2002s-69

NetSA : une architecture multiagent réutilisable pour les environnements riches en informations

Marc Côté, Brahim Chaib-draa et Nader Troudi

Série Scientifique
Scientific Series

Montréal
Juillet 2002

CIRANO
Centre interuniversitaire de recherche en analyse des organisations

CIRANO

Le CIRANO est un organisme sans but lucratif constitué en vertu de la Loi des compagnies du Québec. Le financement de son infrastructure et de ses activités de recherche provient des cotisations de ses organisations-membres, d'une subvention d'infrastructure du ministère de la Recherche, de la Science et de la Technologie, de même que des subventions et mandats obtenus par ses équipes de recherche.

CIRANO is a private non-profit organization incorporated under the Québec Companies Act. Its infrastructure and research activities are funded through fees paid by member organizations, an infrastructure grant from the Ministère de la Recherche, de la Science et de la Technologie, and grants and research mandates obtained by its research teams.

Les organisations-partenaires / The Partner Organizations

- •École des Hautes Études Commerciales
- •École Polytechnique de Montréal
- •Université Concordia
- •Université de Montréal
- •Université du Québec à Montréal
- •Université Laval
- •Université McGill
- •Ministère des Finances du Québec
- •MRST
- •Alcan inc.
- •AXA Canada
- •Banque du Canada
- •Banque Laurentienne du Canada
- •Banque Nationale du Canada
- •Banque Royale du Canada
- •Bell Canada
- •Bombardier
- •Bourse de Montréal
- •Développement des ressources humaines Canada (DRHC)
- •Fédération des caisses Desjardins du Québec
- •Hydro-Québec
- •Industrie Canada
- •Pratt & Whitney Canada Inc.
- •Raymond Chabot Grant Thornton
- •Ville de Montréal

ISSN 1198-8177

NetSA : une architecture multiagent réutilisable pour les environnements riches en informations

Marc Côté[†]*, Brahim Chaib-draa* et Nader Troudi*[‡]

Résumé / *Abstract*

La toile (i.e., Web) exige de nos jours des logiciels capables d'opérer sur des sources d'information hétérogènes placées dans un environnement ouvert et dynamique (comme par exemple l'internet). Elle exige également de revoir certaines applications complexes (les bibliothèques digitales, les services banquiers, les assurances, etc.) comme un ensemble d'agents logiciels autonomes capables d'interagir en vue de réaliser certains objectifs. Dans ce papier, nous proposons une architecture réutilisable multiagent qui intègre de nouveaux développements technologiques (incluant entre autres: la technologie agent, KQML, la médiation entre agents, la technologie web, etc.) en vue de supporter ces exigences. Cette architecture appelée NetSA est composée de trois couches: la couche de communication avec l'utilisateur, la couche de traitement de l'information et la couche d'interrogation et d'extraction d'informations. Ainsi, NetSA traite des aspects I3 (Information, Interaction et Intelligence), au travers respectivement des bases de données qu'elle adresse pour extraire de l'information, des communications entre agents et des stratégies de recherche au moyen de la couche intermédiaire. Nous avons spécifié, développé et validé une telle architecture. Pour la validation, nous avons opté pour une compétition entre agents-banques, en vue de proposer les meilleurs taux hypothécaires à d'éventuels clients. Pour cela, nous avons étudié et développé des algorithmes à base d'enchères qui pourraient optimiser le profit de l'acheteur ou du vendeur selon les conditions de vente. Les premiers résultats de la validation ont clairement montré que l'architecture NetSA : (a) peut facilement satisfaire jusqu'à 15 utilisateurs (au-delà, il faudra penser utiliser plusieurs architectures NetSA); (b) est efficace par rapport à une recherche web classique; (c) est facile d'utilisation; (d) convient bien aux applications faisant intervenir des bases de données héritées.

Today, the web requires software that can operate on heterogeneous sources of information which are generally located in an open and dynamic environment (e.g., the internet). It also requires to reconsider some complex applications (as for instance, digital libraries, bank services, insurance services, etc.) as a set of autonomous agents having the ability to achieve some objectives. In this paper, we propose a reusable multiagent architecture integrating new technologies (as for instance: agent technology, KQML, Mediation between agents, web technology, etc.) that help to support such requirements. This architecture, called NetSA, has three levels: the first is devoted to the communication with users, the second deals with information and the third is in charge of extraction and queries of information. Thus, NetSA

---

† Marc Côté was a graduate student in the department of Computer science and Software-Engineering, Université Laval, Ste-Foy, Québec, Canada. He is now a software consultant, Ottawa, Canada.

* Brahim Chaib-draa is Professor in the department of Computer science and Software-Engineering, Université Laval, Ste-Foy, Québec, Canada; and CIRANO, 2020 University Ave, 25th floor, Montréal QC, H3A 2A5 CANADA; chaib@ift.ulaval.ca.

‡ Nader Troudi was a graduate student in the department of Computer science and Software-Engineering, Université Laval, Ste-Foy, Québec, Canada. He is now with NewTrade, Montréal, Canada.

addresses issues as (i) interaction between agents and agents/users, (ii) reasoning for the mediation between agents and finally, (iii) information through its interaction with legacy systems. We have specified, developed and validated such architecture. To validate it, we have opted for a competition between three banks so that they offer the best mortgage to users. To achieve that, we have studied and developed algorithms from auction theory that can optimize sellers or buyers according to some conditions. Our first results have shown that NetSA (a) can be accessible by a maximum of 15 users (for more users, it might be useful to use many NetSA architectures); (b) offers an efficient information seeking than with classical tools; (c) is easy to use; (d) is very useful if one wants to access to legacy systems.

Mots clés: Technologie agent, architecture multiagent, enchères entre agents.

Keywords: Agent technology, multiagent architecture, auctions.

1 Introduction et Motivations

Les recherches sur les bases de données (BDs) ont, par le passé, mis l'accent sur des environnements statiques de BDs, qu'elles soient centralisées ou distribuées. Dans ces environnements, l'information est généralement gérée de manière centralisée et la structure des données est consistante. Dans ces conditions, l'unification des concepts aux ensembles de données spécifiques est connue au moment de la conception et l'accès aux données peut être optimisé en utilisant des indices pré-traités. L'avènement de la toile (i.e, Web) nous offre tout un autre challenge : Il y a beaucoup plus d'informations et ces informations sont géographiquement distribuées. Sur la toile, il n'y a pas d'entité centrale en charge de la gestion des informations, puisque n'importe qui peut publier son information. Dès lors, la référence aux structures de données est presque inexistante et le peu qui existe a très peu de lien avec la sémantique. Dans ces conditions, il est très difficile de relier les concepts aux structures de données et les requêtes ne peuvent qu'être réduites à des recherches de ‹‹ mots clés ››. À bien des égards, ce type de recherche est limité et il convient de trouver de nouvelles techniques pour relever le défi d'une recherche d'information plus ‹‹intelligente›› sur la toile, particulièrement lorsqu'on désire accéder à des sources d'informations hétérogènes héritées [BBB+98].

Un autre défi qu'il convient de relever, et qui prolonge le précédent, tourne autour du développement de logiciels complexes sur la toile [DKB98]. De tels logiciels sont généralement caractérisés par l'impossibilité de prédire les services offerts dans les temps requis. Par exemple, les bibliothèques digitales, actuellement très en vogue sur la toile, nécessitent des accès faciles à tous les genres d'informations et ce, quelque soit l'emplacement géographique de ladite information. Les techniques classiques de recherche d'informations s'avèrent là aussi bien insuffisantes. Il y a bien d'autres questions qu'on pourrait se poser dans le cadre des bibliothèques digitales : (a) comment les auteurs des documents digitaux peuvent-ils trouver les gens qui les lisent et leur demander des compensations ? (b)

comment les lecteurs peuvent-ils trouver la ‹‹bonne information›› et éviter tout ce qui est sans importance ? (c) comment accéder aux services d'édition, de traduction, etc. ? (d) qui doit fournir ces services ? et pourquoi ? etc. Nous n'avons pas de réponse à ces questions, mais on pourrait aisément dégager des principes de conception de logiciels qui pourraient aider à leur trouver des solutions. Ainsi, il est clair qu'une bibliothèque digitale est par nature distribuée géographiquement et fonctionnellement et que par conséquent, il serait très risqué de la voir comme une seule entité centrale présentant une solution intégrée complète. Au contraire, il serait plus conforme de la considérer comme une équipe d'agents autonomes qui interagissent ensemble en vue d'atteindre certains objectifs (comme les auteurs, les éditeurs et les secrétaires qui travaillent ensemble pour éditer un livre).

En résumé, la toile exige de nos jours des logiciels capables d'opérer sur des sources d'information hétérogènes placées dans un environnement ouvert et dynamique (comme par exemple l'internet). Elle exige également de revoir certaines applications complexes (e.g. les bibliothèques digitales) comme un ensemble d'agents autonomes capables d'interagir en vue de réaliser certains objectifs. Au vu de cela, nous proposons une architecture, appelée NetSA (Networked Software Agent) [CT98], qui intègre de nouveaux développements technologiques en vue de supporter ces exigences. NetSA comprend en particulier :

1. Un ensemble d'agents pouvant représenter les utilisateurs, et encapsuler les BDs héritées et plus généralement, les sources d'information. Dans un tel système, par exemple, ajouter une nouvelle source d'information consiste alors à ajouter un agent tout en annonçant ses capacités. Dès lors, l'utilisation de la technologie agent permet un haut degré de décentralisation des capacités et rend donc aisée l'extensibilité.
2. Des agents spécialisés en médiation, ce qui a pour effet d'unifier les besoins informationnels aux ressources existantes. Dès lors, les requêtes de mise à jour et d'extraction sont acheminées directement aux seules ressources pertinentes.
3. Des outils pertinent à l'Internet, en particulier au niveau du langage de programmation (JAVA), des Applets, des RMIs, etc. ce qui a pour effet de fournir des interfaces utilisateurs assez générales pouvant tourner sur toute plateforme. Dès lors, on pourrait développer des "agents omniprésents" pouvant être déployés au niveau de n'importe quelle source d'information indépendemment de sa localisation ou de sa plateforme de développement.

Il convient maintenant de préciser ce qu'on entend par ‹‹agent››. Jennings [JSW98] définit une telle entité comme suit :

> Un agent est un système informatique situé dans un quelconque environnement capable d'exécuter des actions flexibles* et autonomes† dans le but d'accomplir les objectifs pour lesquels il a été créé.

Cette définition est conforme à notre vision de l'entité agent, et par conséquent, nous avons convenu de l'adopter tout au long de ce papier. Bien entendu, dépendamment des applications, certaines propriétés sont plus importantes que d'autres et il peut même s'avérer que pour certains types d'applications, des propriétés additionnelles soient requises.

Comme pour le cas des agents, un système multiagent [CdJM02, MCd96] comporte différentes caractéristiques. Ses principales caractéristiques sont pour l'essentiel : (1) chaque agent du système possède des informations et des compétences restreintes dans le cas de la résolution d'un problème donné ; (2) il n'y a pas de système central de contrôle ; (3) les données et les informations sont décentralisées et ; (4) le fonctionnement du système multiagent est asynchrone.

L'architecture NetSA que nous proposons dans ce papier est à base d'agents hétérogènes (agent utilisateur, agent médiateur, agent ressource, etc.). Au sens de la définition précédente, NetSA constitue elle-même un système multiagent. Il se peut toutefois qu'on ait des interactions entre plusieurs NetSA et là-aussi, on est dans un environnement multiagent.

Le reste du papier est structuré comme suit. La prochaine section présente les caractéristiques et couches de l'architecture NetSA. La troisième section détaille les agents composant une telle architecture. La quatrième section présente un exemple d'utilisation de NetSA. Finalement, la cinquième section compare NetSA aux autres architectures.

