

Vers des agents mobiles cognitifs

Zakaria Maamar

Groupe Interfonctionnement

Section Technologie des systèmes d'information

Centre de recherches pour la défense Valcartier

2459 boul. Pie-XI Blvd Nord Val-Bélair, QC G3J 1X5, Canada

zakaria. Maamar@drev.dnd.ca

Département d'informatique et Centre de recherche en géomatique

Université Laval, Ste-Foy, QC G1K 7P4, Canada

RESUME: Cet article traite de la problématique des agents mobiles. Ces agents ont la capacité de se déplacer entre des systèmes hôtes et par conséquent de nager à travers les réseaux de communication. De plus, un agent mobile conserve son état interne pendant la migration afin de pouvoir reprendre ses opérations d'exécution une fois arrivé à destination. Les arts mobiles font l'objet de plusieurs projets allant de l'intelligence artificielle aux systèmes distribués. Ce pendant, peu d'attention eut accordé aux agents mobile cognitifs qui seraient capables, indépendamment de leurs concepteurs, de décider quand la migration est préférable au traitement à distance. L'objectif du projet LAMPA est de concevoir des agents mobiles cognitifs.

MOTS-CLES : agents mobiles cognitifs, planification, coût, itinéraire.

1. Introduction

Le paradigme des agents mobiles constitue une nouvelle alternative au traitement traditionnel de type client/serveur, en particulier, et au traitement distribué, en général. L'idée de base de ce paradigme est de fragmenter le programme de l'agent en un ensemble de portions de codes capables d'être exécutées de manière séquentielle sur divers systèmes hôtes. Chaque système appartient à un nœud d'un réseau de communication, comme l'Internet. Avant d'exécuter chaque portion de code, l'agent se déplace ou migre à destination du site approprié. Généralement, la fragmentation du code et par conséquent, la migration dépendent de la disponibilité des ressources nécessaires à l'exécution de l'agent sur le site de migration.

Plusieurs travaux de recherche [3, 5, 6, 7, 11, 8, 13, 14, 16] proposent diverses approches de développement d'agents mobiles. Cependant, il n'existe pas de standards dans le domaine des agents mobiles, que ce soit au niveau des méthodes de conception, des langages de programmation, des mécanismes de spécification des agents, etc. De plus, la majorité de ces travaux de recherche (hormis [11] qui traite seulement de la planification des agents, sans négociation) ont mis l'emphasis sur la spécification statique des opérations d'un agent quand il décide de se déplacer d'un site à un autre site. Dans ce cas, les agents sont *réactifs* et n'ont pas la capacité de raisonner à propos de leurs environnements, d'adapter leurs comportements et d'inférer de nouvelles connaissances s'il y a lieu.

À l'Université Laval, notre groupe de recherche a entrepris un nouveau projet, intitulé LAMPA pour LAval Mobile and Planning Agents, qui s'intéresse au développement d'agents mobiles, dotés de mécanismes de planification. L'objectif du projet LAMPA est d'utiliser des agents *cognitifs* qui évoluent selon leurs propres connaissances et décident quand la migration est préférable au traitement à distance.

Dans cet article, nous analysons l'organisation et la spécification d'un environnement d'agents mobiles cognitifs. L'organisation concerne l'adaptabilité de l'architecture de cet environnement quand les agents se déplacent, alors que la spécification concerne la description des connaissances des agents et de leurs opérations de planification et de négociation. Dans l'objectif d'assigner aux agents de l'environnement LAMPA des capacités de planification et de négociation, nous utilisons plusieurs concepts, tels que plan, service, capacité et place. Nous analysons également la question d'invitation, qui est un type particulier de migration. Dans l'environnement LAMPA, un agent est localisé dans une place qui appartient à un système hôte du réseau. Cet agent a la capacité de construire des plans de tâches et d'identifier les ressources nécessaires à la réalisation de ces plans. Si un agent ne peut pas exécuter un plan par lui-même (c'est-à-dire s'il manque de ressources), il peut requérir les services d'autres agents. Ces agents sont vus comme des ressources. Selon la localisation de ces agents et des coûts de migration, un agent peut soit décider de migrer pour rencontrer certains de ces agents, soit invoquer à distance les services de certains de ces agents ou même les inviter à venir le rejoindre sur son système hôte.

L'article est organisé de la manière suivante. La Section 2 présente les caractéristiques principales de l'environnement LAMPA. La Section 3 présente une approche de construction et d'optimisation des itinéraires des agents de l'environnement LAMPA. La Section 4 discute des perspectives d'implantation dans l'environnement LAMPA. Finalement, la Section 5 conclut l'article.

2. L'environnement LAMPA

Cette section présente diverses caractéristiques de l'environnement LAMPA, en termes d'Architecture, de concepts, de modes de fonctionnement et de structures d'agents.

2.1. Concepts

Divers concepts sont déjà utilisés dans le domaine des agents mobiles, tels que migration, autorité, itinéraire, rencontre, etc. Dans cette section, nous présentons d'autres concepts, dans l'objectif de les appliquer à l'environnement LAMPA. Ces concepts sont les suivants: agent logiciel¹, service, capacité, plan, place et invitation.

Agent logiciel : entité autonome et orienté-but qui coopère avec d'autres agents (humains ou logiciels) pour accomplir ses tâches. Dans l'environnement LAMPA, un agent logiciel construit et exécute des plans de réalisation afin de satisfaire les besoins des usagers. Afin de supporter les caractéristiques de l'environnement d'agents mobiles LAMPA, deux types d'agents sont identifiés:

- Les agents-serveurs, qui sont *stationnaires*, offrent des services aux usagers et délèguent la réalisation de ces services à d'autres agents (clients) du même système hôte. De plus, un agent-serveur assure la cohérence interne de son système hôte, en gardant trace des agents qui opèrent dans ce système.
- Les *agents-clients*, qui deviennent *mobiles* si c'est nécessaire, satisfont les besoins des usagers. Un agent-client peut construire et exécuter des plans, négocier avec d'autres agents clients, se déplacer vers d'autres systèmes hôtes, déléguer à distance l'exécution de plans et inviter d'autres agents à migrer. Un plan peut impliquer un ou plusieurs agents-clients du même système hôte ou d'autres systèmes distants. Dans l'environnement LAMPA, le plan construit par un agent-client est vu comme son itinéraire de réalisation (cf. Section 3).

