qui négocient le prix d'un produit (Chavez et Maes, 1996).
- Négociation coopérative: Les agents négociateurs ont un but global commun et doivent collaborer pour le réaliser. Des mécanismes individuels et globaux de négociation sont mis en jeu. Comme exemple, citons des agents qui négocient pour organiser une réunion pour plusieurs employés d'une compagnie, les agents représentent les employés, connaissent leurs disponibilités et leurs préférences, et doivent se mettre d'accord sur une date qui convient à tous les employés convoqués.

En négociation, les agents doivent prendre des décisions comme « augmenter le prix d'un pas donné jusqu'à un certain seuil », « accepter les offres dont le prix est inférieur à une certaine valeur », etc. Mais généralement, les décisions prises sont plus complexes et dépendent de l'environnement (état du marché, offres disponibles, nombre de participants, etc.) et de plusieurs paramètres dans le contrat négocié.

Plusieurs travaux se sont intéressés aux architectures à base d'agents mobiles pour le commerce électronique, parmi lesquels nous pouvons citer KASBAH, tête-à-tête, MagNet, MAGMA et CONCENSUS.

KASBAH (Chavez et Maes, 1996) : C'est un système multi-agents qui a été élaboré dans les laboratoires de MIT *(Massachusetts Institute of technology). KASBAH* offre une place de marché virtuelle contenant des annonces classées *(Classified Ads)* dans laquelle des agents « clients » négocient le prix d'un produit avec des agents « vendeurs ». L'interaction entre les deux agents négociants est binaire: l'agent client fait une offre à l'agent marchand, ce dernier répond par « oui » ou « non ». Dans cette architecture, La rencontre entre agents, la négociation et la conclusion du contrat est centralisée sur le site web *KASBAH*. La négociation dans *KASBAH* se fait selon un algorithme d'augmentation et de baisse du prix (selon des fonctions linéaires, quadratiques ou

cubiques). *KASBAH* a l'inconvénient de regrouper tous les agents (acheteurs et vendeurs) sur un même site (la place de marché). Une panne sur ce site ou une charge élevée pourrait corrompre le fonctionnement du système. Aussi, chaque acheteur et vendeur doit transmettre ses offres et ses demandes à ce site, ce qui n'est pas très pratique vu la forte dynamique de tels marchés (les produits et les prix changent très souvent).

Tete-a-Tete (Guttman et Maes, 1998) : Également élaboré dans les laboratoires de MIT, ce système implique plusieurs agents qui font la négociation d'une transaction dans les ventes en détail. Celle ci tient compte de plusieurs facteurs tels que le prix, les délais de livraison, la garantie, etc. Les marchands font à leurs clients une offre complète (spécifications du produit, prix, garantie, date de livraison, etc.) qui sera évaluée par les clients. Ces derniers renvoient au marchand une « critique » sur un ou plusieurs paramètres de l'offre, laquelle critique sera prise en compte par le marchand pour modifier son offre initiale et mieux satisfaire le client, ou refuser la transaction. Les propositions, critiques et contre-propositions sont formulées selon un format XML (Sol, 99). Les produits sont exposés, tout comme dans *KASBAH*, dans une place de marché commune *(Blackboard)* dans laquelle interagissent agents acheteurs et agents vendeurs.

MAgNet (Dasgupta et al., 1999) : *Mobile Agent for Networked Electronic Tranding* est un système développé à l'Université de Californie. Dans *MagNet*, un agent acheteur-négociateur visite des sites marchands, recherche un produit spécifique dans des catalogues décrit en XML et négocie une potentielle transaction avec un agent vendeur stationnaire sur le site marchand. *Magnet* permet de traiter plusieurs niveaux de fournisseurs. Il permet par exemple aux agents vendeurs de s'adresser à leur tour aux sites marchands de leurs fournisseurs à chaque fois que besoin est (si par exemple, l'agent acheteur demande une quantité supérieure à celle qui est en stock chez le marchand). *Magnet* utilise un agent qui parcourt une liste de sites marchands et qui retourne avec la meilleure transaction. Une fois qu'un accord est trouvé après

négociation, l'agent procède à la réservation et le paiement du produit est autorisé par le client.

MAGMA (Tsvetovatyy et al., 1997) : *Minnesota Agent Marketplace Architecture* est un prototype de marché virtuel. Il met en jeu plusieurs agents « commerçants » qui font de la vente, l'achat et la négociation de transactions. Dans la négociation, les agents vendeurs essaient d'obtenir le prix le plus haut pour leurs produits, et les agents acheteurs essaient d'obtenir le prix le plus bas possible. La négociation est basée sur la forme simple d'un mécanisme appelé « mécanisme de Vickrey ». La version complète de MAGMA couvre les mécanismes de paiements à travers un organisme financier qui valide les transactions.

CONSENSUS (Benyoucef et al., 2002) : C'est un système automatisé de négociation électronique au moyen d'agents mobiles. Dans ce système, l'utilisateur peut instancier un ou plusieurs agents, leur donne une stratégie de négociation ainsi qu'un mécanisme de coordination, les enregistre sur les serveurs de négociation appropriés et enfin les lance. Les agents mèneront la négociation avec leurs homologues selon les mécanismes et protocoles établis.

KASBAH et *Tete-a-tete* offrent la recherche du produit selon plusieurs caractéristiques ainsi que la négociation des transactions mais centralisent la place de marché. MAGMA simule un marché virtuel assez complet mais la négociation est, tout comme dans KASBAH, centralisée et basée sur le seul facteur « prix » alors que, dans la réalité, plusieurs facteurs peuvent entrer en jeu comme les délais de livraison et la quantité commandée. CONCENCUS est différent de ces deux systèmes par le fait que ses stratégies de négociation sont dynamiques; elles ne sont pas figées et peuvent changer en cours d'exécution et ce, soit par intervention de l'utilisateur, soit par réaction automatique à certains facteurs du marché.

2.2.3 XML et l'interopérabilité entre plate-formes dans le commerce électronique

Un des grands défis à relever dans le commerce électronique en général et la médiation électronique en particulier reste l'interopérabilité dans la communication entre l'agent médiateur et le site marchand (Bichler et al., 1998). Le standard EDI (Electronic Data Exchange) est utilisé en B2B pour la communication électronique entre organismes qui, généralement, ont déjà fait affaire ensemble. En effet, la syntaxe EDI est complexe et nécessite des solutions d'intégration propriétaires chez le client et le marchand, et ce, pour chaque paire client-marchand, ce qui n'est pas pratique dans un marché ouvert comme c'est le cas sur Internet (Glushko et al., 1999). Certains travaux s'orientent vers des standards plus ouverts sur le Web, comme XML et CORBA. Bichler et al. (1998) proposent le prototype *OFFER*, une architecture de médiation (*brokering framework*) développée en Java et basée sur le standard CORBA. Dans *OFFER*, des applets Java « clientes » communiquent avec le programme médiateur (*e-broker*) avec IIOP[3] (Internet Inter-ORB Protocol) et les catalogues électroniques sont implémentés sous forme de serveurs CORBA. XML est un méta-langage de description de données qui permet de définir un langage moyennant une grammaire propre spécifiée dans un DTD (*Document Type Definition*). Tout comme HTML, XML est basé sur des balises qui délimitent les propriétés de l'objet représenté. En commerce électronique, XML est utilisé pour représenter les catalogues des produits et pour faire communiquer et inter-opérer les différents acteurs du marché (vendeur, client, fournisseur, médiateur, etc.). L'interprétation d'un document XML est facilitée par des APIs existantes sur le marché, appelées *parseurs*. Ces derniers peuvent analyser, interpréter et valider un document XML. Comme exemples de parseurs sur le marché, nous pouvons citer *Xerces* d'Apache (http://www.alphaworks.ibm.com), *AlphaWorks* d'IBM (http://www.alphaworks.ibm.com), etc.

Notons au passage que plusieurs essais de standardisation des spécifications de commerce électronique en XML sont réalisés, le plus important a donné naissance à

[3] IIOP est un protocole de communication sur Internet entre plusieurs programmes distribués hétérogènes. IIOP fait partie du standard CORBA.

ebXML (electronic business eXtensible Markup langage) conjointement réalisé par UN/CEFACT[4] et OASIS[5]. ebXML (www.ebxml.org) est une infrastructure basée sur trois principes: 1) Gérer l'interopérabilité dans la communication des données ; 2) Fournir une sémantique assurant l'interopérabilité logique entre partenaires commerciaux ; 3) Fournir un mécanisme complet permettant de gérer une transaction commerciale entière. ebXML se compose de cinq couches logicielles: 1) Les composants du noyau ; 2) Les composants logique métier ; 3) Les profils des partenaires commerciaux et les contrats (ou accords) signés ; 4) Une base de données formant les registres ; 5) Un service de messages normalisés.

Tout compte fait, grâce à sa grammaire simple, sa forte adaptabilité au web et son adoption par la plupart des navigateurs web du marché, XML est de plus en plus utilisé pour la description des données (tels les catalogues électroniques) et la communication entre entités collaboratrices (telles un agent marchand et un agent client). Il offre ainsi une solution efficace pour résoudre certains problèmes d'interopérabilité des plates-formes et surtout l'interopérabilité des langages en commerce électronique.

[4] United Nation Center for Trade Facilitation and Electronic Business Information Standards
[5] Organization for the Advancement of Information Structure Standards

CHAPITRE III
ARCHITECTURE MULTI-AGENTS MOBILES POUR LE COMMERCE ELECTRONIQUE

Dans le chapitre précédent, nous avons présenté le commerce électronique et son architecture classique client-serveur. Nous avons ensuite introduit les agents mobiles et décrit leurs avantages par rapport à l'architecture client-serveur et nous avons présenté quelques domaines d'application. Le commerce électronique fait partie des domaines les plus prometteurs où les agents mobiles peuvent apporter une importante valeur ajoutée. L'agent peut alors rechercher un produit spécifique pour le compte du client en parcourant un certain nombre de sites marchands. Une architecture à base de plusieurs agents mobiles pourrait améliorer les performances de l'opération de recherche de prix mais nécessiterait des mécanismes de communication inter-agents. Dans le présent chapitre, nous concevons une architecture où plusieurs agents collaborent pour rechercher un produit sur un ensemble de marchands virtuels. Nous mettons en place des mécanismes de communication et nous faisons une étude de leur performance.

3.1 Position du problème et scénario général

Avant d'acheter un produit sur le net, le client est généralement intéressé à chercher un marchand proposant un « bon » prix pour ce produit, ou encore le meilleur prix sur le marché. Et bien que ce soit un élément de décision important, le prix n'est pas toujours le seul critère de choix pour le client, qui est généralement intéressé par un bon rapport qualité/prix. La détermination de ce rapport dépend fortement des critères propres de chaque agent (profil) et du degré de satisfaction de l'agent par rapport au produit en question, ce qui implique une intelligence plus ou moins complexe au niveau de l'agent. Pour simplifier le problème, nous allons considérer seulement le critère prix et supposer que le client veuille simplement chercher soit le meilleur prix, soit k « bons » prix pour un produit commercialisé par un certain nombre de marchands virtuels (*e-shops*). Un (ou

des) agent(s) est (sont) alors envoyé(s) à partir de la machine du client et vont parcourir les sites marchands pour effectuer la recherche. Dans cette section, nous étudions les performances de certaines solutions multi-agents mobiles. Nous nous intéressons particulièrement au temps total de recherche et à la charge réseau moyenne induite par chaque solution. Deux types de problèmes peuvent se présenter:

1. le client cherche le meilleur prix.
2. le client cherche ***k*** prix (p_1,..., p_k) inférieurs à un certain prix seuil p_s. Ces prix sont appelés ***prix admissibles***.

Comme l'illustre la Figure 3.1, le client possède au départ une liste connue de N marchands formant le *domaine*, qu'il devra parcourir et où chercher des produits à prix admissibles. m agents sont alors envoyés sur le réseau pour rechercher le meilleur prix ou un/des prix admissible(s) (selon le besoin du client). Le cas $m = 1$ est un cas particulier de solution ***mono-agent***, où un seul agent mobile doit parcourir la totalité des N sites. Dans le cas général où $m \neq 1$, chaque agent A_i parcourt un sous-ensemble de N_i sites $D_i = \{s_1,...,s_{n_i}\}$ du domaine, appelé ***sous-domaine*** de l'agent. Ce sous-domaine est affecté à chaque agent par un module répartiteur (*Dispatcher*).

Problème des k meilleurs prix

Dans ce problème, le client veut chercher les ***k*** meilleurs prix pour un produit, sur un ensemble donné de sites marchands. L'agent (ou les agents) envoyé(s) doit (doivent) parcourir tous les sites du domaine D. À la fin du parcours, il(s) doit(vent) retourner à la machine source avec les ***k*** meilleurs prix trouvés. Le cas particulier k = 1 donnera le meilleur prix pour le produit en question.

Problème des k prix admissibles

Ici, le client cherche ***k*** prix admissibles, c'est à dire inférieurs à un certain seuil P_s. Les m agents envoyés sur le réseau recherchent des prix dans leurs sous-domaines

respectifs jusqu'à accumuler, ensemble, k prix admissibles. Ils retournent ensuite sur la machine

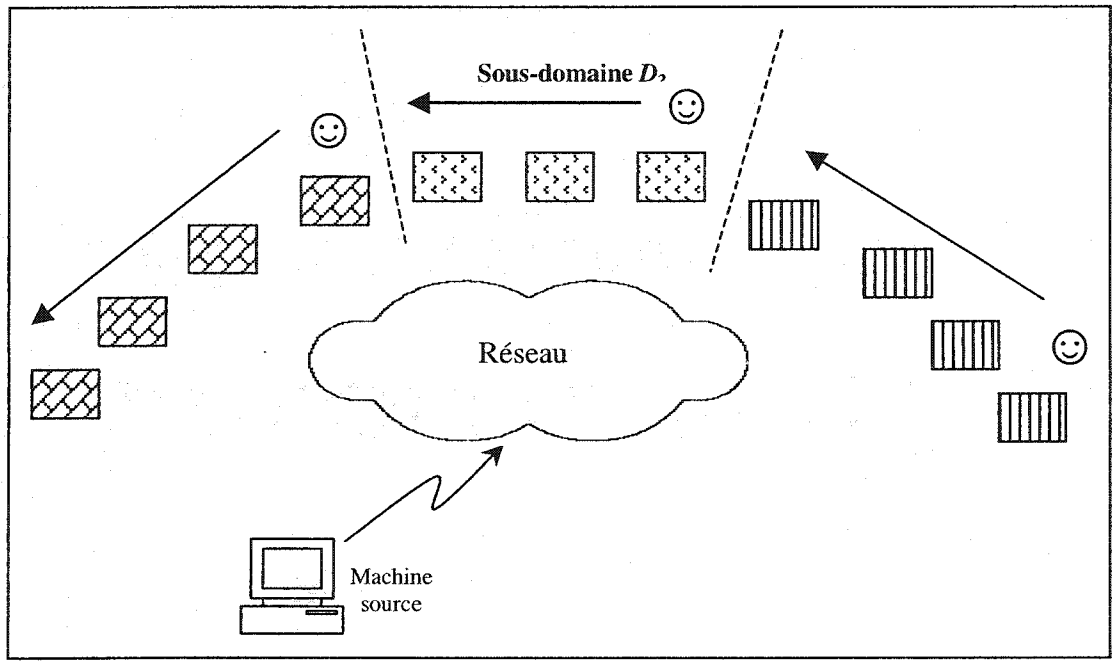


Figure 3.1 Recherche avec plusieurs agents

source avec ces prix. La tâche de chaque agent est interrompue dès que k prix admissibles sont trouvés **collectivement** par l'ensemble des agents, c'est à dire quand :

$$cardinal\left(\bigcup_{i=1}^{m} R_i\right) \geq k$$

m étant le nombre d'agents impliqués et R_i l'ensemble des résultats (prix) admissibles trouvés par l'agent A_i à un instant donné.

Le problème des k prix admissibles trouve son application en pratique, par exemple dans les appels d'offres. Une entreprise lance un appel d'offre pour l'acquisition d'un produit spécifique pour lequel elle a un budget maximal. La commission chargée de l'appel d'offre cherche alors des marchands proposant le produit en question à un prix inférieur ou égal au budget qu'elle est prête à dépenser. Comme elle ne peut pas consulter TOUS les vendeurs du marché et qu'elle doit avoir plus qu'une offre (pour des fins d'équité), elle se fixe un nombre k d'offres admissibles qu'elle étudiera ensuite pour

en choisir la meilleure. Dans le commerce classique, ce sont les marchands qui s'adressent au client en répondant à son appel d'offre. Ici, nous proposons une solution basée sur les agents mobiles, où c'est le client qui va consulter les offres des marchands sur leurs catalogues publics.

Pour faire l'étude d'une solution multi-agents, nous allons considérer un scénario pratique de recherche de prix : un client A veut acheter un (ou plusieurs) ordinateur(s). Il veut visiter N sites marchands qui vendent ces produits. Les sites sont répartis sur le réseau (Internet) et sont publiquement accessibles aux agents. Chaque marchand décrit les caractéristiques de ses produits en XML et les expose sur sa galerie électronique. Les informations relatives à un ordinateur sont : la marque, la vitesse, la capacité du disque, la mémoire vive, l'écran, le prix, etc. L'utilisation de XML normalise la représentation de ces données et facilite leur interprétation par la machine, ou par l'agent (nous reviendrons dans le prochain chapitre sur ce standard de représentation des données). Sont aussi indiquées les informations relatives au revendeur : son nom et son adresse. Le client veut se procurer un ordinateur ayant certaines caractéristiques (par exemple : Un Pentium III à 1000 MHz, avec 20 Go de disque dur, 512 Ko de mémoire vive, un écran de 19'' et dont le prix ne dépasse pas 1200 $). L'utilisateur introduit alors ses critères de recherche à l'aide d'une interface graphique et lance le programme. Un (ou plusieurs) agent(s) est (sont) alors lancé(s) sur le réseau pour chercher un ordinateur ayant les critères indiqués. Il(s) parcoure(ent) des sites marchands et visitent leurs catalogues publics. Si un article correspondant aux critères du client est trouvé, les agents doivent être mis au courant pour qu'ils interrompent leurs parcours et revenir au poste du client. Une commande pourrait être lancée dans une seconde étape, mais nous nous limiterons à l'étape de recherche de prix.

3.2 Solutions et algorithmes proposés

Dans cette section, nous allons proposer des algorithmes régissant le comportement des agents. Après avoir exposé nos solutions, nous ferons une critique et une

comparaison des différents algorithmes. Dans toutes les solutions que nous proposons dans la suite, nous adoptons la terminologie suivante:

T : temps total moyen de recherche. C'est le temps moyen au bout duquel toute l'opération de recherche de prix est terminée.

l : temps de latence. C'est le temps de migration d'un agent d'un site A vers un site B. Nous supposons que ce temps est le même entre toute paire de sites.

tt : temps de traitement. C'est le temps que passe l'agent sur un site donné. Il comprend le temps de traitement de l'agent, le temps qu'il passe dans la file d'attente et le temps d'exécution de l'agent sur le site (principalement recherche et filtrage des prix). En pratique, ce temps dépend de plusieurs facteurs comme la puissance du serveur, l'espace mémoire, la complexité de la tâche à effectuer sur le site, le trafic réseau, le degré de saturation des files d'attente, etc. Cependant, et pour simplifier nos équations, nous supposons que ce temps est le même pour tous les agents et pour tous les sites.

m : nombre d'agents impliqués dans la recherche. Chaque agent est envoyé vers un sous-domaine qui lui est affecté par un module répartiteur sur la machine source.

N_i : nombre de sites marchands dans le sous-domaine de l'agent A_i.

N : nombre total de sites marchands, $N = \sum_i N_i$.

p_a : taille d'un agent en kilo-octets (ko). Nous supposerons que tous les agents ont la même taille.

p_m : taille en kilo-octets (ko) d'un message envoyé par un agent vers un autre agent ou vers la machine source. Les messages échangés contiendront essentiellement l'information sur les prix trouvés. Nous supposons que les messages ont tous la même taille.

t_m : temps d'envoi d'un message d'un agent vers un autre agent ou d'un agent vers une plate-forme (vers un tableau noir, que nous verrons dans la suite). Il comprend le temps de traitement du message + temps de latence.

t_{mm} : temps d'envoi d'un message *multicast* d'un agent vers tous les autres ou d'une plate-forme (par exemple du *blackboard*, que nous verrons dans la suite) vers les agents. Il comprend le temps de traitement du message + le temps de latence.

Par ailleurs, les agents vont être équitablement distribués sur le domaine. Chaque agent aura $N_i = E(N/m) + i$ sites dans son sous-domaine (i = 0 ou 1 selon le reste de la division euclidienne de N par m). Par exemple, si nous avons 10 sites qui vont être parcourus par 3 agents, 2 agents auront 3 sites dans leurs sous-domaines et le 3ème agent devra parcourir 4 sites.

3.2.1 Problème des k meilleurs prix

Dans ce problème, le(s) agent(s) impliqué(s) doit(vent) parcourir tous les sites de leurs sous-domaines respectifs pour s'assurer d'avoir les k meilleurs prix sur la totalité des sites du domaine. Ce cas est illustré à la Figure 3.2.

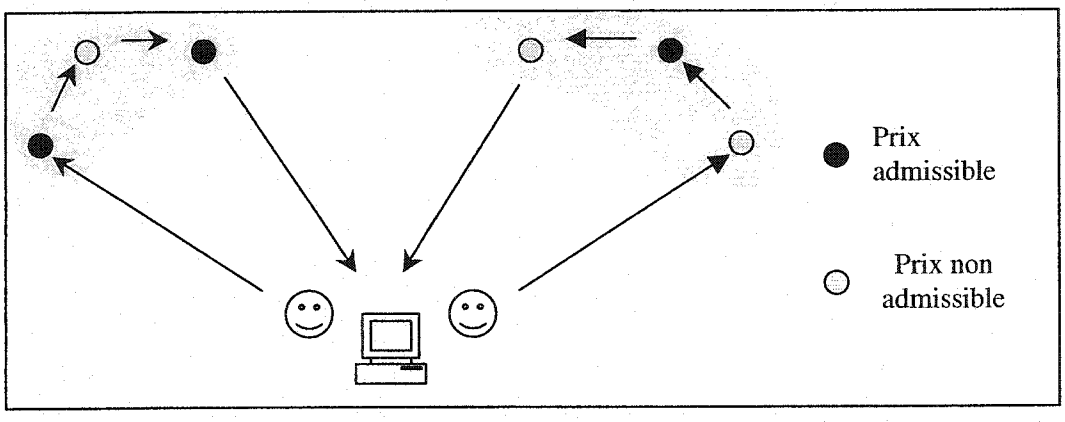


Figure 3.2 Recherche du meilleur prix : parcours des sous-domaines

L'algorithme de recherche du meilleur prix (cas particulier où k=1) pour chaque agent est présenté à la Figure 3.3.

```
ALGORITHME MEILLEUR PRIX
DEBUT
    prix_minimal = très_grand_nombre ;
    Tant que (Di n'est pas vide) FAIRE
        S = premier élément sur la liste Di ;
        Migrer vers S ;
        Chercher (produit) ;
        SI ((produit trouvé) ET (Prix (produit)) < prix_minimal )) ALORS
        prix_minimal = Prix (produit) ;
        Di = Di – {S} ;
    Fin Tant que
    Migrer (adresse_source) ;
    Livrer (prix_minimal) ;
FIN
```

Figure 3.3 Algorithme de recherché du meilleur prix

Pour le cas général (k≠1), au lieu de la variable *prix_minimal*, il faut utiliser une liste triée de *k* éléments, *liste_meilleurs_prix,* qui contiendra les *k* meilleurs prix.