2 Caractéristiques et couches de NetSA

2.1 Caractéristiques de NetSA

Comme nous l'avons déjà dit, NetSA est une architecture multiagent conçue pour oeuvrer particulièrement sur des sources hétérogènes placées dans un environnement ouvert et dynamique (Internet). En la développant, nous l'avons dotée de certaines caractéristiques utiles.

*d'un agent signifie que l'agent a "conscience" de l'environnement dans lequel il évolue et est capable de changer son comportement si son environnement change également.

†implique qu'il est capable d'agir sans l'intervention directe d'autres entités.

2.1.1 Réutilisabilité et portabilité

NetSA se veut une architecture réutilisable dans la mesure où elle été développée comme une coquille vide, c'est-à-dire un ensemble d'agents communiquant via KQML. Pour une application donnée, il suffit alors d'ajouter les sources d'information hétérogènes et d'adapter les connaissances spéciques à l'application au niveau des agents d'interface et de la couche intermédiaire. Dès lors, NetSA peut être utilisée pour la recherche sur Internet, pour les finances ou pour la santé moyennant des changements quant au contenu des agents.

Le fait d'être programmé en Java de *Sun Microsystems* fait de NetSA une architecture portable, c'est-à-dire qu'elle peut être exécutée sur plusieurs systèmes d'exploitation. L'utilisateur peut donc exécuter NetSA sur une station Sun aussi bien que sur un PC ou sur un Macintosh.

NetSA offre un accès facile aux utilisateurs. Le moyen de communication entre les utilisateurs et NetSA se fait via l'Internet par l'entremise de la toile. La diffusion rapide sur Internet et son accessibilité font que NetSA peut être consultée à partir de n'importe quel point dans le monde.

2.1.2 Communication entre agents basée sur JATLite

La couche de communication de NetSA est basée sur JATLite[‡], développé à l'université de Stanford. Le choix pour un tel outil réside dans le fait qu'il a toutes les fonctionnalités nécessaires à la communication entre les agents en plus de fournir un vérificateur de la syntaxe de KQML. Rappelons que KQML [FFMM94] est un langage d'interrogation et de manipulation des connaissances. C'est un langage basé sur les actes du langage naturel [CdV98]. Il est surtout utilisé pour la communication entre agents et les bases de données coopératives. Ce langage n'est pas encore normalisé, mais étant donné sa forte utilisation (particulièrement dans les laboratoires de recherche), on peut dire qu'il est devenu une norme de fait.

Nous avons étendu KQML de façon à ce que le contenu d'un message soit divisé en deux parties. La première partie (`[in]`) est constituée de l'information utile pour traiter la requête. La seconde partie (`[out]`) informe l'agent qui reçoit le message de ce qu'il doit retourner comme information ou ce qu'il doit faire comme travail. Dans l'exemple ci-dessous, l'agent superviseur répond à l'agent utilisateur que la page Web à afficher est la page contenant ‹‹Luc Côté ››, et que pour le faire, il convient d'utiliser le fichier gabarit *result.html*.

[‡] disponible à l'adresse : http ://java.stanford.edu/

```
(reply
        :sender      Supervisor
        :receiver    UserAgent
        :replyWith   Supervisor172539363539546
        :inReplyTo   UserAgent172539363533819
        :context     mortgage
        :content     [in] name = Luc Côté ;
                     [out] pageID = result.html ;)
```

2.1.3 HTML, formulaires et JavaScript

Pour pouvoir élaborer des formulaires, nous avons pensé au départ utiliser des applets. Nous n'avons cependant pas pu contourner une inconsistance au niveau des contrôles graphiques entre les différentes plate-formes. Par exemple, la qualité graphique des stations Sparc de Sun Microsystems provoque un rétrécissement des champs textes programmés sous Windows. De plus, les niveaux de sécurité implantés sur les différents navigateurs Internet nous ont compliqué grandement la tâche. Nous avons donc opté pour des pages HTML contenant des formulaires et des JavaScripts [Net97]. Ces pages sont expédiées et dirigées par un Servlet[§] [Mic98] selon les besoins de l'utilisateur.

Le formulaire est un moyen consistant pour recueillir des données à partir d'un navigateur Internet. Ceci repose sur le fait qu'un navigateur programmé pour un système d'exploitation utilise les contrôles de saisie de ce système et non ceux reprogrammés par Java. De cette façon, nous sommes donc toujours certains d'obtenir un affichage impeccable peu importe le système d'exploitation. Nous avons également convenu d'utiliser des JavaScripts dans les formulaires afin de vérifier les données saisies par l'utilisateur. En les utilisant, on peut vérifier les champs obligatoires et les champs critiques. L'utilisation des JavaScripts nous permet en fait de faire un premier filtrage ou validation de l'information, avant de l'envoyer vers l'agent utilisateur.

2.2 Les trois couches de NetSA et les agents les composant

Le fait qu'on a affaire à des sources d'informations hétérogènes nous a fait opter pour le développement d'agents ressources capables d'accéder à ces sources.

[§]Un Servlet est un programme Java capable de traiter les données reçues via Internet comme le ferait un programme Perl pour la gestion de formulaire.

Il nous a paru également intuitif d'interfacer l'utilisateur au moyen d'agent utilisateur. Finalement, et comme nous l'avons souligné plus haut, le fait qu'on a affaire à des environnements ouverts et dynamiques nous a fait opter pour le développement d'une couche d'agents intermédiaires. On voit donc apparaître trois niveaux d'abstraction au niveau de NetSA : (1) l'unité de communication avec l'utilisateur ; (2) l'unité de traitement et de médiation ; (3) l'unité d'interrogation et d'extraction des données. Il convient de noter ici que ce découpage est le même que celui qu'on trouve habituellement dans les BDs classiques, mais il est très différent quant au contenu de ces unités. Dans NetSA, les trois unités font référence à la technologie-agent et à la médiation entre ces mêmes agents.

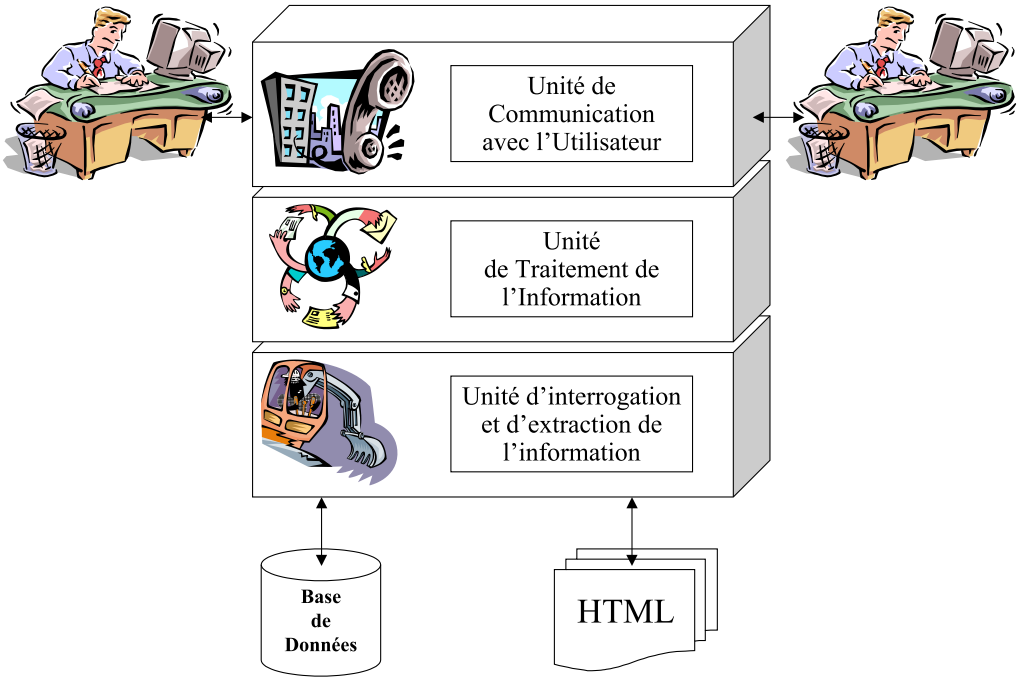

FIG. 1 – Couches abstraites de NetSA

Au niveau du contenu précisément, les trois unités sont comme suit :

Unité de communication avec l'utilisateur : Cette unité est chargée des communications entre NetSA et l'usager. Elle comprend des agents interagissant avec l'utilisateur pour l'aider à réaliser une tâche bien précise. Cette

interaction se traduit par une transformation des requêtes de l'usager qui, transformées en des actes du langage KQML, facilitent la communication avec les agents de l'unité de traitement. L'unité vérifie également la consistance des données fournies par l'utilisateur.

Unité de traitement et de médiation : Cette unité reçoit de l'unité de communication les requêtes à satisfaire ainsi que les informations fournies par l'utilisateur. Elle décompose ces requêtes en sous-plans. Un sous-plan est une succession d'actions à exécuter dans le but d'atteindre un objectif intermédiaire et une série de sous-plans forme un plan global. L'unité comporte également une partie ‹‹médiation›› pour rechercher des données dans le système multiagent. Tel un patrouilleur, elle dirige les agents vers la ressource désirée en fournissant le nom de l'agent en charge de cette ressource.

Unité d'interrogation et d'extraction d'informations : Cette unité est composée d'agents formant une interface entre les bases de données et l'unité de traitement d'informations. Ces agents transforment les requêtes KQML reçues et les traduisent en requêtes SQL pour interroger des bases de données. De ces bases de données, les agents retirent l'information pertinente et la redirigent vers l'unité de traitement de l'information sous un format KQML. Ces agents peuvent également retirer l'information contenue dans une page HTML de l'Internet.

2.3 Propriétés des agents composant NetSA

L'architecture multiagent NetSA comporte différents types d'agent dans des concentrations variables. Dans la structure arborescente de NetSA, nous pouvons facilement distinguer les trois couches d'abstraction et leur composition comme le montre la Figure 2. C'est ainsi que nous trouvons : (1) au moins un agent utilisateur dans l'unité de communication avec l'utilisateur ; (2) au moins un agent superviseur dans l'unité de traitement de l'information ; (3) au moins un agent intermédiaire dans l'unité de traitement de l'information ; (4) plusieurs agents ressources dans l'unité d'interrogation et d'extraction de l'information.

Les agents de NetSA possèdent plusieurs propriétés qui sont conformes aux propriétés générales des agents. Ces propriétés sont pour l'essentiel : (a) *Portabilité* : pour assurer sa portabilité, l'agent est programmé avec le langage Java ; (b) *Autonomie* : l'agent fonctionne sans l'intervention de l'utilisateur ; (c) *Stabilité* : une bonne gestion des exceptions permet à l'agent de demeurer dans un état stable ; (d) *Persistance* : l'agent reprend le cours normal de ses opérations après

une panne du système informatique ; (e) *Flexibilité* : les agents de NetSA doivent être en mesure de pouvoir être reconfigurés afin qu'ils puissent être facilement utilisés pour rechercher des informations dans la plupart des applications.