Les agents-clients et serveurs sont l'ossature de l'environnement d'agents mobiles LAMPA. Cependant, pour qu'un agent-client puisse identifier les "meilleurs" autres agents-clients (ceux-ci offrent les ressources nécessaires) et par conséquent, optimiser son itinéraire avant la réalisation

de ses plans, cet agent-client doit avoir une vue générale de son environnement afin d'être capable de localiser les ressources, de déterminer leurs coûts d'utilisation, les coûts d'accueil des agents et connaître les risques de déplacements. Pour ces raisons, nous introduisons un autre type d'agents, appelé Broker, dans chaque système hôte. En fait, un agent-Broker est un intermédiaire entre les agents-clients qui s'annoncent comme des ressources pouvant être utilisées; dans des plans et les agents-clients qui recherchent des ressources pour exécuter leurs plans. Les agents-Brokers des divers systèmes hôtes s'échangent des informations à propos des services des agents-clients de leur système respectif.

Service: un système hôte intègre différents services qui sont vus comme des opérations à exécuter pour satisfaire certains besoins. Trois catégories de services existent dans l'environnement LAMPA:

- Les Services-usager sont supportés par des agents-serveurs et sont invoqués par des usagers. Un exemple d'un service-usager peut être une recherche d'informations à travers le réseau Internet.
- Les Services-mobilité sont également supportés par des agents-serveurs. Ces services sont utiles pour envoyer des agents-clients à travers le réseau vers d'autres systèmes hôtes et pour installer ces agents afin de reprendre leurs opérations une fois arrivés.
- Les Services-agent caractérisent les opérations que les agents-clients sont capables d'exécuter pour la réalisation des plans. Les services-agent sont mis à la disposition des agents-clients, une fois qu'ils ont été annoncés au niveau de l'agent-Broker. Des exemples de services-agent incluent l'accès à une base de données, l'envoi d'un message électronique, etc.

Afin de satisfaire un service-usager, un agent-client construit des plans de réalisation et utilise obligatoirement ses propres services agent, possiblement les services-agent des autres agents-clients et possiblement les services-mobilité des agents-serveurs.

Capacité : un agent doit pouvoir exécuter des opérations, construire des plans et déclarer des services. Pour le moment, un agent-serveur doit principalement aider les usagers à spécifier leurs besoins, assigner les demandes de ces usagers aux agents-clients disponibles, transmettre et recevoir des agents-clients vers et en provenance d'autres systèmes distants et superviser le fonctionnement interne de son système hôte.

Un agent-client doit être capable d'annoncer ses propres services-agent comme ressources, construire des plans de réalisation, les exécuter, déterminer les autres agents-clients locaux ou distants nécessaires (en passant par l'Agent-Broker), formuler ses demandes de déplacement, d'invitation ou de traitement à distance, minimiser les risques et coûts de déplacement et reprendre ses opérations quand il migre vers d'autres systèmes hôtes.

Un agent-Broker doit être capable de recevoir des annonces de services-agent des agents-clients et répondre aux demandes des agents-clients pour déterminer les services-agent appropriés.

Place : un agent évolue dans une place spécifique de son système hôte. Quand un agent-client se déplace il supprime sa place initiée pour aller rencontrer un autre agent-client dans sa propre place. La même approche s'applique quand les agents-clients reçoivent des invitations de déplacement. En fait, une place peut recevoir un ou plusieurs agents.

Plan : un agent client construit des plans afin de satisfaire les besoins des usagers. Dans l'environnement LAMPA, chaque service-usager est associé à un but global. Selon la complexité du but, un agent-client peut le décomposer en un ensemble de sous-buts, jusqu'à ce que chaque sous-but puisse être identifié par un plan de tâches élémentaires (cf. Section 2.4). Si un agent ne peut pas réaliser un plan par lui-même (c'est-à-dire il n'est pas capable d'exécuter certaines tâches du plan), il peut requérir des services-agent complémentaires d'autres

agents-clients du même système ou d'autres systèmes. Les agents requis sont déterminés par un processus de négociation (cf. Section 2.4).

Plusieurs scénarios de fonctionnement existent quand un agent-client demande les services d'autres agents-clients: il peut se déplacer pour rencontrer certains agents; il peut requérir à distance certains agents; ou il peut inviter certains agents à migrer vers son propre système hôte.

Invitation : un agent-client peut devenir mobile pour deux raisons principales: il décide de migrer ou il reçoit une invitation à migrer. L'invitation est un cas particulier de la migration; une invitation est formulée par un agent-client vers un autre agent-client disposant des services-agent requis.

2.2. Architecture

En utilisant les divers concepts présentés dans la Section 2.1, nous avons conçu l'architecture de l'environnement d'agents mobiles LAMPA (Fig. 1). Cette architecture se caractérise par l'utilisation de trois types d'agents (serveur, client, Broker), les places des agents-clients et serveur et les opérations de collaboration entre les agents (lignes en pointillés/continues/discontinues dans la Figure 1). Les interactions entre les agents-clients et serveur peuvent prendre place : dans un seul système hôte (Fig. 1.A), entre deux systèmes hôtes par traitement à distance (Fig. 1.B) et dans un seul système hôte après un déplacement (Fig. 1.C) ou une invitation (Fig. 1.D).