À la fin de la recherche, chaque agent va livrer le prix minimal sur son sous-domaine. Le prix minimal sur le domaine global sera le minimum des prix minimaux sur les sous-domaines. En supposant que les temps de latence entre toute paire de sites sur le sous-domaine sont égaux, le temps de recherche d'un agent A_i sur son sous-domaine D_i sera:

$$T_i = (N_i + 1).l + N_i.t_t \qquad (3.1)$$

Les agents faisant leur recherche en parallèle, le temps global de recherche est :

$$T = \underset{i}{Max}\ T_i \quad (i = 1..m)$$

La charge réseau induite par cette solution est :

$$C = \sum_{i=1}^{m} (N_i + 1).p_a \qquad (3.2)$$

3.2.2 Problème des k prix admissibles

Dans ce type de problème, un client possède une liste de N marchands proposant différents prix pour un certain produit. Le client cherche k prix admissibles, c'est à dire inférieurs à un certain prix seuil P_s. m agents sont alors envoyés à partir de la machine

du client pour aller visiter les sites marchands, rechercher le produit et, s'il est trouvé, en consulter le prix. Chaque agent visite l'ensemble de sites marchands formant son sous-domaine. Le but est de récolter k prix admissibles sur le domaine entier, c'est à dire en fusionnant les résultats de recherche des m agents sur leurs sous-domaines respectifs. Si, au cours de la recherche, les k prix sont trouvés, alors les agents peuvent interrompre leur parcours et revenir à la machine source, puisque leur but global est atteint.

Pour cela, nous proposons deux types de solution: l'un, que nous appelons « individualiste », est basé sur des agents non collaborant. Le second, que nous appelons « collaboratif » est basé sur la collaboration entre agents. Dans la seconde solution, nous envisageons deux moyens de collaboration: une collaboration centralisée dans la station source autour d'un tableau noir (*blackboard*) et une collaboration décentralisée basée sur une communication inter-agents.

Solution « individualiste » (Sans collaboration)

Chaque agent effectue la recherche de prix dans son sous-domaine indépendamment des résultats des autres agents. Aucune interaction ou communication n'existe entre les agents. La fusion des résultats des différentes recherches se fait quand tous les agents auront retournés sur la machine source. Chaque agent, n'ayant aucune connaissance des résultats des autres agents, doit s'assurer d'avoir à lui seul les k prix admissibles et ne peut interrompre son parcours et rentrer à la machine source que s'il trouve ces k prix. Dans le cas contraire, il doit terminer le parcours de son sous-domaine et retourner à la source avec les $(k - j)$ prix admissibles qu'il aura trouvés $(0 \leq j \leq k)$. À la fin de la recherche, quand tous les agents seront rentrés à la source, leurs résultats seront fusionnés et les meilleurs k prix admissibles seront sélectionnés. L'algorithme général appliqué par chaque agent est décrit à la Figure 3.4.

```
ALGORITHME INDIVIDUALISTE

Liste Liste_prix_admissibles ;

DEBUT
    Tant que ((Di n'est pas vide) ET (Liste_prix_admissibles.taille < k))
    FAIRE
        S = premier élément sur la liste Di ;
        Migrer vers S ;
        Chercher (produit) ;
        SI ( (produit trouvé) ET (Prix (produit) ≤ prix_seuil )) ALORS ajouter
        Prix(produit) à Liste_prix_admissibles ;
        Di = Di – {S} ;
    Fin Tant que
    Migrer (adresse_source) ;
    Livrer (Liste_prix_admissibles) ;
FIN
```

Figure 3.4 Algorithme individualiste

D'après cet algorithme, chaque agent doit visiter les sites de son sous-domaine D_i jusqu'à accumuler k prix admissibles dans sa liste de prix, ou achever la visite de tous les sites. Soit r_i le rang du dernier site visité par un agent A_i $(k \leq r_i \leq N_i)$.

Le temps moyen de recherche pour un agent A_i est donc:

$$T_i = (r_i + 1).l + r_i.t_t \tag{3.3}$$

Et comme $k \leq r_i \leq N_i$, nous déduisons les bornes inférieure et supérieure pour T_i :

$$(k+1).l + k.t_i \leq T \leq (N_i + 1).l + N_i.t_i$$

Le temps total de recherche est conditionné par le dernier agent qui revient à la machine source, car il faut que tous les agents soient retournés à la source pour pouvoir procéder au filtrage des k meilleurs prix:

$$T = \underset{i}{Max}\ T_i \quad (i = 1..m)$$

En supposant que tous les agents aient la même taille, la charge réseau induite par cette solution est :

$$C = \sum_{i=1}^{m} (r_i + 1).\ p_a \tag{3.4}$$

Et : $$\sum_{i=1}^{m}(k+1).p_a \leq C \leq \sum_{i=1}^{m}(N_i+1).p_a$$

Ce qui est équivalent à :

$$m.(k+1).p_a \leq C \leq (m+N).p_a$$

Solutions « collaboratives »

Ici, les agents vont collaborer pour trouver ensemble les k prix admissibles. Les agents ne vont plus baser leurs décisions sur leurs seuls résultats, mais aussi sur les résultats de recherche des autres agents. À tout instant, chaque agent aura connaissance du résultat global de la recherche des prix, c'est à dire les agents ont accumulé, collectivement, les k prix admissibles. Pour cela, il faut des moyens de communication pour informer chaque agent du résultat global. La solution collaborative pourrait optimiser le temps de recherche moyen puisque, contrairement à la solution « individualiste », les agents peuvent interrompre leurs parcours avant d'avoir accumulé individuellement k prix admissibles, s'ils peuvent être informés que le nombre de prix qu'ils ont trouvés complémente ceux des autres agents et que, collectivement, les agents ont trouvés les k prix admissibles. En contrepartie, la charge réseau induite sera plus importante à cause des messages mis en jeu. Une discussion plus élaborée des algorithmes sera abordée plus loin. Nous proposons dans cette section deux méthodes de collaboration : la supervision par tableau noir *(blackboard)* et la communication inter-agents.

Collaboration par tableau noir (blackboard)

Un *blackboard* est un espace d'information commun partagé par tous les agents. Ces derniers peuvent le consulter et y ajouter des données à l'intention des autres agents. Le *blackboard* constitue donc un système médiateur d'échange d'information entre agents. Dans notre cas, c'est la machine source qui offrira ce service. Chaque fois qu'un agent trouve un prix admissible sur un site, il informe le *blackboard*, en y inscrivant le prix trouvé et l'adresse du site sur lequel il l'a trouvé. Les autres agents peuvent savoir, à tout moment, combien de prix ont été trouvés en consultant le *blackboard*. Pour optimiser les

envois de messages dans notre architecture, c'est le *blackboard* qui avertira les agents dès que k prix ont été enregistrés chez lui. L'information sera diffusée par l'envoi d'un message *multicast* à tous les agents. Ces derniers suspendront alors leurs parcours respectifs et retourneront à la machine source. La Figure 3.5 illustre la collaboration par *blackboard*, et la Figure 3.6 décrit son algorithme de fonctionnement.

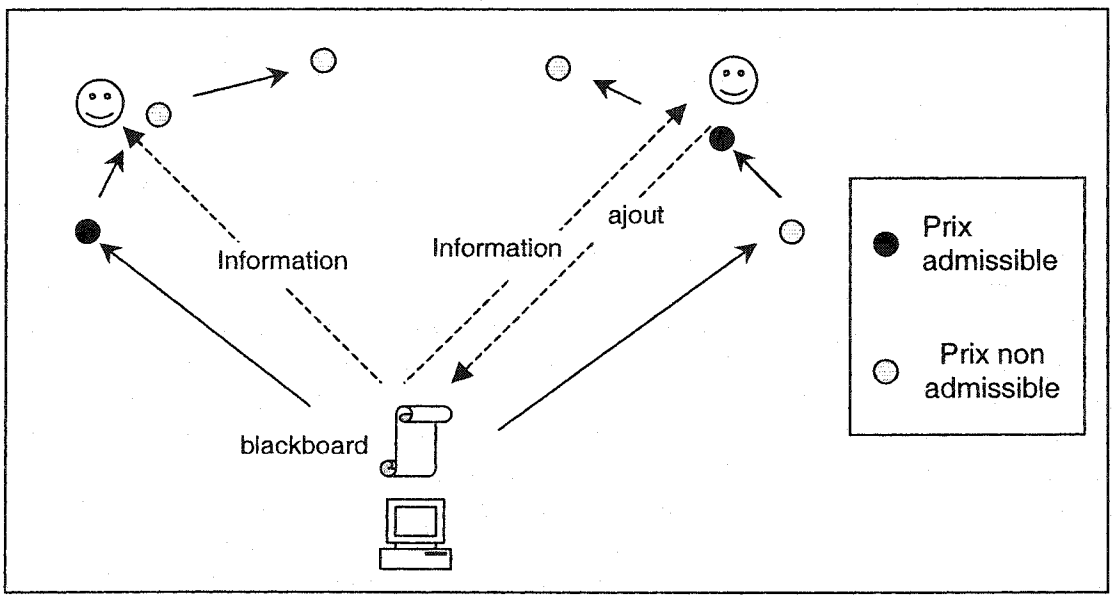


Figure 3.5 Collaboration par tableau noir (*blackboard*)

Le temps moyen de recherche des prix dépend de plusieurs facteurs comme le nombre de prix admissibles trouvés et, si ce nombre est égal à k, le rang du site sur lequel le $k^{ème}$ prix admissible a été trouvé. Pour estimer le temps de recherche global, nous devons séparer deux cas:

- Cas où les agents terminent la recherche des prix en trouvant seulement s prix admissibles $(s < k)$.
- Cas où les agents trouvent k prix admissibles.

```
ALGORITHME BLACKBOARD

DEBUT
   Tant que (Di n'est pas vide) FAIRE
            SI  (Appel_au_retour) ALORS QUITTER ;
            S = premier élément sur la liste Di ;
            Migrer vers S ;
            Chercher ( produit ) ;
            SI ( (produit_trouvé) ET (Prix (produit) ≤ prix_seuil ))  ALORS
                  Informer_Blackboard ( produit ) ;
            Di = Di – {S} ;
      Fin SI
   Fin Tant que
   Migrer (adresse_source) ;
FIN
```

Figure 3.6 Algorithme du tableau noir (*blackboard*)

Cas 1 : la recherche des prix se termine avec moins de k prix admissibles trouvés :

Dans ce cas, tous les agents parcourent la totalité des sites de leurs sous-domaines respectifs. Soit x_i le nombre de prix trouvés par l'agent A_i sur des sites distincts ($\sum_{i=1}^{m} x_i < k$). Alors, le temps approximatif de recherche de l'agent A_i peut s'exprimer ainsi:

$$T_i = (N_i + 1).l + N_i.t_t + x_i.t_m \tag{3.5}$$

Et puisque $0 \le x_i \le k-1$, alors :

$$(N_i + 1).l + N_i.t_t \le T_i \le (N_i + 1).l + N_i.t_t + (k-1).t_m$$

Le temps total de recherche est :

$$T = \underset{i}{Max} T_i$$

La charge réseau approximative induite est :

$$C^1 = \sum_{i=1}^{m} [(N_i + 1).p_a + x_i.p_m] \tag{3.6}$$

et : $$p_a.(N+m) \le C^1 \le p_a.(N+m) + p_m.(k-1)$$

Cas 2 : les agents trouvent k prix admissibles :

Dans ce cas, tous les agents vont suspendre leurs parcours dès que les k prix sont trouvés. Soit r_i le rang du dernier site visité par l'agent A_i avant de rentrer à la source et x_i le nombre de prix admissibles trouvés par l'agent A_i sur des sites distincts ($\sum_{i=1}^{m} x_i = k$).

Soit t_m le temps nécessaire pour l'envoi d'un message de A_i vers le *blackboard*, et soient t_m' le temps d'envoi d'un message du *blackboard* vers A_i et p_m' la taille de ce dernier. Nous supposons qu'un message a la même taille, qu'il contienne l'information sur un ou plusieurs prix trouvés. Le temps de recherche approximatif de l'agent A_i s'exprime comme suit :

$$T_i = (r_i + 1).l + r_i.t_t + x_i.t_m + t_m' \quad (0 \le x_i \le k \ \ et \ \ 0 \le r_i \le N_i) \qquad (3.7)$$

et le temps global approximatif de recherche est :

$$T = \underset{i}{Max} T_i$$

Pour trouver des bornes inférieure et supérieure à T, nous considérons deux cas de figure limites (extrêmes) : le « pire cas » et le « meilleur cas ». Le pire cas de figure qui réalise le plus grand temps de recherche se présente quand les k prix admissibles sont trouvés par un seul agent et qu'en plus, le $k^{ème}$ prix se trouve sur le $N_i^{ème}$ site (dernier site du sous-domaine). Dans ce cas, le temps de recherche est approximativement:

$$T^s = (N_i + 1).l + N_i.t_t + k.t_m \ \textbf{(Borne supérieure)}$$

Le meilleur cas de figure (temps de recherche minimum) se présente quand les k prix admissibles sont distribués équitablement sur les sous-domaines D_i et qu'en plus chaque agent trouve les prix admissibles sur les premiers sites visités. Autrement dit, chaque agent trouverait $g_i = E(k/m) + j_i$ prix admissibles sur les g_i premiers sites visités ($j_i = 0$ *ou* 1 *selon le reste de la division euclidienne de* k *par* m). Dans ce cas, le temps total approximatif serait :

$$T^i = (\underset{i}{Sup}(g_i) + 1).l + \underset{i}{Sup}(g_i).t_t + \underset{i}{Sup}(g_i).t_m + t_m' \ \textbf{(Borne inférieure)}$$

Et on a alors : $T^i \le T \le T^s$

La charge réseau approximative induite peut être exprimée par:

$$C = \sum_{i=1}^{m} [(r_i + 1).p_a + x_i.p_m + p_m^{'}] \quad (0 \le x_i \le k \quad et \quad 0 \le r_i \le N_i) \qquad (3.8)$$

La charge réseau dépend surtout de la position du $k^{ème}$ prix admissible dans le sous-domaine d'un agent.

Le temps moyen global de recherche et la charge moyenne réseau induite sont donc étroitement liés à la distribution et à la position des prix admissibles dans les sous-domaines des agents. Ces paramètres sont probabilistes et dépendent de la probabilité qu'un site donné contienne un prix admissible. Même si nous arrivons à exprimer mathématiquement le temps moyen en fonction de cette probabilité, il demeure qu'en pratique nous n'avons aucune connaissance de la valeur de cette probabilité. Notre estimation du temps moyen et de la charge réseau approximatifs pour chaque algorithme sera donc expérimentale et la comparaison entre les différentes solutions proposées sera empirique. À titre indicatif, nous donnons l'expression du temps moyen de recherche d'un seul prix admissible à l'aide d'un seul agent (k=m=1), en fonction de la probabilité p_i de trouver le prix admissible sur le site A_i:

$$T = l + t_t + p_1.(l + t_m) + \sum_{k=2}^{N} \left[\prod_{j=1}^{k-1} (1 - p_j) \right].(l + t_t + p_k.(l + t_m)) + \left\{ \prod_{j=1}^{N} (1 - p_j) \right\}.l + t_m^{'}$$

Une étude détaillée sur l'évaluation probabiliste des temps de parcours est faite dans (Moizumi, 98).

Collaboration par communication inter-agents

Dans cette deuxième solution collaborative, les agents n'ont pas recours à un intermédiaire commun, comme le *blackboard*, pour collaborer mais communiquent directement. La communication est du type « un à plusieurs » (*one-to-many*). Chaque fois qu'un agent trouve un prix admissible sur un site, il envoie un message de diffusion multipoint à tous les autres agents pour les en informer. De cette manière, tous les agents ont, à chaque instant, un compte rendu du résultat global de la recherche, et dès que le $k^{ème}$ prix admissible est trouvé par l'un des agents, tous les autres interrompent leur

recherche et rentrent à la machine source. Évidemment, si le domaine global de recherche contient moins de k prix admissibles, tous les agents A_i termineront le parcours de leurs sous-domaines respectifs et rentreront à la source avec les (k - k_i) prix trouvés ($\sum_{i=1}^{m}(k-k_i) \leq k$). La Figure 3.7 illustre ce type de collaboration et la Figure 3.8 décrit l'algorithme de collaboration par communication inter-agents.

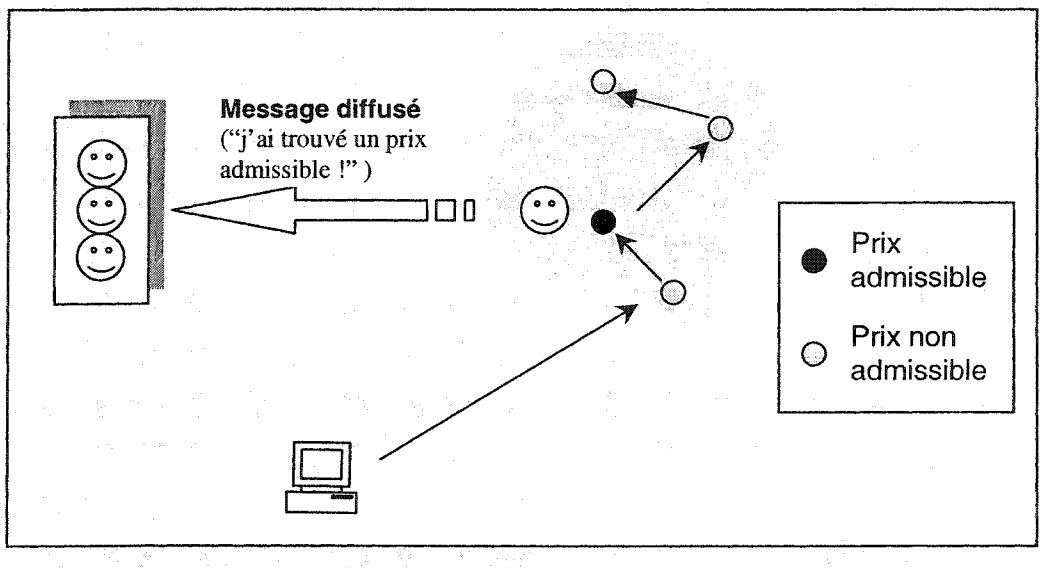


Figure 3.7 Collaboration par communication inter-agents

Quand un agent envoie un message aux autres agents, il ne connaît pas leurs positions respectives sur le réseau, surtout que ces positions sont dynamiques. C'est la plate-forme d'agents mobiles qui, grâce à la nomination et la localisation des agents, connaît à chaque instant la position et l'état de chaque agent qu'elle a envoyé. L'envoi des messages sera donc géré par la plate-forme ce qui ne va pas sans engendrer des temps de traitements et une charge réseau supplémentaires par rapport à la solution avec *blackboard.*

Soit x_i le nombre de prix admissibles trouvés par l'agent A_i sur des sites distincts de son sous-domaine ($0 \leq x_i \leq k$) et r_i le rang du site sur lequel l'agent A_i trouve le $x_i^{ème}$ prix admissible (le dernier dans son sous-domaine). Comme dans le cas de la

collaboration avec *blackboard*, nous pouvons estimer le temps moyen approximatif de recherche de l'agent A_i, en distinguant deux cas :

```
ALGORITHME COMMUNICATION INTER-AGENTS

Liste  Liste_prix_admissibles ; // Liste globale de tous les prix trouvés.
                                // Chaque agent possède la même copie de
                                // cette liste.
DEBUT
   Tant que ((Di n'est pas vide) ET (Liste_prix_admissibles.taille < k)) FAIRE
      S = premier élément sur la liste Di ;
      Migrer vers S ;
      Chercher ( produit ) ;
      SI ( (produit_trouvé) ET (Prix (produit) ≤ prix_seuil ))  ALORS
            envoyer_message_multicast ( Prix (produit) ) ;
            Liste_prix_admissibles ++ ;
      Fin SI
      Di = Di – {S} ;
      Si (message_multicast_recu) ALORS  Liste_prix_admissibles ++ ;

   Fin Tant que
   Migrer (adresse_source) ;
FIN
```

Figure 3.8 Algorithme de collaboration par communication inter-agents

- Cas où les agents terminent la recherche des prix en trouvant seulement s prix admissibles $(s < k)$.
- Cas où les agents trouvent k prix admissibles.