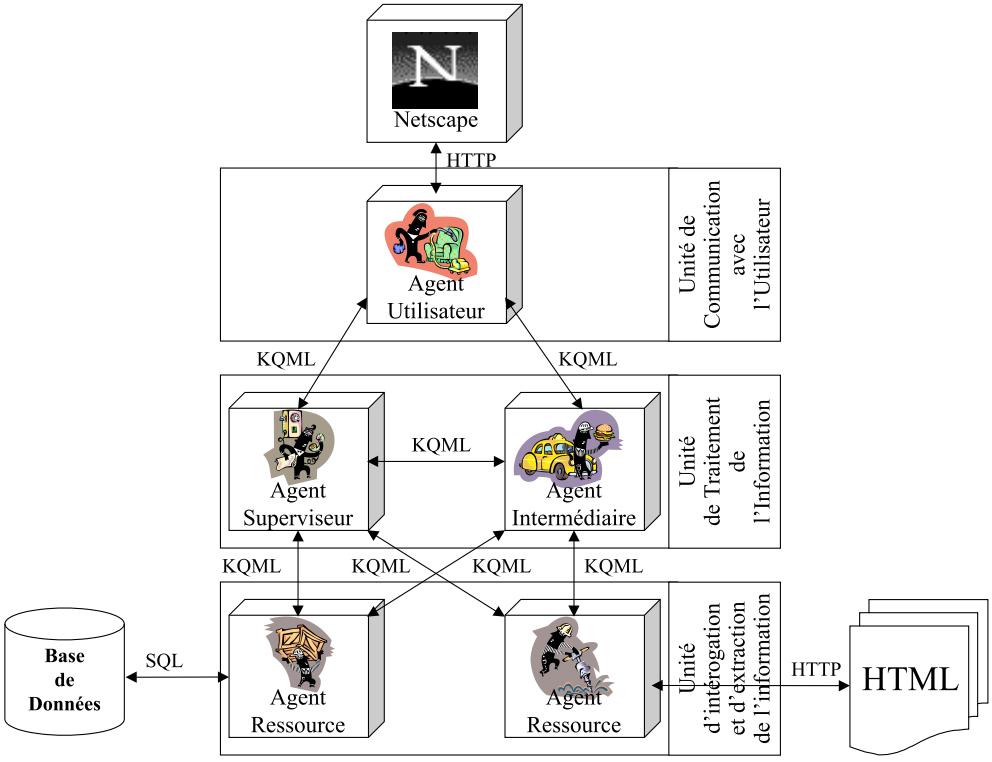

FIG. 2 – Architecture NetSA.

3 Détails des agents composant NetSA

3.1 Agent utilisateur : description et architecture interne

L'agent utilisateur est la porte d'entrée des requêtes externes au système. Il fournit à l'utilisateur le bon formulaire HTML lui permettant de faire une requête. L'agent traduit ensuite le formulaire soumis dans le protocole KQML en vue de son utilisation dans le système multiagent. Chaque agent est capable de s'occuper d'un ou de plusieurs utilisateurs au même moment.

L'agent utilisateur s'occupe de transmettre à l'utilisateur les pages Web requises pour la cueillette et l'affichage des informations pour le domaine auquel il a été destiné. Dans le présent travail, un système d'analyse dynamique (Servlet) a été utilisé pour interagir avec l'utilisateur. L'interaction entre l'agent et son utilisateur se fait pour l'essentiel de la manière suivante. L'utilisateur choisit d'abord le contexte dans lequel il veut travailler. L'agent lui donne alors la première page de cueillette d'informations. Selon les réponses de l'utilisateur, l'agent déterminera la page suivante.

La gestion des pages Web pour la communication avec l'utilisateur est faite à l'aide d'un Servlet. Chaque page Web utilise des formulaires et des JavaScript pour recueillir l'information. Dès lors, nous pouvons gérer la cohérence des contrôles graphiques pour chaque système d'exploitation. Chaque contrôle graphique portera un nom unique qui correspond à la variable dont le contrôle capte l'information. L'ordre dans lequel l'agent doit afficher les pages Web est défini dans l'attribut caché ‹‹nextID›› qui prend la forme suivante : <input type="hidden" name="nextID" value="hypo2.html">.

L'architecture interne de l'agent utilisateur, quant à elle, est composée de trois modules principaux et d'un registre de sauvegarde, comme l'indique la Figure 3. Les trois modules donnent à l'agent une plus grande extensibilité étant donné leur découpage selon les tâches. Les trois modules sont comme suit :

Le module de communication utilisateur : C'est un Servlet qui communique avec le module de traitement via des RMIs (Remote Method Invocation). Il reçoit les données de la page HTML et les transfère au module de traitement. L'opération inverse est également disponible, c'est-à-dire que le module peut recevoir de l'information du module de traitement pour la montrer à l'utilisateur via une page Web.

Le module de communication inter-agents : Il reçoit du module de traitement, des demandes de transmission de messages KQML vers les autres agents. Il transfère également les informations reçues des agents du système multiagent au module de traitement.

Le module de traitement : Il reçoit des données du module de communication utilisateur et les sauve dans le registre de l'agent. Ce registre contient toutes les informations recueillies par l'agent sur les utilisateurs du système multiagent. Il détermine ensuite si toutes les informations nécessaires à la formation de la requête sont disponibles. Dans l'affirmative, il construit un message KQML et demande au module de communication inter-agents de le transmettre. Dans la négative, il demande les informations complémen-

taires à l'utilisateur via une page HTML qui est transmise au module de communication utilisateur ; ce dernier se chargera ensuite de la faire parvenir à l'utilisateur.

Le registre de sauvegarde : Il a pour rôle de sauvegarder les données que l'utilisateur a fournies, de manière à regrouper par thèmes la cueillette d'information venant de celui-ci. Un tel découpage permet en fait de construire des formulaires HTML bien moins chargés et plus conviviaux.

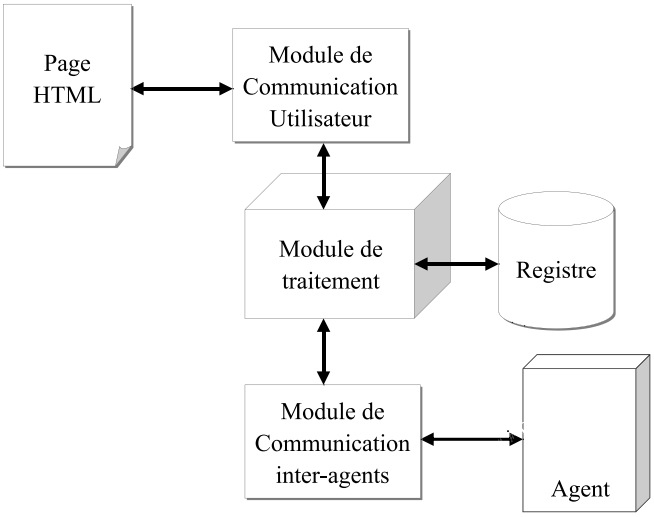

FIG. 3 – Architecture de l'agent utilisateur.

3.2 Agent Ressource : description et architecture interne

Cet agent reçoit des requêtes formulées en KQML et les transforme en requêtes SQL afin d'extraire l'information requise des bases de données ou de rechercher l'information demandée dans une page HTML. L'information trouvée est ensuite traduite en KQML pour répondre à la requête du demandeur. D'une façon générale, un agent ressource gère une seule ressource. De toute évidence, plus le nombre d'agents ressources est grand, plus nous avons accès à une information complète et diversifiée.

Dans NetSA, l'agent ressource a pour but d'interroger, d'extraire et de mettre à jour des données dans une base de données ou dans une page Web structurée. Il

reçoit d'abord une requête d'un agent appelant, traduit cette requête, interroge la base de données et construit une réponse qui sera acheminée vers l'agent appelant.

La structure de la page Web utilisée a un format un peu spécial dans la mesure où nous avons ajouté des mots-clés dans le code HTML pour reconnaître les champs donnant l'information spécifiée. Plus spécifiquement, nous avons introduit une étiquette nommée ‹‹VAR›› qui décrit la donnée contenue dans la page Web.

L'architecture interne de l'agent ressource est, quant à elle, composée de deux modules et de deux interfaces comme indiquée en Figure 4. Ces quatre éléments donnent aux agents une plus grande extensibilité étant donné leur découpage en tâches. Ils sont comme suit :

Le module de communication : Il reçoit des autres agents les requêtes sous la forme d'un message KQML et suite à cela, il appelle le module de traitement. Il reçoit également des demandes de transmission de messages du module de traitement. Ces demandes de transmissions constituent des réponses aux requêtes reçues.

Le module de traitement : Il traite les requêtes reçues par le module de communication. Pour cela, il les transforme en requête SQL et les achemine à l'interface avec la base de données ou ordonne une recherche à l'interface HTTP. Il construit ensuite une réponse sous forme de message KQML qu'il donne au module de communication pour être expédiée.

L'interface avec la base de données : Elle reçoit du module de traitement une demande de recherche de variable dans une base de données. L'interface exécute la requête SQL demandée par le module de traitement et retourne à ce module les valeurs trouvées.

L'interface HTTP : Elle reçoit du module de traitement une demande de recherche de variable dans une page Web. Elle parcourt alors la page Web à la recherche d'une étiquette ‹‹VAR›› définissant la variable demandée. Elle retourne ensuite la valeur de cette variable au module de traitement.

Pour finir avec l'agent ressource, il nous faut donner un aperçu sur ses communications avec les autres agents de NetSA. Pour ce type de communication, nous avons convenu d'utiliser les actes ask-one et ask-all de KQML, en vue d'extraire les données. Le premier acte donne la première instance trouvée et le second retourne toutes les instances trouvées de l'information demandée. Pour modifier la base de données, les actes de langage suivants sont utilisés :

– insert pour insérer un nouvel enregistrement dans la base de données,

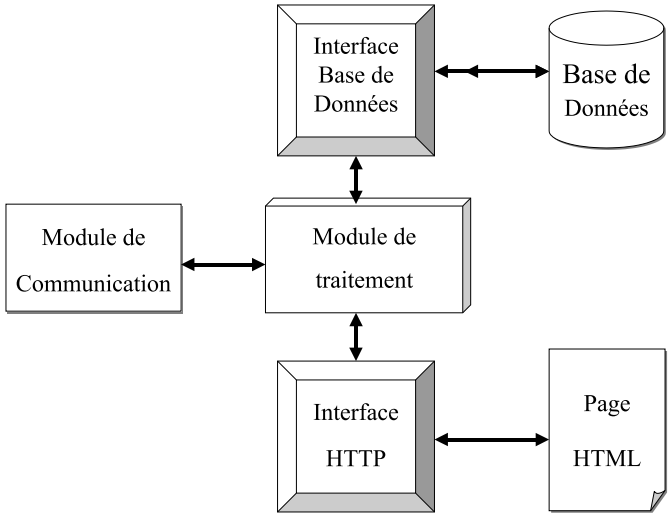

FIG. 4 – Composantes de l'agent ressource.

- delete-one pour effacer la première instance trouvée à partir de l'information donnée,
- delete-all pour effacer toutes les instances trouvées à partir de l'information donnée,
- update pour modifier un enregistrement.

3.3 Agent Intermédiaire : description et architecture interne

Il existe plusieurs manières de ‹‹router›› l'information entre deux agents comme le montre la figure 5 [KH98].