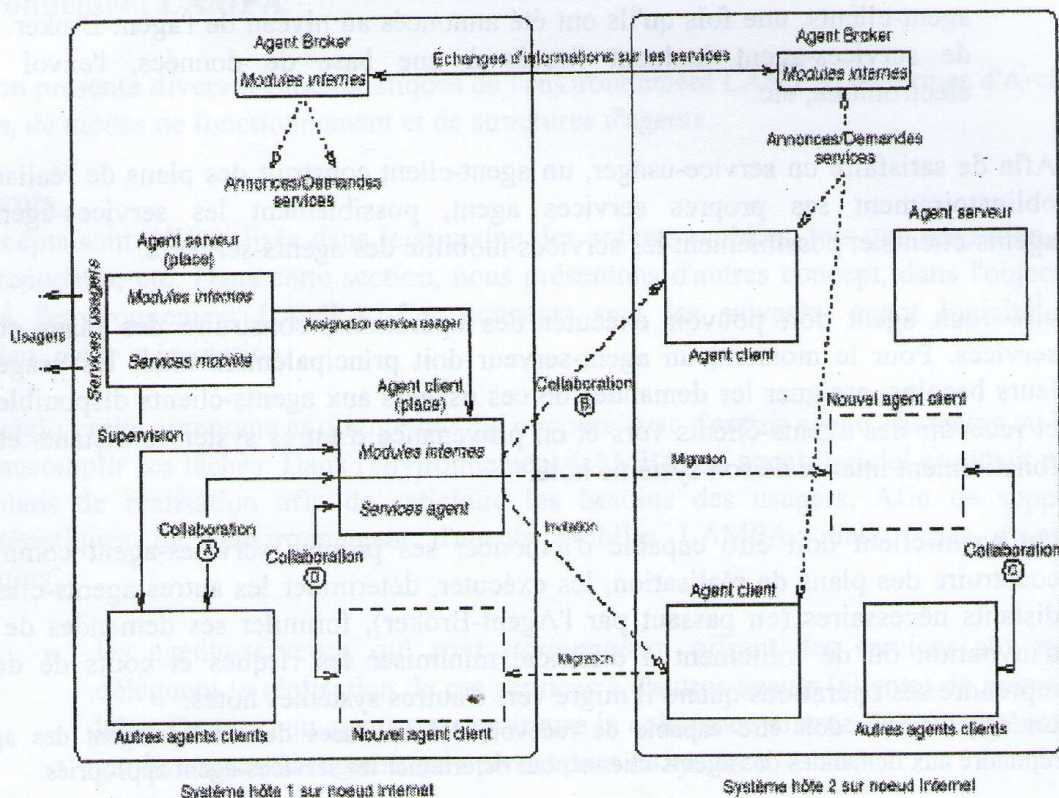


Figure 1. Architecture de l'environnement d'agents mobiles LAMPA

2.3. Mode de Fonctionnement

La Figure 2 montre la chronologie des opérations de réalisation dans l'environnement LAMPA. Le formalisme de représentation en le suivant: un rectangle arrondi représente un système hôte, un rectangle moucheté indique un agent, une ligne orientée identifie une opération exécutée par un agent (la flèche indique l'agent qui supporte l'opération) et finalement les numéros indiquent l'ordre des opérations.

Un usager invoque un service offert par un agent-serveur d'un système hôte (0). Cet agent-serveur sélectionne un agent-client disponible du même système hôte et lui assigne le service (1).

Par la suite, l'agent-client construit des plans de réalisation pour ce service (2). Pour exécuter ces plans, l'agent-client a besoin de ses propres services-agent et possiblement des services-agent des autres agents-clients. Les services-agent requis sont déterminés à la demande de l'agent-client (3) et par l'agent-Broker (4). Une fois que les services-agent nécessaires sont déterminés, l'agent-client adapte ses plans selon les cas suivants: des plans doivent être traités à distance (5.0), cet agent-client doit migrer (6.0, 6.1, 6.2, 6.3, 6.4, 6.5) et d'autres agents-clients doivent recevoir des invitations pour rencontrer cet agent-client dans son système hôte (7.0, 7.1, 7.2, 7.3, 7.4, 7.5, 7.6).

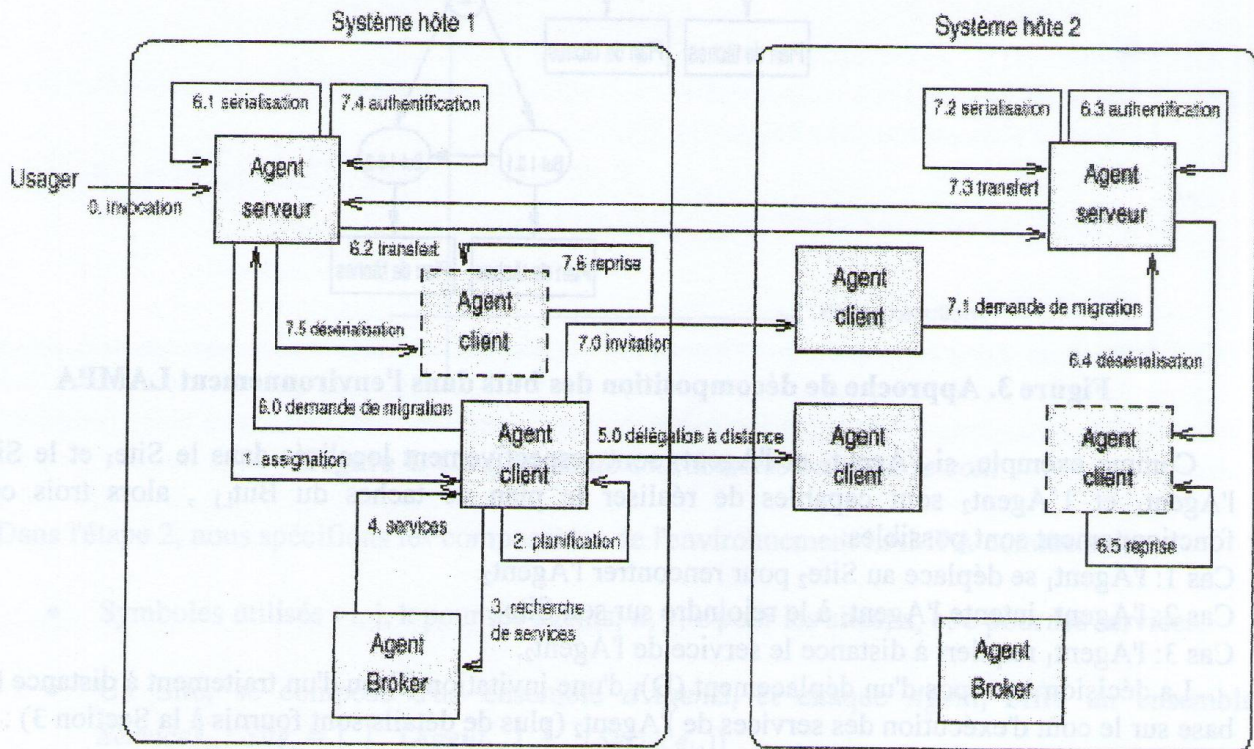


Figure 2. Chronologie des opérations dans l'environnement d'agents mobiles LAMPA

2.4. Décomposition des buts

Il est généralement considéré que la motivation principale de migration d'un agent est l'absence de ressources² dans son système hôte. Cet agent recherche les ressources appropriées dans d'autres systèmes hôtes. De plus, l'autre motivation est la réduction du volume d'informations brutes à transmettre sur le réseau à un usager; il s'agit de pré-traiter ces informations au niveau du système qui les détient pour ne transmettre en fin de compte que les informations pertinentes. Dans l'environnement LAMPA, nous introduisons une nouvelle façon pour les *décisions* de déplacement d'un agent-client. Pour cela, nous caractérisons une opération de déplacement par ses *coûts* et ses *risques*. Pour les besoins de cet article, nous mettons l'accent sur le critère *coût*.