Cas 1 : la recherche des prix se termine avec moins de k prix admissibles trouvés :

Dans ce cas, tous les agents parcourent tous les sites de leurs sous-domaines respectifs ($\sum_{i=1}^{m} x_i < k$). Soit t_{mm} le temps nécessaire pour l'envoi d'un message *multicast* de l'agent A_i vers les autres agents. Nous supposons que t_{mm} est le même, que le message contienne

l'information sur un prix trouvé ou sur plusieurs. Le temps approximatif de recherche de l'agent A_i peut s'exprimer ainsi:

$$T_i = (N_i + 1).l + N_i.t_t + x_i.t_{mm} \tag{3.9}$$

Et puisque $0 \le x_i \le k-1$, alors :

$$(N_i + 1).l + N_i.t_t \le T_i \le (N_i + 1).l + N_i.t_t + (k-1).t_{mm}$$

Le temps total approximatif de recherche est :

$$T = \underset{i}{Max} T_i$$

Si p_{mm} est la taille d'un message de diffusion *multicast* alors la charge réseau approximative induite est :

$$C^1 = \sum_{i=1}^{m} [(N_i + 1).p_a + x_i.p_{mm}] \tag{3.10}$$

$$(N+m).p_a \le C^1 \le (N+m).p_a + (k-1).p_{mm}$$

Cas 2 : les agents trouvent k prix admissibles :

Dans ce cas, tout comme la solution avec *blackboard*, tous les agents vont suspendre leurs parcours dès que les k prix sont trouvés. Soit r_i le rang du dernier site visité par l'agent A_i avant de rentrer à la source et x_i le nombre de prix admissibles trouvés par l'agent A_i sur des sites distincts ($\sum_{i=1}^{m} x_i = k$). Le temps de recherche approximatif de l'agent A_i s'exprime comme suit :

$$T_i = (r_i + 1).l + r_i.t_t + x_i.t_{mm} \quad (0 \le x_i \le k \ \ et \ \ 0 \le r_i \le N_i) \tag{3.11}$$

et le temps global approximatif de recherche est :

$$T = \underset{i}{Max} T_i$$

Pour trouver des bornes inférieure et supérieure à T, nous considérons deux cas de figure limites: le « pire» et le « meilleur ». Le pire cas de figure qui réalise le plus grand temps se présente quand les k prix admissibles sont trouvés par un seul agent et qu'en plus le

$k^{ème}$ prix se trouve sur le $N_i^{ème}$ site (dernier site du sous-domaine). Dans ce cas, le temps de recherche approximatif sera:

$$T^s = (N_i + 1).l + N_i.t_t + k.t_{mm} \quad \textbf{(Borne supérieure)}$$

Le meilleur cas de figure (qui réalise le temps de recherche minimum) se présente quand les k prix admissibles sont distribués équitablement sur les sous-domaines D_i et qu'en plus, chaque agent trouve les prix admissibles sur les premiers sites visités. Autrement dit, chaque agent trouverait:

$g_i = E(k/m) + j_i$ prix admissibles sur les g_i premiers sites visités (j = 0 ou 1 selon le reste de la division euclidienne de k par m). Dans ce cas, le temps total approximatif serait:

$$T^i = (\underset{i}{Sup}(g_i) + 1).l + \underset{i}{Sup}(g_i).t_t + \underset{i}{Sup}(g_i).t_{mm} \quad \textbf{(Borne inférieure)}$$

Et on a alors : $T^i \leq T \leq T^s$

La charge réseau approximative induite peut être exprimée par:

$$C = \sum_{i=1}^{m} [(r_i + 1).p_a + x_i.p_{mm}] \quad (0 \leq x_i \leq k \;\; et \;\; 0 \leq r_i \leq N_i) \qquad (3.12)$$

Et : $k.p_{mm} + (k+m).p_a \leq C \leq k.p_{mm} + (N+m).p_a$

p_{mm} étant la taille d'un message *multicast*. p_{mm} est supposé la même, que l'agent trouve un prix admissible sur le site ou plusieurs prix.

Comme pour la solution avec *blackboard*, le temps moyen et la charge réseau induite moyenne dépendent de la probabilité de trouver un prix admissible sur un site donné.

3.2.3 Analyse et comparaison des algorithmes proposés

L'algorithme « individualiste » ne fait intervenir aucune communication entre les agents, ce qui fait économiser les temps d'envoi de messages. Par contre, l'absence de communication oblige les agents à continuer leur parcours jusqu'à ce qu'ils trouvent individuellement k prix admissibles dans leurs sous-domaines, puisqu'ils ne peuvent pas savoir si les autres agents ont trouvé une partie des k prix ou pas. Ceci oblige les agents à parcourir plus de sites qu'il n'en faudrait s'ils avaient été au courant des résultats

individuels des autres agents, ce qui rend cette solution non optimale par rapport au nombre de sites parcourus.

Aussi, cette solution présente l'avantage de permettre à la machine source de se déconnecter du réseau pendant que les agents font leur recherche.

Le temps total moyen de recherche T est de complexité linéaire en fonction de N_i. Or, pour un nombre de sites donné N, $N_i \equiv o(1/m)$ (m est le nombre d'agents). Donc $T \equiv o(1/m)$.

Quant à la charge moyenne totale C, elle est directement proportionnelle au nombre d'agents m : $C = \sum_{i=1}^{m} (N_i + 1).p_a = (N + m).p_a$. Donc: $C \equiv o(m)$ (complexité linéaire).

Les algorithmes collaboratifs intègrent des mécanismes de collaboration entre les agents. Ils ont pour but d'optimiser la visite des sites. Les agents sont informés dès que le but global de la recherche est atteint. Ils parcourent donc, en moyenne, moins de sites que si chaque agent devait réaliser individuellement le but global (comme dans l'algorithme « individualiste »). La contrepartie est la mise en place de mécanismes de collaboration qui peuvent, selon leur complexité, induire des temps de latence supplémentaires.

L'avantage de la solution avec *blackboard* est qu'elle met en jeu un « superviseur » (le *blackboard*) qui informe les agents par voie de message dès que les k prix admissibles ont été trouvés. Les agents sont donc exemptés de transporter les résultats de recherche des autres agents. Mais l'inconvénient de cette solution est la nécessité pour la machine source (qui offre le service de *blackboard*) d'être constamment connectée au réseau, engendrant des frais de connexion et une utilisation de la bande passante.

Le temps de recherche approximatif est :

- Au meilleur cas:

$$T^i = (G+1).l + G.t_t + G.t_m + t_m^{'} = G.(l + t_t + t_m) + l + t_m^{'}$$

avec $G = Sup_i(g_i)$ et $G \equiv o(1/m)$

- Au pire cas:

$$T^s = (N_i + 1).l + N_i.t_t + (k-1).t_m = N_i.(l + t_t) + l + (k-1).t_m$$

et $N_i \equiv o(1/m)$

Donc, $T \equiv o(1/m)$.

Quant à la charge réseau induite par cette solution, on a:

- Au meilleur cas:

$$C = k.(p_m + p_a) + m.(p_a + p_m')$$

- Au pire cas:

$$C = (m+N).p_a + k.p_m$$

Donc, $C \equiv o(m)$ (complexité linéaire).

La solution collaborative avec communication inter-agents présente l'avantage de permettre à la machine source de se déconnecter pendant que les agents effectuent leur recherche de prix. L'inconvénient, par contre, est que la communication entre agents pourrait engendrer des délais de communication plus grands que ceux de la solution avec *blackboard*, car les messages *multicast* entre un agent et les autres sont plus lourds que les messages simples entre un agent et la machine source. Le temps de recherche approximatif est :

- Au meilleur cas:

$$T^i = (G+1).l + G.t_t + G.t_{mm} = G.(l + t_t + t_{mm}) + l$$

avec $G = Sup_i(g_i)$ et $G \equiv o(1/m)$

- Au pire cas:

$$T^s = (N_i + 1).l + N_i.t_t + k.t_{mm} = N_i.(l + t_t) + l + k.t_{mm}$$

$$\text{et} \quad N_i \equiv o(1/m)$$

Donc, $T \equiv o(1/m)$

Quant à la charge réseau induite par cette solution, on a:

- Au meilleur cas:

$$C = k.(p_a + p_{mm}) + m.p_a$$

- Au pire cas:

$$C = (m+N).p_a + k.p_{mm}$$

Donc: $C \equiv o(m)$ (complexité linéaire).

3.3 Tests et mesures à faire

Dans cette section, nous nous intéressons à l'étude des performances et de l'évolutivité des architectures multi-agents mobiles. Nous voulons étudier l'évolution des paramètres « temps global moyen de recherche », « charge réseau moyenne » et le rapport {temps de recherche / charge réseau} en fonction du nombre d'agents mis en jeu dans la recherche des prix. Nous ferons cette étude de performance pour chacun des algorithmes proposés. Les paramètres qui resteront fixes pour toutes les mesures sont :

- le nombre de sites N ;
- le nombre k de prix admissibles à trouver ;
- le domaine de variation du nombre d'agents m. C'est l'ensemble de valeurs dans lequel nous ferons varier m, il va de 1 (solution mono-agent) jusqu'à N (cas extrême). Par exemple, D_m=*{1, N/3, N/2, 2N/3, N}*.

Le mode opératoire sera comme suit :

- choisir et fixer un algorithme de coordination ;
- mesurer le temps moyen de recherche et la charge réseau pour différentes valeurs de m dans l'intervalle D_m ;
- refaire ces opérations pour les autres algorithmes et comparer leurs performances ;

Les valeurs mesurées du temps global de recherche et de la charge réseau doivent être des valeurs moyennes, nous allons donc choisir plusieurs scénarios dans lesquels les positions des prix admissibles sur le réseau varient. Nous allons par exemple considérer un scénario où les *k* prix admissibles se trouvent sur les premiers sites visités par le premier agent, un autre scénario où ils se trouvent sur les premiers sites des *k* premiers sous-domaines, un troisième scénario où les prix admissibles sont aléatoirement distribués, etc.

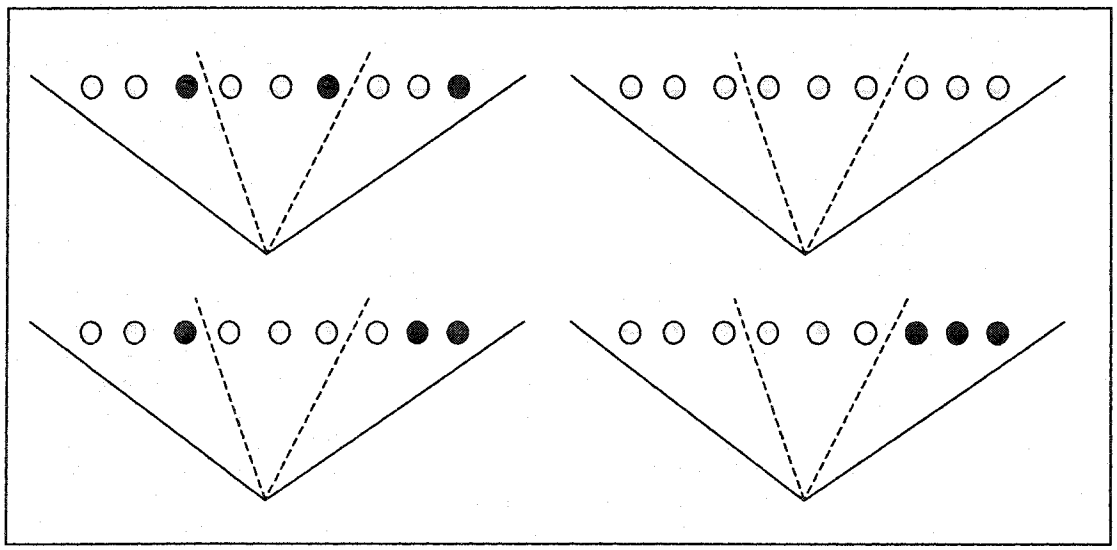


Figure 3.9 Exemples de scénarios expérimentaux pour mesurer le temps de recherche et la charge réseau moyens. Cas particulier où N=9, m=3 et k=3

3.4 Architecture générale et structure des agents

L'architecture générale de notre application à base d'agents est présentée dans la Figure 3.10. Elle repose sur trois parties principales: un agent fixe sur la machine du client, un agent mobile qui sera émis à partir de la machine du client pour aller sur les sites marchands et des agents stationnaires sur les sites marchands.

- L'agent fixe sur la machine du client communique avec l'utilisateur à travers une interface graphique pour saisir les critères de recherche du client par rapport au produit cherché. Il gère aussi les agents mobiles qu'il envoie sur le réseau et reçoit des rapports

d'activité de la part de ces derniers pour prendre les décisions appropriées (interruption de la recherche, gestion d'une situation imprévue, etc.).

- Le (ou les) agent(s) mobile(s) migre(ent) sur le réseau et se déplace(ent) entre les sites marchands. Sur un site marchand, l'agent mobile consulte les catalogues des produits et décide si un produit trouvé correspond ou pas à ses critères de recherche.
- Un agent stationnaire sur chaque site marchand. Cet agent communique avec les agents mobiles qui visitent le site, leur proposent des services comme la recherche d'un produit spécifique dans les catalogues.

L'agent fixe se trouve sur la machine de l'utilisateur et se compose des modules suivants :

- **IntUsr** : Interface graphique pour communiquer avec l'utilisateur. Elle sert à la saisie des paramètres de recherche et à l'affichage des résultats et d'éventuelles erreurs.
- **ExtCom** : Module de communication externe. Ce module sert à communiquer avec le portail. Il reçoit les paramètres de recherche de *IntUsr*, les transmet au portail qui lui renvoie une liste des sites marchands sur lesquels le produit recherché pourrait se trouver. Les sites sur cette liste vont être visités par l'agent mobile.
- **Dispatcher** : Module d'affectation des tâches. Ce module reçoit la liste des sites de la part de *ExtCom* et affecte à chaque agent la liste des sites qu'il devra visiter.
- **CtrlComAg** : Module de contrôle et de communication avec les agents mobiles. Ce module reçoit les paramètres de recherche de la part de *IntUsr* ainsi que les listes d'affectation des sites à chaque agent de la part de *Dispatcher* et les transmet aux agents qu'il envoie.

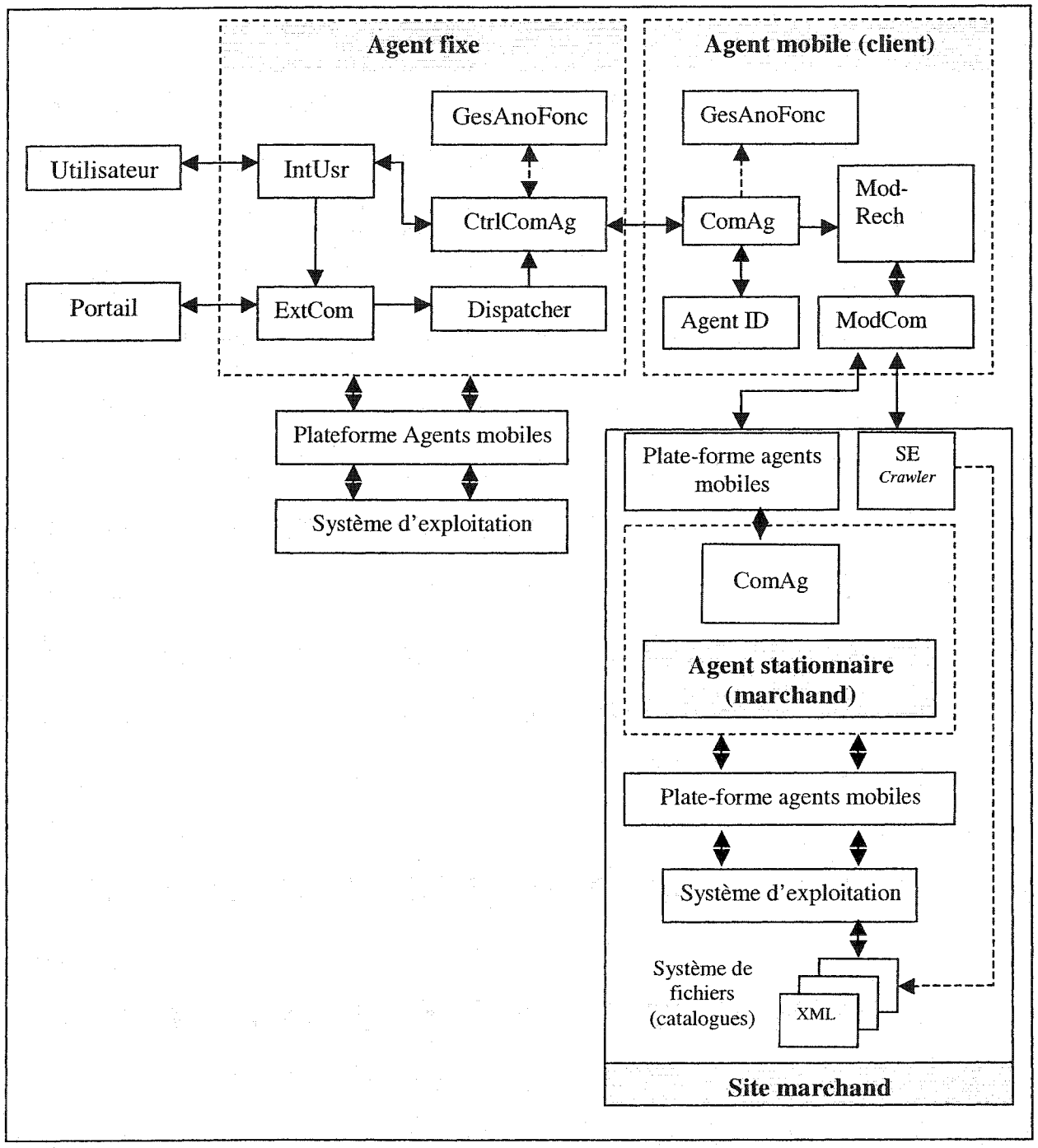


Figure 3.10 Architecture fonctionnelle de l'agent fixe et des agents mobiles

- **GesAnoFonc :** Module de gestion des anomalies de fonctionnement. Ce module se charge de prendre des décisions en cas de dysfonctionnement ou de situation imprévue, comme la perte d'un agent, un délai de réponse trop élevé, etc. Ce module peut décider par exemple d'envoyer un autre agent sur le site sur lequel un agent est perdu. *GesAnoFonc* communique avec *CtrlComAg* pour être notifié d'un mauvais fonctionnement.

Les agents mobiles sont envoyés sur des sites distants par le module *CtrlComAg* de l'agent fixe, ils sont composés des modules suivants :

- **ComAg :** Module de communication avec les autres agents. Ce module reçoit de *CtrlComAg* l'ordre de migration avec les adresses des machines à visiter. Il sert aussi à communiquer avec le *blackboard* ou les autres agents (selon l'algorithme de collaboration employé) pour les informer d'un éventuel prix admissible trouvé.
- **GesAnoFoncMob :** Module de gestion des anomalies de fonctionnement de l'agent mobile. Ce module sert à gérer les anomalies et les cas de fonctionnement imprévus quand le module *GesAnoFonc* fait défaut ou quand l'agent mobile perd contact avec l'agent fixe.
- **ModRech :** Module de recherche. C'est le module principal de l'agent mobile, il a pour rôle l'interrogation des catalogues des produits, l'extraction et l'analyse des données.
- **AgentID :** Module d'identification de l'agent. Ce module renferme les informations d'identification de l'agent, comme son *numéro d'identification*, son *agent émetteur*, ou *station émettrice*, *date de départ*, *liste des sites à visiter*, etc.

  Ces informations sont utiles pour la communication entre l'agent mobile avec sa station source, l'authentification de l'agent mobile auprès de celui ci et la migration de l'agent.
- **ModCom :** Module de communication avec l'agent stationnaire marchand. Lors d'une recherche sur les produits exposés, ce module communique avec le

moteur de recherche interne du site marchand en ouvrant une session, en transmettant les requêtes et en recevant les résultats. Ce module joue aussi le rôle d'un canal de communication entre l'agent mobile et l'agent stationnaire marchand, en transmettant et en recevant les messages échangés.

Les deux modules de communication *ComAg* et *ModCom* auraient pu être intégrés dans le même module, mais leur séparation est motivée par une raison d'évolutivité. En effet, les agents stationnaires des sites marchands peuvent être très hétérogènes et leur interface de communication peut changer avec le temps et selon le marchand. Si une nouvelle fonctionnalité ou une nouvelle interface d'agents stationnaires devait être prise en compte, seul le module de communication *ModCom* serait modifié.

L'agent stationnaire sur le site marchand est composé des modules suivants:

- **ComAg :** Module de communication avec l'agent mobile. Ce module échange les messages avec l'agent mobile qui visite le site marchand.

Le site marchand comprend, mis à part l'agent stationnaire, un moteur de recherche interne **SE** qui aide l'agent mobile visiteur à rechercher un produit, à partir de mots de clés. **SE** se met périodiquement à jour grâce à des robots *(crawlers)*.

Notons que l'agent mobile doit être le plus léger possible, car le cas contraire risque d'engendrer des temps de latences élevés et agir négativement sur les performances (temps de recherche et charge réseau). Un module comme *GesAnoFoncMob* doit se restreindre à une action simple dans la cas de corruption ou de perte d'un agent par exemple, comme la décision de tuer l'agent, mais elle ne devrait pas contenir des algorithmes complexes.

Les agents communiquent avec le système d'exploitation et les systèmes de fichiers à travers une plateforme de gestion des agents mobiles. Cette plateforme doit être installée sur la machine source et sur toutes les machines visitées par les agents. Ceci pourrait être aperçu comme un inconvénient vu que sur la marché, il n'y pas encore de standard

unique de plateformes. Il existe plusieurs plates-formes propriétaires comme *D'agents, Aglet, Voyager* et *Grasshopper*. C'est cette dernière que nous utiliserons par la suite pour nos simulations.

3.5 Communication entre les agents

La communication entre les agents est assurée par un échange de messages. Les messages sont transmis par appels de méthodes. Chaque agent possède un ensemble bien défini de méthodes contenues dans son interface publique, accessible à l'autre agent. Pour qu'un agent ***A*** transmette un message ***msg*** à un agent ***B***, il appellera la méthode $M_G(msg)$. Pour des fins pratiques, c'est l'agent client qui va mener la communication en appelant les méthodes du marchand car l'interface de ce dernier est publique et n'importe quel client peut y avoir accès. Par contre, le marchand ne connaît pas tous les clients et ne peut donc par avoir facilement accès à leurs interface. Un *proxy* jouera le rôle de pont de communication entre les deux agents. Une fois l'agent client est sur le site du marchand, il sollicitera le *proxy* de ce dernier pour établir une communication. La communication entre l'agent client et l'agent marchand via *proxy* est illustrée dans la Figure 3.11.

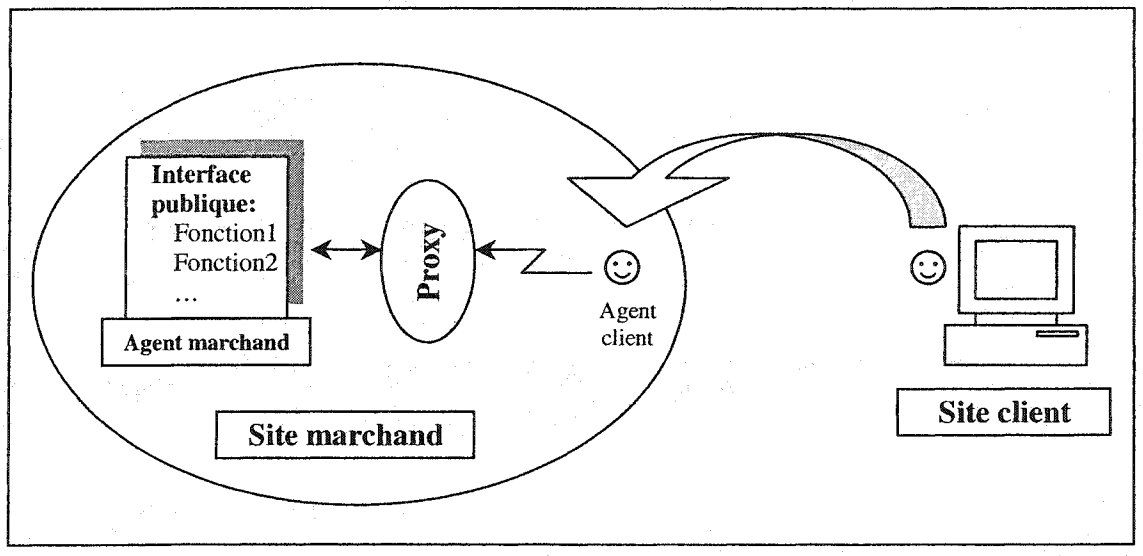


Figure 3.11 Communication entre les agents via un proxy

L'utilisation du *proxy* limite le champ d'action de l'agent du client, en ne lui offrant qu'un ensemble restreint de fonctions qu'il peut utiliser, limitant ainsi l'accès aux ressources et aux informations du marchand. Le *proxy* offre donc une protection pour le site marchand contre d'éventuelles attaques.