Dans le premier cas, un agent producteur effectue directement (en utilisant l'acte Ask) auprès d'un agent consommateur une certaine requête et celui-ci lui répond directement (en utilisant l'acte Tell). Ce cas est en fait un cas classique de routage à base d'adresses. Dans le deuxième cas, l'agent consommateur souscrit un service auprès du médiateur et lorsque celui-ci reçoit d'un agent producteur tiers le contenu requis, il l'envoie à cet agent consommateur. C'est donc bien un routage basé sur le contenu. Dans la médiation du type 1, les agents producteurs annoncent leurs services (en utilisant l'acte Advertise) auprès de l'agent médiateur. Un agent consommateur demande alors à l'intermédiaire d'agir vraiment ‹‹comme intermédiaire ›› (1) en faisant suivre sa demande au producteur con-

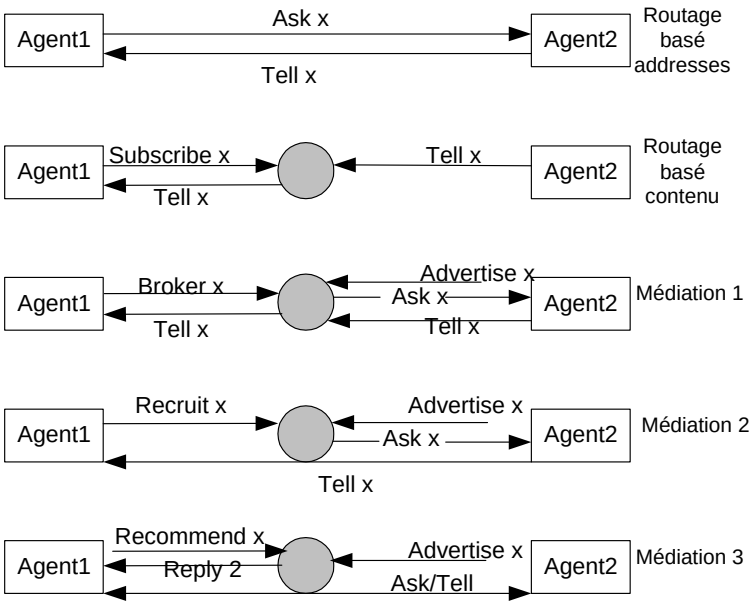

FIG. 5 – Différents modèles de routage de l'information (D'après [KH98]).

cerné (le médiateur sait qui fait quoi) et, (2) dans une deuxième étape, en lui faisant suivre, dans une deuxième étape, la réponse du dit producteur. À noter que l'agent consommateur utilise ici l'acte Broker. Dans le cas de la médiation 2, le producteur demande à l'agent intermédiaire de faire suivre sa demande au producteur concerné, en stipulant que les réponses doivent lui être adressées directement. Pour faire une telle requête, le consommateur utilise l'acte Recruit. Dans le dernier cas finalement, le producteur demande au médiateur de lui recommander quelqu'un qui peut satisfaire sa requête (il utilise pour cela l'acte Recommend). Une fois cela fait, la requête est directement négociée entre producteur et consommateur. Ce cas est parfois aussi appelé ‹‹pages jaunes ››.

L'agent intermédiaire dans NetSA étant du type ‹‹pages jaunes ››, il doit donc associer aux différentes requêtes les agents qui sont capables d'y répondre. Pour cela, les agents ressources doivent s'annoncer auprès de lui, en lui envoyant un message KQML l'avertissant du type de service qu'ils sont capables de fournir. Lorsqu'un agent utilisateur demande à l'agent intermédiaire s'il y a un agent capable de satisfaire sa requête, celui-ci lui répond en lui donnant le nom du ou des agents aptes à satisfaire ladite requête. Il en va de même de tout agent qui n'est plus accessible à partir de NetSA. Il lui revient de se désinscrire (auprès de l'agent pages jaunes) et ainsi, ses services ne sont plus assurés. Dès lors, la dynamique au niveau d'Internet est complètement gérée, dans la mesure où les agents qui se connectent et se déconnectent à NetSA sont complètement gérés.

L'architecture de l'agent intermédiaire, comme le montre la Figure 6, comporte trois modules et deux bases de données. En ce qui concerne les bases de données, la première est une base de règles qui fournit à l'agent le protocole à suivre lors de la communication. La deuxième est une base de faits, dans laquelle l'agent emmagasine les connaissances qu'il a de son environnement. Quant aux trois modules, ils sont comme suit :

Le module de communication : C'est une interface entre l'agent et son environnement. Il est utilisé pour transmettre et recevoir des messages sous la forme KQML.

Le module de réseaux à contrats [Smi80] : Il gère spécialement le protocole de réseau à contrats dont les spécifications sont stockées dans la base de règles. Le réseau à contrats est un protocole de négociation qui s'utilise de la façon suivante : Lorsque l'intermédiaire est en face de plusieurs agents ayant la même spécialité ou des spécialités similaires (ces agents se sont fait connaître préalablement auprès de l'intermédiaire grâce aux ‹‹annonces ››), il fait une annonce mieux ciblée, plus pointue de la requête qu'il a à traiter.

Les sous-contractants potentiels répondent à l'agent appelant et ce dernier choisit l'agent répondant le mieux à ses attentes.

Le module de traitement : C'est un moteur d'inférence développé en Java et basé sur JESS.

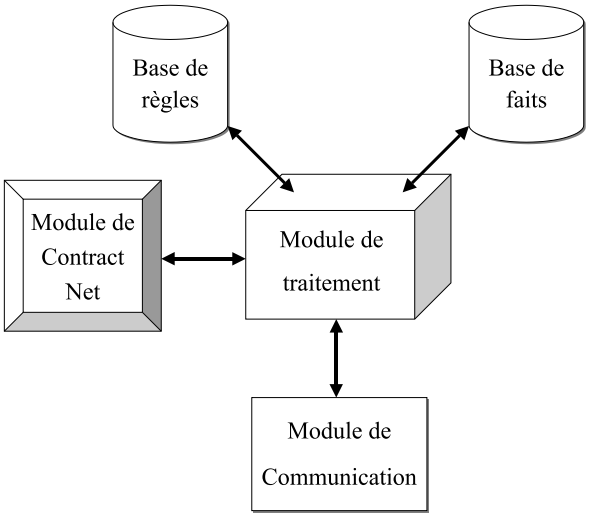

FIG. 6 – Architecture de l'agent intermédiaire.

3.4 Agent Superviseur : Description et architecture interne

L'agent superviseur est responsable de l'exécution à haut niveau des requêtes venant des agents utilisateurs. Lorsqu'il reçoit une requête, celle-ci est exécutée par le biais de plans prédéfinis. Majoritairement, un plan est composé de plusieurs sous-plans. Par exemple, un plan pour accéder à la requête ‹‹trouver toutes les banques qui offrent le service XY››, pourrait être décomposé en sous-plans, de la façon suivante :

1. accéder au fichier banque,
2. trouver celles qui font le service X,
3. trouver celles qui font Y,
4. trouver celles qui font XY,
5. fermer le fichier.

En fait, les plans dans NetSA réfèrent à des schémas, voire des directives, pour un agent. C'est pourquoi nous avons été amené à élaborer : (1) une structure permettant de bien suivre le déroulement des plans et, (2) un langage pour diriger les actions des plans. Cette structure et ce langage forment un nouveau pseudo-paradigme nommé POPA (Programmation Orientée Plan pour les Agents) (le lecteur intéressé par ce langage pourrait se référer à [Côt99]). Dans POPA, un plan est formé d'un ou d'une succession de sous-plans. Les sous-plans sont constitués d'une déclaration et d'actions.

La déclaration d'un sous-plan débute toujours par la primitive SubPlan suivie du nom du sous-plan. Nous terminons toujours un sous-plan par la primitive End SubPlan.

La zone destinée aux actions suit toujours la déclaration SubPlan. Les actions peuvent être des calculs mathématiques, des assignations, des appels de fonctions prédéfinies, etc.

Dans POPA, il est également possible d'exécuter des plans en concurrence avec le mot réservé
`BeginConcurrentPlans`. Voici un exemple :

```
SubPlan {Compute Mortgage}
    BeginConcurrentPlans
        Check Credit
        Check Insurance
        Get Rate
        Get Ratio
        Get Tax
        Get Promotion
    EndConcurrentPlans
```

Lorsque l'agent superviseur manque d'informations pour la réalisation d'un plan, il recherche cette information auprès des agents ressources par une requête formulée en KQML. Après l'exécution du plan, l'agent superviseur synthétise les réponses et transmet une réponse finale à l'agent appelant ayant fait la requête d'origine.

L'agent superviseur est au coeur de l'exécution des requêtes de NetSA. Situé dans la couche centrale de l'architecture, il est responsable de la quasi-totalité des requêtes. Comme nous pouvons l'observer sur la Figure 2, l'agent superviseur est directement connecté à toutes les couches de l'architecture voire en liaison directe avec tous les agents. Comme tout agent de NetSA, l'agent superviseur a sa propre architecture interne (voir Figure 7). Celle-ci est divisée en trois modules :

(1) un module de communication, (2) un module de planification et (3) un module d'évaluation d'expression. Ces trois modules sont accompagnés de deux bases de données : la base de registre et la base des plans.

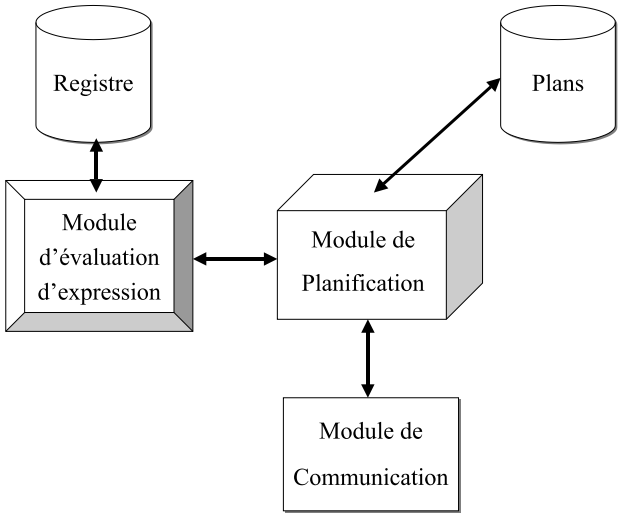

FIG. 7 – Architecture de l'agent superviseur.

Examinons d'un peu plus près chacun des modules de l'agent superviseur.

Module de communication : Le module de communication est le lieu de transition de tous les messages entrant et sortant de l'agent superviseur. Il vérifie la syntaxe des messages reçus et appelle le plan nécessaire à la réalisation de la requête contenue dans le message. Il y a 5 types de messages reconnus et échangés par le module de communication.

1. *Réception d'une requête :* Cette réception de message provoque le démarrage de l'exécution d'un plan. Le contenu du message contient les informations pertinentes à la réalisation du plan dans le champ d'entrée ([in]) et le nom du plan souhaité dans le champ de sortie ([out]) ([in] et [out] ont été introduits en 2.1.2).
2. *Réception d'une réponse :* Les messages de ce type contiennent l'acte de langage reply, ce qui déclenche un processus d'insertion des données contenues dans le champ d'entrée ([in]) vers la base de registre. Le champ de sortie ([out]) est habituellement vide dans ce cas.

3. *Transmission de requêtes ressource :* Le module de communication construit un message KQML pour être envoyé vers un agent ressource en utilisant les actes de langage ask-one ou ask-all. Le champ d'entrée du contenu du message contient les restrictions de la recherche d'information. Le champ de sortie indique quelle information doit être retournée.
4. *Transmission de requête page jaune :* Le module construit ici un message KQML pour être envoyé vers un agent intermédiaire en utilisant les actes de langage recommend-one ou recommend-all. Le champ d'entrée du contenu du message contient les restrictions de la recherche d'un agent. Le champ de sortie contient la valeur Agent-Name qui signifie à l'agent intermédiaire qu'il doit retourner le nom de l'agent trouvé.
5. *Transmission de réponses :* Lorsqu'un plan est terminé, le module de communication construit un message KQML utilisant l'acte de langage reply et le transmet à l'agent utilisateur demandeur. La réponse est donnée dans le champ d'entrée du contenu du message. Pour l'agent utilisateur, il convient d'ajouter dans le champ de sortie le nom de la page HTML modèle qu'il devra utiliser pour afficher la réponse à l'utilisateur.

Le module de communication utilise les classes de JATLite pour effectuer le service de transmission de messages.