Un agent-client a la capacité de construire les plans de réalisation pour les services usagers. Chaque service-usager est associé à un but global qui est décomposé en un ensemble de sous-buts [9], jusqu'à ce que chaque sous-but soit associé à un plan de tâches (Fig. 3; l'agent-client $Agent_i$ est responsable de la décomposition du But_i). Il arrive que ces plans ne puissent pas être pris en charge de manière individuelle par l'agent-client qui a procédé à la décomposition du but global. En fait, cet agent a besoin de services additionnels qui sont disponibles chez d'autres agents-clients du même système hôte ou d'autres systèmes distants.

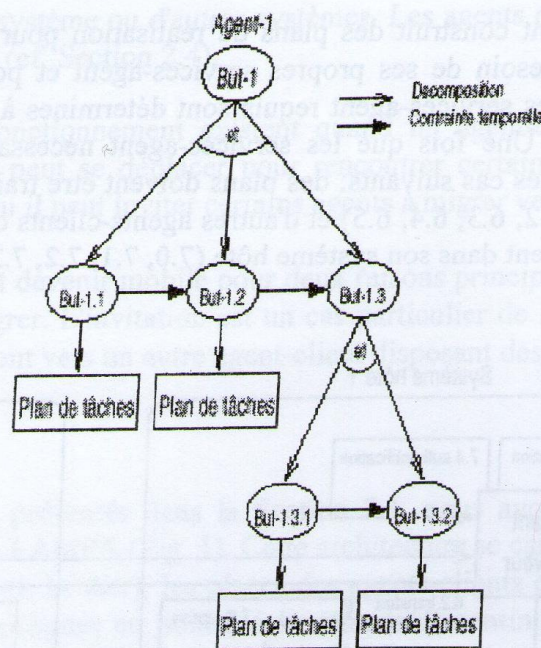


Figure 3. Approche de décomposition des buts dans l'environnement LAMPA

Comme exemple, si l'Agent₁ et l'Agent₂ sont respectivement localisés dans le Site₁ et le Site₂ et l'Agent₁ et l'Agent₂ sont capables de réaliser le plan de tâches du But_{1.1}, alors trois cas de fonctionnement sont possibles:

Cas 1: l'Agent₁ se déplace au Site₂ pour rencontrer l'Agent₂.

Cas 2: l'Agent₁ invite l'Agent₂ à le rejoindre sur son Site₁.

Cas 3: l'Agent₁ requiert à distance le service de l'Agent₂.

La décision à propos d'un déplacement (D), d'une invitation (I) ou d'un traitement à distance (À) se base sur le coût d'exécution des services de l'Agent₂ (plus de détails sont fournis à la Section 3) :

D – Coût $\text{Service}_{\text{Agent2, Site2}}^{21}$ après que Agent₁ se déplace du Site₁ au Site₂
vs.

À – Coût $\text{Service}_{\text{Agent2, Site2}}^{21}$ après que Agent₁ demande à distance Agent₂
Vs.

I – Coût $\text{Service}_{\text{Agent2, Site1}}^{21}$ après que Agent₂ se déplace du Site₂ au Site₁

3. Construction et optimisation des itinéraires des agents mobiles

Dans LAMPA, la fonction "coût" permet à l'agent-Broker d'aider les agents-clients dans leurs prises de décisions, en ce qui concerne leurs *traitements locaux*, leurs *déplacements*, leurs *invitations* et leurs *traitements à distance*. Dans cette section, nous expliquons comment l'agent-Broker identifie les services qui optimisent l'itinéraire de réalisation des plans de tâches que les agents-clients construisent. Pour cela, nous décomposons le «coût» en deux points :

Coût d'accueil d'un agent-client en provenance d'un site distant. Ce coût est à la charge de l'agent-client qui se déplace dans un cas de déplacement et à la charge de l'agent-client qui invite dans le cas d'une invitation.

Coût d'exécution d'un service agent exécuté par un agent-client.

Les étapes suivantes expliquent l'approche proposée pour la construction et l'optimisation des itinéraires de réalisation des plans de tâches.

Dans l'étape 1, nous construisons un système de coordonnées avec les caractéristiques suivantes (Fig. 4) :

Axe-x(0,+∞) = Coût-Exécution (selon traitement-local, déplacement, traitement-à-distance et invitation);
Axe-y(0,+∞) = Coût-Accueil (selon déplacement et invitation);

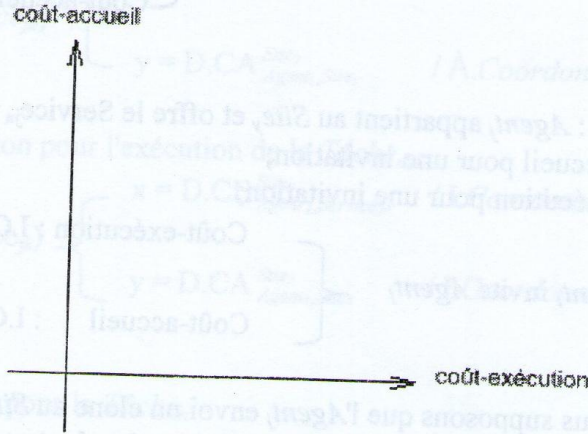


Figure 4. Un système de coordonnées basé sur le coût

Dans l'étape 2, nous spécifions les composantes de l'environnement LAMPA comme suit :

- Symboles utilisés : i, j, k pour les agents; x, y, z pour les sites; a, b, c pour les services.
- Un Site_x se compose d'un ensemble d'Agents $_i$ et chaque Agent $_i$ offre un ensemble de Services $_{ia}$: $\text{Site}_x = \bigcup_i (\text{Agent}_i, \bigcup_a (\text{Service}_{ia}))$
- Un Agent $_i$ du Site_x se caractérise par ses déplacements vers d'autres Sites $_{y,y \neq x}$, ses invitations à d'autres Agents $_{j,j \neq i}$, et ses demandes de traitements locaux ou à distance à d'autres Agents $_{k,k \neq i}$.
- Remarque: si Service $_{ja}$ est identique à Service $_{kb}$ ($j \neq k$ et $\forall a, b$) alors l'Agent $_j$ et l'Agent $_k$ offrent le même service.