La communication entre l'agent stationnaire (*blackboard*) et les agents mobiles est aussi assurée dans les deux sens via un *proxy*. La communication entre les agents dans l'algorithme avec communication inter-agents est aussi assurée via un *proxy* de groupe (*group proxy)* qui permet d'envoyer un message *multicast* à tous les agents mobiles membres du groupe. La Figure 3.12 illustre la communication entre l'agent stationnaire (*blackboard*) et les agents mobiles dans l'algorithme avec *blackboard*. La Figure 3.13 illustre la communication via un *proxy* de groupe entre les agents mobiles dans l'algorithme avec communication inter-agents.

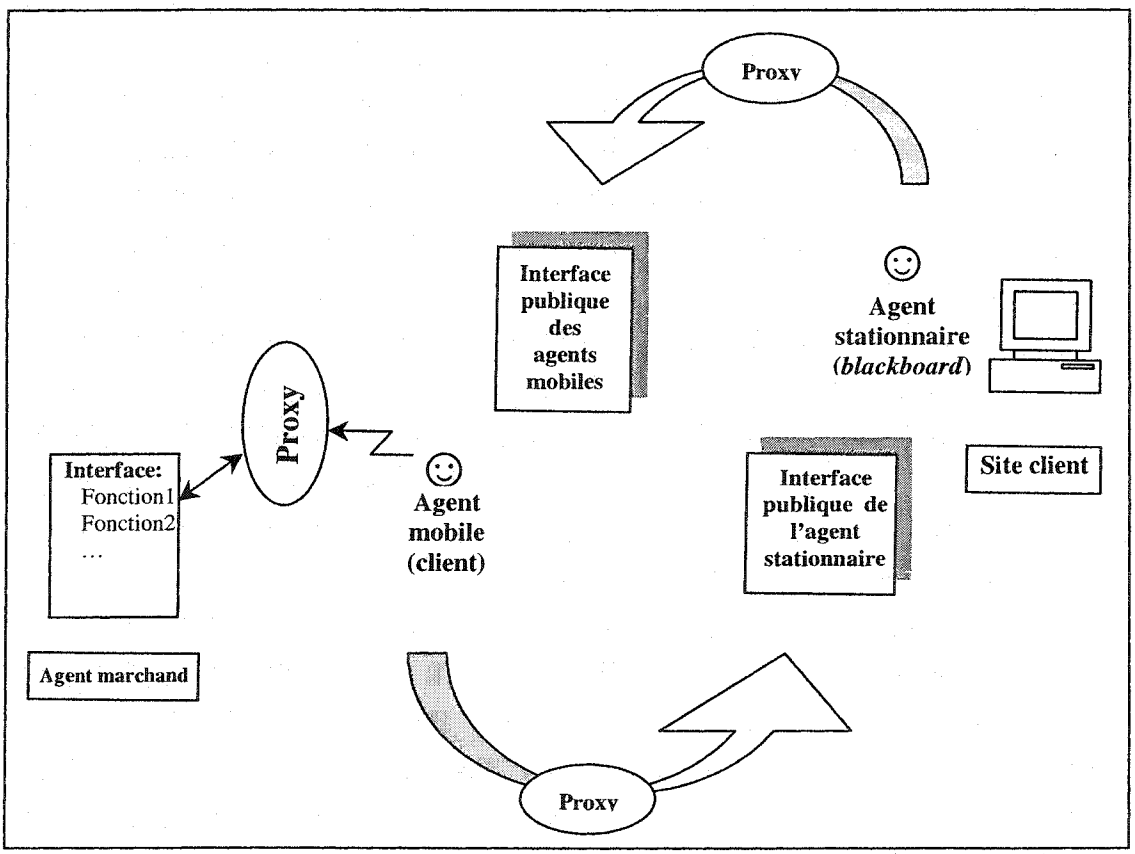


Figure 3.12 Communication via proxy entre l'agent stationnaire et les agents mobiles

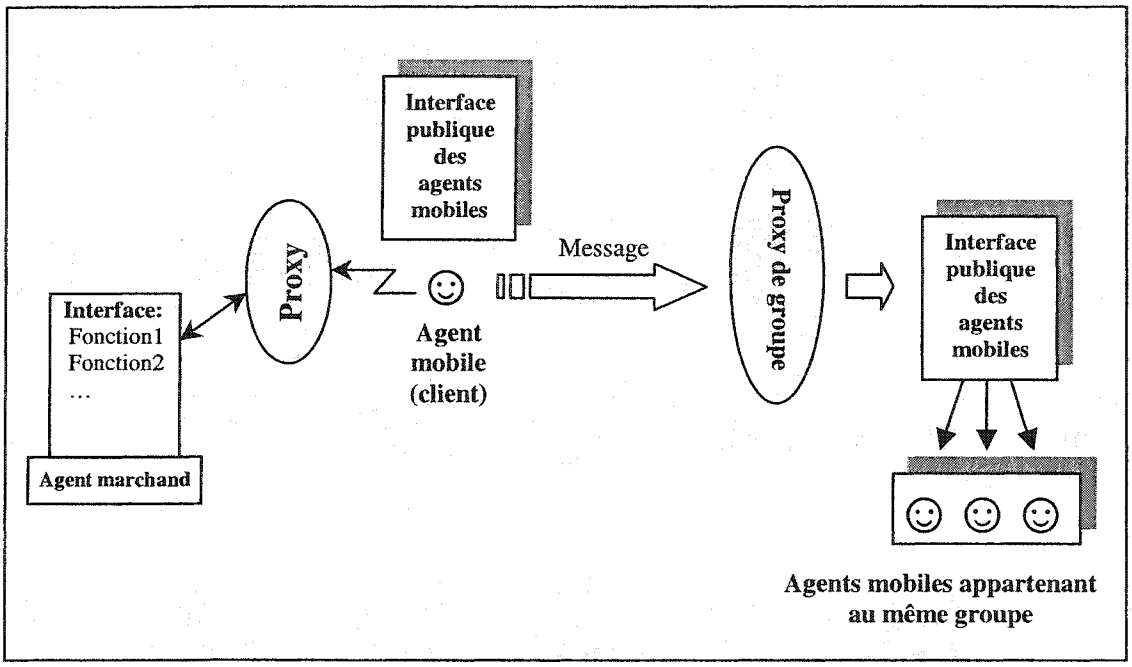


Figure 3.13 Communication inter-agents mobiles via un proxy de groupe

CHAPITRE IV
IMPLÉMENTATION ET RÉSULTATS

Dans le chapitre précédent, nous avons exposé le problème de recherche de prix en commerce électronique, au moyen d'agents mobiles. Nous avons alors présenté une architecture à plusieurs agents mobiles pour rechercher k prix admissibles sur N sites distribués. Chaque agent recherche un produit spécifique pour le compte du client, sur un ensemble prédéfini de sites marchands. Nous avions présenté trois algorithmes de collaboration entre les agents. Le premier, « individualiste », n'utilise aucun mécanisme de communication entre agents. Le second est basé sur un agent moniteur, le tableau noir (*blackboard*) auquel les agents communiquent les résultats de leur recherche. Le troisième algorithme repose sur une communication inter-agents. Chaque agent est alors informé des résultats de recherche de tous les autres agents pendant toute l'opération de recherche de prix. Dans ce chapitre, nous mettons en place notre architecture et nous présentons les résultats de l'étude de performance de chaque algorithme proposé. L'étude de performance est basée sur une mesure et une comparaison des temps et des charges réseau induits par chaque algorithme. Commençons d'abord par décrire notre environnement matériel et logiciel.

4.1 Environnement matériel et logiciel

L'implémentation et les mesures ont été réalisées au sein du LARIM (Laboratoire de Recherche en Réseautique et Informatique Mobile) à l'École polytechnique de Montréal. Nous avons utilisé le réseau local LAN Ethernet/Fast Ethernet 10/100 Mbps du LARIM. Les machines utilisées comme sites marchands sont dotées d'un processeur Pentium III à 800 MHz et de 256 Mo de mémoire vive.

Notre réseau de test n'est pas un réseau dédié. Donc, afin de limiter la marge d'erreur sur les mesures prises, nous avons réalisé nos mesures pendant la nuit, lorsque le réseau du LARIM était inutilisé.

L'environnement logiciel est composé de:

- Système d'exploitation: Windows 2000 Professionnel.
- Environnement et machine virtuelle Java: JDK 1.3.1 inclus dans le kit Borland Jbuilder 7.0.
- Environnement de développement: Borland Jbuilder 7.0, incluant la JVM (Java Virtual Machine), l'environnement graphique et les librairies de développement.
- Plate-forme de gestion d'agents mobiles: Grasshopper 2.2.4b de IKV++ Technologies AG.
- Le parseur XML Xerces d'Apache. C'est un ensemble de librairies basées sur les APIs standards DOM (*Document Object Model*) et qui permettent d'analyser, d'éditer et de valider un document XML.
- EtherPeek 4.0.2 de *AG Group*. C'est un logiciel de mesure de charge réseau. Il permet de suivre en temps réel la charge réseau passant par le nœud sur lequel le logiciel est installé. Il permet aussi de filtrer la charge induite par un protocole donné, ou traversant un nœud ou un port donné.

Pour mesurer la charge réseau, nous avons utilisé le logiciel de capture de trames EtherPeek. Afin de limiter la charge mesurée à celle générée par notre application, nous avons réalisé un filtre sur les trames capturées. En effet, les plates-formes Grasshopper communiquent sur certains ports logiques bien connus (7000, 7002, 7004, etc.). Nous avons alors filtré les trames provenant de, ou destinées à, ces ports pour avoir une mesure plus exacte sur la charge induite par notre application et éviter de compter les autres paquets circulant sur le réseau (paquets de signalisation TCP/IP, paquets de service, etc.).

Pour la manipulation des agents, nous utilisons la plate-forme Grasshopper 2.2.4b. Grasshopper offre un ensemble d'APIs qui permettent d'envoyer, recevoir et exécuter les agents. Les agents sont des classes java pré-compilées (ayant l'extension *.class*) qui doivent hériter de la classe mère *de.ikv.grasshopper.agency.Agent* contenue dans les APIs de Grasshopper. Deux méthodes principales gèrent le cycle de vie d'un agent et

doivent obligatoirement être implémentées dans le corps de l'agent pour que la plate-forme puisse exécuter correctement l'agent:

- *Init (Object() Arguments)*: cette méthode est exécutée au moment de la création de l'agent.
- *Live()*: c'est la méthode principale de l'agent. Elle est exécutée chaque fois que l'agent arrive sur un site, y compris sur le site de création de l'agent. C'est dans cette méthode qu'est implémentée la « logique métier » de l'agent.

Un agent est créé, envoyé et exécuté dans une *place*. Une place est un endroit virtuel au sein d'une *agence*. L'agence est aussi un espace logique contenant des places. La notion de places et agences est offerte par Grasshopper pour fins de séparation logique entre espaces d'exécution des agents. Un agent ne peut être créé et exécuté que dans une place, laquelle se trouve dans une agence. L'agence agit aussi comme serveur pour accueillir et exécuter les agents provenant de sites étrangers. Grasshopper offre une interface textuelle et une autre graphique pour visualiser et gérer les agents. La Figure 4.1 illustre ces interfaces.

Grasshopper supporte également des mécanismes d'enregistrement d'agences et d'agents dans une *région*. Une région est un service de « pages jaunes » publiques qui permet à des agences de s'y enregistrer pour qu'elles soient connues par des agents ou des agences étrangères. Un agent étranger peut interroger une région, tout comme un bottin téléphonique, pour connaître les agences ou les agents qui y sont enregistrés afin de communiquer avec elles/eux. Dans Grasshopper, la région garde la trace de tous les agents qui y sont enregistrés ainsi que de leurs positions sur le réseau pour faciliter la communication entre agents en déplacement.

Pour détecter certains évènements relatifs aux composantes de la plate-forme (comme la création ou la suppression d'un agent ou d'une agence, le départ d'un agent, etc.), Grasshopper utilise des mécanismes d'écoute, appelés *listeners*, qui captent les évènements générés par la plate-forme. Ces évènements sont :

- Création, suppression, migration et arrivée d'un agent ;
- Création et suppression d'une place ;

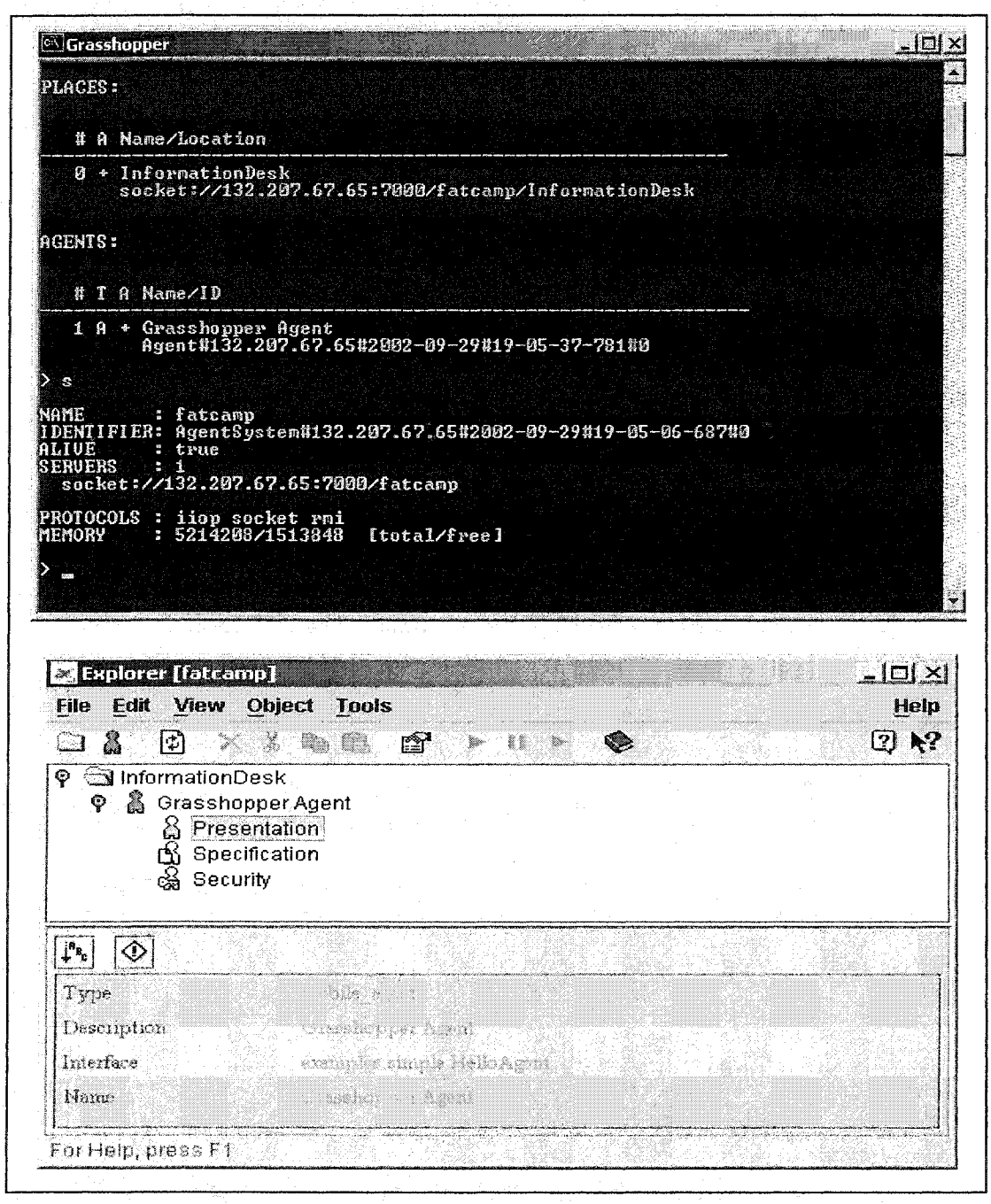


Figure 4.1 Interfaces textuelle et graphique de la plate-forme Grasshopper

- Création et suppression d'une agence.

Des actions réagissant à chaque événement peuvent être implémentées dans les méthodes correspondant à l'événement en question.

4.2 Implémentation et mode opératoire

Rappelons que l'objectif de ce mémoire est de concevoir, mettre en place et tester les performances d'une architectures où plusieurs agents mobiles font de la recherche collective de prix sur un ensemble de sites marchands distribués sur le réseau. Dans un premier temps, nous mettons en place cette architecture et dans un deuxième temps, nous testons différents algorithmes (individualiste, *blackboard*, communication inter-agents) en mesurant à chaque fois le temps total de recherche et la charge réseau induite par chaque solution.

4.2.1 Implémentation de l'architecture

Notre architecture est entièrement implémentée en Java. Elle comporte principalement des classes suivantes:

- *AgentManagerCls*: C'est la classe responsable de créer les agents dans une agence qui tourne déjà sur la machine du client.
- *LocalAgent*: Cette classe définit un agent fixe sur la machine du client qui sera responsable d'envoyer les agents mobiles « chercheurs de prix », leur affecter les adresses qu'ils vont visiter et jouer le rôle de tableau noir (*blackboard*) qui consiste à recevoir les résultats de recherche des différents agents au cours de leurs parcours respectifs et de les ordonner de rentrer à la machine du client une fois l'information globale trouvée.
- *AgentListener*: Cette classe implémente l'interface *de.ikv.grasshopper.agency.ISystemListener* qui permet de capter les évènements qui surviennent sur une agence, comme le départ ou l'arrivée d'un agent.

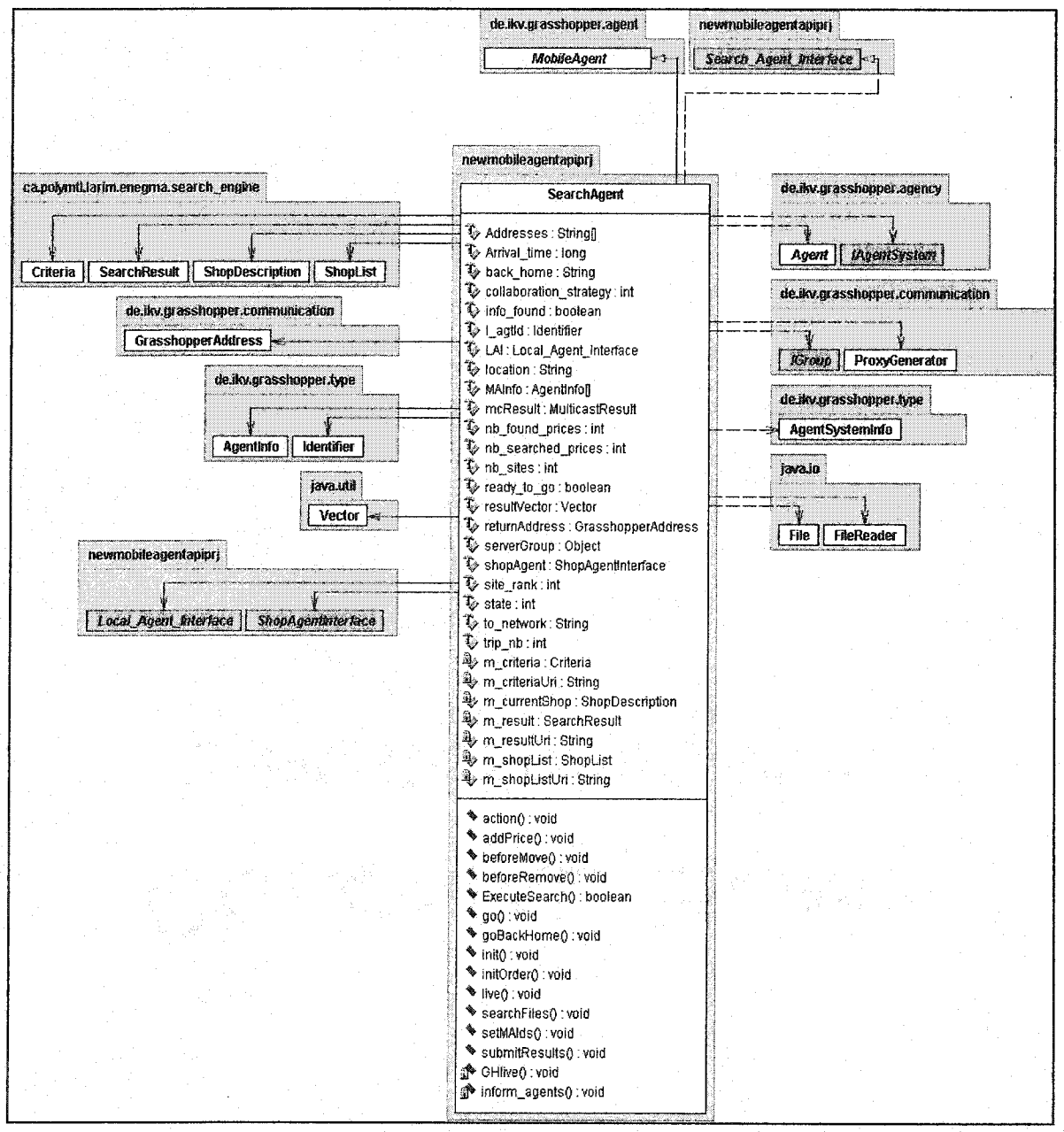


Figure 4.2 Représentation en UML de la classe SearchAgent de l'agent mobile de recherche

- *SearchAgent*: C'est la classe principale qui décrit l'agent « chercheur de prix ». Plusieurs instances de cette classe seront envoyées sur le réseau par l'agent fixe *LocalAgent* pour effectruer la recherche de prix sur les sites marchands.

Les Figures 4.2 et 4.3 représentent respectivement les classes *Search_Agent*, qui définit l'agent mobile, et la classe *Local_Agent*, qui définit l'agent fixe gérant les agents mobiles et jouant aussi le rôle de tableau noir dans l'algorithme avec *blackboard*.

