Guest, système d'agents récursifs, mobiles et multiplates-formes.

Comment enrichir et rendre interopérables diverses plates-formes d'agents mobiles

Thang Viet Pham - Laurent Magnin - Arnaud Dury

CRIM
550 rue Sherbrooke Ouest
Bureau 100, Montréal, Québec
Canada H3A 1B9
{tvpham,lmagnin,adury}@crim.ca

ABSTRACT. *We present Guest, a system for mobile, multi-platform and recursive agents. Guest supports heterogeneous platforms such as Aglets, Jade, Grasshopper and Voyager are supported through specific interfaces. We also show some Guest extensions we developed for these platforms, including dynamic code loading, recursive agents, interaction programming, etc. Theses features are demonstrated through a mobile agents version of the classical prey and predators example.*

RÉSUMÉ. *Le système Guest permet de définir et de déployer des systèmes multiagents récursifs, mobiles et multiplates-formes. Pour cela, des interfaces spécifiques à différentes plates-formes multiagents (Aglets, Jade, Grasshopper et Voyager) ont été implémentées. De plus, des fonctionnalités ne se trouvant pas sur ces plates-formes ont été ajoutées : chargement dynamique de code, agents récursifs, programmation d'interaction, etc. Ces points sont exposés au travers d'une démonstration d'agents mobiles Guest proie-prédateurs.*

KEYWORDS: *Interoperability, mobile agents, recursive agents*

MOTS-CLÉS : *Interopérabilité, agents mobiles, agents récursifs*

1. Introduction

Jusqu'à présent, l'exécution d'un agent était liée à la plate-forme pour laquelle il avait été créé. Ainsi un agent «Aglets» [agl00] ne pouvait fonctionner que sur un serveur basé sur la plate-forme Aglets, et non pas sur un serveur «Grasshopper» [gra01]. Tant que l'on envisage uniquement des applications multiagents propriétaires et fermées, cette limitation n'est pas trop contraignante. Cependant, si l'on songe à des applications mettant en œuvre des agents de provenances diverses devant pouvoir être accueillis sur des serveurs mis en place par des tiers, cette limitation devient rédhibitoire. Et c'est notamment le cas pour Internet, lequel est un environnement fortement hétérogène [MAG 99].

2. La plate-forme Guest

La réalisation de plates-formes d'agents interopérables est un domaine qui a déjà été abordé dans la littérature sous deux aspects différents. D'une part, les travaux de [kqm01], [fip01] ou [OMG 01] se préoccupent de faire communiquer des plates-formes hétérogènes, mais n'abordent pas la migration effective d'agents entre ces plates-formes. D'autre part, les travaux de [TJU 99] répondent au problème de la migration hétérogène, en réalisant une rétro-ingénierie du code de l'agent se trouvant sur la plate-forme source afin de pouvoir le modéliser en vue de le transformer en un code compilable par la plate-forme de destination. Cette solution complexe présente en pratique de nombreux problèmes. Notre approche, de son côté, passe par une interface appelée *Guest* développée au-dessus de plusieurs systèmes hétérogènes existants (Aglets [agl00], Grasshopper [gra01], Jade [jad01] et Voyager [voy01]). *Cette interface permet non seulement aux agents Guest de s'exécuter sur ces plates-formes hétérogènes, mais également de communiquer et de migrer de l'une à l'autre*. *Guest*

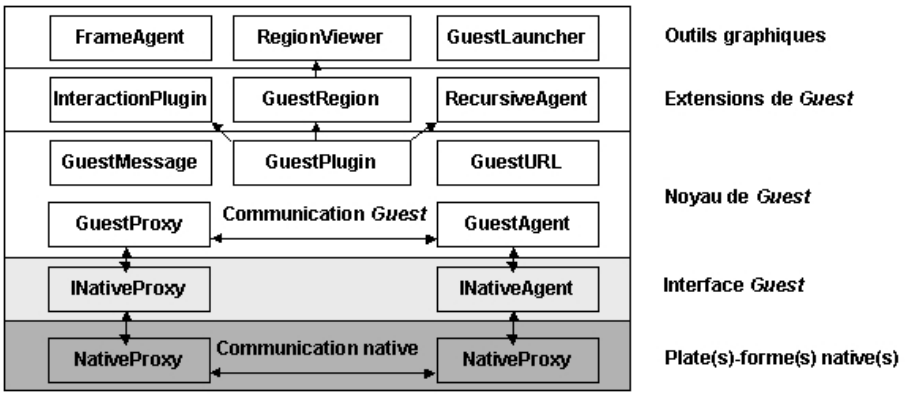

Figure 1. *L'architecture de Guest*

ne se limite pas à implanter le plus petit commun dénominateur des plates-formes qu'il interface, il étend également les fonctionnalités de ces dernières à travers un mécanisme de *plugins*. Une des extensions disponibles sous forme de *plugins* permet par exemple de fournir à *Guest* (et donc à toutes les plates-formes interfacées par celui-ci) un modèle identique d'agents récursifs.

3. Prédateurs et proie

Nous proposons ici un scénario permettant de présenter les caractéristiques de la plate-forme *Guest*. Cette démonstration, tirée de [gra01], consiste en un jeu dans lequel plusieurs prédateurs doivent encercler une proie. La proie est capturée lorsque au moins n prédateurs se retrouvent sur le même serveur qu'elle.