Nous énumérons ci-dessous les types d'interaction qui pourraient exister entre un Agent $_i$ et un Agent $_j$ selon le site de l'Agent $_j$.

1. Cas de déplacement (D) : Agent $_j$ appartient au Site $_y$, et offre le Service $_{ja}$

D.CA : coût d'accueil pour un déplacement;

D.CE : coût d'exécution pour un déplacement;

$$\text{Agent}_i \text{ rencontre Agent}_j \left\{ \begin{array}{l} \text{Coût-exécution : D.CE}_{\text{Agent}_j, \text{Service}_{ja}}^{\text{Site}_y} \\ \text{Coût-accueil : D.CA}_{\text{Agent}_i, \text{Site}_x}^{\text{Site}_y} \end{array} \right.$$

Remarque : Agent $_i$ appartient au Site $_y$.

2. Cas de traitement local (L) : Agent $_j$ appartient au Site $_x$ et offre le Service $_{ja}$

$$\text{Agent}_i \text{ demande localement Agent}_j \left\{ \begin{array}{l} \text{Coût-exécution : L.CE}_{\text{Agent}_j, \text{Service}_{ja}}^{\text{Site}_x} \\ \text{Coût-accueil : 0} \end{array} \right.$$

Remarque: un traitement local peut être vu comme un déplacement avec un coût d'accueil égal à zéro. Cas particulier du traitement local : $Agent_i = Agent_j$.

3. Cas de traitement à distance (Δ): $Agent_j$ appartient au $Site_y$ et offre le $Service_{ja}$
à .CE : coût d'exécution pour un traitement à distance;

$$Agent_i \text{ demande à distance } Agent_j \left\{ \begin{array}{l} \text{Coût-exécution : } \Delta.CE_{Site_y, Agent_j, Service_{ja}} \\ \text{Coût-accueil : } 0 \end{array} \right.$$

4. Cas d'invitation (I) : $Agent_j$ appartient au $Site_y$ et offre le $Service_{ja}$

I.CA: coût d'accueil pour une invitation;

I.CE: coût d'exécution pour une invitation;

$$Agent_i \text{ invite } Agent_j \left\{ \begin{array}{l} \text{Coût-exécution : } I.CE_{Site_x, Agent_j, Service_{ja}} \\ \text{Coût-accueil : } I.CA_{Site_x, Agent_j, Site_y} \end{array} \right.$$

Remarque: nous supposons que l' $Agent_j$ envoie un clone au $Site_x$.

Pour le reste de la section, le cas du traitement local n'est plus pris en compte de manière individuelle. Nous considérons qu'il fait partie du cas de déplacement d'un agent. Dans l'étape 3, nous identifions chaque tâche par le service dont elle a besoin pour être exécutée. Ce service peut être offert par divers agents. Pour qu'une tâche puisse être accomplie, un agent a la possibilité de se déplacer, de demander le traitement à distance ou d'inviter un agent. Ainsi, nous spécifions le fait qu'une tâche ait besoin d'un service comme suit :

$$Tâche_n \text{ besoin } Service_{ja} \left\{ \begin{array}{l} \text{déplacement : } (D.CE, D.CA) \\ \text{à-distances : } (\Delta.CE, 0) \\ \text{Invitation : } (I.CE, I.CA) \end{array} \right.$$

Par la suite, nous positionnons les diverses tâches dans le système de coordonnées de la Figure 4 : un cercle représente graphiquement un cas de déplacement, un carré représente graphiquement un cas d'invitation et un diamant représente graphiquement un cas de traitement à distance.

Comment positionner les tâches ?

Le principe de positionnement des tâches consiste à placer une tâche en tenant compte de l'emplacement de la tâche qui la précède dans l'ordre d'exécution, sauf pour la première tâche du plan (Fig. 5).

- Si m $Services_{ja}$ identiques de divers $Agent_j$ sont offerts pour satisfaire une $Tâche_n$, alors cette $Tâche_n$ aura au plus $3 \times m$ positions dans le système de coordonnées ("3" pour déplacement, invitation et traitement à distance; "au plus" à cause des valeurs nulles).
- La $Tâche_n$ qui correspond à la première tâche du plan est toujours positionnée par rapport au point 0,0.
- Une $Tâche_n$ qui correspond à la n ième tâche du plan est toujours positionnée par rapport à la $Tâche_{n-1}$.

Dans ce qui suit, nous expliquons comment les coordonnées d'une $Tâche_n$ sont obtenues et positionnées par rapport aux coordonnées de la $Tâche_{n-1}$. Les symboles utilisés sont : D pour déplacement ; Δ pour à-distance; I pour invitation.

1. Cas de déplacement pour la $Tâche_n$

Position $Tâche_n$: cas de déplacement pour l'exécution de la $Tâche_{n-1}$.

$$\text{Pos.D} (T\grave{a}che_n, Service_{ja}) \begin{cases} x = D.CE_{Agent_j, Service_{ja}}^{Site_y} / D.Coordonn\acute{e}es(T\grave{a}che_{n-1}) \\ y = D.CA_{Agent_i, Site_x}^{Site_y} / D.Coordonn\acute{e}es(T\grave{a}che_{n-1}) \end{cases}$$

Position $T\grave{a}che_n$: cas de traitement \`a distance pour l'ex\`ecution de la $T\grave{a}che_{n-1}$.

$$\text{Pos.D} (T\grave{a}che_n, Service_{ja}) \begin{cases} x = D.CE_{Agent_j, Service_{ja}}^{Site_y} / \grave{A}.Coordonn\acute{e}es(T\grave{a}che_{n-1}) \\ y = D.CA_{Agent_i, Site_x}^{Site_y} / \grave{A}.Coordonn\acute{e}es(T\grave{a}che_{n-1}) \end{cases}$$

Position $T\grave{a}che_n$: cas d'invitation pour l'ex\`ecution de la $T\grave{a}che_{n-1}$.