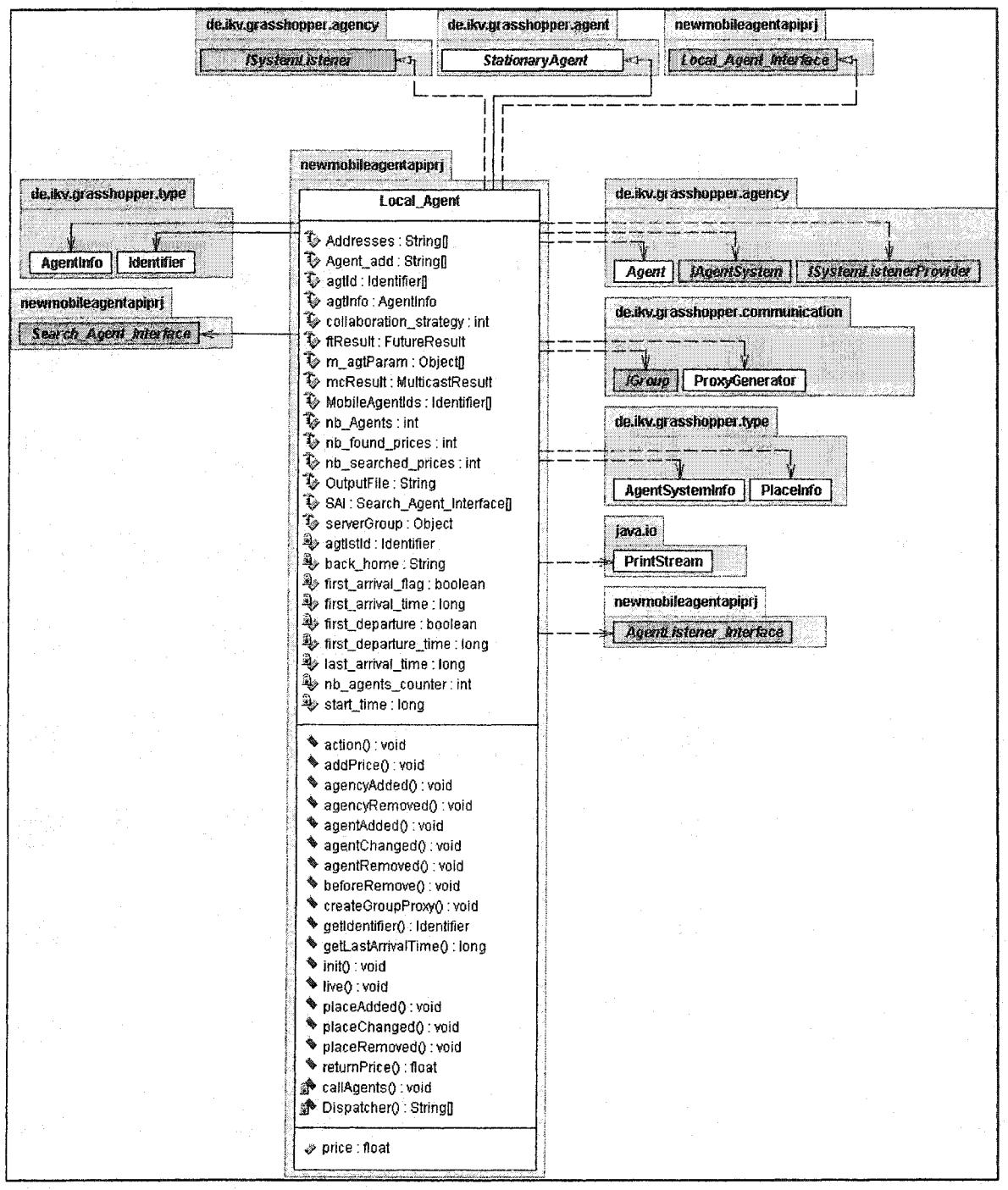


Figure 4.3 Représentation en UML de la classe "Agent local" (Local_Agent)

Les agents *SearchAgent* recherchent dans les galeries publiques des sites marchands un produit spécifié par le client. Pour décrire les produits, nous avons utilisé le standard XML qui est un langage de description par balises. Il a une structure arborescente qui permet de représenter des *éléments* avec leurs caractéristiques et leurs *éléments fils* qui sont, à leur tour, des *éléments*, etc. La Figure 4.4 illustre un exemple simple de description en XML d'un catalogue contenant un (1) produit.

```
<?xml version="1.0" encoding="UTF-8" ?>
  - <Catalog xmlns="http://larim.polymtl.ca/ENegMA"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://larim.polymtl.ca/ENegMA
    Catalog.xsd" version="3.0">
    <Catalog-Name>Voitures neuves</Catalog-Name>
    - <Catalog-Entry>
      - <Product version="3.0">
        <Product-SKU>0021</Product-SKU>
        <Product-Name>Toyota Corolla </Product-Name>
        <Product-Description>4 portes, Rouge, boite
          automatique, R-CD, A/C, Essence, 90 Chev fiscaux
        </Product-Description>
        <Product-Price>20.999</Product-Price>
    </Product>
      <Quantity>10</Quantity>
  </Catalog-Entry>
</Catalog>
```

Figure 4.4 Exemple d'un catalogue en XML

Pour analyser un fichier XML, filtrer des produits spécifiques et extraire notamment l'information sur le prix, l'agent *SearchAgent* utilise un analyseur XML (*XML parser*), basé sur les APIs *Xerces* d'*Apache* (www.apache.org). Ces APIs utilisent le standard DOM (*Document Object Model*) de w3C (http://www.w3.org/DOM/). Nous avons choisi le langage de description XML pour la simplicité de sa syntaxe, sa facilité à être implanté (Il nécessite simplement un analyseur, ou *parser*, et un navigateur web pour l'interpréter) et son interopérabilité (il est interprété par la plupart des navigateurs web du marché). Notons au passage la présence sur le marché d'un ensemble de modules offrant une standardisation en XML des spécifications pour le commerce électronique,

c'est la plate-forme ebXML dont nous avons donné un bref aperçu dans notre revue de littérature (cf. paragraphe 2.2.3). Pour nos besoins de tests, les fonctionnalités de base de XML étaient suffisantes.

Nous ne rentrerons cependant pas dans les détails d'implémentation de notre architecture et nous préférons renvoyer en Annexe 1 tous les modèles des classes de notre architecture, décrits en langage UML (*Unified Modeling Language*).

4.2.2 Mode opératoire et mesure de performance

Dans le chapitre 3, nous avons proposé 3 algorithmes régissant le comportement des agents lors de la recherche de prix. Nous avons présenté les avantages et les inconvénients de chaque algorithme par rapport aux autres. Dans cette section, nous implémentons ces algorithmes et testons leurs performances. Nous mesurons le temps de recherche global et la charge réseau induite par chaque algorithme. Nous nous intéressons à l'évolution de ces deux paramètres en fonction du nombre d'agents. A priori, quand le nombre d'agents augmente, le nombre de sites à parcourir par agent diminue et la tâche globale est exécutée en parallèle par plus d'agents ce qui diminuerait le temps de recherche mais augmenterait la charge réseau. Mais la collaboration entre les agents met en jeu des messages et donc une charge réseau supplémentaire, et peut engendrer des latences supplémentaires. Le temps de recherche et la charge réseau induite n'évolueraient donc pas en fonction du nombre d'agents de la même façon, selon que l'on utilise un algorithme ou l'autre. Nos mesures devront nous permettre de comparer empiriquement les algorithmes proposés.

Dans le problème des prix admissibles, il s'agit pour les agents de trouver k prix sur un ensemble de N sites distribués. Pour chaque nombre d'agents employés dans la recherche, différents cas de figure peuvent se présenter selon la position des prix admissibles sur les sites visités. Par exemple, il est évident que les temps de recherche et la charge réseau seront différents d'un scénario où les k prix admissibles sont trouvés sur les k premiers sites visités par les agents à un autre où ils sont trouvés sur les k derniers sites.

Pour avoir des valeurs moyennes du temps et de la charge réseau, il faudrait, idéalement, faire la moyenne de ces valeurs sur l'ensemble des scénarios possibles (c'est à dire toutes les positions possibles des k prix admissibles sur les N sites) et les pondérer par leurs probabilités de se produire, ce qui nécessite une connaissance préalable de la distribution des probabilités en fonction des scénarios possibles. Ceci est irréaliste et difficile à réaliser en pratique puisque la simple probabilité de trouver un prix admissible sur un site donné est difficile à estimer, comme c'est expliqué dans le paragraphe 3.2.2 du chapitre 3. Nous proposons alors de considérer, pour différentes valeurs du nombre d'agents, deux scénarios extrêmes : le « meilleur » et le « pire ». Dans le « meilleur » cas, les prix admissibles sont trouvés sur les premiers sites visités par les agents. Ce scénario devrait donner le plus petit temps de recherche et la plus petite charge réseau pour l'algorithme considéré. Dans un « pire » cas, les prix admissibles se trouvent sur les tout derniers sites visités, de telle façon que les agents enregistrent le plus grand temps de recherche et la plus grande charge réseau. Dans les tests que nous avons effectués, et que nous exposons dans la prochaine section, nous avons pris un exemple avec $N = 9$ sites et $k = 2$ prix admissibles à trouver. La Figure 4.5 illustre la position des prix admissibles dans le meilleur et le pire cas, pour plusieurs valeurs du nombre d'agents. Les mesures que nous avons faites nous donnent des courbes de variation de la charge totale et du temps de recherche en fonction du nombre d'agents dans le meilleur et le pire cas de figure.

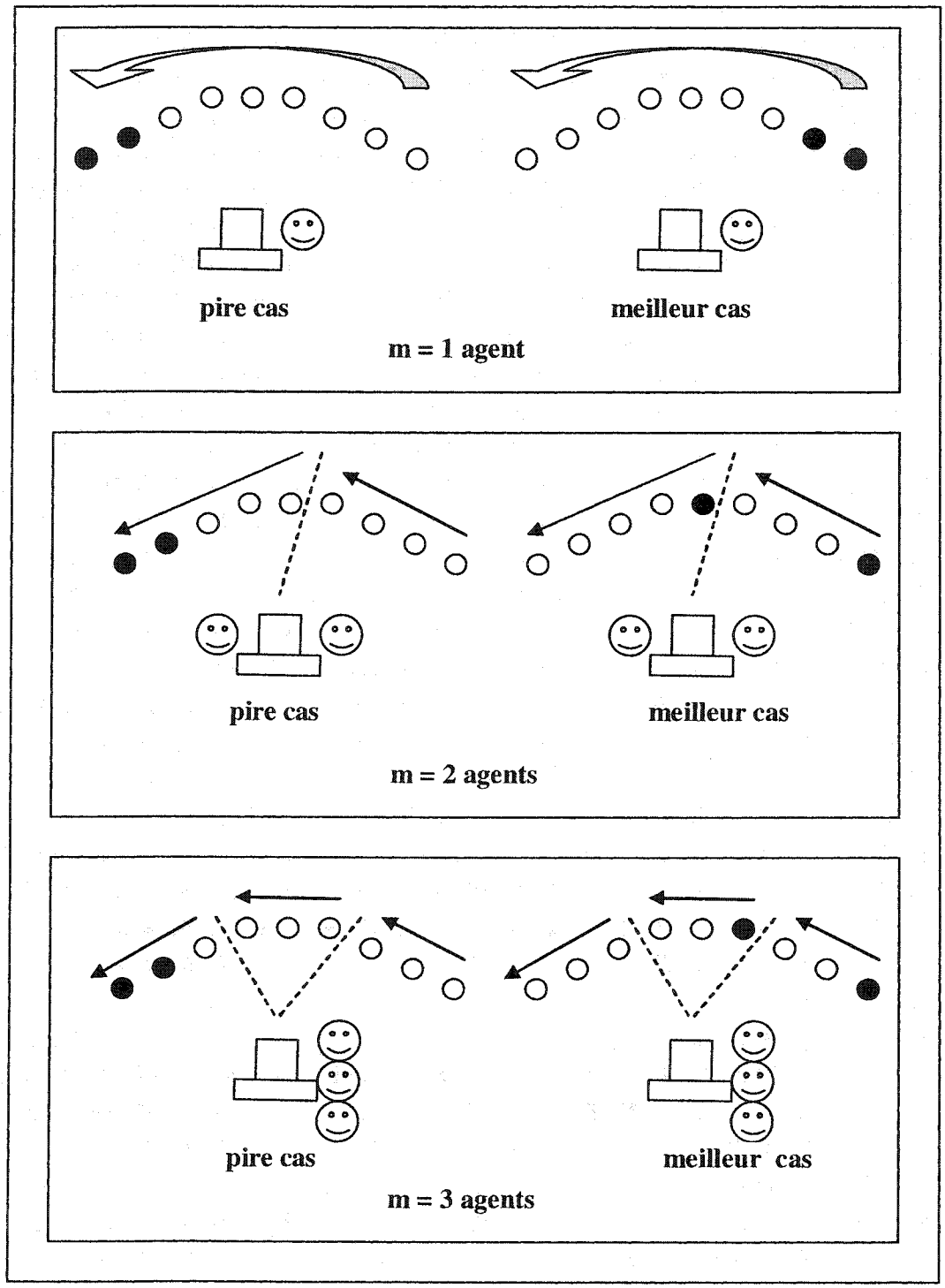


Figure 4.5 Meilleur et pire scénarios pour m=1, 2 et 3 agents (N=9 et k =2)

4.3 Résultats et analyses

Dans cette section, nous présentons les résultats de nos mesures. Nous étudions principalement la variation du temps de recherche et de la charge réseau générée en fonction du nombre d'agents, en vue de tirer des conclusions sur l'évolutivité de notre architecture. Ensuite, nous faisons une comparaison des trois algorithmes proposés.

4.3.1 Temps de recherche

Les Figures 4.6, 4.7 et 4.8 illustrent la variation du temps de recherche total en fonction du nombre d'agents pour, respectivement, les algorithmes *individualiste*, *avec blackboard* et *avec communication inter-agents*.

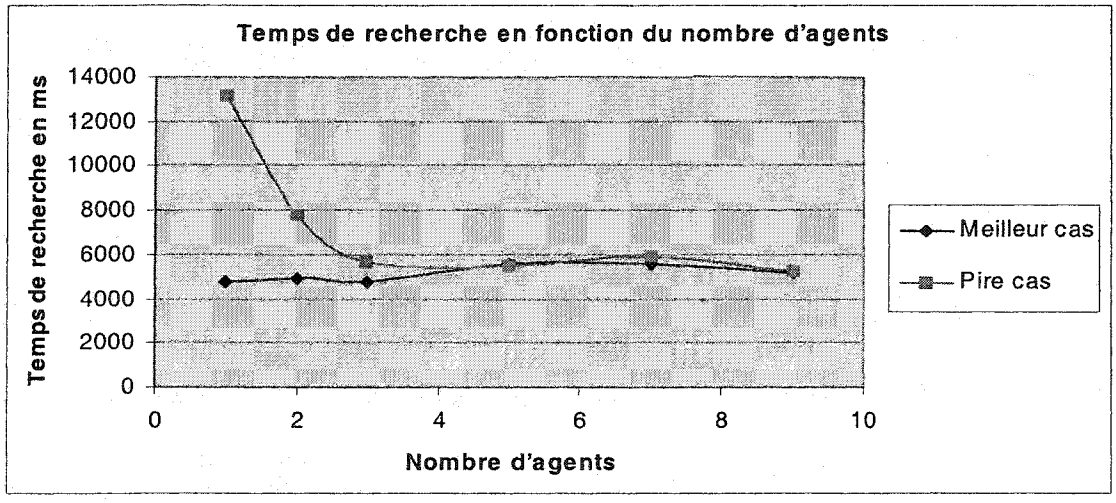


Figure 4.6 Algorithme « individualiste » : variation du temps de recherche en fonction du nombre d'agents

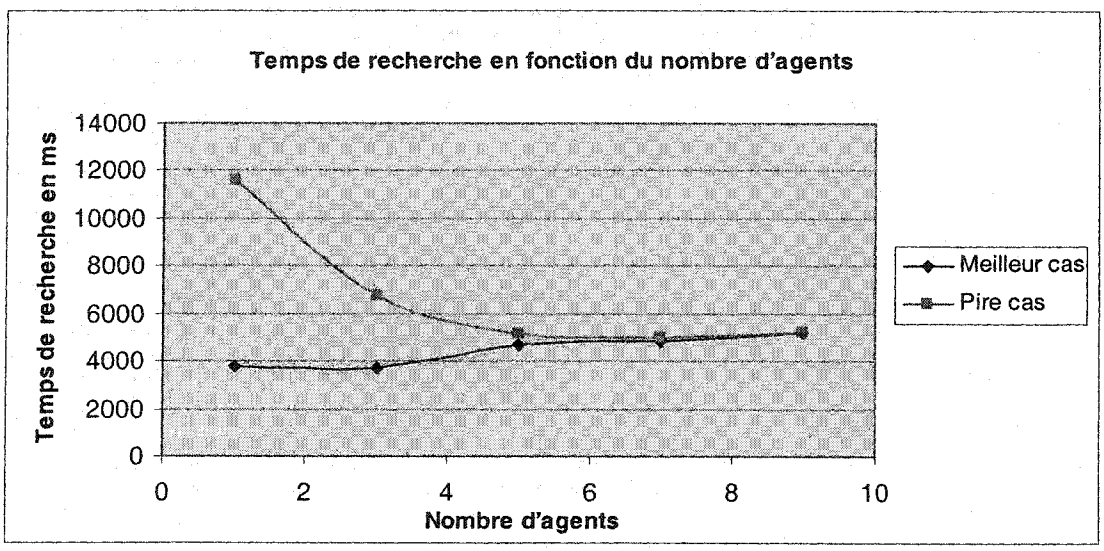


Figure 4.7 Algorithme avec *blackboard* : variation du temps de recherche en fonction du nombre d'agents

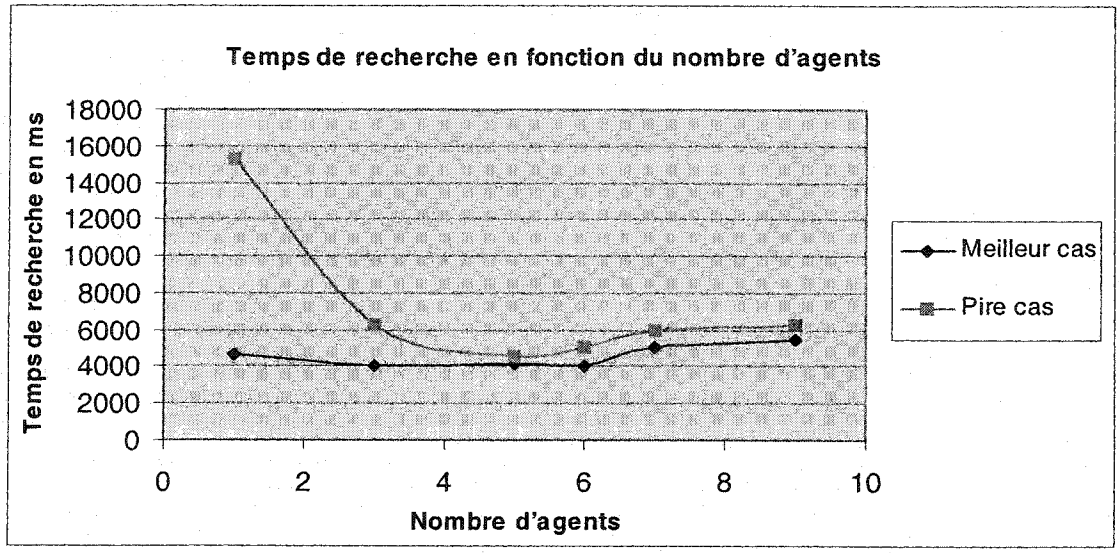


Figure 4.8 Algorithme avec communication inter-agents : variation du temps de recherche en fonction du nombre d'agents

Les Figures 4.6, 4.7 et 4.8 montrent qu'une augmentation du nombre d'agents n'entraîne pas toujours une diminution du temps de recherche. Par exemple, dans l'algorithme avec *blackboard*, quand le nombre d'agents m augmente, le temps de recherche global diminue jusqu'à $m = 5$, mais à partir de cette valeur, le temps de

recherche devient constant et tend même à augmenter dans certains cas, comme on le voit dans les meilleur et pire scénarios avec l'algorithme du *blackboard* (Figure 4.7). En fait, quand le nombre d'agents augmente, le nombre de sites à visiter par chaque agent diminue et les agents parcourent les sites en parallèle, ce qui diminue le temps de recherche global. Mais à partir d'un certain seuil (un certain nombre m d'agents, $m = 5$ dans notre exemple), les temps de traitement, d'envoi et de gestion des agents au niveau de la machine source deviennent importants et compensent négativement le gain de temps que l'augmentation du nombre d'agents est sensée apporter. Les temps de traitement des agents englobent principalement le temps de création des agents et l'implantation des mécanismes de communication inter-agents avant l'envoi de ces derniers. Ces mécanismes correspondent, en pratique, à des ponts de communication (ou *proxies*) à double sens entre les agents mobiles et l'agent stationnaire *blackboard* dans le cas de l'algorithme avec *blackboard* et d'un proxy de groupe dans le cas de l'algorithme avec communication inter-agents (Figures 3.11 et 3.12). Ces temps diffèrent selon les algorithmes et varient avec le nombre d'agents, comme le montre la Figure 4.9.

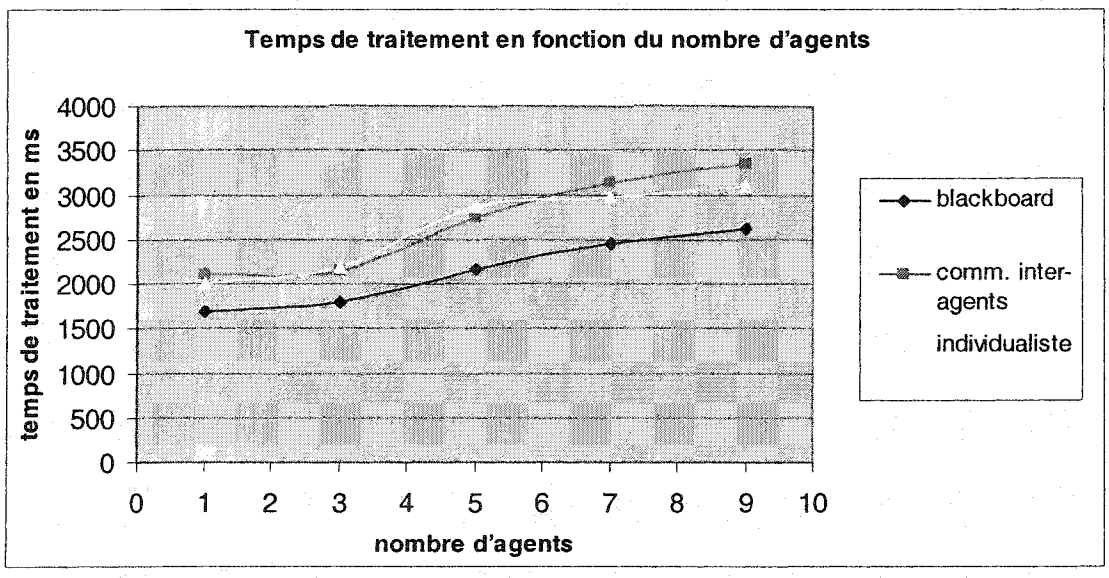


Figure 4.9 Temps de traitement en fonction du nombre d'agents pour les 3 algorithmes

La Figure 4.9 montre que les temps de traitement et d'envoi des agents dans l'algorithme avec *blackboard* et dans celui avec communication inter-agents sont supérieurs à ceux de l'algorithme individualiste à cause des temps d'implantation des mécanismes de communication. À partir d'un certain nombre d'agents (6 dans notre exemple), les temps de traitement dans la communication inter-agents dépassent ceux de la solution avec *blackboard*, ce qui va avoir une influence sur les performances globales des deux algorithmes, comme nous le verrons dans la section intitulée « comparaison des algorithmes ».