Module de planification : Étant donné que le plan est la base du raisonnement de l'agent superviseur, il nous a fallu un module pour interpréter les plans. Tout d'abord, une base de plans est chargée lors de l'initialisation du module de gestion des plans. Par la suite, un registre est créé pour le stockage des données pendant la réalisation du plan. Chaque plan est une suite d'expressions écrites dans le langage POPA (voir plus haut en 3.4). Chacune de ces expressions est donc soumise au module d'évaluation d'expression. L'exécution d'un plan doit être vue comme un processus indépendant, ce qui permet à plusieurs plans de pouvoir s'exécuter en concurrence.

Module d'évaluation d'expression : Le module d'évaluation d'expression permet l'exécution du contenu des plans. Ce module est capable d'évaluer des expressions écrites dans le langage POPA. Il est donc utilisé pour faire des calculs, envoyer des messages, vérifier des informations, etc.

Jusqu'ici, nous avons présenté une architecture multiagent à trois couches. La première couche (l'unité de communication avec l'utilisateur) s'occupe de la traduction des requêtes de l'utilisateur en messages KQML compris par l'architecture. La seconde couche (l'unité de traitement de l'information) s'occupe de la réalisation des requêtes de l'utilisateur. La dernière couche (l'unité d'interrogation et d'extraction de l'information) exploite les informations contenues dans des sources distribuées et hétérogènes (y compris celles qui sont héritées).

4 Exemple d'utilisation de NetSA : la recherche du meilleur taux hypothécaire

Aussi bien les institutions banquaires établies, que les banques nouvelles se doivent aujourd'hui d'annoncer leurs services sur l'Internet. Les premières doivent en général, gérer des bases de données distribuées, hétérogènes et souvent héritées. Les secondes ‹‹poussent ›› comme des champignons (du moins aux USA) et certaines d'entre-elles ne durent pas bien longtemps. En utilisant NetSA, les banques établies pourraient fournir des services en exploitant leur BDs distribuées et héritées, tandis que les nouvelles banques pourraient s'inscrire (et se désinscrire lorsqu'elles font faillite) afin de fournir des services à d'éventuels utilisateurs. Ces derniers peuvent soit accéder à des services spécialisés via NetSA, soit carrément demander à trouver le ‹‹meilleur ›› service, comme par exemple le meilleur taux hypothécaire. Pour valider NetSA et montrer son utilité, nous avons convenu de l'utiliser dans le contexte d'une compétition à base d'enchères entre plusieurs banques pour l'attribution des prêts hypothécaires. Dans cette section, nous optons tout d'abord pour un type d'enchères, avant de venir à l'application elle même et des résultats qu'elle a engendrés.

4.1 Choix d'une enchère

Lorsque nous parlons d'enchères nous imaginons presque toujours la vente d'un produit où le prix de base est fixé par un encanteur[¶] et où les gens misent des montants d'argent de plus en plus élevés jusqu'à ce qu'il ne reste qu'un acheteur potentiel. En fait, il y a bien d'autres techniques d'enchères : la hollandaise, le meilleur prix, la Vickrey, la double et bien d'autres (généralement des variantes).

[¶] Ce terme est un québécisme et il désigne la personne qui procède à l'estimation et à la vente aux enchères.

Ces enchères ont créé un réel engouement sur Internet. Des sites comme *Auction Universe* [Auc98] vendent tous les jours une variété de produits en utilisant les enchères.

Enchères anglaise : Les enchères anglaise, aussi connues sous le nom de criée ou enchères à prix ascendant, sont les enchères les plus populaires. Elles sont utilisées pour vendre des oeuvres d'art, du vin et bien d'autres marchandises. Généralement, l'enchère anglaise débute par le plus bas prix acceptable (prix du marché) pour un bien donné. Par une succession de mises de plus en plus élevées de la part des enchérisseurs, le prix du bien augmente jusqu'à ce qu'un seul enchérisseur demeure dans la course. L'encanteur attribue, après un certain temps sans offre, le bien à cet enchérisseur qui a offert donc, la plus grande offre.

Du point de vue des avantages, il convient de signaler que la simplicité et la popularité de ce type d'enchère en font un excellent moyen de vente. La possibilité de pouvoir annuler une vente qui n'a pas rencontré le prix du marché est un atout pour le vendeur. Du côté des inconvénients, il faut préciser qu'il est généralement facile pour un groupe d'acheteurs de former une coalition qui aurait pour but de faire diminuer le prix des biens. Le fait que les acheteurs doivent être présents lors de la vente constitue un autre inconvénient pour ce type d'enchères. Les acheteurs doivent donc se déplacer, ce qui peut être coûteux pour l'acheteur. Il risque donc d'y avoir moins d'acheteurs si l'encan a lieu dans un endroit éloigné.

Enchères hollandaise : Dans l'enchère hollandaise (ou à prix descendant), une offre est faite dès le début à un prix extrêmement haut. L'encanteur diminue le prix progressivement jusqu'à ce qu'un acheteur désire l'article en criant ‹‹le mien !››, ou en appuyant sur un bouton qui signale la vente de l'article. Quand des articles multiples (comme les fleurs) sont vendues aux enchères, après le premier arrêt du prix, l'enchère continue et les prix descendent jusqu'à ce qu'il n'y ait plus de marchandise. À chaque arrêt, il revient à l'acheteur ayant crié ‹‹le mien !›› de choisir les produits de meilleure qualité. Les enchères hollandaises ont été utilisées, jusqu'ici, pour financer le crédit en Roumanie, pour l'échange de devises étrangères en Bolivie, Jamaïque et Zambie et pour vendre du poisson en Angleterre et en Israël.

Du côté des avantages, il convient de noter qu'étant donné que le prix débute à un seuil très élevé, il y a moins de risque pour le vendeur d'obtenir un prix inférieur au prix du marché. Si une personne veut vraiment un article, elle ne doit pas attendre longtemps sous peine de le voir attribuer à quelqu'un

d'autre. Lorsqu'il y a plusieurs objets du même type à vendre (comme des fleurs), cette méthode de vente peut s'avérer rapide et efficace. Du côté des inconvénients, il convient de voir qu'il n'y a pas de compétition pouvant faire monter le prix d'un article à un niveau exorbitant comme dans le cas de l'enchère anglaise. Il faudrait mettre le prix de base dès le départ à un prix exorbitant pour observer ce phénomène mais cela ralentirait le processus de vente. Comme pour l'enchère anglaise, il est facile pour un groupe d'acheteurs de former une coalition qui a pour but de faire diminuer le prix des marchandises.

Enchères du meilleur prix : Cette enchère a comme caractéristique principale le fait d'être secrète. Les mises sont faites par écrit et mises dans des enveloppes scellées pour conserver la confidentialité. Le gagnant de l'enchère est celui qui a la mise la plus élevée comme son nom l'indique. Dans le cas de la mise en enchère de plusieurs produits du même type (comme des fleurs), la personne ayant la plus haute mise choisira la première, suivie de la deuxième plus haute mise et ainsi de suite jusqu'à ce qu'il n'y ai plus rien à vendre. Donc, ce ne sont pas tous les acheteurs qui auront la chance d'obtenir quelque chose. Intuitivement, on pourrait penser que lorsqu'une personne désire vraiment le produit en vente, il lui est alors possible de l'obtenir en misant une très grosse somme. Or, ce n'est généralement pas le cas et les prix ont plutôt tendance à se rapprocher du prix du marché, car les gens ne veulent pas payer trop cher. Donc une bonne stratégie serait de miser un peu plus bas que le pris du marché pour obtenir un bon prix. Les mises scellées offrent aussi l'avantage de ne pas influencer les enchérisseurs. Il est également plus difficile de former des ententes malhonnêtes, car les enchérisseurs peuvent venir de partout. De plus, les enchérisseurs n'ont pas besoin d'être présents lors de la vente et la mise peut se faire par la poste. Finalement, le plus grand inconvénient de ce type d'enchères réside dans le fait qu'une personne possédant une très grande fortune est fortement avantagée.

Enchères de Vickrey : Cette enchère porte le nom de son créateur, William Vickrey [Vic61] (prix Nobel en économie de 1996), et elle est proche de l'enchère dite de meilleur prix vue précédemment. Là aussi, les mises sont déposées dans des enveloppes scellées pour des raisons d'anonymat. C'est également la plus haute mise qui l'emporte, mais le gagnant paiera le prix de la deuxième plus haute mise. Par exemple, si nous avons trois enchérisseurs : le premier enchérisseur offre 10$, le deuxième 15$ et le troisième

20$. Le troisième enchérisseur sera le vainqueur, car il a misé 20$ mais il paiera seulement 15$ soit la deuxième plus haute mise. Vickery a montré que ce type d'enchères donne un prix au produit qui se rapproche de celui du marché, évitant ainsi, que des gens trop zélés fassent augmenter le prix d'un produit démesurément. Il ne semble toutefois pas intéressant pour un vendeur d'utiliser ce mode de vente, car les prix semblent suivre le marché. En fait, ce n'est qu'une illusion, dans la mesure où des études ont prouvé que les gens misent toujours un peu plus haut, car ils savent que le prix payé sera moindre que la mise faite.

Enchères doubles : Bien que ce ne soit pas un type d'enchères familier, l'enchère double est la principale forme de commerce des institutions financières américaines depuis environ 100 ans. Son nom lui vient du fait que les acheteurs et les vendeurs font une offre pour le produit désiré. Nous avons donc, d'un côté, des offres de vente triées en ordre croissant et de l'autre des offres d'achat triées en ordre décroissant. Nous associons les offres (d'achat et de vente) une à une, tant que l'offre d'achat est supérieure ou égale à l'offre de vente [FR93].

Bien entendu, une personne aimant le risque sera avantagée par ce genre d'enchères. Les économistes prévoient un avenir prometteur pour ce type d'enchères dans le domaine du commerce électronique. Du point de vue des inconvénients, nous n'en connaissons aucun.

En résumé, il convient de préciser que les enchères anglaise et hollandaise sont des enchères itératives où plusieurs séquences d'offres sont soumises. Ce type d'enchères demande énormément de communication entre les deux parties (vendeurs et acheteurs) et ont une durée de vie assez longue étant donnée le grand nombre d'itérations requises pour la conclusion d'une vente d'un objet. Ce type d'enchère itérative peut donc être utilisé lorsqu'il n'y a pas de contrainte de temps ou lorsque cette contrainte est suffisamment grande.

Les enchères Vickrey, meilleur prix et double ne nécessitent qu'une seule itération pour ce qui est des offres. Ces enchères sont donc fortement conseillées dans le contexte d'agents logiciels, puisque la contrainte de temps est limitée. C'est pourquoi nous avons convenu d'utiliser les enchères de type non-itérative pour implanter un système de courtage de prêts hypothécaires dans NetSA.

Dans une première phase, nous avons opté pour une enchère de type ‹‹meilleur prix››, dans la mesure où ce type d'enchères offre des temps de réponse adéquats

pour les utilisateurs tout en étant facile à mettre en œuvre[‖]. À l'avenir nos compléterons nos résultats par des simulations à base d'enchères Vickery et doubles.

4.2 Un aperçu sur l'Hypothèque

Dans la mesure où on veut utiliser NetSA pour la compétition entre plusieurs banques dans le cadre de l'attribution des prêts hypothécaires, nous avons jugé utile de préciser ce qu'est une hypothèque. Nous avons également jugé bon de décrire les éléments qui peuvent faire varier le coût entre les prêts hypothécaires des différentes banques.