$$\text{Pos.D} (T\grave{a}che_n, Service_{ja}) \begin{cases} x = D.CE_{Agent_j, Service_{ja}}^{Site_y} / I.Coordonn\acute{e}es(T\grave{a}che_{n-1}) \\ y = D.CA_{Agent_i, Site_x}^{Site_y} / I.Coordonn\acute{e}es(T\grave{a}che_{n-1}) \end{cases}$$

2. Cas de traitement \`a distance pour la $T\grave{a}che_n$

Position $T\grave{a}che_n$: cas de d\`eplacement pour l'ex\`ecution de la $T\grave{a}che_{n-1}$

$$\text{Pos.\grave{A}} (T\grave{a}che_n, Service_{ja}) \begin{cases} x = \grave{A}.CE_{Agent_j, Service_{ja}}^{Site_y} / D.Coordonn\acute{e}es(T\grave{a}che_{n-1}) \\ y = 0 \end{cases}$$

Position $T\grave{a}che_n$: cas de traitement \`a distance pour l'ex\`ecution de la $T\grave{a}che_{n-1}$

$$\text{Pos.\grave{A}} (T\grave{a}che_n, Service_{ja}) \begin{cases} x = \grave{A}.CE_{Agent_j, Service_{ja}}^{Site_y} / \grave{A}.Coordonn\acute{e}es(T\grave{a}che_{n-1}) \\ y = 0 \end{cases}$$

Position $T\grave{a}che_n$: cas d'invitation pour l'ex\`ecution de la $T\grave{a}che_{n-1}$

$$\text{Pos.\grave{A}} (T\grave{a}che_n, Service_{ja}) \begin{cases} x = \grave{A}.CE_{Agent_j, Service_{ja}}^{Site_y} / I.Coordonn\acute{e}es(T\grave{a}che_{n-1}) \\ y = 0 \end{cases}$$

3. Cas d'invitation pour la $T\grave{a}che_n$

Position $T\grave{a}che_n$: cas de d\`eplacement pour l'ex\`ecution de la $T\grave{a}che_{n-1}$.

$$\text{Pos.I} (T\grave{a}che_n, Service_{ja}) \begin{cases} x = I.CE_{Agent_j, Service_{ja}}^{Site_x} / D.Coordonn\acute{e}es(T\grave{a}che_{n-1}) \\ y = I.CA_{Agent_i, Site_y}^{Site_x} / D.Coordonn\acute{e}es(T\grave{a}che_{n-1}) \end{cases}$$

Position $T\grave{a}che_n$: cas de traitement \`a distance pour l'ex\`ecution de la $T\grave{a}che_{n-1}$

$$\text{Pos.I} (T\grave{a}che_n, Service_{ja}) \begin{cases} x = I.CE_{Agent_j, Service_{ja}}^{Site_x} / \grave{A}.Coordonn\acute{e}es(T\grave{a}che_{n-1}) \\ y = I.CA_{Agent_i, Site_y}^{Site_x} / \grave{A}.Coordonn\acute{e}es(T\grave{a}che_{n-1}) \end{cases}$$

Position $T\grave{a}che_n$: cas d'invitation pour l'ex\`ecution de la $T\grave{a}che_{n-1}$

$$\text{Pos.I} (T\grave{a}che_n, Service_{ja}) \begin{cases} x = I.CE_{Agent_j, Service_{ja}}^{Site_x} / I.Coordonn\acute{e}es(T\grave{a}che_{n-1}) \\ y = I.CA_{Agent_i, Site_y}^{Site_x} / I.Coordonn\acute{e}es(T\grave{a}che_{n-1}) \end{cases}$$

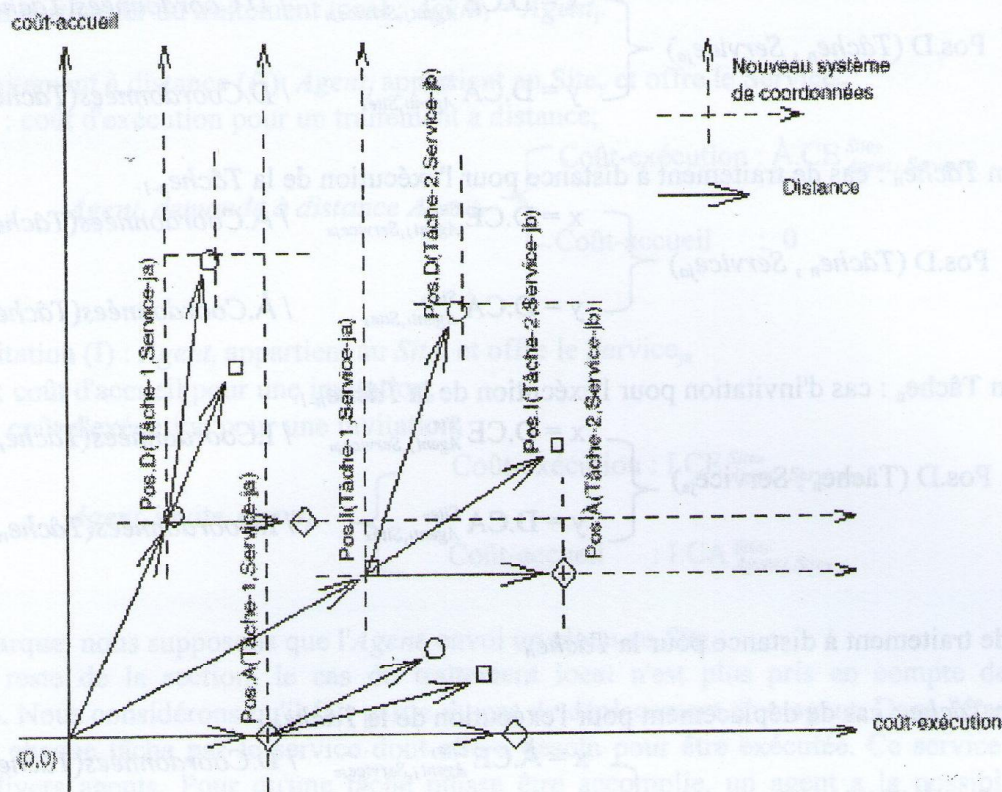


Figure 5. Positionnement des tâches et des services selon les nouveaux système de coordonnées

Dans l'étape 4 (faite en parallèle avec l'étape 3), nous construisons un réseau de tâches-services avec les caractéristiques suivantes:

- Le nœud initial identifie la position 0, 0 du système de coordonnées de la Figure 5.
- Un nœud représente une tâche avec le service dont elle a besoin.
- Un arc relie une tâche à une autre tâche selon l'ordre d'exécution des tâches dans le plan. De chaque nœud doit partir $3 \times m$ arcs (m : nombre d'agents offrant le même service) pour rejoindre les $3 \times m$ nœuds qui identifient les diverses situations de réalisation de la tâche suivante.