Notons aussi que, quand le nombre d'agents tend vers sa limite supérieure (nombre d'agents = nombre de sites), le « meilleur » scénarios tend vers le « pire » scénario et pour le cas extrême où le nombre d'agents est égal au nombre de sites ($m=N=9$), le « meilleur » et le « pire » scénarios se confondent car chaque agent visitera un seul site et, si un prix admissible est trouvé, il ne peut être que sur ce site.

Par ailleurs, si nous comparons la solution mono-agent (m =1) avec la solution multi-agents (m >1), nous constatons qu'à partir de 4 agents, le « pire » cas donne un gain de temps de l'ordre de 1/2.5 et de 1/3 selon l'algorithme. C'est un gain important de la solution multi-agents par rapport à celle mono-agent. Pour cette comparaison, le « meilleur » cas n'est pas très significatif car il ne met pas en évidence ce gain de temps. Ceci s'explique simplement par le fait que, dans le « meilleur » cas, les prix admissibles se trouvent sur les premiers sites visités. Donc, qu'il y ait un ou plusieurs agents, il(s) va (vont) parcourir au plus k sites, c'est-à-dire 2 dans notre exemple. Donc, la différence entre la solution mono-agent et celle multi-agents est de 1 seul site de plus, parcouru par l'agent en solution mono-agent, ce qui, à quelques erreurs expérimentales près, ne fait pas beaucoup de différence dans le temps de recherche global. La différence entre les solutions mono et multi-agents aurait été plus notable dans le « meilleur » cas si k avait été sensiblement plus grand que 2 car, dans ce cas, un agent seul parcourrait k sites avant de trouver les k prix admissibles, alors que chacun des m agents de la solution multi-agents parcourrait $E(k/m)$ sites. Ceci est une limitation pratique dans notre étude due essentiellement au nombre de machines de test qui étaient mises à notre disposition. En

effet, si nous voulons augmenter k, il faut augmenter le nombre de sites N pour ne pas retomber dans la même limitation dans le « pire » cas. Ceci dit, nous pouvons clairement estimer, en observant le « pire » cas (cas significatif), que dans un cas général intermédiaire, une solution multi-agents nous donnera certainement un meilleur temps de recherche qu'une solution mono-agent. Nous n'aurions probablement pas le même rapport de 1/2.5 ou de 1/3, mais nous aurions un gain brut significatif.

Ce que nous pouvons donc retenir à ce niveau, c'est qu'il y a un nombre optimal d'agents qui correspond à un minimum relatif dans les Figures 4.6, 4.7 et 4.8. À partir de ce point, le temps de recherche augmente ou ne diminue plus, selon les algorithmes. Aussi, une solution multi-agents mobiles donne, pour tous les algorithmes, un meilleur temps de recherche que la solution mono-agent. Ce gain en temps n'est pas sans conséquence sur la charge réseau.

4.3.2 Charge réseau

Dans chacun des 3 algorithmes, la charge réseau augmente avec le nombre d'agents, comme nous pouvons le voir sur les Figures 4.10, 4.11 et 4.12. La charge réseau résulte du déplacement des agents et des messages échangés. Quand le nombre d'agents augmente, les messages échangés sont plus nombreux, donc la charge totale augmente.

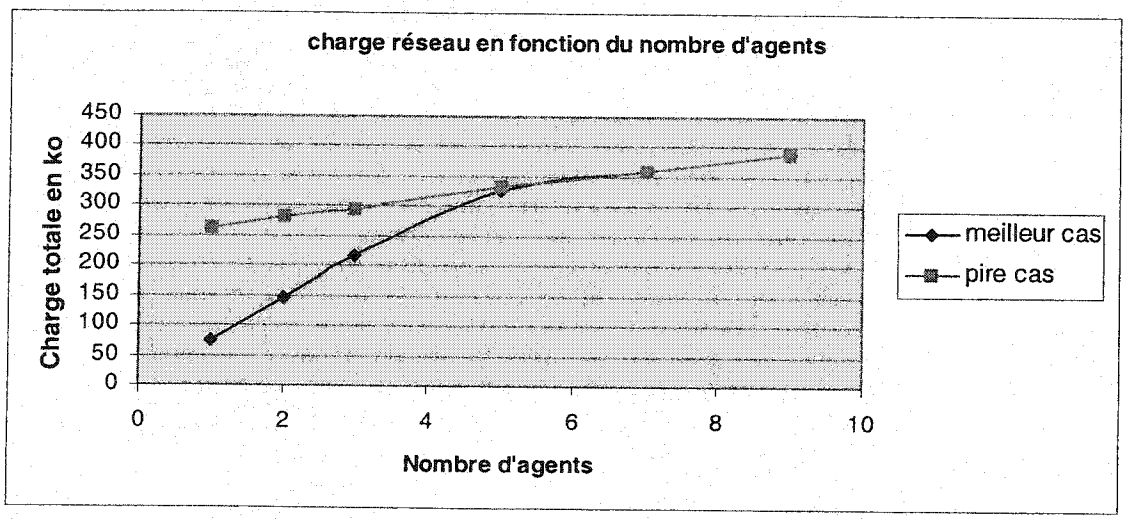


Figure 4.10 Algorithme "individualiste" : variation de la charge réseau induite en fonction du nombre d'agents

La performance de l'architecture multi-agents est définie par rapport à deux facteurs : le temps de recherche et la charge réseau induite. Par analogie au rapport qualité/prix, nous pouvons parler ici du produit (temps * charge réseau) qui permet de comparer les performances de plusieurs cas de figure ou de plusieurs algorithmes.

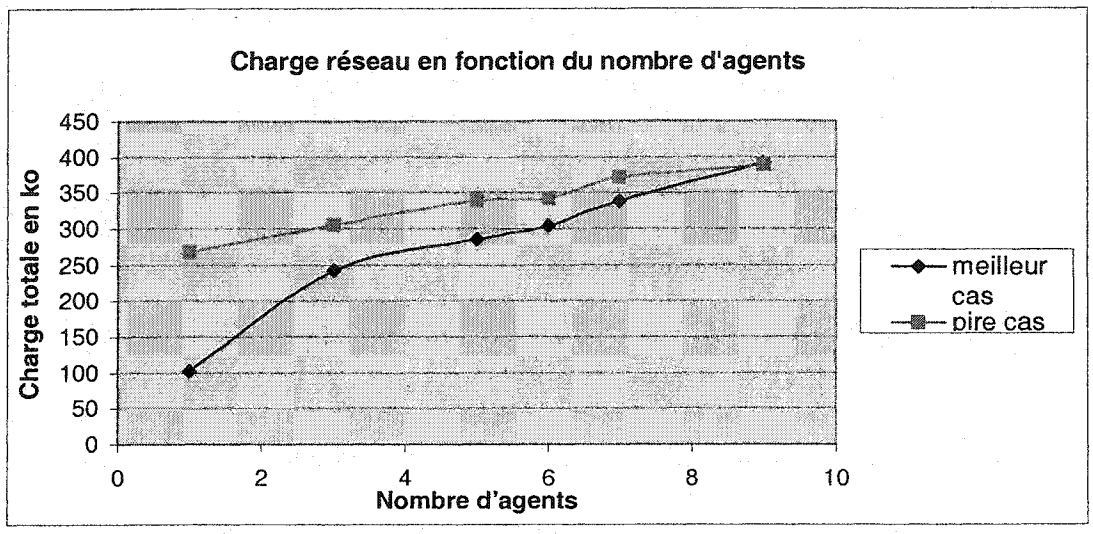


Figure 4.11 Algorithme avec *blackboard* : variation de la charge réseau induite en fonction du nombre d'agents

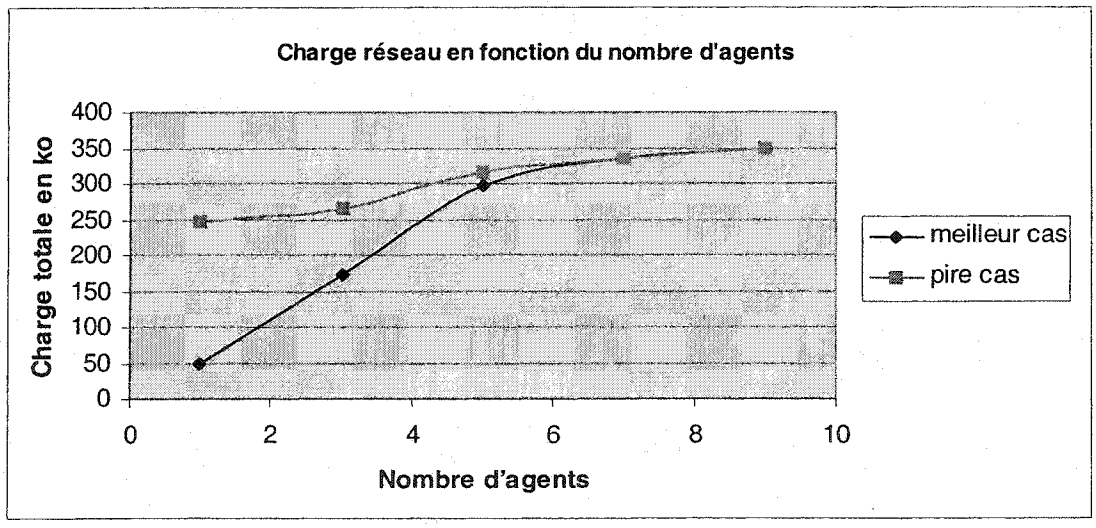


Figure 4.12 Algorithme avec communication inter-agents : variation de la charge réseau induite en fonction du nombre d'agents

Nous parlons d'un produit et non pas d'un rapport parce que les facteurs temps et charge réseau évoluent en sens inverse. En pratique, le choix d'un nombre « optimal » d'agents dépend du cahier des charges de l'utilisateur et de ses contraintes. S'il veut avoir le temps de recherche le plus court en ne se souciant pas de la charge, il doit choisir le nombre optimal d'agents minimisant le temps de recherche. Par contre, s'il a une forte contrainte sur la charge réseau, comme dans le cas du *m-commerce*, il peut choisir une solution mono-agent aux dépens du temps de recherche. Généralement, c'est un compromis qui doit être fait en choisissant un « bon » produit (temps * charge réseau). Le nombre d'agents prendra alors une valeur entre 1 et la valeur optimale pour le temps de recherche. Il est clair que la valeur choisie pour le nombre d'agents ne doit pas être supérieure à la valeur optimale pour le temps de recherche car, au-delà de cette valeur, le temps et la charge réseau augmentent tous les deux, ce qui dégrade la performance. Par souci d'évolutivité de l'architecture, il serait intéressant à ce stade d'étudier la variation **relative** du temps de recherche et de la charge réseau induite en fonction du nombre d'agents. Autrement dit, on voudrait savoir comment varient conjointement le temps et la charge quand le nombre d'agents augmente. Par exemple, si le temps de recherche diminue de moitié pour un nombre d'agents N_1, est ce que la charge diminue dans la même proportion ? Est ce que les coefficients de variation sont linéaires ou exponentiels? etc.

4.3.3 Variation relative du temps versus la charge réseau en fonction du nombre d'agents

Il s'agit ici d'analyser l'effet d'une augmentation du nombre d'agents sur les coefficients de variation relative du temps et de la charge réseau. Nous traçons alors les 3 courbes, pour le « pire » cas, de $charge_i(N) / charge_i(1)$ en fonction de N. N étant le nombre d'agents, $charge_i(N)$ la charge générée dans le « pire » cas par l'algorithme i avec N agents et $charge_i(1)$ la charge générée par l'algorithme i avec UN seul agent, dans le « pire » cas. Ce rapport exprime en fait une valeur relative des deux paramètres étudiés (temps et charge réseau) par rapport à leurs valeurs respectives dans la solution

mono-agent. Nous ne considérons que le « pire » cas, car le « meilleur » ne comporte pas de variations significatives de la charge et du temps. Le comportement des variables dans un cas général intermédiaire serait reflété par celui du « pire » cas.

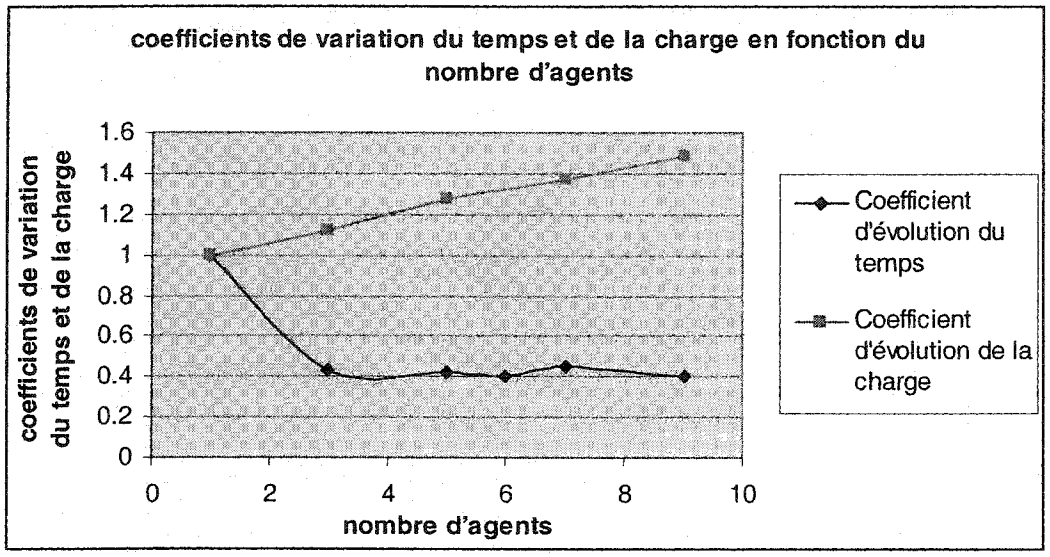


Figure 4.13 Algorithme « individualiste » : variation du temps relatif et de la charge relative en fonction du nombre d'agent

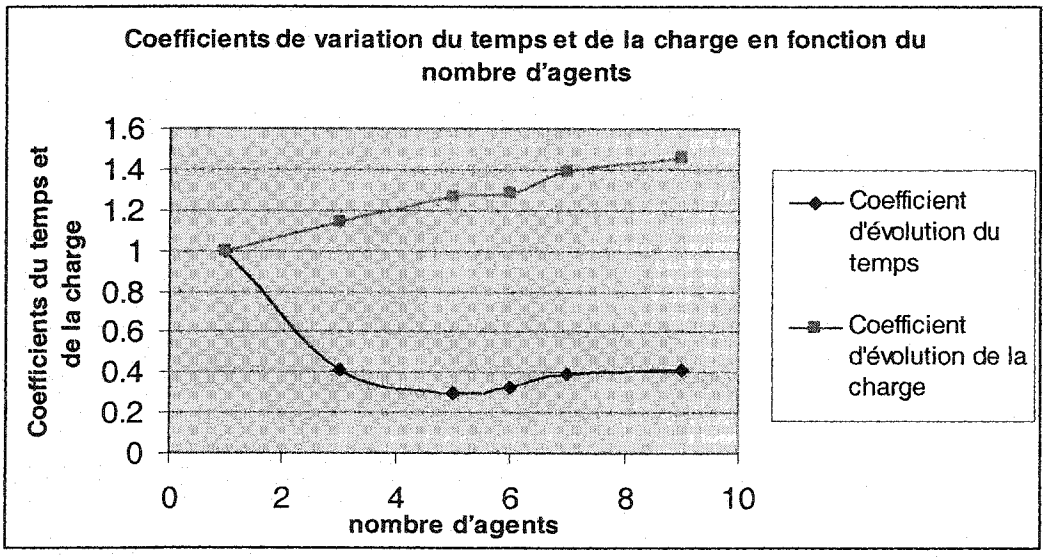


Figure 4.14 Algorithme avec *blackboard* : variation du temps relatif et de la charge relative en fonction du nombre d'agents

À partir des Figures 4.13, 4.14 et 4.15, nous constatons qu'en augmentant le nombre d'agents, le temps et la charge réseau n'évoluent pas de la même manière. Par exemple, à la Figure 4.14, nous observons qu'en augmentant le nombre d'agents de 1 à 5 (5 est le nombre optimal pour le temps de recherche), le coefficient relatif du temps passe à 0.3

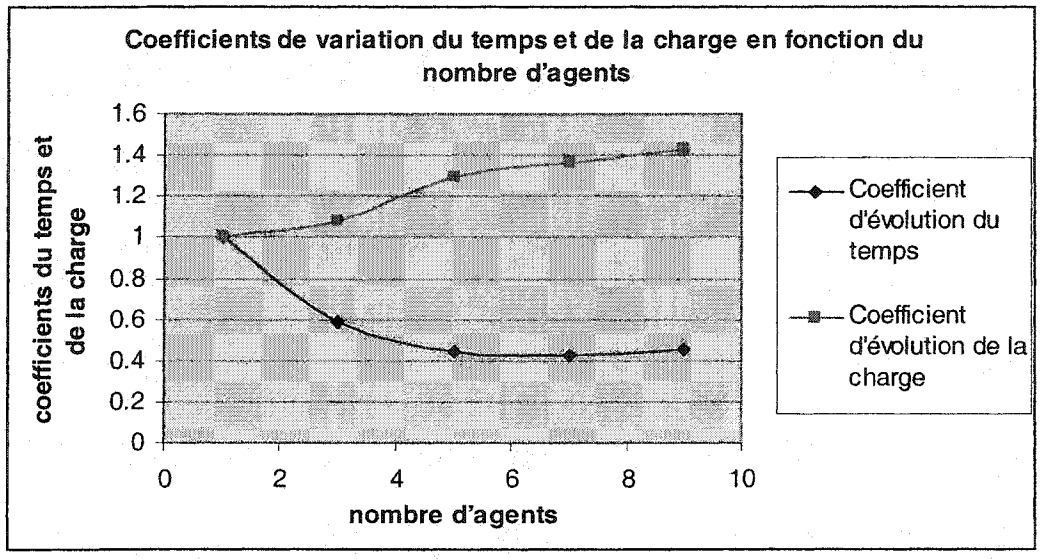


Figure 4.15 Algorithme avec communication inter-agents : variation du temps relatif et de la charge relative en fonction du nombre d'agents

environ, ce qui implique que la valeur du temps de recherche est divisée par 3 (diminue de 70% environ) alors que le coefficient de la charge réseau passe à 1.25 de sa valeur initiale, c'est à dire que la charge augmente seulement de 25% par rapport à sa valeur initiale. Pour l'algorithme « individualiste », le temps diminue à 40% de sa valeur en mono-agent (i.e. diminue de 60%) alors que la charge n'augmente que de 25%. Quant à l'algorithme avec communication inter-agents, pour le nombre optimal de 5 agents, le temps a diminué de 60% alors que la charge a augmenté de 30%. L'analyse de ces coefficients a beaucoup d'importance en pratique car elle permet de savoir jusqu'où nous pouvons aller dans un gain de performance (gain de temps par exemple) tout en limitant la perte sur d'autres paramètres de qualité (sur la charge réseau par exemple). Ceci permet aussi d'estimer le coût d'une certaine qualité de service, c'est à dire de

répondre à la question : « combien je perds en performance de A si je veux gagner tant sur la performance de B ?».

4.4 Comparaison des 3 algorithmes

Pour comparer les 3 algorithmes, nous commençons par comparer la stratégie « individualiste » avec la stratégie collaborative, ensuite, nous comparons les deux algorithmes collaboratifs entre eux.

Les Figures 4.16 et 4.17 illustrent la variation du temps de recherche en fonction du nombre d'agents pour les 3 algorithmes respectivement pour le meilleur et le pire cas.

La Figure 4.16 montre que l'algorithme individualiste donne une moins bonne performance que les deux algorithmes collaboratifs. En effet, malgré que les algorithmes collaboratifs implémentent des mécanismes de communication qui augmentent les temps de recherche, ces derniers sont plus performants du point de vue temps, car ils permettent aux agents de parcourir moins de sites. Le pire cas illustré par la Figure 4.16 n'est pas très significatif pour mettre en évidence ce résultat car, dans ce cas, les 2 prix admissibles se trouvent sur les deux derniers sites visités par l'un des agents. Les agents doivent alors visiter tous les sites de leurs sous-domaines respectifs et l'algorithme individualiste donne alors un meilleur temps de recherche car les agents ne communiquent pas. La Figure 4.16 est donc plus proche d'un scénario intermédiaire et elle nous permet d'affirmer que la collaboration améliore les performances du point de vue temps de recherche.