Un prêt hypothécaire est un prêt accordé par une institution financière à un demandeur dans le but d'acquérir une maison ou un immeuble. L'immeuble ou la maison achetée est la garantie de paiement de l'hypothèque. Dans ce cas, la maison demeure la propriété de l'institution financière tant que le paiement total n'est pas effectué. Autrement dit, la maison fait office de garantie de paiement.

Le calcul d'hypothèque est différent selon le pays considéré. La formule pour calculer une hypothèque aux États-Unis est différente de celle du Canada. Cela dépend des différentes lois en vigueur pour les prêts des différents pays. Certains pays peuvent même utiliser plusieurs formules de calcul d'hypothèque. Pour notre validation, nous nous sommes concentrés sur le calcul d'hypothèque canadien. Pour cela, il nous a fallu calculer la capacité de remboursement de chaque emprunteur et le montant d'argent que chaque emprunteur doit payer mensuellement.

En ce qui concerne la capacité de remboursement, il nous faut faire intervenir (1) le salaire annuel de l'emprunteur, (2) le ratio autorisé par l'institution financière (i.e. 0,40 pour 40%) et, (3) la somme des dépenses mensuelles de l'emprunteur après l'achat d'une maison (taxes foncières, électricité, chauffage, téléphone, carte de crédit, voitures, marges de crédit et autres achats mensuels).

Le montant d'argent que l'emprunteur doit payer mensuellement avec le type d'hypothèque choisi est calculé selon la formule suivante :

$$Montant = Capital \times \left[\frac{\left(1+\frac{taux}{200}\right)^{\frac{1}{6}}-1}{1-\left(\left(1+\frac{taux}{200}\right)^{\frac{1}{6}}\right)^{-nbMois}} \right]$$

où : (1) *capital* : représente la valeur du montant emprunté ; (2) *taux* : le taux d'intérêt en vigueur ; (3) *nbMois* : la durée de l'hypothèque en mois.

[‖] Cela vaut uniquement pour les items non corrélés entre eux. Pour les enchères combinatoires [Sak00] et multi-items [BCdK02], le problème est beaucoup plus complexe.

Pour que le prêt hypothécaire soit consenti à un emprunteur donné, il faut que la capacité de remboursement de cet emprunteur soit supérieure au montant mensuel de l'hypothèque visée.

4.3 Application de NetSA aux enchères entre banques

Tout d'abord, l'utilisateur-emprunteur accède par un navigateur Internet (Netscape, Internet Explorer, HotJava, etc.), à un site web. Sur ce site, il va trouver la page d'accueil de NetSA où l'utilisateur en question est sensé choisir le type d'opération qu'il veut faire. Pour le moment, seul le courtage de prêt hypothécaire est actif (voir Figure 8).

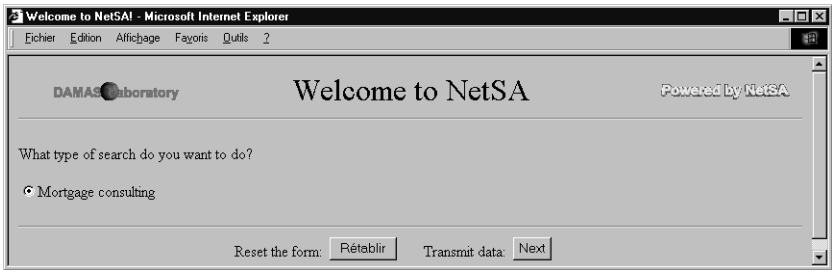

FIG. 8 – Page d'accueil de NetSA.

La deuxième page fait accéder l'utilisateur à un formulaire. Ce formulaire est utilisé pour collecter les informations personnelles propres à chaque utilisateur (nom, prénom, numéro d'assurance sociale, adresse, âge, etc. voir Figure 9). Ces informations sont utilisées pour vérifier l'identité, la solvabilité et le dossier de crédit dudit utilisateur.

Le formulaire suivant demande à l'utilisateur les informations sur la maison qu'il souhaite acheter. Il doit y inscrire l'adresse de la maison pour vérifier son existence et sa valeur. Le prix de vente de la nouvelle maison est également demandé pour le comparer avec l'estimation de la maison afin d'éviter tout problème venant des vendeurs. Finalement, on recueille une estimation des dépenses habituelles (électricité, chauffage, téléphone, etc.). Ces dépenses doivent être soustraites de l'avoir mensuel de l'utilisateur-emprunteur.

NetSA offre à son utilisateur deux autres formulaires de façon à recueillir les dépenses et les gains de l'utilisateur. À l'aide de ces informations, NetSA est en mesure de calculer combien l'utilisateur peut payer chaque mois pour la maison

FIG. 9 – Page pour information personnel.

qu'il désire acheter. Ceci indiquera s'il y a un risque de problèmes financiers pour l'utilisateur en ajoutant cette charge monétaire supplémentaire.

Le dernier formulaire (voir Figure 10) est un formulaire pour aider l'utilisateur à choisir le type de prêt hypothécaire qui lui convient le mieux. Le formulaire débute par un petit sondage qui, selon les réponses données, conseillera l'utilisateur sur le type de prêt hypothécaire le plus rentable pour lui.

S'il le désire, l'utilisateur peut choisir lui-même le type de prêt qu'il veut. Il indique le montant de son prêt, la durée totale du prêt, le type désiré, la durée du contrat, le dépôt initial et les assurances désirées.

Les informations recueillies par les formulaires sont par la suite acheminées à l'agent utilisateur. Ce dernier construit un message KQML contenant l'information venant des formulaires et le transmet à l'agent superviseur en charge de la vente aux enchères.

Le processus de vente aux enchères utilisant le meilleur prix peut alors commencer. Nous avons simulé cette enchère entre trois banques, en vue de procurer à l'utilisateur-emprunteur le meilleur taux d'hypothèque. Pour cela, l'utilisateur emprunteur interagit avec le système NetSA via l'agent utilisateur et demande à un agent-encanteur (appelé aussi superviseur en charge de l'enchère) de lui trouver la meilleure offre. Les trois banques sont elles-mêmes simulées par trois systèmes NetSA et capables de faire des offres à l'agent superviseur-encanteur. Pour l'élaboration d'une offre, chacun des agents superviseurs attachés à une banque

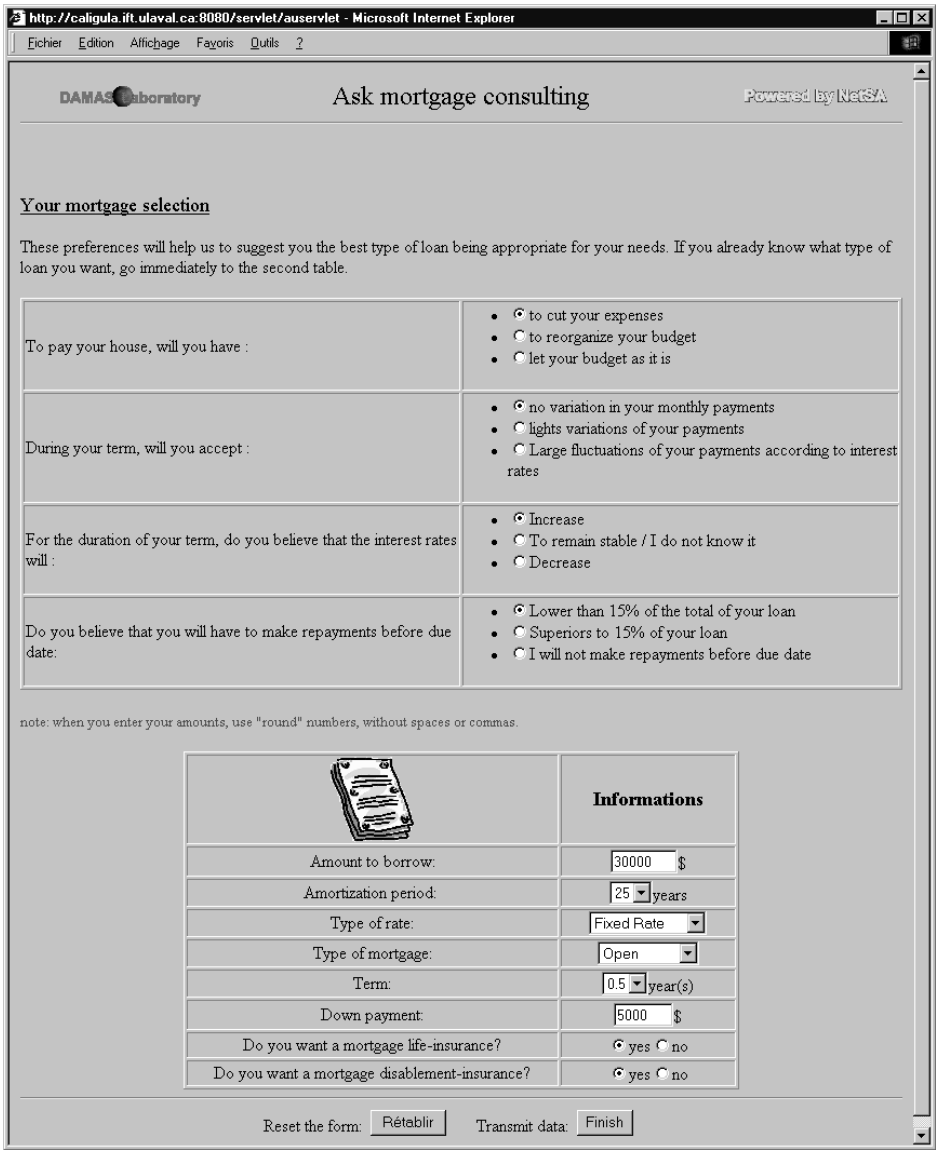

FIG. 10 – Page pour la sélection de l'hypothèque.

consulte ses agents ressources qui lui sont rattachés afin d'obtenir les informations essentielles à l'élaboration d'un bon prêt hypothécaire. Ces informations peuvent être : (1) le dossier de crédit, (2) l'estimation de la maison, (3) les promotions en vigueur, (4) le coût des assurances, (5) les taux d'intérêt, etc.

Le superviseur en charge de l'enchère comptabilise ensuite toute les offres en analysant le coût mensuel, les économies réalisées et le coût total du prêt. Le superviseur annonce à l'utilisateur par l'intermédiaire de l'agent utilisateur, le gagnant ainsi que la position des autres participants comme montré en Figure 11. L'utilisateur reste toujours libre de choisir l'institution financière avec laquelle il veut faire affaire.

FIG. 11 – Page représentant un exemple d'enchères entre trois banques .

4.4 Résultats de la simulation des enchères

Lors des premiers essais sur la compétition entre banques, les plans s'exécutaient de manière séquentielle. Ce type de traitement comportait un désavantage dans la mesure où l'agent superviseur ne pouvait communiquer qu'avec un

seul agent à la fois, ce qui rendait le traitement extrêmement lent. Nous avons alors convenu d'exécuter les plans de manière concurrente. Ainsi, lorsqu'on veut plusieurs informations dispersées sur plusieurs agents, on utilise alors un plan pour chaque agent qui détient l'information désirée. Une telle stratégie nous a alors donné une augmentation des performances de calcul dans la mesure où nous avons pu éliminer une grande partie de la latence due aux communications entre agents.

Nous avons fait des comparaisons entre un calcul de prêt hypothécaire effectué avec une exécution séquentielle et une exécution concurrente des plans. Le graphique de la Figure 12 montre les résultats obtenus. Comme nous pouvons le constater, la distribution concurrente est supérieure à la distribution séquentielle. Ces résultats s'expliquent par la réduction des attentes inutiles provoquées par la distribution séquentielle. Ainsi, au lieu d'envoyer un message et d'attendre sa réponse, il est préférable d'envoyer tous les messages en même temps. Ceci surcharge évidemment un peu plus le réseau, mais la baisse de performance résultante est moindre que le temps d'attente provoqué par la distribution séquentielle des messages.