Comment évaluer les arcs ?

Pour évaluer le poids des arcs, nous calculons la distance métrique qui sépare une tâche d'une autre tâche et nous rajoutons la valeur de l'arc précédant à la valeur de cette distance. Ainsi, le poids des arcs est une succession de cumuls des divers poids des arcs qui précèdent l'arc faisant l'objet de calcul. Le calcul des distances est en fonction des nouveaux systèmes de coordonnées construits au fur et à mesure du positionnement des tâches (Fig. 5). Le réseau de tâches-services obtenu est représenté par la Figure 6.

Dans l'étape 5 et à l'aide du réseau de tâches-services, nous obtenons l'itinéraire de l'agent-client et en même temps optimisons cet itinéraire. Pour cela, nous considérons la plus petite valeur qui permet de rejoindre la dernière tâche du plan de réalisation. L'itinéraire résultant peut avoir la forme suivante (Fig.7) : une invitation pour exécuter la *Tâche*₁, un traitement à distance pour exécuter la *Tâche*₂ et un

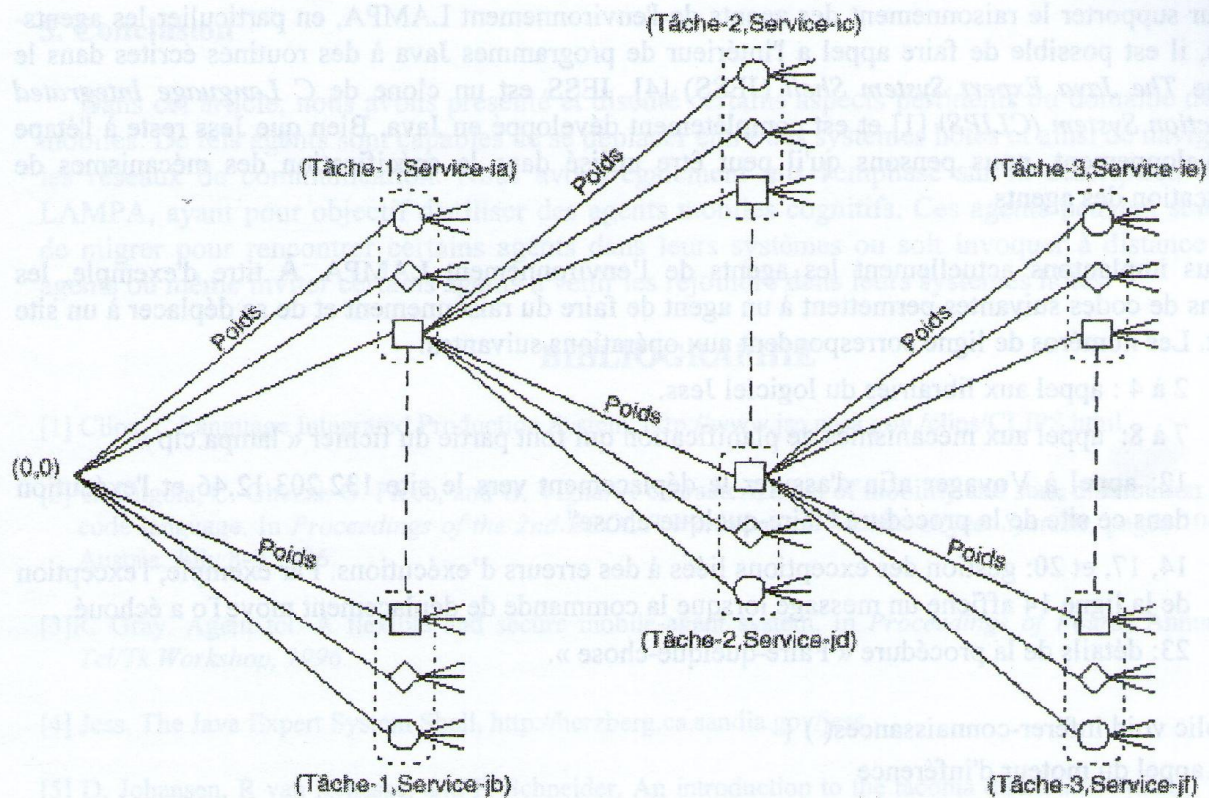


Figure 6. Elaboration du réseau de tâches-services

déplacement pour exécuter la *Tâche*₂. L'avantage de la représentation par réseau tâches-services est qu'un agent-client peut ne pas utiliser ses propres services, si les services d'autres agents sont plus avantageux que les siens.

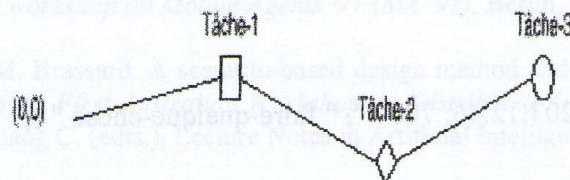


Figure 7. Exemple d'un itinéraire pour un agent-client de l'environnement LAMPA

4. Implantation de l'environnement LAMPA

Trois questions doivent trouver réponses dans l'implantation de l'environnement LAMPA : Comment implanter les agents de l'environnement LAMPA? Comment permettre aux agents de l'environnement LAMPA de se déplacer? Et comment implanter les mécanismes de planification des agents de l'environnement LAMPA?

Actuellement, plusieurs langages de programmation d'agents mobiles existent, tels que Java et Tcl [2]. Malgré sa caractéristique de portabilité et son leadership comme langage de programmation des réseaux, Java est vu comme le langage qui assure la plus "faible" mobilité dans un environnement d'agent mobiles. Java supporte le déplacement des agents qu'une seule fois [12]. Par contre, nous pensons que l'utilisation de l'Object Request Broker (ORB) Voyager [15] permettra d'éliminer cette faiblesse. En effet, Voyager permet de rester en accord avec les tendances actuelles de standardisation entreprises par l'OMG au niveau des systèmes d'objets distribués et offre des services de support pour les agents mobiles.