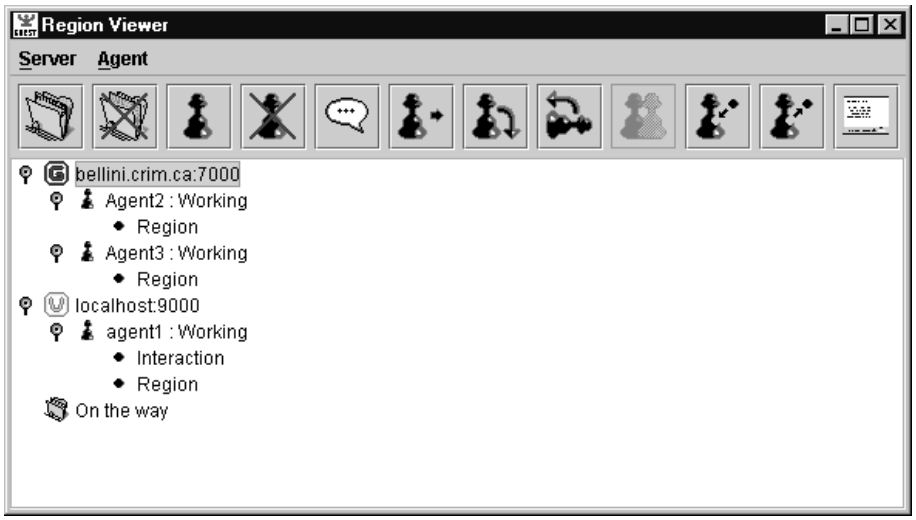

Figure 2. *L'interface graphique de Guest*

Cet exemple permet d'exposer les outils *de lancement, de visualisation et de contrôle des agents* (voir figure 2), indépendamment des plates-formes sur lesquelles ils s'exécutent. Notre application peut s'exécuter *simultanément sur diverses plates-formes multiagents hétérogènes* : non seulement les prédateurs peuvent communiquer les uns avec les autres, mais ils peuvent également migrer entre ces serveurs de façon transparente (tout en continuant à recevoir leurs messages sans que leurs partenaires ne soient explicitement prévenus de leurs déplacements). Afin de pouvoir *constituer automatiquement des groupes d'intérêts*[1], les agents peuvent *dynamiquement intégrer du code* sous forme de *plugins ad hoc* permettant de gérer, dans ce cas, des groupes de

1. Dans notre exemple, celui des prédateurs affamés.

discussion. Enfin, il est possible de considérer non pas simplement des agents ayant différents rôles, mais bien des organisations d'agents possédant une structure propre. *Guest* permet très simplement de réifier ces organisations sous forme d'*agents récursifs*, en spécifiant les agents les composant et l'organisation des communications entre ces agents. Pour revenir à notre exemple, les prédateurs adoptent un modèle d'agent récursif basé sur leur intégration sous le contrôle d'un seul agent natif, lequel lorsqu'il migre entraîne avec lui l'ensemble de ses sous-agents. Ils restent ainsi groupés afin d'être en nombre suffisant afin de pouvoir capturer une éventuelle proie.

4. Conclusion

La possibilité pour des agents de pouvoir communiquer, s'exécuter et migrer indépendament du type des plates-formes les acceuillant est un point essentiel au développement d'applications multiagents dans un réseau hétérogène et ouvert tel qu'Internet. L'approche proposée ici par *Guest*, consistant à fournir une interface entre des agents indépendants et les plates-formes sous-jacentes, permet de façon élégante et relativement facile[2] de résoudre ce problème. Ceci est démontré par une application de type proie-prédateurs mettant en œuvre des agents mobiles se déplaçant entre serveurs hétérogènes. Nous avons de plus intégré à *Guest* des fonctionnalités telles que le chargement dynamique de code et la récursivité des agents.

5. References

[agl00] "Aglets Workbench by IBM Tokyo RL", http://www.trl.ibm.co.jp/aglets/, 2000.

[fip01] "Foundation for Intelligent Physical Agents", http://www.fipa.org/, 2001.

[gra01] "Grasshopper by IKV++", http://www.ikv.de/products/grasshopper/index.html, 2001.

[jad01] "Jade", http://jade.cselt.it, 2001.

[kqm01] http://www.cs.umbc.edu/kqml/, 2001.

[MAG 99] MAGNIN L., "Internet, environnement complexe pour agents situés", *Conférence sur l'Intelligence Artificielle Située*, Paris, octobre 1999, p. 213-221.

[OMG 01] OMG, "MASIF: Mobile Agent System Interoperability Facility", http://www.fokus.gmd.de/research/cc/ima/masif/index.html, 2001.

[TJU 99] TJUNG D., TSUKAMOTO M., NISHIO S., "A converter Approach for Mobile Agent System Integration: A Case of Aglet to Voyager", *First International Workshop on Mobile Agents for Telecommunication Applications*, Ottawa, Canada, octobre 1999, p. 179-195.

[voy01] "Voyager by ObjectSpace", http://www.objectspace.com/voyager/, 2001.

2. Dans le cas où l'on se limite à des plates-formes utilisant le même langage de programmation; Java dans le cas de *Guest*.