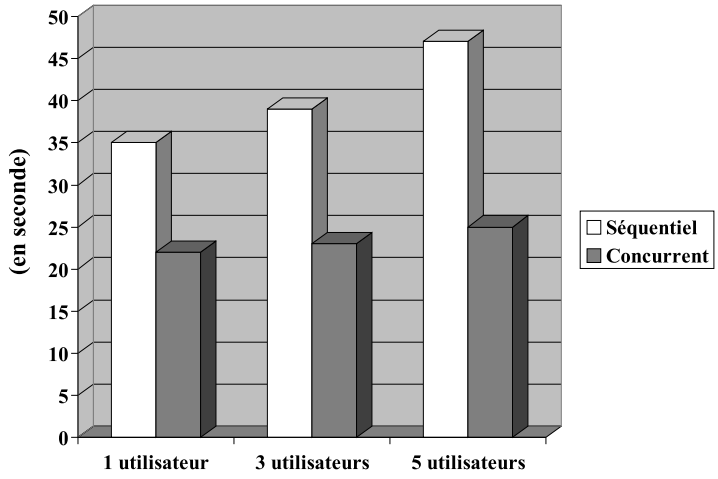

FIG. 12 – Résultats de la simulation des enchères entre banques.

Avec cette technique, nous pouvons actuellement satisfaire jusqu'à 15 utilisateurs en même temps sans que le système ait à souffrir du point de vue performance. Au-dela de ce nombre, il serait préférable d'utiliser plusieurs architectures

NetSA en vue d'alléger la charge de la couche intermédiaire tout en augmentant la fiabilité du système. Une telle architecture complexe exigera certainement une certaine coordination, via les communications KQML, des couches intermédiaires des architectures NetSA individuelles et ce, en fonction de la charge de travail.

Les premiers essais ont montré que le temps de réponse se trouve bien rallongé (en moyenne 5 fois plus) lorsqu'on utilise la recherche classique au lieu de NetSA. En effet, pour les mêmes enchères que celles décrites précédemment, une recherche web classique (sans utiliser un outil comme NetSA) prendrait beaucoup plus de temps, dans la mesure où il faudrait aller chercher soi-même les meilleurs banques et faire ensuite le calcul pour sélectionner le ou les meilleures banques, ce qui bien entendu rallonge le temps de réponse.

Les mêmes essais ont révélé une certaine crainte quant à la divulgation des informations comme le salaire, les crédits, etc. Les utilisateurs auraient tendance à ne fournir ce type d'information que si on leur assure une certaine confidentialité. Nous sommes conscients d'un tel problème et nous nous efforçons de le résoudre dans les prochains mois.

La facilité d'utilisation via les agents utilisateurs a été fortement appréciée par la douzaine de gens soumis aux tests. La forte majorité d'entre eux (dix d'entre eux) ont apprécié le fait de dialoguer avec NetSA via un browser.

Nous avons ensuite montré NetSA à sept spécialistes informaticiens (quatre venant des banques, un du domaine médical, un des assurances, et un de la bibliothèque digitale de l'Université Laval), tous ont reconnu le potentiel de NetSA et ont mis de l'avant ses avantages : (a) portabilité ; (b) réutilisabilité ; (b) flexibilité ; (c) interopérabilité (en particulier avec les systèmes hérités), etc. Comme utilisateurs potentiels, ils ont surtout fait ressortir la prise en compte de la dynamique au niveau d'un environnement changeant où on pourrait avoir des agents qui s'inscrivent et se désinscrivent auprès de l'agent ‹‹pages jaunes››.

Dans le cas de l'architecture NetSA, la réutilisabilité constituait un objectif de départ. Dès lors, cette architecture a été conçue comme une coquille d'agents communiquant via KQML. Le concepteur d'un système NetSA doit alors extraire les connaissances nécessaires à son application et ‹‹remplir ›› ladite coquille avec ces connaissances, afin de l'adapter à son application, qu'elle soit du type bibliothèque digitale, gestion d'assurances, banques, ou tout autre application faisant intervenir des sources d'information distribuées hétérogènes (DBs, HTML, BDs héritées, etc.).

5 Travaux Connexes

Les travaux connexes à NetSA peuvent être regroupés sous deux formes d'architectures : celles qui utilisent et celles qui n'utilisent pas de médiateur. Retsina est une architecture qui n'utilise pas de médiateur, alors que InfoSleuth et UMDL utilisent un médiateur. Dans ce qui suit, nous allons présenter l'une et l'autre forme de ces deux d'architectures en les comparant à NetSA.

5.1 Architectures sans médiateur : l'exemple de Retsina

Les auteurs de Retsina [SPW+96, PSS01] ont développé un ensemble d'agents logiciels coopérants de manière asynchrone pour la quête d'information et pour l'intégration de prises de décisions variées, telles que l'aide à la décision dans les organisations, la gestion de porte-feuille d'actions, etc. Comme toute architecture multiagent, Retsina comporte divers agents remplissant chacun une fonction spécifique.

Agents interfaces : Les agents interfaces interagissent avec les utilisateurs. Différentes tâches leur sont attribuées, comme par exemple : (a) recueillir l'information minimum nécessaire pour initier la résolution de problèmes ; (b) présenter les résultats obtenus ; (c) demander des informations supplémentaires si nécessaire durant la résolution de problèmes et ; (d) demander à l'utilisateur certaines confirmations.

Agents-tâches : Les agents de ce type s'occupent de l'aide à la décision en formulant et en exécutant des plans pour la résolution du problème visé. Ils ont une bonne connaissance d'un domaine particulier et peuvent aider les autres agents sur ce domaine. Un agent de tâches est apte à (a) recevoir les tâches déléguées par l'utilisateur à l'agent interface ; (b) interpréter la tâche et en extraire le but ; (c) construire un plan pour satisfaire ce but ; (d) identifier les besoins en information des sous-buts du plan construit ; (e) décomposer et exécuter le plan construit.

Agents d'informations : L'agent d'informations fournit un accès intelligent aux données hétérogènes et réparties. Il est capable d'extraire l'information pertinente, résoudre les conflits et faire la fusion de certaines données. Cet agent est aussi capable de : (a) fournir de l'information au sujet de la source auquelle il est attaché ; (b) répondre à des requêtes périodiques et répétitives sans consultation de la source d'information ; (c) surveiller l'évolution d'une donnée pour ensuite avertir un agent requérant.

Retsina, et plus généralement le type d'architectures qu'elle représente, semble être une bonne architecture pour la recherche d'information. D'ailleurs, par certains aspects, NetSA ressemble à Retsina (agents interfaces, agents information, etc.). Celle-ci comporte cependant certaines lacunes au niveau de l'optimisation des communications. Entre autres, l'absence d'un agent intermédiaire pouvant diriger immédiatement les agents vers la bonne ressource, oblige les agents de Retsina à instaurer à chaque requête un processus de négociation par contrats [Smi80], ce qui a tendance à surcharger le réseau. Cette absence d'agent intermédiaire rend également l'ouverture du système beaucoup plus complexe. Un nouvel agent désirant entrer sur Retsina doit aviser tous les agents inclus dans l'architecture de sa venue. Les auteurs ont également fait mention d'un problème de goulot d'étranglement au niveau des agents-tâches qui est probablement dû à la complexité de l'architecture interne des agents.

5.2 Architecture avec médiateur : InfoSleuth et UMDL comme exemples

5.2.1 Carnot+InfoSleuth

En 1990, la compagnie Microelectronics and Computer Technology Corporation (MCC) a eu pour mission de résoudre un problème d'intégration d'informations dans des bases de données hétérogènes. Ce projet a reçu le nom de Carnot et s'est terminé en 1995 [WCH+93]. Carnot pouvait d'ores et déjà gérer le flux d'informations, accéder à des bases de données hétérogènes et faire de la découverte d'informations pour certaines entreprises. Les auteurs de Carnot se sont cependant rendus compte que le manque de structure dans la présentation de l'information sur Internet constitue un énorme défi du point de vue découverte, accès, mise à jour et analyse de l'information. Ceci les a alors amené à chercher à améliorer Carnot en vue de lui permettre la recherche d'informations dans des sources géographiquement distantes et dynamiques comme dans le cas de la toile. InfoSleuth [BBB+96, BBB+98] a donc pris la relève de Carnot en 1995. Pour développer InfoSleuth, les chercheurs de MCC ont utilisé des technologies standardisées comme : (a) KQML (Knowledge Query and Manipulation Language) [FFMM94] ; (b) KIF (Knowledge Interface Format) [Gea92] ; (c) HTTP (Hyper Text Transfert Protocol) [BLFF96] ; (d) Java [GM95]. Ceci procure à InfoSleuth un haut niveau d'ouverture du système, de flexibilité et d'extensibilité. L'architecture générale d'InfoSleuth est divisée en trois couches principales : (1) la couche agent, (2) la couche sémantique (ontologies) et (3) la couche application. Ces trois

couches sont détaillées ci-après.

***La couche agent* :** La couche inférieure de l'architecture est la couche agent. Cette couche est composée d'agents coopératifs où les règles d'induction sont implantées en LISP [AGST75], CLIPS [Ril91], LDL++ [Zan91] et en Java. Ces agents annoncent d'abord leurs services et font par la suite des requêtes afin de trouver un agent pouvant répondre à un besoin spécifique. Un besoin peut être divisé en plusieurs sous-requêtes qui sont ensuite distribuées aux agents appropriés. Des réponses provenant de ces derniers sont finalement réunies par l'agent demandeur pour l'obtention de la réponse à la requête originale.

***La couche sémantique* :** Cette couche indique les agents coopératifs ayant le même vocabulaire et le même modèle sémantique qui sont capables d'interagir dans un domaine particulier. Par exemple, dans le cas où le travail se fait sur le domaine des finances, alors les agents aptes à communiquer entre eux sont seulement ceux utilisant des termes propres à la finance.

***La couche application* :** Cette couche couvre trois régions principales soit : (1) la recherche de ressources (par exemple trouver les agents et les bases de données relatives au domaine de l'ontologie), (2) la recherche et l'analyse de gabarits (par exemple par data-mining), (3) la collaboration, la recherche et l'exécution.

Ces régions s'interrelient et se chevauchent. D'un point de vue architecture, le coeur d'InfoSleuth peut être vu comme un nuage d'agents (Voir Figure 13). L'usager envoie des requêtes et reçoit des réponses de ce nuage par l'intermédiaire d'un agent utilisateur. Ce nuage puise ces informations des agents ressource qui communiquent directement avec des bases de données. La communication entre agents se fait à l'aide du protocole KQML [FFMM94] en utilisant le format KIF [Gea92] dans son champ contenu.

5.2.2 UMDL

En 1994, une équipe multi-disciplinaire de l'université du Michigan propose de créer une bibliothèque digitale évolutive [DKB97, DKB98]. Le terme ‹‹bibliothèque numérique ›› est le nom générique pour décrire des structures qui fournissent aux êtres humains des accès intellectuels et physiques aux énormes réseaux mondiaux de l'information dans des formats numériques multimédias.