Pour supporter le raisonnement des agents de l'environnement LAMPA, en particulier les agents-clients, il est possible de faire appel à l'intérieur de programmes Java à des routines écrites dans le langage *The Java Expert System Shell* (JESS) [4]. JESS est un clone de *C Language Integrated Production System* (CLIPS) [1] et est complètement développé en Java. Bien que Jess reste à l'étape de développement, nous pensons qu'il peut être utilisé dans la spécification des mécanismes de planification des agents.

Nous implantons actuellement les agents de l'environnement LAMPA. À titre d'exemple, les portions de codes suivantes permettent à un agent de faire du raisonnement et de se déplacer à un site distant. Les numéros de ligne correspondent aux opérations suivantes:

- 2 à 4 : appel aux bibliothèques du logiciel Jess.
- 7 à 8: appel aux mécanismes de planification qui font partie du fichier « lampa.clp ».
- 12: appel à Voyager afin d'assurer le déplacement vers le site 132.203.12.46 et l'exécution dans ce site de la procédure "faire-quelque-chose".
- 14, 17, et 20: gestion des exceptions liées à des erreurs d'exécutions. Par exemple, l'exception de la ligne 14 affiche un message lorsque la commande de déplacement moveTo a échoué.
- 23: détails de la procédure « Faire-quelque-chose ».

```

0 : public void inférer-connaissances( ) {
1 :   // appel du moteur d'inférence
2 :   NullDisplay nd = new NullDisplay( );
3 :   Rete rete = new Rete(nd);
4 :   rete.AddUserpackage(new Multifunctions( ));
5 :   try {
6 :     // appel du fichier lampa.clp
7 :     FileInputStream fis = new FileInputStream("lampa.clp");
8 :     Jesp j = new Jesp(fis, rete);
9 :     try {
10 :       j.parse(false);
11 :       try {
12 :         moveTo ( " 132.203.12.46: 7000 ", " faire-quelque-chose " );
13 :       }
14 :       catch(VoyagerException exception) {
15 :         System.out.println(exception); }
16 :     }
17 :     catch(ReteException exception) {
18 :       exception.printStackTrace(nd, stderr( )); }
19 :   }
20 :   catch(IOException exception) {
21 :     System.out.println(exception); }
22 : }
23 : public faire-quelque-chose( ) {
24 :   System.out.println(var_agt + " arrive au " + var_site_des + " en provenance du " + var_site_ori);
25 :   try { Thread.sleep( 3000 ); }
26 :   catch( InterruptedException exception ) {
27 :     System.out.println(exception); }
28 : }

```

5. Conclusion

Dans cet article, nous avons présenté et discuté certains aspects pertinents du domaine des agents mobiles. De tels agents sont capables de se déplacer entre des systèmes hôtes et ainsi de naviguer dans les réseaux de communication. Nous avons également mis l'emphase sur la description du projet LAMPA, ayant pour objectif d'utiliser des agents mobiles cognitifs. Ces agents peuvent soit décider de migrer pour rencontrer certains agents dans leurs systèmes ou soit invoquer à distance certains agents ou même inviter certains agents à venir les rejoindre dans leurs systèmes hôtes.

BIBLIOGRAPHIE

- [1] Clips. C Language Integrated Production System, <http://www.jsc.nasa.gov/clips/CLIPS.html>.
- [2] G. Cugola, C. Ghezzi, G. Picco, and G. Vigna. A characterization of mobility and state distribution in mobile code language. In *Proceedings of the 2nd ECOOP Workshop on Mobile Object Systems*, pages 10-19, Linz, Austria, July 8-9 1996.
- [3] R. Gray. Agent tcl: A flexible and secure mobile-agent system. In *Proceedings of Fourth Annual Usenix Tcl/Tk Workshop*, 1996.
- [4] Jess. The Java Expert System Shell, <http://herzberg.ca.sandia.gov/jess>.
- [5] D. Johansen, R van Renesse, and F. Schneider. An introduction to the tacoma distributed system. Technical report, 95-23 Computer Science, University of Tromsø, Norway, 1995.
- [6] D. Lange and M. Oshima. Programming mobile agents in java - with the java aglet api. Technical report, IBM research, Tokyo, 1997.
- [7] A. Lingnau, O. Drobnik, and P. Domel. An http-based infrastructure for mobile agents. Technical report, Frankfurt University, Germany, 1995.
- [8] Mitsubishi Electric ITA. Concordia: An infrastructure for collaborating mobile agents. In *Proceedings of First International workshop on Mobile Agents '97 (MA '97)*, Berlin, Germany, April 7-8 1997.
- [9] B. Moulin and M. Brassard. A scenario-based design method and an environment for the development of multiagent systems. In *First Australian Workshop on Distributed Artificial Intelligence* n. 1087, pages 216-231. D. Lukose, Zhang C. (eds.), Lecture Notes in Artificial Intelligence, Springer-Verlag, 1996.
- [10] H.-S. Nwana. Software agents: An overview. *Knowledge Engineering Review*, 11(3):1-40, September 1996.
- [11] A. Ohsuga, Y. Nagai, Y. Irie, H. Masanori, and S. Hondien. Plangent: An approach to making mobile agents intelligent. *IEEE Internet Computing*, 1(4):50-57, July/August 1997.
- [12] J. Ordille. When agents roam, who can you trust? In *Proc. First Conf. On Emerging Technologies and Applications in Communications (etaCOM)*, May 1996.
- [13] M. StraBer, J. Baumann, and F. Hohl. Mole - a java mobile agent system. In *Proceedings of the 2nd ECOOP, Workshop on Mobile Object Systems*, pages 28-35, Linz, Austria, July 8-9 1996.
- [14] F. Van Aeken, Y. Demazeau, and J. Kozlak. Migration, mobility and the palindrome problem. In *Proceedings of the International Workshop Distributed Artificial Intelligence and Multi-Agent Systems DAIMAS '97*, pages 23-31, St. Petersburg, Russia, June 15-18 1997.
- [15] Voyager. ObjectSpace, <http://www.objectspace.com>.
- [16] J. White. Mobile agents white paper. Technical report, General Magic, 1996. <http://www.genmagic.com/agents/Whitepaper~whitepaper.html>.