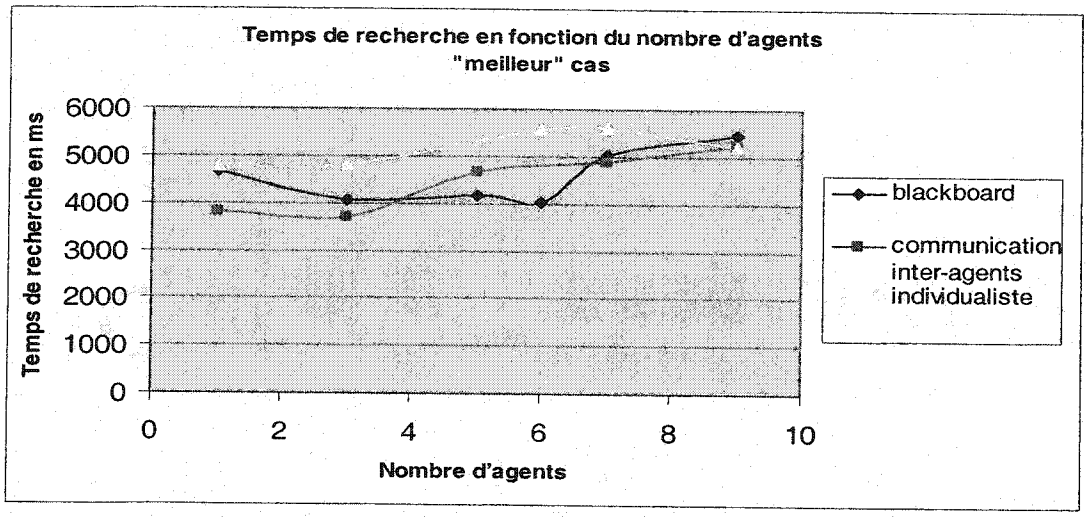


Figure 4.16 Comparaison des temps de recherche des 3 algorithmes dans le « meilleur » cas

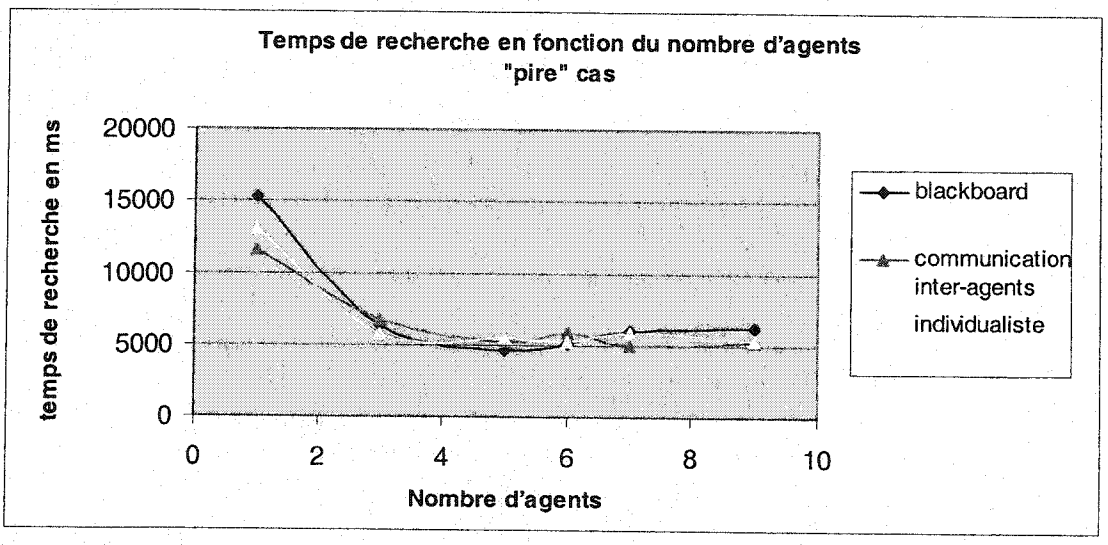


Figure 4.17 Comparaison des temps de recherche des 3 algorithmes dans le « pire » cas

Les Figures 4.16 et 4.17 montrent que pour m=1 (solution mono-agent), les temps de recherche varient selon l'algorithme et ce, quel que soit l'algorithme utilisé. Ceci était assez prévisible puisque, dans une solution mono-agent, l'agent fait exactement le même parcours de sites quelque soit l'algorithme. Il n'y a donc aucune optimisation de parcours puisqu'il n'y pas de collaboration. Il reste que, dans la solution *blackoard* par

exemple, l'agent va quand même utiliser son mécanisme de communication avec le *blackboard*. À chaque prix admissible trouvé, il enverra un message au *blackboard* pour l'informer, et au dernier prix admissible trouvé, l'agent ne pourra terminer sa recherche qu'après avoir reçu l'ordre du *blackboard*. C'est ce mécanisme de communication qui provoque un temps de recherche supérieur avec l'algorithme du *blackboard* par rapport aux autres algorithmes. La communication inter-agents, par contre, n'est pas pénalisée puisque l'agent ne communique avec aucun autre agent.

Cette même explication s'applique aussi à la différence de charge réseau entre les 3 algorithmes dans la solution mono-agent, comme le montrent les Figures 4.18 et 4.19.

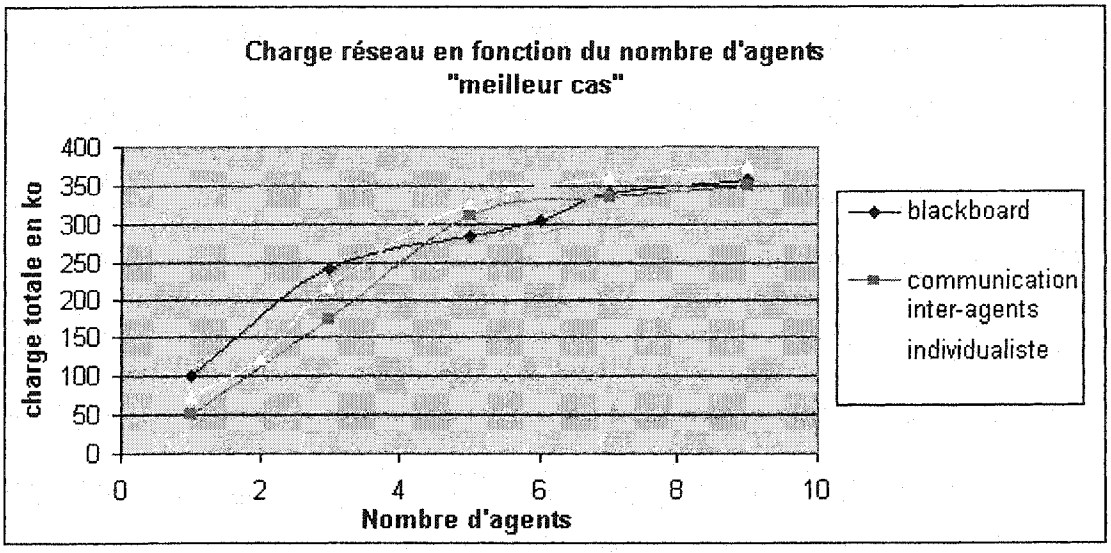


Figure 4.18 Comparaison des charges réseau des 3 algorithmes dans le « meilleur » cas

Ce sont toujours les messages échangés entre l'agent et le *blackboard* qui font que la charge pour l'algorithme du *blackboard* soit plus élevée que les autres algorithmes. Nous constatons par ailleurs que dans le meilleur cas, et comme l'illustre la Figure 4.18, l'algorithme « individualiste » induit une charge réseau plus

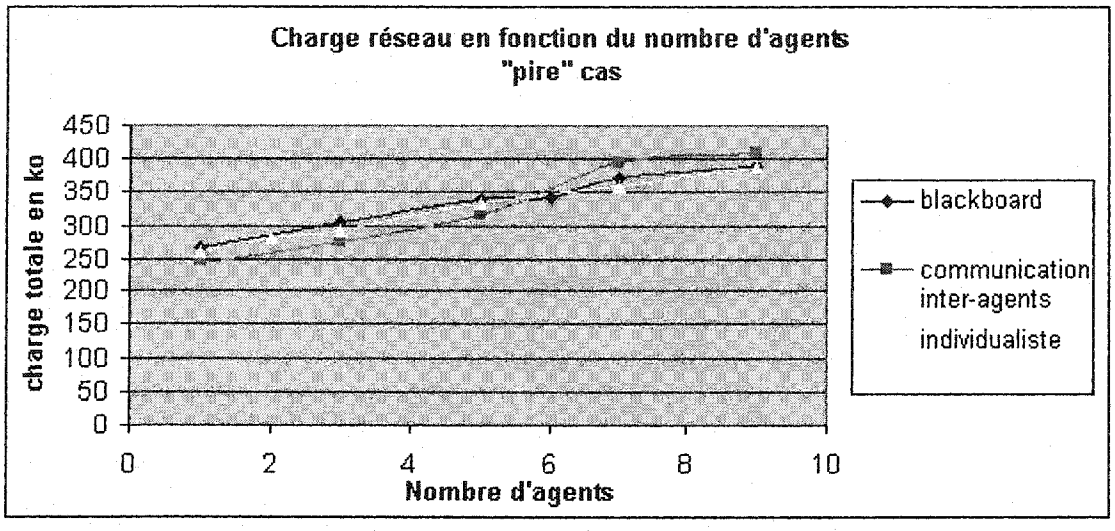


Figure 4.19 Comparaison des charges réseau des 3 algorithmes dans le « pire » cas

élevée que celui avec communication inter-agents. Il donne aussi une charge plus élevée que l'algorithme avec *blackboard* à partir d'une certain nombre d'agents (4 dans notre exemple), car la collaboration permet aux agents de parcourir moins de sites qu'avec l'algorithme individualiste et donc de générer moins de charge réseau.

Dans le pire cas, comme l'illustre la Figure 4.19, la différence entre les charges réseau générées par les trois algorithmes n'est pas très notable car les agents parcourent tous les sites de leurs sous-domaines respectifs, et donc la plus grande partie de la charge générée résulte du déplacement des agents, ce qui ne permet pas de mettre en évidence la différence de charge due à la communication. À partir des constatations sur les Figures 4.18 et 4.19, nous pouvons estimer que, dans un scénario intermédiaire, les algorithmes collaboratifs donneront en moyenne de meilleures performances que l'algorithme individualiste.

À ce niveau, nous pouvons conclure que la collaboration donne, en moyenne, de meilleures performances en temps de recherche et charge réseau par rapport à une stratégie individualiste. Comparons maintenant les deux algorithmes collaboratifs.

D'après la Figure 4.19 qui illustre le meilleur cas, l'algorithme avec communication inter-agents donne un meilleur temps de recherche que celui avec *blackboard,* jusqu'à une certaine valeur seuil m_s du nombre d'agents (m_s= 4 dans notre cas) à partir de laquelle ce dernier commence à donner de meilleures performances. Sur les Figures 4.7 et 4.8, nous constatons que les deux algorithmes donnent un temps de recherche optimal pour un nombre d'agents environ égal à 5. Autour de cette valeur, l'algorithme du *blackboard* donne un meilleur temps de recherche moyen que l'algorithme utilisant la communication inter-agents. Notons aussi qu'il est inutile de considérer un nombre d'agents supérieur au nombre optimal pour les deux algorithmes car, au-delà de ce nombre, le temps de recherche augmente et la charge réseau aussi donc les performances empirent. En pratique, il faut éviter de dépasser le nombre optimal d'agents. Nous comparerons donc les algorithmes pour des valeurs du nombre d'agents inférieures ou égales au nombre optimal.

Du point de vue charge réseau, nous constatons d'après les Figures 4.18 et 4.19 que l'algorithme avec communication inter-agents donne une meilleure performance que celui avec *blackboard* jusqu'à un certain nombre d'agents, à partir duquel il commence à le dépasser en charge réseau générée. Cette valeur est égale à 4 dans le meilleur cas et 6 dans le pire cas. Ceci s'explique par le nombre de messages échangés entre les agents : à chaque fois qu'un agent trouve un prix admissible, il envoie un message *multicast* aux *(m-1)* autres agents, donc la charge réseau totale est fonction de $k * (m-1)$. Pour l'algorithme du *blackboard,* si nous faisons l'hypothèse que les temps d'envoi d'un message d'un agent mobile vers le *blackboard* et du *blackboard* vers l'agent mobile sont égaux, ce temps serait fonction de $(k+m)$. Donc, théoriquement, la charge réseau engendrée par la communication inter-agents dépasse celle engendrée par la communication avec *blackboard* quand:

$$k * (m-1) > (k+m)$$

c'est à dire quand:

$$m > 2.k / (k-1)$$

Dans notre cas où $k = 2$, ceci revient à: $m = 4$. Dans le meilleur cas, nous pouvons bien vérifier cette hypothèse puisque, sur la Figure 4.18, la charge engendrée par la communication inter-agents dépasse celle engendrée par la communication par *blackboard* à partir de $m = 4$ agents. Dans le pire cas, comme nous le voyons sur la Figure 4.19, c'est à partir de $m = 6$ que le dépassement a lieu. L'erreur provient, entre autres, de la différence entre les poids des agents. Cette erreur est accentuée dans le scénario du pire cas car les agents parcourent tous les sites de leurs sous-domaines. En pratique, si nous choisissons le nombre optimal d'agents qui nous donne le temps de recherche minimal, nous devons estimer la charge produite par chacun des algorithmes pour pouvoir décider lequel choisir. Une estimation théorique est donnée par (3.8) moyennant une estimation des poids des messages échangés. En effet, ici aussi, le choix dépend des critères et des contraintes pratiques : si la bande passante n'est pas limitée, nous nous soucierons peu de la charge réseau induite et nous choisirons l'algorithme du *blackboard* avec un nombre optimal d'agents. Si nous voulons limiter la charge réseau, il faut utiliser un « petit » nombre d'agents. Une solution mono-agent est possible mais sa performance en temps de recherche est médiocre. Pour les petites valeurs du nombre d'agents, la communication inter-agents a les meilleures performances puisqu'elle produit moins de charge et un plus faible temps de recherche moyen. Le nombre d'agents choisi ne doit évidemment pas dépasser n_s, n_s étant le nombre à partir duquel la communication inter-agents dépasse en charge générée la communication avec *blackboard*. Par exemple, pour $m = 3$, et d'après les Figures 4.16 et 4.17, les temps de recherche dans le « meilleur cas » pour les deux algorithmes avec *blackboard* et avec communication inter-agents sont respectivement de 4047 et 3686 ms. Dans le « pire cas », ils sont respectivement de 15304 et 11601 ms. Quant à la charge réseau induite, elle est respectivement de 242 et 173 ko dans le « meilleur cas » et de 305.6 et 265.6 ko dans le « pire cas ». Nous voyons donc que, pour $m = 3$ agents, la communication inter-agents est plus performante.

4.5 Synthèse des résultats

Les mesures de performances et observations des différents algorithmes proposés nous permettent de tirer les conclusions suivantes:

- Une augmentation du nombre d'agents dans une architecture multi-agents mobiles n'améliore pas forcément le temps de recherche, bien que cette augmentation permet de réduire le temps de recherche individuel de chaque agent et par suite la tâche globale de recherche. Ceci est principalement dû aux temps de traitement des agents et aux mécanismes de communication.
- Une solution multi-agents donne, dans tous les cas, de meilleurs temps de recherche qu'une solution mono-agent.
- Bien qu'une stratégie individualiste non collaborative offre l'avantage d'alléger l'agent (non porteur de mécanismes de communication tels que les *proxies*) et d'éviter les temps de communication, la collaboration améliore toujours les performances d'une architecture multi-agents. La collaboration entre les agents permet d'éviter des visites de sites inutiles et de limiter ainsi le temps et la charge réseau.
- Il y a un nombre optimal d'agents qui donne le plus petit temps de recherche, et à partir duquel les performances de l'architecture empirent. Ce nombre est déterminé par les mesures de performances, car il dépend de plusieurs facteurs comme l'encombrement du réseau et les temps de traitement sur la machine source.
- Si le temps de recherche est le critère le plus important à minimiser quelle que soit la charge réseau induite, l'algorithme avec *blackboard* est recommandé, avec un nombre d'agents optimal.
- S'il est important de limiter la charge réseau induite en ayant un « bon » compromis (temps de recherche / charge réseau), c'est l'algorithme avec communication inter-agents qui est recommandé, car, pour des petites valeurs du nombre d'agents, il donne de meilleures performances (en temps et en charge réseau) que l'algorithme du *blackboard*. Cependant, il donnera un temps de

recherche plus élevé que celui donné par l'algorithme avec *blackboard* avec un nombre d'agents optimal.

CHAPITRE V
CONCLUSION

Le commerce électronique fait partie des applications les plus populaires sur Internet. À coté des simples opérations d'achat et de vente, plusieurs services sont offerts aux clients, comme les enchères en ligne, la recherche de produits spécifiques sur le réseau ou encore la comparaison des prix d'un certain produit entre plusieurs marchands. Avec l'avènement de l'Internet mobile et la possibilité pour les terminaux portatifs sans fil comme les PDA (Personal Digital Assistant) ou les PocketPCs de se connecter au réseau, le commerce électronique a du s'adapter à ce nouveau mode de connexion et on parle alors de commerce mobile (*m-commerce*). Cependant, l'architecture classique du client-serveur s'adapte mal à ce contexte. En effet, en mode client-serveur, de grands volumes de données transitent souvent entre le client et le serveur et des interactions continues ont lieu entre ces derniers, ce qui produit des délais d'attente élevés pour le client. Ces délais sont plus accentués en mode d'accès sans fil puisque les données échangées doivent transiter par une bande passante limitée.

5.1 Synthèse des travaux

Dans ce travail, nous avons présenté le nouveau paradigme d'agents mobiles où des entités de code exécutable (les agents mobiles) sont capables de migrer entre les machines du réseau pour s'y exécuter. Les agents sont autonomes et munis de l'intelligence nécessaire pour pouvoir représenter le profil de l'utilisateur et prendre certaines décisions pour son compte. Les agents permettent de déplacer le code vers les données et d'éviter que les interactions et les données qui transitent entre le client et le serveur ne passent par le réseau, ce qui procure un gain de temps et de bande passante.

Pour mettre en pratique ce paradigme, nous nous sommes intéressés à une application simple de commerce électronique: la recherche de prix admissibles pour un produit spécifique sur le Net. Les agents parcourent alors des sites marchands répartis sur le

réseau et y recherchent des produits spécifiques à des prix inférieurs à un certain seuil. Nous avons commencé par prouver théoriquement qu'une architecture multi-agents où plusieurs agents mobiles effectuent ensemble la recherche de prix est plus efficace qu'une architecture à un seul agent (mono-agent). Ensuite, nous avons présenté trois algorithmes pour faire coopérer les agents durant leur recherche de prix. Les algorithmes proposés sont :

- Un algorithme dit « individualiste », où les agents ne communiquent pas et aucun mécanisme de collaboration n'est mis en place.
- Un algorithme avec tableau noir, ou *blackboard*, où les agents utilisent un espace commun de consultation et de partage de l'information, qui est le *blackboard*. Ce dernier est un agent stationnaire qui supervise et contrôle tous les agents mobiles chercheurs de prix. Chaque fois qu'un agent trouve un prix sur un site, il informe le *blackboard*. Dès que l'information globale est trouvée, le *blackboard* rappelle tous les agents mobiles.
- Un algorithme basé sur la communication directe entre les agents mobiles : les agents partagent à chaque instant la même information et chaque fois que l'un d'eux trouve un prix admissible, il en informe tous les autres.

Nous avons alors présenté les avantages et les inconvénients de chaque algorithme et nous les avons implémenté pour comparer leurs performances. Les trois algorithmes ont été implémentés sur une architecture où plusieurs agents mobiles visitent des galeries marchandes réparties sur le réseau et y recherchent le produit désiré. Les galeries exposent leurs produits sous forme de documents XML. Ce standard a été choisi pour son interopérabilité, sa simplicité d'implémentation et sa bonne adaptation au commerce électronique.

Pour chacun des algorithmes proposés, les tests de performance comprennent des mesures du temps moyen de recherche et de la charge réseau induite. Généralement, en augmentant le nombre d'agents, le temps moyen de recherche diminue mais la charge réseau induite augmente. Il s'agissait donc de trouver un « bon » compromis entre les délais de recherche de l'information et la charge réseau résultante. Nous nous sommes

particulièrement intéressés à l'évolution de ces deux paramètres en fonction du nombre d'agents en vue d'étudier l'évolutivité de notre architecture avec chacun des algorithmes de collaboration proposés. Pour notre étude de performance, nous avons procédé d'une manière empirique, faute d'estimation rigoureuse sur certains paramètres probabilistes. Nous avons établi des scénarios par rapport à la position de l'information cherchée (les prix admissibles) sur les sites visités par les agents. Les scénarios étudiés représentent deux cas de figure extrêmes qui donnent respectivement le plus petit temps et le plus grand temps de recherche. Les mesures du temps et de la charge réseau de chaque algorithme nous ont permis de tirer les conclusions suivantes:

- Le temps de recherche moyen a une valeur minimale en fonction du nombre d'agents. Ce minimum absolu est obtenu pour un nombre optimal d'agents différent du nombre maximal d'agents. Au-delà de ce nombre optimal, les performances de l'algorithmes se dégradent.
- Une solution multi-agents donne, dans tous les cas, de meilleurs temps de recherche qu'une solution mono-agent.
- Dans tous les cas, la collaboration améliore les performances par rapport à une stratégie sans collaboration.
- L'algorithme avec *blackboard* est recommandé dans les cas où le temps de recherche est le seul critère important devant être minimisé et où il n'y a pas de limitation sur la bande passante. En effet, avec un nombre optimal d'agents, cet algorithme donne le meilleur temps de recherche mais pas la charge réseau la plus petite.
- Dans le cas où un « bon » compromis temps/charge réseau induite est recherché, tout en limitant autant que possible la charge réseau induite, lequel cas est le plus adapté à la réalité, c'est l'algorithme avec communication inter-agents qui est recommandé. En effet, cet algorithme donne de meilleures performances (en temps de recherche et en charge réseau) que les autres algorithmes, pour de petites valeurs du nombre d'agents. Cependant, le temps de recherche est plus grand que celui donné par l'algorithme avec *blackboard* avec un nombre optimal d'agents.

En pratique, nous pouvons faire face à différents cahiers de charges, chacun ayant ses exigences et ses limitations par rapport aux performances recherchées. Une bonne qualité de service consiste en un temps de recherche optimal mais doit payer le prix de la charge réseau induite.

5.2 Limitations des travaux

Parmi les limitations de nos travaux, nous pouvons mentionner l'absence d'un mécanisme de sécurité. En effet, dans un contexte de commerce électronique où la compétition entre marchands est un facteur important, il est essentiel de protéger les données des agents. Les mécanisme de sécurité dépendent des stratégies de collaboration employées et peuvent influencer les performances du système multi-agents. Il serait donc plus rigoureux de tester les performances avec un mécanisme de sécurité implanté.

Une deuxième limitation réside dans le fait que nous n'avons pas comparé les mécanismes de collaboration proposés avec ceux déjà proposés dans d'autres travaux de recherche.

Par ailleurs, à cause du caractère empirique de nos mesures, faute d'estimations exactes sur certains paramètres comme les temps d'exécution ou les probabilités de succès, nos résultats restent relativement restreints à un réseau de petite taille. Des résultats très intéressants peuvent être tirés si les mesures sont faites sur un réseau de plus grande envergure et de plus grande complexité, comme Internet.

5.3 Orientations de recherche futures

Ce travail nous a permis de mettre en place une architecture à plusieurs agents mobiles collaborant et de proposer un algorithme de collaboration pour chaque ensemble de situations. Les mesures ont été réalisées sur un réseau filaire dont le taux d'erreur est pratiquement nul. Il serait intéressant d'étudier les performances de l'architecture proposée dans un environnement sans fil bruité. Le bruit sur le canal augmentera le taux

d'erreur et donc le taux de retransmission des paquets. Par conséquent, le temps de recherche ainsi que la bande passante seront affectées négativement.

Une des orientations futures intéressantes consiste en une étude de l'évolution de la performance des algorithmes proposés en fonction du bruit de canal (ou de la probabilité d'erreur). L'environnement sans fil est en effet le domaine d'application où les agents sont censés marquer la différence avec le mode client serveur. Il serait donc aussi intéressant de comparer les stratégies multi-agents proposées dans ce mémoire avec un mode client serveur classique.