La mission fondamentale des bibliothèques a été de conserver et de fournir un accès intellectuel et physique aux écrits. Bien que cette mission fondamen-

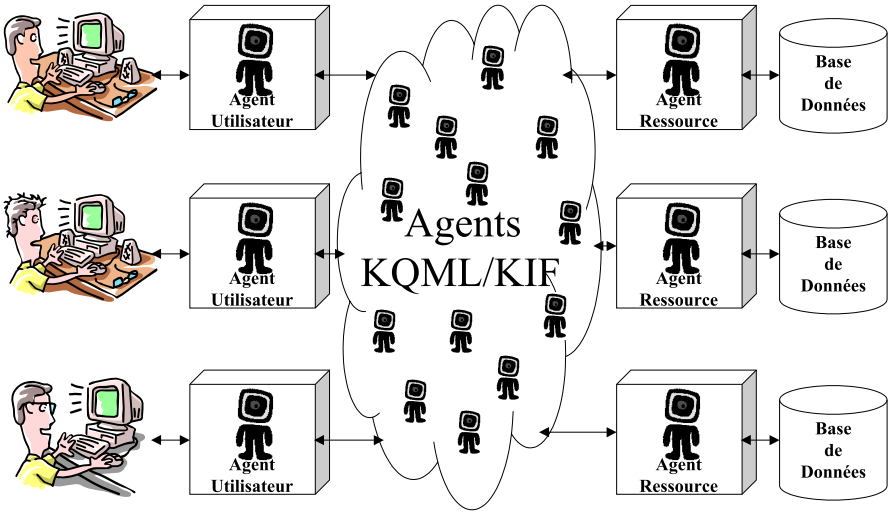

FIG. 13 – Nuage d'agents d'InfoSleuth [BBB+96]

tale continue à être très importante, la représentation de l'information a changé radicalement (document électronique, matériel optique et magnétique, etc.). Ce changement massif de la représentation change la structure de classement, de recherche, d'utilisation et de réutilisation de l'information développée par les bibliothèques classiques. Ceci dit, une bibliothèque digitale est en mesure : (a) de fournir de l'information peu importe l'endroit et le moment ; (b) de fournir l'accès à des collections d'information multimédia construites sur l'intégration de textes, d'images, de graphiques, de sons, de vidéos et d'autres médias ; (c) d'être assez convivial pour assister l'utilisateur dans sa recherche d'information, (d) d'être le cœur d'une nouvelle technologie d'auto-apprentissage et de recherche abaissant les barrières géographiques et temporelles.

Les chercheurs de l'Université du Michigan ont mis en avant une architecture multiagent, appelée UMDL pour contribuer à une bibliothèque digitale évolutive. Cette architecture est composée de 4 types d'agents logiciels, à savoir :

Agent utilisateur : L'agent utilisateur fournit les stratégies de recherche aux utilisateurs d'UMDL. Il guide l'utilisateur en lui donnant des astuces pour chercher la meilleure information désirée ainsi que son meilleur format (texte, image, son, vidéo, etc.).

Agent d'exécution de requêtes : Le développement d'un paradigme de requête permet à l'utilisateur d'extraire l'information désirée facilement par la construction de requêtes complexes dans un environnement distribué. L'agent d'exécution de requêtes garde une trace des requêtes de l'utilisateur pour améliorer ses performances lors des prochaines requêtes. Cette trace lui permet de connaître les préférences de l'usager.

Médiateur : Le médiateur a pour but de contrôler les interactions entre les agents utilisateurs et les agents collecteurs. Souvent, les médiateurs forment une architecture hiérarchique. Les interactions entre les médiateurs ressemblent fortement à la résolution de problèmes coopératifs.

Agent collecteur : Les agents collecteurs sont reliés aux ressources. Ils sont en charge de la liaison entre les informations (base de données, collection d'image, vidéothèque, catalogue, etc.) et le système multiagent.

De par sa conception, NetSA ressemble à InfoSleuth et à UMDL, puisque toutes les trois sont bâties sur le principe du nuage d'agents (Voir Figure 13), avec bien entendu un des agents de ce nuage jouant le rôle de médiateur. Toutefois, la ressemblance s'arrête ici, dans la mesure où les techniques mises en avant pour développer NetSA sont différentes de celles utilisées par InfoSleuth et UMDL. En

particulier, NetSA offre à la fois médiation et stratégies d'interaction issues des réseaux à contracts, tous les deux propres à elle, et faisant intervenir le langage POPA.

6 Conclusion et futurs développements

Dans ce papier, nous nous sommes intéressés à la conception d'une ‹‹coquille ›› de système multiagent pouvant être utilisée pour les applications opérant sur des sources d'information hétérogènes placées dans un environnement ouvert et dynamique (comme par exemple l'internet). Cette exigence de ‹‹réutilisabilité ››, nous a été dictée par le fait que la complexité engendrée par un tel système est parfois supérieure aux grands projets informatiques. De ce fait, le coût de production de ce genre de systèmes est énorme en personne-mois tant au niveau de la conception, que de la réalisation et de la vérification. La réutilisabilité est un moyen pour réduire la charge de travail et par le fait même, le coût de production et c'est ce que nous avons choisi.

Hormis cet avantage crucial, NetSA offre plusieurs autres avantages. C'est ainsi qu'elle est (i) *portable* (puisqu'elle est programmée en Java) ; (ii) *flexible* (facilement adaptable à d'autres applications grâce au langage POPA) et, (iii) *interopérable* (en particulier avec les systèmes hérités via ses agents ressources).

Les premiers essais effectués ont montré que NetSA : (a) peut facilement satisfaire jusqu'à 15 utilisateurs (au delà il faudra penser utiliser plusieurs architectures NetSA) ; (b) est très efficace par rapport à une recherche web classique ; (c) est très facile d'utilisation ; (d) convient très bien aux applications faisant intervenir des bases de données héritées.

Bien que NetSA soit opérationnelle, il y a toujours place pour son amélioration. C'est ainsi que nous prévoyons d'inclure deux nouveaux types d'agents : un agent-serveur d'ontologie et un agent d'extraction de connaissances.

Agent-serveur d'ontologie : Le rôle d'un tel serveur consistera à fournir à de nouveaux agents entrant sur le système NetSA, des définitions propres à l'ontologie utilisée. Il permettra dès lors d'éviter des communications inutiles que pourraient engendrer des agents nouveaux à la recherche de sens. Le serveur d'ontologie fera partie de la couche centrale de NetSA.

Agent d'extraction des connaissances : L'information stockée dans les grandes bases de données est généralement immense et elle pourrait dans bien des cas permettre d'extraire des connaissances. Pour cela, il faudrait adjoindre à NetSA un agent d'extraction de connaissances.

Références

[AGST75] R. A. Amsler, E. M. Greenawalt, J. Slocum, and M. Tyson. *LISP Reference Manual CDC - 6000*. University of Texas at Austin, Computation Center, CCUM 2, Austin, December 1975.

[Auc98] Auction universe, 1998. http ://www.auctions.com.

[BBB+96] R. Bayardo, W. Bohrer, R. Brice, A. Cichocki, G. Fowler, A. Helal, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh, and D. Woelk. Semantic integration of information in open and dynamic environments. Technical Report MCC-INSL-088-96, MCC, October 1996.

[BBB+98] R. J. Bayardo, W. Bohrer, R. Brice, A. Cichocki, J. Fowler, A. Helal, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh, and D. Woelk. Infosleuth : agent-based semantic integration of information in open and dynamic environments. In Huhns M. N. and Singh M. P., editors, *Reading in Agents*, pages 205–216, SF, CA, 1998. Morgan Kaufmann.

[BCdK02] H. Benameur, B. Chaib-draa, and P. Kropf. Multi-items auctions for automated negotiation. *Information and Software Technology*, (à venir), 2002.

[BLFF96] T. Berners-Lee, R. Fielding, and H. Frystyk. Hypertext transfer protocol - http/1.0. Request for comment : 1945, May 1996.

[CdJM02] B. Chaib-draa, I. Jarras, and B. Moulin. Systèmes multiagents : principes généraux et applications. In J .P. Briot and Y. Demazeau, editors, *Principes et architectures des systèmes multi-agents*, Paris, France, à venir en 2002. Collection IC2, Hermes Science Publication.

[CdV98] B. Chaib-draa and D. Vanderveken. Agent communication language : Towards a semantics based on success, satisfaction and recursion. In *Agents : Theories, Architectures and Languages (ATAL'98)*, Paris, Fr, 1998.

[CT98] M. Côté and N. Troudi. NetSA : une architecture multiagent pour la recherche sur internet. *Expertise Informatique*, 3(3), 1998.

[Côt99] M. Côté. NetSA : une architecture multiagent et son application aux services financiers. Technical report, Mémoire de maîtrise du département d'informatique, Université Laval., 1999.

[DKB97] E. H. Durfee, D. L. Kiskis, and W. P. Birmingham. The agent architecture of the university of michigan digital library. *IEE/British Computer Society Proceedings on Software Engineering*, 144(1), 1997.

[DKB98] E. H. Durfee, D. L. Kiskis, and W. P. Birmingham. The agent architecture of the university of michigan digital library. In Huhns M. N. and Singh M. P., editors, *Reading in Agents*, pages 98–108, SF, CA, 1998. Morgan Kaufmann.

[FFMM94] T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an agent communication language. In ACM Press, editor, *Proceedings of the Third International Conference on Information and Knowledge Management*, November 1994.

[FR93] R. A. Feldman and M. Rajnish. Auctions theory and applications. *IMF Staff Papers*, pages 485–511, September 1993.

[Gea92] M. R. Genesereth and R. E. Fikes et al. Knowledge interchange format version 3 reference manual. Technical Report Logic-92-1, Stanford University Logic Group, 1992.

[GM95] J. Gosling and H. McGilton. The Java(tm) language environment : A white paper. Technical report, Sun Microsystems, 1995.

[JSW98] N. R. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. In *Autonomous Agents and Multi-Agent Systems*, pages 7–38, Boston, 1998. Kluwer Academic Publishers.

[KH98] D. Kuokka and L. Harada. Matchmaking for inrmation agents. In Huhns M. N. and Singh M. P., editors, *Reading in Agents*, pages 91–97, SF CA, 1998. Morgan Kaufmann.

[MCd96] B. Moulin and B. Chaib-draa. An overview of distributed artificial intelligence. *Foundations of Distributed Artificial Intelligence,O'Hare, G. and Jennings, N. (eds), Wiley Inter-Science*, pages 3–55, 1996.

[Mic98] Sun Microsystem. The Java servlet API, 1998. http ://www.micro.rmc.ca :8080/system/doc/servlets/api.html.

[Net97] Netscape. *JavaScript Reference*. Netscape Communications Corporation, 1997.

[PSS01] M. Paolucci, O. Sheory, and K. Sycara. Interleaving planning and execution in a multiagent team planning environment. *Electronic Transactions of Artificial Intelligence*, (à venir), 2001.

[Ril91] G. Riley. Clips : An expert system building tool. In *Proceeding of the Technology 2001 Conference*, San Jose, CA, December 1991.

[Sak00] Y. Sakurai. An efficient approximate algorithm for winner determination in combinatorial auctions. In *Proceedings of the 2nd ACM Conference on Electronic Commerce-EC'00*, Minneapolis, Minnesota, 2000.

[Smi80] R. G. Smith. The contract net protocol : high-level communication and control in a distributed problem solver, 1980.

[SPW+96] K. Sycara, A. Pannu, M. Williamson, D. Zeng, and K. Decker. Distributed intelligent agents. *IEEE Expert*, pages 36–46, December 1996.

[Vic61] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, Vol. 16 :8–37, March 1961.

[WCH+93] D. Woelk, P. Cannata, M. Huhns, W. M. Shen, and C. Tomlinson. Using carnot for enterprise information integration (project synopsis). In *Parallel and Distributed Information Systems (PDIS '93)*, pages 133–137, Los Alamitos, Ca., USA, January 1993. IEEE Computer Society Press.

[Zan91] C. Zaniolo. The logical data language (ldl) : An integrated approach to logic and databases. Technical Report Technical Report STP-LD-328-91, MCC, 1991.