Une autre problématique qui pourrait faire partie des travaux futurs est celle de la sécurité. En effet, la sécurité des agents mobiles est un élément essentiel à la viabilité commerciale des agents mobiles. Dans le cas du commerce électronique, il est encore plus important de s'assurer que l'agent n'a pas été corrompu par une agence ou par un autre agent, car un agent mobile peut transporter des informations personnelles sur le client, qui ne doivent pas être dévoilées. La sécurité de la plate-forme qui accueille l'agent est aussi un point d'importance capitale car un agent mal intentionné peut aller sur une agence qui représente un site marchand et provoquer une attaque de type « déni de service » en opérant des requêtes successives et ininterrompues sur le site de manière à ralentir ou bloquer son fonctionnement.

La sécurité constitue donc un axe de recherche primordial pour l'applicabilité des agents mobiles au commerce électronique.

BIBLIOGRAPHIE

Anthony P., Hall W., Dang V.D et Jennings N., «Autonomous agents for participating in multiple online auctions». *In Proc. of the IJCAI Workshop on eBusiness and the Intelligent Web, Seattle WA, USA,* Juillet 2001, pp 54-64.

Barbuceanu M et Fox M.S., «Coordinating multiple agents in the supply chain», *Proceedings of the Fifth Workshops on Enabling Technology for Collaborative Enterprises (WET ICE'96), IEEE Computer Society Press,* 1996, pp. 134-141.

M. Benyoucef, K. Keller, S. Lamoureux, J. Robert, «Towards a generic e-negotiation platform», *In Sixth International Conference on Re-Technologies for Information Systems*, Austrian Computer Society, Zurich, Switzerland, 2000, pp. 95-109.

Benyoucef M., Alj H., Levy K. and Keller R.K. «A Rule-driven Approach for Defining the Behavior of Negotiating Software Agents», *In Proceedings of the Fourth International Conference on Distributed Communities on the Web,* Sydney, Australie, Avril 2002

Bichler M., Beam C. et Segev A., «An electronic broker for business-to-business electronic commerce on the Internet ». *Int. Journal of Cooperative Information Systems,* Vol. 7, 1998, pp. 315-331.

Brugali D, Menga G., Galarraga S., «Inter-Company supply chain, integration via mobile agents», *Globalization of Manufacturing in the Digital Communications Era of the 21 st Century: Innovation, Agility and the Virtual Enterprise*, Kluwer Academic Publisher, 1998, pp. 43-54.

Cabri G., Leonardi L.et Zambonelli F. «Reactive Tuple Spaces for Mobile Agent Coordination». *In Proceedings of the 2nd International Workshop on Mobile Agents (MA'98)*, No 1477 in LNCS, 1998, pp 237-248.

Chan H.C.B., Ho I.S.K., Lee, R.S.T., «Design and implementation of a mobile agent-based auction system», *PACRIM* 2001: *IEEE Pacific Rim Conference on Communications, Computers and signal Processing*, 2001, Vol. 2, pp. 740 -743.

Chavez A., Maes P., «Kasbah: An Agent Marketplace for Buying and Selling Goods», *First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'96)*, 1996, pp 75-90.

Collins J., Bilot C., Jini M., Mobasher B., «Decision processes in decision-based automated contracting», *IEEE Internet computing*, Mars-Avril 2001, pp. 61-72.

Dasgupta P., Narasimhan N., Moser L.E., Melliar-Smith P.M., « MAgNET: mobile agents for networked electronic trading», *IEEE Transactions on Knowledge and Data Engineering,* Vol. 11, No. 4 , Juillet - Août 1999, pp. 509 –525.

Duda A. et Perret S., «Mobile agent architecture for nomadic computing», *In International Conference on Computer Communications*, Cannes, 1997.

Falchuk B., Karmouch A, «A mobile agent prototype for autonomous multimedia information access, interaction, and retrieval», *Proc. Multimedia Modeling'97*, Novembre 1997.

Glitho, R.H., Olougouna, E, Pierre, S., «Mobile agents and their use for information retrieval: a brief overview and an elaborate case study», *IEEE Network* , Vol. 16, No 1, Janvier-Février 2002, pp. 34 – 41.

Glushko R.J., Tenenbaum J.M. and Meltzer B. «An XML framework for agent-based e-commerce», *communications of the ACM,* Vol. 42, No. 3, Mars 1999, pp. 106-114.

Gray R., Kotz D., Cybenko G., Rus D., «Mobile agents: Motivations and state-of-the-art systems», *Technical report*, Darmouth College, Avril 2000.

Green S., Hurst L., Nangle B., Cunningham P., Somers F., Evans R., «Software Agents: A review», *Technical report TCD-CS-1997-06*, Trinity College Dublin, Mai 1997.

Guttman R.H., Maes P., «Agent-mediated Integrative Negotiation for Retail Electronic Commerce», *Dans Noriega P. et Sierra C, editors, «Agent Mediated Electronic Commerce»*, LNAI 1571, Springer, 1998, pp 70-90.

Hu W., Benedicenti, L., Paranjape, R., «Mobile agent network management system performance study in frame relay network», *IEEE CCECE 2002. Canadian Conference on Electrical and Computer Engineering*, Vol. 3, 2002, pp. 1499-1504.

Ito T., Fukuta N., Shintani T., Sycara, K., «BiddingBot: a multiagent support system for cooperative bidding in multiple auctions», *Proceedings Fourth International Conference on MultiAgent Systems*, 2000, pp. 399-400.

Jain, R., Anjum, F., «Mobile agents for personalized information retrieval: when are they a good idea ?», *Wireless Communications and Networking Confernce*, 2000, *WCNC, 2000, IEEE*, Vol. 1, 2000, pp. 242-245.

Kay J., Etzl J., Rao G. et Thies J., «The ATL postmaster: A system for agent collaboration and information dissemination», *International Conference on autonomous agents*, 1998, pp 338-342.

Moizumi K., «The mobile agent planning problem», *PhD thesis, Thayer School of Engineering, Dartmouth College*, Novembre 1998.

Perdikeas M.K., Chatzipapadopoulos, F.G., Venieris, I.S, «An evaluation study of mobile agent technology: standardization, implementation and evolution», *IEEE International Conference on Multimedia Computing and Systems*, Vol. 2, 1999, pp. 287 -291.

Perret S., Duda A., « Implementation of MAP : A system for Mobile Assistant Programming », *Proceedings IEEE International Conference on Parallel and Distributed Systems*, 1996.

Pierre S., Kacan C., Probst W., « An agent-based approach for integrating user profile into a knowledge management process », *Elsevier science, Knowledge based systems*, Vol. 13, 2000, pp 307-314.

Qi L., Yu L., « Mobile agent-based security model for distributed system », *Systems, IEEE International Conference on Man and Cybernetics*, 2001, Vol. 3, 2001, pp. 1754 -1759.

Sandholm T., « An implementation of the *contract net* protocol based on marginal cost calculations ». *In Eleventh National Conference on Articial Intelligence (AAAI-93)*, 1993, pp. 256-262.

Swaminathan J. M., Smith S.F, Sadeh N. M., « Modeling Supply Chain Dynamics: A Multiagent Approach », *Decision Sciences journal*, Vol. 29, No. 3, 1998, pp 607-632.

Tsvetovatyy, M., Mobasher B., Gini M. et Wieckowski Z., «MAGMA: An Agent-Based Virtual Market for Electronic Commerce», *Applied Artificial Intelligence*, Vol. 11, No. 6, Septembre 1997, pp. 501-523.

Van Thanh, D., «Using Mobile Agents in telecommunications», *Proceedings of the 12th International Workshop on Database and Expert Systems Applications*, 2001, pp. 685-688.

Zhao L., Ng W.K, Lim E.P. « Cooperative multi-attribute bilateral online negotiation for e-commerce», *Database and Expert Systems Applications, 2001. Proceedings of the 12th International Workshop on*, 2001, pp. 703 –707.

Zlotkin G., Rosenschein J.S. «Negotiation and Task Sharing Among Autonomous Agents in Cooperative Domains », *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence,*1989, pp 912-917.

RÉFÉRENCES INTERNET

Kong H.P, « The travel industry makeover », *Intisoft e-business technologist*, 2000. (http://www.intisoft.com/index.phtml?scp=resrc_xmlnews2&artuid=286).

Le journal du Net, « Chiffres clés », 2002. (http://www.journaldunet.com/chiffres-cles.shtml).

MobileCity, «Understanding the fundamentals of m-commerce», *White Paper*, Juin 2000. (http://www.pmn.co.uk/corporate/mbusiness/002mcommerce.pdf)

NUA Internet Surveys, «Surveys», Février 2002. http://www.nua.ie/surveys/

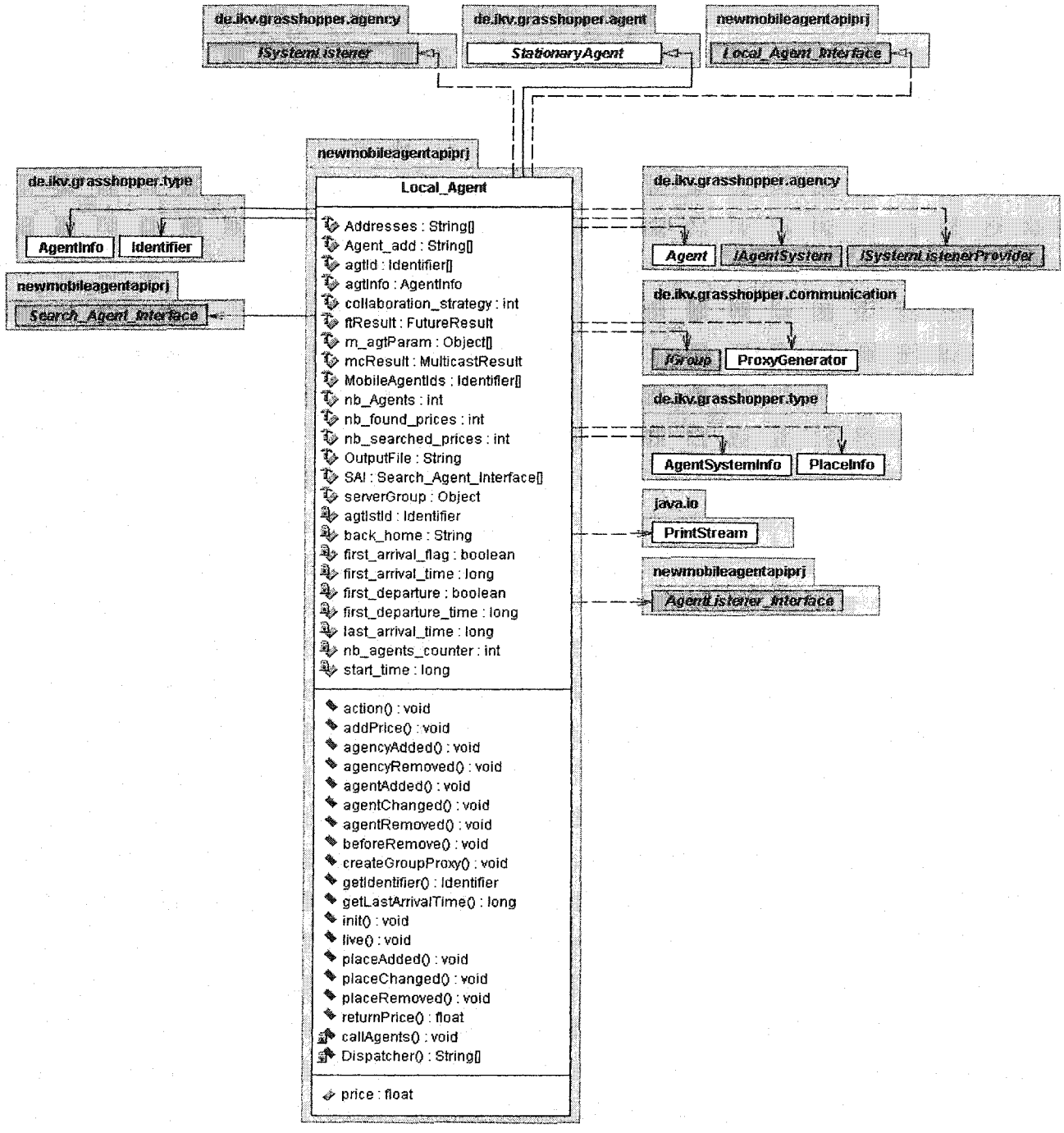
Sol S., « Introduction to XML for Web developers », *WDVL*, Mai 1999. http://wdvl.internet.com/Authoring/Languages/XML/Tutorials/Intro/toc.html

Plate-forme pour agents mobiles **Grasshopper** : www.grasshopper.de

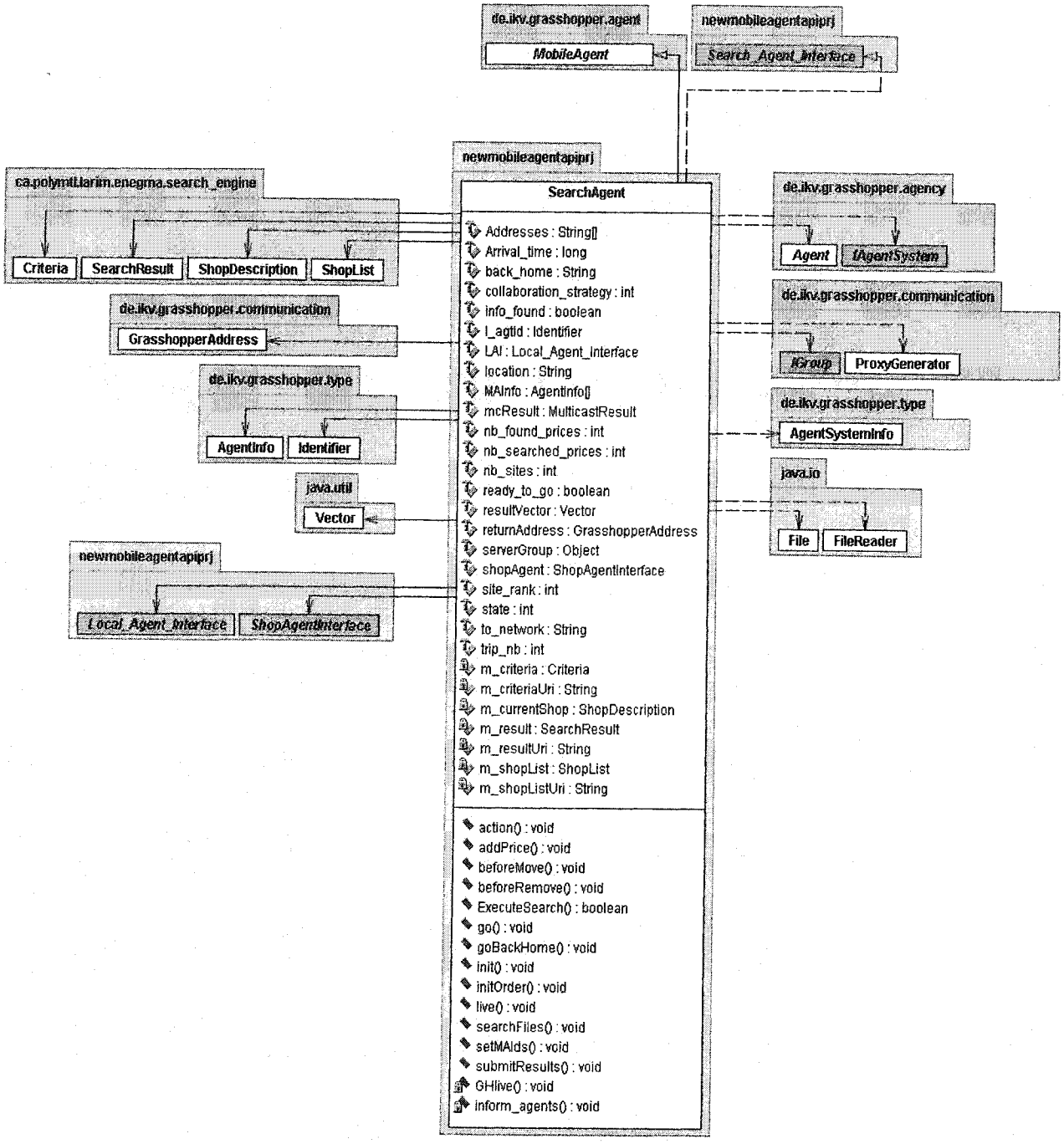
ebXML : www.ebxml.org

ANNEXE I
DIAGRAMMES DES CLASSES EN UML

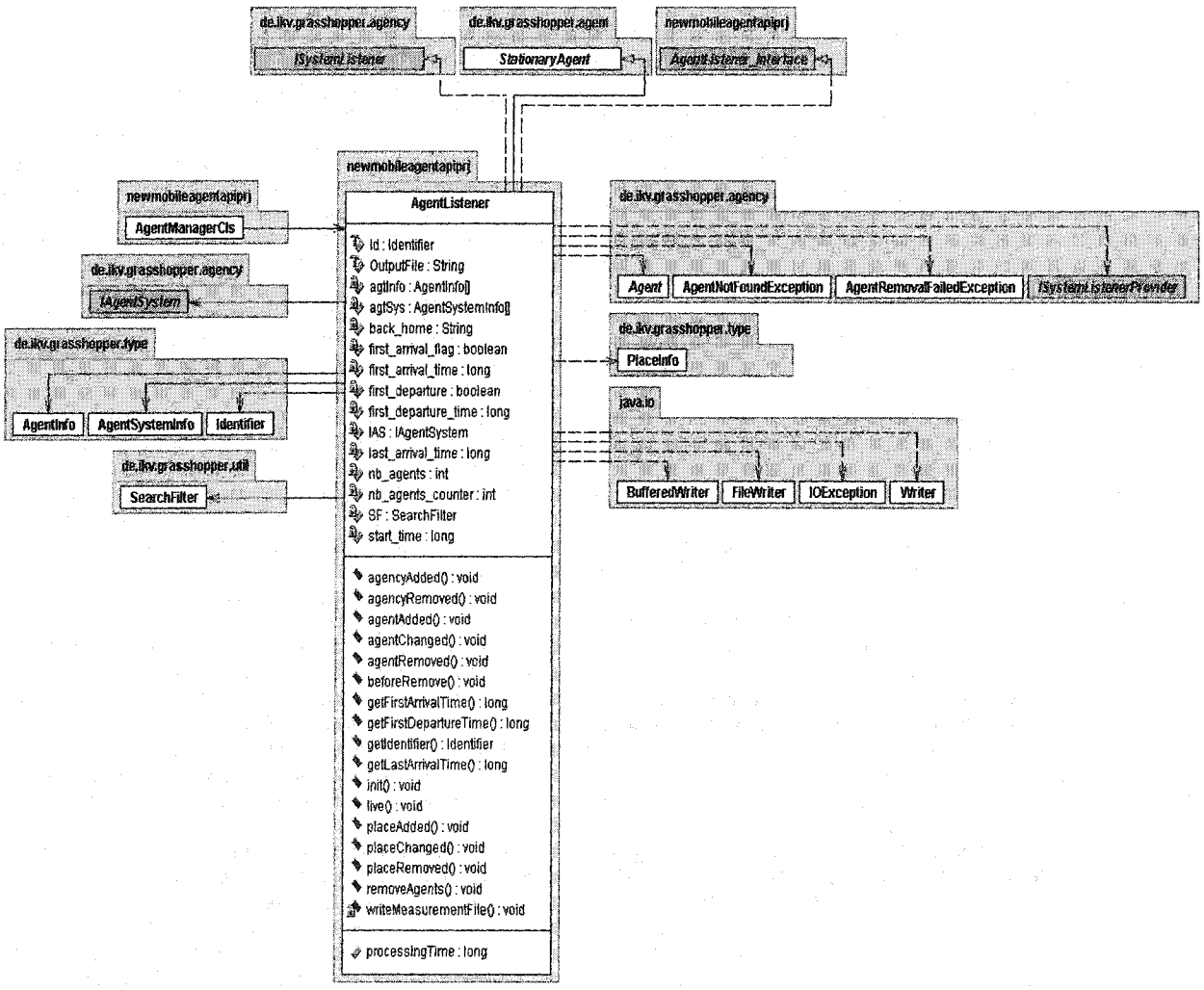
La classe Local_Agent:



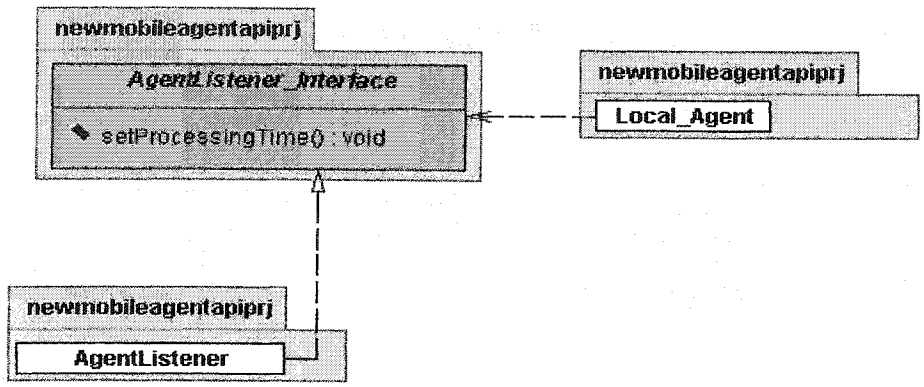
La classe SearchAgent:



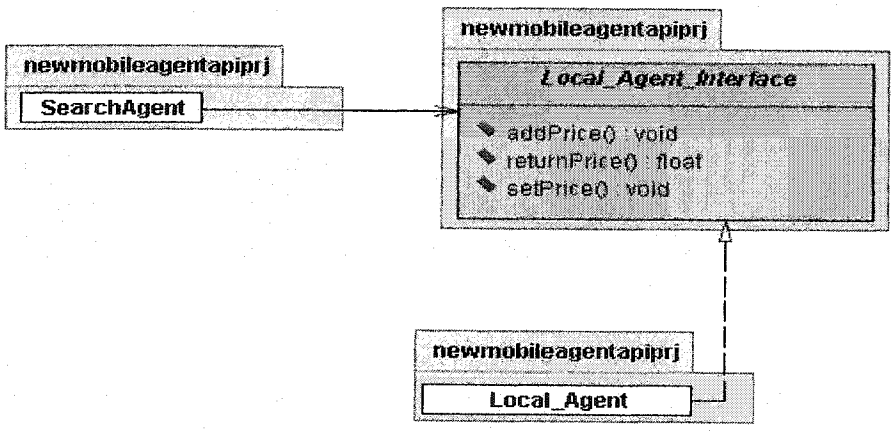
La classe Agent_Listener:



La classe AgentListener_Interface:



La classe Local_Agent_Interface :



La classe Search_Agent_Interface :